HPCI−DIO580シリーズ

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ライブラリ

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

この説明書では

ＨＰＣＩ－ＤＩＯ５８０シリーズのボードは

ＨＰＣＩ－ＤＩＯ５８０

ＨＰＣＩ－ＤＩＯ５４８　です。

ＤＯＳ版ドライバＩ/Ｆライブラリ　として

メモリモデル別に次の４種類があります。

αｉｄｉｏ５８０．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・・・・・ｌｉｄｉｏ５８０．ｌｉｂ

α＝Ｃ・・コンパクトモデル・・・ｃｉｄｉｏ５８０．ｌｉｂ

α＝Ｍ・・ミディアムモデル・・・ｍｉｄｉｏ５８０．ｌｉｂ

α＝Ｓ・・スモールモデル・・・・ｓｉｄｉｏ５８０．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
ＴＥＬ　０３－３８４６－３８０１
ＦＡＸ　０３－３８４６－３７７３
sales@hivertec.co.jp

第３．０版 ２００５年１０月　７日発行

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．

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す恐れのある装置（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．

はじめに

この度は，弊社ＮＣボードシリーズをご採用頂きまして，誠に有り難う御座います．
本書は，ソフトウェアをご使用して頂く場合の取り扱い，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、本ソフトウエアを利用するＮＣボード常設箇所付近の分かりやすい場所に常時保管し，必要に応じて適宜参照・確認頂きますよう，お願い致します．

安全上の注意

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

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

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

１．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．

２．適合ＯＳ

⚠ **警 告**

⚠
本製品は、ＭＳ－ＤＯＳ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．

３．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．

４．対象開発ツール

警 告

本ソフトウエア中のサンプルプログラムは，以下の開発ツールを対象にしています．
・Microsoft C Ver6.0　以上
上記以外の開発ツールでのご利用については，弊社営業までお問合せ下さい．

５．サンプルプログラムの実行

警 告

本ソフトウエア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．

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

６．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，モータや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．

も　く　じ

はじめに

Ⅰ　デバイスドライバのインストール

Ⅱ　ボードを複数枚使用する場合

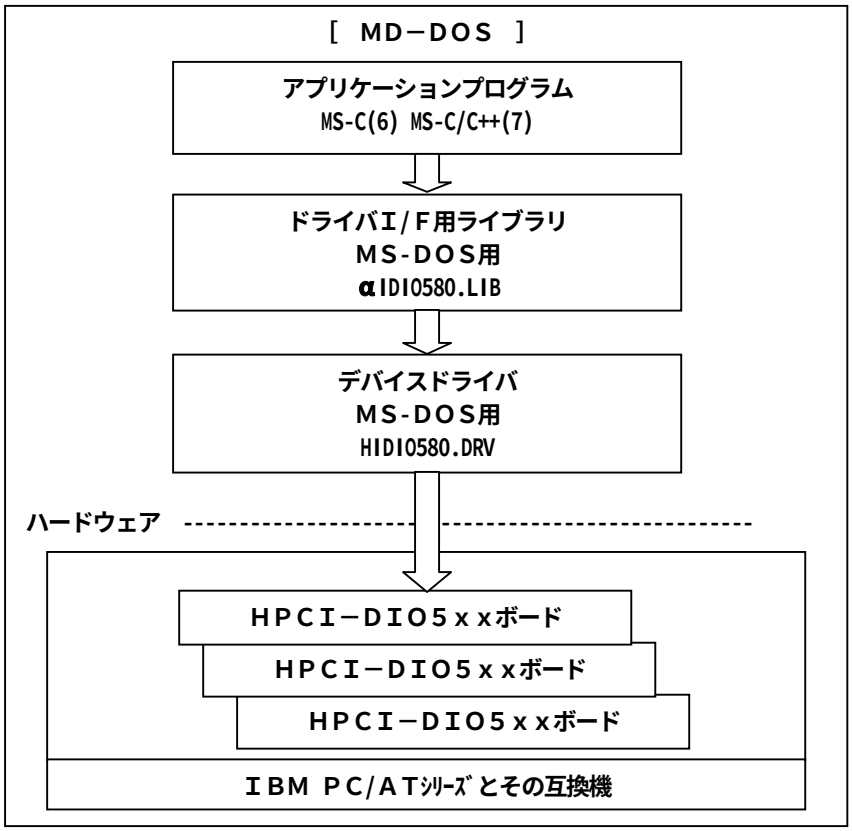
Ⅲ　ドライバＩ／Ｆ用ライブラリの使用方法

はじめに

本書は，ＨＰＣＩ－ＤＩＯ５８０シリーズボード(DIO580,DIO548)のソフトウェアに関して解説を行うものです．
（上記のボード総称として以降は**ＤＩＯ５ｘｘ**と記します。）
このソフトウェアはＭＳ－ＤＯＳで**ＤＩＯ５ｘｘ**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。

１．ソフトウエアの構成



２．フロッピーディスクの内容

ＤＯＳ版デバイスドライバのＦＤには次表に示す各種のファイルが格納されています。

ディレクトリ	ファイル名	記　　事	
￥	history.txt	バージョンアップの内容を記載	
\drv	hippd530.drv	デバイスドライバ本体	
\lib	sidio580.lib	ドライバ I/F ライブラリ本体 (αidio580.lib)	スモールモデル用
	cidio580.lib		コンパクトモデル用
	midio580.lib		ミディアムモデル用
	lidio580.lib		ラージモデル用
	hidio580.h	ﾄﾞﾗｲﾊﾞ･ﾍｯﾀﾞｰﾌｧｲﾙ	ｱﾌﾟﾘｹｰｼｮﾝで #include 定義
\smp	smp580 .exe	サンプルプログラム実行形式 (ﾗｰｼﾞﾓﾃﾞﾙ)	
	clk .bat	実行ファイル作成用バッチファイル (MS-C Ver6.0)	
	smp580 .c	サンプルソースプログラム	
	hidio580.h	ﾄﾞﾗｲﾊﾞ･ﾍｯﾀﾞｰﾌｧｲﾙ	
	lidio580.lib	ラージモデル用ドライバ I/F ライブラリ	

（注）１．サンプルプログラムは"MS-C Ver6.0"を使用して実行ファイルを作成しています。

Ⅰ　デバイスドライバのインストール

１．ＤＯＳ版のインストール

（1）手動設定

新規に HIDIO580 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" HIDIO580.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 = HIDIO580.DRV

（注）指定は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:\HIDIO }ディレクトリにコピーした場合

DEVICE = C:\HIDIO\HIDIO580.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．

