PC／104 bus CPD ボードシリーズ

HPC104－CPD132

ユーザーズマニュアル

〈USB→PC/104 Bridge 編〉

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/

株式会社 ハイパーテック
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 1.2 版 2011 年 5 月 30 日発行

目　次

1. 注意事項

1.1 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，2 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，予めご了承下さい．

1.2 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，予めご了承下さい．

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，本マニュアル（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，予めご了承下さい．

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，予めご了承下さい．

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．また本マニュアル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，予めご了承下さい．

1.3 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．本マニュアルは，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，本マニュアルは，本マニュアルが添付された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．

1.3.1 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．

1.3.2 対応 OS

⚠ **警 告**

⚠ 本ソフトウェアは Windows7 の各エディション，Windows Vista の各エディション，
Windows XP(32bit)の各エディション，Windows2000 Professional，Windows98 Second Edition に対応しています．

1.3.3 対応機種

注 意

HUSB-BRG1040 及び HUSB-BRG1042 は Universal Serial Bus 1.1 または Universal Serial Bus 2.0 を装備した PC/AT 互換機で使用できます.

本ソフトウェアは HPC104-CPD132 を HUSB-BRG1040 及び HUSB-BRG1042 経由で使用する場合，使用できます.

1.3.4 試運転・調整

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.

1.4 マニュアル構成

HUSB-BRG1040(またはHUSB-BRG1042)ボード経由でPC/104バスのCPDシリーズボードを制御する場合には，本マニュアル以外に，次のマニュアルが添付されています.

(1) [個別ボード名]ユーザーズマニュアル < 個別ボード編 >

個々のCPDボードについて，次の項目について説明しています.

- ハードウェアに関する情報
- 添付ソフトウェアのインストール方法
- サンプルソフトの操作
- 「動かしてみる」の操作
- その他ボード固有な機能

（サンプルソフトの操作，「動かしてみる」の操作）：この内容については，本書を参照して下さい.

(2) CPDボードシリーズ ユーザーズマニュアル < ソフトウェア編 >

CPDボードシリーズの次のソフトウェアについて説明しています.
ライブラリ関数(ライブラリ関数レベル1：VC++, VB, DOS)
ドライバ関数(デバイスドライバI/F用ライブラリ：VC++, VB, DOS)

(3) CPDボードシリーズ ユーザーズマニュアル < 共通編 >

- CPDボードシリーズに共通した部分について，チュートリアル形式で説明をしています.
- CPDボードの基本的な運用方法(ライブラリ関数でサンプル表記)
- CPD ボードのより応用的な解説
  (ドライバ関数でサンプル表記・・より自在な運用をするためには，参照する必要があります)
- PCL に基づいた各種レジスタ説明
- その他

(4) USB-PC/104ブリッジボードマニュアル

ブリッジボードについて，次の項目について説明しています.

- ハードウェアに関する情報
- ソフトウェア仕様
- 製品仕様

2. 概　要

HUSB-BRG1040 または HUSB-BRG1042 ボード(以下 BRG)にスタックされた「HPC104-CPD132」(以下 CPD)ボードを制御するためのドライバ I/F 用 DLL として「hubcp130.dll」が提供されます.
この DLL は，Windows 7, Windows Vista, Windows XP, Windows2000, Windows98SE で使用します.

3. ソフトウェアの構成

(1) BRG 用デバイスドライバおよびドライバ関数
「hubcp130. dll」を使用するには，BRG のデバイスドライバおよび BRG のドライバ関数が OS にインストールされている必要があります.
BRG のデバイスドライバのインストールについては BRG のマニュアルをご参照ください.

(2) CPD ボード用ドライバ関数
CPD の I/O ポートを制御する各種関数を「CPD ドライバ関数」と称します.
hubcp130.dll

(3) CPD ボード用ライブラリ関数レベル 1
アプリケーションプログラム用の「ドライバ関数を使用した特定機能処理を行うライブラリ関数」であり，ソースプログラムで提供されます. このライブラリ関数の内容は自由に変更できます.
ドライバ関数と対比して「ライブラリ関数」と称します.

- ライブラリ関数
  - hubcp13l.c, hubcp13l1.h ・・・ Microsoft Visual C++ (6.0 以上)用
  - hubcp13l.bas ・・・ Microsoft Visual Basic (6.0)用

