

モーション制御用省配線システム

motionCAT *series*

PCI Bus Master

HPCI-MCAT520M

CompactPCI Bus Master

HCPCI-MNT720M

ソフトウェアマニュアル

〈Windows 版〉



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



---

本マニュアル及びプログラムの全部又は一部の無断転載、コピーを禁止します。  
本製品の内容に関しましては、改良等により将来予告なしに変更することがあります。  
本製品の内容についてお気づきの点がございましたら、お手数ながら当社までご連絡ください。

Windows は Microsoft Corporation の米国及びその他の国における登録商標です。  
その他、記載されている会社名、製品名は、各社の商標又は登録商標です。

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](mailto:sales@hivertec.co.jp)

第 2.00 版 2011 年 10 月 24 日発行  
不許複製・転載



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。

本製品をご使用される前に



motionCAT シリーズユーザーズマニュアル<導入編>「1. はじめに」  
を必ずご一読の上ご利用をお願い致します。



本書は、motionCAT シリーズユーザーズマニュアル<導入編>の  
1 章, 7 章をご理解いただいた前提で書かれています。

## 目 次

注意事項 .....	1
保証範囲 .....	2
免責事項 .....	2
安全にお使い頂くために .....	2
添付ソフトウェア適合 OS.....	3
試運転・調整 .....	3
1.    はじめに.....	4
1.1    マニュアル構成.....	5
1.1.1    motionCAT シリーズユーザーズマニュアル<導入編>.....	5
1.1.2    motionCAT シリーズユーザーズマニュアル<運用編>.....	5
1.1.3    各種マスターソフトウェアマニュアル.....	5
1.2    添付ソフトウェア.....	5
1.3    呼称.....	5
1.4    ソフトウェアの構成.....	6
2.    サンプルプログラム .....	9
2.1    サンプルプログラムの構成 .....	10
2.2    DIO モジュール(HM-D1616C)サンプルプログラム.....	11
2.2.1    サンプルプログラムの実行.....	11
2.2.2    サンプルプログラムの操作.....	11
2.3    DI モジュール(HM-DI320C)サンプルプログラム.....	13
2.3.1    サンプルプログラムの実行.....	13
2.3.2    サンプルプログラムの操作.....	13
2.4    DO モジュール(DO320C)サンプルプログラム.....	15
2.4.1    サンプルプログラムの実行.....	15
2.4.2    サンプルプログラムの操作.....	15
2.5    DIO モジュール(HM-D2408C)サンプルプログラム.....	17
2.5.1    サンプルプログラムの実行.....	17
2.5.2    サンプルプログラムの操作.....	17
2.6    アナログモジュール(HM-A4401C, HM-A4199C)サンプルプログラム.....	19
2.6.1    サンプルプログラムの実行.....	19
2.6.2    サンプルプログラムの操作.....	19
2.7    位置決めモジュール(HM-P100C, HM-W200C)サンプルプログラム .....	24

2.7.1	サンプルプログラムの実行	24
2.7.2	サンプルプログラムの機能	24
2.7.3	サンプルプログラムの操作	24
2.8	ABS エンコーダ受信モジュール(HM-S100C)サンプルプログラム	29
2.8.1	サンプルプログラムの実行	29
2.8.2	サンプルプログラムの操作	29
3.	ソフトウェアの準備	34
3.1	マスターボードを複数枚使用する場合	35
3.1.1	ボードのデバイス番号	35
3.1.2	ボード ID の使用	35
3.2	アプリケーション作成準備	35
3.2.1	Microsoft Visual C++ (6.0 以上)アプリケーション作成準備	35
3.2.2	Microsoft Visual Basic(.NET2003 以上)アプリケーション作成準備	35
3.2.3	Microsoft Visual Basic 6.0 アプリケーション作成準備	36
3.3	ボードアクセス方法	36
3.3.1	デバイス情報構造体	36
3.3.2	モジュール情報構造体	37
3.4	ボードアクセスの準備手順と終了処理	39
3.4.1	準備手順	39
3.4.2	終了処理	39
3.5	関数の戻り値	40
4.	ライブラリ関数	41
4.1	モーションモジュール用ライブラリ関数	42
4.2	モーションモジュール用ライブラリ関数一覧	42
4.2.1	デバイス関係	42
4.2.2	初期設定	42
4.2.3	状態読み出し	42
4.2.4	動作設定	43
4.2.5	動作制御指令	43
4.2.6	計算関数	43
4.3	モーションモジュール用ライブラリ関数一覧	44
4.3.1	デバイス関係	44
4.3.2	初期設定	46
4.3.3	状態読み出し	52
4.3.4	動作設定	57
4.3.5	動作制御指令	63
4.3.6	計算関数	72
5.	ドライバ関数	74
5.1	ドライバ関数	75
5.2	ドライバ関数一覧	75
5.2.1	デバイス関係	75
5.2.2	オプションポート関係	75
5.2.3	センターデバイス関係	75
5.2.4	DIO モジュール(一部アナログモジュールにも使用)関係	76
5.2.5	モーションモジュール関係	76
5.2.6	アナログモジュール関係	76
5.3	ドライバ関数詳細	76
5.3.1	デバイス関係	77
5.3.2	オプションポート関係	78
5.3.3	センターデバイス関係	79
5.3.4	DIO モジュール(一部アナログモジュールにも使用)関係	86
5.3.5	モーションモジュール関係	87

5.3.6	アナログモジュール関係.....	89
6.	ポート資料 .....	93
6.1	ポート表 .....	94
6.2	オプションポート.....	95
6.2.1	マスターボード汎用入力ポート(HCPCI-MNT720Mのみ).....	95
6.2.2	マスターボード汎用出力設定ポート(HCPCI-MNT720Mのみ).....	95
6.2.3	G9001A ステータス.....	95
6.2.4	G9001A 通信速度設定状態.....	96
6.2.5	PCI 割込許可 .....	96
6.2.6	PCI 割込ステータスポート .....	96
6.2.7	ボード ID 確認ポート.....	96
6.2.8	ボードコード確認ポート .....	97
6.3	PCI コンフィグレーションレジスタ.....	97

## 図 表 目 次

図 1.4-1	32ビット Windows ソフトウェアの構成	6
図 1.4-2	64ビット Windows ソフトウェアの構成	7
図 2.2-1	DIO モジュールサンプル画面	11
図 2.2-2	モジュール ID コンボ BOX	12
図 2.2-3	DIO モジュール操作画面	12
図 2.3-1	DI モジュールサンプル画面	13
図 2.3-2	モジュール ID コンボ BOX	13
図 2.3-3	DI モジュール操作画面	14
図 2.4-1	DO モジュールサンプル画面	15
図 2.4-2	モジュール ID コンボ BOX	15
図 2.4-3	DO モジュール操作画面	16
図 2.5-1	T モジュールサンプル画面	17
図 2.5-2	モジュール ID コンボ BOX	17
図 2.5-3	T モジュール操作画面	18
図 2.6-1	アナログモジュールサンプル画面	19
図 2.6-2	モジュール ID コンボ BOX	19
図 2.6-3	サイクリック通信開始後の画面	20
図 2.6-4	レジスタ設定画面	22
図 2.6-5	レジスタ設定画面(1CH 分)	22
図 2.6-7	レジスタ設定画面の OK ボタンと Cancel ボタン	23
図 2.6-6	プリセットデータ設定画面	23
表 2.7-1	サンプルプログラム G9003 レジスタ初期値	24
図 2.7-1	プログラム選択画面	25
図 2.7-2	デバイスオープン/クローズ画面	25
図 2.7-3	原点復帰動作サンプル画面	26
図 2.7-4	入力極性選択	26
図 2.7-5	動作速度設定	26
図 2.7-6	原点復帰動作開始ボタン	27
図 2.7-7	連続送り動作サンプル画面	27
図 2.7-8	位置決め動作サンプル画面	28
図 2.7-9	複数軸動作サンプル画面	28
図 3.1-1	ボードを複数枚使用	35
表 3.5-1	関数の戻り値	40
表 6.1-1	ボードアドレス	94
図 6.2-1	マスターボード汎用入力ポートのビット構成	95
表 6.2-1	マスターボード汎用入力ポートの内容	95
図 6.2-2	マスターボード汎用出力設定及び出力状態確認ポートのビット構成	95
表 6.2-2	マスターボード汎用出力設定及び出力状態確認ポートの内容	95
図 6.2-3	G9001A ステータス確認ポートのビット構成	95
表 6.2-3	G9001A ステータス確認の内容	95
図 6.2-4	G9001A 通信速度設定状態のビット構成	96
表 6.2-4	G9001A 通信速度設定確認ポートの内容	96
図 6.2-5	PCI 割込許可設定ポートのビット構成	96
表 6.2-5	PCI 割込許可設定ポートの内容	96
図 6.2-6	PCI 割込ステータスポートのビット構成	96
表 6.2-6	PCI 割込ステータスポートの内容	96
図 6.2-7	ボード ID 確認ポートのビット構成	96
表 6.2-7	モジュール ID 読出(ボード ID)の内容	96

表 6.2-8	ボードコード確認ポートの内容 .....	97
表 6.3-1	PCI コンフィギュレーションレジスタ .....	97
表 6.3-2	PCI アドレス空間 .....	97

## 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ください。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ください。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。

## 添付ソフトウェア適合OS



**注 意**



HPCI-MCAT520M, HCPCI-MNT720M の標準添付ソフトウェアは Windows7, Windows Vista, WindowsXP, Windows2000 に対応しております。

## 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ると、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。

# 1. はじめに

## 1.1 マニュアル構成

motionCAT シリーズの製品には次のマニュアルが添付されています。

- motionCAT シリーズユーザーズマニュアル<導入編>
- motionCAT シリーズユーザーズマニュアル<運用編>
- 各種マスターソフトウェアマニュアル

### 1.1.1 motionCATシリーズユーザーズマニュアル<導入編>

このマニュアルには以下の事項が記述されています。

- (1) motionCAT の導入
- (2) マスターボード
- (3) スレーブ
- (4) インストール
- (5) 試運転
- (6) アクセサリ
- (7) 用語の説明
- (8) 接続例

(1),(7)の内容は motionCAT システムを使用する方全ての人を対象としていますので、必ずご一読の上ご利用をお願いいたします。

その他は主として設置・接続・配線をする開発者を対象としています。

### 1.1.2 motionCATシリーズユーザーズマニュアル<運用編>

このマニュアルには motionCAT シリーズ製品に共通するソフトウェア構築のために必要な事項が記述されています。

- (1) 各モジュールの運用
- (2) デバイス資料

### 1.1.3 各種マスターソフトウェアマニュアル

各種マスタに添付されるソフトウェアの使用方法について記述されています。

- (1) ソフトウェアの構成
- (2) サンプルプログラム
- (3) ソフトウェアの準備
- (4) ライブラリ関数
- (5) ドライバ関数
- (6) ポート資料

## 1.2 添付ソフトウェア

motionCAT シリーズのマスタには標準添付として以下の Windows 用ソフトウェアが添付されます。

- (1) Windows 用デバイスドライバ
- (2) 動作確認用プログラム「動かしてみる」
- (3) Microsoft Visual Studio 用サンプルプログラム

添付ソフトウェアの内容については各種マスターソフトウェアマニュアルを参照ください。

## 1.3 呼称

本書では motionCAT シリーズ PCI Bus マスタ HPCI-MCAT520M, CompactPCI Bus マスタ HPCI-MNT720M を MCATと呼称します。

## 1.4 ソフトウェアの構成

弊社の提供するソフトウェアは、ライブラリ関数、ドライバ関数、デバイスドライバの3種類です。

デバイスドライバはMCATへの入出力を行うソフトウェアです。

ドライバ関数はアプリケーションとデバイスドライバをつなぐ入出力関数「デバイスドライバ I/F 用ライブラリ」であり、Win32API関数としてDLLファイルで提供されています。

ライブラリ関数はドライバ関数で構成され、モーションモジュールの初期化、原点復帰、位置決め動作等の基本的な動作を制御することができます。ライブラリ関数はソースファイルで提供されていますので、適宜変更追加することができます。

また各開発言語用ライブラリ関数の仕様は同様になっています。

### (1) Windows 版デバイスドライバ

◇Windows7(32bit),Vista(32bit),XP,2000 用 ... hm520wdm.sys

◇Windows7(64bit),Vista(64bit)用 ... hm520x64.sys

### (2) Windows 版デバイスデバイスドライバ I/F 用 DLL

デバイスドライバ I/F 用 DLL に含まれる各種関数を「ドライバ関数」と称します。

◇Windows 用 ... himnt520.dll(32bit 用と 64bit 用があります)

### (3) Windows 版ライブラリ関数

◇VC++用ライブラリ関数ソースファイル ... hpxl1a.c

VC++用ライブラリ関数ヘッダーファイル ... hpxl1a.h

◇VB6.0 用ライブラリ関数ソースファイル ... hpxl1a.bas

◇VB.NET 用ライブラリ関数ソースファイル ... hpxl1a.vb

◇VC#用ライブラリ関数ソースファイル ... hpxl1a.cs

アプリケーションプログラムとこれらのソフトウェアの関連は下図の通りです。

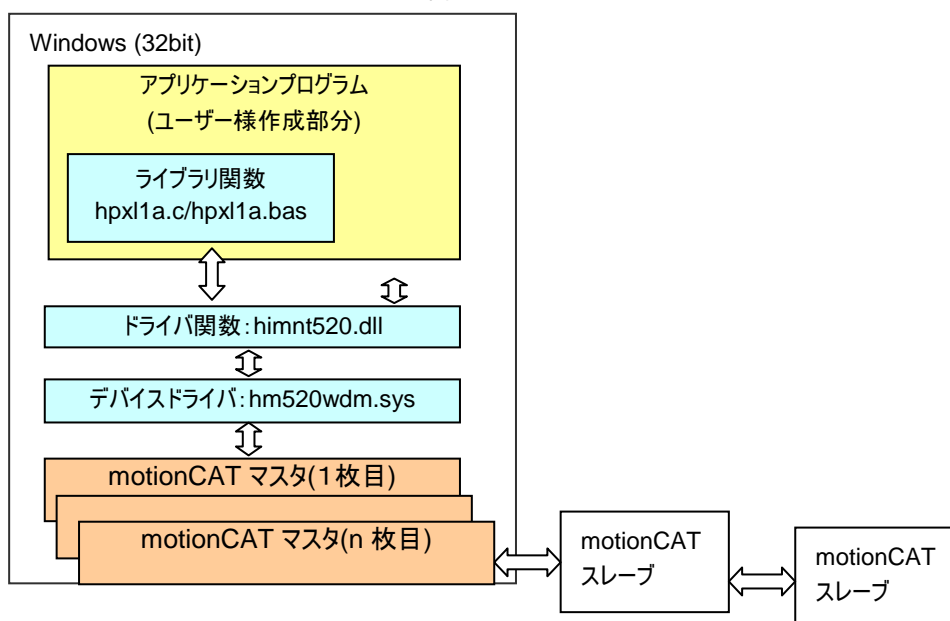


図 1.4-1 32ビット Windows ソフトウェアの構成

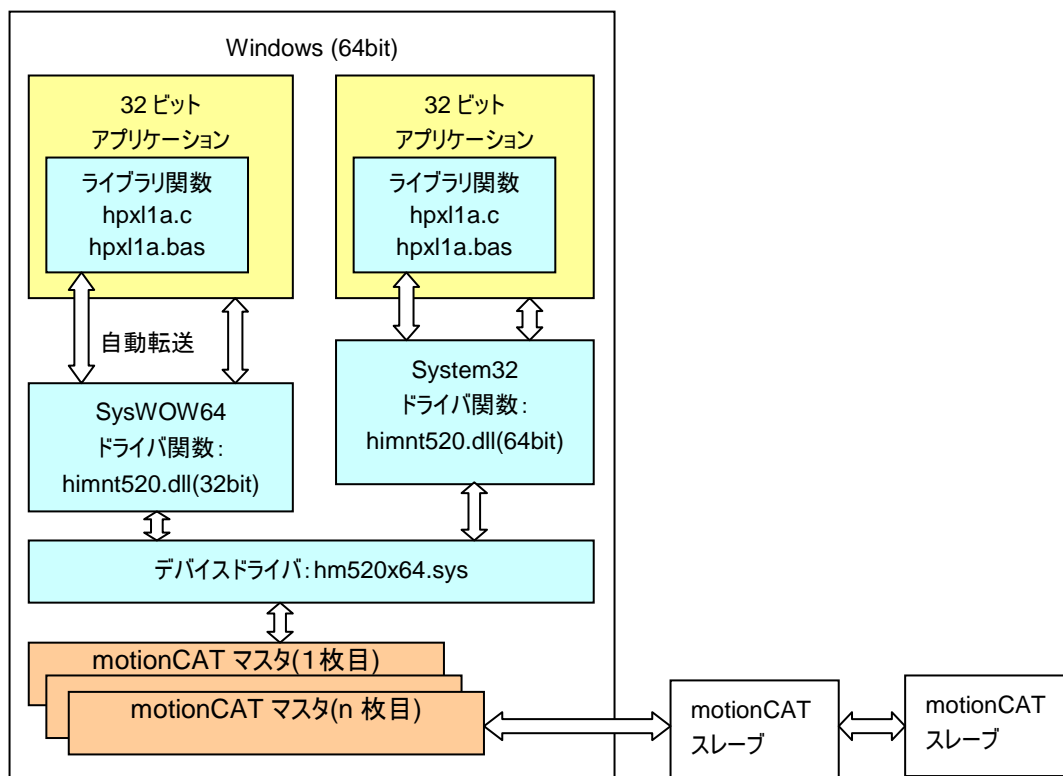


図 1.4-2 64ビット Windows ソフトウェアの構成



## 2. サンプルプログラム

## 2.1 サンプルプログラムの構成

サンプルプログラムは、添付 CD の Sample フォルダの中にあります。ファイルの説明は添付 CD の Readme.txt を参照してください。

サンプルプログラムは Microsoft Visual C++ 6.0, Microsoft Visual Basic 6.0, Microsoft Visual Basic.NET2003, Microsoft Visual C#.NET2003 で作成されています。開発環境に関わらず同様の仕様で作成しています。以降は代表として Microsoft Visual C++ 6.0 サンプルプログラムの説明となります。

## 2.2 DIO モジュール(HM-D1616C)サンプルプログラム

DIO モジュールの為のサンプルプログラムについて説明します。



### サンプルプログラム実行上の注意事項

- 使用するマスタのボード ID は 0 としています。
- 同一ライン上のモジュール ID は重複してはいけません。

### 2.2.1 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。  
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。

### 2.2.2 サンプルプログラムの操作

サンプルプログラムでは初期化は一部ソースプログラムで固定されています。  
初期化の条件を変更して動作させたい場合には、ソースプログラムを変更しリビルドして下さい。

#### (1) 操作画面 1(モジュール選択)

サンプルプログラムが正常に起動されると、次の画面が表示されます。

使用するマスターボードのボード ID を選択します。

使用するライン番号を選択します。

接続モジュール数を入力します。

選択されたマスターボードの PCI デバイス情報が表示されます。

```

*** PCI デバイス情報 ***
バス番号: 1
デバイス番号: 7
BAR2: f2020000h
BAR3: f2021000h
BAR4: 00000000h
IRQ 番号: 20
管理番号: 1
ボード ID: 00h
  
```

図 2.2-1 DIO モジュールサンプル画面

使用するマスターボードのボード ID とライン番号を選択します。その後、システム通信開始ボタンをクリックします。  
システム通信を行うと指定したライン上の DIO モジュールのモジュール ID がモジュール ID コンボ BOX に表示されます。

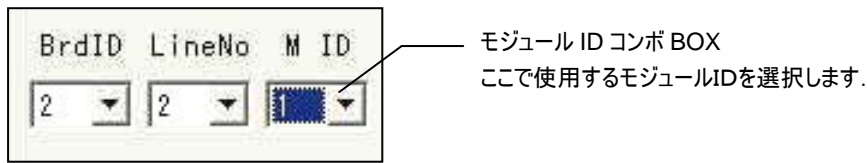


図 2.2-2 モジュール ID コンボ BOX

## (2) サイクリック通信開始

モジュール ID を指定しサイクリック通信開始をクリックすると、使用できるすべてのモジュールへのサイクリック通信が開始され、指定された DIO モジュールの操作ができます。

サイクリック通信停止をクリックすると全てのモジュールへのサイクリック通信が停止され操作画面 1 に戻ります。

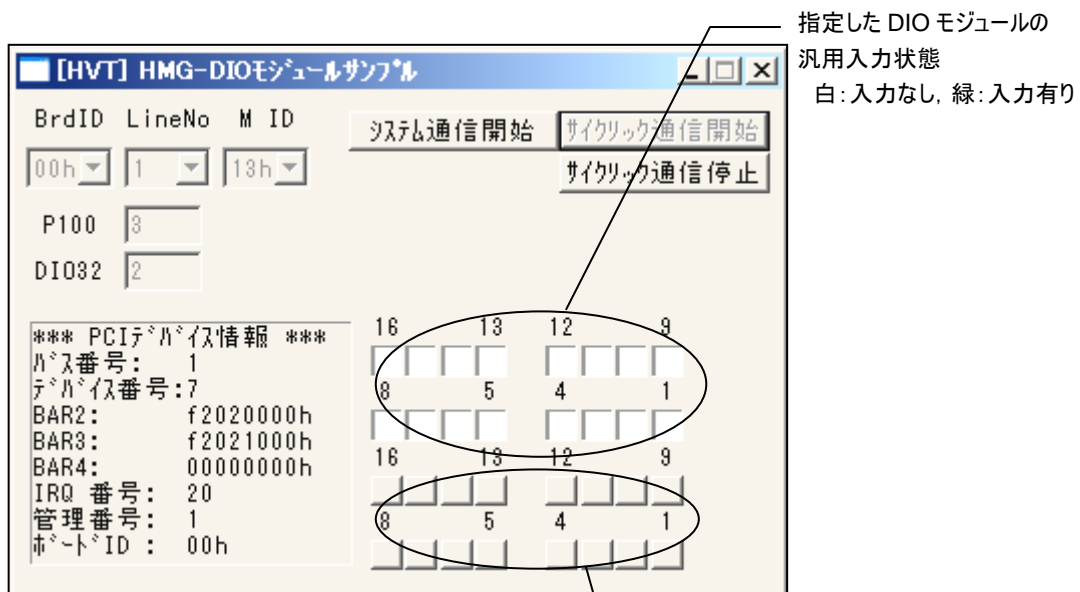


図 2.2-3 DIO モジュール操作画面

指定した DIO モジュールの汎用出力  
および出力状態  
クリックで出力反転  
灰: 出力なし, 緑: 出力中

## 2.3 DI モジュール(HM-DI320C)サンプルプログラム

DI モジュールの為のサンプルプログラムについて説明します。



### サンプルプログラム実行上の注意事項

- 使用するマスタのボード ID は 0 としています。
- 同一ライン上のモジュール ID は重複してはいけません。

### 2.3.1 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。  
サンプル実行ファイル(sdi32000.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。

### 2.3.2 サンプルプログラムの操作

サンプルプログラムでは初期化は一部ソースプログラムで固定されています。  
初期化の条件を変更して動作させたい場合には、ソースプログラムを変更しリビルドして下さい。