（注）Ｗｉｎｄｏｗｓパソコンには，この操作を行わないで下さい．

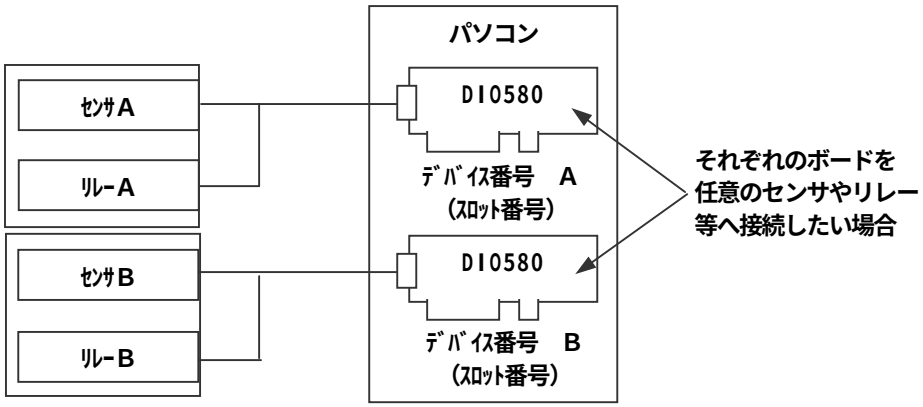
２．アンインストール

手動設定で説明した" CONFIG.SYS "内のドライバの登録行を削除します．

また，ドライバファイル本体を削除します．

II　ボードを複数枚使用する場合

ＤＩＯ５８０ボードを同一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に対応させたい場合について説明いたします．



１．ボードのスロット番号の確認

ＰＣＩバスのカードを装着するスロットには，それぞれのパソコンにより固定の番号が割り振られています．
その番号を知ることにより装着するスロットにより任意の外部装置への接続が可能になります．

２．スロット番号の確認方法

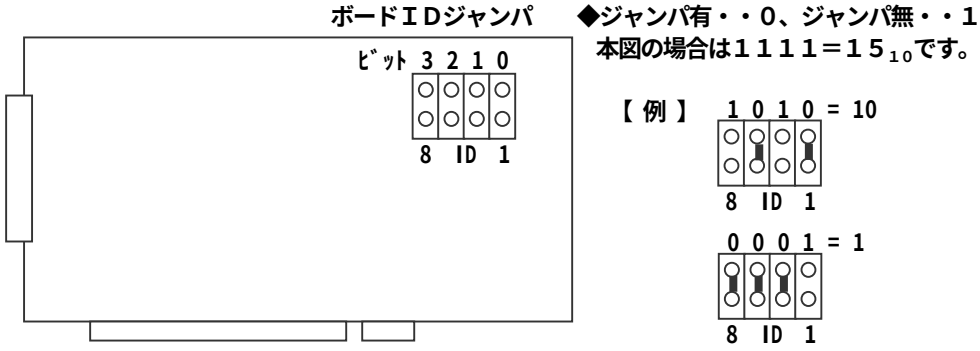
次の実行ファイルを起動することにより，現在スロットに装着されているデバイス番号（スロットの番号）を簡単に確認できます．（ボードは４枚まで）

\smp\smp580.exe　（サンプルプログラム実行ファイル）

このプログラムは，ドライバＩ／Ｆライブラリに対してデバイス情報の取得を行っています．
これにより，デバイス番号（スロット番号）の取得を行うことができます．
（詳細は、「III：５．関数詳細　(2)デバイス情報の取得」をご覧ください．）

３．ボードＩＤ（ボードアドレス）の使用

ＤＩＯ５４８ボードではボード上のジャンパで設定したボードＩＤ（0～15）が使用できます．
（ＤＩＯ５８０ボードは除外）
この使用方法は後述します．



III　ドライバＩ／Ｆ用ライブラリの使用方法

１．概　　要

ＤＯＳ版ライブラリはＭＳ－ＤＯＳにおいて，ＨＰＣＩ－ＤＩＯ５ｘｘボードの制御を行うための関数群です．
ＤＯＳ版ライブラリはアプリケーション結合用の４種類のメモリモデル（ラージ･コンパクト･メディアム･スモール）に対応して，個々のメモリモデル版ライブラリファイルが提供されます．

各関数は"MS-C"等のＣ言語から外部関数として起動されます．
アプリケーションプログラムがライブラリ中の関数を呼び出しを行い，ライブラリはＨＰＣＩ－ＤＩＯ５ｘｘのデバイスドライバにアクセスします．

デバイスドライバは，ＭＳ-ＤＯＳでは"HIDIO580.DRV"が使用されます．

２．関数一覧

ドライバＩ／Ｆ用ライブラリには，次の１５種類の関数が含まれます．

No	関　数　名　称	機　　能	記載ﾍﾟｰｼﾞ
１	hdio580_GetDeviceCount()	ボード枚数の取得	８
	hdio580_GetDevCntVerNo()	ボード枚数とバージョン番号の取得	
２	hdio580_GetDeviceInfo()	デバイス情報の取得	８
３	hdio580_OpenDeviceID()	デバイスのオープン(ﾎﾞｰﾄﾞID参照)	９
４	hdio580_OpenDevice()	デバイスのオープン(ﾎﾞｰﾄﾞID無視)	１０
５	hdio580_CloseDevice()	デバイスのクローズ	１１
６	hdio580_ioswRead()	入出力数切替ポート読込　ＤＩＯ５４８の	１２
７	hdio580_inpReadB()	入力ポートバイト読込	１３
	hdio580_inpReadW()	入力ポートワード読込	
８	hdio580_outpWriteB()	出力ポートバイト書込	１４
	hdio580_outpWriteW()	出力ポートワード書込	
	hdio580_outpReadB()	出力ポートバイト読込	
	hdio580_outpReadW()	出力ポートワード読込	
９	hdio580_intFilterWrite()	割込入力信号 フィルタ設定	１５
	hdio580_intFilterRead()	割込入力信号 フィルタ設定確認	
１０	hdio580_intEdgeWrite()	割込入力信号 エッジ選択	１６
	hdio580_intEdgeRead()	割込入力信号 エッジ選択確認	
１１	hdio580_intIdtSelWrite()	割込入力信号 要因選択	１７
	hdio580_intIdtSelRead()	割込入力信号 要因選択確認	
１２	hdio580_intIdtStsRead()	割込入力信号 割込要因確認	１８
１３	hdio580_intEblWrite()	ボード割込設定	１９
	hdio580_intEblRead()	ボード割込設定確認	
１４	hdio580_SetIntCall()	割込処理関数の登録・消去	２０
（１５）	hdio580_GetLastErr()	エラーコードの取得	７