3.1 ソフトウェア関連図

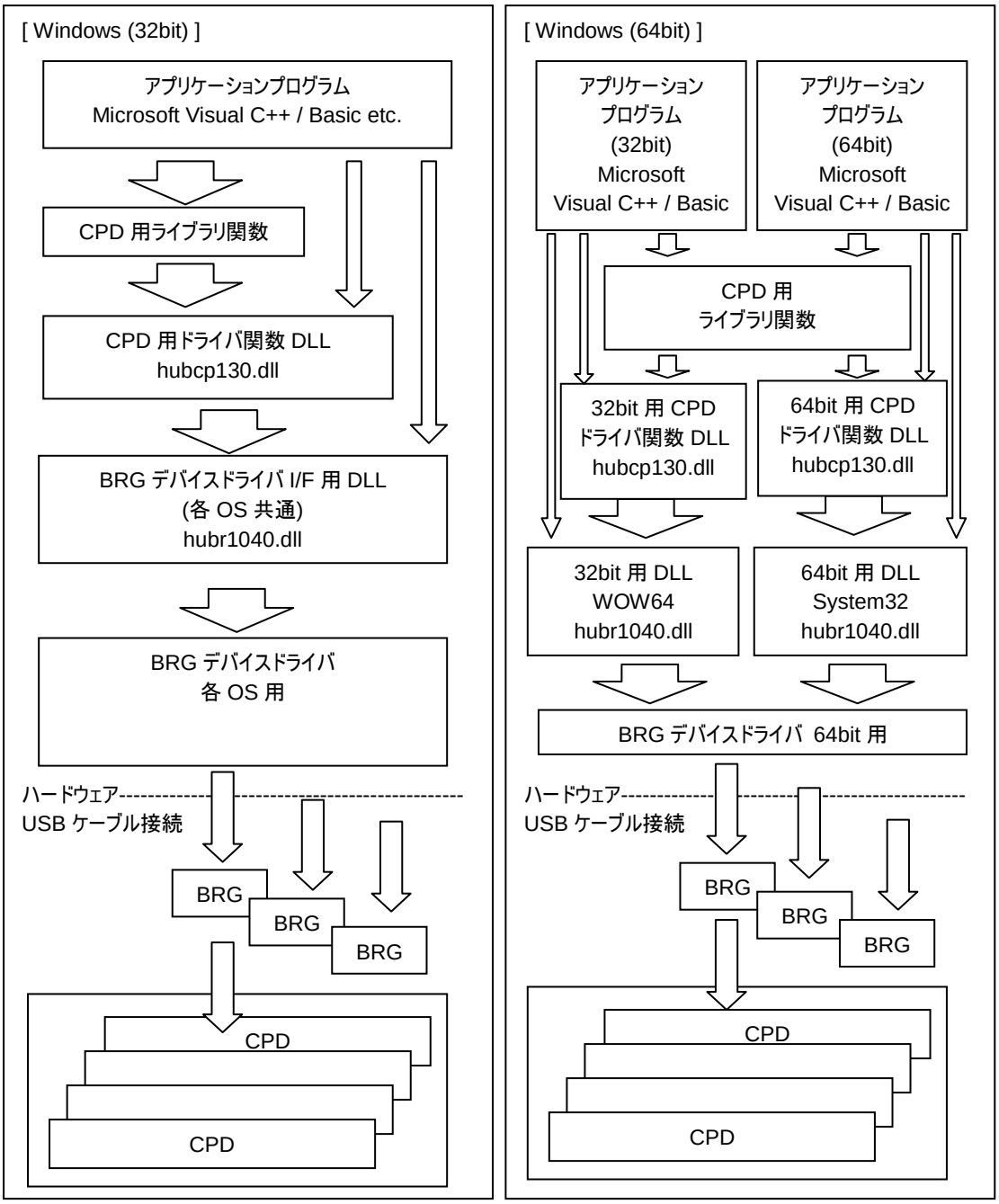


図 2.1-1 ソフトウェアの構成

3.2 ソフトウェアの使用方法

「hubcp130.dll」を実行ファイルと同じフォルダにコピーして使用します.

4. ボードアクセス方法

4.1 デバイス情報構造体

ドライバ関数・ライブラリ関数群では複数の CPD を制御することができます.
ある 1 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, 制御する CPD のデバイスハンドル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.

4.2 デバイス情報構造体

ボード認識のために次に示す HUBDEVINFO 型構造体を, ボード枚数最大 16 枚として, 使用枚数分用意します.
BRG のデバイスハンドルは BRG のデバイスオープンにより得られます.

[C言語: Visual C++]

```
typedef struct _HUBDEVINFO {      // デバイス情報
    DWORD     dwBHndl;            // 使用する BRG のデバイスハンドル
    DWORD     dwIoAdr;            // CPD ボードアドレス
    DWORD     dwIrqNo;            // 使用する IRQ 番号
    DWORD     dwRsrvd;            // 予約
} HUBDEVINFO, * PHUBDEVINFO;
```

[Visual Basic]

```
Public Type HUBDEVINFO
    dwBHndl     As Long     ' 使用する BRG のデバイスハンドル
    dwIoAdr     As Long     ' CPD ボードアドレス
    dwIrqNo     As Long     ' 使用する IRQ 番号
    dwRsrvd     As Long     ' 予約
End Type
```

4.3 ボードアクセス準備手順と終了処理

4.3.1 ドライバ関数

■ 準備手順

(1) BRG のデバイスオープン・・BRG ドライバ関数
制御したい CPD をスタックした BRG のデバイスオープンを行います.
- ◆ br1040_GetDeviceCount() ・・・BRG の枚数取得
- ◆ br1040_GetDeviceInfo() ・・・BRG のデバイス情報取得
- ◆ br1040_OpenDevice() ・・・BRG のデバイスオープン

この処理は, ボード個々に対して行います.

(2) ボード毎にデバイスオープン・・CPD ドライバ関数
制御したい CPD のデバイス情報をデバイスオープン関数に渡します.
この結果そのCPD がオープンされ, デバイスオープン関数はこのボードにアクセスする為のデバイスハンドル値を返してきます. この処理は, ボード個々に対して行います.
- ◆ hubcp130_OpenDevice() ・・・ボードのオープン処理

(3) ボード・オプションポートの初期設定
ボードの各軸初期化を行う前に, 次の関数でボード・オプションポートの設定を行います.
- ◆ hubcp130_wPortB() ・・・オプションポート使用条件の設定

(4) 各ボード・各軸の初期化
上記設定以降に, 使用する全ボードの各軸の初期化を行います.

■ 終了処理・・・CPD ドライバ関数, BRG ドライバ関数

(5) オープンしたデバイスの「クローズ処理」
全ての処理が終了してアプリケーションを終了する場合には, 全デバイスの「クローズ処理」を行います.
この処理は, オープンしたボード個々に対して行います.
- ◆ hubcp130_CloseDevice() ・・・ボードのクローズ処理
- ◆ br1040_CloseDevice() ・・・BRG のデバイスクローズ

4.3.2 ライブラリ関数

■ 準備手順

(1) BRG のデバイスオープン・・BRG ドライバ関数

制御したいCPDをスタックしたBRGのデバイスオープンを行います．この部分はBRGのドライバ関数で処理します．

- ◆ br1040_GetDeviceCount() ・・・BRG の枚数取得
- ◆ br1040_GetDeviceInfo() ・・・BRG のデバイス情報取得
- ◆ br1040_OpenDevice() ・・・BRG のデバイスオープン

この処理は，ボード個々に対して行います．

(2) ボード毎にデバイスオープン・・CPD ライブラリ関数

制御したい CPD のデバイス情報をデバイスオープン関数に渡します．

この結果そのCPDがオープンされ，デバイスオープン関数はこのボードにアクセスする為のデバイスハンドル値を返してきます．以降は,このデバイスハンドルを使用して,その CPD にアクセスすることができるようになります．これにより,通常動作としての各軸パルス出力動作等が可能となります．この処理は，ボード個々に対して行います．

- ◆ hubcp132_DevOpen()・・・ボードのオープン処理とボードのオプションポート設定，各軸の動作条件の設定ドライバ関数処理の(2)～(4)をまとめたものです．初期化の条件はこの関数内で直接行っています．
  内容についてはユーザーズマニュアル＜ソフトウェア編＞を参照下さい．

■ 終了処理・・CPD ライブラリ関数，BRG ドライバ関数

(3) オープンしたデバイスの「クローズ処理」

全ての処理が終了してアプリケーションを終了する場合には，全デバイスの「クローズ処理」を行います．

この処理は，オープンしたボード個々に対して行います．

- ◆ hubcp132_DevClose() ・・・ボードのクローズ処理
- ◆ br1040_CloseDevice() ・・・BRG のデバイスクローズ処理

5. ドライバ関数

5.1 ドライバ関数一覧

ドライバ関数は次表に示す 8 種類です.

