

HPCI-DIO580 シリーズ 添付ソフトウェアマニュアル

Windows版デバイスドライバ
Windows版ドライバ/F用ライブラリ



株式会社ハイバーテック

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

HPCI-DIO580 シリーズのボードは

HPCI-DIO580A
HPCI-DIO580AG
HPCI-DIO580
HPCI-DIO548
HPCI-DI580
HPCI-DO580

です。

Windows 版デバイスドライバ の種別は

Windows 95,98 では	hidio580.vxd
Windows NT4.0 では	hidio580.sys
Windows 2000 では	hd580w2k.sys
Windows XP では	hd580wxp.sys
Windows Vista 32 ビットでは	hd580wdm.sys
Windows Vista 64 ビットでは	hd580x64.sys
Windows 7 32 ビットでは	hd580wdm.sys
Windows 7 64 ビットでは	hd580x64.sys です。

Windows 32 ビット 版ドライバ\F用ライブラリ として

hidio580.dll を共通で使用します。

Windows 64 ビット 版 32 ビットアプリケーション用ドライバ\F用ライブラリ として

hidio580.dll(32 ビットアプリケーション用)を使用します。

Windows 64 ビット 版 64 ビットアプリケーション用ドライバ\F用ライブラリ として

hidio580.dll(64 ビットアプリケーション用)を使用します。

本書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下さい。

Windows95, Windows98, WindowsNT 4.0, Windows2000, WindowsXP Home Edition, WindowsXP Professional, WindowsVista, VisualC++, Visual Basic 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 ハイパーテック
東京都江東区新大橋 1-8-11
三井生命新大橋ビル 6F
TEL 03-3846-3801
FAX 03-3846-3773
sales@hivertec.co.jp

6.0 版 2011 年 6 月 30 日発行
不許複製・転載

目 次

1.	はじめに.....	1
1.1	安全にお使い頂くために.....	1
1.2	保証範囲.....	2
1.3	免責事項.....	2
1.4	対象ユーザー.....	2
1.5	適合OS.....	3
2.	添付ソフトウェアの構成.....	4
2.1	ソフトウェア構成.....	4
2.1.1	32ビット版OS.....	4
2.1.2	64ビット版OS.....	5
2.2	添付ソフトウェアの内容.....	5
3.	デバイスドライバのインストール.....	9
3.1	Windows 7(32ビット), Windows Vista(32ビット)へのインストール.....	9
3.2	Windows 7(64ビット), Windows Vista(64ビット)へのインストール.....	9
3.3	Windows XPへのインストール.....	9
3.4	Windows 2000 へのインストール.....	9
3.5	Windows NT4.0 へのインストール.....	10
3.5.1	デバイスドライバのインストール.....	10
3.5.2	デバイスの開始と停止.....	10
3.6	Windows 9Xへのインストール.....	10
3.7	アンインストール.....	10
3.7.1	Windows XP, Windows 2000, Windows NT 4.0, Windows 9Xの場合.....	10
3.7.2	Windows 7, Windows Vistaの場合.....	10
4.	ボードを複数枚利用する場合.....	11
4.1	ボードのデバイス番号の確認.....	11
4.2	デバイス番号の確認方法.....	11
4.3	DIO548 ボード, DIO580AボードでのボードIDの使用.....	11
5.	ドライバI/F用DLLの使用方法.....	12
5.1	概 要.....	12
5.2	関数一覧.....	12
5.3	準 備.....	13
6.	制御概念.....	14
6.1	ボード(デバイス)認識用のデータ構造体.....	14
6.2	ボードアクセスの準備手順.....	16
6.3	デバイスドライバの異常報告.....	17
7.	関数詳細.....	18
8.	DIO580, DIO548 用サンプルプログラム.....	41
8.1	サンプルプログラムの操作.....	41
8.1.1	ボード(デバイス)の選択.....	42
8.1.2	ボード上の操作と表示.....	42
8.2	サンプルプログラムの変更について.....	43
8.2.1	プロジェクトファイル(ソリューションファイル).....	43
8.2.2	サンプルプログラムを構成する関数.....	44
9.	DI580 用サンプルプログラム.....	45
9.1	サンプルプログラムの操作.....	45
9.1.1	ボード(デバイス)の選択.....	46
9.1.2	ボード上の操作と表示.....	46
9.2	サンプルプログラムの変更について.....	47
9.2.1	プロジェクトファイル(ソリューションファイル).....	47
9.2.2	サンプルプログラムを構成する関数.....	47
10.	DO580 用サンプルプログラム.....	48
10.1	サンプルプログラムの操作.....	48
10.1.1	ボード(デバイス)の選択.....	49

