HPCI-DIO5144

添付ソフトウェアマニュアル

DOS版デバイスドライバ
DOS版ドライバＩ／Ｆ用ライブラリ

株式会社ハイバーテック
http://www.hivertec.co.jp/

DOS 版ドライバは
　hdio5144.drv です.

DOS 版ドライバＩ／Ｆライブラリーとして
　メモリモデル別に次の４種類があります。
　　αdio5144.lib
　　αはメモリモデルを表す英１文字であり、次のようになります。
　　α＝L・・ラージモデル・・・・・ldio5144.lib
　　α＝C・・コンパクトモデル・・・cdio5144.lib
　　α＝M・・ミディアムモデル・・・mdio5144.lib
　　α＝S・・スモールモデル・・・・sdio5144.lib

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6F
ＴＥＬ　03-3846-3801
ＦＡＸ　03-3846-3773
sales@hivertec.co.jp

1.0版　2010年 6月25日発行

目　次

1. はじめに

この度は，弊社 DIO シリーズボードをご採用頂きまして，誠に有り難う御座います.
本書は，HPCI-DIO5144 ボードの添付ソフトウェアに関して解説を行うものです.

この添付ソフトウェアは
　PC-DOS または MS-DOS（以降 **DOS** と記します），
において，**DIO5144** ボードの制御を行う為に使用します.

ボードの制御に関する説明については，個々の関数内で関連項目を取り上げていますが，詳細な説明が必要な場合には，ボード添付のユーザーズマニュアルを参照してください.
尚，本書は，アプリケーション開発環境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.

1.1 安全にお使い頂くために

安全上の注意

本製品のご使用前に，必ず本書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.

1.2 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．

1.3 免責事項

1. 本マニュアルに記載された内容に沿わない，製品の取り付け，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. 本マニュアル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．

1.4 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．

⚠

・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．
・OS の操作およびソフトウェア開発環境に付いて基本的な知識を有している方．

1.5 適合 OS

注　意

本製品は，PC-DOS または MS-DOS 上においてボードの制御を行う為のソフトウェアです.
上記以外の OS でのご使用については，弊社営業までお問合せ下さい.

1.6 ハードウェアの設定･取付け･接続

警　告

ボードの取付け，配線に際しては，ユーザーズマニュアルを良くお読みいただき，ユーザーズマニュアルの内容にしたがって実施願います.

1.7 サンプルプログラム作成ツール

注　意

本添付ソフト中のサンプルプログラムは，以下の開発ツールで作成しています.
・Microsoft C Compiler 6.0

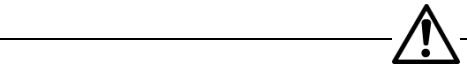
1.8 サンプルプログラムの実行

警　告

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

1.9 ユーザープログラムの作成

警　告



ユーザープログラムの作成にあたっては，装置の特性を考慮し，必要なインターロック・安全対策処理等を十分盛り込んだ設計として下さい.
プログラムコード・データのわずかな違いにより予想外の動作をして，機器や人体に損傷を与える恐れが有ります.
プログラム作成・試運転時共，十分な注意をお願いいたします.

2. 添付ソフトウェアの構成

2.1 ソフトウェア構成

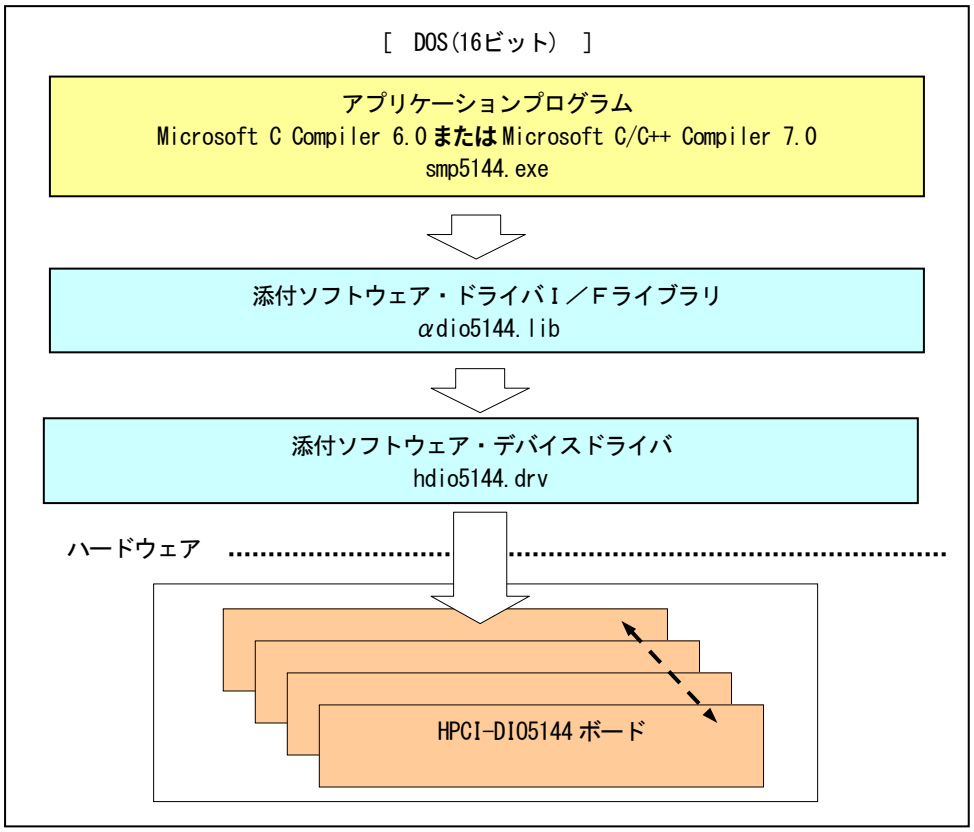


図 2.1-1 DOS(16 ビット) 添付ソフトウェア関連図

2.2 添付ソフトウェアの内容

添付 FD または CD 内の Readme.txt をご参照下さい。

3. デバイスドライバのインストール

3.1 インストール

新規にデバイスドライバを登録する為には、パソコン・ハードディスク内の所定のディレクトリにドライバファイル" HDI05144.DRV "をコピーし, ＤＯＳ起動ドライブにある" CONFIG.SYS "内に次の行を追加します.

```
DEVICE = HDI05144.DRV
```

（注）指定はデバイスドライバファイルを格納した「絶対パス名」を記述します.

[例] { C:¥HDRV }ディレクトリにコピーした場合

DEVICE = C:¥HDRV¥HDI05144.DRV　とします.

" CONFIG.SYS "ファイルへの追加が完了した後, マシンを再起動します.

3.2 アンインストール

インストールで説明した" CONFIG.SYS "内のデバイスドライバの登録行を削除します.
また デバイスドライバファイル本体を削除します.

4. ボードを複数枚利用する場合

DIO5144 ボードをパソコンに複数枚装着し，それぞれのボードを個別に識別する方法を説明します.

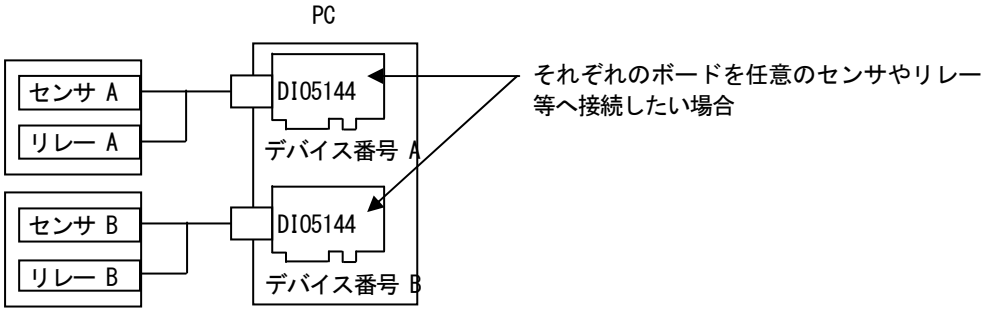


図 4.1-1 ボードを複数枚使用する場合

4.1 ボードのデバイス番号の確認

PCI バススロットにボードを装着すると，パソコンにより，それぞれのボードに固定のデバイス番号が割り振られます．その番号を知ることにより，それぞれのボードを個別に識別できます.

