PCI Bus CPD ボードシリーズ

HPCI-CPD5212M

ユーザーズマニュアル

〈ソフトウェア編　Windows 版〉

NC ボード

多機能･高速　円弧･直線補間･位置決め

株式会社ハイバーテック

http://www.hivertec.co.jp/

この説明書は次のボードに適応しています.

HPCI-CPD5212M

株式会社 ハイバーテック
東京都江東区新大橋 1-8-11
三井生命新大橋ビル
TEL 03-3846-3801
FAX 03-3846-3773
sales@hivertec.co.jp

第 5.01 版 2016 年 10 月 7 日発行

本製品をご使用される前に「注意事項」を必ずご一読の上ご利用をお願い致します.

目　　次

■ 注意事項 1
■ 保証範囲 1
■ 免責事項 1
■ 安全にお使い頂くために 1
■ 対象ユーザー 2
■ 添付ソフトウェア適合 OS 2
■ 動かしてみるプログラム 2
■ サンプルプログラム 3
■ ユーザープログラム 3
■ 試運転・調整 3
■ CPD シリーズのマニュアル構成 4
1. はじめに 5
1.1 ソフトウェアの構成 5
1.2 添付ファイル名 5
1.3 関数名 5
1.4 関数の戻り値 6
1.5 アプリケーション作成準備 7
1.5.1 Microsoft Visual C++ (2008 以上) 7
1.5.2 Microsoft Visual C#(2008 以上) 7
1.5.3 Microsoft Visual Basic(2008 以上) 7
1.5.4 Microsoft Visual Basic 6.0 7
1.5.5 ボードを複数枚使用する場合 8
1.5.6 ボードアクセス方法 8
1.6 アプリケーション作成上の注意 9
2. ライブラリ関数 11
2.1 ライブラリ関数序論 11
2.1.1 軸の指定 11
2.1.2 ライブラリ関数一覧 12
2.2 デバイス関係 13
2.2.1 hcp52c_GetDevInfo()　ボード枚数取得，デバイス情報の取得 13
2.2.2 hcp52c_DevOpen()　デバイスのオープン，レジスタとオプションポートの初期化 14
2.2.3 hcp52c_DevClose()　デバイスのクローズ 15
2.3 初期設定 16
2.3.1 hcp52c_SetOrgMode()　原点復帰モードの設定 16
2.3.2 hcp52c_SetEls()　ELS の設定 17
2.3.3 hcp52c_SetOls()　OLS の設定 17
2.3.4 hcp52c_ SetSvAlm ()　SVALM の設定 18
2.3.5 hcp52c_SetEz()　エンコーダ Z 相の設定 18
2.3.6 hcp52c_SetInpos ()　INPOS の設定 19
2.3.7 hcp52c_SetSvCtrCl()　偏差カウンタクリア出力の設定 19
2.3.8 hcp52c_SetSls()　ソフトリミットの設定 20
2.3.9 hcp52c_SetCmdPulse()　指令パルスの出力形式の設定 21
2.3.10 hcp52c_SetAccProfile()　加減速形式の設定 21

2.3.11 hcp52c_SetAutoDec() 減速開始点計算方式の自動計算/手動計算切り替え 22
2.4 状態読み出し 23
2.4.1 hcp52c_ReadMainSts() メインステータスの読出し 23
2.4.2 hcp52c_ReadErrorSts() エラーステータスの読出し 24
2.4.3 hcp52c_ReadEventSts() イベントステータスの読出し 25
2.4.4 hcp52c_ReadSubSts() サブステータスの読出し 26
2.4.5 hcp52c_ReadExSts() 拡張ステータスの読出し 27
2.4.6 hcp52c_ReadSpd() 指令速度の読出 28
2.4.7 hcp52c_ReadCtr() カウンタの読出 28
2.5 動作設定 29
2.5.1 hcp52c_SetFLSpd() ベース速度の設定 29
2.5.2 hcp52c_SetAuxSpd() 補助速度の設定 29
2.5.3 hcp52c_SetAccRate() 加速レートの設定 30
2.5.4 hcp52c_SetDecRate() 減速レートの設定 31
2.5.5 hcp52c_SetMult() 速度倍率レジスタ値の設定 32
2.5.6 hcp52c_SetEventMask() イベントマスクの設定 33
2.5.7 hcp52c_SetDecPoint() 減速開始点の設定 34
2.6 運用設定 35
2.6.1 hcp52c_WritOpeMode() 動作モードの設定 35
2.6.2 hcp52c_WritFHSpd() 動作速度の設定 36
2.6.3 hcp52c_WritPos() 位置決め移動量の設定 36
2.6.4 hcp52c_WritLine() 直線補間の移動量の設定 37
2.6.5 hcp52c_WritCircl() 円弧補間の移動量の設定 37
2.6.6 hcp52c_WritCtr() カウンタプリセット 38
2.7 動作制御指令 38
2.7.1 hcp52c_DecStop() 減速停止 38
2.7.2 hcp52c_QuickStop() 即停止 38
2.7.3 hcp52c_SyDecStop() 複数軸同時減速停止 39
2.7.4 hcp52c_SyQuickStop() 複数軸同時即停止 39
2.7.5 hcp52c_EmgStop() 非常停止 39
2.7.6 hcp52c_AccStart() 加速スタート 39
2.7.7 hcp52c_ CnstStartFH () FH 定速スタート 40
2.7.8 hcp52c_ CnstStartFL () FL 定速スタート 40
2.7.9 hcp52c_ CnstStartByDec () FH定速スタート後減速停止 40
2.7.10 hcp52c_SvOn() SVON オン 41
2.7.11 hcp52c_SvOff() SVON オフ 41
2.7.12 hcp52c_SvResetOn() SVRST オン 41
2.7.13 hcp52c_SvResetOff() SVRST オフ 42
2.7.14 hcp52c_PMOn() パルスモータ励磁オン 42
2.7.15 hcp52c_PMOff() パルスモータ励磁オフ 42
2.8 加減速レートの計算 43
2.8.1 hcp52c_CalAccRate() 加減速レートの計算 43
3. ドライバ関数 44
3.1 関数の種類 44
3.2 関数の詳細 44
3.2.1 cp52c_GetDeviceCount() ボード枚数の取得 44
3.2.2 cp52c_GetDeviceInfo() デバイス情報の取得 45
3.2.3 cp52c_OpenDevice() デバイスのオープン 46
3.2.4 cp52c_CloseDevice() デバイスのクローズ 46
3.2.5 cp52c_rMstsW() メインステータスの読出し 47
3.2.6 cp52c_rSstsW() サブステータスの読出し 48
3.2.7 cp52c_wCmdW() 制御コマンド書込み 49

3.2.8 cp52c_rReg() レジスタ読出し 51
3.2.9 cp52c_wReg() レジスタ書込み 51
3.2.10 レジスタ制御コマンド・内容 52
3.2.11 cp52c_rPortB() オプションポートバイト読出し 71
3.2.12 cp52c_wPortB() オプションポートバイト書込み 71
3.2.13 cp52c_rPortW() オプションポートワード(2 バイト)読出し 72
3.2.14 cp52c_wPortW() オプションポートワード(2 バイト)書込み 72
3.2.15 cp52c_rBufDW() 入出力バッファ読出し 73
3.2.16 cp52c_wBufDW() 入出力バッファ書込み 74
4. サンプルプログラム 75
4.1 Windows 版サンプルプログラム 75
4.1.1 サンプルプログラムの実行 75
4.1.2 サンプルプログラムの操作 76
5. ポート資料 82
5.1 PCI コンフィグレーションレジスタ 82
5.2 ポート及びレジスタアクセス 82
5.2.1 CMD,BUFx 書込み，読出し方法 82
5.3 ポート表 85
5.4 オプションポート詳細 86
5.4.1 各軸 ELS 極性の設定と読込(ELPOL) 86
5.4.2 コンパレータ 4 比較条件成立で同時スタート信号(STA)出力設定と読込(C4STA) 86
5.4.3 コンパレータ 5 比較条件成立で同時ストップ信号(STP)出力設定と読込(C5STP) 87
5.4.4 X1~U1 コンパレータ 3～5 比較結果外部出力の選択設定と読込(COTSEL1) 87
5.4.5 X2~U3 コンパレータ 3～5 比較結果外部出力の選択設定と読込(COTSEL2) 88
5.4.6 オプションポート設定初期化(OPTRST) 88
5.4.7 ボード割込マスクの設定と読込(BDIEBL) 88
5.4.8 ボード割込状態読込(BDINTS) 89
5.4.9 ボード ID(BID) 89
5.4.10 エンコーダフィルタ設定(ENFIL) 89
5.4.11 ボードコード読み出し 90
6. 更新履歴 91

図　表　目　次

図 1.1-1 ソフトウェアの構成 5
表 1.2-1 添付ファイル名 5
表 1.3-1 関数名 5
表 1.4-1 関数の戻り値 6
図 1.5-1 ボードを複数枚使用 8
表 1.6-1 排他処理が必要な関数 10
表 2.1-1 ライブラリ関数一覧 12
表 3.1-1 ドライバ関数一覧 44
図 4.1-1 サンプルプログラムのエラーメッセージ 75
図 4.1-2 サンプルプログラムの動作選択画面 76
図 5.2-1 CMD ポートの形式 83
図 5.2-2 レジスタ書込み，読出しの CMD,BUF0,BUF1 の形式 83
表 5.3-1 HPCI-CPD5212M ポート表 86
図 5.4-1 X1~U1 コンパレーター致出力 87
図 5.4-2 X2~U3 コンパレーター致出力(オプション) 88
図 5.4-3 割り込み要因 89

■ 注意事項

■ 保証範囲

1. 本製品の保証期間は，お買い上げ頂いた日より 3 年間です．保証期間中に弊社の判断により欠陥が判明した場合には，本製品を弊社に引き取り，修理または交換を行います．

2. 保証期間内外に関わらず，弊社製品の使用，供給(納期)または故障に起因する，お客様及び第三者が被った，直接，間接，二次的な損害あるいは，遺失利益の損害に付いて，弊社は本製品の販売価格以上の責任を負わないものとしますので，予めご了承ください．

■ 免責事項

1. 本書に記載された内容に沿わない，製品の取付，接続，設定，運用により生じた損害に対しましては，一切の責任を負いかねますので，予めご了承ください．

2. 本製品は，一般電子機器用(工作機械・計測機器・FA/OA 機器・通信機器等)に製造された半導体製品を使用していますので，その誤作動や故障が直接，生命を脅かしたり，身体・財産等に危害を及ぼしたりする恐れのある装置(医療機器・交通機器・燃焼機器・安全装置等)に適用できるような設計，意図，または，承認，保証もされていません．
   ゆえに本製品の安全性，品質および性能に関しては，本書(またはカタログ)に記載してあること以外は明示的にも黙示的にも一切保証するものではありませんので，予めご了承ください．

3. 保証期間内外に関わらず，お客様が行った弊社の承認しない製品の改造または，修理が原因で生じた損害に対しましては，一切の責任を負いかねますので，予めご了承ください．

4. 本書に記載された内容について，弊社もしくは，第三者の特許権，著作権，商標権，その他の知的所有権の権利に対する保証または実施権の許諾を行うものではありません．
   また本書に記載された情報を使用したことにより第三者の知的所有権等の権利に関わる問題が生じた場合，弊社は，その責任を負いかねますので，予めご了承ください．

■ 安全にお使い頂くために

この度は，弊社 NC ボードシリーズをご採用頂きまして，誠に有り難う御座います．本書は，本製品をご使用して頂く場合の取扱い，留意点に付いて記入してありますので，必ずご一読の上ご利用をお願い致します．
尚，本書は，本書が添付されたNCボード常設箇所付近の分かりやすい場所に常時保管し，必要に応じて適宜参照・確認頂きますよう，お願い致します．

安全上の注意

本製品のご使用前に，必ずこのユーザーズマニュアル及び付属書類を全て熟読し，内容を理解してから正しくご使用下さい．本製品の知識，安全の情報及び注意事項の全てに付いて習熟してからご使用下さい．
本ユーザーズマニュアルでは，安全注意事項のランクを「警告」，「注意」として区分してあります．

⚠ **警 告**	この表示を無視して，誤った取扱いをすると，人が死亡または重傷を負う可能性が想定される内容を示しています．

⚠ **注 意**	この表示を無視して，誤った取扱いをすると，人が傷害を負う可能性または物的損害が想定される内容を示しています．

■ **対象ユーザー**

⚠ 注 意

本製品およびマニュアルは，以下の様な，ユーザーを対象としています．

・拡張用ボードの増設および配線に付いて基本的な知識を有している方．
・制御用電子機器およびパソコン等に付いて基本的な知識を有している方．

■ **添付ソフトウェア適合 OS**

⚠ 注 意

添付ソフトウェアは，Windows7 以降(32bit/64bit)の各エディションにおいてボードの制御を行う為のソフトウェアです．
Windows XP SP3(32bit)，Windows2000，WindowsNT4.0，Windows98SE についてはマイクロソフト社各 OS サポートのライフサイクル期間に確認したものであり，本マニュアル発行時点での動作を保証するものではありません．
上記以外の OS でのご使用については，弊社営業までお問合せ下さい．

■ **動かしてみるプログラム**

⚠ 警 告

本製品に添付される「動かしてみる」プログラムは，ボードが正しく設定・装着されているか，動作環境が正しく設定されているかを確認するとともに，ボードの機能・動作を理解して頂く為のものです．
故に使用される機器毎に固有な安全対策処理等を含んでいませんので，「動かしてみる」プログラムを定常的に機器運転に使用しないで下さい．

モータや装置を接続して動作させる場合は，モータや装置の特性を考慮した動作条件を設定願います．
特に試運転時は，十分に安全な値で実施し，徐々に所定の値に変更することをお勧めします．

動かしてみるプログラムを使用し装置を動作させる時，最初は速度の低いところで，また機械系に合った設定を行って動作を確認して下さい．機械系に合わない設定で動作を行うと思わぬ動きをすることがあります．

■ **サンプルプログラム**

⚠ 警 告

⚠ 本製品に添付されるサンプルプログラムは，ボードを制御する手順・制御プログラムの作成方法を理解して頂く為のものです．
故に使用される機器毎に固有な安全対策処理等を含んでいませんので，サンプルプログラムを定常的に機器運転に使用しないで下さい．

⚠ モータや装置を接続して動作させる場合は，モータや装置の特性を考慮した動作条件を設定願います．
特に試運転時は，十分に安全な値で実施し，徐々に所定の値に変更することをお勧めします．

⚠ サンプルプログラムを使用し装置を動作させる時，最初は速度の低いところで，また機械系に合った設定を行って動作を確認して下さい．機械系に合わない設定で動作を行うと思わぬ動きをすることがあります．

■ **ユーザープログラム**

⚠ 警 告

⚠ 本製品を使用し装置を動作させる時は，プログラムのデバッグを充分行ってから動作させて下さい．プログラムに間違いがありますと，思わぬ動きをすることがあります．

■ **試運転・調整**

⚠ 警 告

⚠ 本シリーズ製品を使用し装置を動作させる時は，プログラムのデバッグを充分行ってから動作させてください．プログラムに間違いがありますと，思わぬ動きをすることがあります．

⚠ 本シリーズ製品に添付してあるサンプルプログラムを使用し装置を動作させる時，最初は速度の低いところで，また機械系に合った設定を行って動作を確認してください．機械系に合わない設定で動作を行うと思わぬ動きをすることがあります．

■ CPD シリーズのマニュアル構成

CPD シリーズ製品のマニュアルは

(1) CPD シリーズユーザーズマニュアル <導入編>
(2) CPD シリーズユーザーズマニュアル <運用編>
(3) 各製品ユーザーズマニュアル <ハードウェア編>
(4) 各製品ユーザーズマニュアル <ソフトウェア編> (標準添付は Windows 版, DOS 版は別途)

の 4 部構成です.

各マニュアルの内容は以下の通りです.

CPD シリーズユーザーズマニュアル ＜導入編＞
－全ての開発者向け

- CPD シリーズ概要
- インストール
- 試運転
- 用語解説

CPD シリーズユーザーズマニュアル ＜運用編＞
－主としてソフトウェア開発者向け

- 基本的な運用
- 特殊な運用

各製品ユーザーズマニュアル ＜ハードウェア編＞
－主として配線担当者向け

- 製品仕様, 購入時オプション
- ブロック図
- 接続構成
- ボード上の設定
- 外部との接続
- アクセサリ(中継コネクタボード, 接続ケーブルなど)
- 各社サーボアンプとの 接続例

各製品ユーザーズマニュアル ＜ソフトウェア編＞
－主としてソフトウェア開発者向け

- ソフトウェア概要
- ライブラリ関数
- ドライバ関数
- サンプルプログラム
- ポート資料

1. はじめに

本マニュアルは HPCI-CPD5212M(以下，CPD)で使用される Windows 版添付ソフトウェアの API 関数の説明書です．.

1.1 ソフトウェアの構成

ソフトウェアの構成を下図に示します．

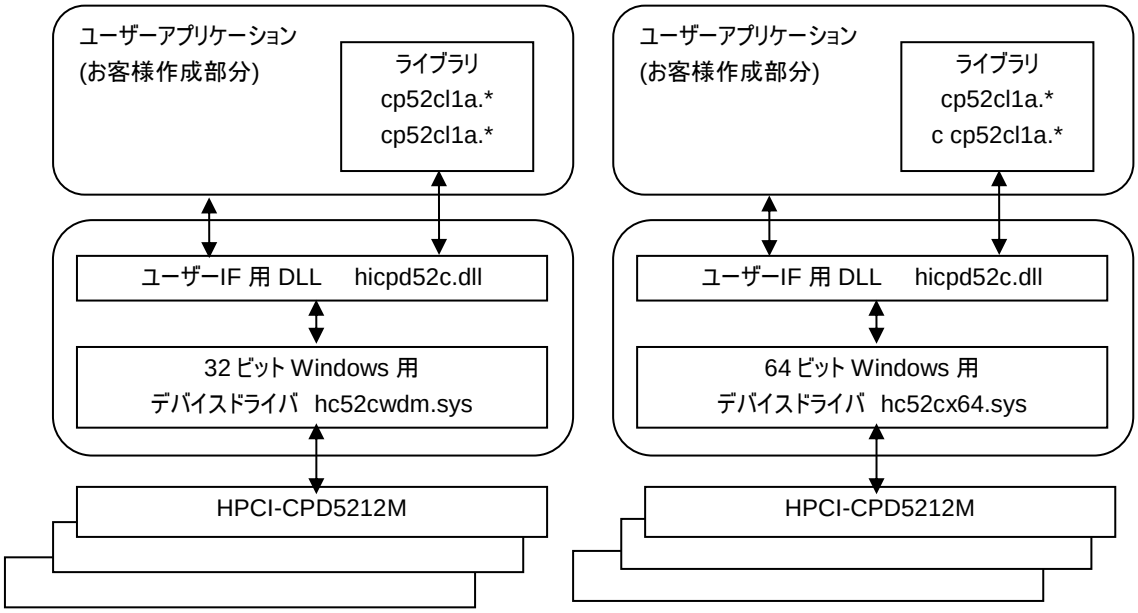


図 1.1-1　ソフトウェアの構成

1.2 添付ファイル名

No.	言語	ファイル名	内容
	VC	hicpd52c.h	ドライバ関数ヘッダファイル
		hicpd52c.lib	ドライバ関数リンク用 LIB ファイル
		cp52cl1a.h	ライブラリ関数ヘッダファイル
		cp52cl1a.c	ライブラリ関数ソースファイル
	VB6	hicpd52c.bas	ドライバ関数用標準モジュール
		cp52cl1a.bas	ライブラリ関数ソースファイル
	VB.NET	hicpd52c.vb	ドライバ関数用定義ファイル
		cp52cl1a.vb	ライブラリ関数用ソースファイル
	VC#	hicpd52c.cs	ドライバ関数用定義ファイル
		cp52cl1a.cs	ライブラリ関数用ソースファイル

表 1.2-1　添付ファイル名

1.3 関数名

添付される API 関数名は，以下の様になります．

No.	種別	関数名
1	ライブラリ関数	hcp52c_xxxx
2	ドライバ関数	cp52c_xxxx

表 1.3-1　関数名

1.4 関数の戻り値

ライブラリおよびドライバの諸関数を使用する時，関数の戻り値が異常値（'0'以外)であった場合には，異常内容に対応した処理を行います．通常，この異常が発生した場合にはアプリケーションプログラムの続行は困難であり，プログラム内容の再検討が必要となります．

No	記号表記	戻り値		異常内容と確認項目
		16 進数表記		
		VC++,VC#	VB, VB.NET	
1	NO_ERROR	0x000	&H0	正常
				異常は発生していません
2	NOT_FOUND	0x001	&H1	デバイスドライバが存在しない
				◎デバイスドライバがインストールされていない ◎デバイスドライバが所定のフォルダに格納されていない
3	ALREADY_OPENED	0x002	&H2	既にオープン済のデバイスをオープン
				◎オープン済みデバイスに更にオープン指令 ◇オープンしたデバイスはクローズするまで使用（多重のオープンは禁止） ◎ボード 2 枚以上使用する場合,オープンするデバイス情報の更新を確認します.
4	INSUFFICIENT_MEMORY	0x004	&H4	デバイス情報格納メモリが不足
				◎アプリケーション用のメモリ不足 ◇パソコン主記憶メモリの不足 ◎システムリソース（OS 用メモリ)の不足 ◇多数のアプリケーション起動 ◇1度に多数のウィンドウを開いた
5	INVALID_HANDLE	0x008	&H8	無効なデバイスハンドルを指定
				◎デバイスオープンで得られた"デバイスハンドル"の不使用 ◎このデバイスは既にクローズされている
6	NOT_READY	0x010	&H10	デバイスの入出力ポートが使用できない
				◎システムが不安定になっている可能性がありますので，弊社サポートまでお問い合わせください
7	ILLEGAL_DEVICE	0x020	&H20	ボード固有情報が不正
				◎ポートの読み出しができない状態です．弊社サポートまでお問い合わせください
8	ILLEGAL_PARAM	0x100	&H100	関数の引数の値が異常
				◇速度倍率設定値の範囲は 2～4095. ◇その他引数の設定値を確認

表 1.4-1 関数の戻り値

なお，サーボ装置・メカセンサに起因する異常(サーボアラームやエンドリミットによる停止など)はこの異常報告に含まれません．個々の要因毎に，異常発生内容を明確にすると共に，適切な処置が求められます．

1.5 アプリケーション作成準備

1.5.1 Microsoft Visual C++ (2008 以上)

プロジェクト作成後，次のファイルをプロジェクトへ追加します．

No.	ファイル名	内　容
1	hicpd52c.h	関数定義ヘッダファイル
2	hicpd52c.lib	関数インポートライブラリファイル 32 ビットアプリケーション用と 64 ビットアプリケーション用があります． 32 ビットアプリケーション用は添付 CD の¥include¥vc_x86 フォルダ内， 64 ビットアプリケーション用は添付 CD の¥include¥vc_x64 フォルダ内にあります．
3	cp52cl1a.c	ライブラリ関数ソースコードファイル
5	cp52cl1a.h	ライブラリ関数定義ヘッダファイル

例．

```
#include "hicpd52c.h"
#include "cp52cl1a.h"
```

1.5.2 Microsoft Visual C#(2008 以上)

プロジェクト作成後，次のファイルをプロジェクトへ追加します．

No.	ファイル名	内　容
1	hicpd52c.cs	関数定義ファイル
2	cp52cl1a.cs	ライブラリ関数ソースコードファイル

1.5.3 Microsoft Visual Basic(2008 以上)

プロジェクト作成後，次のファイルをプロジェクトへ追加します．

No.	ファイル名	内　容
1	hicpd52c.vb	関数定義ファイル
2	cp52cl1a.vb	ライブラリ関数ソースコードファイル

1.5.4 Microsoft Visual Basic 6.0

プロジェクト作成後，次のファイルをプロジェクトへ追加します．

No.	ファイル名	内　容
1	hicpd52c.bas	関数定義標準モジュール
2	cp52cl1a.bas	ライブラリ関数標準モジュール

1.5.5　ボードを複数枚使用する場合

HPCI-CPD5212Mを1台のコンピュータに複数枚装着し，それぞれのボードと外部の接続を1対1に対応させる場合について説明します．

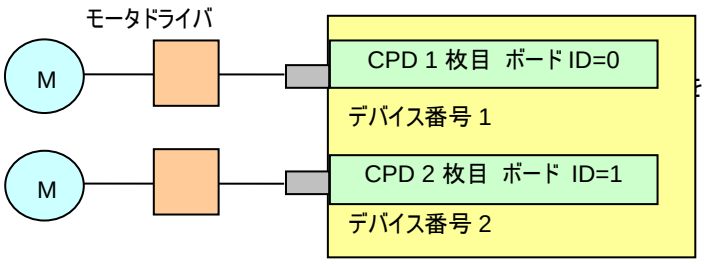


図 1.5-1　ボードを複数枚使用

(1)　ボードのスロット番号とボード ID

PCI ではシステムがボードのアドレス管理をしています．
ボードが装着されるスロットにはシステム側で決めたデバイス番号が割振られます．
しかし，このデバイス番号はシステムによって割振られるため，ボードとスロットの関係が外部からの認識が直接出来ません．
このために，CPD には「ボード ID」が設けられています．これにより，ボードとソフトを対応させることが出来ます．

(2)　ボード ID の使用

ボード ID は No.0-15 が設定出来ます．HPCI-CPD5212M を 16 枚まで扱えます．

1.5.6　ボードアクセス方法

添付される API 関数で複数の CPD を制御することができます．あるひとつの CPD にアクセスするためには，まずこのデバイスをオープンして,アクセスするために必要なデバイスハンドル値を取得する必要があります．
デバイスをオープンするためには,どのようなハードウェアリソースを持つデバイスをオープンするのかという情報が必要となります．
この情報をデバイス情報と呼びます．
(I/O ポートアドレスや IRQ 番号等のハードウェアリソースは,システム側によって確定されます．)

(1)　デバイス情報構造体

```
typedef struct _HCP52CINFO {    // デバイス情報
    DWORD  dwBusNumber;         // バス番号
    DWORD  dwDeviceNumber;      // デバイス番号
    DWORD  dwBaseAddress2;      // ベースアドレス２
    DWORD  dwBaseAddress3;      // ベースアドレス３
    DWORD  dwBaseAddress4;      // ベースアドレス４
    DWORD  dwIrqNo;             // IRQ 番号
    DWORD  dwNumber;            // 管理番号(Windows 98では無視)
    DWORD  dwBoardID;           // ボードID
} HCP52CINFO, *PHCP52CINFO;
```

その他の言語は添付ファイル hicpd52c.*を参照してください．

1.6 アプリケーション作成上の注意

Windows ではマルチスレッドがサポートされていますが，マルチスレッドを使用し複数のスレッドからボードにアクセスする場合は同期が必要になる場合があります．

CPD に搭載されているパルスコントローラ「PCL6045」(以下 PCL)を制御する時には，PCL 内部のレジスタのデータを読出し，または書込みますがこの時に同期が必要になります．以下に解説します．

【 ポートアドレス 】

1 軸分のポートアドレスを以下に記します．（16 ビットアクセス時）

アドレス	読込み(INP)		書込み(OUT)	
	呼称	内　　容	呼称	内　　容
+0	MSTS	メインステータス	CMD	コマンド
+2	SSTS	サブステータス	OTP	不使用（予約）
+4	BUF0	入出力バッファ IN (15- 0)	BUF0	入出力バッファ OUT(15- 0)
+6	BUF1	入出力バッファ IN (31-16)	BUF1	入出力バッファ OUT(31-16)

【 レジスタ読出し，書込み手順 】

レジスタからの読出し，またはレジスタへの書込み手順は以下のようになります．

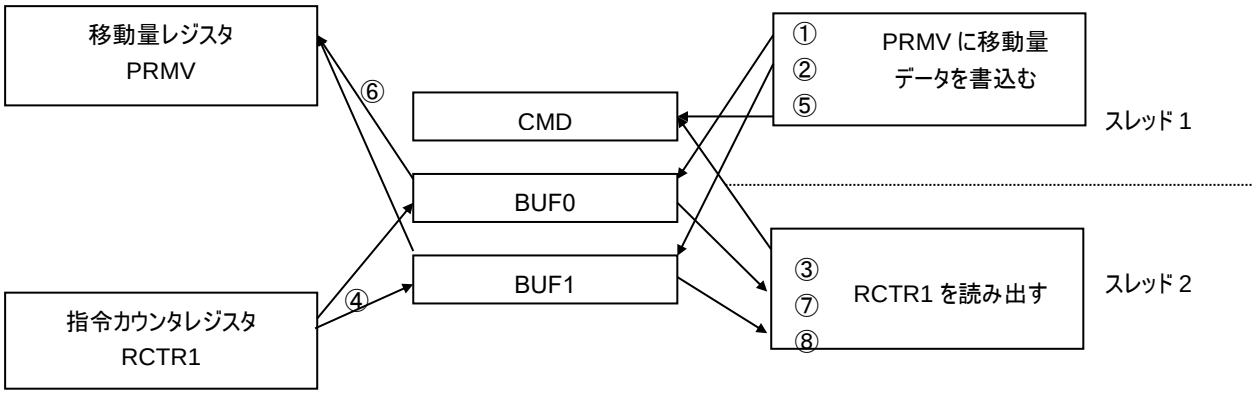
■レジスタ読出し

(1) レジスタ読出しコマンドを CMD に書込みます．
(2) レジスタの内容が BUF0，BUF1 に読み出されます．
(3) BUF0，BUF1 を読出します．

■レジスタ書込み

(1) BUF0，BUF1 にデータを書込みます．
(2) レジスタ書込みコマンドを CMD に書込みます．
(3) レジスタに BUF0，BUF1 のデータが書込まれます．

この時，同期を行わないと，どのように不具合が発生するか考えてみます．
例えば下図のような順序の時は，



(1) スレッド1が BUF0 にデータを書く
(2) スレッド1が BUF1 にデータを書く
(3) スレッド2が CMD に RCTR1 読出しコマンドを書く
(4) RCTR1の内容が BUF0，BUF1 にコピーされる
(5) スレッド1が CMD に PRMV 書込みコマンドを書く
(6) PRMV にスレッド2で読出した RCTR1 のデータが書かれてしまう

となり，PRMV に正しいデータが書けません．
そのため，スレッド 1 とスレッド 2 の間で同期をとる(排他をする)必要があります．

No	関数名“xxx”	グループ	No	関数名“xxx”	グループ
1	hcpxxx_GetDevInfo		35	hcpxxx_DecStop	
2	hcpxxx_DevOpen	○	36	hcpxxx_QuickStop	
3	hcpxxx_DevClose		37	hcpxxx_EmgStop	
4	hcpxxx_SetOrgMode	○	38	hcpxxx_SyDecStop	○
5	hcpxxx_SetEls	○	39	hcpxxx_SyQuickStop	○
6	hcpxxx_SetOls	○	40	hcpxxx_AccStart	○
7	hcpxxx_SetSvAlm	○	41	hcpxxx_CnstStartFH	○
8	hcpxxx_SetEz	○	42	hcpxxx_CnstStartFL	○
9	hcpxxx_SetInpos	○	43	hcpxxx_CnstStartByDec	○
10	hcpxxx_SetSvCtrCl	○	44	hcpxxx_SvOn	
11	hcpxxx_SetSls	○	45	hcpxxx_SvOff	
12	hcpxxx_SetCmdPulse	○	46	hcpxxx_SvResetOn	
13	hcpxxx_SetAccProfile	○	47	hcpxxx_SvResetOff	
14	hcpxxx_SetAutoDec	○	48	hcpxxx_PMOn	
15	hcpxxx_ReadMainSts		49	hcpxxx_PMOff	
16	hcpxxx_ReadErrorSts	○	50	hcpxxx_CalAccRate	
17	hcpxxx_ReadEventSts	○	51	cpxxx_GetDeviceCount()	
18	hcpxxx_ReadSubSts		52	cpxxx_GetDeviceInfo()	
19	hcpxxx_ReadExSts	○	53	cpxxx_OpenDevice()	
20	hcpxxx_ReadSpd	○	54	cpxxx_CloseDevice()	
21	hcpxxx_ReadCtr	○	55	cpxxx_rMstsW()	
22	hcpxxx_SetFLSpd	○	56	cpxxx_rSstsW()	
23	hcpxxx_SetAuxSpd	○	57	cpxxx_wCmdW()	
24	hcpxxx_SetAccRate	○	58	cpxxx_rReg()	○
25	hcpxxx_SetDecRate	○	59	cpxxx_wReg()	○
26	hcpxxx_SetMult	○	60	cpxxx_rPortB()	○
27	hcpxxx_SetEventMask	○	61	cpxxx_wPortB()	○
28	hcpxxx_SetDecPoint	○	62	cpxxx_rPortW()	○
29	hcpxxx_WritOpeMode	○	63	cpxxx_wPortW()	○
30	hcpxxx_WritFHSpd	○	64	cpxxx_rBufDW()	○
31	hcpxxx_WritPos	○	65	cpxxx_wBufDW()	○
32	hcpxxx_WritLine	○			○
33	hcpxxx_WritCircl	○			
34	hcpxxx_WritCtr	○			

○のグループの関数を複数のスレッドで呼ぶ場合は同期をとる必要があります.