３．準　　備

次のファイルをソースプログラムに追加して下さい．

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

次のファイルをリンクファイルとします．
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．

- ・sidio580.lib　（スモールモデル）　[ｺｰﾄﾞ：64KB 未満､ﾃﾞｰﾀ：64KB 未満]
- ・cidio580.lib　（コンパクトモデル）　[ｺｰﾄﾞ：64KB 未満､ﾃﾞｰﾀ：64KB 以上]
- ・midio580.lib　（ミディアムモデル）　[ｺｰﾄﾞ：64KB 以上､ﾃﾞｰﾀ：64KB 未満]
- ・lidio580.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

①ＤＯＳ版デバイスドライバ・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本ＬＩＢ中の関数は、ほとんどが"short 型"を返すようになっています．

③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，極力同一のデータ型名を使用しています．

```
hidio580.h
   typedef  unsigned char  BYTE;
   typedef  unsigned short WORD;
   typedef  unsigned long  DWORD;

   typedef  void (interrupt far * PINTPROC)(); /* HPCI 割込み処理関数 */
   #define  HINTPROC  interrupt far           /* HPCI 割込み処理関数 */
```

４．制御概念

ライブラリ及びデバイスドライバで複数の HPCI-DIO5xx ボードを制御することができます．

ある１つの HPCI-DIO5xx ボードにアクセスするためには，まずこのデバイスをオープンして，アクセスするための足がかりとなるデバイスＩＤ値を取得する必要があります．

デバイスをオープンするためには，どのようなハードウェアリソースを持つデバイスをオープンするのかという情報が必要となります．
（　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、ＤＯＳのシステム側によって確定されます。）

４．１　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．

```
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(※)    */
} HPCDEVINFO, *PHPCDEVINFO
```

※ＤＩＯ５８０ボードでは，ボードＩＤの値は"０ｘ８００ｆ"となります．
　ＤＩＯ５４８ボードでは，ボード上の設定値（０～１５）となります．

４．２　ボードアクセスの準備手順

①この"HPCDEVINFO"型構造体エリア（の配列）内に，全 HPCI-DIO5xx のデバイス情報をまず取得します．
- ◆ hdio580_GetDeviceCount()・・・ボード枚数の確認
- ◆ hdio580_GetDeviceInfo() ・・・全ボードのデバイス情報を取得

②ある１つの HPCI-DIO5xx 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-DIO5xx 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ＩＤ値を返してきます。
- ◆ hdio580_OpenDeviceID()・・・ボードのオープン処理（ ﾎﾞｰﾄﾞ ID 参照 ）
  または
- ◆ hdio580_OpenDevice()　・・・ボードのオープン処理（ ﾎﾞｰﾄﾞ ID 無視 ）

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．

③出力ポートの初期設定
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．
- ◆ hdio580_outpWriteB() または hdio580_outpWriteW()

④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．
- ◆ hdio580_outpReadB() または hdio580_outpReadW()

⑤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．
- ◆ hdio580_CloseDevice()　・・・ボードのクローズ処理（１枚分）

４．３　デバイスドライバの異常報告

デバイスドライバの諸関数を使用する時，関数の戻り値が異常値（０以外）であった場合には，この異常内容を読込み，異常内容に対応した処理を行います．

（１）異常内容の読込み・・・GetLastErr() 関数の起動

```
DWORD    dErrorCode;
dErrorCode = hdio580_GetLastErr(0);        /* 引数はﾀﾞﾐｰです。 */
```

（２）異常内容の一覧

読　出　値	異　常　内　容
０：NO_ERROR	正常（異常は発生していない）
０ｘ０１（　１） NOT_FOUND	◎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．
０ｘ０２（　２） ALREADY_OPENED	◎同一のデバイスがオープン済 オープン済みのデバイス情報でオープン指令を行いました． ◇デバイス情報先頭アドレスを確認して下さい． ◇ボードのオープン条件を変更します．（オープン関数を参照）
０ｘ０４（　４） INSUFFICIENT_MEMORY	◎ドライバ用メモリの不足 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．）
０ｘ０８（　８） INVALID_HANDLE	◎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．
０ｘ１０（１６） NOT_READY	◎デバイスの入出力ポートなし ボードへの入出力処理ができません． ◇デバイス情報"I/Oﾎﾟｰﾄｱﾄﾞﾚｽ"値が０です． ◎割込制御回路なし 割込処理指令で割込がサポートできません． ◇デバイス情報"IRQ番号"値が０です．
０ｘ２０（３２） ILLEGAL_DEVICE	◎デバイス上の回路不良 デバイス情報は認識できますが、ボード回路が不安定です． その為、オープン処理に失敗しました．
０ｘ４０（６４） ILLEGAL_ACCESS	◎読込/書込中の軸への読込/書込指令 ボード上ＬＳＩへの読込/書込中に同一ポートへの読込/書込指令が行われました． この結果としてＬＳＩの動作が保証されなくなりますから、読込/書込指令を禁止しました． ◇マルチタスク処理では、先行タスクの処理終了まで待ちます． ◇割込処理モジュール内では、この処理を"待処理"とし、割込処理を終了させます．

５．関数詳細

（1）

hdio580_GetDeviceCount()	ＤＩＯ５ｘｘボード枚数の取得
hdio580_GetDevCntVerNo()	ボード枚数･バージョン番号の取得

《機能》

現在パソコンに装着されているＤＩＯ５ｘｘボードの枚数を取得します．
ＤＩＯ５８０，ＤＩＯ５４８ボードの全枚数です。

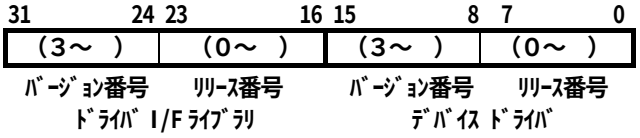
バージョン番号取得関数では、ドライバ本体とドライバ I/F ライブラリのバージョン番号を指定エリアに格納します．

《書式》

```
short hdio580_GetDeviceCount( WORD   dummy );
short hdio580_GetDevCntVerNo( DWORD* verno );
```

《引数》

◆ WORD　dummy・・実際には使用していません．（０として下さい）
◆ DWORD* verno・・バージョン番号が格納されます．



《戻り値》　ＤＩＯ５ｘｘボードの枚数

枚数の値が１以上　・・　実装枚数との一致を確認して下さい．
枚数の値が０の時　・・　「４．３ デバイスドライバの異常報告」を参照して下さい．
この結果は次のようになります．