No	関 数 名	機　　能
1	ubcp130_OpenDevice()	デバイスのオープン
2	ubcp130_CloseDevice()	デバイスのクローズ
3	ubcp130_rMstsW()	メインステータスの読出
4	ubcp130_rSstsW()	サブステータスの読出
5	ubcp130_wCmdW()	制御コマンドの書込
6	ubcp130_rReg()	レジスタの読出
	ubcp130_wReg()	レジスタへ書込
7	ubcp130_rPortB()	オプションポートの読出
	ubcp130_wPortB()	オプションポートへ書込
8	ubcp130_rBufDW()	入出力バッファの読出
	ubcp130_wBufDW()	入出力バッファへ書込

表 5.1-1　関数一覧

5.2 ドライバ関数の戻り値一覧

No	記 号 表 記	戻り値 (HEX)	異常内容と原因
1	NO_ERROR	0000	正　常
2	NOT_FOUND	0001	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	0002	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(多重のオープンは禁止) ◎複数ボード使用の場合,オープンするデバイス情報の更新を確認します.
4	NOT_MEMORY	0004	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 ◎システムリソース(OS 用メモリ)の不足
5	INVALID_HANDLE	0008	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	0010	入出力ポートが使用できない
7	ILLEGAL_ADDRESS	0040	不正なアドレス
8	ILLEGAL_IRQNUM	0080	不正な IRQ 番号

表 5.2-1　関数の戻り値一覧

5.3 ドライバ関数仕様

No	1	**ubcp130_OpenDevice()**	デバイスのオープン
機 能		指定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．	
VC++	書 式	DWORD WINAPI ubcp130_OpenDevice(DWORD * hDevID, HUBDEVINFO* HpcDevInfo);	
	引 数	DWORD hDevID; ・・取得するデバイスハンドルの格納エリア HUBDEVINFO* HpcDevInfo; ・・オープンするボードのデバイス情報格納アドレス	

	呼出例	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 = ubcp130_OpenDevice(&hDevID[0], &HpcDevInfo[0]); //1 番目のデバイス情報
		ret = ubcp130_OpenDevice(&hDevID[1], &HpcDevInfo[1]); //2 番目のデバイス情報	

VB	書 式	Declare Function ubcp130_OpenDevice Lib "hubcp130.dll" _ (ByRef hDevID As Long, HpcDevInfo As HUBDEVINFO) As Long	
	引 数	ByRef hDevID As Long ・・取得するデバイスハンドルの格納エリア HpcDevInfo As HUBDEVINFO ・・オープンするボードのデバイス情報格納アドレス	
	呼出例	Dim ret As Long '関数の戻り値 Dim hDevID(2) As Long 'デバイスハンドル取得エリア ret = ubcp130_OpenDevice(hDevID(0), HpcDevInfo(0)) '1 番目のデバイス情報 ret = ubcp130_OpenDevice(hDevID(1), HpcDevInfo(1)) '2 番目のデバイス情報	

No	2	**ubcp130_CloseDevice()**	デバイスのクローズ
機 能		指定したデバイスハンドルを持つ 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．	
VC++	書 式	DWORD WINAPI ubcp130_CloseDevice(DWORD hDevID);	
	引 数	DWORD hDevID ・・クローズするボードのデバイスハンドル	

	呼出例	DWORD ret; //関数の戻り値
		ret = ubcp130_CloseDevice(hDevID);	

VB	書 式	Declare Function ubcp130_CloseDevice Lib "hubcp130.dll" (ByVal hDevID As Long) As Long	
	引 数	ByVal hDevID As Long ・・クローズするボードのデバイスハンドル	
	呼出例	Dim ret As Long '関数の戻り値 ret = ubcp130_CloseDevice(hDevID)	

No	3	**ubcp130_rMstsW()**	メインステータス読出
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，指定エリアに格納します．	
VC++	書 式	DWORD WINAPI ubcp130_rMstsW(DWORD hDevID, WORD axis, WORD* wMsts);	
	引 数	DWORD hDevID ・・対象デバイスのデバイスハンドル WORD axis ・・軸指定［ 0:X，1:Y ］ WORD* wMsts ・・読出したデータが格納されるエリアのアドレス	

	呼出例	DWORD ret; //関数の戻り値
		WORD msts; //メインステータス
		ret = ubcp130_rMstsW(hDevID, 1, &msts); // Y 軸	

VB	書 式	Declare Function ubcp130_ rMstsW Lib "hubcp130.dll"(ByVal hDevID As Long, ByVal axis As Integer, ByRef wMsts As Integer) As Long	
	引 数	ByVal hDevID As Long ・・対象デバイスのデバイスハンドル ByVal axis As Integer・・軸指定［ 0:X，1:Y ］ ByRef wMsts As Integer・・読出したデータが格納されるエリアのアドレス	
	呼出例	Dim ret As Long '関数の戻り値 Dim msts As Integer 'メインステータス ret = ubcp130_rMstsW(hDevID, 1, msts) ' Y 軸	