表 1.6-1 排他処理が必要な関数

注1. VC 用ライブラリの場合

cp52cl1a.c 内の APP_SYNC を define することでライブラリ関数ごとの排他処理が入ります.
但し, ドライバ関数を直接呼ぶ場合は排他処理が入りませんのでアプリケーションでの排他処理が必要となります.
また, 複数枚数使用される場合はアプリケーションでの排他を推奨します.

注2. VC#(.NET2003 以上)用ライブラリの場合

cp52cl1a.cs 内の APP_SYNC を define することでボード単位での排他処理が入ります.
但し, ドライバ関数を直接呼ぶ場合は排他処理が入りませんのでアプリケーションでの排他処理が必要となります.
また, 複数枚数使用される場合はアプリケーションでの排他を推奨します.

注3. VB(.NET2003 以上)用ライブラリの場合

アプリケーションでの排他処理が必要となります.

2. ライブラリ関数

2.1 ライブラリ関数序論

ライブラリ関数は，ソースプログラムで提供され，アプリケーションと同時にビルドします．

2.1.1 軸の指定

初期設定，状態読み出し，動作設定，運用設定(円弧補間の移動量の設定を除く)の関数の軸指定は
0:X1 軸，1:Y1 軸，2:Z1 軸，3:U1 軸，4:X2 軸，5:Y2 軸，6:Z2 軸，7:U2 軸，8:X3 軸，9:Y3 軸，10:Z3 軸，11:U3 軸
となります．
動作制御指令の関数の軸指定は
1:X1 軸，2:Y1 軸，4:Z1 軸，8:U1 軸，10h:X2 軸，20h:Y2 軸，40h:Z2 軸，80h:U2 軸，
100h:X3 軸，200h:Y3 軸，400h:Z3 軸，800h:U3 軸
となります．

2.1.2　ライブラリ関数一覧

黄色に網掛けされている関数は重要な関数です.

No.	関数名		機　能
1	デバイス関係	hcp52c_GetDevInfo	デバイス個数，デバイス情報の取得
2		hcp52c_DevOpen	デバイスのオープン，レジスタの初期化
3		hcp52c_DevClose	デバイスのクローズ
4	初期設定	hcp52c_SetOrgMode	原点復帰モードの設定
5		hcp52c_SetEls	ELS の設定
6		hcp52c_SetOls	OLS の設定
7		hcp52c_SetSvAlm	SVALM の設定
8		hcp52c_SetEz	エンコーダZ相の設定
9		hcp52c_SetInpos	INPOS の設定
10		hcp52c_SetSvCtrCl	偏差カウンタクリア出力の設定
11		hcp52c_SetSls	ソフトリミットの設定
12		hcp52c_SetCmdPulse	指令パルスの出力形式選択
13		hcp52c_SetAccProfile	加減速形式の設定
14		hcp52c_SetAutoDec	減速開始点の設定方式の自動/手動切り替え
15	状態読出し	hcp52c_ReadMainSts	メインステータスの読出し
16		hcp52c_ReadErrorSts	エラーステータスの読出し
17		hcp52c_ReadEventSts	イベントステータスの読出し
18		hcp52c_ReadSubSts	サブステータスの読出し
19		hcp52c_ReadExSts	拡張ステータスの読出し
20		hcp52c_ReadSpd	指令速度の読出し
21		hcp52c_ReadCtr	カウンタの読出し
22	動作設定	hcp52c_SetFLSpd	ベース速度の設定
23		hcp52c_SetAuxSpd	補助速度の設定
24		hcp52c_SetAccRate	加速レートの設定
25		hcp52c_SetDecRate	減速レートの設定
26		hcp52c_SetMult	速度倍率レジスタ値の設定
27		hcp52c_SetEventMask	イベントマスクの設定
28		hcp52c_SetDecPoint	減速開始点の設定
29	運用設定	hcp52c_WritOpeMode	動作モードの設定
30		hcp52c_WritFHSpd	動作速度の設定
31		hcp52c_WritPos	移動量の設定
32		hcp52c_WritLine	直線補間の移動量の設定
33		hcp52c_WritCircl	円弧補間の移動量の設定
34		hcp52c_WritCtr	カウンタプリセット
35	動作制御指令	hcp52c_DecStop	減速停止
36		hcp52c_QuickStop	即停止
37		hcp52c_EmgStop	非常停止
38		hcp52c_SyDecStop	複数軸同時減速停止
39		hcp52c_SyQuickStop	複数軸同時即停止
40		hcp52c_AccStart	加速スタート
41		hcp52c_CnstStartFH	FH定速スタート
42		hcp52c_CnstStartFL	FL定速スタート
43		hcp52c_CnstStartByDec	FH定速スタート後減速停止
44		hcp52c_SvOn	サーボオン
45		hcp52c_SvOff	サーボオフ
46		hcp52c_SvResetOn	サーボリセットオン
47		hcp52c_SvResetOff	サーボリセットオフ
48		hcp52c_PMOn	パルスモータ励磁オン
49		hcp52c_PMOff	パルスモータ励磁オフ
50	加減速レートの計算	hcp52c_CalAccRate	加減速レートの計算

表 2.1-1　ライブラリ関数一覧

2.2 デバイス関係

2.2.1 hcp52c_GetDevInfo() ボード枚数取得，デバイス情報の取得

機　能	現在パソコンに装着されている CPD の枚数,及びデバイス情報を取得します.

開発言語	書　式
VC++	DWORD hcp52c_GetDevInfo(DWORD* *dwNumber*, HCP52CINFO* *hInfo*);
VB6	Public Function hcp52c_GetDevInfo(_ ByRef *dwNumber* As Long, *hInfo* As HCP52CINFO) As Long
VB.NET	Public Function hcp52c_GetDevInfo(_ ByRef *dwNumber* As Integer, ByRef *hInfo* As HCP52CINFO) As Integer
VC#	public static uint hcp52c_GetDevInfo(ref uint *dwNumber*, ref HCP52CINFO *hInfo*);

引　数	説　明
dwNumber	CPD の枚数
hInfo	CPD のデバイス情報格納の構造体

VC++
記述例
	DWORD count;
	DWORD ret;
	HCP52CINFO HpcDevInfo[0]; //ボードのデバイス情報が格納されているエリアの先頭アドレス
	ret = hcp52c_GetDevInfo(&count, &HpcDevInfo[0]);

2.2.2 hcp52c_DevOpen() デバイスのオープン，レジスタとオプションポートの初期化

機　能	指定したデバイス情報を持つ CPD をオープンし，他の CPD と識別するためのデバイスハンドルを取得します．以降このデバイスハンドルは，指定した CPD にアクセスするために使用します． またオープンした CPD のレジスタとオプションポートの初期化をします．

開発言語	書　式
VC++	DWORD hcp52c_DevOpen(DWORD* *hDevID*, HCP52CINFO * *hInfo*);
VB6	Public Function hcp52c_DevOpen(ByRef *hDevID* As Long, *hInfo* As HCP52CINFO) As Long
VB.NET	Public Function hcp52c_DevOpen(ByRef *hDevID* As Integer, ByRef *hInfo* As HCP52CINFO) As Integer
VC#	public static uint hcp52c_DevOpen(ref uint *hDevID*, ref HCP52CINFO *hInfo*);

引　数	説　明
hDevID	デバイスハンドルの格納先
hInfo	CPD のデバイス情報格納の構造体

VC++
記述例
	// パソコンに CPD が 2 枚装着されていることを想定します．
	// デバイス情報格納エリアとしてデバイス情報構造体の配列 hInfo[2]を準備し，この中には
	// 既に hcp52c_GetDevInfo 関数により全ボードのデバイス情報が入っているものとします．
	DWORD ret; //関数の戻り値
	DWORD hDevID[2]; //デバイスハンドル取得エリア
	ret = hcp52c_DevOpen(hDevID[0],&hInfo[0]); //1 番目のデバイス情報
	ret = hcp52c_DevOpen(hDevID[1],&hInfo[1]); //2 番目のデバイス情報

備　考

レジスタ	内　容	初期値	補　足
PRFL,RFL	ベース速度	200	200pps
PRFH,RFH	動作速度	2000	2000pps
PRUR,RUR	加速レート	1364	直線加減速時 00→2000pps(2000pps→200pps) 加速(減速)時間：約 0.5 秒
PRMG,RMG	速度倍率	299	1 倍
RFA	補助速度	200	200pps
PRMD,RMD	動作モード	08008000h	
RENV1	環境設定 1	20434004h	
RENV2	環境設定 2	0020fd55h	
RENV3	環境設定3	00f00002h	原点復帰モード2(OLS+Z相)等
RIRQ	イベントマスク設定	1	正常停止時
その他		0	

オプションポート	初期値	補　足
ELS 入力極性	0	全軸 B 接
その他	0	

2.2.3 hcp52c_DevClose() デバイスのクローズ

機　能	デバイスハンドルで指定された CPD をクローズします．以降,このデバイスハンドルは無効となります．

開発言語	書　式
VC++	DWORD hcp52c_DevClose(DWORD*hDevID*);
VB6	Public Function hcp52c_DevClose(ByVal *hDevID* As Long) As Long
VB.NET	Public Function hcp52c_DevClose(ByVal *hDevID* As Integer) As Integer
VC#	public static uint hcp52c_DevClose(uint *hDevID*);

引　数	説　明
hDevID	デバイスハンドル

備　考	デバイスクローズの前に CPD の終了処理を行ってください．

2.3 初期設定

2.3.1 hcp52c_SetOrgMode() 原点復帰モードの設定

機　能	デバイスハンドルで指定された CPD の指定された軸の原点復帰モードを設定します.

開発言語	書　式
VC++	DWORD hcp52c_SetOrgMode(DWORD *hDevID*, WORD *axis*, WORD*mode*);
VB6	Public Function hcp52c_SetOrgMode(_ ByVal *hDevID* As Long, ByVal *axis* As Integer, ByVal *mode* As Integer) As Long
VB.NET	Public Function hcp52c_SetOrgMode(_ ByVal *hDevID* As Integer, ByVal *axis* As Short, ByVal *mode* As Short) As Integer
VC#	public static uint hcp52c_SetOrgMode(uint *hDevID*, ushort *axis*, ushort *mode*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
mode	原点復帰モード 0：ORGmode0　OLSoff→on で即停止(加減速動作時は減速停止) 1：ORGmode1　OLSoff→on で即停止(加減速動作時は減速停止)後，補助速度定速で逆方向へ OLSoff まで動作し，その後補助速度で初めの方向へ動作し OLSoff→on で即停止 2：ORGmode2　定速時は OLSoff→on 後の Z 相カウントアップで即停止 加減速動作時は OLSoff→on で減速，Z 相カウントアップで即停止 3：ORGmode3　定速時は OLSoff→on 後の Z 相カウントアップで即停止 加減速動作時は OLSoff→on で減速，Z 相カウントアップで減速停止 4：ORGmode4　OLSoff→on で即停止(加減速動作時は減速停止)後に補助速度定速で逆転，OLSon→off 後の Z 相カウントアップ時に即停止 5：ORGmode5　OLSoff→on で即停止(加減速動作時は減速停止)後に逆転，OLSon→off 後の Z 相カウントアップ時に即停止(加減速動作時は減速停止) 6：ORGmode6　ELSon で停止後，補助速度定速で逆転，ELSoffF で即停止 7：ORGmode7　ELSon で停止後，補助速度定速で逆転、ELSoff 後の Z 相カウントアップ時に即停止 8：ORGmode8　ELSonで停止後に逆転ELSoff後のZ相カウントアップで即停止(加減速動作時は減速停止) 9：ORGmode9　ORGmode0の動作後，機械位置(CTR2)0点復帰 10：ORGmode 10　ORGmode3の動作後，機械位置(CTR2)0点復帰 11：ORGmode 11　ORGmode5の動作後，機械位置(CTR2)0点復帰 12：ORGmode 12　ORGmode8の動作後，機械位置(CTR2)0点復帰

VC++
記述例
	DWORD ret;　　//関数の戻り値
	//Y1 軸を指定，ORGmode1 (OLS 検出後抜出し再突入原点完了)
	ret = hcp52c_SetOrgMode(hDevID, 1, 1);

2.3.2 hcp52c_SetEls() ELS の設定

機　能	デバイスハンドルで指定された CPD の指定された軸の ELS 入力極性と入力時停止方法を設定します.

開発言語	書　式
VC++	DWORD hcp52c_SetEls(DWORD *hDevID*, WORD *axis*, WORD *pol*, WORD*stop*);
VB6	Public Function hcp52c_SetEls(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByVal *pol* As Integer, ByVal st As Integer) As Long
VB.NET	Public Function hcp52c_SetEls(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByVal *pol* As Short, ByVal *stop* As Short) As Integer
VC#	public static uint hcp52c_SetEls(uint *hDevID*, ushort *axis*, ushort *pol* ushort *stop*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
pol	入力極性［0:B 接，1:A 接］
stop	停止方法［0:即停止，1:減速停止］

VC++
記述例
	DWORDret; //関数の戻り値
	ret = hcp52c_SetEls(hDevID, 1, 1, 0); //Y1 軸を指定，A 接，即停止

2.3.3 hcp52c_SetOls() OLS の設定

機　能	デバイスハンドルで指定された CPD の指定された軸の OLS 入力極性を設定します.

開発言語	書　式
VC++	DWORD hcp52c_SetOls(DWORD *hDevID*, WORD *axis*, WORD *pol*);
VB6	Public Function hcp52c_SetOls(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByVal *pol* As Integer) As Long
VB.NET	Public Function hcp52c_SetOls(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByVal *pol* As Short) As Integer
VC#	public static uint hcp52c_SetOls(uint *hDevID*, ushort *axis*, ushort *pol*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
pol	入力極性［0:B 接，1:A 接］

VC++
記述例
	DWORDret; //関数の戻り値
	ret = hcp52c_SetOls(hDevID, 1, 1); // Y1 軸を指定，A 接

2.3.4 hcp52c_ SetSvAlm () SVALM の設定

機　能	デバイスハンドルで指定された CPD の指定された軸の SVALM 入力極性と入力時停止方法を設定します.

開発言語	書　式
VC++	DWORD hcp52c_SetSvAlm(DWORD *hDevID*, WORD *axis*, WORD *pol*, WORD*stop*);
VB6	Public Function hcp52c_SetSvAlm(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByVal *pol* As Integer, ByVal *stop* As Integer) As Long
VB.NET	Public Function hcp52c_SetSvAlm(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByVal *pol* As Short, ByVal *stop* As Short) As Integer
VC#	public static uint hcp52c_SetSvAlm(uint *hDevID*, ushort *axis*, ushort *pol*, ushort *stop*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
pol	入力極性［0:B 接，1:A 接］
stop	停止方法［0:即停止，1:減速停止］

VC++ 記述例	DWORDret;　　//関数の戻り値 ret = hcp52c_SetSvAlm(hDevID, 1, 1, 0);　　//Y1 軸を指定，A 接，即停止

2.3.5 hcp52c_SetEz() エンコーダ Z 相の設定

機　能	デバイスハンドルで指定された CPD の指定された軸のエンコーダ Z 相の入力処理を設定します.

開発言語	書　式
VC++	DWORD hcp52c_SetEz(DWORD *hDevID*, WORD *axis*, WORD *zcount*, WORD *pol*);
VB6	Public Function hcp52c_SetEz(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByVal *zcount* As Integer, ByVal *pol* As Integer) As Long
VB.NET	Public Function hcp52c_SetEz(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByVal *zcount* As Short, ByVal *pol* As Short) As Integer
VC#	public static uint hcp52c_SetEz(uint *hDevID*, ushort *axis*, ushort *zcount*, ushort *pol*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
zcount	原点復帰時の Z 相カウント回数 [0:1 回目,～,15:16 回目]
pol	入力極性［0:A 接，1:B 接］

VC++
記述例
	DWORDret; //関数の戻り値
	ret = hcp52c_SetEz(hDevID, 1, 2, 1);　//Y1 軸を指定，3 回目，A 接

2.3.6 hcp52c_SetInpos () INPOS の設定

機　能	デバイスハンドルで指定された CPD の指定された軸の INPOS の入力信号処理方法を設定します.

開発言語	書　式
VC++	DWORD hcp52c_SetInpos(DWORD *hDevID*, WORD *axis*, WORD *enable*, WORD *pol*);
VB6	Public Function hcp52c_SetInpos(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByVal *enable* As Integer, ByVal *pol* As Integer) As Long
VB.NET	Public Function hcp52c_SetInpos(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByVal *enable* As Short, ByVal *pol* As Short) As Integer
VC#	public static uint hcp52c_SetInpos(uint *hDevID*, ushort *axis*, ushort *enable*, ushort *pol*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
enable	INPOS 制御[0:OFF,1:ON]
pol	入力極性［0:B 接，1:A 接］

VC++
記述例
	DWORDret; //関数の戻り値
	ret = hcp52c_SetInpos(hDevID, 1, 1, 1); //Y1 軸を指定，INPOS 制御 ON，A 接

2.3.7 hcp52c_SetSvCtrCl() 偏差カウンタクリア出力の設定

機　能	デバイスハンドルで指定された CPD の指定された軸の偏差カウンタクリア自動出力の設定をします.

開発言語	書　式
VC++	DWORD hcp52c_SetSvCtrCl(DWORD *hDevID*, WORD *axis*, WORD *enable*);
VB6	Public Function hcp52c_SetSvCtrCl(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByVal *enable* As Integer) As Long
VB.NET	Public Function hcp52c_SetSvCtrCl(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByVal *enable* As Short) As Integer
VC#	public static uint hcp52c_SetSvCtrCl(uint *hDevID*, ushort *axis*, ushort *enable*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
enable	自動出力設定［0:不使用,1:原点完了時,2:異常停止時,3:原点完了及び異常停止時］

VC++
記述例
	DWORDret; //関数の戻り値
	ret = hcp52c_SetSvCtrCl(hDevID, 1, 1); // Y1 軸を指定，原点完了時出力

2.3.8 hcp52c_SetSls() ソフトリミットの設定

機　能	デバイスハンドルで指定された CPD の指定された軸のソフトリミットの設定をします.

開発言語	書　式
VC++	DWORD hcp52c_SetSls(DWORD *hDevID*, WORD *axis*, long *psls*, long *msls*, WORD *enable*, WORD *stop*);
VB6	Public Function hcp52c_SetSls(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByVal *psls* As Long, ByVal *msls* As Long, ByVal *enable* As Integer, ByVal *stop* As Integer) _ As Long
VB.NET	Public Function hcp52c_SetSls(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByVal *psls* As Integer, ByVal *msls* As Integer, ByVal *enable* As Short, ByVal *stop* As Short) _ As Integer
VC#	public static uint hcp52c_SetSls(uint *hDevID*, ushort *axis*, int *psls*, int *msls*, ushort *enable*, ushort us *stop*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
psls	＋SLS[パルス数]
msls	－SLS[パルス数]
enable	使用/不使用[0:不使用,1:使用]
stop	停止方法[0:即停止,1:減速停止]

VC++
記述例
	DWORDret;　　//関数の戻り値
	//Y1 軸, +SLS = 100000, -SLS = -50000, ソフトリミット使用, 即停止
	ret = hcp52c_SetSls(hDevID, 1, 100000, -50000, 1, 0);

備　考	1.ソフトリミット使用時は+SLS は必ず－SLS より大きくして下さい. 2.ソフトリミット不使用時は msls = psls = enable = 0 とします. 3.スタートコマンド書込み時に SLS が ON 状態の場合, SLS が ON になる方向へはスタートはできません(動きません). 逆方向へはスタートできます.

2.3.9 hcp52c_SetCmdPulse()　指令パルスの出力形式の設定

機　能	デバイスハンドルで指定された CPD の指定された軸の指令パルスの出力形式を設定します.

開発言語	書　式
VC++	DWORD hcp52c_SetCmdPulse(DWORD *hDevID*, WORD *axis*, WORD *cmdpls*);
VB6	Public Function hcp52c_SetCmdPulse(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByVal *cmdpls* As Long) As Long
VB.NET	Public Function hcp52c_SetCmdPulse(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByVal *cmdpls* As Integer) As Integer
VC#	public static uint hcp52c_SetCmdPulse(uint *hDevID*, ushort *axis*, ushort *cmdpls*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
cmdpls	指令パルスの出力形式[0:個別指令方式,1:共通指令方式]

VC++ 記述例	DWORD ret;　　//関数の戻り値 ret = hcp52c_SetCmdPulse(hDevID, 0, 1);　//X 軸を指定，共通指令方式

2.3.10 hcp52c_SetAccProfile()　加減速形式の設定

機　能	デバイスハンドルで指定された CPD の指定された軸の加減速形式の設定をします.

開発言語	書　式
VC++	DWORD hcp52c_SetAccProfile(DWORD *hDevID*, WORD *axis*, WORD *pr*);
VB6	Public Function hcp52c_SetAccProfile(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByVal *pr*) As Long
VB.NET	Public Function hcp52c_SetAccProfile(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByVal *pr* As Short) As Integer
VC#	public static uint hcp52c_SetAccProfile(uint *hDevID*, ushort *axis*, ushort *pr*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
pr	加減速形式［0:直線,1:S 字］

VC++ 記述例	DWORD ret;　　//関数の戻り値 ret = hcp52c_SetAccProfile(hDevID, 1, 1);　　//Y1 軸を指定，S 字加減速に設定

2.3.11 hcp52c_SetAutoDec()　減速開始点計算方式の自動計算/手動計算切り替え

機　能	デバイスハンドルで指定された CPD の指定された軸の減速開始点計算方式を手動計算か自動計算か設定します.

開発言語	書　式
VC++	DWORD hcp52c_SetAutoDec(DWORD *hDevID*, WORD *axis*, WORD *para*);
VB6	Public Function hcp52c_SetAutoDec(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByVal *para* As Integer) As Long
VB.NET	Public Function hcp52c_SetAutoDec(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByVal *para* As Short) As Integer
VC#	public static uint hcp52c_SetAutoDec(uint *hDevID*, ushort *axis*, ushort *para*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
para	減速開始点の設定方式［0:自動計算設定,1:手動計算設定］

VC++ 記述例	DWORD ret;　　//関数の戻り値 ret = hcp52c_SetAutoDec(hDevID, 1, 1);　//Y1 軸の減速開始点の設定を手動計算設定

2.4 状態読み出し

2.4.1 hcp52c_ReadMainSts()　メインステータスの読出し

機　能	デバイスハンドルで指定された CPD の指定された軸のメインステータスを読出します.

開発言語	書　式
VC++	DWORD hcp52c_ReadMainSts　(DWORD *hDevID*, WORD *axis*, WORD*　*msts*);
VB6	Public Function hcp52c_ReadMainSts(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByRef *msts* As Integer) As Long
VB.NET	Public Function hcp52c_ReadMainSts(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByRef *msts* As Short) As Integer
VC#	public static uint hcp52c_ReadMainSts(uint *hDevID*, ushort *axis*, ref ushort *msts*);

引　数	説　明		
hDevID	デバイスハンドル		
axis	軸指定		
msts	メインステータス		
	ビット	名称	説明
	0	SSCM	'1':スタート指令が書込まれた.
	1	SRUN	'1':RUN 中
	3	**SEND**	'1':停止状態(*2)
	4	**SERR**	'1':エラー報告あり(エラーステータス(REST)読出しで'0')
	5	**SINT**	'1':イベント報告あり(イベントステータス(RIST)読出しで'0')
	7,6	SSCx	実行中または停止中のシーケンス番号(PRMD.b17,16 設定値)
	12-8	SCMPx	'1':CMP5-1 比較条件成立時
	13	SEOR	'1':位置のオーバライド失敗時(本ステータス読出しで'0')
	14	SPRF	'1':次動作用プリレジスタが満杯('0':書込可能)
	15	SPDF	'1':CMP5 用プリレジスタが満杯('0':書込可能)
	*1. 正常終了でのイベント報告は"イベントマスク設定:自動停止"［RIRQ.b0:ISEN=1］とします. *2. b3(SEND)は状態を示しているビットです. 電源投入直後は '0' であり, 即(減速)停止指令の実行または一度移動実行後の終了状態は'1'となります. 移動中は '0 'を示します. 通常停止中(= '1')か, 動作中(= '0')かを確認したいときに使用します.		

VC++ 記述例	DWORD ret;　　//関数の戻り値 WORD　　msts;　//メインステータス ret = hcp52c_ReadMainSts(hDevID, 1, &msts);　//Y1 軸を指定

2.4.2 hcp52c_ReadErrorSts() エラーステータスの読出し

機　能	デバイスハンドルで指定された CPD の指定された軸のエラーステータスを読出します.