#### (1) 操作画面 1(モジュール選択)

サンプルプログラムが正常に起動されると、次の画面が表示されます。

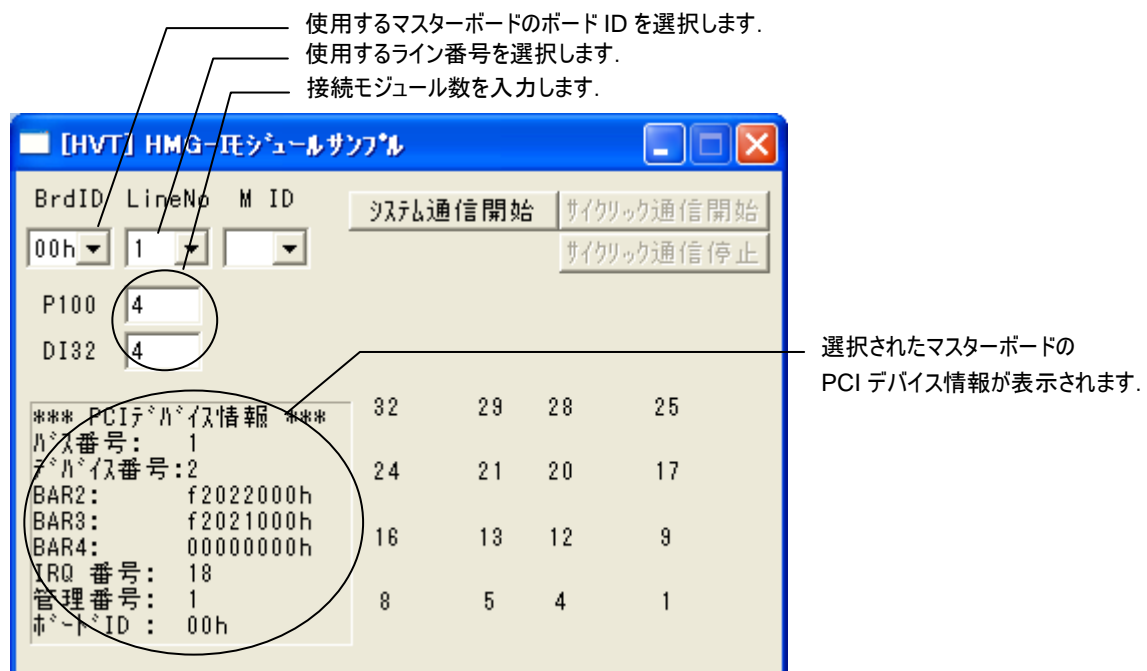


図 2.3-1 DI モジュールサンプル画面

使用するマスターボードのボード ID とライン番号を選択します。その後、システム通信開始ボタンをクリックします。  
システム通信を行うと指定したライン上の DI モジュールのモジュール ID がモジュール ID コンボ BOX に表示されます。

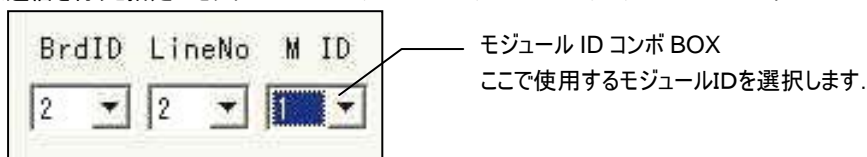


図 2.3-2 モジュール ID コンボ BOX

**(2) サイクリック通信開始**

モジュール ID を指定しサイクリック通信開始をクリックすると、使用できるすべてのモジュールへのサイクリック通信が開始され、指定された DI モジュールの操作ができます。

サイクリック通信停止をクリックすると全てのモジュールへのサイクリック通信が停止され操作画面 1 に戻ります。

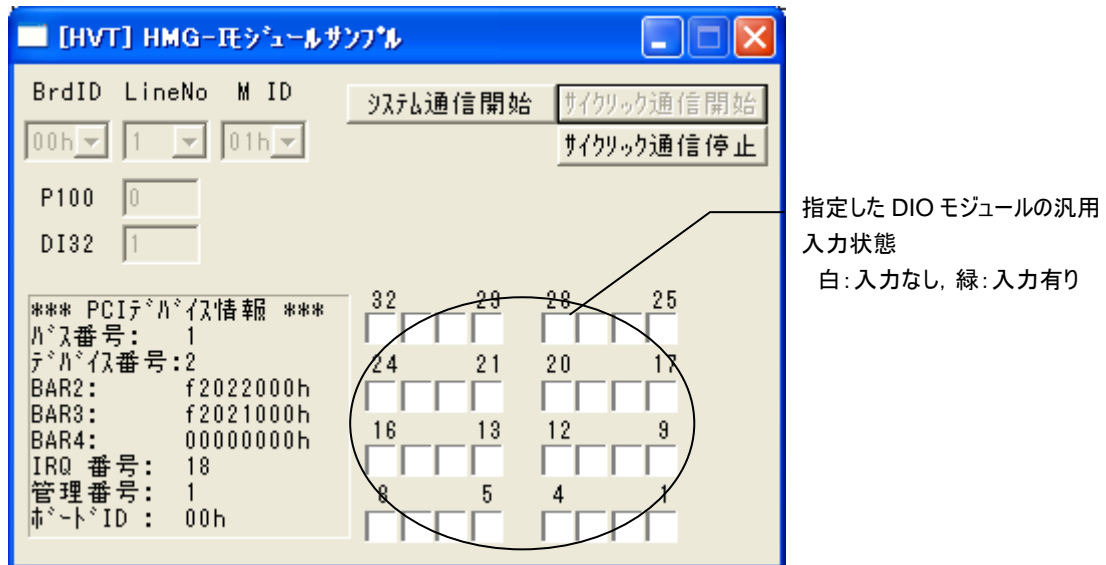


図 2.3-3 DI モジュール操作画面

## 2.4 DO モジュール(DO320C)サンプルプログラム

DO モジュールの為のサンプルプログラムについて説明します。



### サンプルプログラム実行上の注意事項

- 使用するマスタのボード ID は 0 としています。
- 同一ライン上のモジュール ID は重複してはいけません。

### 2.4.1 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。

サンプル実行ファイル(sdo32000.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。

### 2.4.2 サンプルプログラムの操作

#### (1) 操作画面 1(モジュール選択)

サンプルプログラムが正常に起動されると、次の画面が表示されます。

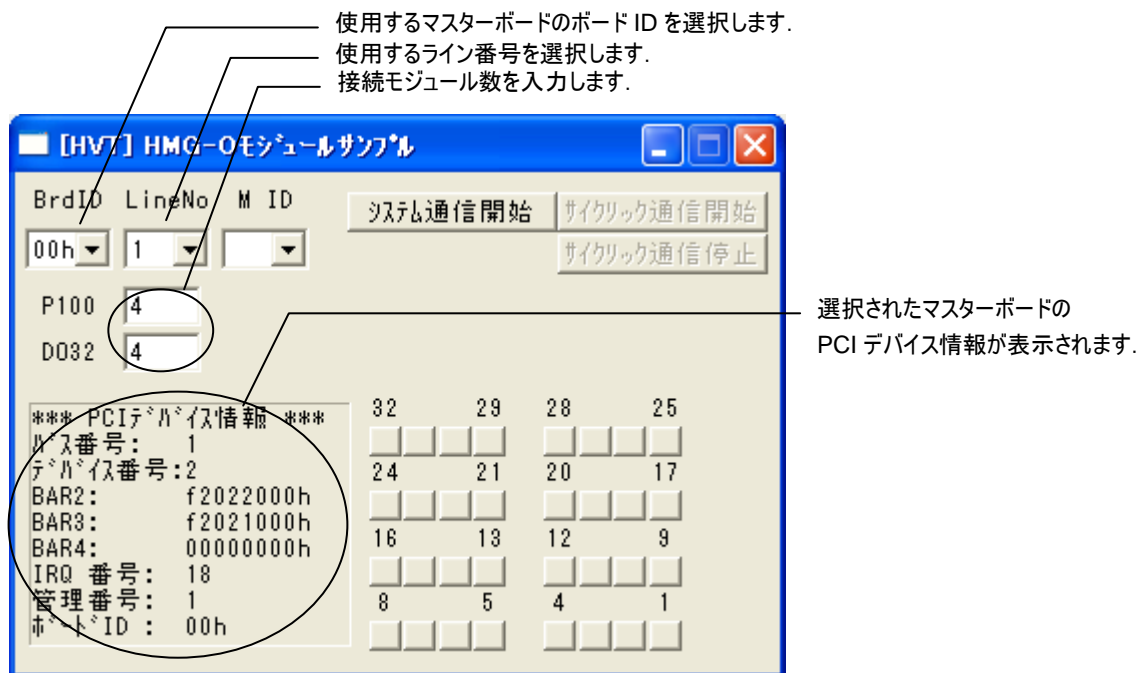


図 2.4-1 DO モジュールサンプル画面

使用するマスターボードのボード ID とライン番号を選択します。その後、システム通信開始ボタンをクリックします。

システム通信を行うと指定したライン上の DO モジュールのモジュール ID がモジュール ID コンボ BOX に表示されます。

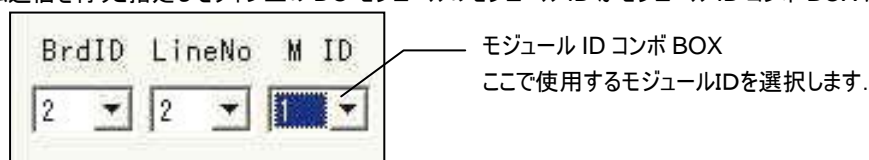


図 2.4-2 モジュール ID コンボ BOX

**(2) サイクリック通信開始**

モジュール ID を指定しサイクリック通信開始をクリックすると、使用できるすべてのモジュールへのサイクリック通信が開始され、指定された DO モジュールの操作ができます。

サイクリック通信停止をクリックすると全てのモジュールへのサイクリック通信が停止され操作画面 1 に戻ります。

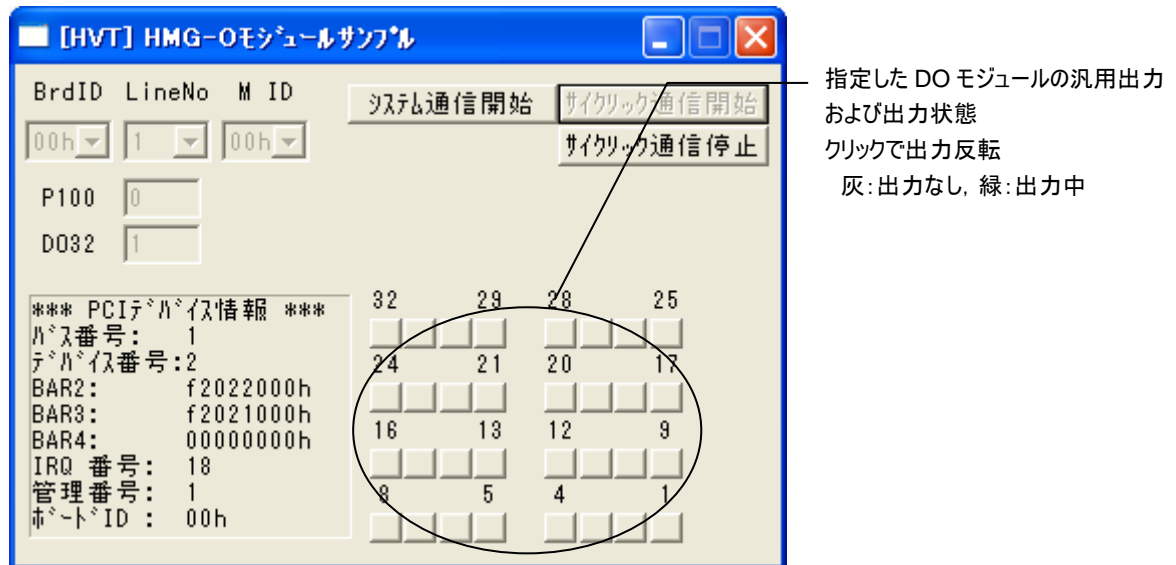


図 2.4-3 DO モジュール操作画面

## 2.5 DIO モジュール(HM-D2408C)サンプルプログラム

DIO モジュール(T モジュール)の為のサンプルプログラムについて説明します。



### サンプルプログラム実行上の注意事項

- 使用するマスタのボード ID は 0 としています。
- 同一ライン上のモジュール ID は重複してはいけません。

### 2.5.1 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。  
サンプル実行ファイル(sd24080.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。

### 2.5.2 サンプルプログラムの操作

サンプルプログラムでは初期化は一部ソースプログラムで固定されています。  
初期化の条件を変更して動作させたい場合には、ソースプログラムを変更しリビルドして下さい。

#### (1) 操作画面 1(モジュール選択)

サンプルプログラムが正常に起動されると、次の画面が表示されます。

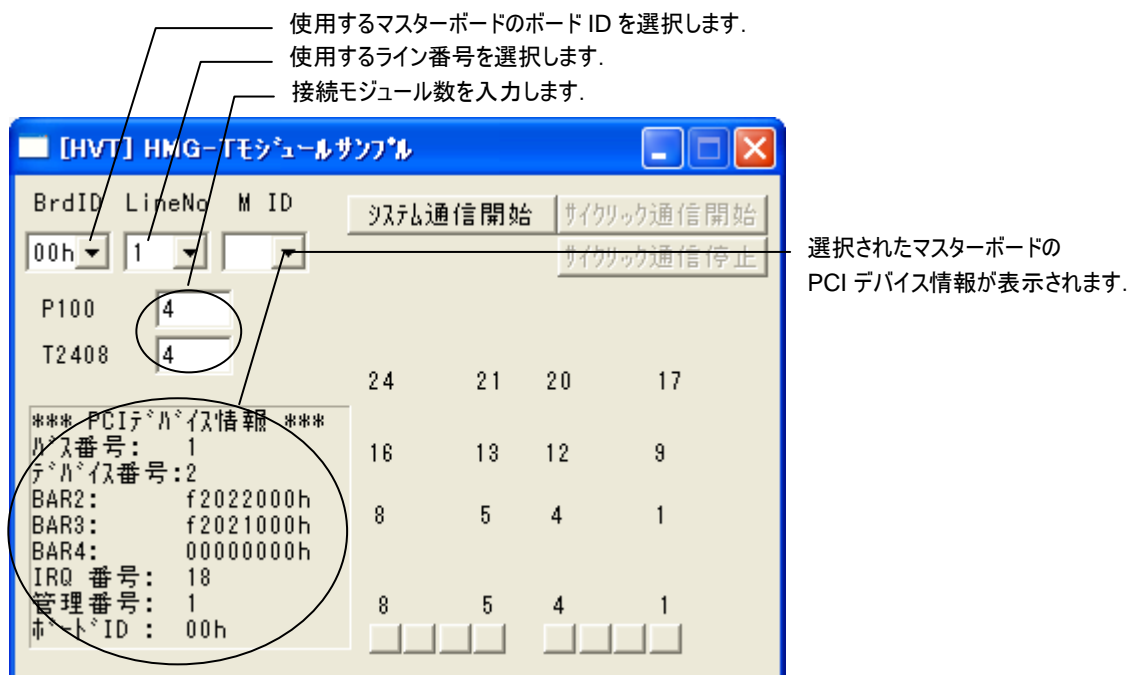


図 2.5-1 Tモジュールサンプル画面

使用するマスターボードのボード ID とライン番号を選択します。その後、システム通信開始ボタンをクリックします。  
システム通信を行うと指定したライン上の T モジュールのモジュール ID がモジュール ID コンボ BOX に表示されます。

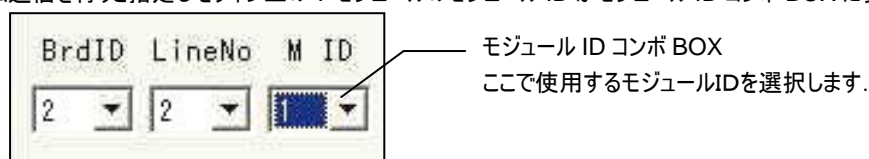


図 2.5-2 モジュール ID コンボ BOX

**(2) サイクリック通信開始**

モジュール ID を指定しサイクリック通信開始をクリックすると、使用できるすべてのモジュールへのサイクリック通信が開始され、指定された T モジュールの操作ができます。

サイクリック通信停止をクリックすると全てのモジュールへのサイクリック通信が停止され操作画面 1 に戻ります。

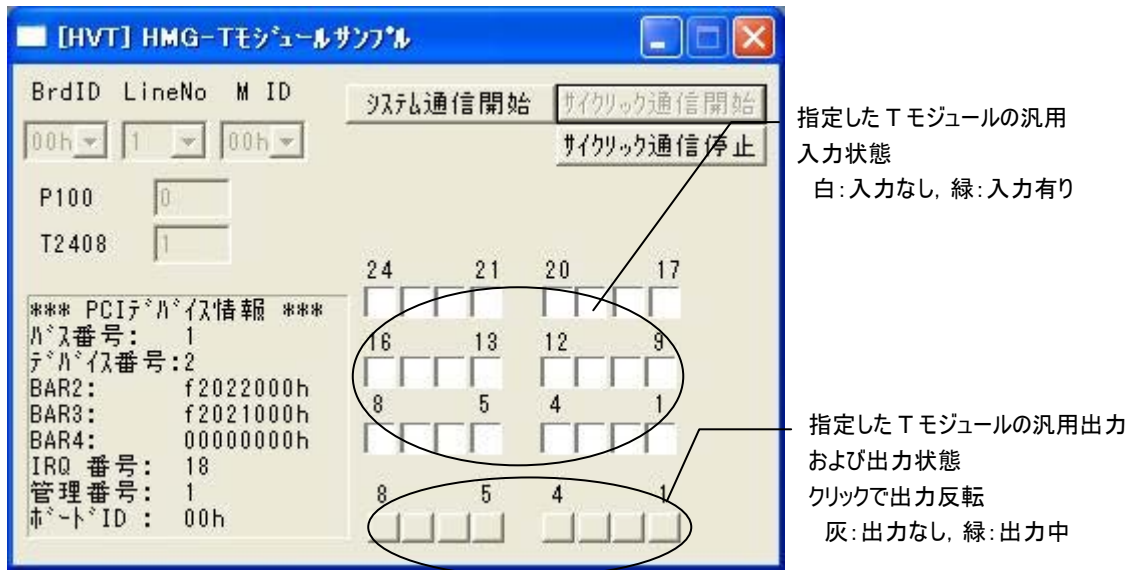


図 2.5-3 T モジュール操作画面

## 2.6 アナログモジュール(HM-A4401C, HM-A4199C)サンプルプログラム

アナログモジュールの為のサンプルプログラムについて説明します。



### サンプルプログラム実行上の注意事項

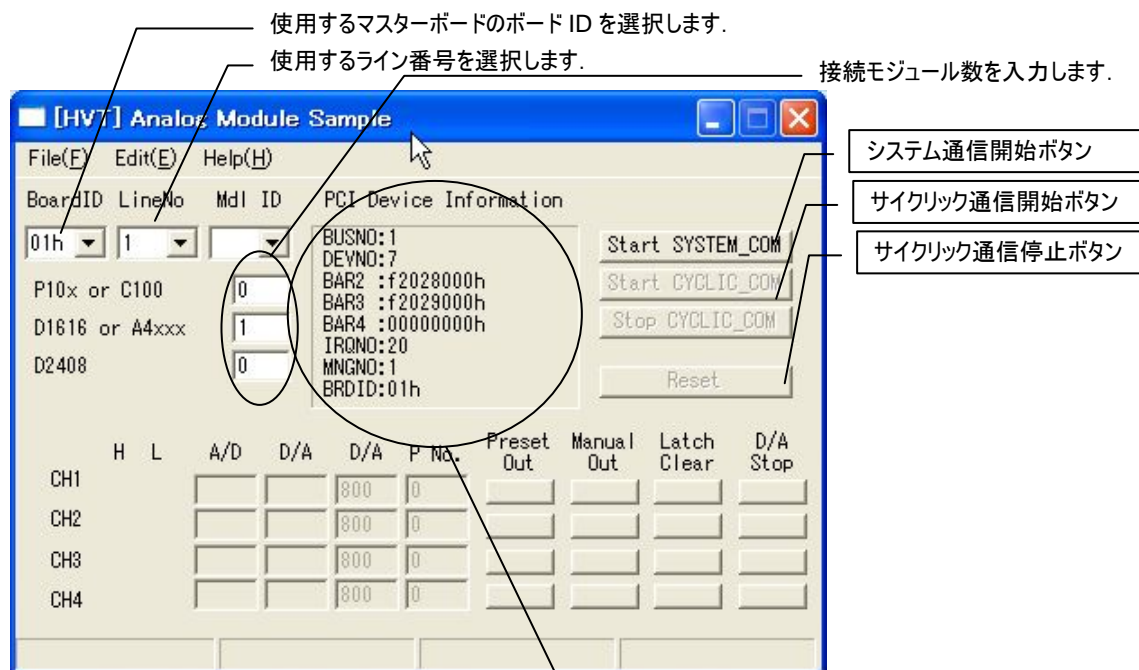
- 使用するマスタのボード ID は 0 としています。
- 同一ライン上のモジュール ID は重複してはいけません。

### 2.6.1 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。  
サンプル実行ファイル(sa44000.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。

### 2.6.2 サンプルプログラムの操作

サンプルプログラムでは初期化は一部ソースプログラムで固定されています。  
初期化の条件を変更して動作させたい場合には、ソースプログラムを変更しリビルドして下さい。  
サンプルプログラムが正常に起動されると、次の画面が表示されます。

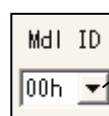


選択されたマスターボードの PCI デバイス情報が表示されます。

図 2.6-1 アナログモジュールサンプル画面

使用するマスターボードのボード ID とライン番号を選択します。

使用されるラインに接続されているモジュール数を入力します。その後、システム通信開始ボタンをクリックします。  
システム通信を行うと接続されるモジュール数を照らし問題がなければ、指定したライン上の DIO モジュールまたはアナログモジュールのモジュール ID がモジュール ID コンボ BOX に表示されます。



モジュール ID コンボ BOX(Hex 表示)  
ここで使用するモジュール ID を選択します。

図 2.6-2 モジュール ID コンボ BOX

## (1) サイクリック通信開始

モジュール ID を指定しサイクリック通信開始をクリックすると、使用できるすべてのモジュールへのサイクリック通信が開始され、指定されたアナログモジュールの操作ができます。サイクリック通信停止をクリックすると全てのモジュールへのサイクリック通信が停止され操作画面に戻ります。

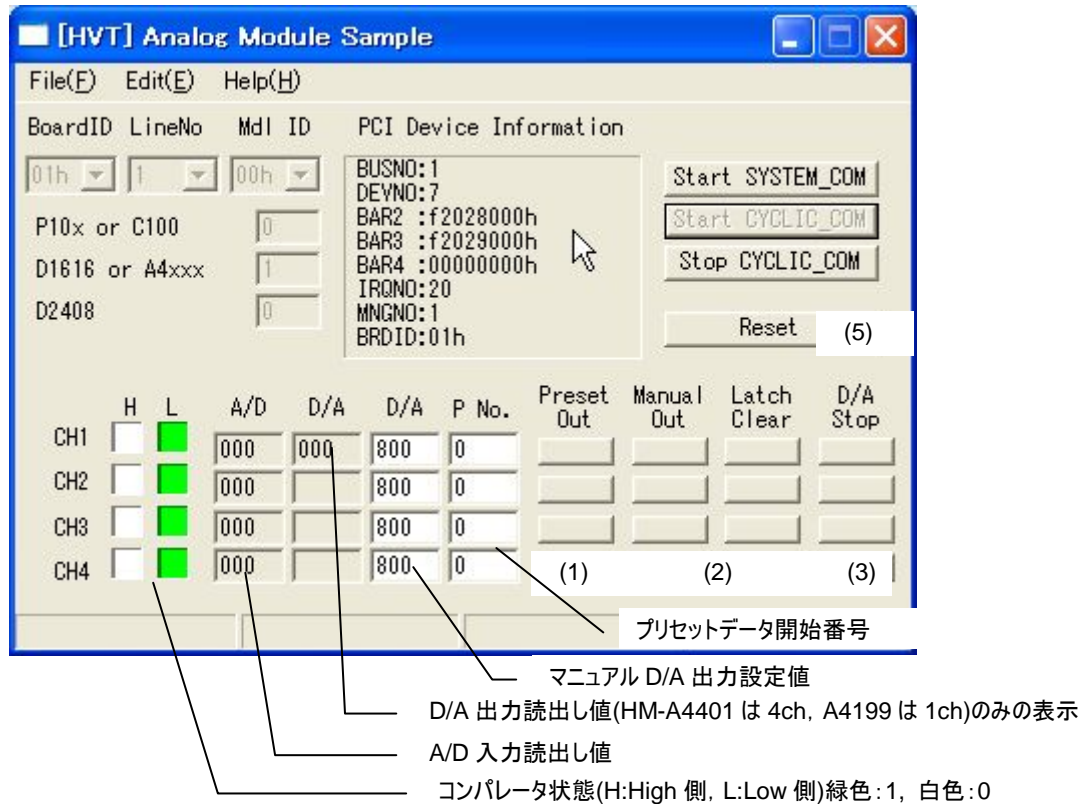


図 2.6-3 サイクリック通信開始後の画面

※HM-A4199 では、使用できる D/A は CH1 のみです。

### <操作ボタン>

- (1) 各 CH プリセット D/A 出力ボタン(Preset Out)  
あらかじめ登録したプリセットデータの番号を指定し出力します。
- (2) 各 CH マニュアル D/A 出力ボタン(Manual Out)  
各 CH の D/A 出力値を指定して出力します。
- (3) ラッチクリアボタン(Latch Clear)  
各 CH のラッチされているコンパレータ状態をクリアします。
- (4) D/A ストップボタン (初期値(D/A 0V または 4mA) を出力)
- (5) リセットボタン(Reset)  
以下の状態にします。
  - D/A 出力 OFF
  - 連続出力実行中止
  - コンパレータラッチクリア
  - 初期値書き込み(下表のサンプルプログラム初期値 1, 2 参照)
  - ステータス領域のクリア
  - 入力変化フラグのクリア