No	4	**ubcp130_rSstsW()**	サブステータス読出
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，指定エリアに格納します.	
VC++	書　式	DWORD WINAPI ubcp130_rMstsW(DWORD hDevID, WORD axis, WORD* wSsts);	
	引　数	DWORD　hDevID　・・対象デバイスのデバイスハンドル WORD　axis　・・軸指定［ 0:X, 1:Y ］ WORD*　wSsts　・・読出したデータが格納されるエリアのアドレス	
	呼出例	DWORD　ret;　//関数の戻り値 WORD　ssts;　//サブステータス ret = ubcp130_rSstsW(hDevID, 1, &ssts);　// Y 軸	
VB	書　式	Declare Function ubcp130_ rSstsW Lib "hubcp130.dll"(ByVal hDevID As Long, ByVal axis As Integer, ByRef wSsts As Integer) As Long	
	引　数	ByVal hDevID　As Long　・・対象デバイスのデバイスハンドル ByVal axis　As Integer・・軸指定［ 0:X, 1:Y ］ ByRef wSsts As Integer・・読出したデータが格納されるエリアのアドレス	
	呼出例	Dim ret　As Long　'関数の戻り値 Dim ssts　As Integer　'サブステータス ret = ubcp130_rSstsW(hDevID, 1, ssts)　' Y 軸	

No	5	**ubcp130_wCmdW ()**	制御コマンド書込
機 能		デバイスハンドルで指定されたボードの指定軸コマンドバッファへ制御コマンドデータを書込みます.	
VC++	書 式	DWORD WINAPI ubcp130_wCmdW(DWORD hDevID, WORD axis, WORD wCmd);	
	引 数	DWORD hDevID ・・対象デバイスのデバイスハンドル WORD axis ・・軸指定 [0:X, 1:Y] WORD wCmd・・コマンドデータ	

	呼出例	DWORD ret; //関数の戻り値
		ret = ubcp130_wCmdW (hDevID, 1, 0x4a); // Y 軸減速停止	

VB	書 式	Declare Function ubcp130_wCmdW Lib "hubcp130.dll" (ByVal hDevID As Long, ByVal axis As Integer, ByVal wCmd As Integer) As Long	
	引 数	ByVal hDevID As Long ・・対象デバイスのデバイスハンドル ByVal axis As Integer・・軸指定 [0:X, 1:Y] ByVal wCmd As Integer・・コマンドデータ	
	呼出例	Dim ret As Long '関数の戻り値 ret = ubcp130_wCmdW (hDevID, 1, &H4A); 'Y 軸減速停止	

[コマンドデータ]

(1) コマンドデータの内容

b15	b14	b13	b12	b11	b10	b9	b8	b7~b0
0	0	0	0	0	0	コマンド実行軸(SELx)		コマンドコード (code)
						Y	X	

(2) 実行軸の指定(SELx)

個々の軸毎に コマンドコードを書込む場合・・この 2 ビットは'0'

2 軸に同一コマンドコードを書込む場合・・X 軸(axis=0)に対してSELxのビットで書込む軸を指定.

(3) コマンドコード一覧表

code	コマンド内容	code	コマンド内容
0x04	ソフトウェアリセット	0x29	ラッチ入力代行
0x05	非常停止	0x2a	自軸のみ, CSTA 入力と同じ
0x06	CSTA 出力(同時スタート)	0x2b	動作用プリレジスタのシフト
0x07	CSTP 出力(同時ストップ)	0x2c	RCMP5 用プリレジスタのシフト
0x10	SVON OFF	0x40	直ちにFL速度へ変更
0x18	SVON ON	0x41	直ちにFH速度へ変更
0x11	SVRESET OFF	0x42	減速してFL速度へ変更
0x19	SVRESET ON	0x43	加速してFH速度へ変更
0x20	カウンタ 1 リセット	0x49	即停止
0x21	カウンタ 2 リセット	0x4a	減速停止
0x22	カウンタ 3 リセット	0x50	FL定速スタート
0x23	カウンタ 4 リセット	0x51	FH定速スタート
0x24	偏差カウンタクリア信号の出力	0x52	FH定速継続スタート後減速停止
0x25	偏差カウンタクリア信号のリセット	0x53	高速スタート
0x26	動作用プリレジスタのキャンセル	0x54	残量FL定速スタート
0x27	RCMP5 用プリレジスタのキャンセル	0x55	残量FH定速スタート
0x28	位置決め管理開始(PCS 入力代行)	0x57	残量高速スタート

表 5.3-1 コマンド一覧

No	6	**ubcp130_rReg()** **ubcp130_wReg()**	レジスタ読出 レジスタ書込
機　能		デバイスハンドルで指定されたボードの, 　読出・・指定軸の指定レジスタ内容を読出し，指定エリアに格納します. 　書込・・指定軸の指定レジスタへ，データを書込みます.	
VC++	書　式	DWORD WINAPI ubcp130_rReg(DWORD hDevID, WORD axis, BYTE byCmd, DWORD* dwReg); DWORD WINAPI ubcp130_wReg(DWORD hDevID, WORD axis, BYTE byCmd, DWORD　dwReg);	
	引　数	DWORD　hDevID;　・・対象デバイスのデバイスハンドル WORD　axis;　・・軸指定［ 0:X, 1:Y ］ BYTE　byCmd;　・・レジスタ読出／書込コマンド DWORD*　dwReg;　・・レジスタ読出：読出データの格納エリアアドレス DWORD　dwReg;　・・レジスタ書込：レジスタ書込データ	
	呼出例	DWORD　ret;　//関数の戻り値 DWORD　reg;　//レジスタのデータ ret = ubcp130_rReg(hDevID, 0, 0xc0, &reg);　// X 軸 PRMV 読出 ret = ubcp130_wReg(hDevID, 0, 0x80, 10000);　// X 軸 PRMV = 10000 書込	
VB	書　式	Declare Function ubcp130_rReg Lib "hubcp130.dll" (ByVal hDevID As Long, 　ByVal axis As Integer,　ByVal byCmd As Byte, ByRef dwReg As Long) As Long Declare Function ubcp130_wReg Lib "hubcp130.dll" (ByVal hDevID As Long, 　ByVal axis As Integer,　ByVal byCmd As Byte, ByVal dwReg As Long) As Long	
	引　数	ByVal hDevID　As Long　・・対象デバイスのデバイスハンドル ByVal axis　As Integer　・・軸指定［ 0:X, 1:Y ］ ByVal byCmd　As Byte　・・レジスタ読出／書込コマンド ByRef dwReg　As Long　・・レジスタ読出：読出データの格納エリアアドレス ByVal dwReg　As Long　・・レジスタ書込：レジスタ書込データ	
	呼出例	Dim　ret　As Long　'関数の戻り値 Dim　reg　As Long　'レジスタのデータ ret = ubcp130_rReg(hDevID, 0, &HC0, reg)　'X 軸 PRMV 読出 ret = ubcp130_wReg(hDevID, 0, &H80, 10000)　'X 軸 PRMV = 10000 書込	
備　考		次ページに「レジスタの種類と読出コマンド・書込コマンド」を記載	