◇ NO_ERROR （０)……デバイス(ボード)は１枚もなし．
◇ NOT_FOUND （１)……デバイスドライバが存在しない．

《呼び出し例》

```
/* Version No. : Device Driver & Driver I/F library */
    union {
      long        l;       /* Read Data Area */
      char        c[4];    /* drv rel[0],ver[1], lib rel[2],ver[3] */
    } dd_vers;
     short  count;

     count = hdio580_GetDeviceCount(0);
     count = hdio580_GetDevCntVerNo(&dd_vers.l);
```

（2）

hdio580_GetDeviceInfo()	ＤＩＯ５ｘｘボードの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ＤＩＯ５ｘｘボードのデバイス情報を取得します．
この結果、HPCDEVINFO 型の配列にデバイス情報が格納されます．
この値は、デバイスオープン時に利用します．

《書式》

```
short hdio580_GetDeviceInfo( short* pcnDevNumber, PHPCDEVINFO pHpcDevinfo );
```

《引数》

◆ short* pcnDevNumber

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

１：成功　・・　指令と戻りの枚数値を確認して下さい．
０：失敗　・・　「４．３ デバイスドライバの異常報告」を参照して下さい．

《呼び出し例》

パソコンにＤＩＯ５ｘｘが３枚装着されていることを想定します．

```
short       ret;                /* 関数の戻り値 */
short       count = 3;          /* 最大枚数は３ */
HPCDEVINFO  HpcDevInfo[3];      /* ３枚の DIO5xx のﾃﾞﾊﾞｲｽ情報がセットされるべきエリア */
ret = hdio580_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 */
```

（3）　hdio580_OpenDeviceID()　デバイスのオープン(ボードＩＤ参照)

《機能》

渡した情報を持つＤＩＯ５４８ボードをオープンし，他と識別するためのデバイスＩＤを取得します．
以降このデバイスＩＤは，このＤＩＯ５４８にアクセスするためのハンドルとなります．
ＤＩＯ５８０ボードは“ボードＩＤ無視”のオープン関数を使用します．

《書式》

```
DWORD hdio580_OpenDeviceID( PHPCDEVINFO pHpcDevInfo );
```

《引数》

◆ 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》　デバイスＩＤ値

INVALID_HANDLE_VALUE（＝－１）：オープン失敗
「４．３ デバイスドライバの異常報告」を参照して下さい．
上記以外：オープン成功
●上位ワード：ボ ー ドＩＤ（ボード上のジャンパ設定値：０～１５）
●下位ワード：デバイスＩＤ（オープン順の番号　　：１～　）

【ヒント】　上位ワードの"ボードＩＤ"をデバイスＩＤとする事ができます．
この場合には、下位ワードを強制的に"０"とします．
[hDevID &= 0xffff0000;]

《呼び出し例》

パソコンに ＤＩＯ５ｘｘ が３枚装着されていることを想定します．
デバイス情報格納エリアとして HPCDEVINFO 型の配列 HpcDevInfo[3]を準備し、この中には既に
hdio580_GetDeviceInfo 関数により全ボードのデバイス情報が入っているものとします。
次に全てのデバイス（ＤＩＯ５ｘｘ）をオープンする場合を示します．

```
DWORD      hDevID[3];              /* デバイス ID 取得エリア */

hDevID[0] = hdio580_OpenDeviceID( &HpcDevInfo[0] );  /* １枚目ｵｰﾌﾟﾝ */
hDevID[1] = hdio580_OpenDeviceID( &HpcDevInfo[1] );  /* ２枚目ｵｰﾌﾟﾝ */
hDevID[2] = hdio580_OpenDeviceID( &HpcDevInfo[2] );  /* ３枚目ｵｰﾌﾟﾝ */
```

《ご注意》

１．ボード上のディップスイッチ
　　ＤＩＯ５４８・・ボードアドレス設定は０～１５　　（スイッチ設定値）
　　ＤＩＯ５８０・・ボードアドレス状態は０ｘ８００ｆ（固定）
ＤＩＯ５４８ボードを２枚以上使用する場合、同一のボードアドレス（０～１５）となっている時，
２枚目のボードオープン処理で異常終了が報告されます。（オープン済み）
このような場合には、ボードアドレスをオープン条件としない設定を行います．

```
HpcDevInfo[0].dwBoardID = INVALID_BOARD_ID;
HpcDevInfo[1].dwBoardID = INVALID_BOARD_ID;
```

２．オープン条件の優先順位（①が最優先）
　① dwBoardID ・・・・・・・・ボードＩＤ
　② nBusNumber,nDevNumber ・・バス番号･デバイス番号
　③ dwIoPortAdrs,dwIrqNo・・・I/O ポートアドレス･IRQ 番号

３．デバイスＩＤの内容
　　上位ワード・・ボードＩＤ（ボードＩＤをつけてオープンした場合）
　　下位ワード・・オープン順の番号（１～．．．）

（４）　hdio580_OpenDevice()　　デバイスのオープン(ボードＩＤ無視)

《機能》

渡した情報を持つＤＩＯ５ｘｘボードをオープンし，他と識別するためのデバイスＩＤを取得します．
以降このデバイスＩＤは，このＤＩＯ５ｘｘにアクセスするためのハンドルとなります．

《書式》

```
DWORD hdio580_OpenDevice( PHPCDEVINFO pHpcDevInfo );
```

《引数》

◆ 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》　　デバイスＩＤ値
INVALID_HANDLE_VALUE（＝-1）：オープン失敗
　　「４．３ デバイスドライバの異常報告」を参照して下さい．
上記以外：オープン成功
　●上位ワード：常に０
　●下位ワード：デバイスＩＤ（オープン順の番号：１～　）

《呼び出し例》

パソコンに ＤＩＯ５ｘｘ が３枚装着されていることを想定します．
デバイス情報格納エリアとして HPCDEVINFO 型の配列 HpcDevInfo[3]を準備し，この中には既に hdio580_GetDeviceInfo 関数により全ボードのデバイス情報が入っているものとします．
次に全てのデバイス（ＤＩＯ５ｘｘ）をオープンする場合を示します．

```
DWORD       hDevID[3];          /* デバイス ID 取得エリア */

hDevID[0] = hdio580_OpenDevice( &HpcDevInfo[0] );  /* １枚目ｵｰﾌﾟﾝ */
hDevID[1] = hdio580_OpenDevice( &HpcDevInfo[1] );  /* ２枚目ｵｰﾌﾟﾝ */
hDevID[2] = hdio580_OpenDevice( &HpcDevInfo[2] );  /* ３枚目ｵｰﾌﾟﾝ */
```