開発言語	書　式
VC++	DWORD hcp52c_ReadErrorSts　(DWORD *hDevID*, WORD *axis*, DWORD* *rest*);
VB6	Public Function hcp52c_ReadErrorSts(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByRef *rest* As Long) As Long
VB.NET	Public Function hcp52c_ReadErrorSts(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByRef *rest* As Integer) As Integer
VC#	public static uint hcp52c_ReadErrorSts(uint *hDevID*, ushort *axis*, ref uint *rest*);

引　数	説　明		
hDevID	デバイスハンドル		
axis	軸指定		
rest	エラーステータス		
	ビット	名　称	説　　明
	0	ESC1	1:CMP1 条件成立で停止 (+SLS)
	1	ESC2	1:CMP2 条件成立で停止 (-SLS)
	2	ESC3	1:CMP3 条件成立で停止
	3	ESC4	1:CMP4 条件成立で停止
	4	ESC5	1:CMP5 条件成立で停止
	5	ESPL	1:+ELS 検出で停止
	6	ESML	1:-ELS 検出で停止
	7	ESAL	1:SVALM 検出で停止
	8	ESSP	1:STPon による停止
	9	ESEM	1:EMGon による停止
	10	ESSD	1:DLS 検出による停止
	12	ESDT	1: 動作データが不正で停止(注意 1)
	13	ESIP	1: 補間動作中に補間他軸の停止による停止
	14	ESPO	1:パルサ入力バッファオーバーフローによる停止
	15	ESAO	1:補間データのレンジオーバによる停止
	16	ESEE	1:エンコーダ信号エラー(停止しない)
	17	ESPE	1:パルサ入力信号エラー(停止しない)
	以下の場合に ESDT=1 になります. ● 1 軸だけ直線補間 1 モード(MOD=60h,61h,68h,69h)にしてスタートコマンドを書き込んだ時. ● 1 軸だけ円弧補間モード(MOD=64h,65h,66h,67h,6Ch,6Dh)にしてスタートコマンドを書き込んだ時. ● 円弧補間モードで RIP 設定(円弧中心座標)を(0,0)にしてスタートコマンドを書き込んだ時. ● 3 軸または 4 軸を円弧補間モードにしてスタートコマンドを書き込んだ時. ● 直線補間 2 モード(MOD=62h,63h,6Ah,6Bh), RIP=0 の状態でスタートコマンドを書き込んだ時. ● U 軸同期の円弧補間モード(MOD=66h,67h)でスタートコマンドを書き込んだ時にU軸が動作しない時. 　または円弧補間動作中に U 軸が動作完了になった時.		

VC++
記述例
	DWORD ret;　　//関数の戻り値
	DWORD rest;//エラーステータス
	ret = hcp52c_ReadErrorSts(hDevID, 1, &rest);　//Y1 軸を指定

2.4.3 hcp52c_ReadEventSts() イベントステータスの読出し

機　能	デバイスハンドルで指定された CPD の指定された軸のイベントステータスを読出します.

開発言語	書　式
VC++	DWORD hcp52c_ReadEventSts (DWORD *hDevID*, WORD *axis*, DWORD* *rist*);
VB6	Public Function hcp52c_ReadEventSts(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByRef *rist* As Long) As Long
VB.NET	Public Function hcp52c_ReadEventSts(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByRef *rist* As Integer) As Integer
VC#	public static uint hcp52c_ReadEventSts(uint *hDevID*, ushort *axis*, ref uint *rist*);

引　数	説　明		
hDevID	デバイスハンドル		
axis	軸指定		
rist	イベントステータス		
	ビット	名　称	説　　明
	0	IREN	1:動作完了報告
	1	IRN	1:次動作継続スタート報告
	2	IRNM	1:動作用プリレジスタ書込み可能報告
	3	IRND	1:CMP5 用プリレジスタ書込み可能報告
	4	IRUS	1:加速開始報告
	5	IRUE	1:加速終了報告
	6	IRDS	1:減速開始報告
	7	IRDE	1:減速終了報告
	8	IRC1	1:CMP1 比較条件成立報告(+SLS)
	9	IRC2	1:CMP2 比較条件成立報告(-SLS)
	10	IRC3	1:CMP3 比較条件成立報告(脱調検出用途)
	11	IRC4	1:CMP4 比較条件成立報告
	12	IRC5	1:CMP5 比較条件成立報告
	15	IROL	1:OLSon によるカウンタ値ラッチ報告
	16	IRSD	1:DLS 信号 OFF→ON 報告
	17	IRPD	1:+DR 信号 OFF→ON 報告
	18	IRMD	1:-DR 信号 OFF→ON 報告
	19	IRSA	1:STA 信号 OFF→ON 報告
	イベントステータスを使用するためにはイベントマスクの設定が必要です.		

VC++ 記述例	DWORD ret;　　//関数の戻り値 DWORD rist;　　//サブステータス ret = hcp52c_ReadEventSts(hDevID, 1, &rist);　//Y1 軸を指定

2.4.4 hcp52c_ReadSubSts() サブステータスの読出し

機　能	デバイスハンドルで指定された CPD の指定された軸のサブステータスを読出します.

開発言語	書　式
VC++	DWORD hcp52c_ReadSubSts　(DWORD *hDevID*, WORD *axis*, WORD*　*ssts*);
VB6	Public Function hcp52c_ReadSubSts(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByRef *ssts* As Integer) As Long
VB.NET	Public Function hcp52c_ReadSubSts(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByRef *ssts* As Short) As Integer
VC#	public static uint hcp52c_ReadSubSts(uint *hDevID*, ushort *axis*, ref ushort *ssts*);

引　数	説　明		
hDevID	デバイスハンドル		
axis	軸指定		
ssts	サブステータス		
	ビット	名　称	説　明
	0	**SVON**	'1':"SVON"出力中
	1	**SVRS**	'1':"SVRST"出力中
	8	SFU	'1':加速中
	9	SFD	'1':減速中
	10	SFC	'1':定速動作中
	11	**SALM**	'1':SVALM(サーボアラーム信号 ON 中)
	12	**SPEL**	'1':+ELS 検出中
	13	**SMEL**	'1':-ELS 検出中
	14	**SOLS**	'1':OLS 検出中
	15	**SDLS**	'1':DLS 検出中

VC++ 記述例	DWORD ret;　　//関数の戻り値 WORD　ssts;　　//サブステータス ret = hcp52c_ReadSubSts(hDevID, 1, &ssts);　　//Y1 軸を指定

2.4.5 hcp52c_ReadExSts() 拡張ステータスの読出し

機　能	デバイスハンドルで指定された CPD の指定された軸の拡張ステータスを読出します.

開発言語	書　式
VC++	DWORD hcp52c_ReadExSts (DWORD *hDevID*, WORD *axis*, DWORD* *rsts*);
VB6	Public Function hcp52c_ReadExSts(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByRef *rsts* As Long) As Long
VB.NET	Public Function hcp52c_ReadExSts(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByRef *rsts* As Integer) As Integer
VC#	public static uint hcp52c_ReadExSts(uint *hDevID*, ushort *axis*, ref uint *rsts*);

引　数	説　明		
hDevID	デバイスハンドル		
axis	軸指定		
rsts	拡張ステータス		
	ビット	名　称	説　　明
	3-0	CND3-0	動作状態 0000:停止中　　1000:パルサ入力待ち 0001:DR 入力待ち　　1001:FA 定速動作中 0010:STA 入力待ち　　1010:FL 定速動作中 0011:条件付きスタート待ち状態　　1011:加速中 0100:他軸の停止待ち　　1100:FH 定速動作中 0101:SVCTRCL タイマ完了待ち　　1101:減速中 0110:方向変化タイマ完了待ち　　1110:INPOSon 待ち 0111:バックラッシュ/スリップ補正中　　1111:その他
	4	SDIR	動作方向　0:＋方向, 1:－方向
	5	SSTA	1:同時スタート信号(STA) on
	6	SSTP	1:同時停止信号(STP) on
	7	SEMG	1:EMGon
	8	SPCS	1:PCSon
	9	SERC	1:サーボ偏差カウンタ･クリア信号 on
	10	SEZ	1:エンコーダZ相信号 on
	11	SDRP	1:＋DRon
	12	SDRM	1:－DRon
	15	SDIN	1:DLSon
	16	SINP	1:INPOSon
	19,18	PFC1,0	RCMP5 用プリレジスタの使用状態 00:未確定, 01:レジスタ確定, 10: 1st レジスタ確定, 11: 2nd レジスタ確定(レジスタフル)
	21,20	PFM1,0	動作用プリレジスタの使用状態 00:未確定, 01:レジスタ確定, 10: 1st レジスタ確定, 11: 2nd レジスタ確定(レジスタフル)

VC++
記述例
	DWORD ret;　　//関数の戻り値
	DWORD rsts; //拡張ステータス
	ret = hcp52c_ReadExSts(hDevID, 1, &rsts); //Y1 軸を指定

2.4.6 hcp52c_ReadSpd() 指令速度の読出

機　能	デバイスハンドルで指定された CPD の指定された軸の指令速度を読出します.

開発言語	書　式
VC++	DWORD hcp52c_ReadSpd(DWORD *hDevID*, WORD* *axis*, WORD* *speed*);
VB6	Public Function hcp52c_ReadSpd(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByRef *speed* As Long) As Long
VB.NET	Public Function hcp52c_ReadSpd(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByRef *speed* As Integer) As Integer
VC#	public static uint hcp52c_ReadSpd(uint *hDevID*, ushort *axis*, ref ushort *speed*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
speed	指令速度. 指令速度[pps]＝読出したデータ×速度倍率

VC++
記述例
	DWORDret; //関数の戻り値
	WORD speed; //速度
	ret = hcp52c_ReadSpd(hDevID, 0, &speed); //X1 軸を指定，格納先

2.4.7 hcp52c_ReadCtr() カウンタの読出

機　能	デバイスハンドルで指定された CPD の指定された軸の,指定されたカウンタを読出します.

開発言語	書　式
VC++	DWORD hcp52c_ReadCtr(DWORD *hDevID*, WORD* *axis*, WORD *selctr*, long* *count*);
VB6	Public Function hcp52c_ReadCtr(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByVal *selctr* As Integer, ByRef *count* As Long) As Long
VB.NET	Public Function hcp52c_ReadCtr(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByVal *selctr* As Short, ByRef *count* As Integer) As Integer
VC#	public static uint hcp52c_ReadCtr(uint *hDevID*, ushort *axis*, ushort *selctr*, ref int *count*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
selctr	カウンタ選択[1:カウンタ 1, 2:カウンタ 2, 3:カウンタ 3, 4:カウンタ 4]
count	カウンタ値

VC++
記述例
	DWORD ret; //関数の戻り値
	long* ctr1;//カウンタ 1(指令パルス出力)の値
	//X1 軸を指定，カウンタ 1 を指定，格納先
	ret = hcp52c_ReadCtr(hDevID, 0, 1, &ctr1);

備　考	入力ソース カウンタ 1・・指令パルス出力 カウンタ 2・・エンコーダ入力 カウンタ 3・・偏差カウンタ(脱調検出用) カウンタ 4・・汎用カウンタ(指令パルス，エンコーダ入力等)

2.5 動作設定

2.5.1 hcp52c_SetFLSpd() ベース速度の設定

機　能	デバイスハンドルで指定された CPD の指定された軸のベース速度(pps)を速度倍率で除算した値を設定.

開発言語	書　式
VC++	DWORD hcp52c_SetFLSpd (DWORD *hDevID*, WORD *axis*, DWORD *speed*);
VB6	Public Function hcp52c_SetFLSpd(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByVal *speed* As Long) As Long
VB.NET	Public Function hcp52c_SetFLSpd(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByVal *speed* As Integer) As Integer
VC#	public static uint hcp52c_SetFLSpd(uint *hDevID*, ushort *axis*, uint *speed*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
speed	ベース速度レジスタ値(RFL)[1～65535, RFL＜RFH で設定]

VC++
記述例
	DWORD ret;//関数の戻り値
	ret = hcp52c_SetFLSpd(hDevID, 1, 500);　//Y1 軸を指定，RFL=500

備　考	ベース速度・・加減速動作時の立ち上がりの速度(本速度から加速，本速度まで減速して停止)

2.5.2 hcp52c_SetAuxSpd() 補助速度の設定

機　能	デバイスハンドルで指定された CPD の指定された軸の補助速度(pps)を速度倍率で除算した値を設定.

開発言語	書　式
VC++	DWORD hcp52c_SetAuxSpd(DWORD *hDevID*, WORD *axis*, DWORD *speed*);
VB6	Public Function hcp52c_SetAuxSpd(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByVal *speed* As Long) As Long
VB.NET	Public Function hcp52c_SetAuxSpd(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByVal *speed* As Integer) As Integer
VC#	public static uint hcp52c_SetAuxSpd(uint *hDevID*, ushort *axis*, uint *speed*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
speed	補助速度レジスタ値(RFA)[1～65535]

VC++
記述例
	DWORD ret;//関数の戻り値
	ret = hcp52c_SetAuxSpd(hDevID, 1, 500);　//Y1 軸を指定，RFA=500

備　考	補助速度・・一部の原点復帰において，原点突入速度等に使用されます.

2.5.3 hcp52c_SetAccRate() 加速レートの設定

機　能	デバイスハンドルで指定された CPD の指定された軸の加速レート(RUR)を設定します.

開発言語	書　式
VC++	DWORD hcp52c_SetAccRate(DWORD *hDevID*, WORD *axis*, DWORD *rate*);
VB6	Public Function hcp52c_SetAccRate(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByVal *rate* As Long) As Long
VB.NET	Public Function hcp52c_SetAccRate(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByVal *rate* As Integer) As Integer
VC#	public static uint hcp52c_SetAccRate(uint *hDevID*, ushort *axis*, uint *rate*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
rate	加速レート(RUR)[1～65535]

VC++
記述例
	DWORD ret; //関数の戻り値
	DWORD rur; //RUR
	// RUR 計算 RFH=5000, RFL=500, 加速時間=100msec(直線加速時)
	ret = hcp52c_CalAccRate(&rur, 100, 5000, 500, 0, 0);
	//Y1 軸に設定
	ret = hcp52c_SetAccRate(hDevID, 1, rur);

備　考	加速レート(RUR)と加速時間の関係 RFH:動作速度レジスタ RFL:ベース速度レジスタ RUR:加速レートレジスタ (1) 直線加速 $$加速時間[sec] = \frac{(RFH - RFL) \times (RUR + 1) \times 4}{19{,}660{,}800}$$ (2) 直線部分のない S 字加速 $$加速時間[sec] = \frac{(RFH - RFL) \times (RUR + 1) \times 8}{19{,}660{,}800}$$

2.5.4 hcp52c_SetDecRate() 減速レートの設定

機　能	デバイスハンドルで指定された CPD の指定された軸の減速レート(RDR)を設定します. 加速時間と減速時間が異なる場合に RDR の設定を行います.

開発言語	書　式
VC++	DWORD hcp52c_SetDecRate(DWORD *hDevID*, WORD *axis*, DWORD *rate*);
VB6	Public Function hcp52c_SetDecRate(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByVal *rate* As Long) As Long
VB.NET	Public Function hcp52c_SetDecRate(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByVal *rate* As Integer) As Integer
VC#	public static uint hcp52c_SetDecRate(uint *hDevID*, ushort *axis*, uint *rate*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
rate	減速レート(RDR)[0～65535] 0 を設定した場合は RUR=RDR として減速レートが決定されます. (加速時間＝減速時間)

VC++
記述例
	DWORD ret;　　//関数の戻り値
	DWORD rdr;　　//RDR
	// RDR 計算 RFH=5000, RFL=500, 減速時間=200msec(直線加速時)
	ret = hcp52c_CalAccRate(&rdr, 200, 5000, 500, 0, 0);
	//Y1 軸に設定
	ret = hcp52c_SetDecRate(hDevID, 1, rdr);

備　考	減速レート(RDR)と減速時間の関係 RFH:動作速度レジスタ RFL:ベース速度レジスタ RDR:減速レートレジスタ (1) 直線減速 $$加速時間[sec] = \frac{(RFH - RFL) \times (RDR + 1) \times 4}{19{,}660{,}800}$$ (2) 直線部分のない S 字減速 $$加速時間[sec] = \frac{(RFH - RFL) \times (RDR + 1) \times 8}{19{,}660{,}800}$$

2.5.5 hcp52c_SetMult() 速度倍率レジスタ値の設定

機　能	デバイスハンドルで指定された CPD の指定された軸の速度倍率レジスタ値(RMG)を設定します.

開発言語	書　式
VC++	DWORD hcp52c_SetMult(DWORD *hDevID*, WORD *axis*, DWORD *rmg*);
VB6	Public Function hcp52c_SetMult(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByVal *rmg* As Long) As Long
VB.NET	Public Function hcp52c_SetMult(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByVal *rmg* As Integer) As Integer
VC#	public static uint hcp52c_SetMult(uint *hDevID*, ushort *axis*, uint *rmg*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
rmg	速度倍率レジスタ値[2～4095]

VC++ 記述例	DWORD　ret;　　//関数の戻り値 ret = hcp52c_SetMult(hDevID, 1, 299);　　//Y1 軸を指定，速度倍率 1 倍

備　考

速度と速度倍率の関係及び速度倍率設定値と速度倍率の関係

RFx は速度レジスタ(RFH，RFL，RFA)の値

$$速度[PPS] = RFx \times 速度倍率 = \frac{RFx \times 300}{RMG+1} \qquad 速度倍率設定値(RMG) = \frac{300}{速度倍率} - 1$$

設定例

RMG(DEC)	RMG(HEX)	速度倍率	出力速度範囲(pps)
2999	bb7	0.1	0.1 ～ 6,553.5
1499	5db	0.2	0.2 ～ 13,107
599	257	0.5	0.5 ～ 32,767.5
299	12b	1	1 ～ 65,535
149	95	2	2 ～ 131,070
59	3b	5	5 ～ 327,675
29	1d	10	10 ～ 655,350
14	e	20	20 ～ 1,310,700
11	b	25	25 1,638,375
9	9	30	30 1,966,050
5	5	50	50 ～ 3,276,750
4	4	60	60 3,932,100
3	3	75	75 4,915,125
2	2	100	100 ～ 6,553,500

2.5.6 hcp52c_SetEventMask() イベントマスクの設定

機　能	デバイスハンドルで指定された CPD の指定された軸のイベントマスクを設定します.

開発言語	書　式
VC++	DWORD hcp52c_SetEventMask(DWORD *hDevID*, WORD *axis*, DWORD *mask*);
VB6	Public Function hcp52c_SetEventMask(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByVal *mask* As Long) As Long
VB.NET	Public Function hcp52c_SetEventMask(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByVal *mask* As Integer) As Integer
VC#	public static uint hcp52c_SetEventMask(uint *hDevID*, ushort *axis*, uint *mask*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
mask	イベントマスクデータ 00001h　正常停止時 00002h　次動作継続スタート時 00004h　動作用プリレジスタフルから空きができた時 00008h　コンパレータ 5 用プリレジスタフルから空きができた時 00010h　加速開始時 00020h　加速終了時 00040h　減速開始時 00080h　減速終了時 00100h　コンパレータ 1 条件成立時 00200h　コンパレータ 2 条件成立時 00400h　コンパレータ 3 条件成立時 00800h　コンパレータ 4 条件成立時 01000h　コンパレータ 5 条件成立時 02000h　CLR 入力によるカウント値のクリア時 04000h　LATCH 入力によるカウント値のラッチ時 08000h　OLS 入力によるカウント値のラッチ時 10000h　DLS 入力 ONN時 20000h　±DR 入力変化時 40000h　CSTA 信号入力 ON 時 上記データの OR したデータを与えることで複数のイベント報告指定.

VC++
記述例
	DWORD ret;　　//関数の戻り値
	//Y1 軸, 正常停止,コンパレータ 5 条件成立時にイベント報告
	ret = hcp52c_SetEventMask(hDevID, 1, 0x1001);

2.5.7 hcp52c_SetDecPoint() 減速開始点の設定

機　能	デバイスハンドルで指定された CPD の指定された軸の減速開始点を設定します.

開発言語	書　式
VC++	DWORD hcp52c_SetDecPoint(DWORD *hDevID*, WORD *axis*, long *dstnc*);
VB6	Public Function hcp52c_SetDecPoint(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByVal *dstnc* As Long) As Long
VB.NET	Public Function hcp52c_SetDecPoint(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByVal *dstnc* As Integer) As Integer
VC#	public static uint hcp52c_SetDecPoint(uint *hDevID*, ushort *axis*, int *dstnc*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
dstnc	減速開始点(pulse)

VC++
記述例
	DWORD　　ret; //関数の戻り値
	ret = hcp52c_SetDecPoint(hDevID, 1, 300);　　//Y1 軸を指定，減速開始点

備　考	残移動量が減速開始点以下になると減速を開始します. [減速開始点計算自動時] 自動で計算された減速開始点に対するオフセット値(単位:pulse)となります. ＋の値を設定すると減速が早めに開始され，減速後ベース速度で動作します. －の値を設定すると減速が遅めに開始され，ベース速度に到達する前に動作完了となります. 0 を設定すると通常の動作になります. [減速開始点計算手動時] 残移動量が設定した値(pulse)以下になると減速を開始します. この場合に設定する値は次式の様になります.（ベース速度到達時に動作完了になる値） (1) 直線減速 $最適値[パルス] = \frac{(RFH^2 - RFL^2) \times (RDR+1)}{(RMG+1) \times 32{,}768}$ (2) 直線部分のない S 字減速 $最適値[パルス] = \frac{(RFH^2 - RFL^2) \times (RDR+1) \times 2}{(RMG+1) \times 32{,}768}$

2.6 運用設定

2.6.1 hcp52c_WritOpeMode() 動作モードの設定

機　能	デバイスハンドルで指定された CPD の指定された軸の動作モードを設定します.

開発言語	書　式
VC++	DWORD hcp52c_WritOpeMode(DWORD *hDevID*, WORD *axis*, WORD *mode*);
VB6	Public Function hcp52c_WritOpeMode(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByVal *mode* As Integer) As Long
VB.NET	Public Function hcp52c_WritOpeMode(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByVal *mode* As Short) As Integer
VC#	public static uint hcp52c_WritOpeMode(uint *hDevID*, ushort *axis*, ushort *mode*);

引　数	説　明	
hDevID	デバイスハンドル	
axis	軸指定	
mode	動作モード	
	00h:コマンド制御による＋方向連続送り	08h:コマンド制御による－方向連続送り
	01h:パルサ入力による連続送り(※1)	02h:±DR 入力による連続送り(※2)
	10h:＋方向原点復帰動作	18h:－方向原点復帰動作
	12h:＋方向原点抜け出し動作	1ah:－方向原点抜け出し動作
	15h:＋方向原点サーチ動作	1dh:－方向原点サーチ動作
	20h:＋ELS 又は＋SLS 位置まで動作	28h:－ELS 又は-SLS 位置まで動作
	22h:＋ELS 又は+SLS 抜け出し動作	2ah:－ELS 又は-SLS 抜け出し動作
	24h:＋方向に Z 相カウント動作	2ch:－方向に Z 相カウント動作
	41h:位置決め動作	42h:PCS 位置決め動作(ライブラリ関数のみ)
	44h:指令位置0点復帰動作	45h:機械位置0点復帰動作(※1)
	46h:＋方向 1 パルス動作	4eh:－方向 1 パルス動作
	47h:タイマ動作	51h:パルサ入力による位置決め動作(※1)
	54h:パルサ入力指令位置 0 点復帰動作(※1)	55h:パルサ入力機械位置 0 点復帰動作(※1)
	56h:±DR 入力による位置決め動作(※2)	
	60h:連続直線補間動作	61h:直線補間動作
	64h:CW 方向円弧補間動作	65h:CCW 方向円弧補間動作

	DWORD ret;　　//関数の戻り値 ret = hcp52c_WritOpeMode(hDevID, 1, 0x41);　//Y1 軸を指定，位置決め動作

2.6.2 hcp52c_WritFHSpd() 動作速度の設定

機　能	デバイスハンドルで指定されたCPDの指定された軸の動作速度(pps)を速度倍率で除算した値(RFH)を設定します.

開発言語	書　式
VC++	DWORD hcp52c_WritFHSpd(DWORD *hDevID*, WORD *axis*, DWORD *speed*);
VB6	Public Function hcp52c_WritFHSpd(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByVal *speed* As Long) As Long
VB.NET	Public Function hcp52c_WritFHSpd(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByVal *speed* As Integer) As Integer
VC#	public static uint hcp52c_WritFHSpd(uint *hDevID*, ushort *axis*, uint *speed*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
speed	動作速度レジスタ値(RFH)[2～65535, RFL＜RFH で設定]

VC++ 記述例	DWORDret;　　//関数の戻り値 ret = hcp52c_WritFHSpd(hDevID, 1, 5000); //Y1 軸を指定，RFH=5000

2.6.3 hcp52c_WritPos() 位置決め移動量の設定

機　能	デバイスハンドルで指定された CPD の指定された軸の位置決め動作の移動量を設定します.

開発言語	書　式
VC++	DWORD hcp52c_WritPos(DWORD *hDevID*, WORD *axis*, long *dstnc*);
VB6	Public Function hcp52c_WritPos(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByVal *dstnc* As Long) As Long
VB.NET	Public Function hcp52c_WritPos(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByVal *dstnc* As Integer) As Integer
VC#	public static uint hcp52c_WritPos(uint *hDevID*, ushort *axis*, int *dstnc*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
dstnc	移動量(pulse)[-134,217,728～134,217,727(28 ﾋﾞｯﾄ)]

VC++ 記述例	DWORDret;　　//関数の戻り値 ret = hcp52c_WritPos(hDevID, 1, 15000);　//Y1 軸を指定，+方向 15000pulse

2.6.4 hcp52c_WritLine() 直線補間の移動量の設定

機　能	デバイスハンドルで指定された CPD の指定された軸の直線補間の移動量を設定します.

開発言語	書　式
VC++	DWORD hcp52c_WritLine(DWORD *hDevID*, WORD *axis*, long *dstnc*);
VB6	Public Function hcp52c_WritLine(ByVal *hDevID* As Long, ByVal *axis* As Integer, _ ByVal *dstnc* As Long) As Long
VB.NET	Public Function hcp52c_WritLine(ByVal *hDevID* As Integer, ByVal *axis* As Short, _ ByVal *dstnc* As Integer) As Integer
VC#	public static uint hcp52c_WritLine(uint *hDevID*, ushort *axis*, int *dstnc*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
dstnc	移動量(pulse)[-134,217,728～134,217,727(28 ﾋﾞｯﾄ)]

VC++ 記述例	DWORD ret;　　//関数の戻り値 ret = hcp52c_WritLine(hDevID, 0, -30000);　//補間代表軸：X1 軸を指定，ー方向 30000pulse ret = hcp52c_WritLine(hDevID, 1, -15000);　//従軸：Y1 軸を指定，ー方向 15000pulse

2.6.5 hcp52c_WritCircl() 円弧補間の移動量の設定

機　能	デバイスハンドルで指定された CPD の指定された軸の円弧補間の移動量を設定します. 終点位置及び中心位置 1, 2 の順番は X 軸に近い軸の順番になります. データの設定方法については「ユーザーズマニュアル<運用編>」を参照して下さい.

開発言語	書　式
VC++	DWORD hcp52c_WritCircl(DWORD *hDevID*,WORD *axis*,long *dstnc1*,long *dstnc2*, long *center1*,long *center2*);
VB6	Public Function hcp52c_WritCircl(_ ByVal *hDevID* As Long, ByVal *axis* As Integer, ByVal *dstnc1* As Long, ByVal *dstnc2* As Long, _ ByVal *center1* As Long, ByVal *center2* As Long) As Long
VB.NET	Public Function hcp52c_WritCircl(_ ByVal *hDevID* As Integer, ByVal *axis* As Short, ByVal *dstnc1* As Integer, _ ByVal *dstnc2* As Integer, ByVal *center1* As Integer, ByVal *center2* As Integer) As Integer
VC#	public static uint hcp52c_WritCircl(uint *hDevID*, ushort *axis* int *dstnc1*, int *dstnc2*, int *center1*, int *center2*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定 0:X1−Y1, 1:X1−Z1, 3:Y1−Z1, 4:Y1−U1, 5:Z1−U1, 6:X2−Y2, 7:X2−Z2, 8:X2−U2, 9:Y2−Z2, 10:Y2−U2, 11:Z2−U2, 12:X3−Y3, 13:X3−Z3, 14:X3−U3, 15:Y3−Z3, 16:Y3−U3, 17:Z3−U3
dstnc1	終点位置 1[-134,217,728～134,217,727(28 ﾋﾞｯﾄ)]
dstnc2	終点位置 2[-134,217,728～134,217,727(28 ﾋﾞｯﾄ)]
center1	中心位置 1[-134,217,728～134,217,727(28 ﾋﾞｯﾄ)]
center2	中心位置 2[-134,217,728～134,217,727(28 ﾋﾞｯﾄ)]

VC++
記述例
	DWORD ret;　　//関数の戻り値
	ret = hcp52c_WritCircl(hDevID, 0, 0, 0, 1000, 0);//X1Y1 軸円弧(半径 1000pulse の真円)
	//終点(X,Y)=(0,0),中心(X,Y)=(1000,0)

2.6.6 hcp52c_WritCtr() カウンタプリセット

機　能	デバイスハンドルで指定されたCPDの指定された軸の指定されたカウンタへプリセット(座標値の書込み)をします.