## 【 サンプルプログラム初期値 1 】

No	内 容	値(Hex)	備 考
1	システム動作条件	0x0000	100ms scale モード
2	A/D 範囲外通知条件	0x00ff	CH1~4 全有効, L/H=OR
3	コンパレータ動作条件-1	0x00ff	CH1~4 Low/High コンパレータ全有効
4	コンパレータ動作条件-2	0x0000	極性:条件成立時 1
5	プリセットデータ切替条件-1(CH1~4)	0x0000	コンパレータによる D/A 切替無効
6	プリセットデータ切替条件-2(CH1~4)	0x0000	DI(デジタル入力)による D/A 切替無効
7	プリセットデータ D/A 値 1	0x0800	
	プリセットデータ D/A 値 2	0x0400	
	プリセットデータ D/A 値 3	0x0200	
	プリセットデータ D/A 値 4	0x0100	
	プリセットデータ D/A 値 5	0x0080	
	プリセットデータ D/A 値 6	0x0000	
	プリセットデータ D/A 値 7	0x0400	
	プリセットデータ D/A 値 8	0x0800	
	プリセットデータ D/A 値 9	0x0c00	
	プリセットデータ D/A 値 10	0x0e00	
	プリセットデータ D/A 値 11	0x0fff	
プリセットデータ D/A 値 12	0x0fff		
プリセットデータ D/A 値 13-15	0x0000		
8	Low 側 A/D コンパレータ基準値(CH1~4)	0x0000	
9	High 側 A/D コンパレータ基準値(CH1~4)	0x0fff	

## 【 サンプルプログラム初期値 2 】

No	内 容	値(Hex)	備 考
10	プリセットデータ到達時間とイベント切替有効/無効 1	0x0e0a	到達時間=10x(Scale)
	プリセットデータ到達時間とイベント切替有効/無効 2	0x0e0a	"
	プリセットデータ到達時間とイベント切替有効/無効 3	0x0e0a	"
	プリセットデータ到達時間とイベント切替有効/無効 4	0x0e0a	"
	プリセットデータ到達時間とイベント切替有効/無効 5	0x0e0a	"
	プリセットデータ到達時間とイベント切替有効/無効 6	0x0e02	到達時間=2x(Scale)
	プリセットデータ到達時間とイベント切替有効/無効 7	0x0e05	到達時間=5x(Scale)
	プリセットデータ到達時間とイベント切替有効/無効 8	0x0e05,	"
	プリセットデータ到達時間とイベント切替有効/無効 9	0x0e05	"
	プリセットデータ到達時間とイベント切替有効/無効 10	0x0e05	"
	プリセットデータ到達時間とイベント切替有効/無効 11	0x0e05	"
	プリセットデータ到達時間とイベント切替有効/無効 12	0x0e32	到達時間=50x(Scale)
	プリセットデータ到達時間とイベント切替有効/無効(13-15)	0x0000	
11	プリセットデータ保持時間	0x0ff4	保持時間=500ms
12	プリセットデータ連続出力条件	0x0ccc,	終点番号=12
13	D/A オフセット値	0x0000	
14	A/D オフセット値(CH1~4)	0x0000	

## (2) レジスタ設定画面

以下の項目についてレジスタ設定します。

● レジスタ設定項目

	表示	内容
1	Comparator A/D	コンパレータ基準 A/D 値
2	D/A Offset	D/A 出力オフセット値
3	A/D Offset	A/D 入力オフセット値
4	System Mode	システム動作条件
5	Comp Alarm Logic	A/D 入力範囲外通知条件
6	Each Comp Logic	各コンパレータ動作条件

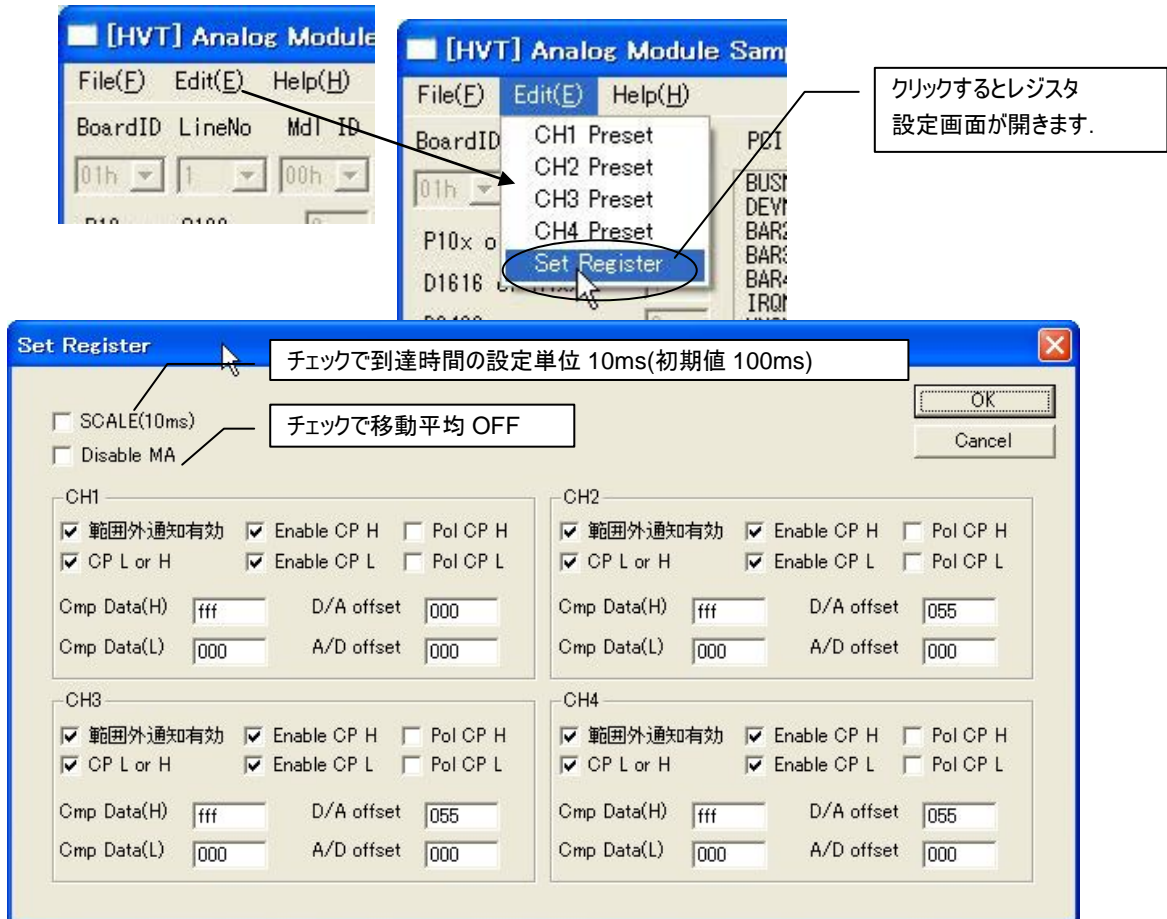


図 2.6-4 レジスタ設定画面

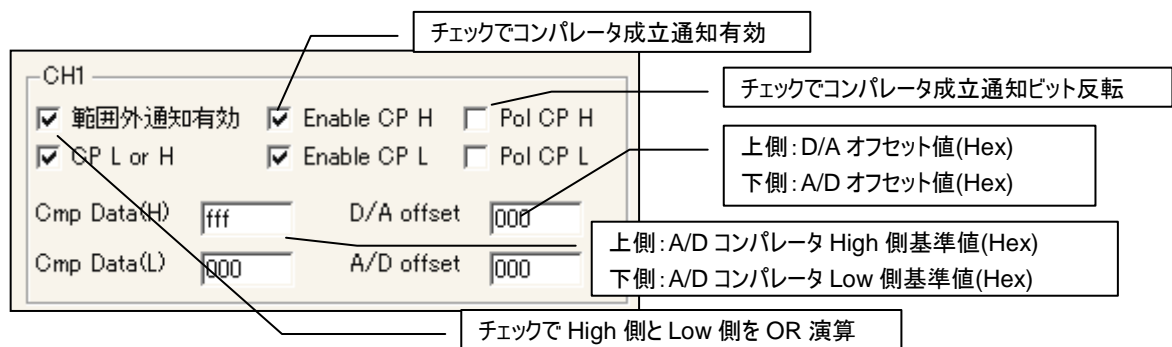
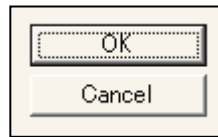


図 2.6-5 レジスタ設定画面(1CH 分)



OK ボタンクリックでデータ設定  
Cancel またはクローズ時はデータ設定されません。

図 2.6-7 レジスタ設定画面の OK ボタンと Cancel ボタン

【 プリセットデータ設定画面 】

	表示	内容
1	Preset D/A	プリセットデータ D/A 値
2	Trans Time	プリセットデータ 到達時間
3	Keep Time	プリセットデータ 保持時間
4	Preset Change Logic	プリセットデータイベント切替条件
5	Continue Mode	連続出力実行条件

クリックするとプリセットデータ設定画面が開きます。

プリセットデータ D/A 値

プリセットデータ 到達時間

到達時間 DI 切替時有効

到達時間コンパレータ切替時有効

到達時間手動切替時有効

プリセットデータ 保持時間

保持時間 DI 切替時有効

保持時間コンパレータ切替時有効

保持時間手動切替時有効

OK ボタンクリックでデータ設定.  
Cancel またはクローズ時はデータ設定されません。

High 側と Low 側のコンパレータ OR 演算

High 側コンパレータでの切替有効

Low 側コンパレータでの切替有効

DI での切替時の極性設定

イベントによるプリセットデータ開始番号

手動切替時終点プリセット番号

コンパレータによる切替設定

DI入力による切替

コンパレータ切替時終点プリセット番号

DI切替時プリセット番号

図 2.6-6 プリセットデータ設定画面

※HM-A4199 では、使用するプリセットデータは CH1 のみです。

## 2.7 位置決めモジュール(HM-P100C, HM-W200C)サンプルプログラム

### 2.7.1 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。  
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。

### 2.7.2 サンプルプログラムの機能

デバイスオープン/クローズサンプル	...	デバイス個数、情報取得、デバイスオープン/クローズ及びモジュール数のチェックを行います。
原点復帰動作	...	原点復帰動作のサンプルです。
連続送り動作	...	連続送り動作のサンプルです。
位置決め動作	...	1 軸位置決め動作のサンプルです。
複数軸動作	...	2 軸位置決め動作(同時スタート)のサンプルです。



#### サンプルプログラム実行上の注意事項

- 同一ライン上のモジュール ID は重複してはいけません。
- モジュール数チェックのためデバイスオープン/クローズのサンプルを起動してから他のサンプルを起動してください。
- 複数軸動作のサンプルは同一スレーブ内の 2 個のモーションモジュールで動作します。
- 複数軸動作のサンプルは異なるスレーブのモーションモジュールの動作はできません。
- W モジュールは SVALM 端子がありません。ソフトウェアでの入力極性は B 接設定にしてください。

### 2.7.3 サンプルプログラムの操作

#### (1) サンプルプログラム初期値

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。  
また、使用するボード ID は 0、使用するラインはライン 1 としています。  
初期化の条件を変更して動作させたい場合には、ソースプログラムを変更してリビルドしてください。

レジスタ	内容	レジスタ初期値	備考
RFL	ベース速度	200	200pps
RFH	動作速度	2000	2000pps
RUR	加速レート	1387	200pps→2,000ppsの加減速時間約500msec(直線加減速時)
RMG	速度倍率	199	1倍
RFA	補助速度	200	200pps
RENV1	環境設定1	00434004h	指令パルス出力形式: CW/CCW, SVCTRCL自動出力しない, SVCTRCL出力パルス幅:13ms, DLS, OLS, SVALM: B接, ELS, SVALM入力時即停止, DLSラッチしない, SVRDY, INPOS: A接, LATC, CLR: 立下りエッジ
RENV2	環境設定2	000004FFh	サーボI/F出力設定, エンコーダ入力4通倍
RENV3	環境設定3	00700002h	原点復帰モード2(OLS+Z), 原点復帰完了時カウンタ1~3をクリア
RIRQ	イベントマスク設定	1	正常停止時
上記以外			0

表 2.7-1 サンプルプログラム G9003 レジスタ初期値

## (2) プログラム選択画面

サンプルプログラムが正常に起動されると、次の画面が表示されます。ここでプログラムを選択します。



図 2.7-1 プログラム選択画面

## (3) デバイスオープン/クローズサンプル

プログラム選択画面のデバイスオープン/クローズラジオボタンをクリックすると以下の画面が表示されます。デバイスオープン/クローズのサンプルです。

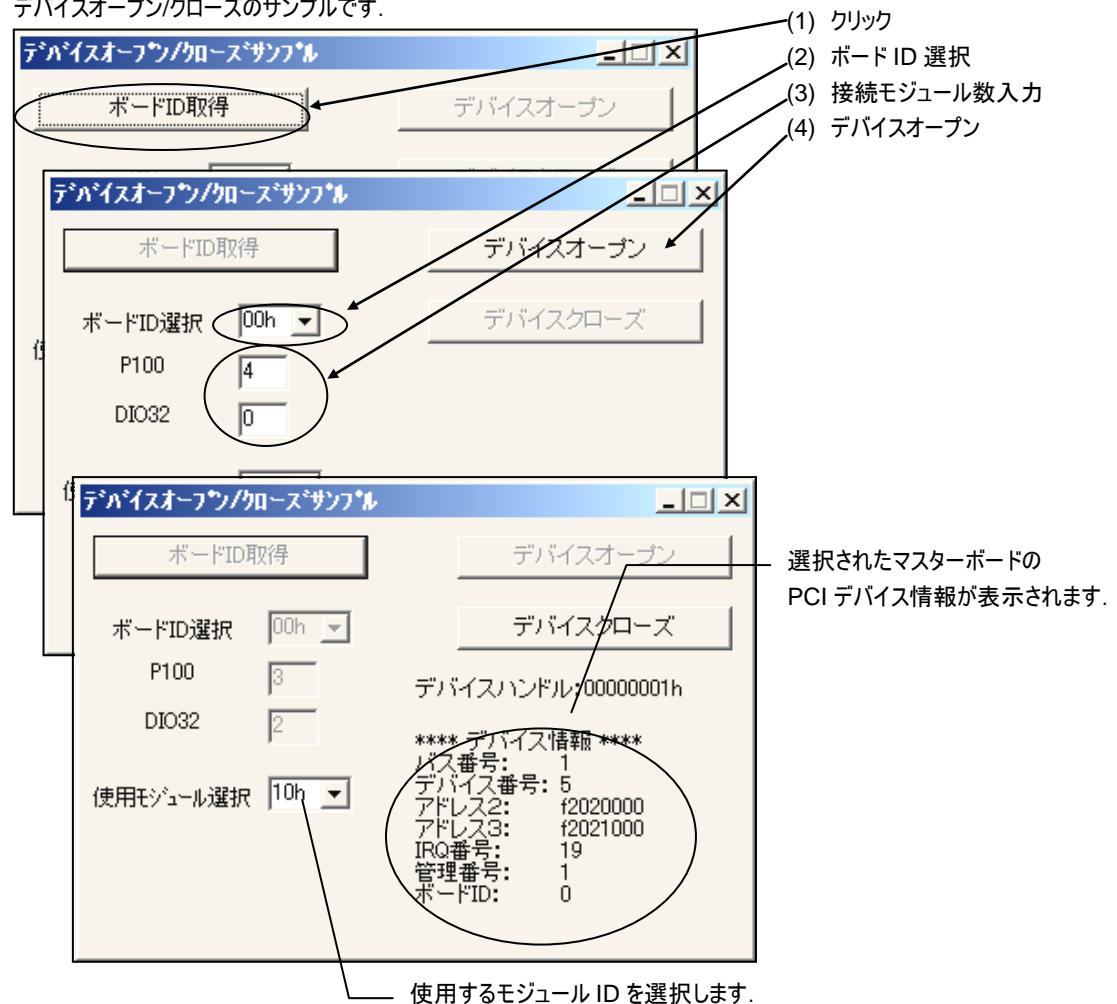


図 2.7-2 デバイスオープン/クローズ画面

#### (4) 原点復帰動作サンプル

原点復帰動作ラジオボタンをクリックすると以下の画面が表示されます。原点復帰動作の設定と原点復帰動作を行います。

【 操作画面 】

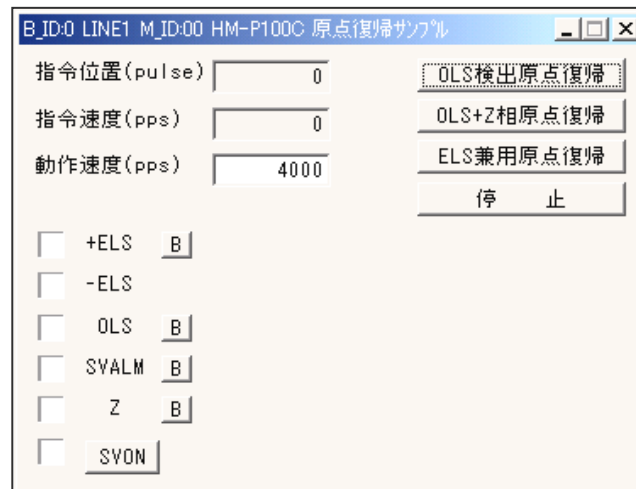


図 2.7-3 原点復帰動作サンプル画面

#### 【 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。

- OLS検出原点復帰 …… 原点復帰動作1: OLS検出後減速停止し反転拔出し, 再突入して完了.
- OLS+Z相原点復帰 …… 原点復帰動作2: OLS検出後減速しエンコーダZ相検出で完了.
- ELS兼用原点復帰 …… 原点復帰動作6: ELS検出後減速停止し反転ELS拔出して完了

原点復帰動作の詳細は「motionCAT シリーズユーザーズマニュアル<運用編>」を参照して下さい。

#### ■ 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, Z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。

**SVON** 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。

入力極性ボタン: 入力極性を切替えます。

また現在設定されている極性を表示します。

A: A接設定/B: B接設定

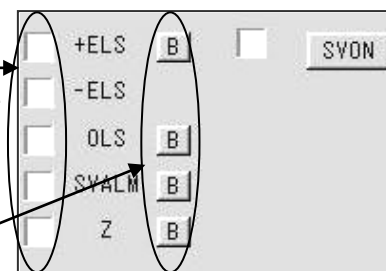


図 2.7-4 入力極性選択

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。

注 2. A 接は端子に電流が流れたときに「ON(検出)」,

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

#### ■ 動作速度設定

動作速度は 1~100,000(PPS)の範囲で設定できます。

初期値は 4,000(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 200(PPS)に設定していますので、動作速度を 200(PPS)以下に設定すると、

または OLS ON で減速すべきところで、200(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。



図 2.7-5 動作速度設定

### ■ 原点復帰動作開始

極性選択、動作速度を設定した後、"OLS 検出原点復帰"、"OLS+Z 相原点復帰"、"ELS 兼用原点復帰" ボタンをクリックすると、それぞれの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。



図 2.7-6 原点復帰動作開始ボタン

OLS の検出はエッジ検出ですので、動作開始時に OLS ON 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 OLS OFF 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。

ELS 兼用原点復帰実行時は ELS ON で減速停止です。

減速距離が、ELS ON になる位置から軸の機械的な終端までの距離以下になるような速度で実行して下さい。

## (5) 連続送り動作サンプル

加減速連続送り動作を行います。

【 操作画面 】

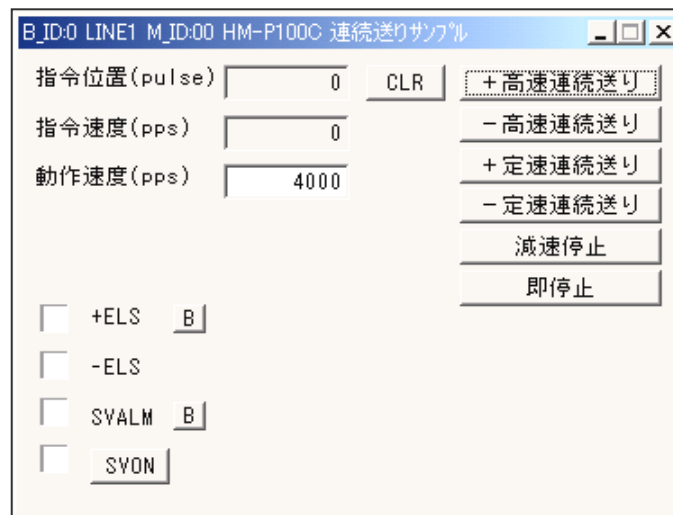


図 2.7-7 連続送り動作サンプル画面

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。

動作速度を 1~100,000(PPS)の範囲で設定し、**+高速連続送り**、**-高速連続送り** ボタンをクリックすると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。**CLR** ボタンでカウンタをリセットできます。

## (6) 位置決め動作サンプル

加減速位置決め動作を行います

【 操作画面 】

図 2.7-8 位置決め動作サンプル画面

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。

動作速度を 1~100,000(PPS)の範囲で設定し、移動量(符号付)を設定してから **高速位置決め** ボタンをクリックすると加減速位置決め動作を行います。 **停止** ボタンで動作を途中停止することができます。

**CLR** ボタンでカウンタをリセットできます。

## (7) 2 軸位置決め動作サンプル

ここでは同一ライン上の指定した 2 つのモーションモジュールの位置決めを行います。

【 操作画面 】

図 2.7-9 複数軸動作サンプル画面

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。

動作速度を 1~100,000(PPS)の範囲で設定し、移動量(符号付)を設定してから **スタート** ボタンをクリックすると指定した 2 軸の加減速位置決め動作を行います。 **停止** ボタンで 2 軸の動作を途中停止することができます。 **CLR** ボタンでカウンタをリセットできます。