4.2 デバイス番号の確認方法

次の実行ファイルを起動することにより，現在 PCI バススロットに装着されている DIO5144 ボードのデバイス番号を確認できます.

「 ¥smp¥smp5144.exe」

このプログラムは，起動直後に DIO5144 ボードのデバイス情報の取得を行なっています.
これにより DIO5144 ボードのデバイス番号の取得を行ない，その情報を画面上部に表示しています.
詳細は，「7. 関数詳細 (3) デバイス情報の取得」をご覧ください.

4.3 DIO5144 ボードでのボード ID の使用

DIO5144 ボードはボード上のロータリーディップ SW で設定したボード ID（0～15 まで任意に設定可）が使用できます.

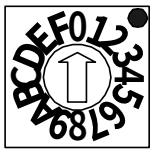


図 4.3-1 HPCI-DIO5144 ボード ID ロータリーディップ SW

5. ドライバ I/F ライブラリの使用方法

5.1 概　要

ドライバ I/F ライブラリは，DOS および DOS アプリケーション上において，**HPCI-DIO5144** ボードの制御を行うための関数群です.

各関数（ライブラリ関数）は DOS アプリケーション上で外部関数として静的にリンクされ，**HPCI-DIO5144** のデバイスドライバにアクセスします.

本ボードで使用するデバイスドライバ（hdio5144.drv）は Intel 互換ＣＰＵを搭載したマシン上で，かつ 16 ビット DOS 上でのみ使用できます.

5.2 関数一覧

ドライバ I/F ライブラリには，次の 8 種類 10 関数が含まれます.

No	関　数　名　称	機　　能
1	hdio5144_GetDeviceCount()	ボード枚数の取得
2	hdio5144_GetDevVerNo()	バージョン番号の取得
3	hdio5144_GetDeviceInfo()	デバイス情報の取得
4	hdio5144_OpenDevice()	デバイスのオープン
5	hdio5144_CloseDevice()	デバイスのクローズ
6	hdio5144_rPortB()	ポートバイト読込
	hdio5144_rPortW()	ポートワード読込
7	hdio5144_wPortB()	ポートバイト書込
	hdio5144_wPortW()	ポートワード書込
8	hdio5144_SetIntCall()	割り込み処理関数の登録

表 5.2-1 関数一覧

5.3 準　備

次の記述をソースプログラムに追加し，プログラムと同一フォルダーにこのファイルを配置します.

```
#include    "hidio580.h"
```

次のライブラリーファイルをリンクファイルとします．リンクするファイルは適当なフォルダーへ配置します.
また，使用するファイルはアプリケーションプログラムのメモリモデルと同一とします.

- **s**dio5144.lib　（スモールモデル）　[ｺｰﾄﾞ：64KB 未満､ﾃﾞｰﾀ：64KB 未満]
- **c**dio5144.lib　（コンパクトモデル）　[ｺｰﾄﾞ：64KB 未満､ﾃﾞｰﾀ：64KB 以上]
- **m**dio5144.lib　（ミディアムモデル）　[ｺｰﾄﾞ：64KB 以上､ﾃﾞｰﾀ：64KB 未満]
- **l**dio5144.lib　（ラージモデル）　[ｺｰﾄﾞ：64KB 以上､ﾃﾞｰﾀ：64KB 以上]

※ 注意事項

①デバイスドライバおよびライブラリ・ソフトウエアは，Intel 互換ＣＰＵを搭載したマシン以外のプラットフォームには対応しておりません.

②ライブラリではWindows 版とのソースプログラム互換性を考慮して，関数名やコーリングシーケンス，機能が同じ仕様になるよう構成されておりますが，一部異なる部分もありますのでご留意ください.

6. 制御概念

ライブラリ関数及びデバイスドライバは複数の **HPCI-DIO5144** ボードを制御することができます.
HPCI-DIO5144 ボードにアクセスするために，デバイスをオープンしてアクセスするためのデバイスハンドルを取得します.
デバイスをオープンするためにボードを認識するための情報（ハードウェアリソース）を取得します．この情報をデバイス情報と呼びます.
（ハードウェアリソースすなわち I/O ポートアドレスや IRQ 番号等は，システム側によって確定されます．）
デバイスハンドルにより，ライブラリ関数及びデバイスドライバを使用してで複数の HPCI-DIO5144 ボードを制御することができます.

6.1 ボード (デバイス) 認識用のデータ構造体

ボード認識のために次に示す HPCDEVINFO 型構造体を用意します．この構造体は実装されるボード枚数分の配列が必要です.

```
typedef  struct  {
    WORD  nBusNumber;          /* バス番号     (0-255) */
    WORD  nDevNumber;          /* デバイス番号(0- 31) (PCI ｽﾛｯﾄ番号) */
    WORD  dwIoPortAdrs;        /* I/O ポートアドレス */
    WORD  dwIrqNo;             /* IRQ 番号  (1-15)  */
    WORD  dwNumber;            /* 管理番号  (1-32)  */
    WORD  dwBoardID;           /* ボードＩＤ(0-15)  */
} HPCDEVINFO, *PHPCDEVINFO
```

6.2 ボードアクセスの準備手順

(1) 使用する全ボードのデバイス情報の取得
"**HPCDEVIINFO**"型構造体エリア（の配列）内に，全 **HPCI-DIO5144** のデバイス情報をまず取得します.
◆ hdio5144_GetDeviceCount()・・・ボード枚数の確認
◆ hdio5144_GetDeviceInfo() ・・・全ボードのデバイス情報を取得

(2) ボード毎にデバイスオープン
ある 1 つの **HPCI-DIO5144** 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 **HPCI-DIO5144** 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ためのデバイスハンドルを返します.

ボード枚数が 2 枚以上の場合には，個々のボード毎にこの処理を行います.
◆ hdio5144_OpenDevice() ・・・ボードのオープン処理

(3) 各入出力ポートの初期化
ボードへの電源投入・遮断時には出力ポートへの出力は"0"を書込んだ状態となっていますが，上記設定以降に，使用する全ボードの入出力ポートの初期化を行います.

①出力ポートの初期設定
◆ hdio5144_wPortB() または hdio5144_wPortW()

②プログラム開始時の入力ポート取込み
外部入力信号の"変化"で何らかの処理を行いたい時，基準となる外部信号状態を取込みます.
◆ hdio5144_rPortB() または hdio5144_rPortW()

(4) 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，オープンしたデバイスの「クローズ処理」を行って下さい．複数枚オープンしている場合は，オープンしたデバイス個々に対してクローズを行う必要があります.
◆ hdio5144_CloseDevice() ・・・ボードのクローズ処理（1 枚分）

6.3 関数の戻り値

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

Ｎｏ	戻り値		異常内容と確認項目
	記号表記	１６進数表記	
１	NO_ERROR	0x0000	正常 異常は発生していません
２	NOT_FOUND	0x0001	対象デバイスが見つからない ◎デバイスドライバなし ◇ドライバのインストールを確認します. ◇［＞mem /d /p ］でﾃﾞﾊﾞｲｽﾄﾞﾗｲﾊﾞ有無が確認できます. ◇ﾄﾞﾗｲﾊﾞは $DIO580$ として表示されます. ◎指定デバイス(ボード)なし デバイス情報と一致するデバイスがありません. ◇デバイス(ボード)枚数の確認を行います. ◇デバイス情報先頭アドレスを確認して下さい.
３	ALREADY_OPENED	0x0002	既にオープン済のデバイスをオープン ◎同一のデバイスがオープン済 ◇オープンしたデバイスはクローズするまで使用 （多重オープンは禁止） ◇デバイス情報先頭アドレスを確認して下さい. ◎ボード２枚以上使用する場合,オープンするデバイス情報の更新を確認します.
４	NOT_MEMORY	0x0004	デバイス情報格納メモリが不足 ◎デバイス管理用メモリが確保できません. （ドライバは最大１６枚のデバイスを管理します．）
５	INVALID_HANDLE	0x0008	無効なデバイスハンドルを指定 ◎デバイスＩＤが無効 ◇正常オープン結果の"デバイスＩＤ"を使用して下さい.
６	NOT_READY	0x0010	デバイスの入出力ポートが使用できない ◎システムが不安定になっている可能性がありますので, ◇弊社サポートまでお問い合わせください
７	ILLEGAL_DEVICE	0x0020	ボード固有情報が不正 ◎ポートの読み出しができない状態です. ◇弊社サポートまでお問い合わせください
８	ILLEGAL_PARAM	0x0100	関数の引数の値が異常 ◎関数の引数の値が設定不可 ◇引数の設定値を確認（マニュアル照合）
９	OTHER_BOARD	0x0200	上記以外の定義されない不明なエラー ◎想定外のエラーを認識しました. ◇弊社サポートまでお問い合わせください

7. 関数詳細

（1） hdio5144_GetDeviceCount()　　ＤＩＯ５１４４ボード枚数の取得

《機能》
現在パソコンに装着されているＤＩＯ５１４４ボードの枚数を取得します.