開発言語	書　式
VC++	DWORD hcp52c_WritCtr(DWORD *hDevID*, WORD *axis,* long *preset*, WORD *selctr*);
VB6	Public Function hcp52c_WritCtr(_ ByVal *hDevID* As Long, ByVal *axis* As Integer, ByVal *preset* As Long, ByVal *selctr* As Long) _ As Long
VB.NET	Public Function hcp52c_WritCtr(_ ByVal *hDevID* As Integer, ByVal *axis* As Short, ByVal *preset* As Integer, ByVal *selctr* As Integer) _ As Integer
VC#	public static uint hcp52c_WritCtr(uint *hDevID*, ushort *axis*, int nData, ushort *selctr*);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
preset,	プリセット値[-134,217,728～134,217,727(28 ﾋﾞｯﾄ)]
selctr	カウンタ選択[1:カウンタ 1, 2:カウンタ 2, 3:カウンタ 3, 4:カウンタ 4]

VC++ 記述例	DWORD ret;　　//関数の戻り値 ret = hcp52c_WritCtr(hDevID, 0, 5000, 1);　　//X1 軸，プリセット値=5000，カウンタ 1

2.7 動作制御指令

引数はデバイスハンドル及び軸指定となります．複数軸への同時指定は OR したデータになります.

2.7.1 hcp52c_DecStop() 減速停止

機　能	デバイスハンドルで指定された CPD の指定された軸を減速停止します.

開発言語	書　式
VC++	DWORD hcp52c_DecStop　(DWORD *hDevID*, WORD *axis*);
VB6	Public Function hcp52c_DecStop(ByVal *hDevID* As Long, ByVal *axis* As Integer) As Long
VB.NET	Public Function hcp52c_DecStop(ByVal *hDevID* As Integer, ByVal *axis* As Short) As Integer
VC#	public static uint hcp52c_DecStop(uint *hDevID*, ushort *axis*);

VC++ 記述例	DWORD ret;　　//関数の戻り値 ret = hcp52c_DecStop(hDevID, 0x07);　　//X1Y1Z1 軸減速停止

2.7.2 hcp52c_QuickStop() 即停止

機　能	デバイスハンドルで指定された CPD の指定された軸を即停止します.

開発言語	書　式
VC++	DWORD hcp52c_QuickStop(DWORD *hDevID*, WORD *axis*);
VB6	Public Function hcp52c_QuickStop(ByVal *hDevID* As Long, ByVal *axis* As Integer) As Long
VB.NET	Public Function hcp52c_QuickStop(ByVal *hDevID* As Integer, ByVal *axis* As Short) As Integer
VC#	public static uint hcp52c_QuickStop(uint *hDevID*, ushort *axis*);

VC++ 記述例	DWORD ret;　　//関数の戻り値 ret = hcp52c_QuickStop(hDevID, 0x07);　　//X1Y1Z1 軸即停止

2.7.3 hcp52c_SyDecStop() 複数軸同時減速停止

機　能	デバイスハンドルで指定された CPD の指定された軸を減速停止します.

開発言語	書　式
VC++	DWORD hcp52c_DecStop (DWORD *hDevID*, WORD *axis*);
VB6	Public Function hcp52c_DecStop(ByVal *hDevID* As Long, ByVal *axis* As Integer) As Long
VB.NET	Public Function hcp52c_DecStop(ByVal *hDevID* As Integer, ByVal *axis* As Short) As Integer
VC#	public static uint hcp52c_DecStop(uint *hDevID*, ushort *axis*);

VC++
记述例
	DWORD ret; //関数の戻り値
	ret = hcp52c_DecStop(hDevID, 0x0FFF); //全軸減速停止

2.7.4 hcp52c_SyQuickStop() 複数軸同時即停止

機　能	デバイスハンドルで指定された CPD の指定された軸を即停止します.

開発言語	書　式
VC++	DWORD hcp52c_QuickStop(DWORD *hDevID*, WORD *axis*);
VB6	Public Function hcp52c_QuickStop(ByVal *hDevID* As Long, ByVal *axis* As Integer) As Long
VB.NET	Public Function hcp52c_QuickStop(ByVal *hDevID* As Integer, ByVal *axis* As Short) As Integer
VC#	public static uint hcp52c_QuickStop(uint *hDevID*, ushort *axis*);

VC++
记述例
	DWORD ret; //関数の戻り値
	ret = hcp52c_QuickStop(hDevID, 0x0FFF); //全軸即停止

2.7.5 hcp52c_EmgStop() 非常停止

機　能	デバイスハンドルで指定された CPD の指定された軸を非常停止します.

開発言語	書　式
VC++	DWORD hcp52c_EmgStop (DWORD *hDevID*, WORD *axis*);
VB6	Public Function hcp52c_EmgStop(ByVal *hDevID* As Long, ByVal *axis* As Integer) As Long
VB.NET	Public Function hcp52c_EmgStop(ByVal *hDevID* As Integer, ByVal *axis* As Short) As Integer
VC#	public static uint hcp52c_EmgStop(uint *hDevID*, ushort *axis*);

VC++
记述例
	DWORD ret; //関数の戻り値
	ret = hcp52c_EmgStop(hDevID, 0x07); //X1Y1Z1 軸非常停止

2.7.6 hcp52c_AccStart() 加速スタート

機　能	デバイスハンドルで指定された CPD の指定された軸を加速スタートします.

開発言語	書　式
VC++	DWORD hcp52c_AccStart (DWORD *hDevID*, WORD *axis*);
VB6	Public Function hcp52c_AccStart (ByVal *hDevID* As Long, ByVal *axis* As Integer) As Long
VB.NET	Public Function hcp52c_ AccStart (ByVal *hDevID* As Integer, ByVal *axis* As Short) As Integer
VC#	public static uint hcp52c_ AccStart (uint *hDevID*, ushort *axis*);

VC++
记述例
	DWORD ret; //関数の戻り値
	ret = hcp52c_AccStart(hDevID, 0x07); //X1Y1Z1 軸加速スタート

2.7.7 hcp52c_ CnstStartFH () FH 定速スタート

機　能	デバイスハンドルで指定された CPD の指定された軸をFH定速スタートします.

開発言語	書　式
VC++	DWORD hcp52c_CnstStartFH　(DWORD *hDevID*, WORD *axis*);
VB6	Public Function hcp52c_CnstStartFH (_ ByVal *hDevID* As Long, ByVal *axis* As Integer) As Long
VB.NET	Public Function hcp52c_CnstStartFH (_ ByVal *hDevID* As Integer, ByVal *axis* As Short) As Integer
VC#	public static uint hcp52c_CnstStartFH (uint *hDevID*, ushort *axis*);

VC++ 記述例	DWORD ret;　　//関数の戻り値 ret = hcp52c_ CnstStartFH (hDevID, 0x07);　　//X1Y1Z1 軸 FH 定速スタート

2.7.8 hcp52c_ CnstStartFL () FL 定速スタート

機　能	デバイスハンドルで指定された CPD の指定された軸をFL 定速スタートします.

開発言語	書　式
VC++	DWORD hcp52c_CnstStartFL　(DWORD *hDevID*, WORD *axis*);
VB6	Public Function hcp52c_CnstStartFL (_ ByVal *hDevID* As Long, ByVal *axis* As Integer) As Long
VB.NET	Public Function hcp52c_CnstStartFL (_ ByVal *hDevID* As Integer, ByVal *axis* As Short) As Integer
VC#	public static uint hcp52c_CnstStartFL (uint *hDevID*, ushort *axis*);

VC++ 記述例	DWORD ret;　　//関数の戻り値 ret = hcp52c_ CnstStartFL (hDevID, 0x07);　　//X1Y1Z1 軸 FL 定速スタート

2.7.9 hcp52c_ CnstStartByDec () FH定速スタート後減速停止

機　能	デバイスハンドルで指定された CPD の指定された軸をFH定速スタート後減速停止します.

開発言語	書　式
VC++	DWORD hcp52c_CnstStartByDec(DWORD *hDevID*, WORD *axis*);
VB6	Public Function hcp52c_CnstStartByDec (_ ByVal *hDevID* As Long, ByVal *axis* As Integer) As Long
VB.NET	Public Function hcp52c_CnstStartByDec (_ ByVal *hDevID* As Integer, ByVal *axis* As Short) As Integer
VC#	public static uint hcp52c_CnstStartByDec (uint *hDevID*, ushort *axis*);

VC++ 記述例	DWORD ret;　　//関数の戻り値 ret = hcp52c_ CnstStartByDec(hDevID, 0x07);　　//X1Y1Z1 軸FH定速スタート後減速停止

2.7.10 hcp52c_SvOn()　SVON オン

機　能	デバイスハンドルで指定された CPD の指定された軸の SVON をオンします.

開発言語	書　式
VC++	DWORD hcp52c_SvOn (DWORD *hDevID*, WORD *axis*);
VB6	Public Function hcp52c_SvOn (ByVal *hDevID* As Long, ByVal *axis* As Integer) As Long
VB.NET	Public Function hcp52c_SvOn (ByVal *hDevID* As Integer, ByVal *axis* As Short) As Integer
VC#	public static uint hcp52c_SvOn (uint *hDevID*, ushort *axis*);

VC++ 記述例	DWORD ret;　　//関数の戻り値 ret = hcp52c_SvOn(hDevID, 0x07);　　//X1Y1Z1 軸サーボオン

備考	SVON は X1,Y1,Z1,U1 のみあります.

2.7.11 hcp52c_SvOff()　SVON オフ

機　能	デバイスハンドルで指定された CPD の指定された軸の SVON をオフします.

開発言語	書　式
VC++	DWORD hcp52c_SvOff　(DWORD *hDevID*, WORD *axis*);
VB6	Public Function hcp52c_SvOff (ByVal *hDevID* As Long, ByVal *axis* As Integer) As Long
VB.NET	Public Function hcp52c_SvOff (ByVal *hDevID* As Integer, ByVal *axis* As Short) As Integer
VC#	public static uint hcp52c_SvOff (uint *hDevID*, ushort *axis*);

VC++ 記述例	DWORD ret;　　//関数の戻り値 ret = hcp52c_SvOff(hDevID, 0x07);　　//X1Y1Z1 軸 SVON オフ

備考	SVON は X1,Y1,Z1,U1 のみあります.

2.7.12 hcp52c_SvResetOn()　SVRST オン

機　能	デバイスハンドルで指定された CPD の指定された軸の SVRST をオンします.

開発言語	書　式
VC++	DWORD hcp52c_SvResetOn (DWORD *hDevID*, WORD *axis*);
VB6	Public Function hcp52c_SvResetOn (ByVal *hDevID* As Long, ByVal *axis* As Integer) As Long
VB.NET	Public Function hcp52c_SvResetOn (ByVal *hDevID* As Integer, ByVal *axis* As Short) As Integer
VC#	public static uint hcp52c_SvResetOn (uint *hDevID*, ushort *axis*);

VC++ 記述例	DWORD ret;　　//関数の戻り値 ret = hcp52c_SvResetOn(hDevID, 0x07);　　//X1Y1Z1 軸 SVRST オン

備考	SVRST は X1,Y1,Z1,U1 のみあります.

2.7.13 hcp52c_SvResetOff()　SVRST オフ

機　能	デバイスハンドルで指定された CPD の指定された軸の SVRST をオフします.

開発言語	書　式
VC++	DWORD hcp52c_SvResetOff(DWORD *hDevID*, WORD *axis*);
VB6	Public Function hcp52c_SvResetOff (ByVal *hDevID* As Long, ByVal *axis* As Integer) As Long
VB.NET	Public Function hcp52c_SvResetOff (ByVal *hDevID* As Integer, ByVal *axis* As Short) As Integer
VC#	public static uint hcp52c_SvResetOff (uint *hDevID*, ushort *axis*);

VC++ 記述例	DWORD ret;　　//関数の戻り値 ret = hcp52c_SvResetOff(hDevID, 0x07);　　//X1Y1Z1 軸 SVRST オフ

備考	SVRST は X1,Y1,Z1,U1 のみあります.

2.7.14 hcp52c_PMOn()　パルスモータ励磁オン

機　能	デバイスハンドルで指定された CPD の指定された軸のパルスモータ励磁をオンします.

開発言語	書　式
VC++	DWORD hcp52c_PMOn (DWORD *hDevID*, WORD *axis*);
VB6	Public Function hcp52c_PMOn (ByVal *hDevID* As Long, ByVal *axis* As Integer) As Long
VB.NET	Public Function hcp52c_PMOn (ByVal *hDevID* As Integer, ByVal *axis* As Short) As Integer
VC#	public static uint hcp52c_PMOn (uint *hDevID*, ushort *axis*);

VC++ 記述例	DWORD ret;　　//関数の戻り値 ret = hcp52c_PMOn(hDevID, 0x07);　　//X1Y1Z1 軸励磁オン

備　考	SVON をパルスモータドライバのモーターフリー(出力電流オフ)信号に接続した場合に使用可能.

2.7.15 hcp52c_PMOff()　パルスモータ励磁オフ

機　能	デバイスハンドルで指定された CPD の指定された軸のパルスモータ励磁オフ(モーターフリー)にします.

開発言語	書　式
VC++	DWORD hcp52c_PMOff (DWORD *hDevID*, WORD *axis*);
VB6	Public Function hcp52c_PMOff (ByVal *hDevID* As Long, ByVal *axis* As Integer) As Long
VB.NET	Public Function hcp52c_PMOff (ByVal *hDevID* As Integer, ByVal *axis* As Short) As Integer
VC#	public static uint hcp52c_PMOff (uint *hDevID*, ushort *axis*);

VC++ 記述例	DWORD ret;　　//関数の戻り値 ret = hcp52c_PMOff(hDevID, 0x07);　　//X1Y1Z1 軸パルスモータ励磁オフ

備　考	SVON をパルスモータドライバのモーターフリー(出力電流オフ)信号に接続した場合に使用可能.

2.8 加減速レートの計算

2.8.1 hcp52c_CalAccRate() 加減速レートの計算

機　能	加減速時間, RFH, RFL 等より加減速レート(RUR, RDR)を計算します.

開発言語	書　式
VC++	DWORD hcp52c_CalAccRate(DWORD* *r*, DWORD *t*, DWORD *fh*, DWORD *fl*, WORD *p*, WORD *s*);
VB6	Public Function hcp52c_CalAccRate(_ ByRef *r* As Long, ByVal *t* As Long, ByVal *fh* As Long, ByVal *fl* As Long, ByVal *p* As Integer, _ ByVal *s* As Integer) As Long
VB.NET	Public Function hcp52c_CalAccRate(_ ByRef *r* As Integer, ByVal *t* As Integer, ByVal *fh* As Integer, ByVal *fl* As Integer, _ ByVal *p* As Short, ByVal *s* As Short) As Integer
VC#	public static uint hcp52c_CalAccRate(ref uint *r*, uint *t*, uint *fh*, uint *fl*, ushort *p*, ushort *s*);

引　数	説　明
r	計算結果(加減速レート[1～65535])
t	加減速時間(msec)
fh	FH レジスタ値[1～65535]
fl	FL レジスタ値[1～65535]
p	加減速形式[0:直線, 1:S 字]
s	予約:常に"0"を設定してください.

VC++
記述例
	DWORD ret;　　//関数の戻り値
	DWORD rate;　　//加減速レート
	　　//加減速時間 200ms, RFH=10000, RFL=500, 直線加減速
	　　ret = hcp52c_CalAccRate(&rate, 200, 10000, 500, 0, 0);
	　　//加速レート設定
	　　ret = hcp52c_SetAccRate(hDevID, 0, rate);

3. ドライバ関数

ドライバ関数には，CPD の制御を行うための関数が含まれます.
各関数は"VC++(6.0 以上)"，"VB(5.0/6.0/.NET2002/ .NET2003/2005/2008)"，VC#(.NET2003/2005/2008)から外部関数として起動されます.

3.1 関数の種類

No.	関数名	機 能
1	cp52c_GetDeviceCount()	ボード枚数の取得
2	cp52c_GetDeviceInfo()	デバイス情報の取得
3	cp52c_OpenDevice()	デバイスのオープン
4	cp52c_CloseDevice()	デバイスのクローズ
5	cp52c_rMstsW()	メインステータスの読出し
6	cp52c_rSstsW()	サブステータスの読出し
7	cp52c_wCmdW()	制御コマンドの書込
8	cp52c_rReg()	レジスタの読出し
9	cp52c_wReg()	レジスタへ書込
10	cp52c_rPortB()	オプションポートの読出し(1 バイト)
11	cp52c_wPortB()	オプションポートへ書込(1 バイト)
12	cp52c_rPortW()	オプションポートの読出し(2 バイト)
13	cp52c_wPortW()	オプションポートへ書込(2 バイト)
14	cp52c_rBufDW()	入出力バッファの読出し
15	cp52c_wBufDW()	入出力バッファへ書込

表 3.1-1 ドライバ関数一覧

3.2 関数の詳細

3.2.1 cp52c_GetDeviceCount() ボード枚数の取得

機 能	現在パソコンに装着されている CPD の枚数を取得します.

開発環境	書 式
VC++	DWORD WINAPI cpxxx_GetDeviceCount(DWORD* count);
VB6	Declare Function cpxxx_GetDeviceCount Lib "*hicpd52c.dll*" _ (ByRef count As Long) As Long
VB.NET	Declare Function cpxxx_GetDeviceCount Lib "*hicpd52c.dll*" _ (ByRef count As Integer) As Integer
VC#	[DllImport("*hicpd52c.dll* ")] public static extern uint cpxxx_GetDeviceCount(ref uint count);

引 数	説 明
count	取得した CPD 枚数

VC++
記述例
	DWORD* count; //CPD の枚数
	DWORD ret; //関数の戻り値
	ret = cpxxx_GetDeviceCount(&count);

3.2.2 cp52c_GetDeviceInfo()　　デバイス情報の取得

機　能	現在パソコンに装着されている指定枚数 CPD のデバイス情報を取得します. この結果，デバイス情報構造体の配列にデバイス情報が格納されます. このデバイス情報は，デバイスオープン時に利用します.

開発環境	書　式
VC++	DWORD WINAPI cpxxx_GetDeviceInfo(DWORD*count, HCP52CINFO * hInfo);
VB6	Declare Function cpxxx_GetDeviceInfo Lib "hicpd52c.dll" (ByRef count As Long, _ hInfo As HCP52CINFO) As Long
VB.NET	Declare Function cpxxx_GetDeviceInfo Lib "hicpd52c.dll" (ByRef count As Integer, _ ByRef hInfo As HCP52CINFO) As Integer
VC#	[DllImport("*hicpd52c.dll* ")] public static extern uint cpxxx_ GetDeviceInfo (ref uint count, ref HCP52CINFO hInfo);

引　数	説　明
count	CPD 枚数
hInfo	取得するデバイス情報

VC++ 記述例	DWORD ret;　　//関数の戻り値 DWORD count=2;　　//最大枚数は2 HCP52CINFO HpcDevInfo[2];　//2 枚の CPD のデバイス情報格納配列 ret = cp52c_GetDeviceInfo(&count, &HpcDevInfo[0]);

備　考	デバイス情報を格納するエリアはボード枚数分を確保します.

3.2.3 cp52c_OpenDevice()　　デバイスのオープン

機　能	渡したデバイス情報を持つ CPD をオープンし，他と識別するためのデバイスハンドルを取得します． 以降このデバイスハンドルは，この CPD にアクセスするためのハンドルとなります．

開発環境	書　式
VC++	DWORD WINAPI cpxxx_OpenDevice(DWORD* hDevID, HCP52CINFO *hInfo);
VB6	Declare Function cpxxx_OpenDevice Lib "hicpd52c.dll"(ByRef hDevID As Long, _ hInfo As HCP52CINFO) As Long
VB.NET	Declare Function cpxxx_OpenDevice Lib "hicpd52c.dll"(ByRef hDevID As Integer, _ ByRef hInfo As HCP52CINFO) As Integer
VC#	[DllImport("*hicpd52c.dll* ")] public static extern uint cpxxx_ OpenDevice (ref uint hDevID, ref HCP52CINFO hInfo);

引　数	説　明
hDevID	デバイスハンドル
hInfo	デバイス情報

VC++ 記述例	DWORD ret;　　//関数の戻り値 DWORD count=2;　　//最大枚数は2 HCP52CINFO HpcDevInfo[2];　//2 枚の CPD のデバイス情報格納配列 ret = cp52c_GetDeviceInfo(&count, &HpcDevInfo[0]);

3.2.4 cp52c_CloseDevice()　　デバイスのクローズ

機　能	デバイスハンドルで指定された CPD をクローズします． 以降このデバイスハンドルは，無効となり，この CPD にアクセスはできません．

開発環境	書　式
VC++	DWORD WINAPI cpxxx_CloseDevice(DWORD hDevID);
VB6	Declare Function cpxxx_CloseDevice Lib "hicpd52c.dll"(ByVal hDevID As Long) As Long
VB.NET	Declare Function cpxxx_CloseDevice Lib "hicpd52c.dll"(ByVal hDevID As Integer) As Integer
VC#	[DllImport("*hicpd52c.dll* ")] public static extern uint cpxxx_ CloseDevice(uint hDevID);

引　数	説　明
hDevID	デバイスハンドル

VC++ 記述例	DWORD ret;　　//関数の戻り値 ret = cpxxx_CloseDevice(hDevID);

備　考	デバイスクローズの前に必ずパルス停止などの終了処理をしてください．

3.2.5 cp52c_rMstsW()　　メインステータスの読出し

機　能	デバイスハンドルで指定されたボードの指定軸メインステータスを読出します.

開発環境	書　式
VC++	DWORD WINAPI cpxxx_rMstsW(DWORD hDevID, WORD axis, WORD* wMsts);
VB6	Declare Function cpxxx_rMstsW Lib "hicpd52c.dll"(ByVal hDevID As Long, _ ByVal axis As Integer, ByRef wMsts As Integer) As Long
VB.NET	Declare Function cpxxx_rMstsW Lib "hicpd52c.dll"(ByVal hDevID As Integer, _ ByVal axis As Short, ByRef wMsts As Short) As Integer
VC#	[DllImport("*hicpd52c.dll* ")] public static extern uint cpxxx_ rMstsW (uint hDevID, ushort axis, ref ushort wData);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
wMsts	メインステータス

VC++ 記述例	DWORD ret;　　//関数の戻り値 WORD　wMsts;　//メインステータス ret = cpxxx_rMstsW(hDevID, 1, & wMsts);　//Y 軸

備　考	ステータスの監視は常に対象となる軸の MSTS に対して行います. スタートコマンドの発行後は MSTS を常にポーリングします. 移動終了は b5(SINT)及び b4(SERR)を監視し，これらが'0'の場合は移動中です. 移動が終了すると正常終了では b5(SINT)が'1'となり，**イベントステータス**を読み出します. (*1) 異常終了では b4(SERR)が'1'となり，**エラーステータス**の処理をします.

(1) メインステータスの内容

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPDF	SPRF	SEOR	SCMP5	SCMP4	SCMP3	SCMP2	SCMP1	SSC1	SSC0	SINT	SERR	SEND	0	SRUN	SSCM

bit	名　称	内　容	備考
0	SSCM	1 =　スタートコマンド書き込み済み	スタートコマンド書き込み後，即動作が始まらない場合があります. (条件付スタートのスタート保留状態など)
1	SRUN	1 = RUN 中	
2	---	予約	
3	SEND	1　＝停止中(電源投入直後は'0')	本ビットの変化タイミングは，INPOS 制御設定により異なります.
4	SERR	1　＝エラーステータスに要因有	REST 読み出しで 1→0
5	SINT	1　＝イベントステータスに要因有	RIST 読み出しで 1→0　イベントマスク(RIRQ)の設定が必要.
7,6	SSC1,0	実行中の RMD.b17,16	次動作連続実行中の動作確認などに使用
8	SCMP1	1 = CMP1 比較条件成立状態	比較条件成立中の間のみ'1'となります. (既に通過してしまった場合は'0'となります) 条件が成立したことを監視する場合はイベントステータス (RIST)を使用します.
9	SCMP2	1 = CMP2 比較条件成立状態	
10	SCMP3	1 = CMP3 比較条件成立状態	
11	SCMP4	1 = CMP4 比較条件成立状態	
12	SCMP5	1 = CMP5 比較条件成立状態	
13	SEOR	1 =　位置オーバライド失敗	本ステータスリードでクリア. 停止中に RMV を書込んでも"1".
14	SPRF	1 =　動作用プリレジスタフル	
15	SPDF	1 =　コンパレータ用プリレジスタフル	

*1. 正常終了でのイベント報告は"イベントマスク設定:自動停止"[RIRQ.b0:ISEN=1]とします.

*2. b3(SEND)は状態を示しているビットです.
　電源投入直後は '0' であり，即(減速)停止指令の実行または一度移動実行後の終了状態は'1'となります.
　移動中は '0 'を示します. 通常停止中(= '1')か，動作中(= '0')かを確認したいときに使用します.

*3. b13(SEOR)は HPCI-CPD574N，HPCI-CPD578N，HPCIe-CPD674N のみ対応しています.

3.2.6 cp52c_rSstsW()　　サブステータスの読出し

機　能	デバイスハンドルで指定されたボードの指定軸サブステータスを読出します.

開発環境	書　式
VC++	DWORD WINAPI cpxxx_rSstsW(DWORD hDevID, WORD axis, WORD* wSsts);
VB6	Declare Function cpxxx_rSstsW Lib "hicpd52c.dll"(ByVal hDevID As Long, _ ByVal axis As Integer, ByRef wSsts As Integer) As Long
VB.NET	Declare Function cpxxx_rSstsW Lib "hicpd52c.dll"(ByVal hDevID As Integer, _ ByVal axis As Short, ByRef wSsts As Short) As Integer
VC#	[DllImport("*hicpd52c.dll* ")] public static extern uint cpxxx_ rSstsW (uint hDevID, ushort axis, ref ushort wData);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
wSsts	サブステータス

VC++ 記述例	DWORD ret;　　//関数の戻り値 WORD　wSsts;　//サブステータス ret = cp52c_rSstsW(hDevID, 1, & wSsts);　　//Y 軸

備　考	1. b0(SVON), ビット 1(SVRS)はサーボ出力指令のモニタビットです. 2. b12(SALM)が'1'の場合, 動作方向のELS検出中(SPEL,SMEL)の軸に対する動作指令は異常終了となります. 3. 両方向のELSが共に検出中(SPEL=SMEL='1')となる場合は, オプションポートのELS極性の設定(A接/B接)が逆となっている事が考えられます.

(1) サブステータスの内容

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
SDLS	**SOLS**	**SMEL**	**SPEL**	**SALM**	SFC	SFD	SFU	0	0	0	0	0	0	**SVRS**	**SVON**

bit	名称	説　明	備　考
0	SVON	'1' = SVON ON	指令状態のモニタ
1	SVRST	'1' = SVRST ON	
7-2	---	不使用(予約)	
8	SFU	'1' = 加速中	レベル検出
9	SFD	'1' = 減速中	
10	SFC	'1' = 定速動作中	
11	SALM	'1' = SVALM 検出中	入力極性設定が反映される. A 接設定時はカプラ電流 ON で 1 B 接設定時はカプラ電流 OFF で 1
12	SPEL	'1' =＋ELS 検出中	
13	SMEL	'1' =－ELS 検出中	
14	SOLS	'1' = OLS 検出中	
15	SDLS	'1' = DLS 検出中	

3.2.7 cp52c_wCmdW()　　制御コマンド書込み

機　能	デバイスハンドルで指定されたボードの指定軸のコマンドバッファへ制御コマンドデータを書込みます.

開発環境	書　式
VC++	DWORD WINAPI cpxxx_wCmdW(DWORD hDevID, WORD axis, WORD wCmd);
VB6	Declare Function cpxxx_wCmdW Lib "hicpd52c.dll"(ByVal hDevID As Long, _ ByVal axis As Integer, ByVal wCmd As Integer) As Long
VB.NET	Declare Function cpxxx_wCmdW Lib "hicpd52c.dll"(ByVal hDevID As Integer, _ ByVal axis As Short, ByVal wCmd As Short) As Integer
VC#	[DllImport("*hicpd52c.dll* ")] public static extern uint cpxxx_ wCmdW (uint hDevID, ushort axis, ushort wCmd);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
wCmd	コマンドデータ

VC++ 記述例	DWORD ret;　　//関数の戻り値 ret = cp52c_wCmdW(hDevID, 1, 0x4a);　　//Y 軸減速停止

(1) コマンドデータについて

(1-1)コマンドデータのビット構成

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					実行	軸の	指定	(SELx)								
	0	0	0	0	U B	Z A	Y W	X V	コ　マ　ン　ド							

- 実行軸の指定(SELx)
  個々の軸にコマンドを書込む場合はこの 4 ビットは全て"0"にします.
  2~4 軸に同一コマンドを書込む場合(同時にスタート，停止など)は，SELx のビットで書込む軸を指定し，軸指定した PCL の任意の軸に書き込みます.