注 1.このサンプルは複数軸を独立位置決めさせるサンプルです。

補間動作ではありませんので、片方の軸がサーボアラーム等により異常停止した場合は同時に停止しません。

## 2.8 ABS エンコーダ受信モジュール(HM-S100C)サンプルプログラム

### 2.8.1 サンプルプログラムの実行

ABS エンコーダ受信モジュールの為のサンプルプログラムについて説明します。



#### サンプルプログラム実行上の注意事項

- 使用するマスタのボード ID は 0 としています。
- 同一ライン上のモジュール ID は重複してはいけません。

### 2.8.2 サンプルプログラムの操作

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。

サンプル実行ファイル(ss10000.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) 初期画面

サンプルプログラムを起動すると、以下の画面が表示されます。

サンプルプログラムを起動した1回目は、①の[デバイスオープン/クローズ]を選択します。

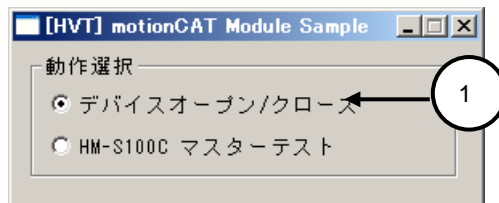


図 2.8-1 初期画面

#### (2) デバイスオープン/クローズ画面

①を選択すると下記画面が表示されます。②の<ボード ID 取得>ボタン押下で ID 取得後、③の<デバイスオープン>が有効になります。さらにこのボタンをクリックすると、デバイスハンドルなどのマスターボード情報(④)が表示されます。

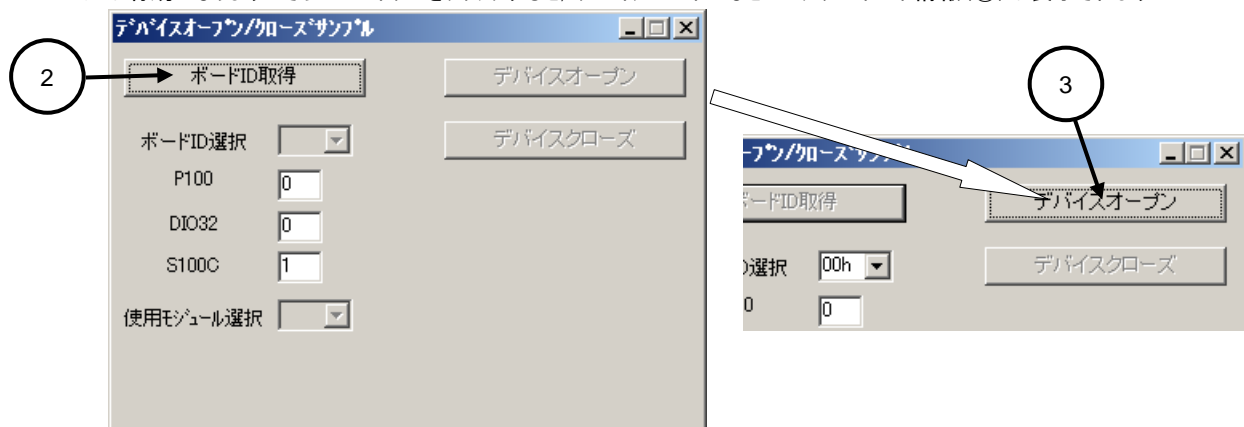


図 2.8-2 オープン/クローズ画面(ボード ID 取得, デバイスオープン)

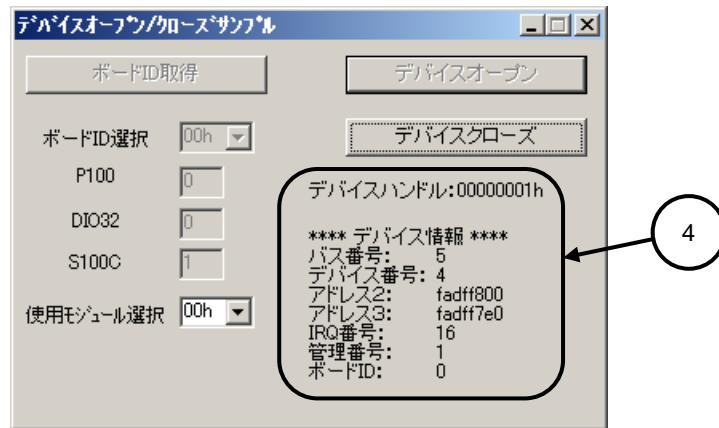


図 2.8-3 オープン/クローズ画面(デバイス情報表示)

デバイスの電源が入っていない、または正しく接続されていない場合、下記エラーが表示されます。

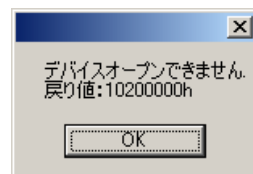



図 2.8-4 マスターオープンエラー

正常にオープンされた後、⑤の<デバイスクローズ>ボタンでデバイスクローズした後、でこの画面を終了します。

(初期画面に戻ります)

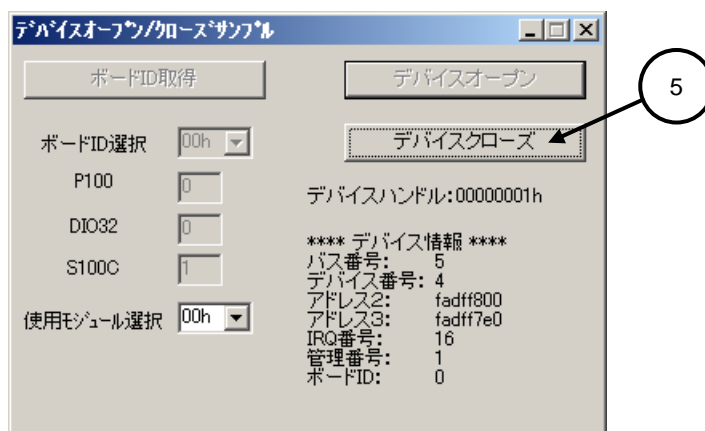


図 2.8-5 オープン/クローズ画面(デバイスクローズ)

### (3) HM-S100C サンプル操作画面

初期画面中の⑥の [HM-S100C マスターテスト] を選択します。



図 2.8-6 初期画面(サンプル操作画面オープン)

下図の画面が表示されます。この画面上にある⑦の <FIFO リセット> ボタンをクリックします。これにより HM-S100C モジュールへのコマンド送信が可能になります。

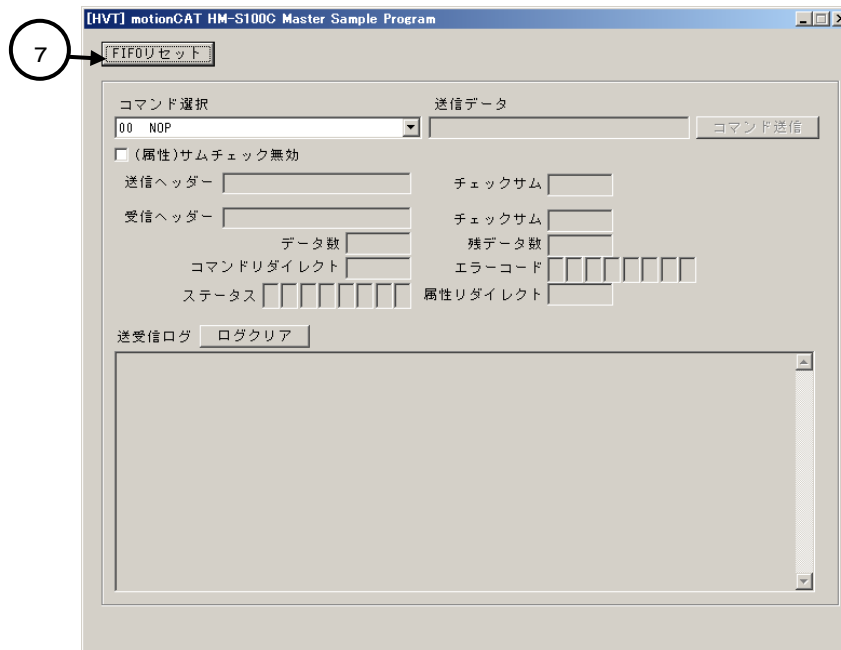


図 2.8-7 サンプル操作画面(初期オープン状態)

コマンド送信の操作を行う時は、⑧のコンボボックスからコマンドを選択し、⑨の <コマンド送信> ボタンをクリックします。HM-S100C モジュールと通信を行い、その結果が画面に表示されます。

⑩の[(属性) サムチェック無効] をチェックすると、サムチェック(チェックサム)が無効になります。

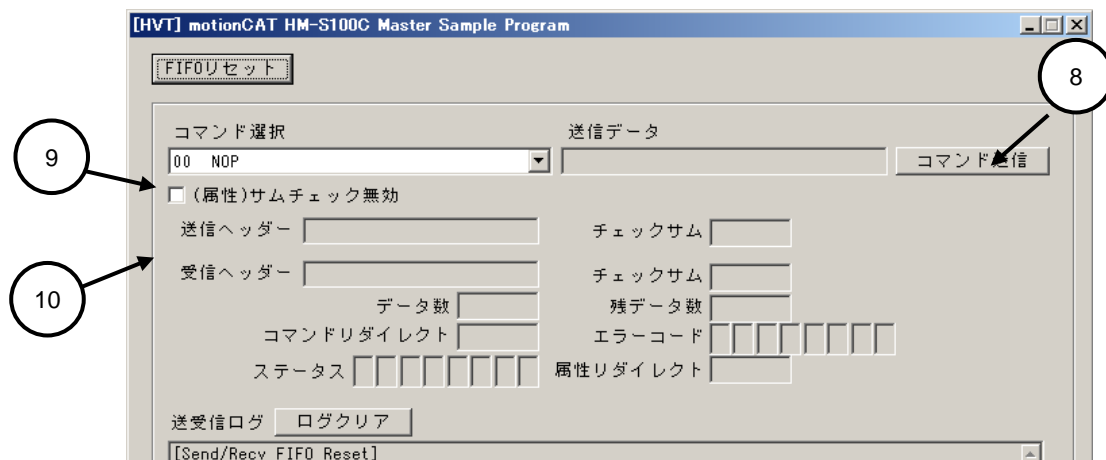


図 2.8-8 サンプル操作画面(FIFO リセット後)

コマンド実行の結果が、次の内容で表示されます。

- ・送信したコマンド(8 バイトの 16 進表記)
  - ・チェックサム(2 バイトの 16 進表記)
- …領域⑪
- ・受信したヘッダー(8 バイトの 16 進表記)
  - ・チェックサム(2 バイトの 16 進表記)
  - ・データ数
  - ・コマンドリダイレクト
  - ・ステータス(8 ビット, 左がビット7)
  - ・残データ数
  - ・エラーコード(8 ビット, 左がビット7)
  - ・属性リダイレクト
- …領域⑫
- ・送信・受信のログ …領域⑬
- マスタから HM-S100C モジュールへ送信したコマンドフレームが[Send \*\*\*\* Block]に、モジュールからマスタへの応答フレームが[Receive \*\*\*\* Block]として表示されます。

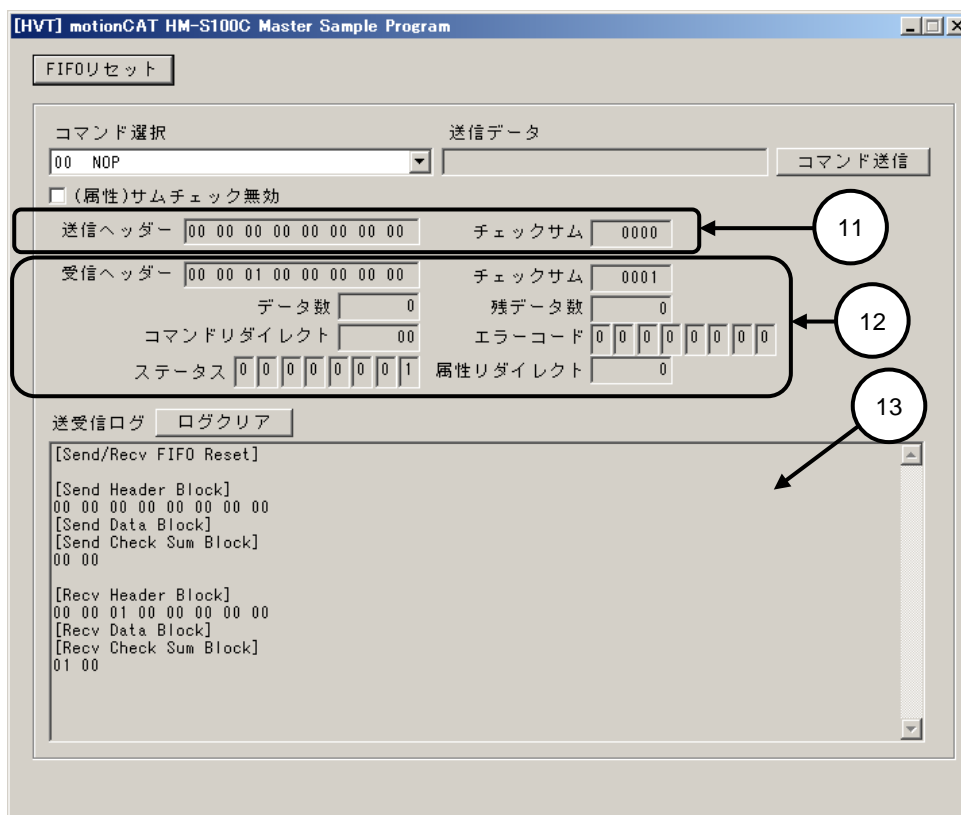


図 2.8-9 サンプル操作画面(コマンド結果表示-1)

「RS232C 通信条件設定」コマンドなど、パラメータが必要なコマンドを選択すると、⑭の[送信データ]欄が入力可能となります。

データ入力形式は 16 進数を 2 桁で表現した書式で行います。例えば、文字列「D8PNS1」は「4438504E5331」と入力します。

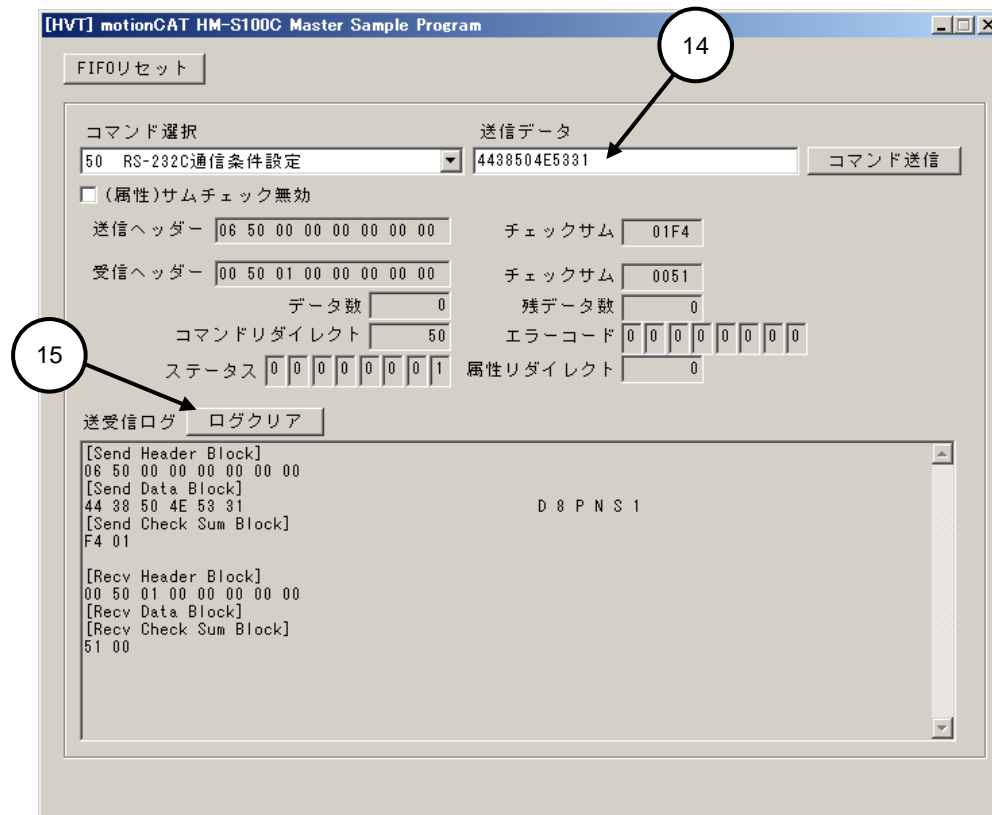



図 2.8-10 サンプル操作画面(パラメータが不可するコマンド実行時)

⑮の<ログクリア>ボタンをクリックすると、ログ表示領域が消去されます。また表示ログ数が一定量を越すと、自動的消去します。をクリックするとサンプル操作画面を閉じて初期画面に戻ります。

#### (4) サンプルプログラムの終了


初期画面で、をクリックするとサンプルプログラムを終了します。



図 2.8-11 初期画面(終了画面)

## 3. ソフトウェアの準備

### 3.1 マスターボードを複数枚使用する場合

MCAT 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。

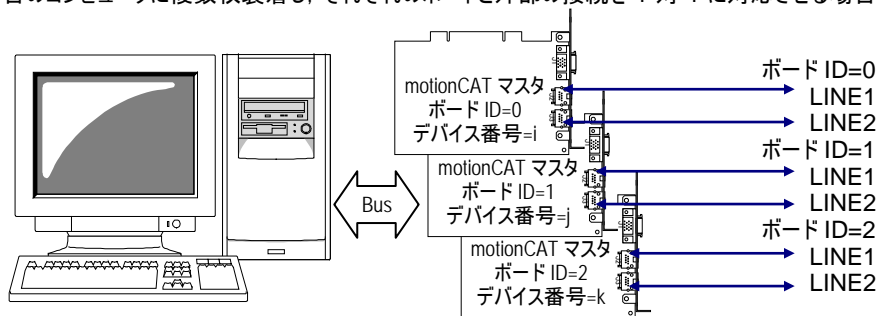


図 3.1-1 ボードを複数枚使用

#### 3.1.1 ボードのデバイス番号

PCI では BIOS 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より BIOS 側で決めたデバイス番号が割振られます。これをデバイス番号と呼びます。

#### 3.1.2 ボードIDの使用

デバイス番号は BIOS 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認識しにくい場合があります。

個々のボードとソフトウェアを対応しやすくするために MCAT には「ボード ID 設定ロータリ SW」が設けられています。

ボード ID は 0h~Fh(0~15)が設定出来ますので MCAT を 16 枚まで扱えます。

ボード ID の設定については「motionCAT シリーズユーザズマニュアル<導入編>」を参照して下さい。

### 3.2 アプリケーション作成準備

#### 3.2.1 Microsoft Visual C++ (6.0 以上)アプリケーション作成準備

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	himnt520.h	関数定義ヘッダファイル
2	himnt520.lib	関数インポートライブラリファイル
3	hpxl1a.c	モーションモジュール(位置決め)用ライブラリ関数ソースコードファイル
4	hpxl1a.h	モーションモジュール(位置決め)用ライブラリ関数定義ヘッダファイル

またマスタを制御するソースコードには各ヘッダファイルをインクルードします。

例. `#include "hpxl1a.h"` // himnt520.h は、このファイルの中で"#include"されています。

#### 3.2.2 Microsoft Visual Basic(.NET2003 以上)アプリケーション作成準備

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	himnt520.bas	関数定義標準モジュール
2	hpxl1a.bas	モーションモジュール(位置決め)用ライブラリ関数標準モジュール

### 3.2.3 Microsoft Visual Basic 6.0 アプリケーション作成準備

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	himnt520.vb	関数定義ファイル
2	hpxl1a.vb	モーションモジュール(位置決め)用ライブラリ関数ソースコードファイル

## 3.3 ボードアクセス方法

ドライバ関数・ライブラリ関数では複数の MCAT を制御することができます。制御したい MCAT にアクセスするためには、まずこの MCAT をオープンして、デバイスハンドルを取得します。MCAT をオープンするために必要なハードウェアリソース(※)をデバイス情報と呼びます。※: ハードウェアリソース・・・ボード ID, ベースアドレス, 割込番号等

### 3.3.1 デバイス情報構造体

#### (1) Microsoft Visual C++ (6.0 以上)

```
// デバイス情報構造体
typedef struct __HMNT520INFO {           // 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
    DWORD    dwBoardID;                 // ボード ID(0~15)
} HMNT520INFO, /PHMNT520INFO;
```

#### (2) Microsoft Visual Basic(.NET2003 以上)

```
Public Structure HMNT520INFO
    Dim dwBusNumber As Integer ' バス番号
    Dim dwDeviceNumber As Integer ' デバイス番号
    Dim dwBaseAddress2 As Integer ' ベースアドレス 2
    Dim dwBaseAddress3 As Integer ' ベースアドレス 3
    Dim dwBaseAddress4 As Integer ' ベースアドレス 4
    Dim dwIrqNo As Integer ' IRQ 番号
    Dim dwNumber As Integer ' 管理番号
    Dim dwBoardID As Integer ' ボード ID(0~15)
End Structure
```

**(3) Microsoft Visual Basic 6.0**

```
Public Type HMNT520INFO
    dwBusNumber      As Long      ' バス番号
    dwDeviceNumber   As Long      ' デバイス番号
    dwBaseAddress2   As Long      ' ベースアドレス 2
    dwBaseAddress3   As Long      ' ベースアドレス 3
    dwBaseAddress4   As Long      ' ベースアドレス 4
    dwIrqNo          As Long      ' IRQ 番号
    dwNumber         As Long      ' 管理番号
    dwBoardID        As Long      ' ボードID(0~15)
End Type
```

**3.3.2 モジュール情報構造体**

モジュール数確認のために次に示す HMNTMDLINFO 型構造体を用意します。

**(1) Microsoft Visual C++ (6.0 以上)**

```
typedef struct __HMNTMDLINFO {          // モジュール情報
    BYTE  byLine1g9002a;                // 予約
    BYTE  byLine1g9002b;                // 予約
    BYTE  byLine1g9002c;                // 接続される DIO モジュール数
    BYTE  byLine1g9002d;                // 予約
    BYTE  byLine1g9002e;                // 予約
    BYTE  byLine1g9x03a;                // 接続されるモーションモジュール数
    BYTE  byLine1g9004a;                // 予約
    BYTE  byLine1g9004b;                // 予約
    BYTE  byLine2g9002a;                // 予約
    BYTE  byLine2g9002b;                // 予約
    BYTE  byLine2g9002c;                // 予約
    BYTE  byLine2g9002d;                // 予約
    BYTE  byLine2g9002e;                // 予約
    BYTE  byLine2g9x03a;                // 予約
    BYTE  byLine2g9004a;                // 予約
    BYTE  byLine2g9004b;                // 予約
} HUBMDLINFO;
```

**(2) Microsoft Visual Basic(.NET2003 以上)**

```

Public Structure HMNTMDLINFO
    Dim byLine1g9002a As Byte ' 予約
    Dim byLine1g9002b As Byte ' 予約
    Dim byLine1g9002c As Byte ' 接続される DIO モジュール数
    Dim byLine1g9002d As Byte ' 予約
    Dim byLine1g9002e As Byte ' 予約
    Dim byLine1g9x03a As Byte ' 接続されるモーションモジュール数
    Dim byLine1g9004a As Byte ' 予約
    Dim byLine1g9004b As Byte ' 予約
    Dim byLine2g9002a As Byte ' 予約
    Dim byLine2g9002b As Byte ' 予約
    Dim byLine2g9002c As Byte ' 予約
    Dim byLine2g9002d As Byte ' 予約
    Dim byLine2g9002e As Byte ' 予約
    Dim byLine2g9x03a As Byte ' 予約
    Dim byLine2g9004a As Byte ' 予約
    Dim byLine2g9004b As Byte ' 予約
End Structure

```

**(3) Microsoft Visual Basic 6.0**

```

Public Type HMNTMDLINFO
    byLine1g9002a As Byte ' 予約
    byLine1g9002b As Byte ' 予約
    byLine1g9002c As Byte ' 接続される DIO モジュール数
    byLine1g9002d As Byte ' 予約
    byLine1g9002e As Byte ' 予約
    byLine1g9x03a As Byte ' 接続されるモーションモジュール数
    byLine1g9004a As Byte ' 予約
    byLine1g9004b As Byte ' 予約
    byLine2g9002a As Byte ' 予約
    byLine2g9002b As Byte ' 予約
    byLine2g9002c As Byte ' 予約
    byLine2g9002d As Byte ' 予約
    byLine2g9002e As Byte ' 予約
    byLine2g9x03a As Byte ' 予約
    byLine2g9004a As Byte ' 予約
    byLine2g9004b As Byte ' 予約
End Type