10.1.2	ボード上の操作と表示.....	49
10.2	サンプルプログラムの変更について.....	50
10.2.1	プロジェクトファイル(ソリューションファイル).....	50
10.2.2	サンプルプログラムを構成する関数.....	50

図 表 目 次

図 2.1-1	32ビット版OS 添付ソフトウェア関連図.....	4
図 2.1-2	64ビット版OS 添付ソフトウェア関連図.....	5
図 4.1-1	ボードを複数枚使用する場合.....	11
表 4.3-1	HPCI-DIO548 ボードIDの設定.....	11
図 4.3-1	HPCI-DIO580AボードIDロータリータイプSW.....	11
表 5.2-1	関数一覧.....	12
表 6.3-1	異常内容一覧.....	17

1. 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、HPCI-DIO580 シリーズボード(DIO580A,DIO580,DIO548,D1580,DO580)の添付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。

この添付ソフトウェアは



Windows95/98	(以降 Win9X と記します),
WindowsNT4. 0	(以降 WinNT と記します),
Windows2000	(以降 Win2K と記します),
Windows XP Home Edition/Professional	(以降 WinXP と記します),
Windows Vista の各エディション	(以降 WinVista と記します)
Windows 7 の各エディション	(以降 Win7 と記します)

において、**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。

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	
	<p>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。</p> <ul style="list-style-type: none">・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、Windows95/98、WindowsNT4.0、Windows2000、WindowsXP Home Edition、WindowsXP Professional、Windows Vista の各エディション、Windows 7 の各エディションにおいてボードの制御を行う為のソフトウェアです。 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. 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。 <ul style="list-style-type: none">・Microsoft Visual C++ 6.0・Microsoft Visual C#.NET 2003・Microsoft Visual Basic 6.0・Microsoft Visual Basic NET 2003

5. 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。

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、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あります。
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