《書式》
```
short  hdio5144_GetDeviceCount( short count );
```

《引数》
◆ short* count・・装着枚数の格納アドレス

《戻り値》　処理結果
０：　成功　・・　count に検出したボード枚数が格納.
その他：失敗　・・　関数実行が失敗. 「関数の戻り値」を参照.

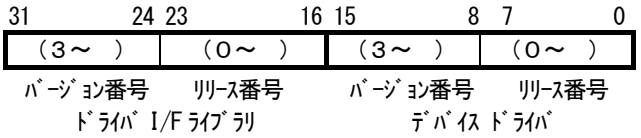
《呼び出し例》
```
/* Get Device Count */
  short  count;          /* ボード枚数の格納場所 */
  short  ans;            /* 関数の戻り値 */

  ans  = hdio5144_GetDeviceCount(&count);
  if(!ans) {
      printf("Board count = %d", count);   /* 取得したボード枚数の表示 */
  }
  else {
      printf("Error Code = %d", ans);      /* 発生したエラーコードの表示 */
  }
```

（２）　hdio5144_GetDevVerNo()　　バージョン番号の取得

《機能》
実装されているデバイスドライバーならびにドライバ I/F ライブラリーのバージョン番号を取得します.

《書式》
```
short  hdio5144_GetDevVerNo( DWORD* verno );
```

《引数》
◆ DWORD*　verno・・バージョン番号が格納されます.



《戻り値》　処理結果
０：　成功　・・　verno にバージョン番号が格納.
その他：失敗　・・　関数実行が失敗. 「関数の戻り値」を参照.

《呼び出し例》
```
/* Get Version No  */
  union {              /* バージョン番号格納場所 */
      DWORD  l;      /* Read Data Area */
      BYTE   c[4];   /* locate : drv rel[0],ver[1], lib rel[2],ver[3] */
  }  verno;
  short  ans;          /* 関数の戻り値 */

  ans  = hdio5144_GetDevVerNo(&verno.l);
  if(!ans) {
      printf("Drv Ver = %d.%02d¥n", (verno.c[1], verno.c[0]);    /* デバイスドライバー Ver 表示 */
      printf("Lib Ver = %d.%02d", (verno.c[3], verno.c[2]);      /* ライブラリ関数 Ver 表示 */
  }
  else {
      printf("Error Code = %d", ans);
  }
```

（3）　hdio5144_GetDeviceInfo ()　　ＤＩＯ５１４４ボードのデバイス情報取得

《機能》

実装されているＤＩＯ５１４４ボードのデバイス情報を取得します.
デバイス情報はHPCDEVINFO 型で指定ボード枚数分の配列で格納されます.
この値はデバイスオープン時に利用します.

《書式》

```
short  hdio5144_GetDeviceInfo( short* pcnDevNo, PHPCDEVINFO pHpcDevinfo );
```

《引数》

◆ short* pcnDevNo

情報を取得するボードの最大枚数が指定されたshort 型エリアのアドレスを渡します.
関数の呼び出し後, 実際に情報を取得したボードの枚数が格納されます.

◆ PHPCDEVINFO pHpcDevInfo

各ボードのデバイス情報がセットされるべきエリアのアドレス, すなわち HPCDEVINFO 型の配列の先頭アドレスを渡します.

《戻り値》　処理結果

０：　成功　・・　pcnDevNo に情報を取得したボードの枚数, pHpcDevInfo に各ボードのデバイス情報が格納.
その他：失敗　・・　関数実行が失敗. 「関数の戻り値」を参照.

《呼び出し例》

パソコンに ＤＩＯ５１４４ が３枚装着されており, これらすべてのボードからデバイス情報を取得,
, その情報の中からボードIDを表示することを示した例です.

```
short       count = 3;              /* ボード枚数格納場所(MAX3 枚までを取得) */
short       ans;                    /* 関数の戻り値 */
HPCDEVINFO  HpcDevInfo[3];          /* DIO5144 のﾃﾞﾊﾞｲｽ情報が最大 3 枚分格納されるエリア */
                                      (取得するボード枚数と同じか, それよりより大きな配列が必要)
ans = hdio5144_GetDeviceInfo(
                    &count,             /* count のアドレスを渡す */
                    &HpcDevInfo[0] );   /* 配列の先頭アドレスを渡す */
if(!ans) {
    printf("Count = %d¥n", count);                          /* 情報を取得したボード枚数表示 */
    printf("Board-1 ID = %d¥n", HpcDevInfo[0].dwBoardID);   /* 1 枚目ボードのボード ID 表示 */
    printf("Board-2 ID = %d¥n", HpcDevInfo[1].dwBoardID);   /* ２枚目ボードのボード ID 表示 */
    printf("Board-3 ID = %d", HpcDevInfo[2].dwBoardID);     /* ３枚目ボードのボード ID 表示 */
}
else {
    printf("Error Code = %d", ans);
}
```

（４）　hdio5144_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ＤＩＯ５１４４ボードをオープンし，他と識別するためのデバイスＩＤを付与します.

以降このデバイスＩＤは，このＤＩＯ５１４４にアクセスするためのハンドルとなります.

《書式》

```
short  hdio5144_OpenDevice(DWORD*  hDevID, PHPCDEVINFO pHpcDevInfo );
```

《引数》

◆ DWORD* hDevID

デバイスＩＤが格納されます.

◆ PHPCDEVINFO pHpcDevInfo

オープンするデバイスの情報がセットされたエリアのアドレスを渡します.

《戻り値》　処理結果

０：　成功　・・　デバイス ID が持つたボードのデバイス情報が格納.

その他：失敗　・・　関数実行が失敗. 「関数の戻り値」を参照.

《呼び出し例》

パソコンにＤＩＯ５１４４ が３枚装着されており，デバイス情報格納エリア HpcDevInfo[MAX]には既にこのボード３枚分の情報が取得・格納されていると想定，全てのＤＩＯ５１４４をオープンする場合の例を示します.

```
short       ans;              /* 関数の戻り値 */
HPCDEVINFO  HpcDevInfo[3];    /* DIO5144 のﾃﾞﾊﾞｲｽ情報が最大 3 枚分格納されるエリア */
DWORD       hDevID[3];        /* デバイス ID 取得エリア */

hDevID[0] = hdio5144_OpenDevice( &HpcDevInfo[0] );    /* １枚目ｵｰﾌﾟﾝ */
hDevID[1] = hdio5144_OpenDevice( &HpcDevInfo[1] );    /* ２枚目ｵｰﾌﾟﾝ */
hDevID[2] = hdio5144_OpenDevice( &HpcDevInfo[2] );    /* ３枚目ｵｰﾌﾟﾝ */

if(!ans) {
    printf("Count = %d¥n", count );                    /* 情報を取得したボード枚数表示 */
    printf("Board-1 Device ID = %d¥n", hDevID[0] );    /* １枚目ボードのデバイス ID 表示 */
    printf("Board-2 Device ID = %d¥n", hDevID[1] );    /* ２枚目ボードのデバイス ID 表示 */
    printf("Board-3 Device ID = %d", hDevID[2] );      /* ３枚目ボードのデバイス ID 表示 */
}
else {
    printf("Error Code = %d", ans);
}
```

（５）　hdio5144_CloseDevice()　　デバイスのクローズ

《機能》
　デバイスＩＤで指定されたＤＩＯ５１４４ボードをクローズします.
　以降、このデバイスＩＤは無効となります.