(1-2)スタートコマンド

No.	名称	コマンド (HEX)	備考
1	加速スタート	53	加速→FH 定速→減速
2	FH 定速スタート	51	
3	FH 定速スタート後減速停止	52	FH 定速→減速
4	FL 定速スタート	50	
5	残量 FL 定速スタート	54	
6	残量 FH 定速スタート	55	
7	残量加速スタート	57	
8	残量 FH 定速スタート後減速停止	56	
9	同時スタート信号(STA)出力	06	約 407nsec のワンショット出力

(1-3)速度変更コマンド

No.	名称	コマンド (HEX)	備考
1	FL 定速瞬時速度変更	40	
2	FH 定速瞬時速度変更	41	
3	FL 速度まで減速	42	
4	FH 速度まで加速	43	

(1-4)停止コマンド

No.	名称	コマンド (HEX)	備考
1	即停止	49	
2	減速停止	4A	ベース速度動作中は即停止. その他動作中は減速停止.
3	非常停止	05	エラー停止(EMG 入力による停止と同等)
4	同時停止信号(STP)出力	07	約 407nsec のワンショット出力

(1-5)コントロールコマンド

No.	名称	コマンド (HEX)	備考
1	ソフトウェアリセット	04	任意の軸への書き込みで 4 軸全てのレジスタがリセット(POWON 状態)になります. オプションポートはリセットされません.
2	CTR1 リセット	20	
3	CTR2 リセット	21	
4	CTR3 リセット	22	
5	CTR4 リセット	23	
6	SVCTRCL 信号出力	24	
7	SVCTRCL 信号リセット	25	SVCTRCL 出力幅レベル時の SVCTRCL 出力リセット
8	動作用プリレジスタキャンセル	26	
9	CMP5 用プリレジスタキャンセル	27	
10	動作用プリレジスタシフト	2B	
11	CMP5 用プリレジスタシフト	2C	
12	カウンタラッチ	29	CTR1～CTR4 を一斉にラッチ
13	PCS コマンド	28	PCS 信号の代わりに PCS を実行
14	SVON 信号 ON	18	
15	SVON 信号 OFF	10	
16	SVRST 信号 ON	19	
17	SVRST 信号 OFF	11	

3.2.8 cp52c_rReg()　　レジスタ読出し

機　能	デバイスハンドルで指定されたボードの，指定軸の指定レジスタ内容を読出します.

開発環境	書　式
VC++	DWORD WINAPI cpxxx_rReg DWORD hDevID, WORD axis, BYTE byCmd, DWORD* dwData);
VB6	Declare Function cpxxx_rReg Lib "hicpd52c.dll"(ByVal hDevID As Long, _ ByVal axis As Integer, ByVal byCmd As Byte, ByRef dwData As Long) As Long
VB.NET	Declare Function cpxxx_rReg Lib "hicpd52c.dll"(ByVal hDevID As Integer, _ ByVal axis As Short, ByVal byCmd As Byte, ByRef dwData As Integer) As Integer
VC#	[DllImport("*hicpd52c.dll* ")] public static extern uint cpxxx_ rReg (uint hDevID, ushort axis, byte byCmd, ref uint dwData);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
byCmd	レジスタ読出コマンド
dwData	レジスタ読出データ

VC++ 記述例	DWORD ret;　　//関数の戻り値 DWORD dwData;　　//レジスタのデータ ret = cpxxx_rReg(hDevID, 0, 0xc0, &dwData); //X 軸 PRMV 読出

3.2.9 cp52c_wReg()　　レジスタ書込み

機　能	デバイスハンドルで指定されたボードの，指定軸の指定レジスタへ，データを書込みます.

開発環境	書　式
VC++	DWORD WINAPI cpxxx_wReg DWORD hDevID, WORD axis, BYTE byCmd, DWORD dwData);
VB6	Declare Function cpxxx_wReg Lib "hicpd52c.dll"(ByVal hDevID As Long, _ ByVal axis As Integer, ByVal byCmd As Byte, ByVal dwData As Long) As Long
VB.NET	Declare Function cpxxx_wReg Lib "hicpd52c.dll"(ByVal hDevID As Integer, _ ByVal axis As Short, ByVal byCmd As Byte, ByVal dwData As Integer) As Integer
VC#	[DllImport("*hicpd52c.dll* ")] public static extern uint cpxxx_wReg (uint hDevID, ushort axis, byte byCmd, uint dwData);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
byCmd	レジスタ書込コマンド
dwData	レジスタ書込データ

VC++ 記述例	DWORD ret;　　//関数の戻り値 ret = cpxxx_wReg(hDevID, 0, 0x80, 10000);　　//X 軸 PRMV=10000 書込

3.2.10 レジスタ制御コマンド・内容

(1) レジスタ・プリレジスタ:読出コマンド・書込コマンド一覧

No	内容	レジスタ			プリレジスタ		
		名称	コマンド(HEX)		名称	コマンド(HEX)	
			読出	書込		読出	書込
1	移動量	RMV	d0	90	PRMV	c0	80
2	ベース速度設定	RFL	d1	91	PRFL	c1	81
3	動作速度設定	RFH	d2	92	PRFH	c2	82
4	加速レートを決定するパラメータ	RUR	d3	93	PRUR	c3	83
5	減速レートを決定するパラメータ	RDR	d4	94	PRDR	c4	84
6	速度倍率設定	RMG	d5	95	PRMG	c5	85
7	減速開始点	RDP	d6	96	PRDP	c6	86
8	動作モード	RMD	d7	97	PRMD	c7	87
9	円弧補間中心位置	RIP	d8	98	PRIP	c8	88
10	加速時 S 字区間設定	RUS	d9	99	PRUS	c9	89
11	減速時 S 字区間設定	RDS	da	9a	PRDS	ca	8a
12	補助速度設定	RFA	db	9b			
13	環境設定 1	RENV1	dc	9c			
14	環境設定 2	RENV2	dd	9d			
15	環境設定 3	RENV3	de	9e			
16	環境設定 4	RENV4	df	9f			
17	環境設定 5	RENV5	e0	a0			
18	環境設定 6	RENV6	e1	a1			
19	環境設定 7	RENV7	e2	a2			
20	カウンタ 1(指令パルス出力)	RCTR1	e3	a3			
21	カウンタ 2(エンコーダ入力)	RCTR2	e4	a4			
22	カウンタ 3(偏差カウンタ)	RCTR3	e5	a5			
23	カウンタ 4(汎用カウンタ)	RCTR4	e6	a6			
24	コンパレータ 1 用データ	RCMP1	e7	a7			
25	コンパレータ 2 用データ	RCMP2	e8	a8			
26	コンパレータ 3 用データ	RCMP3	e9	a9			
27	コンパレータ 4 用データ	RCMP4	ea	aa			
28	コンパレータ 5 用データ	RCMP5	eb	ab	PRCP5	cb	8b
29	イベントマスク設定	RIRQ	ec	ac			
30	カウンタ 1 ラッチデータ	RLTC1	ed				
31	カウンタ 2 ラッチデータ	RLTC2	ee				
32	カウンタ 3 ラッチデータ	RLTC3	ef				
33	カウンタ 4 ラッチデータ	RLTC4	f0				
34	拡張ステータス	RSTS	f1				
35	エラーステータス	REST	f2				
36	イベントステータス	RIST	f3				
37	位置決めカウンタ	RPLS	f4				
38	Z 相カウンタ，指令速度モニタ	RSPD	f5				
39	減速開始点計算値	RSDC	f6				
40	円弧補間歩進数	RCI	fc	bc	PRCI	cc	8c
41	円弧補間歩進カウンタ	RCIC	fd				
42	補間ステータス	RIPS	ff				

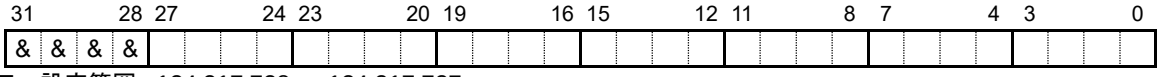
*1. カウンタ 1＝指令現在位置，カウンタ 2＝エンコーダフィードバック現在位置
*2. 動作速度のモニタ(下位 16 ビット)・・［速度倍率を乗算して実速度］
*3. メインステータス:SINT=1 とするイベントを設定
*4. 動作終了時のイベント発生内容確認ステータス(メインステータス:SINT=1)
*5. 動作終了時のエラー終了内容の確認ステータス(メインステータス:SERR=1)

(2) レジスタ・プリレジスタデータ内容

(2-1) 動作用レジスタ

(2-1-1) RMV(PRMV)移動量レジスタ　Read:d0(c0), Write:90(80) ･･･(括弧内はプリレジスタ)

補間動作，位置決め動作において移動量を相対位置で設定します．



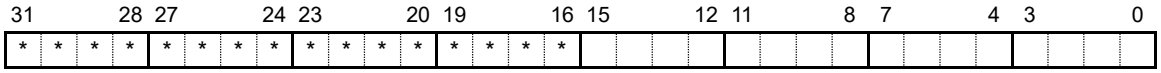
■ 設定範囲 -134,217,728～+134,217,727

■ 書込みはプリレジスタ PRMV に書込む．

■ &表記のビットは，書込み時には無視され，読出し時には空欄表示の最上位ビットと同一になります．(符号拡張)

(2-1-2) RFL(PRFL)ベース速度レジスタ　Read:d1(c1), Write:91(81) ･･･(括弧内はプリレジスタ)

ベース速度(加減速送りに於いて加速開始する速度または減速時に減速終了する速度)を設定．



■ 設定範囲 1～65,535(0xffff) ただし，速度(PPS)は速度倍率レジスタ(RMG)の速度倍率によります．

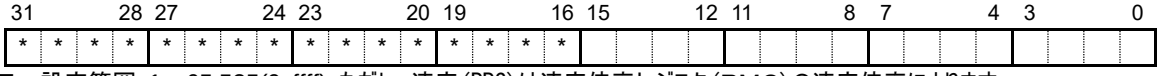
$$速度F = RFL \times \frac{300}{RMG+1}[PPS]$$

■ 書込みはプリレジスタ PRFL に書込む．

■ *表記のビットは書込み時には無視され，読出し時には 0 になる．

(2-1-3) RFH(PRFH)動作速度レジスタ　Read:d2(c2), Write:92(82) ･･･(括弧内はプリレジスタ)

動作速度を設定するレジスタです．



■ 設定範囲 1～65,535(0xffff) ただし，速度(PPS)は速度倍率レジスタ(RMG)の速度倍率によります．

$$速度F = RFH \times \frac{300}{RMG+1}[PPS]$$

■ 書込みはプリレジスタ PRFH に書込む．

■ *表記のビットは書込み時には無視され，読出し時には 0 になる．

(2-1-4) RUR(PRUR)加速レートレジスタ　Read:d3(c3), Write:93(83) ･･･(括弧内はプリレジスタ)

高速動作(加減速動作)の場合の加速特性を設定します．設定値と加速時間の関係は次式の様になります．

- 直線加速(RMD.b10 (acc 方法) = 0)

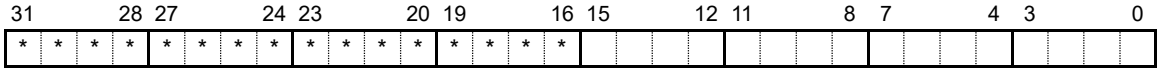
$$加速時間[sec] = \frac{(RFH - RFL) \times (RUR + 1) \times 4}{19{,}660{,}800}$$

- 直線部分のないS字加速(RMD.b10 (acc 方法) = 1 かつ RUS(加速 S 字区間) = 0)

$$加速時間[sec] = \frac{(RFH - PRFL) \times (RUR + 1) \times 8}{19{,}660{,}800}$$

- 直線部分のあるS字加速(RMD.b10 (acc 方法) = 1 かつ RUS(加速 S 字区間) > 0)

$$加速時間[sec] = \frac{(RFH - RFL + 2 \times RUS) \times (RUR + 1) \times 4}{19{,}660{,}800}$$



■ 設定範囲 1～65,535(0xffff)

■ 書込みはプリレジスタ PRUR に書込む．

■ *表記のビットは書込み時には無視され，読出し時には 0 になる．

(2-1-5) RDR(PRDR)減速レートレジスタ　Read:d4(c4), Write:94(84)　･･･(括弧内はプリレジスタ)

高速動作(加減速動作)の場合の減速特性を設定します. "0" を設定した場合は RUR の設定値を設定した時と同じ減速特性になります. 設定値と減速時間の関係は次式の様になります.

- 直線減速(RMD.b10(acc 方法) = 0)

$$減速時間[sec] = \frac{(RFH - RFL) \times (RDR + 1) \times 4}{19{,}660{,}800}$$

- 直線部分のないS字減速(RMD.b10 (acc 方法) = 1 かつ RDS(減速 S 字区間) = 0)

$$減速時間[sec] = \frac{(RFH - PRFL) \times (RDR + 1) \times 8}{19{,}660{,}800}$$

- 直線部分のあるS字減速(RMD.b10 (acc 方法) =1 かつ RDS(減速 S 字区間) > 0)

$$減速時間[sec] = \frac{(RFH - RFL + 2 \times RDS) \times (RDR + 1) \times 4}{19{,}660{,}800}$$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*																

■ 設定範囲 0～65,535(0xffff)

■ 書込みはプリレジスタ PRDR に書込む.

■ *表記のビットは書込み時には無視され, 読出し時には 0 になる.

(2-1-6) RMG(PRMG)速度倍率レジスタ　Read:d5(c5), Write:95(85)　･･･(括弧内はプリレジスタ)

速度倍率を設定します.

$$速度倍率[倍] = \frac{300}{RMG + 1}$$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*												

■ 設定範囲 2～4,095(0xfff)

■ 書込みはプリレジスタ PRMG に書込む.

■ *表記のビットは書込み時には無視され, 読出し時には 0 になる.

(2-1-7) RDP(PRDP)減速開始点レジスタ Read:d6(c6), Write:96(86)　･･･(括弧内はプリレジスタ)

加減速動作を伴なう位置決め時の減速開始点を設定するレジスタです.

自動減速開始点計算 (RMD.b13='0') の時は自動計算値に対するオフセット値となります.

手動減速開始点計算 (RMD.b13='1') の場合は終点からの減速移動量(パルス数)を設定します.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
#	#	#	#	#	#	#	#																								

■ 書込みはプリレジスタ PRDP に書込む.

■ #表記のビットは, 書込み時に無視され, 読出し時に RMD.b13(減速開始点計算方法)によって異なる.

- RMD.b13＝'0'自動設定値に対するオフセット量を示し, 正数は早めに減速する値. 負数は遅めの減速.
  オフセットが必要ない場合は "0" に設定します. #表記のビットはビット 23 と同じ.
  設定範囲は -8,388,608 (0x 800000)～ 8,388,607 (0x 7FFFFF) の範囲で設定します.
- RMD.b13＝'1'手動減速開始点計算設定. 移動残量が設定値以下になると減速開始をする.
  減速開始点のパルス数を 0～16,777,215(0FFFFFFh)の範囲で設定します.
  減速開始点の最適値は次式の様になります.
  - 直線減速(RMD.b10(acc 方法) = 0)

$$最適値[パルス] = \frac{(RFH^2 - RFL^2) \times (RDR + 1)}{(RMG + 1) \times 32{,}768}$$

  - 直線部分のないS字減速(RMD.b10 (acc 方法) = 1 かつ RDS(減速 S 字区間) = 0)

$$最適値[パルス] = \frac{(RFH^2 - RFL^2) \times (RDR + 1) \times 2}{(RMG + 1) \times 32{,}768}$$

  - 直線部分のあるS字減速(RMD.b10 (acc 方法) =1 かつ RDS(減速 S 字区間) > 0)

$$最適値[パルス] = \frac{(RFH + RFL) \times (RFH - RFL + 2 \times RDS) \times (RDR + 1)}{(RMG + 1) \times 32{,}768}$$

  (位置決めカウンタ値) ≦ (RDP 設定値) のタイミングで減速を開始します.

(2-1-8) RMD(PRMD)動作モードレジスタ　Read:d7(c7), Write:97(87)　･･･(括弧内はプリレジスタ)

15			12	11			8	7			4	3		0
合成速度	PCS有効	減速開始点	0	CTR 1DIS	acc 方法	INPSE	DLSE	M			O	D		

31			28	27			24	23			20	19		16
0	0	MSDC	0	MPIE	FH 補正	他軸停止	同時ストップ	停止指定軸設定				条件付スタート	SEQNo	

■　書込みはプリレジスタ PRMD に書込む.

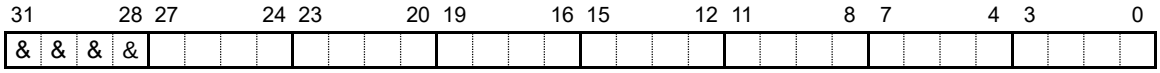
■　動作モード(MOD)

No	動作分類	MOD(HEX)	動作モード	記　事
1	連続動作	00	＋方向連続送り	
		08	－方向連続送り	
2	位置決め動作	41	位置決め動作(相対位置指定)	
		42	位置決め動作(指令位置指定)	ドライバ関数のみ設定可
		44	指令位置0点復帰動作	
		45	機械位置0点復帰動作	
		46	＋方向 1 パルス動作	
		4e	－方向 1 パルス動作	
		47	タイマ動作	
3	直線補間動作	60	直線補間連続送り	
		61	直線補間	
		62	2 個の PCL 間の直線補間連続送り	
		63	2 個の PCL 間の直線補間	
4	円弧補間動作	64	CW 方向円弧補間	
		65	CCW 方向円弧補間	
		66	U(B)軸同期 CW 方向円弧補間	
		67	U(B)軸同期 CCW 方向円弧補間	
5	原点復帰動作	10	＋方向原点復帰動作	
		18	－方向原点復帰動作	
		12	＋方向原点抜出し	
		1a	－方向原点抜出し	
		15	＋方向原点サーチ	
		1d	－方向原点サーチ	
6	ELS, SLS 動作	20	＋ELS または＋SLS 位置まで動作	
		28	－ELS または－SLS 位置まで動作	
		22	＋ELS または＋SLS 抜出し動作	
		2a	－ELS または－SLS 抜出し動作	
7	Z 相移動	24	＋方向にZ相カウント分動作	
		2c	－方向にZ相カウント分動作	
8	パルサ動作	01	パルサ入力による連続動作	手動パルサ送り
		51	パルサ入力による位置決め動作	
		54	パルサ入力による指令位置 0 点復帰動作	
		55	パルサ入力による機械位置 0 点復帰動作	
9	JOG 送り動作	02	±DR 入力による連続動作	JOG 送り
		56	±DR入力による位置決め動作	

bit	名 称	説 明
7-0	MOD	動作モード(前ページ参照)
8	DLS 有効	0:DLS 無効, 1:DLS 有効
9	INPOS 有効	0:INPOS 無効, 1:INPOS 有効
10	加減速方式	0:直線加減速, 1:S 字加減速
11	CTR1 カウント禁止	1:指令位置カウンタ(CTR1) カウント禁止
12	動作完了タイミング	0:完了タイミングをパルスの周期完了とします.
13	減速開始点計算方法	0:減速開始点自動計算, 1:減速開始点手動計算
14	PCS 有効	0:PCS 無効, 1:PCS 有効
15	補間合成速度	0:補間時合成速度一定制御 OFF, 1:補間時合成速度一定制御 ON
17,16	SEQ No	MSTS.b7,6 に現在実行中の SEQ No が反映されます.
19-18	スタート条件	00:スタートコマンド書込み後即スタート. 01:STA 入力によるスタート. 10:他軸の条件一致によるスタート. 11:指定軸の停止によるスタート.
23-20	軸停止によるスタート時の軸指定	指定軸の停止によるスタート(RMD.b19,18=11)の場合, 軸を指定します. bit20:X(V,X1,X2,X3,X4)軸の停止 bit21:Y(W,Y1,Y2,Y3,Y4)軸の停止 bit22:Z(A,Z1,Z2,Z3,Z4)軸の停止 bit23:U(B,U1,U2,U3,U4)軸の停止 ＜例＞ 0001:X(V,X1,X2,X3,X4)軸の停止で次動作スタート 1001:X(V,X1,X2,X3,X4)軸の停止かつ U(B,U1,U2,U3,U4)軸の停止で次動作スタート
24	STP 有効	1:STP 入力により停止します.
25	異常時 STP 自動出力	1:異常停止時に STP(同時停止信号)を自動出力します.
26	FH 補正	0:ON, 1:OFF
27	MPIE (円弧自動終点引込み)	1:円弧自動終点引込み
28	予約	
29	MSDC	1:減速開始点自動計算時, 減速開始点=加速パルス数とする. 0:補間動作を合成速度一定制御 ON 時のみ減速開始点=加速パルス数とする. (位置決め動作の減速開始点計算は内部自動演算)
31,30	予約	

(2-1-9) RIP(PRIP)円弧補間中心位置レジスタ Read:d8(c8), Write:98(88) ･･･(括弧内はプリレジスタ)

円弧補間時の中心位置または2つ以上のPCLにまたがる直線補間の長軸移動量を設定するレジスタ.



■ RIP は, RMD の MOD(b7-0)が次の動作モードの時有効.
円弧補間(MOD＝0x64, 0x65), 2個以上の PCL 間の直線連続送り(MOD=0x62),
2個以上の PCL 間の直線補間(MOD=0x63)

■ 円弧補間の場合は円弧中心位置を相対値で設定する.

■ 2 個以上の PCL 間の直線連続送り, 直線補間の場合は長軸の移動量を相対値で設定する.

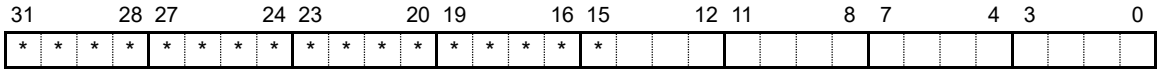
■ 設定範囲 -134,217,727～+134,217,727

■ ＆表記のﾋﾞｯﾄは, 書込み時には無視され, 読出し時には空欄表示の最上位ﾋﾞｯﾄと同一になります. (符号拡張)

(2-1-10) RUS(PRUS)加速 S 字区間レジスタ　Read:d9(c9), Write:99(89)　・・・(括弧内はプリレジスタ)

(2-1-11) RDS(PRDS)減速 S 字区間レジスタ　Read:da(ca), Write:9a(8a)　・・・(括弧内はプリレジスタ)

RUSはS字加速時のS字区間を設定するレジスタ. RDSはS字減速区間を設定するレジスタ.

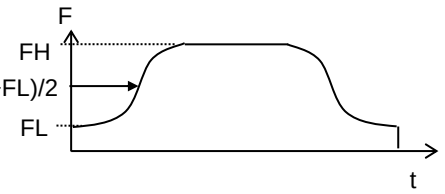


■ *表記のビットは，書込み時には無視され，読出し時には 0 になる.

■ 設定範囲 1～32,767

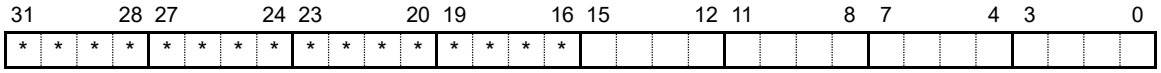
■ 0 を設定すると $\frac{RFH-RFL}{2}$ の値が自動的に設定される.(FH+FL)/2

すなわち，RUS=RDS=0 の場合には完全に S 字となる.



(2-1-12) RFA 補助速度レジスタ　Read:db, Write:9b

バックラッシュ，スリップ補正時の速度および原点復帰モード 4，6，7 において原点突入速度として使用されます.



■ 設定範囲 1～65,535　ただし，速度(PPS)は速度倍率レジスタの設定値により変化する.

■ *表記のビットは，書込み時には無視され，読出し時には 0 になる.

速度 $F = RFA \times \frac{300}{RMG+1}$ [PPS]

(2-2)　環境設定レジスタ

(2-2-1)　RENV1 環境設定レジスタ 1 Read:dc, Write:9c

指令出力，マシンインターフェイス信号，サーボインターフェイス信号など入出力極性，パルス幅，動作などを設定します．

15			12	11			8	7			4	3			0
ERCTL	EPW2	EPW1	EPW0	ERCTO	ERCTE	ALML	ALMM	OLL	SDL	SDLT	SDM	ELM	PMD2	PMD1	PMD0
31			28	27			24	23			20	19			16
0	0	INTM	DTMF	DRF	FLTR	DR	PCS	LTC	INPS	CLR1	CLR0	CSTP	CSTA	ETW1	ETW0

bit	名　称	説　　明		
2-0	PMD2-0	信号端子名	100：個別パルス方式	指令出力
		xCWP xCWN	0v 0v CW	CW パルス出力
		xCCWP xCCWN	0v 0v CCW	CCW パルス出力
		信号端子名	010：共通パルス方式	指令出力
		xCWP xCWN	0v 0v	パルス列
		xCCWP xCCWN	CW 方向 0v CCW 方向 0v	方向出力
		信号端子名	101：位相差方式(B 相進相)	指令出力
		xCWP xCWN	CW 方向　CCW 方向 0v 0v	A 相出力
		xCCWP xCCWN	0v 0v	B 相出力
		信号端子名	110：位相差方式(A 相進相)	指令出力
		xCWP xCWN	CW 方向　CCW 方向 0v 0v	A 相出力
		xCCWP xCCWN	0v 0v	B 相出力
3	ELM	ELS 検出時の停止方法　0:即停止，1:減速停止(減速距離に注意)		
4	SDM	DLS 検出時の動作方法　0:減速後定速移動継続，1:減速停止		
5	SDLT	DLS のラッチ機能　0:使用しない，1:使用する		
6	SDL	DLS の入力極性　0:B 接，1:A 接		
7	OLL	OLS の入力極性　0:B 接，1:A 接		
8	SVALMM	SVALM 入力時停止方法　0:即停止，1:減速停止		
9	SVALML	SVALM 信号入力極性　0:B 接，1:A 接		
10	ERCTE	1:異常即停止時のサーボ偏差カウンタクリア信号(SVCTRCL)自動出力する		
11	ERCTO	1:原点復帰完了時サーボ偏差カウンタクリア信号(SVCTRCL)自動出力する		
14-12	EPW2-EPW0	サーボ偏差カウンタクリア信号幅(SVCTRCL)出力幅 000:12us，001:0.1ms，010:0.4ms，011:1.6ms，100:13ms，101:52ms，110:104ms， 111:レベル		
15	SVCTRCL	1:サーボ偏差カウンタクリア信号の出力論理反転		
17-16	ETW1-ETW0	サーボ偏差カウンタクリア信号出力後ﾃﾞｨﾚｲ時間　00:0，01:12us，10:1.6ms，11:104ms		

15			12	11			8	7			4	3			0
ERCTL	EPW2	EPW1	EPW0	ERCTO	ERCTE	ALML	ALMM	OLL	SDL	SDLT	SDM	ELM	PMD2	PMD1	PMD0
31			28	27			24	23			20	19			16
0	0	INTM	DTMF	DRF	FLTR	DR	PCS	LTC	INPS	CLR1	CLR0	CSTP	CSTA	ETW1	ETW0

bit	名 称	説 明
18	CSTA	STA 入力仕様 0:レベル, 1:エッジ
19	CSTP	STP 入力時停止方法 0:即停止, 1:減速停止(注意 1)
22	INPOS	インポジション信号入力極性 0:B 接, 1:A 接
24	PCS	PCS入力信号極性 0:B 接, 1:A 接
25	DR	DR入力信号極性 0:B 接, 1:A 接
28	DTMF	方向変化タイマ ON/OFF 0:ON, 1:OFF
29	INTM	0:この軸からの割込信号出力, 1:この軸からの割込信号出力禁止

注 意

次動作連続実行時に STP 入力時減速停止に設定した場合, STP を外部から入力する場合は停止するまで入力 ON 状態を保持してください.
同時ストップコマンド使用時はプリレジキャンセルコマンドを発行してから同時ストップコマンドを発行してください.

(2-2-2) RENV2 環境設定レジスタ 2 Read :d, Write:9d

エンコーダ A/B 相カウント方法/方向, Z 相極性, パルサ A/B 相カウント方法/方向を設定します.