2.1.1 32ビット版 OS

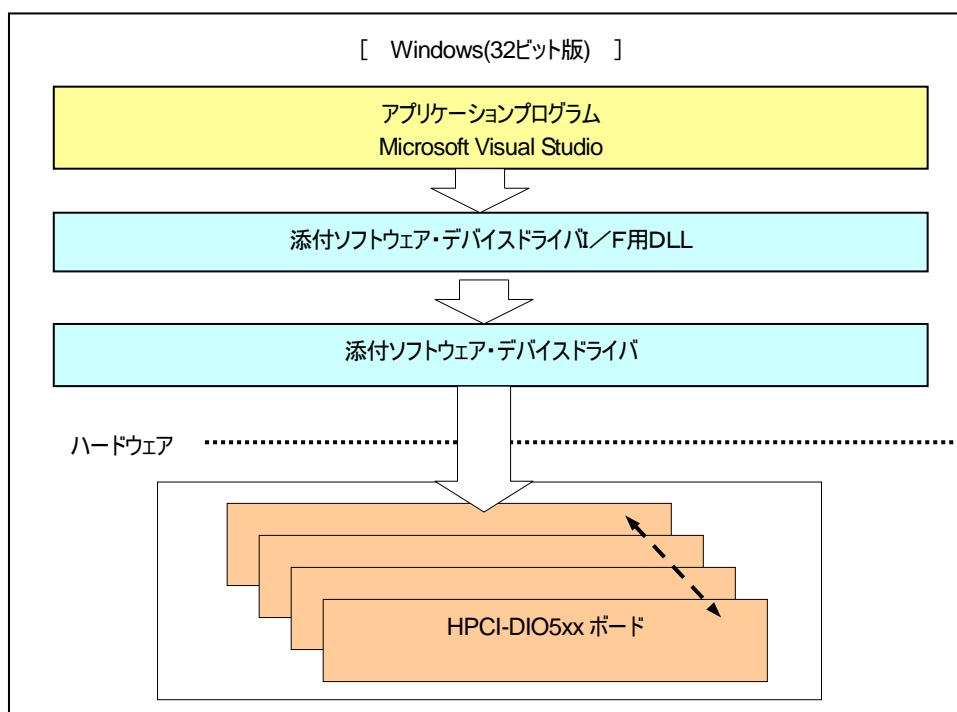


図 2.1-1 32ビット版 OS 添付ソフトウェア関連図

2.1.2 64ビット版 OS

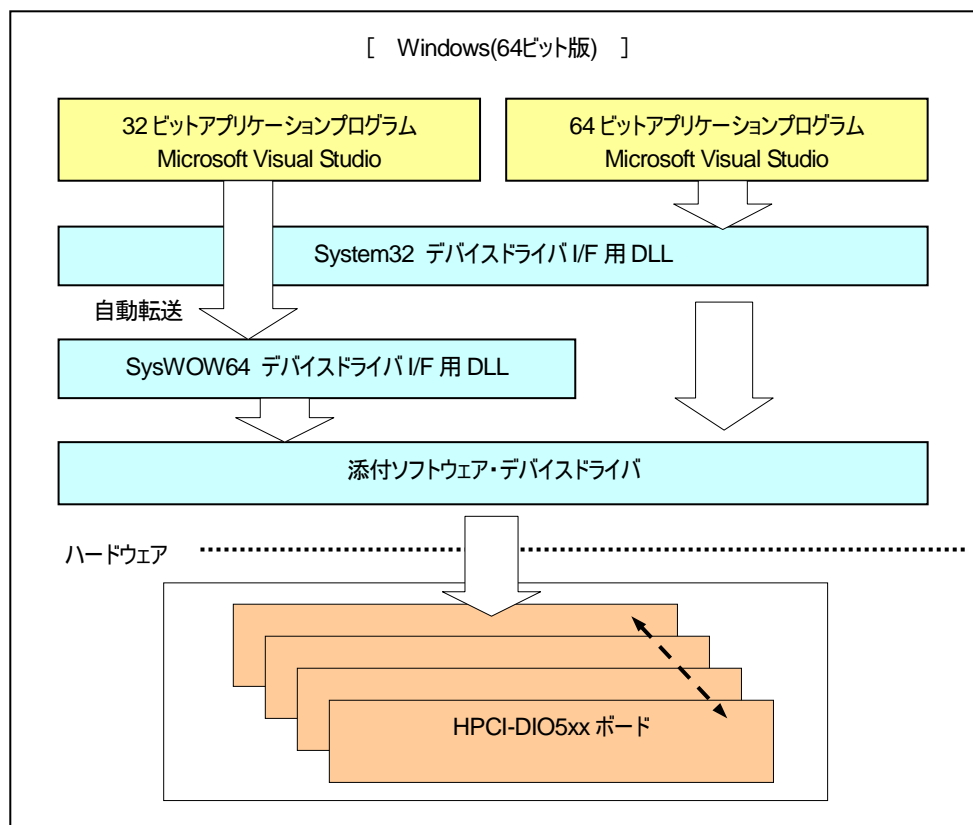


図 2.1-2 64ビット版 OS 添付ソフトウェア関連図

2.2 添付ソフトウェアの内容

CD:¥

—History.txt	… 更新履歴
—document	
—DIO548_U_1_23J.pdf	… HPCI-DIO548 ユーザーズマニュアル
—DIO580AG_U_1_00J.pdf	… HPCI-DIO580AG ユーザーズマニュアル
—DIO580A_U_1_01J.pdf	… HPCI-DIO580A ユーザーズマニュアル
—DIO580_U_1_22J.pdf	… HPCI-DIO580 ユーザーズマニュアル
—DI580_U_1_00J.pdf	… HPCI-DI580 ユーザーズマニュアル
—DO580_U_1_00J.pdf	… HPCI-DO580 ユーザーズマニュアル
—DIO580_W_6_00J.pdf	… DIO580 シリーズ添付ソフトウェアマニュアル

(次のページへ続く)

(前のページからの続き)

—win9x Windows9X
—HIDIO580.INF インストール情報ファイル
—HIDIO580.VXD デバイスドライバ
—winnt WindowsNT4.0
—D580INST.BAT インストール用バッチファイル
—D580INST.INF インストール情報ファイル
—HIDIO580.SYS デバイスドライバ
—win2k Windows2000
—HD580W2K.INF インストール情報ファイル
—HD580W2K.SYS デバイスドライバ
—winxp WindowsXP
—hd580wpx.cat セキュリティカタログファイル
—Hd580wpx.inf インストール情報ファイル