《ご注意》

１．オープン条件の優先順位（①が最優先）

① nBusNumber,nDevNumber ・・バス番号･デバイス番号
② dwIoPortAdrs,dwIrqNo・・・I/O ポートアドレス･IRQ 番号

（5） hdio580_CloseDevice()　　デバイスのクローズ

《機能》

渡したデバイスＩＤで指定されたＤＩＯ５ｘｘ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 hdio580_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

１：成功
０：失敗　・・「４．３ デバイスドライバの異常報告」を参照して下さい．

《呼び出し例》

```
short               ret;            /* 関数の戻り値 */

ret = hdio580_CloseDevice( hDevID );   /* デバイス ID */
```

（6） hdio580_ioswRead()　　入出力数切替ポート読込

《機能》

デバイスＩＤで指定されたＤＩＯ５４８の，入出力数設定スイッチ（ジャンパ設定）内容を読込み、指定エリアに格納します．

これにより，ボード設定状態の確認が行えます．

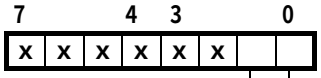
《書式》

```
short hdio580_ioswRead( DWORD hDevID, BYTE*  swin);
```

《引数》

◆ DWORD　hDevID・・・対象デバイスのデバイスＩＤ

◆ BYTE*　swin　・・・読込んだ設定データが格納される１バイトエリアのアドレス



ポート読込　（　x：ビットの値は不定です．）

予備スイッチ

値	入出力設定	入力ポート	出力ポート
0	24in/24out	1～3	1～3
1	32in/16out	1～4	1～2

※ボード出荷状態

ジャンパ有：入力値＝０（ 24in/24out ）

《戻り値》　処理結果

１：成功

０：失敗　・・「４．３ デバイスドライバの異常報告」を参照して下さい．

《呼び出し例》

```
short    ret;              //関数の戻り値
BYTE     swin;             //読込結果の格納エリア

ret = hdio580_ioswRead( hDevID,swin );     //デバイスＩＤ
```

（7）

hdio580_inpReadB()	入力ポートバイト読込
hdio580_inpReadW()	入力ポートワード読込

《機能》

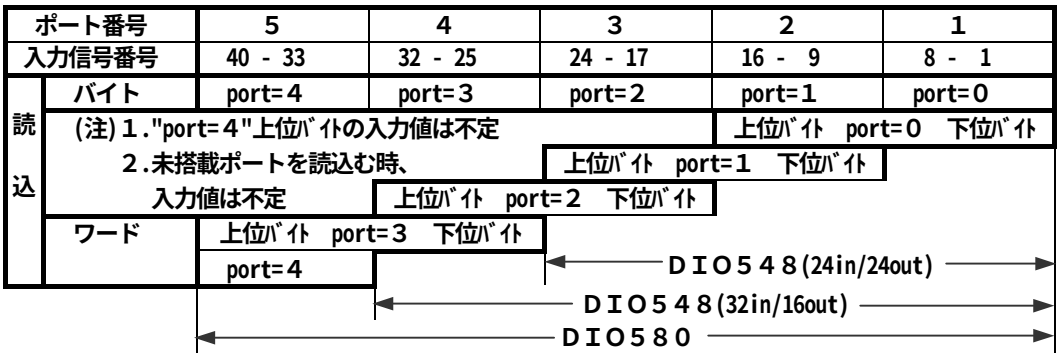
デバイスＩＤで指定されたＤＩＯ５ｘｘの、port で指定された入力ポートから
入力ポートバイト読込・・・１バイトを読込み，指定したエリアに格納します．
入力ポートワード読込・・・２バイトを読込み，指定したエリアに格納します．

《書式》

```
short hdio580_inpReadB( DWORD hDevID, WORD port, BYTE*  bytin);
short hdio580_inpReadW( DWORD hDevID, WORD port, WORD*  wrdin);
```

《引数》

◆ DWORD hDevID・・・対象デバイスのデバイス ID
◆ WORD port ・・・ポート番号指定　０～４　（ポート番号－１）



◆ BYTE* bytin・・・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* wrdin・・・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

１：成功
０：失敗　・・「４．３ デバイスドライバの異常報告」を参照して下さい．

《呼び出し例》

```
short       ret;                //関数の戻り値
BYTE        bytin;              //格納先(ﾊﾞｲﾄ)
WORD        wrdin;              //格納先(ﾜｰﾄﾞ)

ret = hdio580_inpReadB( hDevID , 0, &bytin );   //ﾎﾟｰﾄ１をバイト入力
ret = hdio580_inpReadW( hDevID , 0, &wrdin );   //ﾎﾟｰﾄ１をワード入力
```

(8)

hdio580_outpWriteB()	出力ポートバイト書込
hdio580_outpWriteW()	出力ポートワード書込
hdio580_outpReadB()	出力ポートバイト読込
hdio580_outpReadW()	出力ポートワード読込

《機能》

デバイスＩＤで指定されたＤＩＯ５ｘｘの、port で指定された出力ポートへ
出力ポートバイト書込・・・指定１バイトを書込みます.
出力ポートワード書込・・・指定２バイトを書込みます.
デバイスＩＤで指定されたＤＩＯ５ｘｘの、port で指定された出力ポートから
出力ポートバイト読込・・・１バイトを読込み、
出力ポートワード読込・・・２バイトを読込み、
指定したエリアに格納します. これにより出力データの確認が行えます.

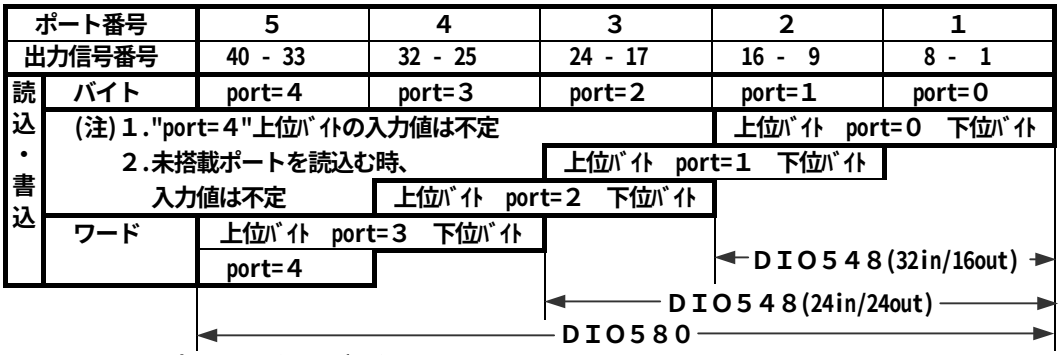
《書式》

```
short hdio580_outpWriteB( DWORD hDevID,WORD port,BYTE  bytdt);
short hdio580_outpWriteW( DWORD hDevID,WORD port,WORD  wrddt);

short hdio580_outpReadB ( DWORD hDevID,WORD port,BYTE* bytin);
short hdio580_outpReadW ( DWORD hDevID,WORD port,WORD* wrdin);
```

《引数》

◆ DWORD hDevID・・・対象デバイスのデバイスＩＤ
◆ WORD port ・・・ポート番号指定 ０～４ （ポート番号－１）



◆ BYTE bytdt・・・書込む１バイトデータ
◆ WORD wrddt・・・書込む２バイトデータ
◆ BYTE* bytin・・・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* wrdin・・・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

１：成功
０：失敗 ・・「４．３ デバイスドライバの異常報告」を参照して下さい.