15			12	11			8	7			4	3			0
1	1	1	1	x	1	0	1	0	1	x	x	0	1	0	1
31			28	27			24	23			20	19			16
POFF	EOFF	SMAX	PMSK	0	PDIR	PULSR		EZL	EDIR	ENCM		0	0	0	0

bit	名 称	説 明
19-0	ボード仕様 (固定)	汎用入出力ポートの仕様設定. b19～b0 は必ず次のように初期設定する. HPCI-CPD574N, HPCI-CPD578N ・・ 0x0fd55, その他 ・・ 0x0f555
21-20	ENCM	エンコーダ逓倍設定 00:1 逓倍, 01:2 逓倍, 10:4 逓倍, 11:up/down パルス
22	EDIR	1:エンコーダ A 相, B 相カウント方向逆転
23	EZL	1:エンコーダ Z 相入力極性反転
25-24	PULSR	パルサ逓倍設定 00:1 逓倍, 01:2 逓倍, 10:4 逓倍, 11:up/down パルス
26	PDIR	1:パルサ A 相, B 相カウント方向逆転
28	PMSK	1:指令パルス出力禁止(CTR1 は動作します)
29	SMAX	1:自軸を含めた指定軸停止によるスタート有効(RMD.b23-20)
30	EOFF	1:エンコーダ入力をマスク
31	POFF	1:パルサ入力をマスク

(2-2-3)　RENV3 環境設定レジスタ 3 Read:de, Write:9e

原点復帰方法指定，原点復帰 Z 相カウント設定，CTR 動作方法の設定などを行います．

15			12	11			8	7			4	3			0
0	CTR4G	CTR4IP		CTR3IP		CTR2IP		Z 相CT				ORGmode			
31			28	27			24	23			20	19			16
CTR4 DIS	CTR3 DIS	CTR2 DIS	0	CTR4 BS	CTR3 BS	CTR2 BS	CTR1 BS	CTR4 ORG	CTR3 ORG	CTR2 ORG	CTR1 ORG	0	0	0	0

bit	名　称	説　　明
3-0	ORG mode	0000:OLSon で完了 0001:OLSon 停止後 RFA 反転 OLSoff まで動作後 RFA 再反転 OLSon で完了 0010:OLSon 減速+Z 相カウントで完了 0011:OLSon 後の Z 相カウントで完了 0100：OLSon 停止後 RFA 反転+ Z 相カウントで完了 0101：OLSon 停止後加速反転+ Z 相カウントで完了 0110：ELSon 停止後 RFA 反転 ELSoff で完了 0111：ELSon 停止後 RFA 反転+ Z 相カウントで完了 1000：ELSon 停止後加速反転+ Z 相カウントで完了 1001：ORGmode0 の動作後 CTR2 零点復帰 1010：ORGmode3 の動作後 CTR2 零点復帰 1011：ORGmode5 の動作後 CTR2 零点復帰 1100：ORGmode8 の動作後 CTR2 零点復帰
7-4	Z 相 CT	原点復帰時のエンコーダZ相カウント値：0000(1 回)～1111(16 回)
9-8	CTR2IP	CTR2 入力元選択　00:ENC 入力，01:指令パルス，10:パルサ入力
11-10	CTR3IP	CTR3 入力元選択　00:指令パルスと ENC 入力の偏差，01:指令パルスとパルサ入力の偏差
13-12	CTR4IP	CTR4 入力元選択　00:指令パルス，01:ENC 入力，10:パルサ入力，11: 9.8304MHz CLK カウント
14	CTR4G	1:動作中のみ CTR4 カウント
23-20	CTRxORG	1:原点復帰完了時 CTRx クリア
27-24	CTRxBS	1:バックラッシュ動作，スリップ動作中も CTRx カウント
31-29	CTRxDIS	1:CTRx カウントしない

(2-2-4)　RENV4 環境設定レジスタ 4 Read:df, Write:9f

コンパレータ(CMP)は 5 組あります．CMP1～4 を設定します．(CMP5 は RENV5)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C2RM	CMP2M		CMP2D			CMP2C		C1RM	CMP1M		CMP1D			CMP1C	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMP4M			CMP4D			CMP4C		0	CMP3M		CMP3D			CMP3C	

bit	名　称	説　　明
1-0	CMP1C	CMP1 比較カウンタ　　00:CTR1，01:CTR2，10:CTR3，11:CTR4
4-2	CMP1D	CMP1 条件 001:RCMP1=比較 CTR(UP/DOWN)，010: RCMP1=比較 CTR(UP 時) 011:RCMP1=比較 CTR(DOWN 時)，100: RCMP1>比較 CTR 101: RCMP1<比較 CTR，　　110:+SLS(RCMP1<CTR1)，その他:比較せず
6-5	CMP1M	CMP1 条件成立時の処理 00:処理なし，01:即停止，10:減速停止，11:動作プリレジスタのデータに変更
	C1RM	1:RCMP1 と対で CTR1 を回転軸カウンタとして使用
9-8	CMP2C	CMP2 比較カウンタ　　00:CTR1，01:CTR2，10:CTR3，11:CTR4
12-10	CMP2D	CMP2 条件 001:RCMP2=比較 CTR(UP/DOWN)，010: RCMP2=比較 CTR(UP 時) 011:RCMP2=比較 CTR(DOWN 時)，100: RCMP2>比較 CTR 101: RCMP2<比較 CTR，110:-SLS(RCMP2>CTR1)，その他:比較せず
14-13	CMP2M	CMP2 条件成立時の処理 00:処理なし，01:即停止，10:減速停止，11:動作プリレジスタのデータに変更
15	C2RM	1: RCMP2 と対で CTR2 を回転軸カウンタとして使用
17-16	CMP3C	CMP3 比較カウンタ　　00:CTR1，01:CTR2，10:CTR3，11:CTR4
20-18	CMP3D	CMP3 条件 001:RCMP3=比較 CTR(UP/DOWN)，　010: RCMP3=比較 CTR(UP 時) 011:RCMP3=比較 CTR(DOWN 時)，　100: RCMP3>比較 CTR 101: RCMP3<比較 CTR，　　110:設定禁止，その他:比較せず
22-21	CMP3M	CMP3 条件成立時の処理 00:処理なし，01:即停止，10:減速停止，11:動作プリレジスタのデータに変更
25-24	CMP4C	CMP4 比較カウンタ　　00:CTR1，01:CTR2，10:CTR3，11:CTR4
29-26	CMP3D	CMP4 条件 0001:RCMP4=比較 CTR(UP/DOWN)　　1000:定ピッチ信号出力用として使用(UP/DOWN) 0010:RCMP4=比較 CTR(UP 時)　　1001:定ピッチ信号出力用として使用(UP 時) 0011:RCMP4=比較 CTR(DOWN 時)　　1010:定ピッチ信号出力用として使用(DOWN 時) 0100: RCMP4>比較 CTR　　その他:比較せず 0101: RCMP4<比較 CTR
31-30	CMP4M	CMP4 条件成立時の処理 00:処理なし，01:即停止，10:減速停止，11:動作プリレジスタのデータに変更

注　意

比較対象カウンタとして CTR3 を選択した場合はカウント値の絶対値(0～32,767)と比較データとの比較となります．
+SLS，-SLS に設定する場合は比較対象カウンタに CTR1 を選択します．

(2-2-5) RENV5 環境設定レジスタ 5 Read:e0, Write:a0

CMP5 の設定，アイドリングパルス数，カウンタのラッチ条件，他軸スタート条件などの設定を行います．

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CTLCHF	CTLCH		PDSM	IDLP			CMP5M		CMP5D			CMP5C		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	CU4L	CU3L	CU2L	CU1L	0	0	TRIGRINsel		TRIGROTsel			

bit	名 称	説 明
2-0	CMP5C	CMP5 比較対象 000:CTR1，001:CTR2，010:CTR3，011:CTR4，100:PCTR，101:現在速度
5-3	CMP5D	CMP5 条件 001:RCMP5=比較 CTR(UP/DOWN)， 010: RCMP5=比較 CTR(UP 時) 011:RCMP5=比較 CTR(DOWN 時)， 100: RCMP5>比較 CTR 101: RCMP5<比較 CTR， その他:比較せず
7-6	CMP5M	CMP5 条件成立時の処理 00:処理なし，01:即停止，10:減速停止，11:動作プリレジスタのデータに変更
10-8	IDLP	アイドリングパルス数(0～7)
11	PDSM	1: PA/PB 及び±DR による連続動作時，動作方向の ELS によるエラー割り込みが発生します． 再スタートにはスタートコマンドが必要です．(CPDxxxN のみ)
13-12	CTLCH	CTR ラッチ条件 01:OLSoff→on，10:CMP4 条件成立時，11:CMP5 条件成立時
14	CTLCHF	1:CTR3 の代わりに速度レジスタをラッチ
19-16	TRIGROTsel	他軸をスタートさせるトリガ条件設定 0001:CMP1 条件成立時 1000:加速開始時 0010:CMP2 条件成立時 1001:加速終了時 0011:CMP3 条件成立時 1010:減速開始時 0100:CMP4 条件成立時 1011:減速終了時 0101:CMP5 条件成立時 その他:トリガ OFF
21-20	TRIGRINsel	自軸がスタートするためのトリガ軸の選択 00:X(V)軸，01:Y(W)軸，10:Z(A)軸，11:U(B)軸
27-24	CUxL	1:CTRx のラッチ後 CTRx クリア

(2-2-6) RENV6 環境設定レジスタ 6 Read:e1, Write:a1

バックラシュ/スリップ補正量，パルサ分周/逓倍比を設定します．．

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	BLSHsel		BLSH											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PMG				PD											

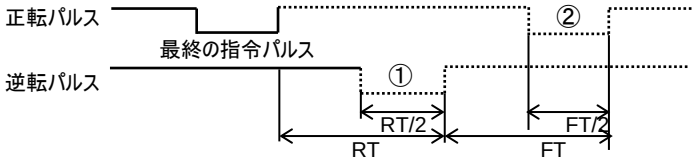
bit	名 称	説 明
11-0	BLSH	バックラッシュ/スリップ補正量
13-12	BLSHsel	00:補正 off，01:バックラッシュ補正，10:スリップ補正
26-16	PD	パルサ分周比 (設定値) / 2,048 に分周 0:分周 OFF
31-27	PMG	パルサ逓倍比 (設定値+1)倍に逓倍

(2-2-7) RENV7 環境設定レジスタ 7 Read:e2, Write:a2

ステッピングモータの停止時特性を改善する「制振機能」を設定します.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FT															

bit	名 称	説 明	
15-0	RT	最終出力パルス後の逆転パルス出力タイミング	RT×1.6us(1.6us～104ms)
31-16	FT	上記逆転パルス出力後の正転パルス出力タイミング	FT×1.6us(1.6us～104ms)



最終指令パルス出力後に逆転パルス①と正転パルス②が出力され，ステッピングモータの振動抑制が行われる.

(2-3) カウンタレジスタ

(2-3-1) RCTR1 指令位置カウンタレジスタ(Read Only) Read:e3, Write:a3

指令パルス用カウンタレジスタです. カウンタはアップダウン・カウンタです.

特定のタイミングでカウンタを読み出す場合は "ラッチ" を使用し，RLTC1 から読出します.

31	28	27	24	23	20	19	16	15	12	11	8	7	4	3	0
& & & &															

■ 設定範囲 -134,217,728～+134,217,727

■ &表記のビットは，書込み時には無視され，読出し時には空欄表示の最上位ビットと同一になります. (符号拡張)

(2-3-2) RCTR2 指令位置カウンタレジスタ(Read Only) Read:e4, Write:a4

機械位置用カウンタレジスタです. カウンタはアップダウン・カウンタです.

エンコーダ信号，パルサ信号および指令パルスが選択できます. (RENV3.b9-b8)

特定のタイミングでカウンタを読み出す場合は "ラッチ" を使用し，RLTC2 から読出します.

31	28	27	24	23	20	19	16	15	12	11	8	7	4	3	0
& & & &															

■ 設定範囲 -134,217,728～+134,217,727

■ &表記のビットは，書込み時には無視され，読出し時には空欄表示の最上位ビットと同一になります. (符号拡張)

(2-3-3) RCTR3 脱調カウンタレジスタ (Read Only) Read:e5, Write:a5

脱調検出用 偏差カウンタレジスタです. 2つの信号の偏差(差分)をカウントします.

指令パルスとエンコーダ信号，指令パルスとパルサ信号の何れかを選択できます. (RENV3.b11-b10)

特定のタイミングでカウンタを読み出す場合は "ラッチ" を使用し，RLTC2 から読出します.

31	28	27	24	23	20	19	16	15	12	11	8	7	4	3	0
& & & &	& & & &	& & & &	& & & &												

■ 設定範囲 -32,768～+32,767

■ &表記のビットは，書込み時には無視され，読出し時には空欄表示の最上位ビットと同一になります. (符号拡張)

(2-3-4) RCTR4 汎用カウンタレジスタ (Read Only) Read:e6, Write:a6

機械位置用カウンタレジスタです. カウンタはアップダウン・カウンタです.

指令パルス，エンコーダ信号，パルサ信号および 9.8304MHz クロック信号が選択入力できます(RENV3.b13-b12)

特定のタイミングでカウンタを読み出す場合は "ラッチ" を使用し，RLTC2 から読出します.

31	28	27	24	23	20	19	16	15	12	11	8	7	4	3	0
& & & &															

■ 設定範囲 -134,217,728～+134,217,727

■ &表記のビットは，書込み時には無視され，読出し時には空欄表示の最上位ビットと同一になります. (符号拡張)

(2-4)　コンパレータ比較レジスタ

(2-4-1)　RCMP1 コンパレータ 1 比較レジスタ　Read:e7, Write:a7

(2-4-2)　RCMP2 コンパレータ 2 比較レジスタ　Read:e8, Write:a8

(2-4-3)　RCMP3 コンパレータ 3 比較レジスタ　Read:e9, Write:a9

(2-4-4)　RCMP4 コンパレータ 4 比較レジスタ　Read:ea, Write:aa

コンパレータ 1～4 用比較データを設定します.

31	28	27	24	23	20	19	16	15	12	11	8	7	4	3	0
&	&	&	&												

■　設定範囲 -134,217,728～+134,217,727

■　&表記のビットは，書込み時には無視され，読出し時には空欄表示の最上位ビットと同一になります．(符号拡張)

(2-4-5)　RCMP5(PRCP5) コンパレータ 5 比較レジスタ　Read:eb(cb), Write:ab(8b)　･･･(括弧内はプリレジスタ)

コンパレータ 5 用比較データを設定します．プリレジスタがあります.

31	28	27	24	23	20	19	16	15	12	11	8	7	4	3	0
&	&	&	&												

■　設定範囲 -134,217,728～+134,217,727

■　&表記のビットは，書込み時には無視され，読出し時には空欄表示の最上位ビットと同一になります．(符号拡張)

(2-5)　イベントマスクレジスタ

(2-5-1)　RIRQ イベントマスクレジスタ(Read Only)　Read:ec, Write:ac

RIST イベントステータスレジスタに対するイベント報告を設定するレジスタです.

各ビットを 1 にするとイベントステータス(RIST)に反映されます.

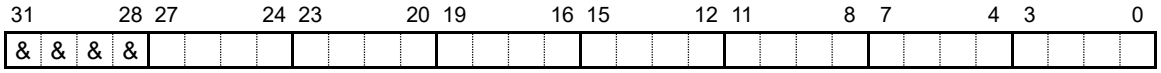
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IROL	IRLT	IRCL	IRC5	IRC4	IRC3	IRC2	IRC1	IRDE	IRDS	IRUE	IRUS	IRND	IRNM	IRN	IREN
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	IRSA	IRDR	IRSD

bit	名　称	内　容
0	IREN	動作完了報告
1	IRN	次動作継続スタート報告
2	IRNM	動作用プリレジスタフル→空き報告
3	IRND	CMP5 用プリレジ書込み可能報告
4	IRUS	加速開始報告
5	IRUE	加速終了報告
6	IRDS	減速開始報告
7	IRDE	減速終了報告
12-8	IRCn	CMPn 比較条件成立報告
13	IRCL	CLR 入力によるカウンタクリア報告
14	IRLT	LTCH 入力によるカウンタラッチ報告
15	IROL	OLS 信号入力時にカウンタラッチ報告
16	IRSD	DLS 信号 OFF→ON
17	IRDR	±DR 信号 OFF→ON
18	IRSA	STA 信号 OFF→ON
19-31	未定義	常に'0'を設定してください

(2-6) カウンタラッチレジスタ

(2-6-1) RLTC1 カウンタラッチレジスタ(Read Only) Read:ed

RCTR1 のカウンタラッチレジスタです.

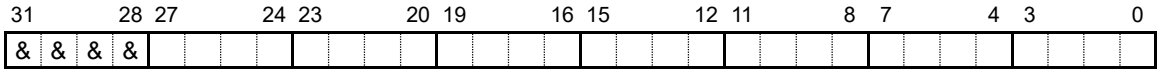


■ データ範囲 -134,217,728～+134,217,727

■ &表記のビットは，書込み時には無視され，読出し時には空欄表示の最上位ビットと同一になります. (符号拡張)

(2-6-2) RLTC2 カウンタラッチレジスタ(Read Only) Read:ee

RCTR2 のカウンタラッチレジスタです.

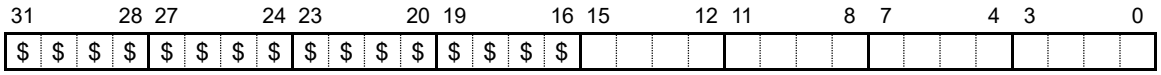


■ データ範囲 -134,217,728～+134,217,727

■ &表記のビットは，書込み時には無視され，読出し時には空欄表示の最上位ビットと同一になります. (符号拡張)

(2-6-3) RLTC3 カウンタラッチレジスタ(Read Only) Read:0xef

RCTR3のカウンタラッチレジスタです. RENV5.LTFD(b14)=0 の時はRCTR3をLTFD=1の時は現在速度をラッチします.



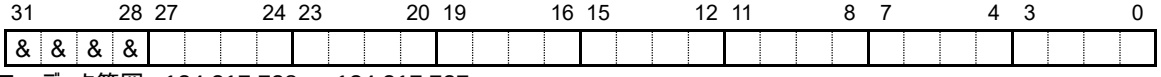
■ データ範囲

- LTFD=0 の時は -32,768 ～ +32,767
- LTFD=1 の時は 0 ～ 65535

■ $表示のビットは,
RENV5.LTFD(b14)=0 の時は空欄表示の最上位ビットと同一になります(符号拡張)
RENV5.LTFD(b14)=1 の時は "0" になります

(2-6-4) RLTC4 カウンタラッチレジスタ(Read Only) Read:0xf0

RCTR4 のカウンタラッチレジスタです.



■ データ範囲 -134,217,728～+134,217,727

■ &表記のビットは，書込み時には無視され，読出し時には空欄表示の最上位ビットと同一になります. (符号拡張)

(2-7) ステータスレジスタ

(2-7-1) RSTS 拡張ステータスレジスタ(Read Only) Read:f1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDIN	SLTC	SCLR	SDRM	SDRP	SEZ	SERC	SPCS	SEMG	SSTP	SSTA	SDIR	CND3	CND2	CND1	CND0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	PFM1	PFM0	PFC1	PFC0	0	SINP

bit	名称	内　　容
0 1 2 3	CND0 CND1 CND2 CND3	動作状態を表す 0000:停止中 0001:DR 入力待ち 0010:STA 入力待ち 0011:スタートコマンドが発行されていて，軸スタート条件成立待ち状態. 0100:指定軸の停止によるスタート待ち 0101:サーボ偏差カウンタ･クリアタイマ on 中 0110:指令方向信号切替中(共通パルス指令時方向切替時間 0.2ms) 0111:バックラシュ補正中(バックラッシュ補正中の時間は補正パルス量と補助速度による) 1000:パルサ入力待ち 1001:FA(補助速度)定速で動作中 1010:FL 定速で動作中 1011:加速中 1100:FH 定速で動作中 1101:減速中 1110:INPOS 信号待ち 1111:その他
4	SDIR	動作方向(0:＋方向，1:－方向)
5	SSTA	‘1’= 同時スタート信号 on 状態
6	SSTP	‘1’= 同時停止信号 on 状態
7	SEMG	‘1’= EMG 信号入力中
8	SPCS	‘1’= PCS 位置決めスタート信号入力中
9	SERC	‘1’= サーボ偏差カウンタクリア信号出力中
10	SEZ	‘1’= エンコーダ Z 相信号入力中
11	SDRP	‘1’= ＋DR 信号入力中
12	SDRM	‘1’= －DR 信号入力中
13	SCLR	‘1’= CLR 信号入力中
14	SLTC	‘1’= LTCH 信号入力中
15	SDIN	‘1’= DLS 信号入力中
16	SINP	‘1’= INPOS 信号入力中
17	未定義	常に'0'
19,18	PFC1,0	RCMP5 用プリレジスタの使用状態 00:未確定，01:レジスタ確定，10:1st プリレジスタ確定，11:2nd プリレジスタ確定(プリレジスタフル)
21,20	PFM1,0	動作用プリレジスタ(RCMP5 用以外)の使用状態 00:未確定，01:レジスタ確定，10:1st プリレジスタ確定，11:2nd プリレジスタ確定(プリレジスタフル)
31-22	未定義	常に'0'

(2-7-2)　REST エラーステータスレジスタ(Read Only)Read:f2

エラー報告の発生要因を確認できます.
エラー報告が発生した時に対応するビットが"1"になります. またこの時 MSTS.SERR(b4)=1 となります.
このレジスタは、読み出しによりリセットされ, MSTS.SERR(b4)=0となります. エラーステータスはマスクすることは出来ません.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ESAO	ESPO	ESIP	ESDT	0	ESSD	ESEM	ESSP	ESAL	ESML	ESPL	ESC5	ESC4	ESC3	ESC2	ESC1
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	ESPE	ESEE

bit	名　称	内　容	備考
0	ESC1	1 = CMP1 条件成立で停止 (+SLS)	
1	ESC2	1 = CMP2 条件成立で停止 (-SLS)	
2	ESC3	1 = CMP3 条件成立で停止	
3	ESC4	1 = CMP4 条件成立で停止	
4	ESC5	1 = CMP5 条件成立で停止	
5	ESPL	1 ＝+ELS による停止	停止中に+ELS が ON しても 1 になりません.
6	ESML	1 ＝-ELS による停止	停止中に-ELS が ON しても 1 になりません.
7	ESAL	1 ＝サーボアラームによる停止	停止中に SVALM が ON しても 1 になりません.
8	ESSP	1 = STP 入力 ON による停止 (または同時ストップコマンド)による停止	次動作使用時はプリレジスタキャンセルコマンド(26h)を書き込んでください.
9	ESEM	1 = EMG 入力 ON による停止	非常停止コマンドによる停止または非常停止機能を使用している場合
10	ESSD	1 = DLS 検出による減速停止	停止中に DLS が ON しても 1 になりません.
11	未定義	常に'0'	
12	ESDT	1 = 動作データが不正で停止	
13	ESIP	1 = 補間他軸の異常停止による停止	
14	ESPO	1 =パルサ用バッファオーバーフローによる停止	(パルサ入力×パルサ逓倍)>動作速度の状態が続いた場合発生します.
15	ESAO	1 = 補間データのレンジオーバによる停止	
17,16	ESPE ,ESEE	1 = エンコーダ A/B 相が同時に変化	停止しません.
31-18	未定義	常に'0'	

注意 1. 以下の場合に ESDT=1 になります.

- 1軸だけ直線補間1モード(MOD=60h,61h,68h,69h)にしてスタートコマンドを書き込んだ時.
- 1軸だけ円弧補間モード(MOD=64h,65h,66h,67h,6Ch,6Dh)にしてスタートコマンドを書き込んだ時.
- 円弧補間モードで RIP 設定(円弧中心座標)を(0,0)にしてスタートコマンドを書き込んだ時.
- 3 軸または 4 軸を円弧補間モードにしてスタートコマンドを書き込んだ時.
- 直線補間 2 モード(MOD=62h,63h,6Ah,6Bh), RIP=0 の状態でスタートコマンドを書き込んだ時.
- U 軸同期の円弧補間モード(MOD=66h,67h)でスタートコマンドを書き込んだ時にU軸が動作しない時.
  または円弧補間動作中に U 軸が動作完了になった時.

(2-7-3)　RIST イベントステータス (Read Only) Read:f3

イベントマスク(RIRQ)の設定をすることでイベントステータスが有効となります.
イベント報告の発生要因を確認できます.
イベント報告が発生した時に対応するビットが"1"になります. またこの時 MSTS.SINT(b5)=1 となります.
このレジスタは、読み出しによりリセットされ, MSTS.SINT(b5)=0 となります.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISOL	ISLT	ISCL	ISC5	ISC4	ISC3	ISC2	ISC1	ISDE	ISDS	ISUE	ISUS	ISND	ISNM	ISN	ISEN
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	ISSA	ISMD	ISPD	ISSD

bit	名　称	内　容
0	ISEN	正常停止. 本ビットの変化タイミングは, INPOS 制御設定により異なります.
1	ISN	次動作継続スタート.
2	ISNM	動作用プリレジスタフル→空き
3	ISND	CMP5 用プリレジスタフル→空き
4	ISUS	加速開始
5	ISUE	加速終了
6	ISDS	減速開始
7	ISDE	減速終了
12-8	ISCn	CMPn 比較条件成立
13	ISCL	CLR 入力によるカウンタクリア時
14	ISLT	LTCH 入力によるカウンタラッチ時
15	ISOL	OLS 入力によるカウンタラッチ時
16	ISSD	DLS 信号　OFF→ON
17	ISPD	＋DR 信号　OFF→ON
18	ISMD	－DR 信号　OFF→ON
19	ISSA	STA 信号　OFF→ON
31-20	未定義	常に'0'

(2-8)　その他読み出しレジスタ

(2-8-1)　RPLS 移動残パルスレジスタ(Read Only) Read:f4

位置決めカウンタの移動残量を確認出来ます．スタート時にRMVの絶対値になり，パルス出力ごとにダウンカウントされます．
途中停止時残パルスが読めます．

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0																												

(2-8-2)　RSPD　EZ カウンタ・指令速度モニタレジスタ(Read Only) Read:f5

現在速度の他に原点復帰時エンコーダ Z 相カウント値，アイドリングカウント値が読めます．
指令速度は読んだ値に速度倍率を掛けた結果が指令速度となります．

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	IDL CT			ZCT				C	R	N	T	V	E	L	O								

ビット	名　称	説　　明
15-0	CRNT VELO	この値に速度倍率を掛けた値が指令速度．停止時は "0"
19-16	ZCT	原点復帰に使用されている Z 相カウント値
22-20	IDL CT	アイドリングカウント値を読出す

(2-8-3)　RSDC　減速開始点計算値レジスタ(Read Only) Read:0xf6

位置決め動作における減速開始点自動演算値を確認できます．

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0																								

(2-9)　円弧補間歩進数レジスタ

(2-9-1)　RCI(PRCI) 円弧補間歩進数レジスタ Read:fc(cc)，Write:bc(8c)　･･･(括弧内はプリレジスタ)

X(V)，Y(W)，Z(A)軸のみに存在し，U(B)軸にはありません．
円弧補間時に減速を行う場合に円弧補間に必要な歩進数を設定します．
0 以外の値を設定することにより自動減速開始点設定による減速が行えます．(ただし合成速度一定制御 ON 時は不可)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*																															

■　設定範囲 0 ～ 2,147,483,647

(2-9-2)　RCIC　円弧補間歩進数カウンタレジスタ(Read Only) Read:fd

円弧補間歩進カウント値を読み出すレジスタ．円弧補間スタート時に RCI の値になり円弧補間パルス出力毎にダウンカウントします．
ただし，カウント値=0 になるとダウンカウントしません．また，円弧補間完了後のカウント値は，次に円弧補間動作を開始するまで記憶されています．
X 軸からの読み出しで X～U 軸の円弧補間歩進数．V 軸からの読み出しで V～B 軸の円弧補間歩進数．

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0																															

■　データ範囲 0 ～ 2,147,483,647

(2-10)　補間ステータスレジスタ(参考)

(2-10-1) RIPS　補間ステータスレジスタ(Read Only) Read:ff

補間設定状態と動作状態を読み出します.

X 軸からの読み出しで X～U 軸の補間ステータス. V 軸からの読み出しで V～B 軸の補間ステータス.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IPFu	IPFz	IPFy	IPFx	IPSu	IPSz	IPSy	IPSx	IPEu	IPEz	IPEy	IPEx	IPLu	IPLz	IPLy	IPLx
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	SED1	SED0	SDM1	SDM0	IPCC	IPCW	IPE	IPL

ビット	名　称	説　　明
0	IPLx	1:X(V)軸が直線補間モード
1	IPLy	1:Y(W)軸が直線補間モード
2	IPLz	1:Z(A)軸が直線補間モード
3	IPLu	1:U(B)軸が直線補間モード
4	IPEx	1:X(V)軸が PCL 間の直線補間モード
5	IPEy	1:Y(W)軸が PCL 間の直線補間モード
6	IPEz	1:Z(A)軸が PCL 間の直線補間モード
7	IPEu	1:U(B)軸が PCL 間の直線補間モード
8	IPSx	1:X(V)軸が円弧補間モード
9	IPSy	1:Y(W)軸が円弧補間モード
10	IPSz	1:Z(A)軸が円弧補間モード
11	IPSu	1:U(B)軸が円弧補間モード
12	IPFx	1:X(V)軸が合成速度一定指定
13	IPFy	1:Y(W)軸が合成速度一定指定
14	IPFz	1:Z(A)軸が合成速度一定指定
15	IPFu	1:U(B)軸が合成速度一定指定
16	IPL	1:直線補間で動作中
17	IPE	1:PCL 間の直線補間で動作中
18	IPCW	1:CW 方向円弧補間動作中
19	IPCC	1:CCW 方向円弧補間動作中
21-20	SDM1-0	円弧補間の現在象限　00:第 1 象限, 01:第 2 象限, 10:第 3 象限, 11:第 4 象限
23-22	SED1-0	円弧補間の終点象限　00:第 1 象限, 01:第 2 象限, 10:第 3 象限, 11:第 4 象限

3.2.11 cp52c_rPortB()　オプションポートバイト読出し

機　能	デバイスハンドルで指定されたボードの指定オプションポートまたは入力ポート・出力ポートの内容を読出します.