—hd580wpx.sys デバイスドライバ
—HIDIO580.DLL ドライバ I/F 用 DLL(32 ビット版)
—win7_x86	
—dpinst.exe ドライバパッケージインストーラ
—dpinst.xml ドライバパッケージインストーラ用 XML ファイル
—hd580wdm.cat セキュリティカタログファイル
—hd580wdm.inf インストール情報ファイル
—hd580wdm.sys デバイスドライバ
—HIDIO580.DLL ドライバ I/F 用 DLL(32 ビット版)
—win7_x64	
—dpinst.exe ドライバパッケージインストーラ
—dpinst.xml ドライバパッケージインストーラ用 XML ファイル
—hd580x64.cat セキュリティカタログファイル
—hd580x64.inf インストール情報ファイル
—hd580x64.sys デバイスドライバ
—HIDIO580.DLL ドライバ I/F 用 DLL(32 ビット版)
—hidio580.x64 ドライバ I/F 用 DLL(64 ビット版)
—include アプリケーション構築用
—vb6 VB 6.0 アプリケーション構築用
—hidio580.bas DLL 関数定義標準モジュール
—vb_net VB.NET アプリケーション構築用
—hidio580.vb DLL 関数定義ファイル
—vcs VC#アプリケーション構築用
—hidio580.cs DLL 関数定義ファイル
—vc_x86 VC(32 ビット)アプリケーション構築用
—Hidio580.h DLL 関数定義ヘッダーファイル
—hidio580.lib DLL インポートライブラリ(32 ビット版)
—vc_x64 VC(64 ビット)アプリケーション構築用
—Hidio580.h DLL 関数定義ヘッダーファイル
—HIDIO580.LIB DLL インポートライブラリ(64 ビット版)

(次のページへ続く)

(前のページからの続き)