《書式》
```
short  hdio5144_CloseDevice( DWORD hDevID );
```

《引数》
　◆ DWORD hDevID　・・　クローズするボードのデバイスＩＤ

《戻り値》　　処理結果
　０：　　成功　・・　デバイスが正常に解放.
　その他：失敗　・・　関数実行が失敗. 「関数の戻り値」を参照.

《呼び出し例》
```
short       ans;              /* 関数の戻り値 */
DWORD       hDevID;           /* 取得済みデバイス ID */

ans = hdio5144_CloseDevice( hDevID );
```

（６）

hdio5144_rPortB()	ポートバイト読込
hdio5144_rPortW()	ポートワード読込

《機能》

デバイスＩＤで指定されたＤＩＯ５１４４の指定されたポートから
ポートバイト読込・・・１バイトを読込み，指定したエリアに格納します.
ポートワード読込・・・２バイトを読込み，指定したエリアに格納します.

《書式》

```
short hdio5144_rPortB( DWORD hDevID, WORD PortNo, BYTE* byInp );
short hdio5144_rPortW( DWORD hDevID, WORD PortNo, WORD* wdInp );
```

《引数》

◆ DWORD hDevID・・・対象デバイスのデバイス.ID
◆ WORD portNo・・・ポート番号指定（ポート番号－１）
◆ BYTE* byInp ・・・読込んだデータを格納する１バイトエリアのアドレス
◆ WORD* wdInp ・・・読込んだデータを格納する２バイトエリアのアドレス

《戻り値》　処理結果

０：　成功 ・・ ポートから読み出したデータが格納
その他：失敗 ・・ 関数実行が失敗. 「関数の戻り値」を参照.

《呼び出し例》

```
short       ans;            /* 関数の戻り値 */
DWORD       hDevID;         /* 取得済みデバイス ID */
BYTE        byData;         /* バイトデータ格納先 */
WORD        wdData;         /* ワードデータ格納先 */

/* byte read  */
ans = hdio5144_rPortB( hDevID , 0x10, &bytData );   //出力ポート1の出力状態をバイト読み出し
if(!ans) {
    printf("Output Port-1 = 0x%02x", btData );
}
else {
    printf("Error Code = %d", ans);
}

/* word read  */
ans = hdio5144_rPortW( hDevID , 0x36, &wdData );   //ラッチ信号ステータスのワード読み出し
```

（7）

hdio5144_wPortB ()	ポートバイト書込
hdio5144_wPortW ()	ポートワード書込

《機能》

デバイスＩＤで指定されたＤＩＯ５１４４のポートへ
ポートバイト書込・・・指定１バイトを書込みます.
ポートワード書込・・・指定２バイトを書込みます.

《書式》

```
short hdio5144_wPortB(DWORD  hDevID, WORD PortNo, BYTE byOutp );
short hdio5144_wPortW(DWORD  hDevID, WORD PortNo, WORD wdOutp );
```

《引数》

◆ DWORD hDevID ・・・対象デバイスのデバイスＩＤ
◆ WORD PortNo ・・・ポート番号指定（ポート番号－１）
◆ BYTE byOutp ・・・書込む１バイトデータ
◆ WORD wdOutp ・・・書込む２バイトデータ

《戻り値》 処理結果

０： 成功 ・・ ポートへの書き込み正常終了
その他：失敗 ・・ 関数実行が失敗. 「関数の戻り値」を参照.

《呼び出し例》

```
short       ans;              /* 関数の戻り値 */
DWORD       hDevID;           /* 取得済みデバイス ID */

/* byte write  */
ans = hdio5144_wPortB( hDevID , 0x10, 0xff );   //出力ポート１に”0xff”を書き込み

/* word write  */
ans = hdio5144_wPortW( hDevID , 0x22, 0x0fff ); //フィルタ有効選択の全ビット有効(0xfff書き込み)
```

（８）　hdio5144_SetIntCall()　　割込処理関数の登録・登録削除

《機能》
デバイスＩＤで指定されたＤＩＯ５１４４ボードからの割込が発生した場合の処理関数を登録, または登録した割り込み処理を削除します.

《書式》
```
short hdio5144_SetIntCall( DWORD hDevID, PINTPROC fnIntCall );
```

《引数》
◆ DWORD　hDevID　・・　対象デバイスのデバイスＩＤ
◆ PINTPROC　fnIntCall　・・　割込が発生した時の割込処理モジュールのアドレスまたは"NULL"（削除）

《戻り値》　処理結果
０：　成功　・・　割込処理関数の登録または削除成功
その他：失敗　・・　関数実行が失敗. 「関数の戻り値」を参照.

《呼び出し例》
```
short       ans;              /* 関数の戻り値 */
DWORD       hDevID;           /* 取得済みデバイス ID */
PINTPROC    fIntCall;

/* 登録 */
fIntCall = (PINTPROC)int_module;
ans = hdio5144_SetIntCall(
                    hDevID,         /* デバイス ID */
                    fIntCall );     /* 割込処理モジュール */

/* 登録抹消 */
ans = hdio5144_SetIntCall(
                    hDevID,         /* デバイス ID */
                    NULL );         /* 関数登録削除 */
```

《ご注意》
割込を使用する場合には、次の点に留意して下さい。

(1) 初期化時の関数登録の順序
① 割込以外の全てのボード初期化を実行する。
② hdio5144_SetIntCall()　関数で割込処理モジュールを設定する。

(2) 割込処理モジュール
① ボード単位で処理が必要です。
② モジュール内でＣＰＵに対して”割込許可”をしないで下さい。
③ 作成方法はサンプルプログラムを参照して下さい。

(3) 終了時に忘れてはならないこと。
① hdio5144_SetIntCall()　関数で割込処理モジュールを削除する。

8. サンプルプログラム

ソフトウエアには「Ｃ言語」で作成されたサンプルプログラムが同梱されています。
このサンプルプログラムは、次の目的で使用して下さい。

①装着ボードの確認
デバイスドライバのインストールを行い、電源ＯＦＦ後ボードを装着し、パソコンの電源再投入後、サンプルプログラムの実行ファイル起動を行いますと、装着ボードの「デバイス情報」表示とボードの動作確認ができます。

②ドライバ I/F ライブラリの各種関数の使用例
アプリケーションプログラムは「ドライバ I/F ライブラリ」を「デバイスドライバ」経由でボードへの各種操作を行います。
この各種操作の一例をサンプルプログラムで表します。

8.1 サンプルプログラムの構成

サンプルプログラム・ソースファイルは次の構成となっています。

```
├──README.TXT          ファイル構成について
├──HISTORY.TXT         バージョンアップ履歴
│
├──SMP                 サンプルプログラム
│   ├──smp5144.exe       サンプルプログラム実行ファイル
│   ├──clk.bat           実行ファイル作成用バッチファイル      (MS-C V60)
│   ├──smp5144.c         サンプルプログラム･メインソース
│   ├──ldio5144.lib      リンク用ドライバ関数ライブラリファイル(ラージモデル用)
│   └──hdio5144.h        ドライバ関数用ヘッダーファイル
│
├──DRV                 デバイスドライバ
│   └──hdio5144.drv      デバイスドライバ本体
│
└──LIB                 ドライバ関数
    ├──ldio5144.lib      ラージモデル用
    ├──mdio5144.lib      ミディアムモデル用
    ├──cdio5144.lib      コンパクトモデル用
    ├──sdio5144.lib      スモールモデル用
    └──hdio5144.h        ドライバ関数用ヘッダーファイル
```

8.2 サンプルプログラムの起動

実行ファイル"ｓｍｐ５１４４．ｅｘｅ"を起動すると，下記の画面が表示されます。

```
*** HPCI-DIO5144 Sample: hidio5144.lib Ver1.0  &  hidio5144.drv Ver1.00 ***
                Bus=01 Dev=03  I/O=0x0000 IRQ=17 CTL=00  BID=00  Open?= _
```