```

## 3.4 ボードアクセスの準備手順と終了処理

### 3.4.1 準備手順

#### (1)ボード毎にデバイスオープン

まず MCAT の枚数を取得し、ボード認識用のデータ構造体を枚数分用意します。

次に MCAT のデバイス情報を取得し、制御したい MCAT 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 MCAT がオープンされます。

また、デバイスオープン関数はこの MCAT 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。

MCAT 枚数が 2 枚以上の場合には、個々の MCAT 毎にこの処理を行います。

- ◆mnt520\_\_GetMstBrdCount( )     ...MCAT の枚数取得
- ◆mnt520\_\_GetMstBrdInfo( )     ...MCAT の PCI デバイス情報取得
- ◆mnt520\_\_OpenMstBrd( )        ...MCAT のオープン処理

#### (2)初期設定

MCAT のオープンに成功した後は、システム通信を開始し、ローカルデバイス情報を自動設定します。

ポートデータの初期値を設定し、サイクリック通信を開始します。サイクリック通信開始後、データ通信も可能になりますので、ローカルデバイスの初期化を行います。

- ◆mnt520\_\_rSysLclInfo( )     ...ローカルデバイス(DIO, モーション)の情報取得
- ◆mnt520\_\_wCenCmd( )        ...センターデバイスへのコマンド書込み
- ◆mnt520\_\_rCenMsts( )        ...センターデバイスからのセンターメインステータス読出し
- ◆mnt520\_\_wIoPortB( )        ...DIO デバイスのポートへの書込み
- ◆mnt520\_\_wIoPortW( )        ...DIO デバイスのポートへの書込み
- ◆mnt520\_\_wLclSetInt( )     ...ローカルデバイスのポート入力変化割込設定
- ◆mnt520\_\_wPclPortB( )     ...モーションデバイスのポートへの書込み
- ◆mnt520\_\_wPclSetInt( )     ...モーションデバイスのポート入力変化割込設定
- ◆mnt520\_\_wPclReg( )        ...モーションデバイスのレジスタへの書込み

#### (3)運用

初期設定終了後はデータ通信、サイクリック通信によりローカルデバイスを制御します。

- ◆mnt520\_\_rIoPortB( )        ...DIO デバイスのポートからの読出し
- ◆mnt520\_\_wIoPortB( )        ...DIO デバイスのポートへの書込み
- ◆mnt520\_\_rIoPortW( )        ...DIO デバイスのポートからの読出し
- ◆mnt520\_\_wIoPortW( )        ...DIO デバイスのポートへの書込み
- ◆mnt520\_\_rLclInt( )        ...ローカルデバイスのポート入力変化割込読出し
- ◆mnt520\_\_wLclInt( )        ...ローカルデバイスのポート入力変化割込リセット
- ◆mnt520\_\_rPclPort( )        ...モーションデバイスのポートからの読出し
- ◆mnt520\_\_wPclPort( )        ...モーションデバイスのポートへの書込み
- ◆mnt520\_\_rPclMsts( )        ...モーションデバイスのモーションメインステータス読出し
- ◆mnt520\_\_wPclCmd( )        ...モーションデバイスへのコマンド書込み
- ◆mnt520\_\_rPclReg( )        ...モーションデバイスのレジスタからの読出し
- ◆mnt520\_\_wPclReg( )        ...モーションデバイスのレジスタへの書込み

### 3.4.2 終了処理

#### (4)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、システム上の終了処理を終えてから、オープンしたデバイスの「クローズ処理」を行います。

MCAT 枚数が 2 枚以上の場合には、個々の MCAT 毎にこの処理を行います。

- ◆mnt520\_\_CloseMstBrd( )     ...MNT520 のクローズ処理

## 3.5 関数の戻り値

No	記号表記	16進数	内容
1	NO_ERROR	0x00000000	正 常
2	NOT_FOUND	0x00000001	デバイスドライバ未インストール, 未接続, OS に認識されていない.
3	ALREADY_OPENED	0x00000002	既にオープン済のデバイスをオープン.
4	INSUFFICIENT_MEMORY	0x00000004	デバイス情報格納メモリが不足.
5	INVALID_HANDLE	0x00000008	無効なデバイスハンドルを指定.
6	NOT_READY	0x00000010	デバイスの入出力ポートが使用できない.
7	ILLEGAL_DEVICE	0x00000020	ボード固有情報が不正.
8	ILLEGAL_PARAM	0x00000100	不正なパラメータ.
以下は Motionnet に起因			
9	CYC_COM_ERROR	0x00010000	サイクリック通信エラー.
10	CYC_COM_STOP	0x00020000	サイクリック通信停止中.
11	DATA_COM_ERROR	0x00040000	データ通信エラー.
12	COM_OTHER_ERROR	0x00080000	その他の通信エラー(センター割込ステータスで詳細を確認)
以下はライブラリ関数のみ			
13	MODULE_OVERCOUNT	0x00100000	通信ラインに接続されているモジュールの数が保証外.
14	MODULE_COUNT_ERROR	0x00200000	モジュールの数が合わない. モジュールの接続, 電源供給状態, ケーブル等確認
15	ILL_ACCESS_COM	0x00400000	通信中の不正なアクセス
16	SEND_DATA_NONUSE	0x00800000	未使用デバイスにデータ送信
17	LINE1_ERROR	0x10000000	ライン 1 の系統でのエラーが発生
18	LINE2_ERROR	0x20000000	ライン 2 の系統でのエラーが発生
19	OTHER_ERROR	0x80000000	その他のエラー

表 3.5-1 関数の戻り値

## 4. ライブラリ関数

## 4.1 モーションモジュール用ライブラリ関数

ライブラリ関数はドライバ関数で構成され、モーションモジュールの初期化、原点復帰、位置決め動作等の基本的な動作を制御することができます。また各開発言語用ライブラリ関数の仕様は同様になっています。

関数の戻り値は「[3.5 関数の戻り値](#)」を参照してください。

## 4.2 モーションモジュール用ライブラリ関数一覧

### 4.2.1 デバイス関係

No	関数名	機能	備考
1	hpx_GetDevInfo	マスターボードのデバイス個数とデバイス情報取得	
2	hpx_DevOpen	マスターボードのオープンとモーションモジュール初期化	
3	hpx_DevOpenEx	マスターボードのオープンとモーションモジュール初期化	モジュールチェック有
4	hpx_DevClose	マスターボードのクローズ	

### 4.2.2 初期設定

No	関数名	機能	備考
5	hpx_InitMotionMod	P, C モジュールの初期化	
6	hpx_SetOrgMode	原点復帰モードの設定	
7	hpx_SetEls	ELS 入力の設定	
8	hpx_SetOls	OLS 入力の設定	
9	hpx_SetDls	DLS 入力の設定	
10	hpx_SetLtc	Ltch 入力の設定	
11	hpx_SetClr	CLR 入力の設定	
12	hpx_SetCmp	コンパレータ条件の設定	
13	hpx_SetEz	エンコーダ Z 相の設定	
14	hpx_SetInpos	INPOS 入力の設定	
15	hpx_SetSvAlm	SVALM 入力の設定	
16	hpx_SetSvCtrCl	SVCTRCL 出力の設定	
17	hpx_SetSvRdy	SVRDY 入力の設定	
18	hpx_SetCmdPulse	指令パルス出力形式の設定	
19	hpx_SetAccProfile	S 字／直線加減速の切替	
20	hpx_SetAutoDec	減速開始点の計算の自動／マニュアル切替	
21	hpx_SetSls	ソフトウェアリミットの設定	
22	hpx_SetCtr3	カウンタ 3 の入力ソースの設定	
23	hpx_SetFhAdj	FH 補正機能の ON/OFF	

### 4.2.3 状態読み出し

No	関数名	機能	備考
24	hpx_ReadMainSts	メインステータスの読み出し	
25	hpx_ReadErrorSts	エラーステータスの読み出し	
26	hpx_ReadEventSts	イベントステータスの読み出し	
27	hpx_ReadExSts	拡張ステータスの読み出し	
28	hpx_ReadOutp	出力ポート状態読み出し	
29	hpx_ReadSpd	指令速度の読み出し	
30	hpx_ReadCtr	カウンタの読み出し	

## 4.2.4 動作設定

No	関数名	機能	備考
31	hpx_SetFLSpd	ベース速度	
32	hpx_SetAuxSpd	補助速度の設定	
33	hpx_SetAccRate	加速レートの設定	
34	hpx_SetDecRate	減速レートの設定	
35	hpx_SetMult	倍率設定値の設定	
36	hpx_SetEventMask	イベントマスクの設定	
37	hpx_SetDecPoint	減速開始点の設定	
38	hpx_WritOpeMode	動作モードの設定	
39	hpx_WritSta	STA 入力時の動作設定	
40	hpx_WritStp	STP 入力時の動作設定と STP 自動出力の設定	
41	hpx_WritFHSpd	動作速度の設定	
42	hpx_WritPos	移動量の設定	
43	hpx_WritCtr	カウンタリセット	

## 4.2.5 動作制御指令

No	関数名	機能	備考
44	hpx_DecStop	減速停止	
45	hpx_QuickStop	即停止	
46	hpx_EmgStop	非常停止	
47	hpx_AccStart	加速スタート	
48	hpx_CnstStartFH	FH 定速スタート	
49	hpx_CnstStartFL	FL 定速スタート	
50	hpx_MvAccStart	移動量設定+加速スタート	
51	hpx_MvCnstStartFH	移動量設定+FH 定速スタート	
52	hpx_MvCnstStartFL	移動量設定+FL 定速スタート	
53	hpx_SetGroup	グループ設定	
54	hpx_GrpStop	グループ停止	
55	hpx_GrpAccStart	グループ加速スタート	
56	hpx_GrpCnstStartFH	グループ FH 定速スタート	
57	hpx_GrpCnstStartFL	グループ FL 定速スタート	
58	hpx_SvOn	サーボオン	
59	hpx_SvOff	サーボオフ	
60	hpx_SvResetOn	サーボリセットオン	
61	hpx_SvResetOff	サーボリセットオフ	
62	hpx_SvTlOn	サーボトルク制限オン	
63	hpx_SvTlOff	サーボトルク制限オフ	
64	hpx_SvGainOn	サーボゲイン切替オン	
65	hpx_SvGainOff	サーボゲイン切替オフ	
66	hpx_PMon	パルスモータ励磁オン	
67	hpx_PMOff	パルスモータ励磁オフ	

## 4.2.6 計算関数

No	関数名	機能	備考
68	hpx_CalAccRate	加減速レート計算	
69	hpx_CalDecPoint	減速開始点計算	

### 4.3 モーションモジュール用ライブラリ関数一覧

ここでは VC++表記で説明します。VB, VB.NET を使用される場合、データ型の対応は下表の通りです。

言語	VC++	VB	VB.NET
データ型			
8bit	BYTE	Byte	Byte
16bit	WORD	Integer	Short
32bit	DWORD	Long	Integer
データ 例	0x0000	&H0	&H0
	0x1000	&H1000	&H1000

#### 4.3.1 デバイス関係

##### (1) hpx\_GetDevInfo() ボード枚数と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 MCAT の枚数,及びデバイス情報を取得します。
言語	書 式
VC++	DWORD hpx_GetDevInfo( DWORD* HpcNum, HMNT520INFO* MntInfo );
引 数	説 明
MntNum	MCAT の枚数
MntInfo	デバイス情報構造体

##### (2) hpx\_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 MCAT をオープンし, 他の MCAT 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 MCAT にアクセスするために使用します。またオープンした MCAT に接続されたモーションモジュールの初期化(※1)をします。
言語	書 式
VC++	DWORD hpx_DevOpen( DWORD* hDevID, HMNT520INFO* MntInfo );
引 数	説 明
hDevID	デバイスハンドル
MntInfo	オープンする MCAT のデバイス情報
備 考	<p>&lt;呼び出し例&gt;            パソコンに MCAT が 2 枚装着されていることを想定します。            デバイス情報構造体として HMNT520INFO 型の配列 MntInfo[2]を準備し,この中には既に hpx_GetDevInfo 関数により全 MCAT のデバイス情報が入っているものとします。</p> <pre>           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 = hpx_DevOpen( hDevID[0], &amp;MntInfo[0] );    //1 番目のデバイス情報           ret = hpx_DevOpen( hDevID[1], &amp;MntInfo[1] );    //2 番目のデバイス情報           </pre>

## (3) hpx\_DevOpenEx() 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 MCAT をオープンし、他の MCAT 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 MCAT にアクセスするために使用します。またオープンした MCAT に接続されたモーションモジュールの初期化(※1)をします。
----	--

言語	書式
VC++	DWORD hpx_DevOpen( DWORD* hDevID, HMNT520INFO* MntInfo, HMNTMDLINFO* MdlInfo WORD w);

引数	説明
hDevID	デバイスハンドル
MntInfo	オープンする MCAT のデバイス情報
MdlInfo	オープンする MCAT のモジュール情報
w	予約(常に 0 を設定してください)

備考	<p>&lt;呼び出し例&gt;          パソコンに MCAT が 2 枚装着されていることを想定します。          デバイス情報構造体として HMNT520INFO 型の配列 MntInfo[2]、モジュール情報構造体として MNTMDLINFO 型の配列 MdlInfo[2]を準備し、MntInfo の中には既に hpx_GetDevInfo 関数により全 MCAT のデバイス情報、MdlInfo の中には使用するモジュール種別毎の数が入っているものとします。</p> <pre>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 = hpx_DevOpenEx( hDevID[0], &amp;MntInfo[0], &amp;MdlInfo[0], 0 ); //1 番目のデバイス情報 ret = hpx_DevOpenEx( hDevID[1], &amp;MntInfo[1], &amp;MdlInfo[1], 0 ); //2 番目のデバイス情報</pre>
----	---

## ※1. モーションモジュール レジスタの初期値

レジスタ	内容	レジスタ初期値	備考
RFL	ベース速度	200	200pps
RFH	動作速度	2000	2000pps
RUR	加速レート	1387	200pps→2,000pps の加減速時間約 500msec(直線加減速時)
RMG	速度倍率	199	1 倍
RFA	補助速度	200	200pps
RENV1	環境設定 1	0x01434004	指令パルス出力形式: CW/CCW, SVCTRCL 自動出力しない, SVCTRCL 出力パルス幅:13ms, DLS,OLS,SVALM:B 接, ELS,SVALM 入力時即停止, DLS ラッチしない, SVRDY,INPOS:A 接, LATC,CLR:立下りエッジ
RENV2	環境設定 2	0x000004FF	サーボ I/F 出力設定, エンコーダ入力 4 通倍
RENV3	環境設定 3	0x00700002	原点復帰モード 2(OLS+Z), 原点復帰完了時カウンタ 1~3 をクリア
RIRQ	イベントマスク設定	1	正常停止時
上記以外		0	

## (4) hpx\_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マスターボードをクローズします。以降、このデバイスハンドルは無効となります。
----	--

言語	書式
VC++	DWORD hpx_DevClose( DWORD hDevID);

引数	説明
hDevID	デバイスハンドル

備考	デバイスのクローズ処理ではデバイスハンドルを無効にするのみで、パルス出力の停止や、サーボオフは行いません。モーションモジュールに対する終了処理をした後でデバイスをクローズして下さい。
----	---

## 4.3.2 初期設定

## (5) hpx\_InitMotionMod() 位置決めモジュールの初期化

機能	指定した位置決めモジュールを初期化します。
言語	書式
VC++	DWORD hpx_InitMotionMod(DWORD hDevID, WORD wLine, WORD wMid );
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)

## (6) hpx\_SetOrgMode() 原点復帰モードの設定

機能	指定したモジュールの原点復帰モードを設定します。
言語	書式
VC++	DWORD hpx_SetOrgMode(DWORD hDevID, WORD wLine, WORD wMid, WORD wMode );
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)
wMode	原点復帰モード(0-12)

## (7) hpx\_SetEIs() ELS入力の設定

機能	指定したモジュールの ELS 入力の設定をします。
言語	書式
VC++	DWORD hpx_SetEIs(DWORD hDevID, WORD wLine, WORD wMid, WORD wPol, WORD wMot );
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)
wPol	入力極性(0:B 接/1:A 接)
wMot	入力時動作(0:即停止/1:減速停止)

## (8) hpx\_SetOIs() OLS入力の設定

機能	指定したモジュールの OLS 入力の設定をします。
言語	書式
VC++	DWORD hpx_SetOIs(DWORD hDevID, WORD wLine, WORD wMid, WORD wPol );
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)
wPol	入力極性(0:B 接/1:A 接)

**(9) hpx\_SetDis() DLS入力の設定**

機能	指定したモジュールの DLS 入力の設定をします。
言語	書式
VC++	DWORD hpx_SetDis( DWORD hDevID, WORD wLine, WORD wMid, WORD wEnbl, WORD wPol, WORD wMot, WORD wLtc );
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)
wEnbl	DLS 使用(0:不使用 / 1:使用)
wPol	入力極性(0:B 接/1:A 接)
wMot	入力時動作(0:減速のみ/1:減速停止)
wLtc	ラッチ設定(0:ラッチしない / 1:ラッチする)

**(10) hpx\_SetLtc() LATCH入力の設定**

機能	指定したモジュールの LATCH 入力の設定をします。
言語	書式
VC++	DWORD hpx_SetLtc(DWORD hDevID, WORD wLine, WORD wMid, WORD wEdg );
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID
wEdg	入力エッジ(0:フォトカプラ ON→OFF, 1: フォトカプラ OFF→ON)

**(11) hpx\_SetClr() CLR入力の設定**

機能	指定したモジュールの CLR 入力の設定をします。
言語	書式
VC++	DWORD hpx_SetClr(DWORD hDevID, WORD wLine, WORD wMid, WORD wSpc, WORD wCtr );
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)
wSpc	カウンタクリアのタイミング (0:フォトカプラ ON→OFF でクリア, 1: フォトカプラ OFF→ON でクリア, 2: フォトカプラ OFF でクリア, 3: フォトカプラ ON でクリア)
wCtr	CTR 入力によりクリアされるカウンタの選択

**(12) hpx\_SetCmp() コンパレータ条件の設定**

機能	指定したモジュールのコンパレータ条件の設定をします。
言語	書式
VC++	DWORD hpx_SetCmp(DWORD hDevID, WORD wLine, WORD wMid, WORD wCtr, WORD wCon, WORD wIndex, long lCmp);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号1/1:ライン番号2)
wMid	モジュールID(0-63)
wCtr	コンパレータ比較カウンタ
wCon	コンパレータ条件
wIndex	同期(定ピッチ)出力有効/無効
lCmp	コンパレータ比較データ

**(13) hpx\_SetEz() エンコーダZ相の設定**

機能	指定したモジュールのエンコーダZ相入力の設定をします。
言語	書式
VC++	DWORD hpx_SetEz(DWORD hDevID,WORD wLine, WORD wMid,WORD wCount,WORD wEdg);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号1/1:ライン番号2)
wMid	モジュールID(0-63)
wCount	原点復帰時Z相カウントアップ回数(0:1回~15:16回)
wEdg	Z相カウント仕様(0:OFF→ON / 1:ON→OFF)

**(14) hpx\_SetInpos() INPOS入力の設定**

機能	指定したモジュールのINPOS入力の設定をします。
言語	書式
VC++	DWORD hpx_SetInpos(DWORD hDevID,WORD wLine, WORD wMid,WORD wEnbl, WORD wPol);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号1/1:ライン番号2)
wMid	モジュールID(0-63)
wEnbl	INPOS制御(0:不使用/1:使用)
wPol	入力極性(0:B接/1:A接)

**(15) hpx\_SetSvAlm() SVALM入力の設定**

機能	指定したモジュールの SVALM 入力の設定をします。
言語	書式
VC++	DWORD hpx_SetSvAlm(DWORD hDevID,WORD wLine,WORD wMid,WORD wPol,WORD wMot);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
wPol	入力仕様(0:B 接 / 1:A 接)
wMot	入力時停止方法(0:即停止/1:減速停止)

**(16) hpx\_SetSvCtrCl() SVCTRCL出力の設定**

機能	指定したモジュールの SVCTRCL 出力の設定をします。
言語	書式
VC++	DWORD hpx_SetSvCtrCl( DWORD hDevID, WORD wLine, WORD wMid, WORD wSpc);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
wSpc	0:不使用 / 1:原点復帰完了時自動出力 / 2:異常停止時自動出力 / 3: 原点復帰完了時, 異常停止時自動出力

**(17) hpx\_SetSvRdy() SVRDY入力の設定**

機能	指定したモジュールの SVRDY 入力の設定をします。
言語	書式
VC++	DWORD hpx_SetSvRdy( DWORD hDevID, WORD wLine, WORD wMid, WORD wPol);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
wPol	入力極性(0:B 接/1:A 接)

**(18) hpx\_SetCmdPulse() 指令パルス出力形式の設定**

機能	指定したモジュールの指令パルス出力形式の設定をします。
言語	書式
VC++	DWORD hpx_SetCmdPulse(DWORD hDevID, WORD wLine, WORD wMid, WORD wSpc);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
wSpc	指令パルス出力形式 (0: 個別 CW/CCW / 1:共通 パルス列+方向信号(+:H / -:L) / 2:共通 パルス列+方向信号(+:L /

	-:H)
--	------

**(19) hpx\_SetAccProfile() S字/直線加減速の切り替え**

機能	指定したモジュールの加減速方式を設定します。
言語	書式
VC++	DWORD hpx_SetAccProfile( DWORD hDevID, WORD wLine, WORD wMid, WORD wAcc);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)
wAcc	加減速形式(0:直線加減速 / 1:S 字加減速)

**(20) hpx\_SetAutoDec() 減速開始点の計算の自動/手動切り替え**

機能	指定したモジュールの減速開始点の計算方式を設定します。
言語	書式
VC++	DWORD hpx_SetAutoDec(DWORD hDevID, WORD wLine, WORD wMid, WORD wCalc);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)
wCalc	減速開始点計算方式 (0:自動 / 1:手動)

**(21) hpx\_SetSIs() ソフトウェアリミットの設定**

機能	指定したモジュールのソフトウェアリミットの設定をします。
言語	書式
VC++	DWORD hpx_SetSIs( DWORD hDevID, WORD wLine, WORD wMid, long IPsl, long IMsl, WORD wEnbl, WORD wMot);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)
IPsl	+SLS
IMsl	-SLS
wEnbl	SLS 有効/無効(0:無効 / 1:有効)
wMot	SLSon 時の動作(0:即停止/1:減速停止)

**(22) hpx\_SetCtr3() カウンタ 3 の入力ソースの設定**

機能	指定したモジュールのカウンタ 3(汎用カウンタ)の入力ソースを設定します。
言語	書式
VC++	DWORD hpx_SetCtr3(DWORD hDevID, WORD wLine, WORD wMid, WORD wSrc);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
wSrc	CTR3 の入力ソース (0:指令位置 / 1:機械位置 / 2:手動パルス入力 / 3:20MHz の 1/4096 分周クロックカウント / 4:偏差カウント)

**(23) hpx\_SetFhAdj() FH補正機能のON/OFF**

機能	指定したモジュールの FH 補正機能を ON/OFF します。
言語	書式
VC++	DWORD hpx_SetFhAdj(DWORD hDevID, WORD wLine, WORD wMid, WORD wEnbl);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID
wEnbl	FH 補正(0:OFF / 1:ON)