└─sample サンプルプログラム
├─vb6 Visual Basic 6.0
│ └─SPD58002.vbw サンプル プロジェクトワークスペース
│ └─SPD58002.VBP サンプル プロジェクト
│ └─hidio580.bas DLL 関数定義標準モジュール
│ └─spd58002.bas サンプル標準モジュール
│ └─spd58002.frm サンプル フォームモジュール
│ └─SPD58002.EXE サンプル実行ファイル
├─vb_net Visual C#.NET 2003
│ └─spd58003.sln サンプル ソリューションファイル
│ └─spd58003.vbproj サンプル プロジェクトファイル
│ └─spd58003.vbproj.user サンプル ユーザーオプションファイル
│ └─AssemblyInfo.vb アセンブリ情報ファイル
│ └─hidio580.vb DLL 関数定義ファイル
│ └─spd58003.vb サンプルソースファイル
│ └─spd58003Module.vb サンプル標準モジュール
│ └─spd58003.resx メインフォームモジュールリソーステンプレート
│ └─obj	
│ └─Release	
│ └─spd58003.exe サンプル実行ファイル
├─vc Visual C6.0
│ └─spd58000	
│ │ └─SPD58000.DSP サンプル プロジェクト
│ │ └─SPD58000.DSW サンプル プロジェクトワークスペース
│ │ └─SPD58000.OPT サンプル オプションファイル
│ │ └─BGREEN.BMP サンプル ビットマップ(緑)
│ │ └─BWHITE.BMP サンプル ビットマップ(白)
│ │ └─HIDIO580.xxx DLL 関数定義ヘッダーファイル, DLL インポートライブラリ(*** = h,lib)
│ │ └─SPD58000.C サンプル ソースファイル
│ │ └─SPD58000.H サンプル ヘッダー
│ │ └─spd58000.rc サンプル リソースファイル
│ │ └─resource.h サンプル リソースヘッダー
│ │ └─SPD58000.EXE サンプル実行ファイル
│ └─spdi5800	
│ │ └─SPDI5800.DSP サンプル プロジェクト
│ │ └─SPDI5800.DSW サンプル プロジェクトワークスペース
│ │ └─BGREEN.BMP サンプル ビットマップ(緑)
│ │ └─BWHITE.BMP サンプル ビットマップ(白)
│ │ └─HIDIO580.xxx DLL 関数定義ヘッダーファイル, DLL インポートライブラリ(*** = h,lib)
│ │ └─SPDI5800.C サンプル ソースファイル
│ │ └─SPDI5800.H サンプル ヘッダー
│ │ └─spdi5800.rc サンプル リソースファイル
│ │ └─resource.h サンプル リソースヘッダー
│ │ └─SPDI5800.EXE サンプル実行ファイル
└─spdo5800	
│ └─SPDO5800.DSP サンプル プロジェクト
│ └─SPDO5800.DSW サンプル プロジェクトワークスペース
│ └─BGREEN.BMP サンプル ビットマップ(緑)
│ └─BWHITE.BMP サンプル ビットマップ(白)
│ └─HIDIO580.xxx DLL 関数定義ヘッダーファイル, DLL インポートライブラリ(*** = h,lib)
│ └─SPDO5800.C サンプル ソースファイル
│ └─SPDO5800.H サンプル ヘッダー
│ └─spdo5800.rc サンプル リソースファイル
│ └─resource.h サンプル リソースヘッダー
└─SPDO5800.EXE サンプル実行ファイル

(次のページへ続く)

3. デバイスドライバのインストール

3.1 Windows 7(32ビット), Windows Vista(32ビット)へのインストール

- (1) HPCI-DIO5xx をパソコンの PCI バススロットに装着する前に、パソコンの電源を ON にして Windows を起動します。
- (2) **CD ドライブ:**¥win7_x86¥dpinst.exe を起動します。
- (3) "dpinst.exe"が起動されたら「次へ」をクリックして続行します。
- (4) インストーラー完了後、パソコンの電源を OFF し、HPCI-DIO5xx をパソコンの PCI バススロットに装着します。
- (5) パソコンの電源を ON にして Windows を起動します。
- (6) デバイスのインストールが自動的に行われ、再起動を促されますので再起動してインストールが完了します。

3.2 Windows 7(64ビット), Windows Vista(64ビット)へのインストール

- (1) HPCI-DIO5xx をパソコンの PCI バススロットに装着する前に、パソコンの電源を ON にして Windows を起動します。
- (2) **CD ドライブ:**¥win7_x64¥dpinst.exe を起動します。
- (3) "dpinst.exe"が起動されたら「次へ」をクリックして続行します。
- (4) インストーラー完了後、パソコンの電源を OFF し、HPCI-DIO5xx をパソコンの PCI バススロットに装着します。
- (5) パソコンの電源を ON にして Windows を起動します。
- (6) デバイスのインストールが自動的に行われ、再起動を促されますので再起動してインストールが完了します。