図８．２－１　サンプルプログラムの起動・ボード選択画面

ここで表示されたボードの選択をキー入力１文字で行います．選択できるボードは１枚だけです．

◎ 割込機能付きで選択　・・・'Ｉ'または'ｉ'
◎ 割込機能なしで選択　・・・'Ｙ'または'ｙ'
◎ 選択しない（次のボード）・・ その他キー．ただし最後のボードを表示している場合は，そのボードが割り込み無しで選択されます．

ボード選択が終了するとメイン画面が表示されます．（下記は割込使用時の表示内容）

```
*** HPCI-DIO5144 Sample: hidio5144.lib Ver1.0  &  hidio5144.drv Ver1.00 ***
                                        割込ｽﾃｰﾀｽ: 0xffff   割込回数:      0
 IN  48 ---- 41  40 ---- 33  32 ---- 25  24 ---- 17  16 ---- 9    8 ---- 1
 IN  96 ---- 89  88 ---- 81  80 ---- 73  72 ---- 65  64 ---- 57  56 ---- 49
    6:oooooooo  5:oooooooo  4:oooooooo  3:oooooooo  2:oooooooo  1:ooooooo1
    f tttttttt  f tttttttt  F hhhhhhhh  F hhhhhhhh  f iiiiiiii  f iiiiiiii
   12:oooooooo 11:oooooooo 10:oooooooo  9:oooooooo  8:1o1o1o1o  7:o1o1o1o1
    F ........  F ........  f ........  f ........  f pppppppp  f pppppppp
  OUT 48 ---- 41  40 ---- 33  32 ---- 25  24 ---- 17  16 ---- 9    8 ---- 1
    6:11111111  5:11111111  4:oooooooo  3:oooooooo  2:oooooooo  1:ooooooox
     ........    ........    ........    ........    ........    ....s..s
 ｶｳﾝﾀ入力ﾊﾟﾙｽ数: 65535         連続ﾃﾞｰﾀ  FIFOﾃﾞｰﾀ数: 256 / Readﾃﾞｰﾀ数: 2000
 PWM 出力残数   :     0         ｶｳﾝﾀﾃﾞｰﾀ  FIFOﾃﾞｰﾀ数: 256 / Readﾃﾞｰﾀ数: 2000

 ***** 2.Execute Command list *****      ===== 3.Opt/Int Port Regist. ====
0:DO 出力    1:ﾌｨﾙﾀ切    2:ﾗｯﾁ切        0:出力切替  1:ﾌｨﾙﾀ       2:ﾗｯﾁ
3:ﾄﾗﾝｽﾌｧ入  4:連続入    5:PWM 出力     3:ﾄﾗﾝｽﾌｧ    4:連続読込  5:PWM 出力
6:ｶｳﾝﾀ入力  7:FIFO 読取  8:FIFO ｸﾘｱ      6:ｶｳﾝﾀ      7:ｺﾝﾊﾟﾚｰﾀ   8:ｲﾍﾞﾝﾄﾀｲﾏ
9:ﾗｯﾁｸﾘｱ    A:ｶｳﾝﾀｸﾘｱ                  9:割込
  [ 0:End  1:BoardInit  2:Cmd Exec.  3:Register ] = _
```

図８．２－２　サンプルプログラムのメイン画面

メイン画面は下記の各部分に分けられ，それぞれ次の意味や機能を持ちます．

（１）タイトル・認識ボード枚数とソフトのバージョン番号

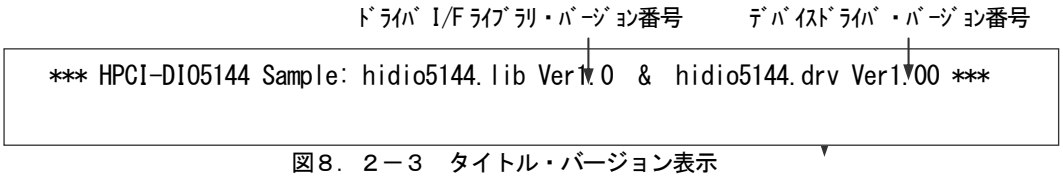


図８．２－３　タイトル・バージョン表示

（２）割り込み状態

```
                                        割込ｽﾃｰﾀｽ: 0xffff    割込回数:       0
```

図８．２－４　割り込み状態表示

割り込み使用時に表示されます．不使用時はタイトルのみで数値部分は表示されません．

○ 割込ステータス・・・ＩＮ１～１６の割込み状態を表します．
○ 割込み回数　　・・・発生した割込みによって実行された割り込み処理の回数を表示します．

（３）入力ポート状態

```
 IN  48 ---- 41  40 ---- 33  32 ---- 25  24 ---- 17  16 ---- 9    8 ---- 1
 IN  96 ---- 89  88 ---- 81  80 ---- 73  72 ---- 65  64 ---- 57  56 ---- 49
    6:oooooooo  5:oooooooo  4:oooooooo  3:oooooooo  2:oooooooo  1:ooooooo1
    f tttttttt  f tttttttt  F hhhhhhhh  F hhhhhhhh  f iiiiiiii  f iiiiiiii
   12:oooooooo 11:oooooooo 10:oooooooo  9:oooooooo  8:1o1o1o1o  7:o1o1o1o1
    F ........  F ........  f ........  f ........  f pppppppp  f pppppppp
```

図８．２－５　入力ポート状態表示

入力ポート１２ポート分の入力状態およびフィルタ，ラッチ，トランスファ接点，連続データ読み出しの各機能の設定状態を各ビット，各ポート毎に２段で表現します．

○ IN　1-48 上段 ・・・ 入力ポート１～６ の入力状態．
　　下段 ・・・ 入力ポート１～６の機能設定状態．
○ IN 49-96 上段 ・・・ 入力ポート７～１２の入力状態．
　　下段 ・・・ 入力ポート７～１２の機能設定状態．

ＩＮ	ビット値	各ﾎﾟｰﾄ 欄外	IN96-81	IN80-65	IN64-49	IN48-33	IN32-17	IN16-1
	名 称	ﾌｨﾙﾀ ON	ﾎﾟｰﾄ 12/11	ﾎﾟｰﾄ 10/9	ﾎﾟｰﾄ 8/7	ﾎﾟｰﾄ 6/5	ﾎﾟｰﾄ 4/3	ﾎﾟｰﾄ 2/1
上段	１表示	F	1	1	1	1	1	1
	０表示	f	o	o	o	o	o	o
下段	１表示		.	.	p	t	h	i
	０表示		.	.	.	.	.	.

表８．２－１　入力信号状態表示文字一覧

各記号の意味は次の通りです．

１／ｏ　・・・ 入力信号の ON (1) / OFF (o) 状態．ビット単位の表示
Ｆ／ｆ　・・・ フィルターON (F) / OFF(f) 状態．　ポート単位の表示
ｉ／．　・・・ 入力割り込みの設定状態（i:有効 / .:無効）．ビット単位の表示
ｈ／．　・・・ ラッチ機能の設定状態（h:有効 / .:無効）．ビット単位の表示
ｔ／．　・・・ トランスファー接点の設定状態（t:有効 / .:無効）．ビット単位の表示
ｐ／．　・・・ 連続データのラッチ有効の設定状態（p:有効 / .:無効）．ビット単位の表示

（4）出力ポートの状態

```
OUT 48 ---- 41  40 ---- 33  32 ---- 25  24 ---- 17  16 ---- 9    8 ---- 1
6:11111111  5:11111111  4:oooooooo  3:oooooooo  2:oooooooo  1:ooooooox
  ........    ........    ........    ........    ........    ....s..s
```

図８．２－６　出力ポート状態表示

出力ポート６ポート分の出力状態およびコンパレータ，ＰＷＭ出力，イベントタイマ出力の各機能の設定状態を各ビット，各ポート毎に２段で表現します．

○ OUT 1-48 上段 ・・・ 出力ポート１～６の入力状態．
　　下段 ・・・ 出力ポート１～６機能設定状態．

ＯＵＴ	ビット値	OUT48-33	OUT32-17	OUT16-5	OUT4-1
	名 称	Port6/5	Port4/3	Port2/1	
上段	１表示	1	1	1	
	０表示	o	o	o	
下段	１表示	.	.	.	s
	０表示	.	.	.	.