[レジスタ・プリレジスタ：読出コマンド・書込コマンド]

No	内　　容	レ ジ ス タ			プリレジスタ		
		名称	コマンド		名称	コマンド	
			読出	書込		読出	書込
1	移動量，目標位置	RMV	0xd0	0x90	PRMV	0xc0	0x80
2	初速度	RFL	0xd1	0x91	PRFL	0xc1	0x81
3	動作速度	RFH	0xd2	0x92	PRFH	0xc2	0x82
4	加速レート	RUR	0xd3	0x93	PRUR	0xc3	0x83
5	減速レート	RDR	0xd4	0x94	PRDR	0xc4	0x84
6	速度倍率	RMG	0xd5	0x95	PRMG	0xc5	0x85
7	減速点	RDP	0xd6	0x96	PRDP	0xc6	0x86
8	動作モード	RMD	0xd7	0x97	PRMD	0xc7	0x87
9	円弧補間中心位置	RIP	0xd8	0x98	PRIP	0xc8	0x88
10	加速時S字区間	RUS	0xd9	0x99	PRUS	0xc9	0x89
11	減速時S字区間	RDS	0xda	0x9a	PRDS	0xca	0x8a
12	移動量補正速度	RFA	0xdb	0x9b			
13	環境設定 1	RENV1	0xdc	0x9c			
14	環境設定 2	RENV2	0xdd	0x9d			
15	環境設定 3	RENV3	0xde	0x9e			
16	環境設定 4	RENV4	0xdf	0x9f			
17	環境設定 5	RENV5	0xe0	0xa0			
18	環境設定 6	RENV6	0xe1	0xa1			
19	環境設定 7	RENV7	0xe2	0xa2			
20	カウンタ 1（指令パルス出力）	RCUN1	0xe3	0xa3			
21	カウンタ 2（エンコーダ入力）	RCUN2	0xe4	0xa4			
22	カウンタ 3（偏差カウンタ）	RCUN3	0xe5	0xa5			
23	カウンタ 4（汎用カウンタ）	RCUN4	0xe6	0xa6			
24	コンパレータ 1 用データ	RCMP1	0xe7	0xa7			
25	コンパレータ 2 用データ	RCMP2	0xe8	0xa8			
26	コンパレータ 3 用データ	RCMP3	0xe9	0xa9			
27	コンパレータ 4 用データ	RCMP4	0xea	0xaa			
28	コンパレータ 5 用データ	RCMP5	0xeb	0xab	PRCP5	0xcb	0x8b
29	イベントマスク設定	RIRQ	0xec	0xac			
30	カウンタ 1 ラッチデータ	RLTC1	0xed				
31	カウンタ 2 ラッチデータ	RLTC2	0xee				
32	カウンタ 3 ラッチデータ	RLTC3	0xef				
33	カウンタ 4 ラッチデータ	RLTC4	0xf0				
34	拡張ステータス	RSTS	0xf1				
35	エラーステータス	REST	0xf2				
36	イベントステータス	RIST	0xf3				
37	位置決めカウンタ	RPLS	0xf4				
38	EZ カウンタ，速度モニタ	RSPD	0xf5				
39	スローダウンポイント	RSDC	0xf6				
40	円弧補間歩進数	RRCI	0xfc	0xbc	PRCI	0xcc	0x8c
41	補間ステータス	RIPS	0xff				

No	7	**ubcp130_rPortB()** **ubcp130_wPortB()**	オプションポート読出 オプションポート書込
機　能		デバイスハンドルで指定されたボードの, 読出・・指定オプションポート内容を読出し, 指定エリアに格納します. 書込・・指定オプションポートへ, データを書込みます.	
VC++	書　式	DWORD WINAPI ubcp130_rPortB(DWORD hDevID, BYTE byAdrs, BYTE* byData); DWORD WINAPI ubcp130_wPortB(DWORD hDevID, BYTE byAdrs, BYTE　byData);	
	引　数	DWORD　hDevID　・・ボードのデバイスハンドル BYTE　byAdrs　・・読出オプションポート/書込オプションポート BYTE*　byData　・・ポート読出:読出データの格納エリアアドレス BYTE　byData　・・ポート書込:オプションポート書込データ	
	呼出例	DWORD　ret;　//関数の戻り値 BYTE　byData; ret = ubcp130_rPortB(hDevID, 0x10, &byData);　// 各軸 ELS 極性設定を読出 ret = ubcp130_wPortB(hDevID, 0x10, 0x01);　// X 軸のみ A 接に設定	
VB	書　式	Declare Function ubcp130_rPortB Lib "hubcp130.dll" (ByVal hDevID As Long, ByVal byAdrs As Byte, ByRef byData As Byte) As Long Declare Function ubcp130_wPortB Lib "hubcp130.dll" (ByVal hDevID As Long, ByVal byAdrs As Byte, ByVal byData As Byte) As Long	
	引　数	ByVal hDevID　As Long　・・ ボードのデバイスハンドル ByVal byAdrs　As Byte　・・ 読出オプションポート/書込オプションポート ByRef byData　As Byte　・・ ポート読出:読出データの格納エリアアドレス ByVal byData　As Byte　・・ ポート書込:オプションポート書込データ	
	呼出例	Dim　ret　As Long　'関数の戻り値 Dim　byData　As Byte ret = ubcp130_rPortB(hDevID, &H10, byData)　'各軸 ELS 極性設定を読出 ret = ubcp130_wPortB(hDevID, &H10, &H1)　'X 軸のみA接に設定	
備　考			