《呼び出し例》

```
short     ret;              //関数の戻り値
BYTE      bytin;            //格納先(ﾊﾞｲﾄ)

ret = hdio580_outpReadB( hDevID , 0, &bytin );        //ﾎﾟｰﾄ１をバイト入力
if(ret) {
      bytin &= 0x0f;                                  //ﾎﾟｰﾄ１上位４ﾋﾞｯﾄoff
      ret = hdio580_outpWriteB( hDevID ,0,bytin );    //ﾎﾟｰﾄ１へバイト出力
}
```

（9）

hdio580_intFilterWrite()	割込入力信号 フィルタ設定
hdio580_intFilterRead()	割込入力信号 フィルタ設定確認

《機能》

デバイスＩＤで指定されたＤＩＯ５ｘｘの割込入力信号に対して
書込・・フィルタ機能の“使用／不使用”を設定します.
読込・・フィルタ機能設定状況を読込み、指定した１バイトのエリアに格納します.

《書式》

```
short hdio580_intFilterWrite( DWORD hDevID, BYTE  fil_data );
short hdio580_intFilterRead ( DWORD hDevID, BYTE* fil_data );
```

《引数》

◆ DWORD hDevID ・・ 対象デバイスのデバイスＩＤ
◆ BYTE fil_data ・・ 割込入力信号フィルタ設定ポートへの１バイトデータ
◆ BYTE* fil_data ・・ 読込んだデータが格納される１バイトエリアのアドレス

ボード区分	7	6	5	4	3	2	1	0	
ＤＩＯ５８０	0	0	0	0	IN4	IN3	IN2	IN1	・・書込
	x	x	x	x	IN4	IN3	IN2	IN1	・・読込（ｘは不定）
ＤＩＯ５４８	0	0	0	0	0	IN3	IN2	IN1	・・書込
	x	x	x	x	x	IN3	IN2	IN1	・・読込（ｘは不定）

各ビットの値・・０：フィルタ不使用
・・１：フィルタ使用（１５０us～２００us）

《戻り値》 処理結果

１：成功
０：失敗 ・・「４．３ デバイスドライバの異常報告」を参照して下さい.

《呼び出し例》

```
short     ret;              /* 関数の戻り値 */
BYTE      fil_data;

ret = hdio580_intFilterWrite(
                  hDevID,          /* デバイスＩＤ */
                  0x03 );          /* IN2,IN1:フィルタ設定 */
ret = hdio580_intFilterRead(
                  hDevID,          /* デバイスＩＤ */
                  &fil_data );     /* 格納先のアドレス */
```

（１０）

hdio580_intEdgeWrite()	割込入力信号 エッジ選択
hdio580_intEdgeRead()	割込入力信号 エッジ選択確認

《機能》

デバイスＩＤで指定されたＤＩＯ５ｘｘの，割込入力信号に対して
書込・・割込発生とする入力信号の変化（割込条件）を設定します．
読込・・割込条件を読込み、指定した１バイトのエリアに格納します．

《書式》

```
short hdio580_intEdgeWrite( DWORD hDevID, BYTE  edge_data );
short hdio580_intEdgeRead ( DWORD hDevID, BYTE* edge_data );
```

《引数》

◆ DWORD hDevID ・・ 対象デバイスのデバイスＩＤ
◆ BYTE edge_data ・・ 割込発生とする入力信号の変化（割込条件）設定ポートへの１バイトデータ
◆ BYTE* edge_data ・・ 読込んだデータが格納される１バイトエリアのアドレス

ボード区分	7	6	5	4	3	2	1	0	
ＤＩＯ５８０	0	0	0	0	IN4	IN3	IN2	IN1	・・書込
	x	x	x	x	IN4	IN3	IN2	IN1	・・読込（ｘは不定）
ＤＩＯ５４８	0	0	0	0	0	IN3	IN2	IN1	・・書込
	x	x	x	x	x	IN3	IN2	IN1	・・読込（ｘは不定）

各ビットの値・・０：入力信号０→１への変化
・・１：入力信号１→０への変化

《戻り値》 処理結果
１：成功
０：失敗 ・・「４．３ デバイスドライバの異常報告」を参照して下さい．

《呼び出し例》

```
short       ret;            /* 関数の戻り値 */
BYTE        edge_data;

ret = hdio580_intEdgeWrite(
                hDevID,         /* デバイス ID */
                0x01 );         /* IN1:１→０ */
ret = hdio580_intEdgeRead(
                hDevID,         /* デバイス ID */
                &edge_data );   /* 格納先のアドレス */
```

（１１）

hdio580_intIdtSelWrite()	割込入力信号 要因選択
hdio580_intIdtSelRead()	割込入力信号 要因選択確認

《機能》

デバイスＩＤで指定されたＤＩＯ５ｘｘの，割込入力信号について

書込・・割込入力信号を設定します．

読込・・設定された割込入力信号を読込み、指定した１バイトのエリアに格納します．

《書式》

```
short hdio580_intIdtSelWrite( DWORD hDevID, BYTE  sel_data );
short hdio580_intIdtSelRead ( DWORD hDevID, BYTE* sel_data );
```

《引数》

◆ DWORD hDevID ・・ 対象デバイスのデバイスＩＤ

◆ BYTE sel_data ・・ 割込入力信号設定データ

◆ BYTE* sel_data ・・ 読込んだデータが格納される１バイトエリアのアドレス

ボード区分	7	6	5	4	3	2	1	0	
ＤＩＯ５８０	0	0	0	0	IN4	IN3	IN2	IN1	・・書込
	x	x	x	x	IN4	IN3	IN2	IN1	・・読込（ｘは不定）
ＤＩＯ５４８	0	0	0	0	0	IN3	IN2	IN1	・・書込
	x	x	x	x	x	IN3	IN2	IN1	・・読込（ｘは不定）