表８．２－２　出力信号状態表示文字一覧

各記号の意味は次の通りです．

１ / ｏ ・・・ 出力信号の ON (1) / OFF (o) 状態．ビット単位の表示
ｓ / ． ・・・ 特殊機能出力 ON (s) / OFF(.) 状態．ビット単位の表示

（5）パルス入出力，ＦＩＦＯ入力状態

```
ｶｳﾝﾀ入力ﾊﾟﾙｽ数: 65535          連続ﾃﾞｰﾀ  FIFOﾃﾞｰﾀ数: 256 / Readﾃﾞｰﾀ数: 2000
PWM 出力残数   :     0          ｶｳﾝﾀﾃﾞｰﾀ  FIFOﾃﾞｰﾀ数: 256 / Readﾃﾞｰﾀ数: 2000
```

図８．２－７　パルス入出力，ＦＩＦＯ入力状態表示

エンコーダからのパルスカウンタ入力値，ＰＷＭ出力パルス数，および連続読込データとカウンタ入力データのＦＩＦＯへの取込み状況を表示します．表示は１０進で表します．

○ ｶｳﾝﾀ入力ﾊﾟﾙｽ数　　・・・ エンコーダからの入力パルスの計数値
○ PWM 出力残数　　・・・ ＰＷＭ出力の出力残りパルス数
○ 連続ﾃﾞｰﾀ FIFOﾃﾞｰﾀ数　・・・ ＦＩＦＯに取り込まれている連続読込データのデータ数
○ 連続データ Readﾃﾞｰﾀ数 ・・・ ＦＩＦＯから既に読み出したデータ数
○ ｶｳﾝﾀﾃﾞｰﾀ FIFOﾃﾞｰﾀ数　・・・ ＦＩＦＯに取り込まれているカウントデータのデータ数
○ ｶｳﾝﾀﾃﾞｰﾀ Readﾃﾞｰﾀ数　・・・ ＦＩＦＯから既に読み出したデータ数

（６）動作指令の選択

```
 ***** 2.Execute Command list *****      ===== 3.Opt/Int Port Regist. ====
0:DO 出力     1:ﾌｨﾙﾀ切     2:ﾗｯﾁ切          0:出力切替  1:ﾌｨﾙﾀ        2:ﾗｯﾁ
3:ﾄﾗﾝｽﾌｧ入  4:連続入     5:PWM 出力       3:ﾄﾗﾝｽﾌｧ     4:連続読込  5:PWM 出力
6:ｶｳﾝﾀ入力  7:FIFO 読取  8:FIFO ｸﾘｱ        6:ｶｳﾝﾀ       7:ｺﾝﾊﾟﾚｰﾀ   8:ｲﾍﾞﾝﾄﾀｲﾏ
9:ﾗｯﾁｸﾘｱ     A:ｶｳﾝﾀｸﾘｱ                    9:割込
  [ 0:End  1:BoardInit  2:Cmd Exec. 3:Register ] = _
```

図８．２－７　動作指令表示

メインメニューの番号（０～３）をキーボードから１文字入力で選択します.

０ (End)　　・・・ プログラムを終了します.
１ (BoardInit)　・・・ ボードの初期化を行い，出力をデフォルトの状態にします.
２ (Cmd Exec.)　・・・ 動作コマンドを実行します.
３ (Register)　・・・ 各種パラメータの設定と変更を行います.

上記メインメニューの’２’または’３’を選択すると，サブメニュー番号の入力が促されます.
希望する項目を，画面の中に表示されている番号で指定します.

◎ メインメニュ’２’(Cmd Exec.) 選択時

０ (DO 出力)　　・・・指定したDO ポートに対してバイト出力を行います.
１ (ﾌｨﾙﾀ無効/切/入)　・・・ フィルター機能を On/Off します.
２ (ﾗｯﾁ無効/切/入)　・・・ ラッチ機能を On/Off します.
３ (ﾄﾗﾝｽﾌｧ無効/切/入)・・・ トランスファ接点機能を On/Off します.
４ (連続無効/切/入)　・・・ 連続データ読込機能を On/Off します.
５ (PWM 無効/出力)　・・・ ＰＷＭ出力を On/Off します.
６ (ｶｳﾝﾀ無効/入力)　・・・ カウンタ入力を On/Off します.
７ (FIFO 読取ﾀ)　・・・連続読込データまたはカウントデータをＦＩＦＯから読み取ります.
８ (ﾗｯﾁｸﾘｱ)　・・・ ラッチをリセットします.
９ (ｶｳﾝﾀ読取ﾀ)　・・・ カウンタをＯリセットします.

※項目１～６については，関連する設定が正しく行われていない場合，画面上に”無効”と表示され，
　Ｏｎ操作はできません.

◎ メインメニュ’３’(Register) 選択時

０ (出力切替)　・・・特殊出力と通常ＤＯ出力を切り替えます.
１ (ﾌｨﾙﾀ)　　・・・ フィルタ有効選択とフィルター時間を設定します.
２ (ﾗｯﾁ)　　・・・ ラッチＢｉｔ選択とラッチ条件を設定します.
３ (ﾄﾗﾝｽﾌｧ)　・・・ トランスファ接点有効選択を設定します.
４ (連続読込)　・・・ 連続データ読込選択と読込条件を設定します.
５ (PWM 出力)　・・・ ＰＷＭ出力条件を設定します.
６ (ｶｳﾝﾀ)　　・・・ カウンタ入力条件を設定します.
７ (ｺﾝﾊﾟﾚｰﾀ)　・・・カウンタ比較値とコンパレータ条件を設定します.
８ (ｲﾍﾞﾝﾄﾀｲﾏ)　・・・ イベントタイマをを設定します.
９ (割込)　　・・・ 割り込みＢｉｔ選択，割り込み信号条件，割り込みソース切替を設定します.

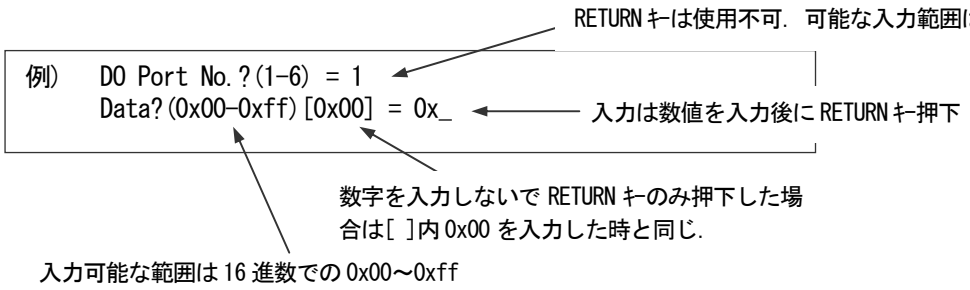
8.3 サンプルプログラムの操作

サンプルプログラムでは，次の手順で操作が進められます.
画面上に表示される割り込み，入出力ポート，カウンタ，ＰＷＭ，ＦＩＦＯ状態は常時読込が行われ，表示が更新されます.

8.3.1 入力規則および表示規則，注意事項について

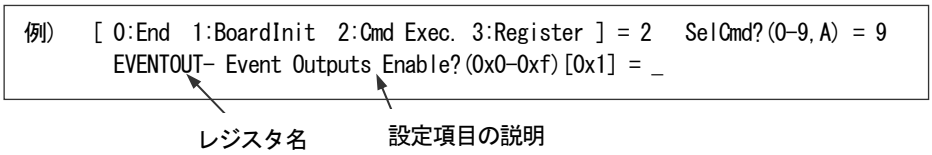
■ 入力規則

① 複数桁の値を入力する場合は，値を入力した後ＲＥＴＵＲＮキーで確定します．一文字入力の場合は指定された数字または文字を入力した時点で確定します.
② 数値や文字を入力しないでＲＥＴＵＲＮキーのみを押した場合は[ ]で囲まれた数値または状態が設定されます．[ ]の無い設定の場合ＲＥＴＵＲＮキーは無効です.
③ ＥＳＣキーは入力を中止（Cancel）する時に使います．入力中のどの時点でも使用できます.
④ 入力した文字を１文字消す場合はＢＡＣＫＳＰＡＣＥキーうを使います.
⑤ 入力個所に”0x”と表示されている場合は１６進数で入力します．その他の場合は１０進数での入力で行います．なお，１６進数でのＡ～Ｆは大文字，小文字どちらでも構いません.
⑥ 入力範囲や文字種は，画面に表示されている指示（( )内の範囲）に従います.