**4.3.3 状態読み出し****(24) hpx\_ReadMainSts() モーションメインステータスの読み出し**

機能	指定したモーションモジュールのモーションメインステータスの読み出しと割り込みリセットを行います。
言語	書式
VC++	DWORD hpx_ReadMainSts(DWORD hDevID, WORD wLine, WORD wMid, WORD* wMMSts);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
wMMSts	モーションメインステータス
備考	<モーションメインステータスの内容> bit 0 (SINT) MMSTS の bit1,2,3 のいずれかが'1'で'1' bit 1 (SEND) 動作停止時'1' bit 2 (SERR) 異常停止,位置のオーバーライド失敗,エンコーダ信号異常時'1' bit 3 (SEVT) RIRQ で設定されたイベント発生時'1' bit 8 (SBSY) パルス出力中'1'

## (25) hpx\_ReadErrorSts() エラーステータスの読出し

機 能	指定したモジュールのエラーステータスの読み出しとクリアを行います。
言語	書 式
VC++	DWORD hpx_ReadErrorSts(DWORD hDevID, WORD wLine, WORD wMid, DWORD* dwRest);
引 数	説 明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)
dwRest	エラーステータス
備 考	<p>&lt;エラーステータスの内容&gt;</p> <ul style="list-style-type: none"> <li>bit 0(ESC1) +SLS による停止時</li> <li>bit 1(ESC2) -SLS による停止時</li> <li>bit 2(ESC3) CMP3 条件成立による停止時</li> <li>bit 3(ESPL) +ELS 入力 ON による停止時</li> <li>bit 4(ESML) -ELS 入力 ON による停止時</li> <li>bit 5(ESAL) SVALM 入力 ON による停止時</li> <li>bit 6(ESSP) STP 入力 ON による停止時</li> <li>bit 7(ESEM) EMG 入力 ON による停止時</li> <li>bit 8(ESSD) DLS 入力 ON による減速停止時</li> <li>bit 9(ESPO) 手動パルスバッファカウンタのオーバーフロー発生時</li> <li>bit10(ESNT) 断線検出による停止時</li> <li>bit11 未定義</li> <li>bit12(ESOR) 位置のオーバーライド失敗(停止中に位置のオーバーライドを実行しようとした)</li> <li>bit13(ESEE) エンコーダ入力の同時変化時(停止しない)</li> <li>bit14(ESPE) エンコーダ入力の同時変化時(停止しない)</li> </ul>

## (26) hpx\_ReadEventSts() イベントステータスの読出し

機 能	指定したモジュールのイベントステータスの読み出しとクリアを行います。
言語	書 式
VC++	DWORD hpx_ReadEventSts(DWORD hDevID, WORD wLine, WORD wMid, DWORD* dwRist);
引 数	説 明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)
dwRist	イベントステータス
備 考	<イベントステータスの内容> bit 0 (ISEN) 正常停止時 bit 1 (ISUS) 加速開始時 bit 2 (ISUE) 加速終了時 bit 3 (ISDS) 減速開始時 bit 4 (ISDE) 減速終了時 bit 5 (ISC1) CMP1 条件成立時 bit 6 (ISC2) CMP2 条件成立時 bit 7 (ISC3) CMP3 条件成立時 bit 8 (ISCL) CLR 入力によるカウント値のリセット時 bit 9 (ISLT) LATCH 入力によるカウント値のラッチ時 bit10 (ISOL) OLS 入力によるカウント値のラッチ時 bit11 (ISSD) DLS 入力 ON 時 bit12 (ISSA) STA 入力 ON 時 bit13 (ISNA) グループスタート時 bit14 (ISNP) グループ停止時

## (27) hpx\_ReadExSts() 拡張ステータスの読出し

機能	指定したモジュールの拡張ステータスを読み出します。																																																												
言語	書式																																																												
VC++	DWORD hpx_ReadExSts(DWORD hDevID, WORD wLine, WORD wMid, DWORD* dwRsts);																																																												
引数	説明																																																												
hDevID	デバイスハンドル																																																												
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)																																																												
wMid	モジュール ID(0-63)																																																												
dwRsts	拡張ステータス																																																												
備考	<p>&lt;拡張ステータスの内容&gt;</p> <table> <tr> <td>bit 3-0</td> <td>0000 停止中</td> <td>bit 4(SDIR) 動作方向(0:+方向 1:-方向)</td> </tr> <tr> <td>0001 STA 入力待ち</td> <td>0010 SVCTRCL タイマ完了待ち</td> <td>bit 5(SALM) SVALMon</td> </tr> <tr> <td>0100 バックラッシュ補正動作中</td> <td>0101 方向変化タイマ完了待ち</td> <td>bit 6(SPEL) +ELSon</td> </tr> <tr> <td>0101 手動パルサ入力待ち</td> <td>0110 FA 定速動作中</td> <td>bit 7(SMEL) -ELSon</td> </tr> <tr> <td>0110 FL 定速動作中</td> <td>1000 加速中</td> <td>bit 8(SORG) OLSon</td> </tr> <tr> <td>1000 加速中</td> <td>1001 FH 定速動作中</td> <td>bit 9(SSD) DLSon(ラッチ状態)</td> </tr> <tr> <td>1001 FH 定速動作中</td> <td>1010 減速中</td> <td>bit10(SDIN) DLSon(端子状態)</td> </tr> <tr> <td>1010 減速中</td> <td>1011 INPOS 待ち</td> <td>bit11(SSTA) STAon</td> </tr> <tr> <td>1011 INPOS 待ち</td> <td>1111 スタート制御中</td> <td>bit12(SSTP) STPon</td> </tr> <tr> <td>1111 スタート制御中</td> <td></td> <td>bit13(SEMG) EMGon</td> </tr> <tr> <td></td> <td></td> <td>bit14(SPCS) SVRDY(PCS)on</td> </tr> <tr> <td></td> <td></td> <td>bit15(SERC) SVCTRCLon</td> </tr> <tr> <td></td> <td></td> <td>bit16(SEZ) ENCZon</td> </tr> <tr> <td></td> <td></td> <td>bit17(SCLR) CLRon</td> </tr> <tr> <td></td> <td></td> <td>bit18(SLTC) LATCHon</td> </tr> <tr> <td></td> <td></td> <td>bit19(SINP) INPOSon</td> </tr> <tr> <td></td> <td></td> <td>bit20(SCP1) CMP1 比較条件成立時</td> </tr> <tr> <td></td> <td></td> <td>bit21(SCP2) CMP2 比較条件成立時</td> </tr> <tr> <td></td> <td></td> <td>bit22(SCP3) CMP3 比較条件成立時</td> </tr> <tr> <td></td> <td></td> <td>bit23(SPLS) パルス出力 ON</td> </tr> </table>	bit 3-0	0000 停止中	bit 4(SDIR) 動作方向(0:+方向 1:-方向)	0001 STA 入力待ち	0010 SVCTRCL タイマ完了待ち	bit 5(SALM) SVALMon	0100 バックラッシュ補正動作中	0101 方向変化タイマ完了待ち	bit 6(SPEL) +ELSon	0101 手動パルサ入力待ち	0110 FA 定速動作中	bit 7(SMEL) -ELSon	0110 FL 定速動作中	1000 加速中	bit 8(SORG) OLSon	1000 加速中	1001 FH 定速動作中	bit 9(SSD) DLSon(ラッチ状態)	1001 FH 定速動作中	1010 減速中	bit10(SDIN) DLSon(端子状態)	1010 減速中	1011 INPOS 待ち	bit11(SSTA) STAon	1011 INPOS 待ち	1111 スタート制御中	bit12(SSTP) STPon	1111 スタート制御中		bit13(SEMG) EMGon			bit14(SPCS) SVRDY(PCS)on			bit15(SERC) SVCTRCLon			bit16(SEZ) ENCZon			bit17(SCLR) CLRon			bit18(SLTC) LATCHon			bit19(SINP) INPOSon			bit20(SCP1) CMP1 比較条件成立時			bit21(SCP2) CMP2 比較条件成立時			bit22(SCP3) CMP3 比較条件成立時			bit23(SPLS) パルス出力 ON
bit 3-0	0000 停止中	bit 4(SDIR) 動作方向(0:+方向 1:-方向)																																																											
0001 STA 入力待ち	0010 SVCTRCL タイマ完了待ち	bit 5(SALM) SVALMon																																																											
0100 バックラッシュ補正動作中	0101 方向変化タイマ完了待ち	bit 6(SPEL) +ELSon																																																											
0101 手動パルサ入力待ち	0110 FA 定速動作中	bit 7(SMEL) -ELSon																																																											
0110 FL 定速動作中	1000 加速中	bit 8(SORG) OLSon																																																											
1000 加速中	1001 FH 定速動作中	bit 9(SSD) DLSon(ラッチ状態)																																																											
1001 FH 定速動作中	1010 減速中	bit10(SDIN) DLSon(端子状態)																																																											
1010 減速中	1011 INPOS 待ち	bit11(SSTA) STAon																																																											
1011 INPOS 待ち	1111 スタート制御中	bit12(SSTP) STPon																																																											
1111 スタート制御中		bit13(SEMG) EMGon																																																											
		bit14(SPCS) SVRDY(PCS)on																																																											
		bit15(SERC) SVCTRCLon																																																											
		bit16(SEZ) ENCZon																																																											
		bit17(SCLR) CLRon																																																											
		bit18(SLTC) LATCHon																																																											
		bit19(SINP) INPOSon																																																											
		bit20(SCP1) CMP1 比較条件成立時																																																											
		bit21(SCP2) CMP2 比較条件成立時																																																											
		bit22(SCP3) CMP3 比較条件成立時																																																											
		bit23(SPLS) パルス出力 ON																																																											

**(28) hpx\_ReadSpd() 指令速度の読出し**

機能	指定したモジュールの指令速度を読み出します。
言語	書式
VC++	DWORD hpx_ReadSpd(DWORD hDevID, WORD wLine, WORD wMid, DWORD* dwRspd);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
dwRspd	指令速度/速度倍率[pps]
備考	

**(29) hpx\_ReadOut() 出力ポートの読出し**

機能	指定したモジュールの出力ポートを読み出します。
言語	書式
VC++	DWORD hpx_ReadOut( DWORD hDevID, WORD wLine, WORD wMid, BYTE* byData );
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
byData	読出データ
備考	

**(30) hpx\_ReadCtr() カウンタの読出し**

機能	指定したモジュールの指定したカウンタを読み出します。
言語	書式
VC++	DWORD hpx_ReadCtr (DWORD hDevID, WORD wLine WORD wMid, WORD wCtr, long* lValue);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
wCtr	読み出すカウンタ選択(1:CTR1 / 2:CTR2 / 3:CTR3)
lValue	カウンタ読出し値
備考	

## 4.3.4 動作設定

## (31) hpx\_SetFLSpd() ベース速度の設定

機能	指定したモジュールのベース速度を設定します。
言語	書式
VC++	DWORD hpx_SetFLSpd(DWORD hDevID, WORD wLine, WORD wMid, DWORD dwRfl);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
dwRfl	RFL 設定値(1-10000)

## (32) hpx\_SetAuxSpd() 補助速度の設定

機能	指定したモジュールの補助速度を設定します。
言語	書式
VC++	DWORD hpx_SetAuxSpd(DWORD hDevID, WORD wLine, WORD wMid, DWORD dwRfa);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID
dwRfa	補助速度(1-100000)

## (33) hpx\_SetAccRate() 加速レートの設定

機能	指定したモジュールの加速レートを設定します。
言語	書式
VC++	DWORD hpx_SetAccRate(DWORD hDevID, WORD wLine, WORD wMid, DWORD dwRur);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
dwRur	加速レート(1-65535)

## (34) hpx\_SetDecRate() 減速レートの設定

機能	指定したモジュールの減速レートを設定します。
言語	書式
VC++	DWORD hpx_SetDecRate(DWORD hDevID, WORD wLine, WORD wMid, DWORD dwRdr);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID
dwRdr	減速レート(0-65535)

**(35) hpx\_SetMult() 速度倍率設定値の設定**

機 能	指定したモジュールの速度倍率設定値を設定します。
言語	書 式
VC++	DWORD hpx_SetMult(DWORD hDevID, WORD wLine, WORD wMid, DWORD dwRmg);
引 数	説 明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)
dwRmg	速度倍率設定値(2-4095)
備 考	

**(36) hpx\_SetEventMask() イベントマスクの設定**

機 能	指定したモジュールのイベントマスクを設定します。
言語	書 式
VC++	DWORD hpx_SetEventMask(DWORD hDevID, WORD wLine, WORD wMid, DWORD dwRirq);
引 数	説 明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)
dwRirq	イベントマスク(RIRQ)設定 G9003:RIRQ(0-0x00007fff) / G9003:RIRQ(0-0x0001ffff)
備 考	<イベントマスクの設定> bit 0(IREN) 正常停止時 bit 1(IRUS) 加速開始時 bit 2(IRUE) 加速終了時 bit 3(IRDS) 減速開始時 bit 4(IRDE) 減速終了時 bit 5(IRC1) コンパレータ 1 条件成立時 bit 6(IRC2) コンパレータ 2 条件成立時 bit 7(IRC3) コンパレータ 3 条件成立時 bit 8(IRCL) CLR 信号入力によるカウント値のリセット時 bit 9(IRLT) LATCH 入力によるカウント値のラッチ時 bit10(IROL) OLS 入力によるカウント値のラッチ時 bit11(IRSD) DLS 入力 ON 時 bit12(IRSA) STA 入力 ON 時 bit13(IRNA) グループスタートコマンド(2x01h)受信時 bit14(IRNP) グループストップコマンド(2x02h)による停止時

**(37) hpx\_SetDecPoint() 減速開始点の設定**

機能	指定したモジュールの減速開始点を設定します。
言語	書式
VC++	DWORD hpx_SetDecPoint(DWORD hDevID, WORD wLine, WORD wMid, long IRdp);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
IRdp	減速開始点(パルス)
備考	注意:位置決めまたは補間時の減速開始点計算方法により意味が異なる (1) 自動計算時は自動計算値からのオフセット量 負の値の場合は遅めに減速を開始しベース速度より早い速度で停止 正の値の場合は早めに減速を開始しベース速度で動作してから停止 (2) 手動計算時は残移動量が設定値以下になった時に減速を開始

**(38) hpx\_WritOpMode() 動作モード書込み**

機能	指定したモジュールの動作モードを設定します。
言語	書式
VC++	DWORD hpx_WritOpMode(DWORD hDevID,WORD wLine,WORD wMid,WORD wMode);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
wMode	動作モード(詳細はユーザーズマニュアル<運用編>) 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 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:+EL または +SL 位置まで動作 28h:-EL または -SL 位置まで動作 22h:-EL または -SL 抜け出し動作 2Ah:+EL または +SL 抜け出し動作 24h:(+)方向に EZ カウント分だけ動作 2Ch:(-)方向に EZ カウント分だけ動作 41h:位置決め動作(目標相対位置指定) 44h:指令位置(CTR1)0 点復帰動作 45h:機械位置(CTR2)0 点復帰動作 46h:(+)方向 1 パルス動作 4Eh:(-)方向 1 パルス動作 47h:タイマ動作 01h:パルサ(PA/PB)入力による連続動作 51h:パルサ(PA/PB)入力による位置決め動作 54h:パルサ(PA/PB)入力による指令位置 0 点復帰動作 55h:パルサ(PA/PB)入力による機械位置 0 点復帰動作
備考	注意:機械座標位置指定位置決め時はフィードバック制御ではないので位置決め終了時の機械位置が

目標位置と異なる場合があります。
------------------

**(39) hpx\_WritSta() STA入力時の動作設定**

機能	指定したモジュールの STA 入力時の動作を設定します。
言語	書式
VC++	DWORD hpx_WritSta( DWORD hDevID, WORD wLine, WORD wMid, WORD wEnbl, WORD wTrg);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号1/1:ライン番号2)
wMid	モジュールID(0-63)
wEnbl	STA 入力有効設定(0:無効 / 1:有効)
wTrg	STA 入力仕様(0:レベル / 1:エッジ)
備考	

**(40) hpx\_WritStp() STP入力時の動作設定とSTP自動出力の設定**

機能	指定したモジュールの STP 入力時の動作設定と異常停止時 STP 自動出力の設定をします。
言語	書式
VC++	DWORD hpx_WritStp( DWORD hDevID, WORD wLine, WORD wMid, WORD wEnbl, WORD wMot, WORD wOut);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号1/1:ライン番号2)
wMid	モジュールID(0-63)
wEnbl	STP 入力有効設定(0:無効 / 1:有効)
wMot	STP 入力時の停止方法(0:即停止/1:減速停止)
wOut	異常停止時 STP 出力設定(0:出力/1:出力しない)
備考	

**(41) hpx\_WritFHSpd() 動作速度の設定**

機能	指定したモジュールの動作速度を設定します。
言語	書式
VC++	DWORD hpx_WritFHSpd(DWORD hDevID, WORD wLine, WORD wMid, DWORD dwRfh);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号1/1:ライン番号2)
wMid	モジュールID(0-63)
dwRfh	動作速度レジスタ(RFH)値(1-100000)
備考	



**(42) hpx\_WritPos() 移動量の設定**

機能	指定したモジュールの移動量を設定します。
言語	書式
VC++	DWORD hpx_WritPos(DWORD hDevID, WORD wLine, WORD wMid, long IRmv);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
IRmv	移動量[パルス](-134217728-134217727)

**(43) hpx\_WritCtr() カウンタへの書き込み**

機能	指定したモジュールの指定されたカウンタへ書き込みます。
言語	書式
VC++	DWORD hpx_WritCtr(DWORD hDevID,WORD wLine, WORD wMid, long IValue, WORD wCtr);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
IValue	書き込むカウンタ値
wCtr	書き込むカウンタ指定(1:CTR1 / 2:CTR2 / 3:CTR3)

**4.3.5 動作制御指令****(44) hpx\_DecStop() 減速停止**

機能	指定したモジュールを減速停止させます。
言語	書式
VC++	DWORD hpx_DecStop(DWORD hDevID, WORD wLine, WORD wMid );
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)

**(45) hpx\_QuickStop() 即停止**

機能	指定したモジュールを即停止させます。
言語	書式
VC++	DWORD hpx_QuickStop(DWORD hDevID, WORD wLine, WORD wMid );
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)

**(46) hpx\_EmgStop() 非常停止**

機能	指定したモジュールを非常停止させます。
言語	書式
VC++	DWORD hpx_EmgStop(DWORD hDevID, WORD wLine, WORD wMid);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
備考	

**(47) hpx\_AccStart() 加速スタート**

機能	指定したモジュールを加速スタートさせます。
言語	書式
VC++	DWORD hpx_AccStart ( DWORD hDevID, WORD wLine, WORD wMid );
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
備考	

**(48) hpx\_CnstStartFH() FH定速スタート**

機能	指定したモジュールを FH 定速スタートさせます。
言語	書式
VC++	DWORD hpx_CnstStartFH (DWORD hDevID, WORD wLine, WORD wMid);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
備考	

**(49) hpx\_CnstStartFL() FL定速スタート**

機能	指定したモジュールを FL 定速スタートさせます。
言語	書式
VC++	DWORD hpx_CnstStartFL(DWORD hDevID, WORD wLine, WORD wMid);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
備考	

**(50) hpx\_MvAccStart() 移動量設定+加速スタート**

機能	指定したモジュールの移動量の設定と加速スタートをします。
言語	書式
VC++	DWORD hpx_MvAccStart(DWORD hDevID, WORD wLine, WORD wMid, long IRmv);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
IRmv	移動量[パルス](-134217728-134217727)
備考	

**(51) hpx\_MvCnstStartFH() 移動量設定+FH定速スタート**

機能	指定したモジュールの移動量の設定とFH定速スタートをします。
言語	書式
VC++	DWORD hpx_MvCnstStartFH( DWORD hDevID, WORD wLine, WORD wMid, long IRmv);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
IRmv	移動量[パルス](-134217728-134217727)
備考	

**(52) hpx\_MvCnstStartFL() 移動量設定+FL定速スタート**

機能	指定したモジュールの移動量の設定とFL定速スタートをします。
言語	書式
VC++	DWORD hpx_MvCnstStartFL( DWORD hDevID, WORD wLine, WORD wMid, long IRmv);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
IRmv	移動量[パルス](-134217728-134217727)
備考	

**(53) hpx\_SetGroup() グループ設定**

機能	指定したモジュールを指定したグループに設定します。
言語	書式
VC++	DWORD hpx_SetGroup(DWORD hDevID, WORD wLine, WORD wMid, WORD wGrp);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wMid	モジュール ID(0-63)
wGrp	グループ番号(0-7) 但し0を指定した場合はグループ解除

**(54) hpx\_GrpStop() グループ停止**

機能	指定したグループのモジュールを停止させます。
言語	書式
VC++	DWORD hpx_GrpStop(DWORD hDevID, WORD wLine, WORD wGrp);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wGrp	グループ番号(0-7), 但し"0"は全グループ
備考	注意:この関数を使用する場合は指定グループのモジュールに対し、「STP 入力時に停止する」設定を行う必要があります。この設定をしていない場合は停止しません。

**(55) hpx\_GrpAccStart() グループ加速スタート**

機能	指定したグループのモジュールを加速スタートさせます。
言語	書式
VC++	DWORD hpx_GrpAccStart (DWORD hDevID, WORD wLine, WORD wGrp, );
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wGrp	グループ番号(0-7), 但し"0"は全グループ
備考	注意:この関数を使用する場合は指定グループのモジュールに対し、「STA 入力によるスタート」の設定を行う必要があります。この設定をしない場合は同時スタートではなく順々にスタートします。

**(56) hpx\_GrpCnstStartFH() グループFH定速スタート**

機能	指定したグループのモジュールを FH 定速スタートさせます。
言語	書式
VC++	DWORD hpx_GrpCnstStartFH (DWORD hDevID, WORD wLine, WORD wGrp );
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定(0:ライン番号 1/1:ライン番号 2)
wGrp	グループ番号(0-7), 但し"0"は全グループ
備考	注意:この関数を使用する場合は指定グループのモジュールに対し、「STA 入力によるスタート」の設定を

行う必要があります。この設定をしない場合は同時スタートではなく順々にスタートします。
--

**(57) hpx\_GrpCnstStartFL() グループFL定速スタート**

機能	指定したグループのモジュールを FL 定速スタートさせます。
言語	書式
VC++	DWORD hpx_GrpCnstStartFL (DWORD hDevID, WORD wLine, WORD wGrp );
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wGrp	グループ番号(0-7), 但し"0"は全グループ
備考	注意:この関数を使用する場合は指定グループのモジュールに対し、「STA 入力によるスタート」の設定を行う必要があります。この設定をしない場合は同時スタートではなく順々にスタートします。

**(58) hpx\_SvOn() サーボオン**

機能	指定したモジュールの SVON を ON します。
言語	書式
VC++	DWORD hpx_SvOn (DWORD hDevID, WORD wLine, WORD wMid );
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)
備考	

**(59) hpx\_SvOff() サーボオフ**

機能	指定したモジュールの SVON を OFF します。
言語	書式
VC++	DWORD hpx_SvOff (DWORD hDevID, WORD wLine, WORD wMid );
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)
備考	

**(60) hpx\_SvResetOn() サーボリセットオン**

機能	指定したモジュールの SVRST を ON します。
言語	書式
VC++	DWORD hpx_SvResetOn (DWORD hDevID, WORD wLine, WORD wMid );
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)
備考	



**(61) hpx\_SvResetOff() サーボリセットオフ**

機 能	指定したモジュールの SVRST を OFF します。
言語	書 式
VC++	DWORD hpx_SvResetOff (DWORD hDevID, WORD wLine, WORD wMid );
引 数	説 明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)
備 考	

**(62) hpx\_SvTIOOn() サーボトルク制限オン(オプション)**

機 能	指定したモジュールの SVTL を ON します。
言語	書 式
VC++	DWORD hpx_SvTIOOn (DWORD hDevID, WORD wLine, WORD wMid );
引 数	説 明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)
備 考	

**(63) hpx\_SvTIOOff() サーボトルク制限オフ(オプション)**

機 能	指定したモジュールの SVTL を OFF します。
言語	書 式
VC++	DWORD hpx_SvTIOOff (DWORD hDevID, WORD wLine, WORD wMid );
引 数	説 明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)
備 考	