各ビットの値・・０：割込入力信号とはしない

・・１：割込入力信号に指定

《戻り値》 処理結果

１：成功

０：失敗 ・・「４．３ デバイスドライバの異常報告」を参照して下さい。

《呼び出し例》

```
short       ret;            /* 関数の戻り値 */
BYTE        sel_data;

ret = hdio580_intIdtSelWrite(
                hDevID,         /* デバイスＩＤ */
                0x01 );         /* IN1:選択 */
ret = hdio580_intIdtSelRead(
                hDevID,         /* デバイスＩＤ */
                &sel_data );    /* 格納先のアドレス */
```

（１２）　hdio580_intIdtStsRead()　　割込入力信号 割込要因確認

《機能》

デバイスＩＤで指定されたＤＩＯ５ｘｘの，割込処理関数内で，割込入力信号を読込みます．

《書式》

```
short hdio580_intIdtStsRead( DWORD hDevID, BYTE* idt_data );
```

《引数》

◆ DWORD　hDevID　　・・　対象デバイスのデバイスＩＤ
◆ BYTE*　idt_data　・・　読込んだデータが格納される１バイトエリアのアドレス

ボード区分	7	6	5	4	3	2	1	0	
ＤＩＯ５８０	x	x	x	x	IN4	IN3	IN2	IN1	・・読込（ｘは不定）
ＤＩＯ５４８	x	x	x	x	x	IN3	IN2	IN1	・・読込（ｘは不定）

各ビットの値・・０：割込入力信号ではない
　　・・１：割込入力信号です．

《戻り値》　処理結果
１：成功
０：失敗　・・「４．３ デバイスドライバの異常報告」を参照して下さい。

《呼び出し例》

```
short       ret;            /* 関数の戻り値 */
BYTE        idt_data;

ret = hdio580_intIdtStsRead(
                hDevID,             /* デバイスＩＤ */
                &idt_data );        /* 格納先のアドレス */
```

（１３）

hdio580_intEblWrite()　　ボード割込設定
hdio580_intEblRead()　　ボード割込設定確認

《機能》

デバイスＩＤで指定されたＤＩＯ５ｘｘの，割込入力信号の各種設定を行った上で
書込・・ボードからＣＰＵへの割込信号の出力を設定します．
読込・・ボードからＣＰＵへの割込信号の出力設定を読込み、指定した１バイトのエリアに格納します．

《書式》

```
short hdio580_intEblWrite( DWORD hDevID, BYTE  ebl_data );
short hdio580_intEblRead ( DWORD hDevID, BYTE* ebl_data );
```

《引数》

◆ DWORD hDevID　・・ 対象デバイスのデバイスＩＤ
◆ BYTE ebl_data　・・ ボード割込設定データ
◆ BYTE* ebl_data　・・ 読込んだデータが格納される１バイトエリアのアドレス

ボード区分	7	6	5	4	3	2	1	0	
ＤＩＯ５８０	0	0	0	0	0	0	0	EBL	・・書込
ＤＩＯ５４８	x	x	x	x	x	x	x	EBL	・・読込（ｘは不定）

各ビットの値・・０：ボードからＣＰＵへ割込信号を出力しない
・・１：ボードからＣＰＵへ割込信号を出力する

《戻り値》　処理結果

１：成功
０：失敗　・・「４．３ デバイスドライバの異常報告」を参照して下さい。

《呼び出し例》

```
short        ret;             /* 関数の戻り値 */
BYTE         ebl_data;

ret = hdio580_intEblWrite(
                  hDevID,         /* デバイスＩＤ */
                  0x01 );         /* ボード→ＣＰＵ割込信号出力を設定 */
ret = hdio580_intEblWrite(
                  hDevID,         /* デバイスＩＤ */
                  0x00 );         /* ボード→ＣＰＵ割込信号出力をキャンセル */
ret = hdio580_intEblRead(
                  hDevID,         /* デバイスＩＤ */
                  &ebl_data );    /* 格納先のアドレス */
```

（１４）　hdio580_SetIntCall()　　割込処理関数の登録・消去

《機能》
デバイスＩＤで指定されたＤＩＯ５ｘｘボードからの割込が発生した場合の処理関数を指定します.

《書式》
```
short hdio580_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"（消去）

《戻り値》　　処理結果
１：成功
０：失敗　・・「４．３ デバイスドライバの異常報告」を参照して下さい。

《呼び出し例》
```
short          ret;            /* 関数の戻り値 */
PINTPROC       int_module;

ret = hdio580_SetIntCall(
                hDevID,            /* デバイス ID */
                int_module );      /* 割込処理モジュール */
```

《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) 初期化時の指令関数の順序
① 割込以外の全てのボード初期化を実行する。
② hdio580_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 ボード単位で処理を行います。（全搭載軸の一括処理）
② モジュール内ではＣＰＵに対して ”割込許可 ” としないで下さい。
③ 割込要因は"hdio580_intInpStsReset()"関数で読込みます。
④ 上記関数起動で割込要因がない場合もあります。
⑤ 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 hdio580_SetIntCall()　関数で割込処理モジュールを消去する。

６．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、電源遮断後ボードを装着し、パソコンの電源再投入を行った後、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可能です。

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←→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。

６．１　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。

```
ｃｌｋ．ｂａｔ・・・・・・・・・・・・MS-C(V6.0)用実行ファイル作成用バッチファイル
├──ｓｍｐ５８０．ｃ ・・・・・・・・ソースプログラム
│　　└──  ｈｉｐｐｄ５８０．ｈ ・・ヘッダーファイル　"smp580.c"で"#include"
└──ｌｉｐｐｄ５８０．ｌｉｂ ・・・・ライブラリファイル（ｌ：ラージモデル）
```

６．２　サンプルプログラムの起動

実行ファイル"ｓｍｐ５８０．ｅｘｅ"を起動する時、下記の画面が表示されます。

```
 *** HPCI-DIO580,DIO548 : Device Driver Sample [ Board = 2 ] ***
    ----- lidio580.lib Ver3.00  &  hidio580.drv Ver3.00 -----

                Bus= 0  Dev=11  I/O=0x7000  IRQ= 5   CTL= 1   BID=#15 Open?=I
                Bus= 0  Dev=12  I/O=0x7800  IRQ=11   CTL= 2   BID= 10 Open?=_
```

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・・・・・・・その他

```
 *** HPCI-DIO580,DIO548 : Device Driver Sample [ Board = 2 ] ***
    ----- lidio580.lib Ver3.00  &  hidio580.drv Ver3.00 -----