■ 表示規則

① 表示されている数値，または入力する数値の前に”0x”と表示されている場合は１６進数での表記になっています.
② パラメータ設定で，[ ]内に表示される値は，現在設定されている値です.
③ ”Cmd Exec.”（ﾒｲﾝﾒﾆｭｰ'2'）での項目１～６のタイトルは，操作を行った時に動作する内容を表した表示となります.
（例として表示が”PWM 出力”となっている場合　→　現在は PWM は停止しており，このコマンドを実行する事によって出力に切り替える事ができる事を表現しています）
④ コマンド入力やパラメータ設定などで画面を操作している間は，画面上の状態表示は更新されません.
⑤ 実行した操作が正常に終了した場合は　″=＞Normal End”が表示されます．エラー発生した場合は″=＞Illigal End ″と表示され，さらに画面右上部にエラー内容が表示されます.
⑥ パラメータ設定では，設定するレジスタを明確にするため入力項目の左端にレジスタ名を記載しています.



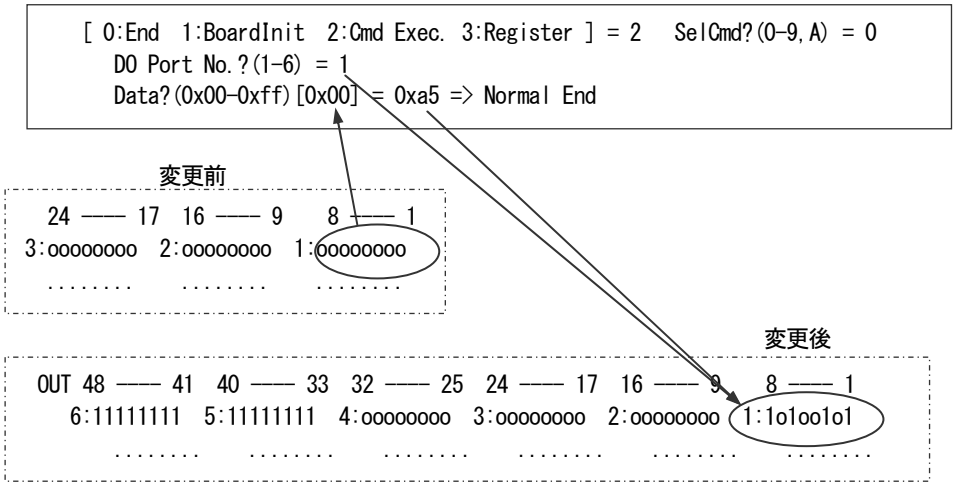
■ 注意事項

① パラメータの設定および変更では入力範囲のチェックは行っていますが，他のパラメータとの整合性のチェック，および設定した値による効果の評価や判定はしていません.
② ”FIFO 読取”にて読み取ったデータはカレントドライブに保存されます．この時，容量不足や書き込み禁止等で書き込みができなくなった場合，アプリケーションではエラー処理をしていない為ＯＳが直接エラーを返します．この場合はＣＰＵの再起動が必要になります.

8.3.2 動作コマンド実行の操作

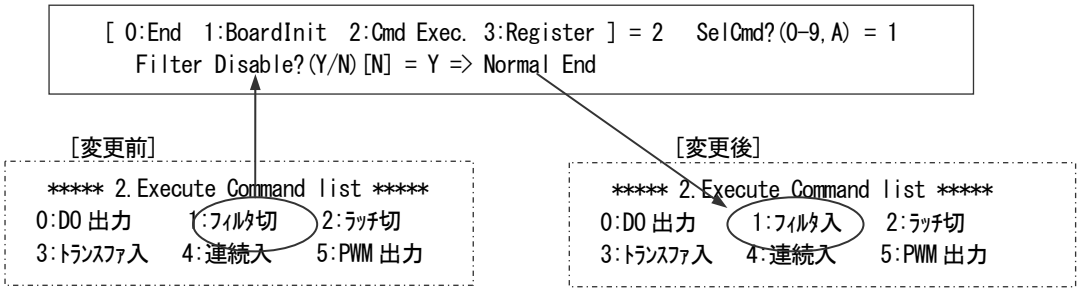
（1）ＤＯ出力

指定した出力ポートに１バイトのデータを書き込みます.



（2）フィルタ動作

フィルタ機能を全ポート無効または有効（設定で有効に設定したポートのみ）に切り替えます.



変更前の設定は"ﾌｨﾙﾀ入"になっているので, ﾒﾆｭ画面は反対の動作である"切"の表示である

"ﾌｨﾙﾀ切"に変更後は, ﾒﾆｭ画面は反対の動作である"入"表示になる

ﾌｨﾙﾀに関係するﾊﾟﾗﾒｰﾀ設定が正しく行われていない場合, ﾒﾆｭには以下のように表示され, "ﾌｨﾙﾀ入"には出来ません. この場合は先にﾊﾟﾗﾒｰﾀの設定を正しく行う必要があります.

※ 設定と表示方法, 操作に関してはﾗｯﾁ動作, ﾄﾗﾝｽﾌｧ動作, 連続動作, PWM 出力動作, ｶｳﾝﾀ動作についても同様です.

- ﾌｨﾙﾀ無効／入／切
- ﾗｯﾁ無効／入／切
- ﾄﾗﾝｽﾌｧ無効／入／切
- 連続無効／入／切
- PWM 無効／出力／停止
- ｶｳﾝﾀ無効／入力／停止

（3）ラッチ動作

ラッチ機能を全ビット無効または有効（設定で有効に設定したビットのみ）に切り替えます.

```
[ 0:End  1:BoardInit  2:Cmd Exec. 3:Register ] = 2   SelCmd?(0-9,A) = 2
  Latch Disable?(Y/N)[N] = Y => Normal End
```

（4）トランスファ動作

トランスファ接点を全点無効または，設定で有効に設定した接点のみ有効に切り替えます.

```
[ 0:End  1:BoardInit  2:Cmd Exec. 3:Register ] = 2   SelCmd?(0-9,A) = 3
  Transfer Enable?(Y/N)[N] = Y => Normal End
```

（5）連続動作

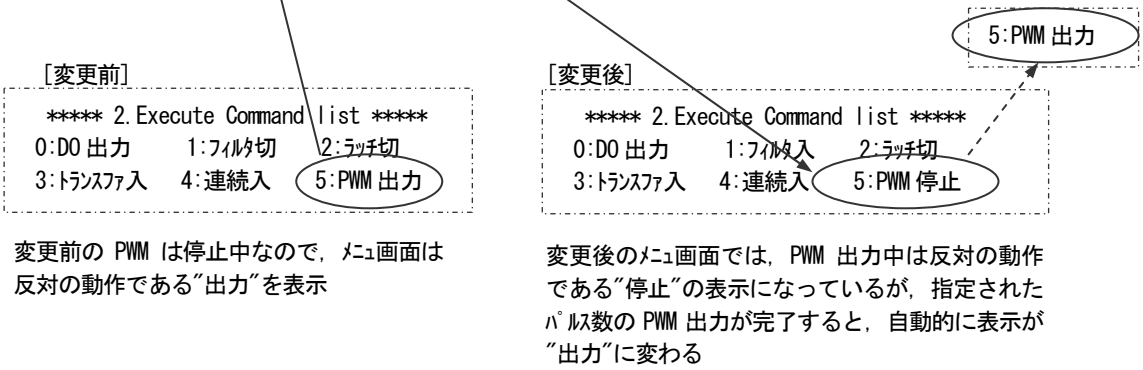
連続読込を無効または有効に切り替えます.

```
[ 0:End  1:BoardInit  2:Cmd Exec. 3:Register ] = 2   SelCmd?(0-9,A) = 4
  Parallel Inputs Data Sampling Stop?(Y/N)[N] = Y => Normal End
```