**(64) hpx\_SvGainOn() サーボゲイン切り替えオン(オプション)**

機 能	指定したモジュールの SVGAIN を ON します。
言語	書 式
VC++	DWORD hpx_SvGainOn (DWORD hDevID, WORD wLine, WORD wMid );
引 数	説 明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)
備 考	

**(65) hpx\_SvGainOff() サーボゲイン切り替えオフ(オプション)**

機 能	指定したモジュールの SVGAIN を OFF します。
言語	書 式
VC++	DWORD hpx_SvGainOff (DWORD hDevID, WORD wLine, WORD wMid);
引 数	説 明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)
備 考	

**(66) hpx\_PMON パルスモータ励磁オン**

機 能	指定したモジュールのパルスモータ励磁をオンします。
言語	書 式
VC++	DWORD hpx_PMON (DWORD hDevID, WORD wLine, WORD wMid);
引 数	説 明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)
備 考	SVON をドライバの励磁 ON/OFF 端子に接続している場合有効

**(67) hpx\_PMOFF パルスモータ励磁オフ**

機 能	指定したモジュールのパルスモータ励磁をオフ(モーターフリー)します。
言語	書 式
VC++	DWORD hpx_PMOFF (DWORD hDevID, WORD wLine, WORD wMid);
引 数	説 明
hDevID	デバイスハンドル
wLine	ライン番号指定( 0:ライン番号 1/1:ライン番号 2 )
wMid	モジュール ID(0-63)
備 考	SVON をドライバの励磁 ON/OFF 端子に接続している場合有効

## 4.3.6 計算関数

## (68) hpx\_CalAccRate 加減速レート計算

機能	加減速時間,RFH,RFL 等より加減速レートを計算します.
言語	書式
VC++	DWORD hpx_CalAccRate( DWORD* dwRur, DWORD dwTim, DWORD dwRfh, DWORD dwRfl, WORD wAcc, DWORD dwRus);
引数	説明
dwRur	計算結果格納先
dwTim	加(減)速時間(ミリ秒)
dwRfh	RFH レジスタ値(1-100000)
dwRfl	RFL レジスタ値(1-100000)
wAcc	加減速方式(0:直線/1:S字)
dwRus	S字区間(1-50000)
備考	<p>(1) <math>dwRfh \leq dwRfl</math> の時はエラーを返します.</p> <p>(2) <math>(dwRfh-dwRfl)/2 &lt; dwRus</math> の時はエラーを返します.</p> <p>&lt;加(減)速時間の計算式&gt;</p> <p>RXR:加速時は RUR / 減速時は RDR</p> <p>RXS:加速時は RUS / 減速時は RDS</p> <p>(1) 直線加速</p> <p><math>T(\text{加速時間[msec]}) = (RFH-RFL) * (RXR+1) / 5000</math></p> <p><math>RXR = 5000 * T / (RFH-RFL) - 1</math></p> <p>(2) S字加速</p> <p><math>T(\text{加速時間[msec]}) = (RFH-RFL) * (RXR+1) / 2500</math></p> <p><math>RXR = 2500 * T / (RFH-RFL) - 1</math></p> <p>(3) S字加速(直線区間有)</p> <p><math>T(\text{加速時間[msec]}) = (RFH-RFL+2 * RXS) * (RXR+1) / 5000</math></p> <p><math>RxR = 5000 * T / (RFH-RFL+2 * RXS) - 1</math></p>

**(69) hpx\_CalDecPoint 減速開始点計算**

機能	減速開始点手動設定時の減速開始点最適値を計算します。 位置決め動作時に、この値より大きな値を減速開始点レジスタに設定するとベース速度で動作する時間が長くなり、逆に小さな値を減速開始点レジスタに設定するとベース速度に到達する前に停止します。
----	--

言語	書式
VC++	<pre>DWORD hpx_CalDecPoint( DWORD* dwRdp, DWORD dwRfh, DWORD dwRfl, DWORD dwRmg, DWORD dwRdr, WORD wAcc, DWORD dwRds);</pre>

引数	説明
dwRdp	計算結果格納先
dwRfh	RFH レジスタ値
dwRfl	RFL レジスタ値
dwRmg	RMG レジスタ値
dwRdr	RDR レジスタ値
wAcc	加減速方式(0:直線/1:S字)
dwRds	RDS レジスタ値

備考	
----	--

## 5. ドライバ関数

## 5.1 ドライバ関数

ドライバ関数はアプリケーションとデバイスドライバをつなぐ入出力関数「デバイスドライバ I/F 用ライブラリ」であり、Win32API 関数として DLL ファイルで提供されています。関数の戻り値は「[3.5 関数の戻り値](#)」を参照してください。

## 5.2 ドライバ関数一覧

### 5.2.1 デバイス関係

No.	関 数 名	機 能
1	mnt520_GetMstBrdCount( )	MCAT枚数の取得
2	mnt520_GetMstBrdInfo( )	MCATデバイス情報の取得
3	mnt520_OpenMstBrd( )	MCATのオープン
4	mnt520_CloseMstBrd( )	MCATのクローズ

### 5.2.2 オプションポート関係

No.	関 数 名	機 能
5	mnt520_rOptPortW( )	オプションポート2バイト読出し
6	mnt520_wOptPortW( )	オプションポート2バイト書込み

### 5.2.3 センターデバイス関係

No.	関 数 名	機 能
7	mnt520_rCenMsts( )	センターメインステータス読出し
8	mnt520_wCenCmd( )	センターデバイスコマンド書込み
9	mnt520_rCenIsts( )	センター割込ステータス読出し
10	mnt520_rCenBuf( )	センターデバイス入出力バッファ読出し
11	mnt520_wCenBuf( )	センターデバイス入出力バッファ書込み
12	mnt520_rCenRFiFo( )	センターデバイス受信用FIFO読出し
13	mnt520_wCenSFiFo( )	センターデバイス送信用FIFO書込み
14	mnt520_rLclCycErr( )	サイクリック通信エラーフラグ読出し
15	mnt520_wLclCycErr( )	サイクリック通信エラーフラグリセット
16	mnt520_rLclInfo( )	ローカルデバイス(DIO, モーション)情報読出し
17	mnt520_wLclInfo( )	ローカルデバイス(DIO, モーション)情報書込み
18	mnt520_rLclSetInt( )	ローカルデバイス入カポート変化フラグ設定状態読出し
19	mnt520_wLclSetInt( )	ローカルデバイス入カポート変化フラグ設定書込み
20	mnt520_rLclInt( )	ローカルデバイス入カポート変化フラグ読出し
21	mnt520_wLclInt( )	ローカルデバイス入カポート変化フラグリセット
22	gu00_rCenPortW	センターデバイス指定アドレス 2 バイト読出し
23	gu00_wCenPortW	センターデバイス指定アドレス 2 バイト書込み

### 5.2.4 DIOモジュール(一部アナログモジュールにも使用)関係

No.	関 数 名	機 能
24	mnt520_rIoPortB()	DIOデバイス指定ポート1バイト読出し
25	mnt520_wIoPortB()	DIOデバイス指定ポート1バイト書込み
26	mnt520_rIoPortW()	DIOデバイス指定ポート2バイト読出し
27	mnt520_wIoPortW()	DIOデバイス指定ポート2バイト書込み

### 5.2.5 モーションモジュール関係

No.	関 数 名	機 能
28	mnt520_rPclPort()	モーションデバイス汎用出力ポート出力状態読出し
29	mnt520_wPclPort()	モーションデバイス汎用出力ポート出力設定書込み
30	mnt520_rPclMsts()	モーションメインステータス読出し
31	mnt520_wPclCmd()	モーションデバイス制御コマンド書込
32	mnt520_rPclReg()	モーションデバイスレジスタ読出し
33	mnt520_wPclReg()	モーションデバイスレジスタ書込

### 5.2.6 アナログモジュール関係

No.	関 数 名	機 能
34	mnt520_rAmodAin()	アナログモジュールアナログ入力データ読出し
35	mnt520_wAmodAout()	アナログモジュールアナログ出力データ書込み
36	mnt520_rAmodAout()	アナログモジュールアナログ出力データ読出し
37	mnt520_wAmodCmd()	アナログモジュール制御コマンド書込み
38	mnt520_wAmodReg()	アナログモジュールレジスタ設定データ書込み
39	mnt520_rAmodReg()	アナログモジュールレジスタ設定データ読出し
40	mnt520_rAmodStat()	アナログモジュールステータス読出し

## 5.3 ドライバ関数詳細

ここでは VC++表記で説明します。

VB, VB.NET を使用される場合, データ型の対応は下表の通りです。

引数のデータ型, 16 進数表記の対応

言 語		VC++	VB	VB.NET	VC#
データ型	8bit	BYTE	Byte	Byte	byte
	16bit	WORD, short	Integer	Short	ushort, short
	32bit	DWORD, long	Long	Integer	ulong, long
データ 例		0x0000	&H0	&H0	0x0000
		0x1000	&H1000	&H1000	0x1000

## 5.3.1 デバイス関係

## (1) mnt520\_GetMstBrdCount PCに接続されているデバイス数の取得

機能	PC に接続されている MCAT の数を取得します。
言語	書式
VC++	DWORD mnt520_GetMstBrdCount (DWORD* dwCnt);
引数	説明
dwCnt	デバイスの個数

## (2) mnt520\_GetMstBrdInfo デバイス情報の取得

機能	PC に接続されている MCAT のデバイス情報を取得します。
言語	書式
VC++	DWORD mnt520_GetMstBrdInfo (DWORD* dwCnt, MCAT00INF* hInfo);
引数	説明
dwCnt	デバイスの個数
hInfo	デバイス情報

備考	<pre>// デバイス情報構造体 typedef struct __HMNT520INFO {     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     DWORD    dwBoardID;         // ボード ID(0~15) } HMNT520INFO, /PHMNT520INFO;</pre>
----	--

## (3) mnt520\_OpenMstBrd デバイスのオープン

機能	デバイス情報で指定した MCAT をオープンし、デバイスハンドルを取得します。 ここで取得したデバイスハンドルにより、MCAT を指定します。
言語	書式
VC++	DWORD mnt520_OpenMstBrd (DWORD* hDevID, MCAT00INF* hInfo);
引数	説明
hDevID	デバイスハンドル
hInfo	デバイス情報

**(4) mnt520\_CloseMstBrd デバイスのクローズ**

機 能	デバイスハンドルで指定した MCAT をクローズします。
言語	書 式
VC++	DWORD mnt520_CloseMstBrd (DWORD hDevID);
引 数	説 明
hDevID	デバイスハンドル
備 考	デバイスクローズ以前にアプリケーションの終了処理を行ってください。

**5.3.2 オプションポート関係****(5) mnt520\_rOptPortW オプションポート読出し**

機 能	指定した MCAT の指定したアドレスのオプションポートからデータを読出します。
言語	書 式
VC++	DWORD mnt520_rOptPortW( DWORD hDevID, WORD wCmd, WORD* wData);
引 数	説 明
hDevID	デバイスハンドル
wCmd	オプションポート読出しコマンド(0202h:通信速度設定状態)
wData	オプションポートデータ
備 考	<通信速度設定状態> bit3,2 00:20Mbps / 01:10Mbps / 10:5Mbps / 11:2.5Mbps

**(6) mnt520\_wOptPortW オプションポート書込み**

機 能	指定した MCAT のオプションポートヘータを書込みます。
言語	書 式
VC++	DWORD mnt520_wOptPortW( DWORD hDevID, WORD wCmd, WORD wData );
引 数	説 明
hDevID	デバイスハンドル
wCmd	オプションポート書込みコマンド
wData	オプションポートデータ
備 考	

## 5.3.3 センターデバイス関係

## (7) mnt520\_rCenMsts センターメインステータス読出し

機能	デバイスハンドルで指定された MCAT のセンターメインステータスを読み出します。
----	---

言語	書式
VC++	DWORD mnt520_rCenMsts(DWORD hDevID, WORD wReserved, WORD* wCmsts);

引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wCmsts	センターメインステータス

<センター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
BBSY	DBSY	RBSY	SBSY	0	RDBB	TDBB	REF	0	CAER	ERAE	EDTE	EIOE	IOPC	BRKF	CEND

bit	名称	説明
0	CEND	データ送信用 FIFO 書き込み可能時に 1 になります。システム通信、またはデータ通信が完了して、データ送信用 FIFO に次データの書き込みが可能になった時に 1 になります。本ビットのクリア方法は RENV0.bit9 の状態によります。
1	BRKF	ブレイクフレーム受信時に 1 になります。本ビットのクリア方法は、RENV0.bit9 の状態によります。
2	IOPC	「入力変化割り込み設定」を 1 にセットした入力ポートの状態が変化した時に 1 になります。「入力ポート変化フラグ」の全 256 ビットの OR 信号です。全ビットが 0 になると、このビットは 0 に戻ります。
3	EIOE	サイクリック通信エラー発生時に 1 になります。「サイクリック通信エラーフラグ」の全 64 ビットの OR 信号です。
4	EDTE	データ通信エラー発生時に 1 になります。本ビットのクリア方法は、RENV0.bit9 の状態によります。
5	ERAE	ローカルデバイス側受信処理エラー発生時に 1 になります。本ビットのクリア方法は、RENV0.bit9 の状態によります。
6	CAER	アプリケーションのアクセスエラーです。送信データが空のままデータ送信コマンドを書き込む等、アプリケーションから不適切なアクセスがあると 1 になります。本ビットのクリア方法は、RENV0.bit9 の状態によります。
8	REF	未送信の出力ポートデータがある時 1 になります。出力ポートエリアにデータを書き込むと 1 になり、全ポートへのサイクリック通信を 2 回以上エラー無しで行った後に 0 に戻ります。
9	TDBB	データ送信用 FIFO に送信データがある時に 1 になります。データ送信用 FIFO に書き込むと 1 になり、データ送信コマンド、または送信用 FIFO リセットコマンドを書き込んだ時に 0 に戻ります。
10	RDBB	データ受信用 FIFO に受信データがある時に 1 になります。モーションモジュール等のデータデバイスからデータを受信すると 1 になり、アプリケーションが受信データを全て読み出すと 0 に戻ります。
12	SBSY	サイクリック通信スタート中に 1 になります。
13	RBSY	リセット処理中に 1 になります。
14	DBSY	システム通信中、またはデータ通信中に 1 になります。

## (8) mnt520\_wCenCmd センターデバイスコマンド書込み

機能	デバイスハンドルで指定された MCAT ヘンターデバイスコマンドを書込みます。
----	---

言語	書式
VC++	DWORD mnt520_wCenCmd(DWORD hDevID, WORD wLine, WORD wCcmd);

引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wCcmd	センターデバイスコマンド

No.	センターデバイスコマンド	コマンド(hex)	備考
1	無効コマンド	0000	
2	ソフトウェアリセット	0100	
3	送信用 FIFO リセット	0200	
4	受信用 FIFO リセット	0300	
5	センター割込ステータスのクリア	04##	##は割込みの種類
6	エラーカウンタクリア	0600	
7	全モジュールへのシステム通信	1000	
8	サイクリック通信除外中の全モジュールへのシステム通信	1100	
9	指定したモジュールへのシステム通信	12##	##はモジュール ID
10	指定したモジュールの属性情報の取得	13##	##はモジュール ID
11	指定したグループへのコマンド送信	2#\$\$	#はグループ指定, \$\$はコマンド
12	サイクリック通信の開始	3000	
13	サイクリック通信の停止	3100	
14	データ通信	40##	##はモジュール ID
15	RENV0 書込みコマンド	5500	
16	RENV0 読出しコマンド	6500	
17	エラーカウンタリードコマンド	6501	
18	サイクリック周期レジスタリードコマンド	6502	
19	受信アドレスレジスタリードコマンド	6503	

## (9) mnt520\_rCenIsts センター割込ステータス読出し

機能	デバイスハンドルで指定された MCAT のセンター割込ステータスを読み出します。
言語	書式
VC++	DWORD mnt520_rCenIsts(DWORD hDevID, WORD wLine, WORD* wCists);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wCists	センター割込ステータス

<センター割込ステータスの内容>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAE3	CAE2	CAE1	CAE0	ERA3	ERA2	ERA1	ERA0	LNRV	0	EDN5	EDN4	EDN3	EDN2	EDN1	EDN0

bit	名称	説明
5-0	EDN5-0	CMSTS の, EDTE=1 または ERAE=1 のエラー発生時のモジュール ID で, 次回エラー発生まで記憶されます。
7	LNRV	モジュール側データ未受信時に 1 になります。 データ通信がエラーで終了した(EDTE=1)場合, モジュール側がセンターデバイスからのデータを受信出来なかった時は 1 になり, 受信出来た時は 0 になります。次回エラー発生まで記憶されます。
11-8	ERA3-0	モジュールが正常受信したにもかかわらず, そのデータ内容がモジュールの種類と不整合の場合, 以下のようなコードを本ビット部分に記憶します。このコードは次回エラー発生まで記憶されます。 0001: ローカルデバイス情報エリアの I/O の設定情報と, モジュール側の I/O の組合せが異なっている場合 0010: DIO またはアナログモジュールがデータ通信を受信した時 0011: モジュールが自己の持っている受信バッファ容量以上のデータ通信を受信した時
15-12	CAE3-0	センターデバイスに対して不正なアクセスを行った場合, 以下のようなコードを本ビット部分に記憶します。このコードは次回エラー発生まで記憶されます。 0001: 使用モジュールなしでサイクリック通信スタートコマンドを書き込んだ時 0010: 送信データを送信用 FIFO に設定しないでデータ送信スタートコマンド書き込んだ時 0011: CMSTS.DBSY=1 の時に, システム通信, またはデータ通信スタートコマンドを書込んだ時 0100: 未使用モジュール扱いとなっているモジュールに対して, データ通信を行った時

**(10) mnt520\_rCenBuf センターデバイス入出力バッファ読み出し**

機能	デバイスハンドルで指定された MCAT の入出力バッファからデータを読み出します。								
言語	書式								
VC++	DWORD mnt520_rCenBuf(DWORD hDevID, WORD wLine, WORD* wCbuf);								
引数	説明								
hDevID	デバイスハンドル								
wLine	ライン番号(0:ライン 1, 1:ライン 2)								
wCbuf	センターデバイス入出力バッファデータ								
備考	<p>センターデバイスのレジスタからのデータ読み出し手順は</p> <p>(1) センターデバイスコマンドを書込む</p> <p>(2) 入出力バッファを読み出す</p> <p>となります。</p> <p>センターデバイスコマンドは以下の通りです。</p> <table border="0"> <tr> <td>RENV0 読み出しコマンド</td> <td>6500h</td> </tr> <tr> <td>エラーカウンタリードコマンド</td> <td>6501h</td> </tr> <tr> <td>サイクリック周期レジスタリードコマンド</td> <td>6502h</td> </tr> <tr> <td>受信アドレスレジスタリードコマンド</td> <td>6503h</td> </tr> </table>	RENV0 読み出しコマンド	6500h	エラーカウンタリードコマンド	6501h	サイクリック周期レジスタリードコマンド	6502h	受信アドレスレジスタリードコマンド	6503h
RENV0 読み出しコマンド	6500h								
エラーカウンタリードコマンド	6501h								
サイクリック周期レジスタリードコマンド	6502h								
受信アドレスレジスタリードコマンド	6503h								

**(11) mnt520\_wCenBuf センターデバイス入出力バッファ書き込み**

機能	デバイスハンドルで指定された MCAT の入出力バッファへデータを書込みます。
言語	書式
VC++	DWORD mnt520_wCenBuf(DWORD hDevID, WORD wLine, WORD wCbuf);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wCbuf	センターデバイス入出力バッファデータ
備考	<p>センターデバイスのレジスタへのデータ書き込み手順は</p> <p>(1) 入出力バッファにデータを書込む</p> <p>(2) センターデバイスコマンドを書込む</p> <p>となります。</p> <p>センターデバイスコマンドは以下の通りです。</p> <p>RENV0 書き込みコマンド 5500h</p>

**(12) mnt520\_rCenRFifo センターデバイス受信用FIFO(1ワード)読み出し**

機能	デバイスハンドルで指定された MCAT の受信用 FIFO から 1 ワードのデータを読み出します。
言語	書式
VC++	DWORD mnt520_rCenRFifo(DWORD hDevID, WORD wLine, WORD* wRfifo);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wRfifo	受信用 FIFO から読み出されたデータ

**(13) mnt520\_wCenSFifo センターデバイス送信用FIFO(1ワード)書き込み**

機能	デバイスハンドルで指定された MCAT の送信用 FIFO へ 1 ワードのデータを書込みます。
言語	書式
VC++	DWORD mnt520_wCenSFifo(DWORD hDevID, WORD wLine, WORD wSfifo);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wSfifo	送信用 FIFO へ書込むデータ

**(14) mnt520\_rLclInfo ローカルデバイス情報読出し**

機能	指定されたモジュールのローカルデバイス情報を取得します。
言語	書式
VC++	DWORD mnt520_rLclInfo(DWORD hDevID, WORD wLine, WORD wMid, BYTE* byData);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMid	モジュール ID(0-63)
byData	ローカルデバイス情報

**(15) mnt520\_wLclInfo ローカルデバイス情報書き込み**

機能	指定されたモジュールのローカルデバイス情報を書込みます。
言語	書式
VC++	DWORD mnt520_wLclInfo(DWORD hDevID, WORD wLine, WORD wMid, BYTE byData);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMid	モジュール ID(0-63)
byData	ローカルデバイス情報

**(16) mnt520\_rLclCycErr サイクリック通信エラーフラグ読出し**

機能	指定されたモジュールのサイクリック通信エラーフラグを読み出します。
言語	書式
VC++	DWORD mnt520_rLclCycErr(DWORD hDevID, WORD wLine, WORD wMidPrm, WORD* wFlag);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMidPrm	モジュール ID 指定パラメータ(0:MID 0~15 / 1:MID 16~31 / 2:MID 32~47 / 3:MID 49~64)
wFlag	サイクリック通信エラーフラグ

## (17) mnt520\_wLclCycErr サイクリック通信エラーフラグリセット

機能	指定されたモジュールのサイクリック通信エラーフラグリセットします。
言語	書式
VC++	DWORD mnt520_wLclCycErr(DWORD hDevID, WORD wLine, WORD wMidPrm, WORD wFlag);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMidPrm	モジュール ID 指定パラメータ(0:MID 0~15 / 1:MID 16~31 / 2:MID 32~47 / 3:MID 49~64)
wFlag	サイクリック通信エラーフラグ
備考	読み出されたエラーフラグのデータを書込むことでエラーフラグリセットします。

## (18) mnt520\_rLclSetInt ローカルデバイス入力ポート変化フラグ設定状態読出し

機能	指定したモジュールのローカルデバイス入力ポート変化フラグ設定状態を読み出します。																																																
言語	書式																																																
VC++	DWORD mnt520_rLclSetInt(DWORD hDevID, WORD wLine, WORD wMidPrm, WORD* wFlag);																																																
引数	説明																																																
hDevID	デバイスハンドル																																																
wLine	ライン番号(0:ライン 1, 1:ライン 2)																																																
wMidPrm	モジュール ID 指定パラメータ ( 0:MID= 0- 3 / 1:MID= 4- 7 / 2:MID= 8-11 / 3:MID=12-15 / 4:MID=16-19 / 5:MID=20-23 / 6:MID=24-27 / 7:MID=28-31 / 8:MID=32-35 / 9:MID=36-39 / 10:MID=40-43 / 11:MID=44-47 / 12:MID=48-51 / 13:MID=52-55 / 14:MID=56-59 / 15:MID=60-63)																																																
wFlag	入力ポート変化フラグ設定状態																																																
備考	<p>&lt;入力ポート変化フラグ設定データ&gt;</p> <table border="1"> <tr> <td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td> </tr> <tr> <td colspan="4">MID= wMidPrm*4+3</td> <td colspan="4">MID= wMidPrm*4+2</td> <td colspan="4">MID= wMidPrm*4+1</td> <td colspan="4">MID= wMidPrm*4</td> </tr> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> </table> <p>ポート3 → 3, 2, 1, 0          ポート2 → 7, 6, 5, 4          ポート1 → 11, 10, 9, 8          ポート0 → 15, 14, 13, 12</p>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	MID= wMidPrm*4+3				MID= wMidPrm*4+2				MID= wMidPrm*4+1				MID= wMidPrm*4																			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																		
MID= wMidPrm*4+3				MID= wMidPrm*4+2				MID= wMidPrm*4+1				MID= wMidPrm*4																																					