3.3 Windows XP へのインストール

- (1) パソコンの電源が OFF であることを確認した後、HPCI-DIO5xx ボードをパソコンの PCI バススロットに装着します。
- (2) パソコンの電源を ON にして Windows を起動します。
- (3) Windows が起動すると、HPCI-DIO5xx がシステムにより検出され、自動的に必要なデバイスドライバのインストール画面が表示されます。添付ディスクを CD ドライブに挿入します。
- (4) ソフトウェアを自動的にインストールする(推奨)をチェックします。
- (5) Hivertec HPCI-DIO580(WinXP)を選択します。
- (6) 「Windows ロゴテストに合格していません」との警告が表示される場合がありますが、**続行**を選択してインストールを続けてください。
- (7) 後はシステムの指示に従ってインストールを完了させます。

3.4 Windows 2000 へのインストール

- (1) パソコンの電源が OFF であることを確認した後、HPCI-DIO5xx ボードをパソコンの PCI バススロットに装着します。パソコンの電源を ON にして Windows を起動します。
- (2) Windows が起動すると、HPCI-DIO5xx がシステムにより検出され、自動的に必要なデバイスドライバのインストール画面が表示されます。
- (3) システムがインストール元ディレクトリの指定を要求してきたら、添付ディスクを CD ドライブに挿入します。
- (4) 検索場所の指定 のチェックボックスをチェックします。
- (5) **CD ドライブ:** ¥WIN2K を指定して下さい。
- (6) 後はシステムの指示に従ってインストールを完了させます。

3.5 Windows NT4.0 へのインストール

3.5.1 デバイスドライバのインストール

- (1) 添付 CD をディスクドライブに挿入します。
- (2) NT エクスプローラを起動し、CD ドライブ:¥WinNT¥d580inst.inf を選択します。
- (3) マウスの右ボタンをクリックします。表示されるポップアップメニューから「インストール」を選択します。この操作によりデバイスドライバのインストールが開始されます。
- (4) 後はシステムの指示に従ってインストールを完了させます。
コマンドプロンプトから、CD ドライブ:¥WinNT¥d580inst.bat を実行させても同様にインストールが開始されます。

3.5.2 デバイスの開始と停止

インストール完了後、デバイスドライバは「自動開始」に設定されており、WindowsNT 起動時に HPCI-DIO5xx ボードに対するサービスも開始されます。

何らかの理由により停止への変更が必要である場合は次の作業を行います。

- (1) コントロールパネルから「デバイス」アイコンをダブルクリックし、デバイス一覧の中から「Hivertec HPCI-DIO580」を選択します。
- (2) 「スタートアップ」ボタンを押すことにより「スタートアップの種類」ダイアログが表示されます。
- (3) 「停止」を選択し、「OK」を押します。

HPCI-DIO5xx デバイスを再開させる場合も、コントロールパネルの「デバイス」操作を行います。

「Hivertec HPCI-DIO580」を選択し、「開始」ボタンを押します。

3.6 Windows 9X へのインストール

- (1) パソコンの電源が OFF であることを確認した後、HPCI-DIO5xx ボードをパソコンの PCI バススロットに装着します。パソコンの電源を ON にして Windows を起動します。
- (2) Windows が起動すると、HPCI-DIO5xx がシステムにより検出され、自動的に必要なデバイスドライバのインストール画面が表示されます。
- (3) システムがインストール元ディレクトリの指定を要求してきたら、添付ディスクを CD ドライブに挿入します。
- (4) 検索場所の指定のチェックボックスを必ずチェックします。(Windows95 では場所の指定ボタンをクリックします)
- (5) **CD ドライブ:¥WIN9X** を指定して下さい。
- (6) 後はシステムの指示に従ってインストールを完了させます。

3.7 アンインストール

3.7.1 Windows XP, Windows 2000, Windows NT 4.0, Windows 9X の場合

- (1) 添付ディスクを CD ドライブに挿入します。
- (2) エクスプローラを起動し、CD ドライブ:¥d580uins.exe を実行します。または、コマンドプロンプトから、CD ドライブ:¥d580uins.exe を実行します。