開発環境	書　式
VC++	DWORD WINAPI cpxxx_rPortB(DWORD hDevID, BYTE byCmd, BYTE* byData);
VB6	Declare Function cpxxx_rPortB Lib "hicpd52c.dll" _ (ByVal hDevID As Long, ByVal byCmd As Byte, ByRef byData As Byte) As Long
VB.NET	Declare Function cpxxx_rPortB Lib "hicpd52c.dll" _ (ByVal hDevID As Integer, ByVal byCmd As Byte, ByRef byData As Byte) As Integer
VC#	[DllImport("*hicpd52c.dll* ")] public static extern uint cpxxx_ rPortB (uint hDevID, byte byCmd, ref byte byData);

引　数	説　明
hDevID	デバイスハンドル
byCmd	オプションポート読出しコマンド オプションポート詳細を参照
byData	オプションポート読出データ データ内容についてはオプションポート詳細を参照

VC++ 記述例	DWORD ret;　　//関数の戻り値 BYTE　　byData; ret = cp52c_rPortB(hDevID, 0x80, &byData);　// 各軸 ELS 入力極性設定を読出し

3.2.12 cp52c_wPortB()　オプションポートバイト書込み

機　能	デバイスハンドルで指定されたボードの指定オプションポートまたは出力ポートへデータを書込みます.

開発環境	書　式
VC++	DWORD WINAPI cpxxx_wPortB(DWORD hDevID, BYTE byCmd, BYTE　byData);
VB6	Declare Function cpxxx_wPortB Lib "hicpd52c.dll" _ (ByVal hDevID As Long, ByVal byCmd As Byte, ByVal byData As Byte) As Long
VB.NET	Declare Function cpxxx_wPortB Lib "hicpd52c.dll" _ (ByVal hDevID As Integer, ByVal byCmd As Byte, ByVal byData As Byte) As Integer
VC#	[DllImport("*hicpd52c.dll* ")] public static extern uint cpxxx_wPortB (uint hDevID, byte byCmd, byte byData);

引　数	説　明
hDevID	デバイスハンドル
byCmd	オプションポート書込コマンド コマンド内容についてはオプションポート詳細を参照
byData	オプションポート書込データ データ内容についてはオプションポート詳細を参照

VC++ 記述例	DWORD ret;　//関数の戻り値 ret = cp52c_wPortB(hDevID, 0x80, 0x01);　// X 軸 ELS のみ A 接に設定

3.2.13 cp52c_rPortW() オプションポートワード(2 バイト)読出し

機　能	デバイスハンドルで指定されたボードの指定オプションポートまたは入力ポート・出力ポートの内容を読出します.

開発環境	書　式
VC++	DWORD WINAPI cp52c_rPortW(DWORD hDevID, BYTE byCmd, WORD* wData);
VB6	Declare Function cp52c_rPortW Lib "hicpd52c.dll" _ (ByVal hDevID As Long, ByVal byCmd As Byte, ByRef wData As Integer) As Long
VB.NET	Declare Function cp52c_rPortW Lib "hicpd52c.dll" _ (ByVal hDevID As Integer, ByVal byCmd As Byte, ByRef wData As Short) As Integer
VC#	[DllImport("*hicpd52c.dll* ")] public static extern uint cpxxx_ rPortW (uint hDevID, byte byCmd, ref ushort wData);

引　数	説　明
hDevID	デバイスハンドル
byCmd	オプションポート読出しコマンド コマンド内容についてはオプションポート詳細を参照
wData	オプションポート読出データ データ内容についてはオプションポート詳細を参照

VC++ 記述例	DWORD　ret;　//関数の戻り値 WORD　wData; ret = cp52c_rPortW(hDevID, 0x80, &wData);　// 各軸 ELS 極性設定を読出し

3.2.14 cp52c_wPortW() オプションポートワード(2 バイト)書込み

機　能	デバイスハンドルで指定されたボードの指定オプションポートまたは出力ポートへデータを書込みます.

開発環境	書　式
VC++	DWORD WINAPI cpxxx_wPortW(DWORD hDevID, BYTE byCmd, WORD　wData);
VB6	Declare Function cpxxx_wPortW Lib "hicpd52c.dll" _ (ByVal hDevID As Long, ByVal byCmd As Byte, ByVal wData As Integer) As Long
VB.NET	Declare Function cpxxx_wPortW Lib "hicpd52c.dll" _ (ByVal hDevID As Integer, ByVal byCmd As Byte, ByVal wData As Short) As Integer
VC#	[DllImport("*hicpd52c.dll* ")] public static extern uint cpxxx_wPortW (uint hDevID, byte byCmd, ushort wData);

引　数	説　明
hDevID	デバイスハンドル
byCmd	オプションポート書込コマンド コマンド内容についてはオプションポート詳細を参照
wData	オプションポート書込データ データ内容についてはオプションポート詳細を参照

VC++ 記述例	DWORD　ret;　//関数の戻り値 ret = cp52c_wPortW(hDevID, 0x80, 0x0001);　　// X 軸 ELS のみ A 接に設定

3.2.15 cp52c_rBufDW() 入出力ﾊﾞｯﾌｧ読出し

機　能	デバイスハンドルで指定されたボードの指定軸の入出力ﾊﾞｯﾌｧを読出します.

開発環境	書　式
VC++	DWORD WINAPI cpxxx_rBufDW(DWORD hDevID, WORD axis, DWORD* dwData);
VB6	Declare Function cpxxx_rBufDW Lib "hicpd52c.dll" _ (ByVal hDevID As Long, ByVal axis As Integer, ByRef dwData As Long) As Long
VB.NET	Declare Function cpxxx_rBufDW Lib "hicpd52c.dll" _ (ByVal hDevID As Integer, ByVal axis As Short, ByRef dwData As Integer) As Integer
VC#	[DllImport("*hicpd52c.dll* ")] public static extern uint cpxxx_rBufDW (uint hDevID, ushort axis, ref uint dwData);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
dwData	入出力ﾊﾞｯﾌｧ読出データ

VC++ 記述例	DWORD ret;　　//関数の戻り値 DWORD dwData;　　//入出力ﾊﾞｯﾌｧデータ ret = cpxxx_wCmdW(hDevID, 0, 0x03c0);　　//XY 軸 PRMV 読み出し ret = cpxxx_rBufDW(hDevID, 0, &dwData);　　//X 軸入出力ﾊﾞｯﾌｧから読出 ret = cpxxx_rBufDW(hDevID, 1, &dwData);　　//Y 軸入出力ﾊﾞｯﾌｧから読出

備　考	＜レジスタ読出関数との相違＞ レジスタ読出関数　　・・PCL の指定軸入出力ﾊﾞｯﾌｧを経由して目的レジスタを対象 入出力ﾊﾞｯﾌｧ操作関数　・・PCL の指定軸入出力ﾊﾞｯﾌｧの読出です. ＜レジスタ読出関数(cpxxx_rBufDW())の応用＞ 複数軸のレジスタデータを同じタイミングで一括読出しを行います. (同一 PCL 内の軸) ● cpxxx_wCmdW()関数の"コマンド"で複数軸の読出したいプリレジスタを指定. ● 軸指定(axis)は X 軸(0), 制御コマンドデータ(cmd)中のコマンド実行軸(SELx)に 2 軸以上設定, コマンドコード(code)に読出コマンドを設定 ● コマンド実行軸(SELx)で指定した全ての軸の入出力ﾊﾞｯﾌｧを読出

3.2.16 cp52c_wBufDW()入出力ﾊﾞｯﾌｧ書込み

機　能	デバイスハンドルで指定されたボードの指定軸の入出力ﾊﾞｯﾌｧにデータを書込みます.

開発環境	書　式
VC++	DWORD WINAPI cpxxx_wBufDW(DWORD hDevID, WORD axis, DWORD　dwData);
VB6	Declare Function cpxxx_wBufDW Lib "hicpd52c.dll" _ (ByVal hDevID As Long, ByVal axis As Integer, ByVal dwData As Long) As Long
VB.NET	Declare Function cpxxx_wBufDW Lib "hicpd52c.dll" _ (ByVal hDevID As Integer, ByVal axis As Short, ByVal dwData As Integer) As Integer
VC#	[DllImport("*hicpd52c.dll* ")] public static extern uint cpxxx_wBufDW (uint hDevID, ushort axis, uint dwData);

引　数	説　明
hDevID	デバイスハンドル
axis	軸指定
dwData	入出力ﾊﾞｯﾌｧ書込データ

VC++ 記述例	DWORD ret;　　//関数の戻り値 ret = cpxxx_wBufDW(hDevID, 0, 10000);　　//X 軸入出力ﾊﾞｯﾌｧへ書込 ret = cpxxx_wBufDW(hDevID, 1, 20000);　　//Y 軸入出力ﾊﾞｯﾌｧへ書込 ret = cpxxx_wCmdW(hDevID, 0, 0x0380);　　//XY 軸 PRMV 書込

備　考	＜レジスタ書込関数との相違＞ 　レジスタ書込関数　　・・PCL の指定軸入出力ﾊﾞｯﾌｧを経由して目的レジスタを対象. 　入出力ﾊﾞｯﾌｧ操作関数　・・PCL の指定軸入出力ﾊﾞｯﾌｧの書込みです. ＜レジスタ書込関数(cpxxx_wBufDW())の応用＞ 　複数軸へのレジスタデータを同じタイミングで一括書込を行います. (同一 PCL 内の軸) ● 書込みを行う全ての軸入出力ﾊﾞｯﾌｧに所定データを書込 ● cpxxx_wCmdW()関数の"コマンド"で複数軸の読出したいプリレジスタ・レジスタを指定. ● 軸指定(axis)はＸ軸(0), 制御コマンドデータ(cmd)中のコマンド実行軸(SELx)に２軸以上設定, コマンドコード(code)に書込コマンドを設定.

4. サンプルプログラム

4.1 Windows 版サンプルプログラム

ライブラリ(レベル1)関数の使用方法を解説する目的のサンプルプログラムを添付しています.
サンプルプログラムは次の2種類があり, ほぼ同一の画面表示と操作になっています.
以降のサンプルプログラム説明では, (1)の「C コーディング」を用います.

(1) Visual C++ 2008	【 spc52c00.exe 】
(2) Visual Basic (6.0)	【 spc52c02.exe 】
(3) Visual Basic 2008	【 spc52c03.exe 】
(4) Visual C# 2008	【 spc52c04.exe 】

4.1.1 サンプルプログラムの実行

サンプルプログラムを使用する場合は, お客様のハードディスクにコピーして使用します.
個々のサンプル実行ファイル(*.exe)は"マウスのダブルクリック"操作を行う事で実行できます.

(1)サンプルプログラム実行上の注意事項

- Visual C++ サンプルは
  開発ツールとして Microsoft Visual C++ 2008 以降がインストールされている必要があります.
  2008 以前の開発環境を使用する場合は, プロジェクトを開発環境に合わせて作成してください.
  VC2010 以降を使用する場合, 本サンプルプログラムを開くと変換ウィザードが起動しますので, それに従って自動で変換を行って下さい. 不十分な部分がある場合は手動で変換を行ってください.
- Visual Basic サンプルは
  開発ツールとして Microsoft Visual Basic 5.0(6.0)がインストールされている必要があります.
- Visual Basic .NET サンプルは
  開発ツールとして VB 2008 以降がインストールされている必要があります.
  2008 以前の開発環境を使用する場合は, プロジェクトを開発環境に合わせて作成してください.
  VB2010 以降を使用する場合, 本サンプルプログラムを開くと変換ウィザードが起動しますので, それに従って自動で変換を行って下さい. 不十分な部分がある場合は手動で変換を行ってください.
- Visual C# サンプルは
  開発ツールとして VC# 2008 以降がインストールされている必要があります.
  2008 以前の開発環境を使用する場合は, プロジェクトを開発環境に合わせて作成してください.
  VC#2010 以降を使用する場合, 本サンプルプログラムを開くと変換ウィザードが起動しますので, それに従って自動で変換を行って下さい. 不十分な部分がある場合は手動で変換を行ってください.
- OS が WindowsNT4.0 で, 開発ツールとして Visual Basic 6.0 を使用されている場合, spc52c02.exe は実行することができない場合があります. この場合, プロジェクトファイル(spc52c02.vbp) を開き, spc52c02.exe を作成しなおすことで, spc52c02.exe が実行できるようになります.
- ある1枚の CPD ボードは,ボード ID を "0"に設定して下さい.
- CPD ボードを 2 枚以上で使用する場合, ボード ID は重複しないようにして下さい.
  ボード ID が重複した場合は, 最初に見つけられたボードが動作します.
- 実行開始時に次のエラーメッセージが表示される場合には,プログラムは動作しません.

(2)エラーメッセージの表示

※CPD が装着されていない.
または, システムが認識していない.

※ ボード ID = 0 のボード装着されていない.

図 4.1-1 サンプルプログラムのエラーメッセージ

4.1.2　サンプルプログラムの操作

サンプルプログラムでは各軸の初期化は一部ソースプログラムで固定されています.
その為に,初期化の条件を変更して動作させたい場合には,ソースプログラム変更の必要があります.

サンプルプログラムが正常に起動されると,次の動作選択画面が表示されます.

[動作選択画面]

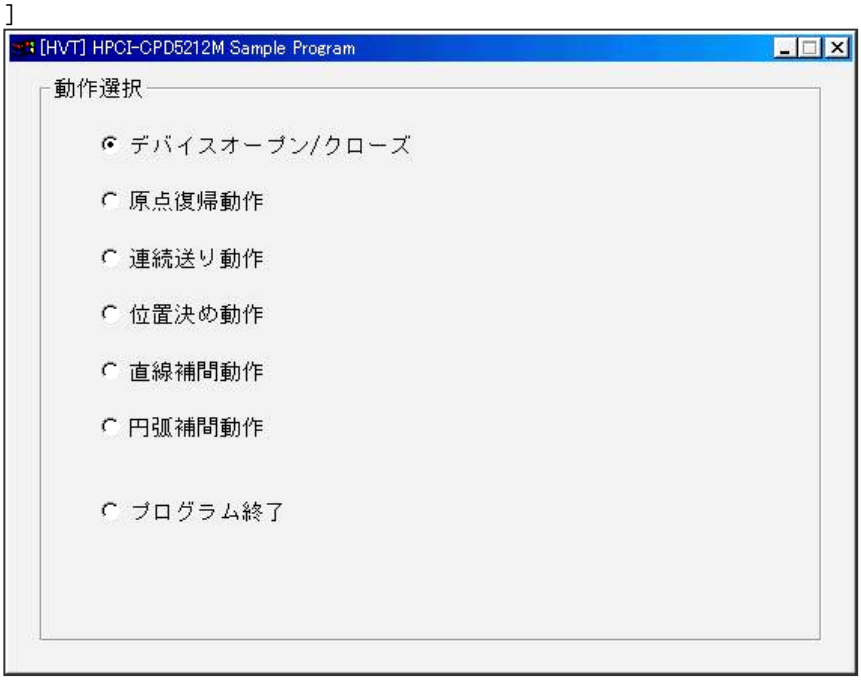


図 4.1-2　サンプルプログラムの動作選択画面

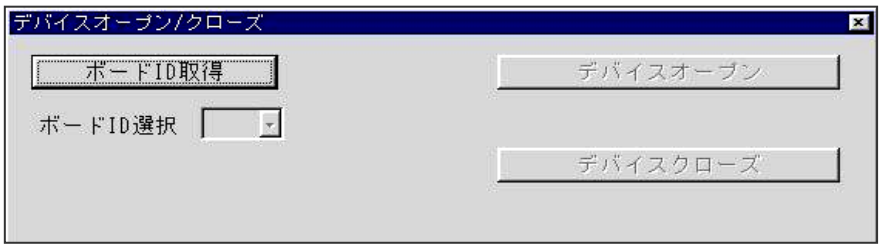
動作を選択すると,その動作のサンプルが実行されます.
（ VC++, VB.NET, VC# サンプルではシングルクリック, VBサンプルではダブルクリックで動作選択されます. ）

(1)デバイスオープン／クローズ

デバイス情報の取得とデバイスのオープン/クローズを行います.
CPD にアクセスするためには，まずこのデバイスをオープンして，アクセスするためにデバイスハンドル値を取得する必要があります．デバイスオープン関数ではデバイスハンドルを取得すると同時に，各レジスタ及び，オプションポートの初期化も行います．
このサンプルではボード ID 取得でボードを選択し，デバイスオープンで選択したボードのデバイス情報を取得し，そのボードのデバイスをオープンしています.
またデバイスクローズで，そのボードのデバイスをクローズしています.
デバイスのオープン/クローズ，デバイス情報についてはユーザーズマニュアル＜共通編＞を参照して下さい.

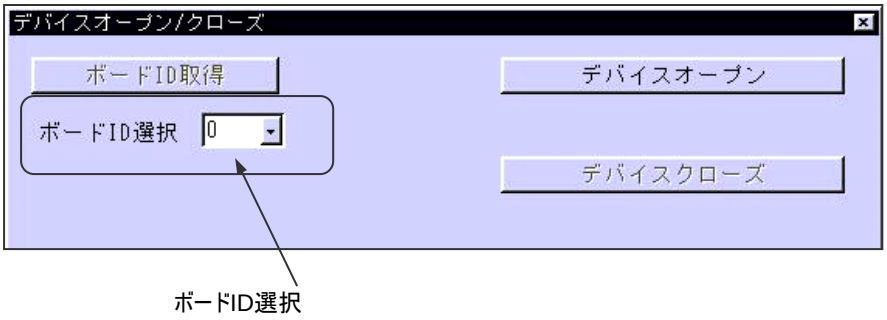
以下にサンプルの操作方法を示します.

[デバイスオープン/クローズ画面]



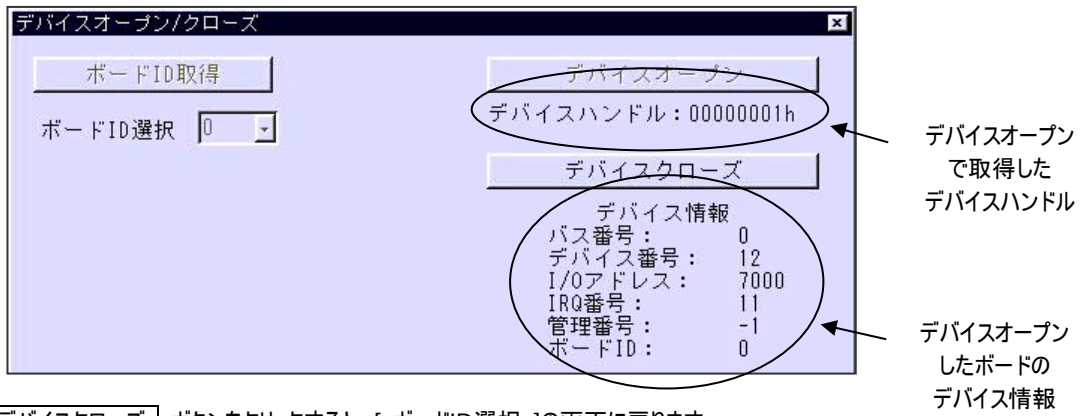
この画面で ボードID取得 ボタンをクリックすると，ボードID選択の中に装着されているボードのボードIDが表示されます.

[ボードID選択]



ここでボードIDを選択し，デバイスオープン ボタンをクリックし，デバイスオープンします.

[デバイスオープンボタンをクリックした時の画面]

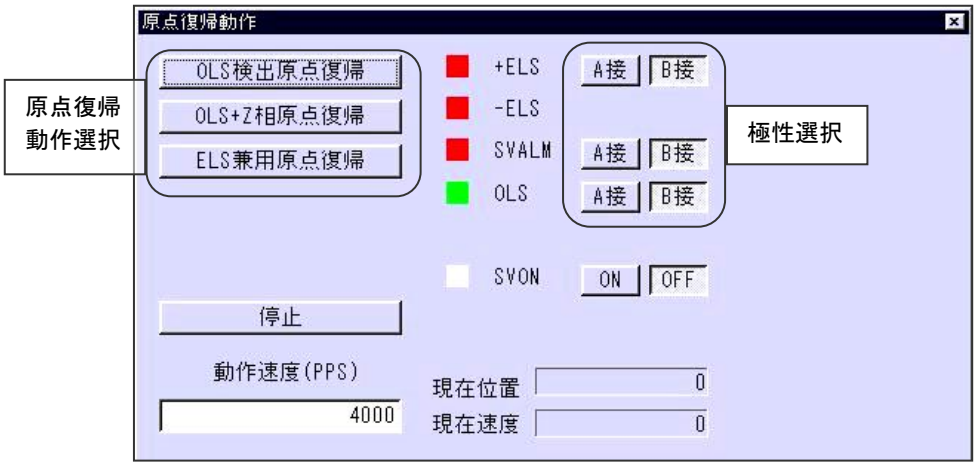


デバイスクローズ ボタンをクリックすると，[ボードID選択]の画面に戻ります.

(2)原点復帰動作

原点復帰動作の設定と原点復帰動作を行います.

[原点復帰動作初期画面]



■動作準備

① 極性選択

センサが入力されている場合，矢印部分の色が変わります.

+ELS，－ELS，SVALMが入力されると赤色，OLSは緑色になります.

SVONが出力されていると，緑色になります.

A接, B接 ボタンをクリックすることによって

入力極性を切り替えることができます.

ON, OFF ボタンをクリックすることによって

サーボオン／オフします.

パルスモータドライバの場合・・・ OFF で励磁オン, ON で励磁オフになります. ）

（注）1. +ELS，－ELS，SVALMが入力されていると動作をしません.

各センサの状態を確認してから，動作を開始して下さい.

※SVONは，所定の接続が行われているものとします.

2. A 接は端子に電流が流れたとき「ON（検出）」,

B 接は端子に常時流れている電流が切れたとき「ON（検出）」のことを云います.

② 動作速度設定

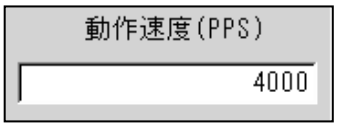
動作速度は 1-65535(PPS)の範囲で設定できます.

初期値は 4000(PPS)になっていますので，必要に応じて適当な値に設定して下さい.

また，ベース速度を 400(PPS)に設定していますので，動作速度を 400(PPS 以下に設定すると，OLSon で減速すべきところで，400(PPS)に加速することになります.

このような場合，サンプルソースプログラムを変更し，ベース速度を適当な値に設定して下さい.

動作速度設定



■原点復帰動作の実行

次の原点復帰動作方法が選択できます.

0:OLS検出原点復帰　・・・原点復帰動作1:OLS検出後抜出し，再突入して完了.

1:OLS+Z相原点復帰　・・・原点復帰動作2:OLSon検出とエンコーダZ相検出.

2:ELS兼用原点復帰　・・・原点復帰動作6:ELS検出で反転，ELS抜出しで完了

※原点復帰動作の詳細は「ユーザーズマニュアル<ソフトウェア編＞」を参照して下さい.

動作準備をした後，上記3種類の指定 原点復帰 ボタンをクリックすると，原点復帰動作が実行されます.

停止 ボタンをクリックすることで，途中で停止することができます.

現在位置表示は指令パルスカウンタを表示しています.

現在速度表示で現在出力されているパルス速度（PPS）がわかります.

(注) OLSの検出はOLSoffからOLSonのエッジ検出ですので，動作開始時にOLSonの状態の時はOLSを検出しません.
この場合は，連続送り動作でOLSoffの状態になるまで引き出してから，原点復帰動作を実行してください.

(3)連続送り動作

高速連続送り動作,及び定速連続送り動作を行います

［ 連続送り動作画面 ］

原点復帰動作の時と同様に，センサの接続等を確認してから動作を開始して下さい.

+高速連続送り，+定速連続送り，—高速連続送り，—定速連続送り ボタンをクリックし，それぞれの動作を行います. 停止 ボタンで動作を停止することができます.

(4)位置決め動作

高速位置決め動作,及び定速位置決め動作を行います

[位置決め動作画面]

原点復帰動作の時と同様に，センサの接続等を確認してから動作を開始して下さい.

高速位置決め , 定速位置決め ボタンをクリックし，それぞれの動作を行います.

移動量をパルス単位で設定します.（符号付）

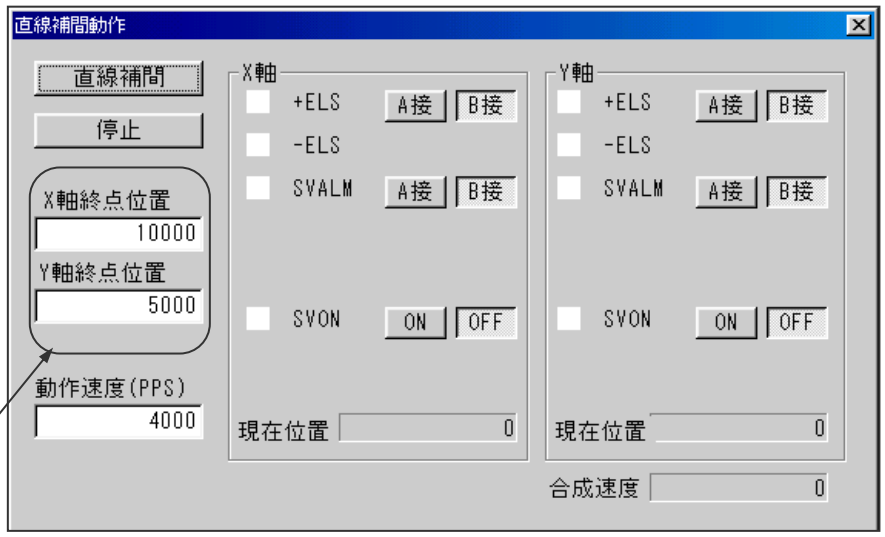
カウンタリセット ボタンをクリックすると，現在位置を ”0” にできます.

停止 ボタンで動作を停止することができます.

(5)直線補間動作

高速で直線補間動作を行います．合成速度は一定です.

[直線補間動作画面]



原点復帰動作の時と同様に，センサの接続等を確認してから動作を開始して下さい.

X1 軸と Y1 軸の終点位置を設定します.

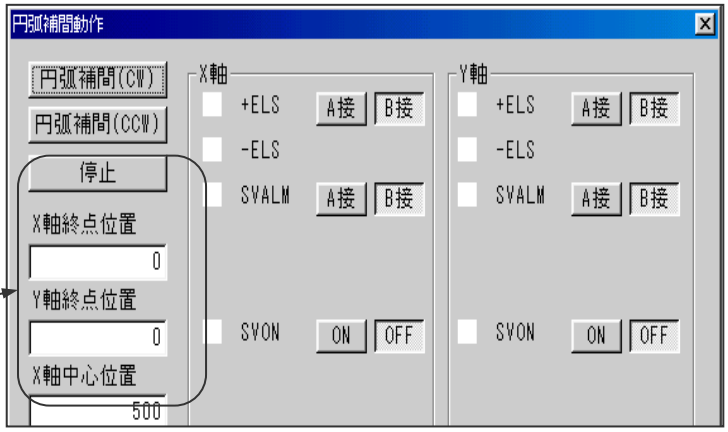
直線補間 ボタンをクリックし，直線補間動作を行います.

停止 ボタンで動作を停止することができます.

(6)円弧補間動作

ベース速度で円弧補間動作を行います.（ベース速度 ＝ 500(PPS), 合成速度一定制御 ON）

［ 円弧補間動作画面 ］



原点復帰動作の時と同様に, センサの接続等を確認してから動作を開始して下さい.

X1 軸と Y1 軸の終点位置, 中心位置を設定します.

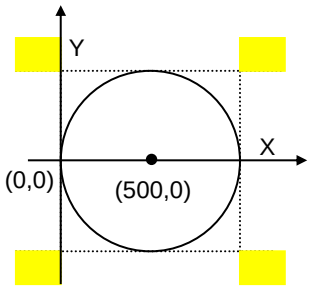
（注）現在点を始点とし, この点から見た終点座標値を終点位置とします.

始点からみた円の中心座標を中心位置とします.

終点値が(0,0)の場合は真円になります.

終点座標が円周上にない場合, X1 軸または Y1 軸が終点位置に達したところから終点引き込みを開始します.