## (19) mnt520\_wLclSetInt ローカルデバイス入力ポート変化フラグ設定書込み

機能	指定したモジュールのローカルデバイス入力ポート変化フラグ設定をします。
言語	書式
VC++	DWORD mnt520_wLclSetInt(DWORD hDevID, WORD wLine, WORD wMidPrm, WORD wFlag);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMidPrm	モジュール ID 指定パラメータ ( 0:MID= 0- 3 / 1:MID= 4- 7 / 2:MID= 8-11 / 3:MID=12-15 / 4:MID=16-19 / 5:MID=20-23 / 6:MID=24-27 / 7:MID=28-31 / 8:MID=32-35 / 9:MID=36-39 / 10:MID=40-43 / 11:MID=44-47 / 12:MID=48-51 / 13:MID=52-55 / 14:MID=56-59 / 15:MID=60-63)
wFlag	入力ポート変化フラグ設定データ

## (20) mnt520\_rLclInt ローカルデバイス入力ポート変化フラグ読出し

機能	指定したモジュールのローカルデバイス入力ポート変化フラグを読み出します。
言語	書式
VC++	DWORD mnt520_rLclInt(DWORD hDevID, WORD wLine, WORD wMidPrm, WORD* wFlag);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMidPrm	モジュール ID 指定パラメータ ( 0:MID= 0- 3 / 1:MID= 4- 7 / 2:MID= 8-11 / 3:MID=12-15 / 4:MID=16-19 / 5:MID=20-23 / 6:MID=24-27 / 7:MID=28-31 / 8:MID=32-35 / 9:MID=36-39 / 10:MID=40-43 / 11:MID=44-47 / 12:MID=48-51 / 13:MID=52-55 / 14:MID=56-59 / 15:MID=60-63)
wFlag	ローカルデバイス入力ポート変化フラグ。

## (21) mnt520\_wLclInt ローカルデバイス入力ポート変化フラグリセット

機能	指定したモジュールのローカルデバイス入力ポート変化フラグをリセットします。
言語	書式
VC++	DWORD mnt520_wLclInt(DWORD hDevID, WORD wLine, WORD wMidPrm, WORD wFlag);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMidPrm	モジュール ID 指定パラメータ ( 0:MID= 0- 3 / 1:MID= 4- 7 / 2:MID= 8-11 / 3:MID=12-15 / 4:MID=16-19 / 5:MID=20-23 / 6:MID=24-27 / 7:MID=28-31 / 8:MID=32-35 / 9:MID=36-39 / 10:MID=40-43 / 11:MID=44-47 / 12:MID=48-51 / 13:MID=52-55 / 14:MID=56-59 / 15:MID=60-63)
wFlag	ローカルデバイス入力ポート変化フラグ。

**(22) mnt520\_rCenPortW センターデバイス指定アドレス 2 バイト読出し**

機能	デバイスハンドルで指定された MCAT の指定されたアドレスのポートから 2 バイトのデータを読出します。
言語	書式
VC++	DWORD mnt520_rCenPortW( DWORD hDevID, WORD wAdrs, WORD* wData);
引数	説明
hDevID	デバイスハンドル
wAdrs	MCAT 内アドレス(但し奇数アドレスの指定不可)
wData	読出されたデータ

**(23) mnt520\_wCenPortW センターデバイス指定アドレス 2 バイト書込み**

機能	デバイスハンドルで指定された MCAT の指定されたアドレスのポートへ 2 バイトのデータを書込みます。
言語	書式
VC++	DWORD mnt520_wCenPortW(DWORD hDevID, WORD wAdrs, WORD wData);
引数	説明
hDevID	デバイスハンドル
wAdrs	MCAT 内アドレス(但し奇数アドレスの指定不可)
wData	書込みデータ

**5.3.4 DIOモジュール(一部アナログモジュールにも使用)関係****(24) mnt520\_rloPortB DIOモジュール指定ポート 1 バイト読出し**

機能	指定した DIO モジュールの指定したポートから 1 バイト読出します。
言語	書式
VC++	DWORD mnt520_rloPortB(DWORD hDevID, WORD wLine,WORD wMid, WORD wPort, BYTE* byData);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMid	モジュール ID
wPort	DIO モジュールポート
byData	読出されたデータ

**(25) mnt520\_wloPortB DIOモジュール指定ポート 1 バイト書込み**

機能	指定した DIO モジュールの指定したポートへ 1 バイト書込みます。
言語	書式
VC++	DWORD mnt520_wloPortB(DWORD hDevID, WORD wLine,WORD wMid, WORD wPort, BYTE byData);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMid	モジュール ID
wPort	DIO モジュールポート
byData	書込むデータ

**(26) mnt520\_rloPortW DIOモジュール指定ポート 2 バイト読出し**

機 能	指定した DIO モジュールの指定したポートから 2 バイト読出します。
言語	書 式
VC++	DWORD mnt520_rloPortW(DWORD hDevID, WORD wLine, WORD wMid, WORD wPort, WORD* wData);
引 数	説 明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMid	モジュール ID(0-63)
wPort	DIO モジュールポート(但し奇数ポートの指定不可)
wData	読出されたデータ

**(27) mnt520\_wloPortW DIOモジュール指定ポート 2 バイト書込み**

機 能	指定した DIO モジュールの指定したポートへ 2 バイト書込みます。
言語	書 式
VC++	DWORD mnt520_wloPortW(DWORD hDevID, WORD wLine, WORD wMid, WORD wPort, WORD wData);
引 数	説 明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMid	モジュール ID(0-63)
wPort	DIO モジュールポート(但し奇数ポートの指定不可)
wData	書込むデータ

**5.3.5 モーションモジュール関係****(28) mnt520\_rPclPort モーションモジュール汎用出力ポート読出し**

機 能	指定したモーションモジュールの汎用出力ポートを読出します。
言語	書 式
VC++	DWORD mnt520_rPclPort(DWORD hDevID, WORD wLine, WORD wMid, BYTE* byData);
引 数	説 明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMid	モジュール ID(0-63)
byData	汎用出力ポートデータ

**(29) mnt520\_wPciPort モーションモジュール汎出力ポート書き込み**

機能	指定したモーションモジュールの汎出力ポートヘータを書込みます。
言語	書式
VC++	DWORD mnt520_wPciPort(DWORD hDevID, WORD wLine, WORD wMid, BYTE byData);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMid	モジュール ID(0-63)
byData	汎出力ポートデータ

**(30) mnt520\_rPciMsts モーションメインステータス読出し**

機能	指定したモーションモジュールのモーションメインステータスを読出します。
言語	書式
VC++	DWORD mnt520_rPciMsts(DWORD hDevID, WORD wLine, WORD wMid, WORD* wMmsts);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMid	モジュール ID(0-63)
wMmsts	モーションメインステータス

**(31) mnt520\_wPciCmd モーションモジュール制御コマンド書き込み**

機能	指定したモーションモジュールへ制御コマンドを書込みます。
言語	書式
VC++	DWORD mnt520_wPciCmd(DWORD hDevID, WORD wLine, WORD wMid, WORD wMcmd);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMid	モジュール ID(0-63)
wMcmd	制御コマンド

**(32) mnt520\_rPciReg モーションモジュールレジスタ読出し**

機能	指定したモーションモジュールの指定したレジスタからデータを読出します。
言語	書式
VC++	DWORD mnt520_rPciReg(DWORD hDevID, WORD wLine, WORD wMid, WORD wCmd, DWORD* dwReg);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMid	モジュール ID(0-63)
wCmd	レジスタ読出しコマンド
dwReg	レジスタデータ

**(33) mnt520\_wPciReg モーションモジュールレジスタ書込み**

機能	指定したモーションモジュールの指定したレジスタヘータを書込みます。
言語	書式
VC++	DWORD mnt520_wPciReg(DWORD hDevID, WORD wLine, WORD wMid, WORD wCmd, DWORD dwReg);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMid	モジュール ID(0-63)
wCmd	レジスタ書込みコマンド
dwReg	レジスタデータ

**5.3.6 アナログモジュール関係****(34) mnt520\_rAmodAin アナログモジュールアナログ入力データ読み出し**

機能	指定したアナログモジュールの指定したチャンネルからアナログ入力データを読み出します。
言語	書式
VC++	DWORD mnt520_rAmodAin( DWORD hDevID, WORD wLine, WORD wMid, WORD wCh, WORD* wDat, WORD* wRes);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMid	モジュール ID(0-63)
wCh	チャンネル番号(0:CH1 / 1:CH2 / 2:CH3 / 3:CH4)
wDat	アナログ入力データ
wRes	ステータス情報

**(35) mnt520\_wAmodAout Aモジュールアナログ出力データ書込み**

機能	指定したアナログモジュールの指定したチャンネルへ出力データを書き込みます。
言語	書式
VC++	DWORD mnt520_wAmodAout(DWORD hDevID, WORD wLine, WORD wMid, WORD wCh, WORD wDat, WORD* wSts);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMid	モジュール ID(0-63)
wCh	チャンネル番号(0:CH1 / 1:CH2 / 2:CH3 / 3:CH4)
wDat	アナログ出力データ
wSts	ステータス情報

**(36) mnt520\_rAmodAout Aモジュールアナログ出力データ読み出し**

機能	指定したアナログモジュールの指定したチャンネルからアナログ出力データを読み出します。
言語	書式
VC++	DWORD mnt520_rAmodAout(DWORD hDevID, WORD wLine, WORD wMid, WORD wCh, WORD* wDat, WORD* wSts);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMid	モジュール ID(0-63)
wCh	チャンネル番号(0:CH1 / 1:CH2 / 2:CH3 / 3:CH4)
wDat	アナログ出力データ
wSts	ステータス情報

**(37) mnt520\_wAmodCmd Aモジュール制御コマンド書き込み**

機能	指定したアナログモジュールへ制御コマンドを書き込みます。
言語	書式
VC++	DWORD mnt520_wAmodCmd(DWORD hDevID, WORD wLine, WORD wMid, WORD wCmd, WORD wDat, WORD* wSts);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMid	モジュール ID(0-63)
wCmd	制御コマンド番号 (0:プリセット手動切り替え / 1:コンパレータラッチクリア, NOP / 2:D/A 0V 出力, 連続切替実行中止)
wDat	書き込みデータ
wSts	ステータス情報

**(38) mnt520\_wAmodAout Aモジュールアナログ出力データ書き込み**

機能	指定したアナログモジュールの指定したチャンネルへ出力データを書き込みます。
言語	書式
VC++	DWORD mnt520_wAmodAout( DWORD hDevID, WORD wLine, WORD wMid, WORD wCh, WORD wDat, WORD* wSts);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMid	モジュール ID(0-63)
wCh	チャンネル番号(0:CH1 / 1:CH2 / 2:CH3 / 3:CH4)
wDat	アナログ出力データ
wSts	ステータス情報

## (39) mnt520\_rAmodReg アナログモジュールレジスタ読出し

機能	指定したアナログモジュールのレジスタからデータを読出します。
言語	書式
VC++	DWORD mnt520_rAmodReg(DWORD hDevID, WORD wLine, WORD wMid, WORD wCmd, WORD* wReg, WORD* wSts);
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMid	モジュール ID(0-63)
wCmd	レジスタ読出しコマンド
wReg	レジスタデータ
wSts	読出しステータス情報
備考	<p>&lt;アナログモジュールレジスタ読出しコマンド&gt;</p> <p>3000h:システム動作条件設定,  3100h:全コンパレータ集約動作設定,  320xh:各チャンネルコンパレータ動作設定(x=0:設定 1,1:設定 2)  33xyh:プリセットデータイベント切り替え設定(x=チャンネル番号 0-3,y=0:コンパレータ,1:DI)  34xxh, 35xxh, 36xxh:未定義  37xxh:予約(使用不可)  38xyh:プリセットデータ D/A 値登録 (x=チャンネル番号 0-3,y=プリセットデータ番号 1~15)  39xyh:A/D コンパレータ基準値登録(x=チャンネル番号 0-3,y=0:LOW 設定,1:High 設定)  3Axyh:プリセットデータ到達時間(x=チャンネル番号 0~3,y=プリセットデータ番号 1~15)  3Bxyh:プリセットデータ保持時間(x=チャンネル番号 0~3,y=プリセットデータ番号 1~15)  3Cx0h:プリセットデータ連続出力条件設定(x=チャンネル番号 0~3)  3Dx0h:D/A オフセット値登録(x=チャンネル番号 0~3)  3Ex0h:A/D オフセット値登録(x=チャンネル番号 0~3)  3Fxxh:未定義</p>

## (40) mnt520\_wAmodReg アナログモジュールレジスタ書込み

機能	指定したアナログモジュールのレジスタヘータを書込みます。
言語	書式
VC++	DWORD mnt520_wAmodReg( DWORD hDevID, WORD wLine, WORD wMid, WORD wCmd, WORD wReg, WORD* wSts );
引数	説明
hDevID	デバイスハンドル
wLine	ライン番号(0:ライン 1, 1:ライン 2)
wMid	モジュール ID(0-63)
wCmd	レジスタ読出しコマンド
wReg	レジスタデータ
wSts	書込みステータス情報
備考	<p>&lt;アナログモジュールレジスタ書込みコマンド&gt;</p> <p>1000h:システム動作条件設定,  1100h:全コンパレータ集約動作設定,  120xh:各チャンネルコンパレータ動作設定(x=0:設定 1,1:設定 2)  13x0h:プリセットデータイベント切り替え設定(x=チャンネル番号 0~3, y=0:コンパレータ,1:DI)  14xxh, 15xxh, 16xxh:未定義  18xyh:プリセットデータ D/A 値登録(x=チャンネル番号 0~3,y=プリセットデータ番号 1~15)  19xyh:A/D コンパレータ基準値登録(x=チャンネル番号 0~3,y=0:LOW 設定,1:High 設定)  1Axyh:プリセットデータ到達時間設定(x=チャンネル番号 0~3,y=プリセットデータ番号 1~15)  1Bxyh:プリセットデータ保持時間設定(x=チャンネル番号 0~3,y=プリセットデータ番号 1~15)  1Cx0h:D/A 連続切り替え実行条件設定(x=チャンネル番号 0~3)  1Dx0h:D/A オフセット値登録(x=チャンネル番号 0~3)  1Ex0h:A/D オフセット値登録(x=チャンネル番号 0~3)  1Fxxh:未定義</p>

## 6. ポート資料

## 6.1 ポート表

HPCI-MCAT520MとHCPCI-MNT720Mのアドレスマップは同じです。ポートはすべてメモリマップです。

BAR	区分	Address (Hex)	読み込み(INP)		書き込み(OUT)		
			呼称	内 容	呼称	内 容	
BAR2	#1 ライン1	0000	CMSTS	センターメインステータス(15-0)	CCMD	コマンド(15-0)	
		0002	CISTS	センター割込ステータス(15-0)	-	Reserved	
		0004	CBUF	入出力バッファIN(15-0)	CBUF	入出力バッファ OUT(15-0)	
		0006	RFIFO	データ受信FIFO(15-0)	SFIFO	データ送信FIFO(15-0)	
		0008 ~ 0077	-	Reserved	-	Reserved	
		0078	DINFO	ローカルデバイス情報(78-B7)	DINFO	ローカルデバイス情報(78-B7)	
		00B8	IOERR	サイクリック通信エラーフラグ(B8-BF)	IOERR	サイクリック通信エラーフラグ(B8-BF)	
		00C0	IINT	入力ポート変化フラグ設定状態(C0-DF)	IINT	入力ポート変化フラグ設定(C0-DF)	
		00E0	IRES	入力ポート変化フラグ(E0-FF)	IRES	入力ポート変化フラグリセット(E0-FF)	
		0100	DATA	ポートデータ IN(100-1FF)	DATA	ポートデータ OUT(100-1FF)	
		#2 ライン2	0000	CMSTS	センターメインステータス(15-0)	CCMD	コマンド(15-0)
	0002		CISTS	センター割込ステータス(15-0)	-	Reserved	
	0004		CBUF	入出力バッファIN(15-0)	CBUF	入出力バッファ OUT(15-0)	
	0006		RFIFO	データ受信FIFO(15-0)	SFIFO	データ送信FIFO(15-0)	
	0008 ~ 0077		-	Reserved	-	Reserved	
	0078		DINFO	ローカルデバイス情報(78-B7)	DINFO	ローカルデバイス情報(78-B7)	
	00B8		IOERR	サイクリック通信エラーフラグ(B8-BF)	IOERR	サイクリック通信エラーフラグ(B8-BF)	
	00C0		IINT	入力ポート変化フラグ設定状態(C0-DF)	IINT	入力ポート変化フラグ設定(C0-DF)	
	00E0		IRES	入力ポート変化フラグ(E0-FF)	IRES	入力ポート変化フラグリセット(E0-FF)	
	0100		DATA	ポートデータ IN(100-1FF)	DATA	ポートデータ OUT(100-1FF)	
	BAR3		オプション ポート	0000	DIN	マスターボード汎用入力状態 (J1: MDR14)(注)	-
		0002		DOUT	マスターボード汎用出力状態(注)	DOUT	マスターボード汎用出力設定 (J1: MDR14)(注)
0004		STS		G9001Aステータス	-	Reserved	
0006		SPD		通信速度設定状態	-	Reserved	
0008		INTE		PCI割込許可状態	INTE	PCI割込許可	
000A		INTS		PCI Bus割込ステータス	-	Reserved	
0010		BID		ボードID設定状態	-	Reserved	
0012		BCOD		Board Code 'MNT520_ _'	-	Reserved	

BAR:ベースアドレス

注: HPCI-MCAT520MにはDIN, DOUTはありません。

表 6.1-1 ボードアドレス

## 6.2 オプションポート

オプションポートはマスターボード全体に 1 組おかれています。

オプションポートからデータの読み出しは

```
mnt520_rOptPortW() ... 2 バイトデータ読み出し
```

データの書き込みは

```
mnt520_wOptPortW() ... 2 バイトデータ書き込み
```

を使用します。

### 6.2.1 マスターボード汎用入力ポート(HCPCI-MNT720Mのみ)

BAR3+00H: DIN (Read only)

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	0	0	0	0	IN4	IN3	IN2	IN1

図 6.2-1 マスターボード汎用入力ポートのビット構成

bit	記号	機能	備考
3-0	IN4-1	汎用入力状態 ON で"1"	
15-4	(未定義)	常に"0"	

表 6.2-1 マスターボード汎用入力ポートの内容

### 6.2.2 マスターボード汎用出力設定ポート(HCPCI-MNT720Mのみ)

BAR3+02H: DOUT (Read/Write)

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	0	0	0	OUT4	OUT3	OUT2	OUT1

図 6.2-2 マスターボード汎用出力設定及び出力状態確認ポートのビット構成

bit	記号	機能	備考
3-0	OUT4-1	汎用出力設定"1"で ON, 汎用出力状態 ON で"1"	
15-4	(未定義)		書き込みした値が読めます

表 6.2-2 マスターボード汎用出力設定及び出力状態確認ポートの内容

### 6.2.3 G9001Aステータス

BAR3+04H: STS (Read only)

15-12	11	10	9	8	7~4	3	2	1	0
0	2MSYN	2MERF	2MERR	2MCRY	0	1MSYN	1MERF	1MERR	1MCRY

図 6.2-3 G9001A ステータス確認ポートのビット構成

bit	記号	機能	備考
8,0	xMCRY	1: 信号ライン信号検出で一定時間	
9,1	xMERR	1: 異常フレーム受信時と、無応答時に一定時間	
10,2	xMERF	1: エラー応答フレーム受信時一定時間 L レベル	
11,3	xMSYN	サイクリック周期毎に反転	

表 6.2-3 G9001A ステータス確認の内容

## 6.2.4 G9001A通信速度設定状態

BAR3+06H: SPD(Read Only)

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	0	0	0	SPD0	SPD1	DSW2	DSW1

図 6.2-4 G9001A 通信速度設定状態のビット構成

bit	記号	機 能
0	DSW1	通信設定 SW bit0 ON で "1"
1	DSW2	通信設定 SW bit1 ON で "1"
3,2	SPD1,0	通信設定 bit 読み出し ON で"1" 00:20Mbps,10:10Mbps,01:5Mbps,11:2.5Mbps

表 6.2-4 G9001A 通信速度設定確認ポートの内容

## 6.2.5 PCI割込許可

BAR3+08H: INTE(Read/Write)

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	0	0	0	0	0	0	INTE

図 6.2-5 PCI 割込許可設定ポートのビット構成

bit	記号	機 能	備 考
0	INTE	PCIBus への割込を許可します。“1”で許可，“0”で禁止。 (割込要因は G9001A に依ります)	Windows 版ソフトウェアではサポートしていません。
15-1	(未定義)		書き込みした値が読めます

表 6.2-5 PCI 割込許可設定ポートの内容

## 6.2.6 PCI割込ステータスポート

BAR3+0AH: INTS (Read only)

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	0	0	0	0	0	INTS2	INTS1

図 6.2-6 PCI 割込ステータスポートのビット構成

bit	記号	機 能	備 考
0	INTS1	“1”で Line1 から PCIBus への割込が発生していることを示します。	
1	INTS2	“1”で Line2 から PCIBus への割込が発生していることを示します。	
15-2	(未定義)	常に“0”	

表 6.2-6 PCI 割込ステータスポートの内容

## 6.2.7 ボードID確認ポート

BAR3+10H: BID (Read only)

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	0	0	0	BID4	BID3	BID2	BID1

図 6.2-7 ボード ID 確認ポートのビット構成

bit	記号	機 能
3-0	BID4-1	基板上のボード ID 設定ロータリ SW の状態が読み出せます。
15-4	(未定義)	常に“0”

表 6.2-7 モジュール ID 読出(ボード ID)の内容

## 6.2.8 ボードコード確認ポート

BAR3+12H..17:BCOD (Read only)

アドレス	データ	内容	備考
+12H	4E4D H	N,M	
+14H	5254 H	52,T	
+16H	0000 H	_,0	

表 6.2-8 ボードコード確認ポートの内容

## 6.3 PCI コンフィグレーションレジスタ

HPCI-MCAT520MとHPCI-MNT720MのPCIコンフィグレーションレジスタの内容は同じです。

PCIコンフィグレーションレジスタの内容を次表に示します。

31	24	23	16	15	8	7	0	アドレス
デバイスID 1034h				ベンダID 14a9h				00h
デバイスステータス				デバイス制御				04h
クラスコード							リビジョンID (00h)	08h
基本クラス (06h)	サブクラス (80h)	プログラム インターフェース						
セルフテスト	ヘッダタイプ	マスタレイテンシタイム		キャッシュライン			0ch	
ベ ー ス ア ド レ ス レ ジ ス タ	00000000h(予 約)							10h
	PCI9030(PCIコンフィグレーションレジスタ)							14h
	<b>G9001ベースアドレス(BAR2)</b>							18h
	<b>オプションポートベースアドレス(BAR3)</b>							1ch
	00000000h(予 約)							20h
	00000000h(予 約)							24h
カードバスCISポインタ								28h
サブシステムID 1034h				サブシステムベンダID 14a9h				2ch
予 約								30h
								7 fch

表 6.3-1 PCI コンフィグレーションレジスタ

ベースアドレス	ターゲット	空間	Bus 幅	備考
BAR2	#1G9001A	512 Byte	16bit	
BAR2+200h	#2G9001A	512 Byte	16bit	
BAR3	オプションポート	16 Byte	16bit	

BAR: ベースアドレス

表 6.3-2 PCI アドレス空間