アドレス Base＋	読出し		書込み	
	呼称	内　容	呼称	内　容
＋10	ELPOL	各軸ELS極性設定状態読込	ELPOL	各軸ELS極性設定
＋12	DLS/PCS	DLS/PCS入力選択設定状態読込	DLS/PCS	DLS/PCS入力選択設定
＋1c	BDIEBL	割込許可状態	BDIEBL	割込許可設定
＋1e	BDINTS	割込確認	-	-

<table>
<tr><td>No</td><td>8</td><td colspan="2">ubcp130_rBufDW()
ubcp130_wBufDW()</td><td>入出力バッファ読出
入出力バッファ書込</td></tr>
<tr><td colspan="2">機 能</td><td colspan="3">デバイスハンドルで指定されたボードの,
読出・・指定軸の入出力バッファを読出し，指定エリアに格納します.
書込・・指定軸の入出力バッファにデータを書込みます.</td></tr>
<tr><td rowspan="3">VC++</td><td>書 式</td><td colspan="3">DWORD WINAPI ubcp130_rBufDW(DWORD hDevID, WORD axis, DWORD* dwData);
DWORD WINAPI ubcp130_wBufDW(DWORD hDevID, WORD axis, DWORD dwData);</td></tr>
<tr><td>引 数</td><td colspan="3">DWORD hDevID; ・・対象デバイスのデバイスハンドル
WORD axis; ・・軸指定 [0:X, 1:Y]
DWORD* dwData; ・・レジスタ読出：読出データの格納エリアアドレス
DWORD dwData; ・・レジスタ書込：レジスタ書込データ</td></tr>
<tr><td>呼出例</td><td colspan="3">DWORD ret;//関数の戻り値
DWORD dwData; //入出力バッファデータ
ret = ubcp130_rBufDW(hDevID, 0x01, &dwData); // Y 軸入出力バッファから読出
ret = ubcp130_wBufDW(hDevID, 0x01, 10000); // Y 軸入出力バッファへの書込</td></tr>
<tr><td rowspan="3">VB</td><td>書 式</td><td colspan="3">Declare Function ubcp130_rBufDW Lib "hubcp130.DLL"
(ByVal hDevID As Long, ByVal axis As Integer, ByRef dwData As Long) As Long
Declare Function ubcp130_wBufDW Lib "hubcp130.DLL"
(ByVal hDevID As Long, ByVal axis As Integer, ByVal dwData As Long) As Long</td></tr>
<tr><td>引 数</td><td colspan="3">ByVal hDevID As Long ・・対象デバイスのデバイスハンドル
ByVal axis As Integer ・・軸指定 [0:X, 1:Y]
ByRef dwReg As Long ・・レジスタ読出：読出データの格納エリアアドレス
ByVal dwReg As Long ・・レジスタ書込：レジスタ書込データ</td></tr>
<tr><td>呼出例</td><td colspan="3">Dim ret As Long '関数の戻り値
Dim dwData As Long '入出力バッファデータ
ret = ubcp130_rBufDW(hDevID, &H1, dwData) 'Y 軸入出力バッファから読出
ret = ubcp130_wBufDW(hDevID, &H1, 10000) 'Y 軸入出力バッファへの書込</td></tr>
<tr><td colspan="2">備 考</td><td colspan="3">《 レジスタ読出/書込関数との相違 》
◆レジスタ読出/書込関数・・PCL の指定軸入出力バッファを経由して目的レジスタを対象
◆入出力バッファ操作関数 ・PCL の指定軸入出力バッファとの読出/書込みです.
《 レジスタ読出関数（ubcp130_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みを行います.（同一 PCL 内の軸）
◇ubcp130_wCmdW()関数の"コマンド"で複数軸の読出み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バッファを読出
《 レジスタ書込関数（ubcp130_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バッファに所定データを書込
◇ubcp130_wCmdW()関数の"コマンド"で複数軸の読出みたいプリレジスタ・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 軸設定
コマンドコード(code)に書込コマンドを設定</td></tr>
</table>

6. 添付ソフトウェア

本製品の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.

Visual C++ (6.0)・・ C コーディング 【 subcp1300.exe 】
Visual Basic (6.0) 【 subcp1302.exe 】

6.1 サンプルプログラム

6.1.1 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 Visual C++ 6.0 以上がインストールされている必要があります.
- ◆ Visual Basic サンプルは開発ツールとして Visual Basic 6.0 がインストールされている必要があります.
- ◆ CPD のボードアドレスは”0120h”に設定して下さい. サンプルプログラムで「デフォルトアドレス」としています.
- ◆ CPD を 2 枚以上で使用する場合, ボードアドレスは重複しないようにして下さい.
- ◆ 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（Windows XP の場合）

※DLLがコピーされていない.
hubcp130.dll を実行ファイルと同じフォルダにコピーしてください.
または BRG のデバイスドライバのインストールを確認してください.