（6）PWM動作

PWM出力を無効または有効に切り替えます.

```
[ 0:End  1:BoardInit  2:Cmd Exec. 3:Register ] = 2   SelCmd?(0-9,A) = 5
  PWM Outputs Start?(Y/N)[N] = Y  => Normal End
```



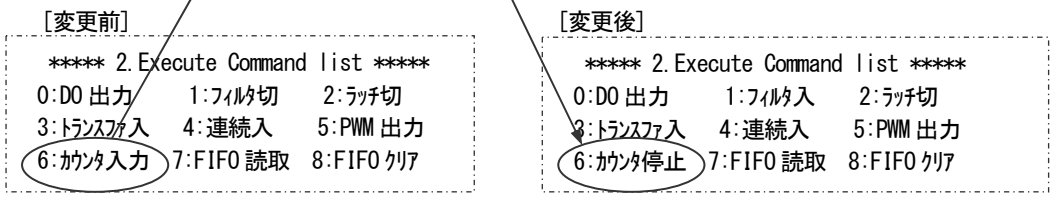
変更前の PWM は停止中なので，ﾒﾆｭ画面は反対の動作である"出力"を表示

変更後のﾒﾆｭ画面では，PWM 出力中は反対の動作である"停止"の表示になっているが，指定されたﾊﾟﾙｽ数の PWM 出力が完了すると，自動的に表示が"出力"に変わる

（7）カウンタ動作

パルス入力カウントを無効または有効に切り替えます.

```
[ 0:End  1:BoardInit  2:Cmd Exec. 3:Register ] = 2   SelCmd?(0-9,A) = 6
  Counter Start?(Y/N)[N] = Y => Normal End
```



変更前の設定はｶｳﾝﾀ停止になっているので，ﾒﾆｭ画面は反対の動作である"入力")の表示である

"ｶｳﾝﾀ入力"に変更後は，ﾒﾆｭ画面は反対の動作である"停止"の表示になる

（8）ＦＩＦＯ読取

ＦＩＦＯに取り込まれたデータを取り出します.

FIFO 読取をﾊﾟﾙｽ入力ﾃﾞｰﾀ (1:Counter) または連続読出ﾃﾞｰﾀ (2:Parallel) どちらに対して行うか選択します.

FIFO から読み出したﾃﾞｰﾀを保存するﾌｧｲﾙ名を指定します. ﾌｧｲﾙ名は最大 8 文字で, 拡張子は入力不要です (txt 固定) . 同名ﾌｧｲﾙの存在はﾁｪｯｸせず上書きします.

```
[ 0:End  1:BoardInit  2:Cmd Exec. 3:Register ] = 2   SelCmd?(0-9,A) = 7
  FIFO Select?(1:Counter/2:Parallel) = 1
  Data File Name?(*.txt/A-Z,0-9)[DAT10000] = AAAAAAAA
  Read Start?(Y/N)[N] = Y => FIFO reading... (Stop: Push any Key)
  Read Data Saving... => Normal End
```

取得を中止, または取得したﾃﾞｰﾀが 9999 データになると読取を終了し, それまでのﾃﾞｰﾀをﾌｧｲﾙに保存します.

FIFO 読取中に表示されます. 読取中に任意のｷｰを押下すると取得を中止します.

（9）ＦＩＦＯクリア

ＦＩＦＯに残っているデータを破棄します.

```
[ 0:End  1:BoardInit  2:Cmd Exec. 3:Register ] = 2   SelCmd?(0-9,A) = 8
  FIFO Select?(1:Counter/2:Parallel) = 1
  FIFO Clear?(Y/N)[N] = Y => Normal End
```

（１０）ラッチクリア

ラッチされている全ビットを解除します.

```
[ 0:End  1:BoardInit  2:Cmd Exec. 3:Register ] = 2   SelCmd?(0-9,A) = 9
  Latch Data All Clear?(Y/N)[N] = Y => Normal End
```

（１１）カウンタクリア

カウンタ値を設定されたプリセット値で初期化します.

```
[ 0:End  1:BoardInit  2:Cmd Exec. 3:Register ] = 2   SelCmd?(0-9,A) = A
  Counter Clear By Preset Value?(Y/N)[N] = Y => Normal End
```

8.3.3 パラメータ設定の操作

パラメータ値の詳細については別冊「ＨＰＣＩ－ＤＩＯ５１４４　ユーザーズマニュアル」を参照下さい.

（１）出力切替

```
[ 0:End  1:BoardInit  2:Cmd Exec. 3:Register ] = 2   SelReg?(0-9) = 0
  EVENTOUT- Event Outputs Enable?(0x0-0xf)[0x1] = 0xf => Normal End
```

（２）フィルタ

```
[ 0:End  1:BoardInit  2:Cmd Exec. 3:Register ] = 2   SelReg?(0-9) = 1
   DIFILSEL- Filter Enable?(0x0-0xfff)[0xc0c] = 0xfff
   DIGIFIL - Filtering Value?(0x0-0xff)[0x01] = 0xff => Normal End
```

（３）ラッチ

```
[ 0:End  1:BoardInit  2:Cmd Exec. 3:Register ] = 2   SelReg?(0-9) = 2
  LATENA - Latch Enable?(0x0-0xffff)[0xf00f] = 0x0000
  LATEDGL- Trigger Type Low Byte?(0x0-0xffff)[0x0000] = 0x5555
  LATEDGH- Trig. Type High?(0x0-0xffff)[0x0000] = 0x5555 => Normal End
```

（４）トランスファ

```
[ 0:End  1:BoardInit  2:Cmd Exec. 3:Register ] = 2   SelReg?(0-9) = 3
  CHATT- Transfer Enable?(0x0-0xff)[0x81] = 0x00 => Normal End
```

（５）連続読込

```
[ 0:End  1:BoardInit  2:Cmd Exec. 3:Register ] = 2   SelReg?(0-9) = 4
  STROBE  - Continuation Read Enable?(0x0-0xffff)[0xffff] = 0x00ff
  STRB_TRG- Cont. Read Condition?(0x0-0x11)[0x00] = 0x01 => Normal End
```

（６）ＰＷＭ

```
[ 0:End  1:BoardInit  2:Cmd Exec. 3:Register ] = 2   SelReg?(0-9) = 5
  PULSSET - Output Pulse Condition?(0x0-0x1b)[0x00] = 0x00
  PWMWIDTH- PWM Pulse Width?(1-65534)[1] = 5000
  PWMBASE - PWM Cyclic Distance?(2-65535)[2] = 10000
  PULSCNT - Output Pulse Count?(1-65534,65535)[65535] = 10000 => Normal End
```

（７）カウンタ

```
[ 0:End  1:BoardInit  2:Cmd Exec. 3:Register ] = 2   SelReg?(0-9) = 6
  ENCSET - Counter Input Condition?(0x0000-0x311f)[0x100c] = 0x000c
  COUNTCL- Preset Value?(0x0-0xffff)[0x0000] = 0x8000 => Normal End
```

（８）コンパレータ

```
[ 0:End  1:BoardInit  2:Cmd Exec. 3:Register ] = 2   SelReg?(0-9) = 7
  COMPREATER- Comparative Value?(0x0-0xffff)[0x0000] = 0x8000
  CMPSET    - Comp. Logic?(0x0-0xff06)[0x0005] = 0x0a05 => Normal End
```

（9）イベントタイマー

```
[ 0:End  1:BoardInit  2:Cmd Exec. 3:Register ] = 2   SelReg?(0-9) = 8
  EVENTTM- Event Timer Value?(0,1-255)[1] = 0 => Normal End
```

（１０）割込

```
[ 0:End  1:BoardInit  2:Cmd Exec. 3:Register ] = 2   SelReg?(0-9) = 9
  INTENA - Interrupt Enable?(0x0-0xffff)[0x0000] = 0x0001
  INTEDGL- Edge Type Low Byte?(0x0-0xffff)[0x0000] = 0x5555
  INTEDGH- Edge Type High?(0x0-0xffff)[0x0000] = 0x5555
  INTSRC - Interrupt Source?(0x0-0xfff0)[0x0000] = 0x0010  => Normal End
```