ただし, 右図の ■ 部分に終点位置を指定した場合は停止しません.



設定例

円弧補間 ボタンをクリックし, 円弧補間動作を行います.

停止 ボタンで動作を停止することができます.

5. ポート資料

ポートの説明では，ビット毎に各種の機能が割り振られていますが，この表記については次の通りです．

英数字の意味	英数字は設定及び読込ビットの呼称です．
数値 ‘0/1’	読込ビットでは，個々の状態(1/0) が読込まれます． 設定ビットではこの値を書込み，読込時にはこの値が読込まれます．
英字 ‘x’	設定ビットでは ‘0’ を書込み，読込時にはこのビットを無視します．
英字 ‘n’	軸名称
記号 ‘*’	不定

5.1 PCI コンフィグレーションレジスタ

CPD5212 のPCIコンフィギュレーションレジスタの情報が必要な場合は，別途ご請求ください．

5.2 ポート及びレジスタアクセス

本節ではポートへ書込み及びポートからの読出しをドライバ関数により行う方法を解説します．
ドライバ関数により直接ポートにアクセスする事により自在な運用が行えます．
各軸に CMD,BUF0,BUF1,MSTS,SSTS ポートがあります．
PCL はコマンド，データを CMD,BUFx ポート経由で指定するレジスタに書くことによって，動作条件の書込み，読出しを行います．動作開始，停止などは CMD ポートに書込むことにより直接行います．

5.2.1 CMD,BUFx 書込み，読出し方法

CMD ポートに与えられるコマンドは次の 3 通りに分類されます．

(1) 動作コマンド・・・・・・・・・データを伴わないコマンド．(スタート，ストップ，速度変更などのコマンド)
(2) 制御コマンド・・・・・・・・・データを伴わないコマンド．
(カウンタリセット，カウンタラッチ，偏差カウンタクリア，プリレジスタ制御などのコマンド)
(3) レジスタ制御コマンド・・BUF0,1 レジスタ書込みデータを伴うコマンド．あるいはレジスタデータ読出しコマンド．
(コマンド書込み後，BUF0,1 にレジスタデータがセットされる)

(1) データを伴わない CMD 形式

動作および制御コマンド形式は図 5.1-1 の通りです.

図 5.2-1 CMD ポートの形式

■発行手順:CMD 部と SEL 部を一括して CMD ポートへ書く(X 軸:Board_ADR＋0, Y 軸: Board_ADR＋8, ･･･)

■CMD 部:動作コマンドあるいは制御コマンド.

■SEL 部:原則として動作軸を指定する. ただし, 全ての SEL ビットが 0 の場合は自軸指定と同じ.

動作コマンドあるいは制御コマンドで軸指定の際に考慮すべき場合があります.

◆補間制御以外の場合

自軸を含み他軸も指定した場合は, 他軸も同じコマンドで動作する.

(同時実行となる. あるいはスタートコマンドの場合は同時スタートとなる.)

他軸のみ指定してコマンドを書いたときは, 他軸がこのコマンドで動作する.

＜注意＞ 複数の PCL にまたがる同時スタートは「同時スタートコマンド(06h)」を使用します.

◆補間制御の場合

自軸を含む補間軸全てを指定する. (該当軸の動作モードを補間モードにしてあること.)

(補間モードを指定しない軸を混在させてコマンドを書いた場合は, 該当モードで動作する.)

(2) レジスタ書込み, 読出し

■レジスタは次のグループに分類されます.

(1) 書込み, 読出し可能なレジスタ群 「動作レジスタ」(移動など動作に直接関係する)
「環境レジスタ」(初期に設定し動作環境を作る), その他, カウンタやコンパレータを操作するレジスタ.

(2) 読出し専用レジスタ群 拡張ステータスなど状態の読出しのみ出来るレジスタ

■レジスタ書込みの時期とグループ

(1) 一度初期化すれば良いレジスタ群 「環境レジスタ」

(2) 初期化後必要によっては操作するレジスタ群 カウンタやコンパレータを操作するレジスタ

(3) 原則として動作コマンド発行前にセットするレジスタ群 「動作レジスタ」

■「動作レジスタ」書込みの注意

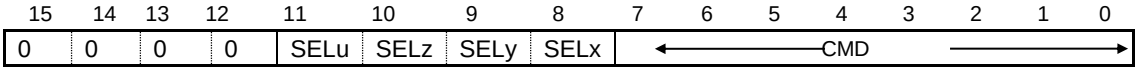
(1) 動作レジスタにはプリレジスタが置かれています.

(2) 動作レジスタの書込みは通常「プリレジスタ(2nd プリレジスタ)」に対して書き込みます.

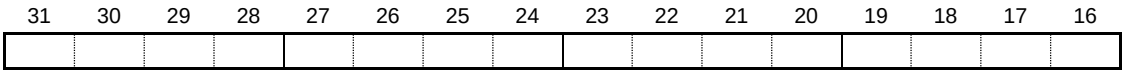
(3) 位置オーバライド・速度オーバライドなどは現在動作中の軸の「レジスタ」に直接書込みます.

(1) レジスタ書込み, 読出しのコマンド, データ形式

■CMD 形式



■BUF1 形式(上位)



■BUF0 形式(下位)

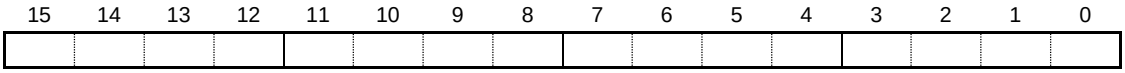


図 5.2-2 レジスタ書込み, 読出しの CMD,BUF0,BUF1 の形式

(2) レジスタ書込み

■発行手順: ①レジスタへの書込みデータを BUF1,BUF0 へ書く.
②書込み目的レジスタコマンドを CMD 部,書込み軸を SEL 部へ設定し,CMD ポートへ書込む.

■CMD 部: 書込み目的レジスタコマンド

■SEL 部: ①自軸のレジスタを指定するとき, SEL 部を 0 にする. あるいは自軸のみ指定.
②自軸と他軸を指定した場合, 他軸のレジスタには他軸の BUF0,1 の内容が書込まれる.
＜注意＞同時に複数軸のレジスタに書く時は各軸の BUFx にあらかじめデータを書いておきます.

■BUF0, 1: ①BUF0,BUF1 ポートへの書込み順は自由. (ドライバ関数使用では一括データ処理)
②数値データは右詰に書込む. (レジスタ長によらず常に 32 ビット長として書込む)
③論理データはレジスタのビット長以外のビットは 0 である

(3) レジスタ読出し

■発行手順: ①読出し目的レジスタコマンドを CMD 部,読込み軸を SEL 部へ設定し,CMD ポートへ書込む.
②読出されたデータを BUF0,BUF1 ポートから読む.

■CMD 部:読出し目的のレジスタコマンドを書く.

■SEL 部: ①自軸のレジスタを指定するとき, SEL 部を 0 にする. あるいは自軸のみ選択する.
②自軸と他軸を指定した場合,他軸のレジスタ内容も該当軸 BUF0,1 に読出される.

■BUF0, 1: ①BUF0,BUF1 ポートからの読出し順は自由. (ドライバ関数使用では一括データ処理)
②数値データは右詰に読出され符号拡張される.
③論理データはレジスタのビット長以外のビットは 0 が詰められる.
④レジスタ長によらず常に 32 ビット長として読む.

5.3 ポート表

ポートはすべてメモリマップです．BAR2 はベースアドレス 2 の略です．BAR3 はベースアドレス 3 の略です．

区分		アドレス (HEX)	読込み(IN)		書込み(OUT)	
			呼称	内　　容	呼称	内　　容
#1PCL	PCL X1軸 (第1軸)	BAR2+00	MSTS	メインステータス	CMD	コマンド
		+02	SSTS	サブステータス	OTP	不使用（予約）
		+04	BUF0	入出力バッファ IN (15- 0)	BUF0	入出力バッファ OUT(15- 0)
		+06	BUF1	入出力バッファ IN (31-16)	BUF1	入出力バッファ OUT(31-16)
	PCL Y1軸 (第2軸)	+08	MSTS	メインステータス	CMD	コマンド
		+0a	SSTS	サブステータス	OTP	不使用（予約）
		+0c	BUF0	入出力バッファ IN (15- 0)	BUF0	入出力バッファ OUT(15- 0)
		+0e	BUF1	入出力バッファ IN (31-16)	BUF1	入出力バッファ OUT(31-16)
	PCL Z1軸 (第3軸)	+10	MSTS	メインステータス	CMD	コマンド
		+12	SSTS	サブステータス	OTP	不使用（予約）
		+14	BUF0	入出力バッファ IN (15- 0)	BUF0	入出力バッファ OUT(15- 0)
		+16	BUF1	入出力バッファ IN (31-16)	BUF1	入出力バッファ OUT(31-16)
	PCL U1軸 (第4軸)	+18	MSTS	メインステータス	CMD	コマンド
		+1a	SSTS	サブステータス	OTP	不使用（予約）
		+1c	BUF0	入出力バッファ IN (15- 0)	BUF0	入出力バッファ OUT(15- 0)
		+1e	BUF1	入出力バッファ IN (31-16)	BUF1	入出力バッファ OUT(31-16)
#2PCL	PCL X2軸 (第1軸)	+20	MSTS	メインステータス	CMD	コマンド
		+22	SSTS	サブステータス	OTP	不使用（予約）
		+24	BUF0	入出力バッファ IN (15- 0)	BUF0	入出力バッファ OUT(15- 0)
		+26	BUF1	入出力バッファ IN (31-16)	BUF1	入出力バッファ OUT(31-16)
	PCL Y2軸 (第2軸)	+28	MSTS	メインステータス	CMD	コマンド
		+2a	SSTS	サブステータス	OTP	不使用（予約）
		+2c	BUF0	入出力バッファ IN (15- 0)	BUF0	入出力バッファ OUT(15- 0)
		+2e	BUF1	入出力バッファ IN (31-16)	BUF1	入出力バッファ OUT(31-16)
	PCL Z2軸 (第3軸)	+30	MSTS	メインステータス	CMD	コマンド
		+32	SSTS	サブステータス	OTP	不使用（予約）
		+34	BUF0	入出力バッファ IN (15- 0)	BUF0	入出力バッファ OUT(15- 0)
		+36	BUF1	入出力バッファ IN (31-16)	BUF1	入出力バッファ OUT(31-16)
	PCL U2軸 (第4軸)	+38	MSTS	メインステータス	CMD	コマンド
		+3a	SSTS	サブステータス	OTP	不使用（予約）
		+3c	BUF0	入出力バッファ IN (15- 0)	BUF0	入出力バッファ OUT(15- 0)
		+3e	BUF1	入出力バッファ IN (31-16)	BUF1	入出力バッファ OUT(31-16)
#3PCL	PCL X3軸 (第1軸)	+40	MSTS	メインステータス	CMD	コマンド
		+42	SSTS	サブステータス	OTP	不使用（予約）
		+44	BUF0	入出力バッファ IN (15- 0)	BUF0	入出力バッファ OUT(15- 0)
		+46	BUF1	入出力バッファ IN (31-16)	BUF1	入出力バッファ OUT(31-16)
	PCL Y3軸 (第2軸)	+48	MSTS	メインステータス	CMD	コマンド
		+4a	SSTS	サブステータス	OTP	不使用（予約）
		+4c	BUF0	入出力バッファ IN (15- 0)	BUF0	入出力バッファ OUT(15- 0)
		+4e	BUF1	入出力バッファ IN (31-16)	BUF1	入出力バッファ OUT(31-16)
	PCL Z3軸 (第3軸)	+50	MSTS	メインステータス	CMD	コマンド
		+52	SSTS	サブステータス	OTP	不使用（予約）
		+54	BUF0	入出力バッファ IN (15- 0)	BUF0	入出力バッファ OUT(15- 0)
		+56	BUF1	入出力バッファ IN (31-16)	BUF1	入出力バッファ OUT(31-16)
	PCL U3軸 (第4軸)	+58	MSTS	メインステータス	CMD	コマンド
		+5a	SSTS	サブステータス	OTP	不使用（予約）
		+5c	BUF0	入出力バッファ IN (15- 0)	BUF0	入出力バッファ OUT(15- 0)
		+5e	BUF1	入出力バッファ IN (31-16)	BUF1	入出力バッファ OUT(31-16)
予 約		+60～+7e		予 約		予 約

(次ページに続く)

区分	アドレス (HEX)	読込み(IN)		書込み(OUT)	
		呼 称	内 容	呼 称	内 容
	BAR3+00	---	不使用(予約)	ELSPOL	各軸ELS極性設定
	+02	C4STA	CMP4条件成立STA出力設定状態	C4STA	CMP4条件成立STA出力設定
	+04	C5STP	CMP5条件成立STP出力設定状態	C5STP	CMP5条件成立STP出力設定
	+06	COTSEL1	J1,J2CMP(X1-U1)出力設定状態	COTSEL1	J1,J2CMP(X1-U1)出力設定
	+08	COTSEL2	J4CMP(X2-U3)出力設定状態	COTSEL2	J4CMP(X2-U3)出力設定
	+0a	---	不使用(予約)	OPTRST	オプションポート初期化
	+0c	BDIEBL	ボード割込設定状態	BDIEBL	ボード割込出力設定
	+0e	BDINTS	ボード割込状態	---	不使用(予約)
	+10	BID	ボードID	---	不使用(予約)
	+12	ENCFIL	A相B相 Filter設定状態	ENCFIL	A相B相 Filter
	+14	BCOD0	ボード内ローカルデータ(43h(C))	---	不使用(予約)
	+15	BCOD1	ボード内ローカルデータ(50h(P))	---	不使用(予約)
	+16	BCOD2	ボード内ローカルデータ(44h(D))	---	不使用(予約)
	+17	BCOD3	ボード内ローカルデータ(52h(R))	---	不使用(予約)
	+18	BCOD4	ボード内ローカルデータ(12h(DC2))	---	不使用(予約)
	+19	BCOD5	ボード内ローカルデータ(4Dh(M))	---	不使用(予約)
	+1e～		予 約		予 約

表 5.3-1 HPCI-CPD5212M ポート表

5.4オプションポート詳細

5.4.1 各軸 ELS 極性の設定と読込(ELPOL)

Read/Write コマンド:00h

各軸の±ELS の入力極性を設定します.

A 接:カプラに電流が流れて ELS 検出状態, B 接:カプラに電流が OFF で ELS 検出状態

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	U3ELS	Z3ELS	Y3ELS	X3ELS	U2ELS	Z2ELS	Y2ELS	X2ELS	U1ELS	Z1ELS	Y1ELS	X1ELS

ビット	ビット名	内 容
11-0	nELS	0:nELS B 接(POW ON 時),1:nELS A 接

5.4.2 コンパレータ 4 比較条件成立で同時スタート信号(STA)出力設定と読込(C4STA)

Read/Write コマンド:02h

各軸コンパレータ 4 比較条件成立時に他 PCL または他 CPD ボードに対し同時スタート信号(STA)を出力することができます.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	U3C4	Z3C4	Y3C4	X3C4	U2C4	Z2C4	Y2C4	X2C4	U1C4	Z1C4	Y1C4	X1C4

ビット	ビット名	内 容
11-0	nC4	0:n 軸コンパレータ 4 比較条件成立時, 同時スタート信号(STA)を出力しない. (POW ON 時) 1:n 軸コンパレータ 4 比較条件成立時, 同時スタート信号(STA)を出力する.

5.4.3　コンパレータ 5 比較条件成立で同時ストップ信号(STP)出力設定と読込(C5STP)

Read/Write コマンド:04h

各軸コンパレータ 5 比較条件成立時に他 PCL または他 CPD ボードに対し同時ストップ信号(STP)を出力することができます.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	U3C5	Z3C5	Y3C5	X3C5	U2C5	Z2C5	Y2C5	X2C5	U1C5	Z1C5	Y1C5	X1C5

ビット	ビット名	内　容
11-0	nC5	0:n 軸コンパレータ 5 比較条件成立時，同時ストップ信号(STP)を出力しない. (POW ON 時) 1:n 軸コンパレータ 5 比較条件成立時，同時ストップ信号(STP)を出力する.

5.4.4　X1~U1 コンパレータ 3～5 比較結果外部出力の選択設定と読込(COTSEL1)

Read/Write コマンド:06h

各軸のコンパレータ比較条件成立中の信号を外部に出力できます. 使用できるコンパレータは CMP3，CMP4，CMP5 のいずれかです. コネクタ J1(X1,Y1)，J2(Z1,U1)，J4(X1~U1)の各端子から出力可能です.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
J4U1CP1	J4U1CP0	J4Z1CP1	J4Z1CP0	J4Y1CP1	J4Y1CP0	J4X1CP1	J4X1CP0	J2U1CP1	J2U1CP0	J2Z1CP1	J2Z1CP0	J1Y1CP1	J1Y1CP0	J1X1CP1	J1X1CP0

ビット	ビット名	内　容
3-0	J1nCP1-0	00:nCMP3（POW　ON 時），01:nCMP4，10:nCMP5，11: nCMP3,4,5 の AND
7-4	J2nCP1-0	00:nCMP3（POW　ON 時），01:nCMP4，10:nCMP5，11: nCMP3,4,5 の AND
15-8	J4nCP1-0	00:nCMP3（POW　ON 時），01:nCMP4，10:nCMP5，11: nCMP3,4,5 の AND

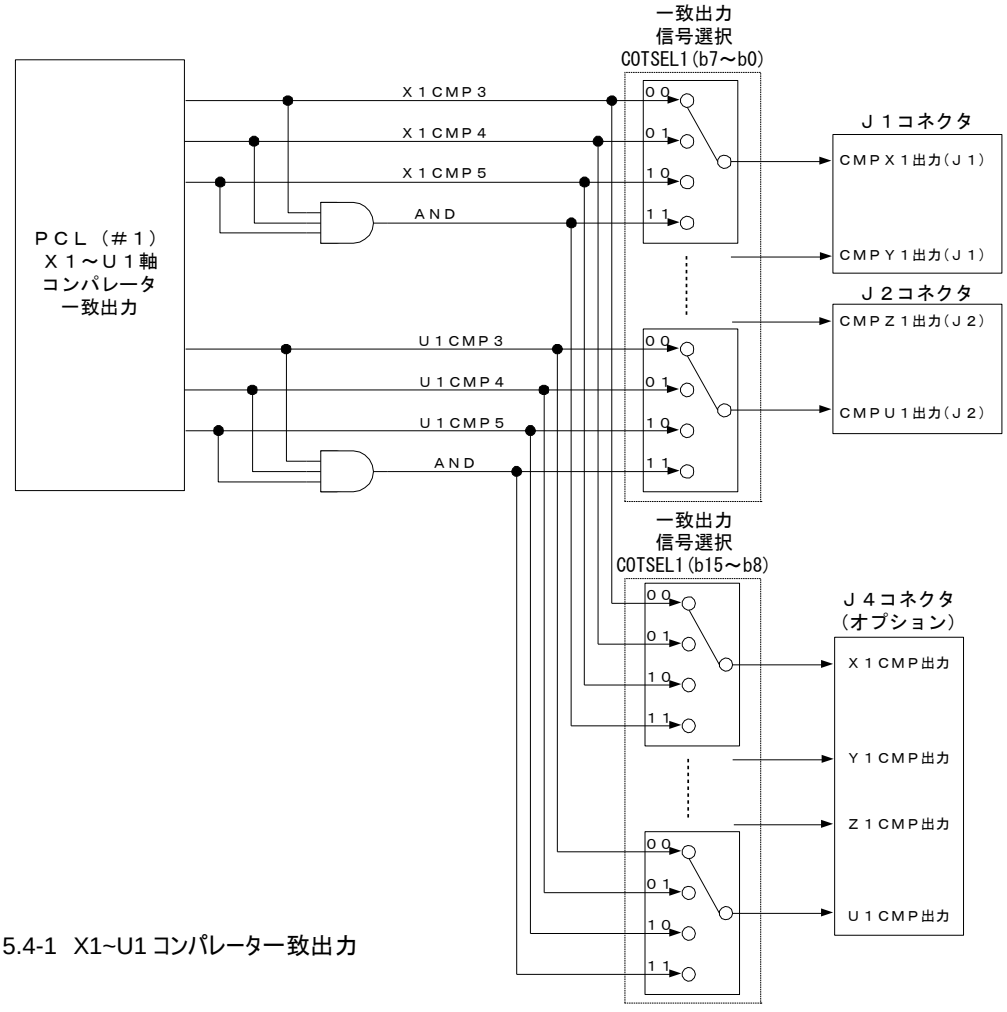


図 5.4-1　X1~U1 コンパレーター致出力

5.4.5　X2~U3 コンパレータ 3～5 比較結果外部出力の選択設定と読込(COTSEL2)

Read/Write コマンド:08h

各軸のコンパレータ比較条件成立中の信号を外部に出力できます. 使用できるコンパレータは CMP3, CMP4, CMP5 のいずれかです. コネクタ J4(X2~U3)の各端子から出力可能です. (J4 コネクタはオプション)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
J4U3CP1	J4U3CP0	J4Z3CP1	J4Z3CP0	J4Y3CP1	J4Y3CP0	J4X3CP1	J4X3CP0	J4Z2CP1	J4Z2CP0	J4Z2CP1	J4Z2CP0	J4Y2CP1	J4Y2CP0	J4X2CP1	J4X2CP0

ビット	ビット名	内　容
15-0	J4nCP1-0	00:nCMP3（POW　ON 時), 01:nCMP4, 10:nCMP5, 11: nCMP3,4,5 の AND

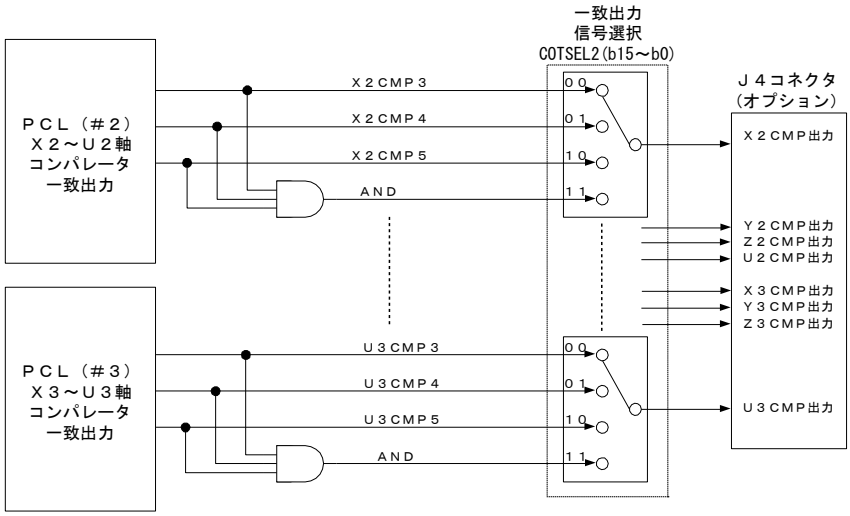


図 5.4-2　X2~U3 コンパレーター致出力(オプション)

5.4.6　オプションポート設定初期化(OPTRST)

Write コマンド:0ah

"0"を書き込むとオプションポートがリセットされる.（POW　ON 時の状態）

初期化されるオプションポートは次表の通りです.

アドレス	ポート名	ポートでの設定機能	記　事
BAR3+00	ELSPOL	各軸 ELS 極性	全軸 B 接
+02	C4STA	CMP4 条件成立 STA 出力設定	全軸出力禁止
+04	C5STP	CMP5 条件成立 STP 出力設定	全軸出力禁止
+06	COTSEL1	J1,J2,J4CMP(X1-U1)出力設定	全軸 CMP3 出力
+08	COTSEL2	J4CMP(X2-U3)出力設定	全軸 CMP3 出力
+0c	BDIEBL	ボード割込出力設定	割込み禁止

5.4.7　ボード割込マスクの設定と読込(BDIEBL)

Read/Write コマンド:0ch (Windows では使用禁止, デバイスドライバ内で使用しているため)

ボードから PCI Bus への割込みマスクを設定します.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	IMASK

ビット	ビット名	内　容
0	IMASK	0 :割込みマスク(割込み禁止) (POW ON 時), 1:割込みアンマスク(割込み許可)

5.4.8 ボード割込状態読込(BDINTS)

Read コマンド:0eh

PCL からの割込み要因を表します. (Windows では使用禁止, デバイスドライバ内で使用しているため)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	*	*	*	*	*	#3PCL	#2PCL	#1PCL	0	0	0	PCLINT

ビット	ビット名	内 容	記 事
0	PCLINT	0 :割込み発生なし, 1:割込み発生中	bit6-4 の OR 結果
4	#1PCL	0 :X1~U1 割込み発生なし, 1:割込み発生中	該当軸の割り込み要因読出しでクリア
5	#2PCL	0 :X2~U2 割込み発生なし, 1:割込み発生中	該当軸の割り込み要因読出しでクリア
6	#3PCL	0 :X3~U3 割込み発生なし, 1:割込み発生中	該当軸の割り込み要因読出しでクリア

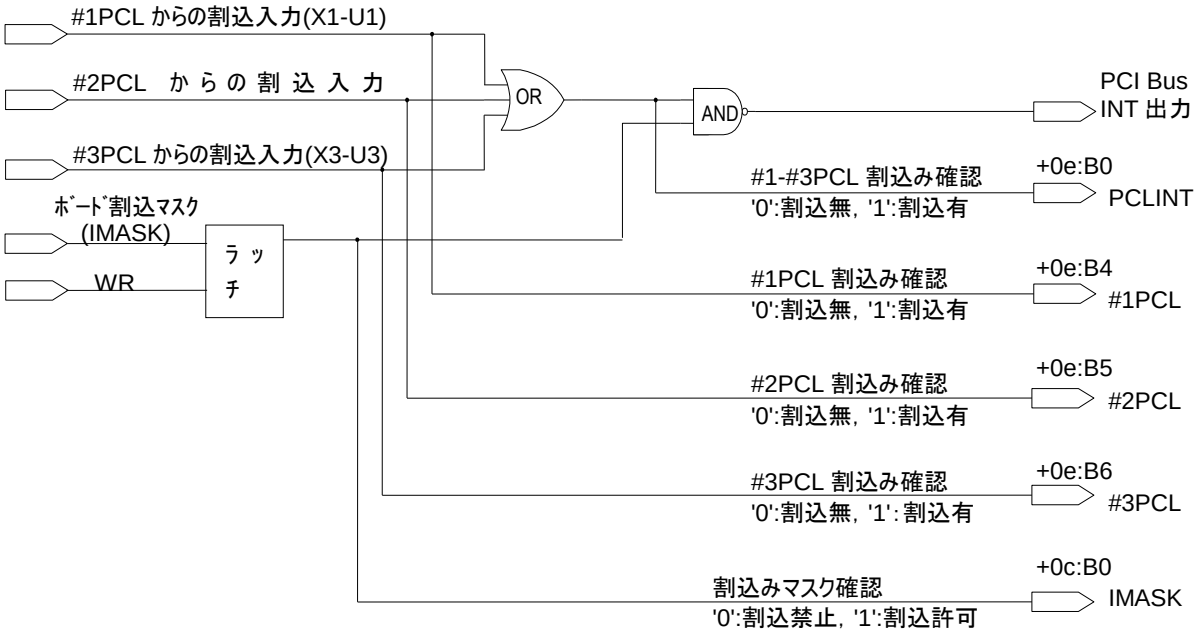


図 5.4-3 割り込み要因

5.4.9 ボード ID(BID)

Read コマンド:10h

ボード ID 設定用ロータリースイッチの値を読出します.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	*	*	*	*	*	*	*	*	ID3	ID2	ID1	ID0

ビット	ビット名	内 容
3-0	ID3-0	ボード ID 設定用ロータリースイッチの値(出荷状態は=0)

5.4.10 エンコーダフィルタ設定(ENFIL)

Read/Write コマンド:12h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	UFIL	ZFIL	YFIL	XFIL	0	0	0	0	UCRECT	UCRECT	UCRECT	UCRECT

ビット	名 称	内 容
3-0	nCRECT	0:エンコーダ入力波形補正なし(POW ON 時), 1:エンコーダ入力波形補正
11-8	nFIL	0:エンコーダ入力フィルタなし(POW ON 時), 1:エンコーダ入力 0:50ns フィルタ有

5.4.11 ボードコード読み出し

Read コマンド:14h, 15h, 16h, 17h, 18h, 19h
ボード内の固有コードが読み取れます.

アドレス	呼　称	ビット / 機能 対応
BAR3+14	BCOD0	43h(C)
+15	BCOD1	50h(P)
+16	BCOD2	44h(D)
+17	BCOD3	52h
+18	BCOD4	12h
+19	BCOD5	4Dh(M)

6. 更新履歴

日付	版	更新内容
2014/02/13	5.00	マニュアル構成変更による HPCI-CPD5212M ユーザーズマニュアル <ソフトウェア編 Windows 版> 新規作成
2016/10/07	5.01	対応 OS 変更(Windows10 対応)