図 6.1-1 サンプルプログラムのエラーメッセージ

6.1.2 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 ］

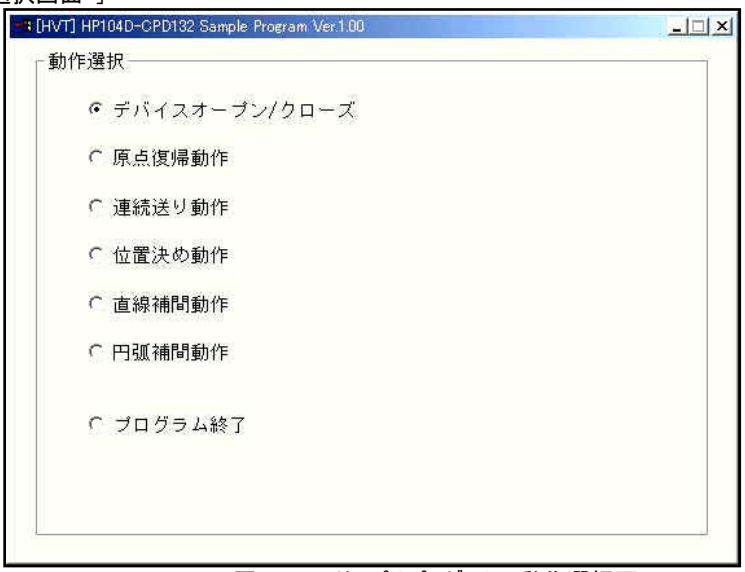


図 6.1-2 サンプルプログラムの動作選択画面

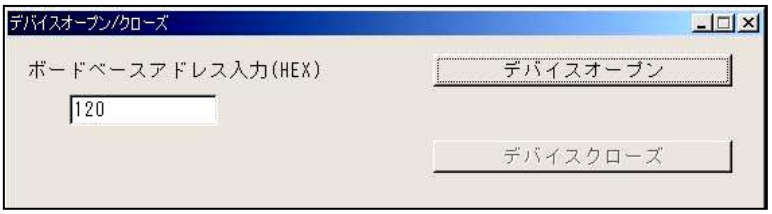
各動作を選択すると,その動作のサンプルが実行されます.
(VC++サンプルではシングルクリック, VBサンプルではダブルクリックで動作選択されます.)

(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，デバイスをオープンして，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アドレスを 0120h，軸数は 2 軸固定とし，そのボードをオープンします.
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ます.

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]



ボードアドレスが0120hではない時，ボードベースアドレス入力欄 にキー入力で正しい値を設定します.
また，ボードが2枚以上の場合には，ここで設定を変更します.

ここでボードアドレスを入力し，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，[デバイスオープン]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．

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．

■原点復帰動作の実行

次の原点復帰動作方法が選択でき，ボタンのクリックで実行を開始します．

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：OLS on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．

[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下さい．

(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]

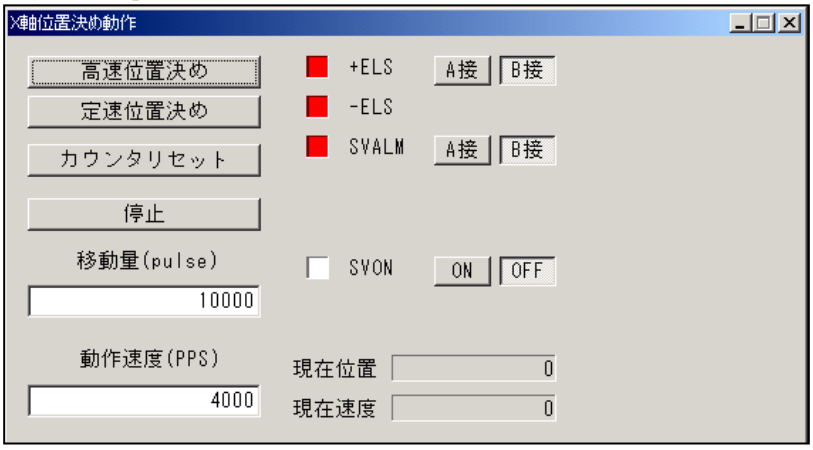
原点復帰動作の時と同様に，センサの接続等を確認してから動作を開始して下さい．

[＋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.

X軸とY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

定速で円弧補間動作を行います.（動作速度 ＝ 500(PPS)固定，周速一定制御）

[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.

X軸とY軸の終点位置，中心位置を設定します.

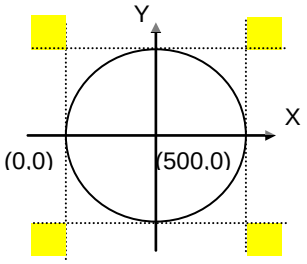
現在点を始点とし，この点から見た終点座標値を終点位置とします.

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，X 軸または Y 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.

6.2 動かしてみる

「動かしてみる」プログラムは,ボードをパソコンへ装着するだけで,最小限の動作をディスプレイ上で確認できるソフトです.

添付ソフトウェアフロッピーディスクの「(A:)¥test¥tubcp130.exe」を実行して下さい.

≪ ご注意 ≫

CPD ボードを 2 枚以上で使用する場合，ボードアドレスは重複しないようにして下さい.

ボードアドレスが重複した場合は，正常に動作しません.

本アプリケーションでは，安全の為，軸動作中の画面変更はしません.

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.

【 エラーメッセージの表示 】

※DLLがコピーされていない.

Hubcp130.dll をカレントフォルダにコピーしてください.

またはBRGのデバイスドライバのインストールを確認してください.

図 6.2-1 「動かしてみる」のエラーメッセージ

6.2.1 「動かしてみる」の動作確認画面

「動かしてみる」プログラム実行で次の画面が表示されます.