1.ID=00001 IO=5/5 Bus= 0  Dev=11  I/O=0x7000  IRQ= 5   CTL= 1   BID=#15 Open?=I
                  Bus= 0  Dev=12  I/O=0x7800  IRQ=11   CTL= 2   BID= 10 Open?=



 1.        40 ---- 33   32 ---- 25   24 ---- 17   16 ---- 9     8 ---- 1
  input  5: oooooooo  4: oooooooo  3: oooooooo  2: oooooooo  1: oooooooo

  out/r  5: oooooooo  4: oooooooo  3: oooooooo  2: oooooooo  1: oooooooo
  out/w  5: oooooooo  4: oooooooo  3: oooooooo  2: oooooooo  1: oooooooo

  Filter: oooo  Edge: oooo  DataSel: 4321
  IntBit:     oooo  IN1=      0  IN2=      0  IN3=      0  IN4=      0

 [ 0:End  1:Output  2:IntMode  3:BitOut ] = _
```

図６．２－１　サンプルプログラムの起動画面（割込使用）

（1）タイトル・認識ボード枚数とソフトのバージョン番号

（2）ボード情報

```
1.ID=00001 IO=5/5 Bus= 0  Dev=11  I/O=0x7000  IRQ= 5   CTL= 1   BID=#15 Open?=I
                  Bus= 0  Dev=12  I/O=0x7800  IRQ=11   CTL= 2   BID= 10 Open?=
```

（a）選択ボードの表示

ＩＤ：デバイスオープン時の“デバイスＩＤ値”です．（１６進数表示）

ＩＯ：入力ポート数と出力ポート数を表します．

5/5・・・ＤＩＯ５８０［入力・出力共に5ポート，各４０点］

3/3・・・ＤＩＯ５４８［入力・出力共に3ポート，各２４点］

4/2・・・ＤＩＯ５４８［入力4ポート３２点，出力2ポート１６点］

（b）全ボード共通の表示

以下の表示は，取得したデバイス情報です．

Ｂｕｓ・・バス番号

Ｄｅｖ・・デバイス番号

Ｉ／Ｏ・・ボードに割り振られたアドレス

ＩＲＱ・・割込番号

ＣＴＬ・・管理番号

ＢＩＤ・・ボードＩＤ（ＤＩＯ５８０：０ｘ８００ｆを“＃１５”と表示）

（3）ボード状態

```
1.        40 ---- 33   32 ---- 25   24 ---- 17   16 ---- 9     8 ---- 1
 input  5: oooooooo  4: oooooooo  3: oooooooo  2: oooooooo  1: oooooooo

 out/r  5: oooooooo  4: oooooooo  3: oooooooo  2: oooooooo  1: oooooooo
 out/w  5: oooooooo  4: oooooooo  3: oooooooo  2: oooooooo  1: oooooooo

 Filter: oooo  Edge: oooo  DataSel: 4321
 IntBit:     oooo  IN1=      0  IN2=      0  IN3=      0  IN4=      0
```

（a）入出力ポート

ｉｎｐｕｔ・・・・入力ポートの状態を表します．

ｏｕｔ／ｒ・・・・出力ポートの読込結果を表します・

ｏｕｔ／ｗ・・・・出力ポートへの出力状態です．

（b）割込使用時の表示

Ｆｉｌｔｅｒ・・・「割込入力信号 フィルタ設定」の状態

Ｅｄｇｅ・・・・・「割込入力信号 エッジ選択」の状態

ＤａｔａＳｅｌ・・「割込入力信号 要因選択」の状態

ＩｎｔＢｉｔ・・・ 最新の「割込入力信号 割込要因」

ＩＮ１～ＩＮ４・・ 各入力信号毎の割込発生回数

（4）動作指令の選択

```
[ 0:End  1:Output  2:IntMode  3:BitOut ] = _
```

キーボードから１文字入力で選択します．

'0'・・プログラム終了です．

'1'・・指定出力ポートへの一括データ出力を行います．

'2'・・割込使用時の，各種設定値の変更が可能です．

'3'・・ビット単位での出力ポートへの出力を行います．

６．３　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．
入力信号の状態は，常時読込が行われ，表示が更新されます．

（1）出力ポートへの出力操作
２種類の出力ポートへの出力操作があります．

（a）指定ポートへの一括出力
下図に示す操作となります．

```
[ 0:End  1:Output  2:IntMode  3:BitOut ] = 1

  Port No(1,2,..) = 1
  DO output[0x00] = 12_
```

```
out/r  5: oooooooo  4: oooooooo  3: oooooooo  2: oooooooo  1: ooolool0
out/w  5: oooooooo  4: oooooooo  3: oooooooo  2: oooooooo  1: ooolool0
```

（b）ビット単位の出力ポート操作

```
1.        40 ---- 33   32 ---- 25   24 ---- 17   16 ---- 9     8 ---- 1
 input  5: oooooooo  4: oooooooo  3: oooooooo  2: oooooooo  1: oooooooo

 out/r  5: oooooooo  4: oooooooo  3: ooooolOO  2: lOOOllll  1: OllOllOl
 out/w  5: oooooooo  4: oooooooo  3: ooooolOO  2: lOOOllll  1: OllOllOl

 Filter: oooo  Edge: oooo  DataSel: 4321
 IntBit:     oooo  IN1=      0  IN2=      0  IN3=      0  IN4=      0

[ 0:End  1:Output  2:IntMode  3:BitOut ] = 3
```

ビット単位の出力ポート操作では，下記のキーを使って，カーソル位置の出力ビットの反転出力となります．

“ＥＳＣ“キー・・・この操作をキャンセルします．
“　←　“キー・・・カーソル位置を左側に移動します．（ﾛｰﾃｰｼｮﾝ）
“　→　“キー・・・カーソル位置を右側に移動します．（ﾛｰﾃｰｼｮﾝ）
“ ｽﾍﾟｰｽ“バー・・・カーソル位置の出力を反転します．

（2）割込使用時
割込入力信号（ＤＩＯ５８０：ＩＮ1～ＩＮ４，ＤＩＯ５４８：ＩＮ1～ＩＮ３）の各種選択設定を行います．
３種類（Filter,Edge,DataSel）の設定では，１６進数入力を行い，その結果は入力信号の番号［1～4］で表示されます．
最後の設定では，画面に表示された各信号の割込入力回数のクリアが指令できます．

```
 Filter: o321  Edge: o3o1  DataSel: 4321
 IntBit:     oooo  IN1=      0  IN2=      0  IN3=      0  IN4=      0

[ 0:End  1:Output  2:IntMode  3:BitOut ] = 2

  Filter [0x0f] = 7  Edge [0x00] = 5  DataSel [0x0f] = f
  Counter Clear(Y/y) ? = y
```