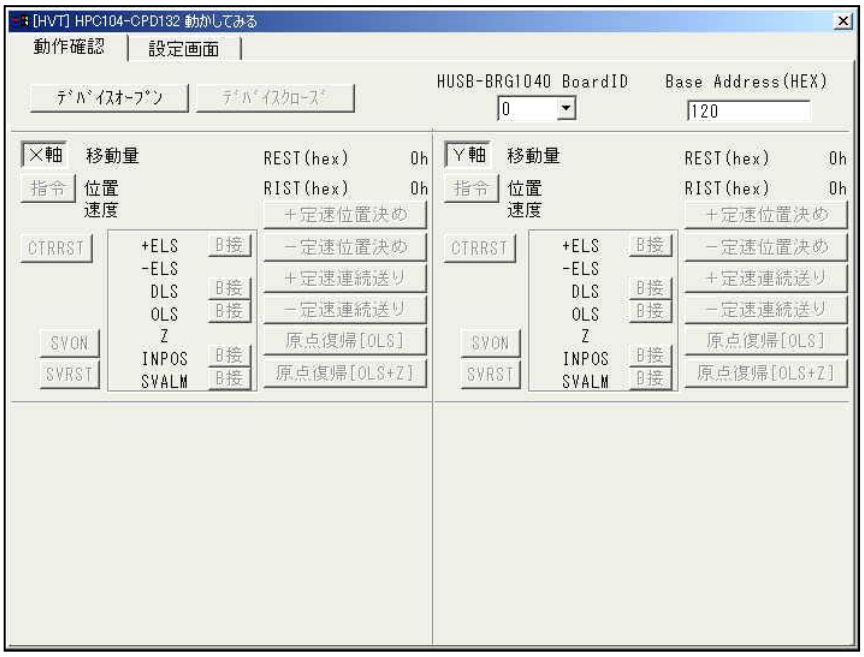


図 6.2-2 「動かしてみる」の起動時画面

(1)デバイスのオープン

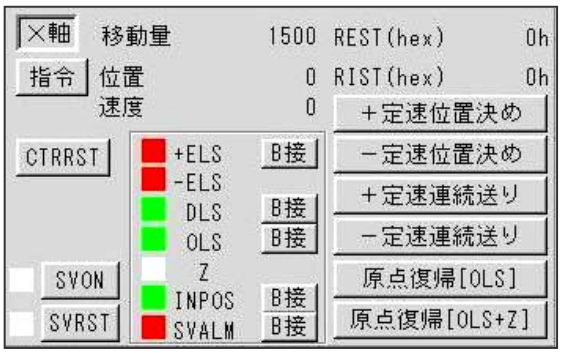
BRGのボードIDをボードアドレス設定値の確認を行い，画面表示値と異なる場合にはキー入力で変更し

ます.

① BRGのボードID選択
② ボードアドレス設定

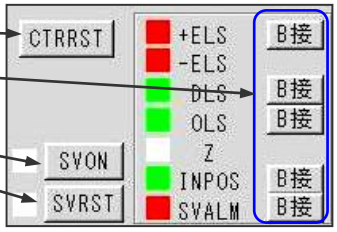
(2)個々の軸表示と動作指令

ボード上の個々の軸に対する操作は同一です.
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.



①軸の動作条件の変更と軸のステータス

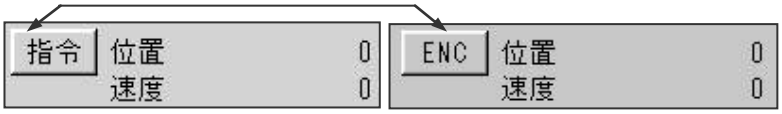
カウンタを "0" にします.
入力極性を変えます.
サーボオンします.
SVERSETをオンします.



②軸の現在位置・動作速度表示

各軸の現在位置および動作中の速度は約0.1秒毎に更新されます.
現在位置は, 「指令出力パルス」の表示と「エンコーダフィードバック」の表示が選択できます.
位置の単位はパルス, 速度は PPS で表示されます.

「 指令パルス」　－－切替－－　「エンコーダフィードバック」



③軸への動作開始・停止指令

個々の軸に対する動作は, 定速／高速位置決め動作, ±定速／高速連続送り動作, 定速／高速原点復帰動作, 停止があります. 位置決め動作, および連続送り動作の高速／定速の切り替えは設定画面で行います.

定速動作に設定

① ベース速度で＋方向へ定速位置決め.（位置決め量は設定画面）
② ベース速度で－方向へ定速位置決め.（位置決め量は設定画面）
③ ＋方向にベース速度で定速動作します.
④ －方向にベース速度で定速動作します.
⑤ －方向にベース速度で定速原点復帰します. OLS on で即停止し, 原点復帰を完了します.
⑥ －方向に動作速度で高速原点復帰します. OLS on で減速し, ベース速度まで減速後エンコーダZ相入力1回目で即停止し, 原点復帰を完了します.

高速動作に設定

① 動作速度で＋方向へ高速位置決め.（位置決め量は設定画面）
② 動作速度で－方向へ高速位置決め.（位置決め量は設定画面）
③ ＋方向に動作速度で高速動作します.
④ －方向に動作速度で高速動作します.
⑤ －方向にベース速度で定速原点復帰します. OLS on で即停止し, 原点復帰を完了します.
⑥ －方向に動作速度で高速原点復帰します. OLS on で減速し, ベース速度まで減速後エンコーダZ相入力1回目で即停止し, 原点復帰を完了します.

(注)

1. 動作中は左図の表示となります. このボタンを押せば停止します.
2. 加減速は直線加減速です.
3. DLS は有効になっていますので, 使用しない場合は"A接"にして, 入力していない状態にして下さい.
4. INPOS は有効になっていますので, 使用しない場合は"B 接"にして, 常に入力されている状態にして下さい.
5. OLS の検出は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下さい.

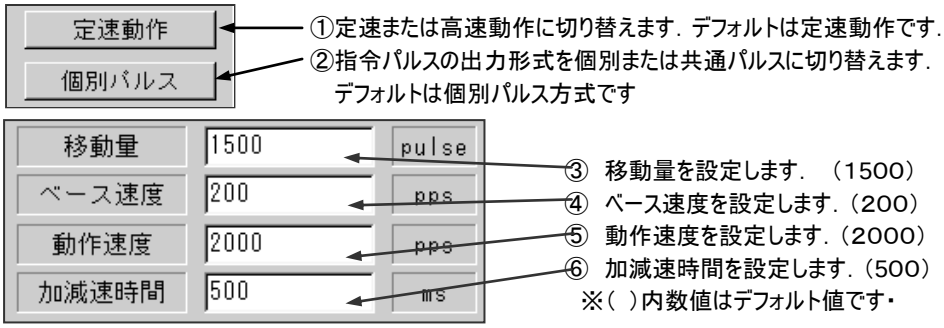
6.2.2　「動かしてみる」の設定画面

(1)変更可能な軸動作条件

「動作確認」画面で全ての軸を停止させて"設定"を選択しますと下記画面が表示されます.

図 6.2-3 「動かしてみる」の設定画面

動作可能な全ての軸について,個々に動作条件が設定出来ます.



(注)1. 移動量の設定範囲は－999999～＋9999999 パルスです.

2. ベース速度, 動作速度の設定範囲は 1～65535PPS です. ただし組み合わせによっては設定できない場合があります. 同様に加減速時間も動作速度, ベース速度との組み合わせによっては設定できない場合があります.

3. また, 動作速度をベース速度以下に設定すると, DLSon, OLSon, または停止, 減速すべきところでベース速度に加速することになります.

4. 【 エラーメッセージの表示 】

※ 設定できない値を入力し, 動作確認画面に戻ろうとした時に表示されます. 設置値を見直してください.