3.7.2 Windows 7, Windows Vista の場合

Windows の「スタート」→「コントロールパネル」→「プログラムのアンインストール」

→「Windows ドライバ パッケージ Hivertec HPCI-DIO580」を右クリックしアンインストールを行います。

4. ボードを複数枚利用する場合

DIO5xx 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。

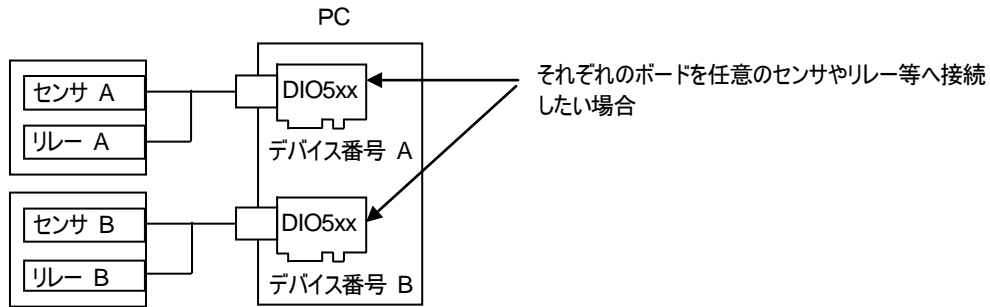


図 4.1-1 ボードを複数枚使用する場合

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 DIO5xx ボードのデバイス番号を確認できます。

「¥sample¥vc¥spd58000.exe」

このプログラムは、DIO5xx ボードのデバイス情報の取得を行っています。

これにより、DIO5xx ボードのデバイス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。

4.3 DIO548 ボード、DIO580A ボードでのボード ID の使用

DIO548 ボードはボード上のジャンパで設定したボード ID (0~15 まで任意に設定可) が使用できます。

DIO580A 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。

DIO580 ボードにはありません。

この使用方法は後述します。

ボード ID の設定値とジャンパ状態は次表のようになります。(出荷状態は ID=0)

詳細は HPCI-DIO548 ユーザーズマニュアルのボード ID 選択を参照してください。

ボード ID 設定値	0	5	7	10	15
ジャンパ状態	8 ID 1 	8 ID 1 	8 ID 1 	8 ID 1 	8 ID 1
(2進表記)	0000	0101	0111	1010	1111

表 4.3-1 HPCI-DIO548 ボード ID の設定

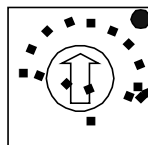


図 4.3-1 HPCI-DIO580A ボード ID ロータリーディップ SW

5. ドライバ I/F 用 DLL の使用方法

5.1 概 要

ドライバ I/F 用 DLL は, Windows9X, WindowsNT, Windows2K, WindowsXP, WindowsVistaにおいて, HPCI-DIO5xx ボードの制御を行うための関数群です.

各関数は"Visual C++ 6.0 以上", "Visual Basic 5.0/6.0/ .NET"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から DLL の関数を呼び出し, DLL 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 Windows9X用に hidio580.vxd, WindowsNT 用に hidio580.sys, Windows2K用に hd580w2k.sys, WindowsXP用に hd580wxp.sys, Windows7,Vista(32 ビット版)用に hd580wdm.sys , Windows7,Vista(64 ビット版)用に hd580x64.sys が使用されます.

5.2 関数一覧

ドライバ I/F 用 DLL は, 次の 13 種類 20 関数が含まれます.

No	関 数 名 称	機 能
1	hdio580_GetDeviceCount()	ボード枚数の取得
2	hdio580_GetDeviceInfo()	デバイス情報の取得
3	hdio580_OpenDeviceID()	デバイスのオープン(ボード ID 参照)
4	hdio580_OpenDevice()	デバイスのオープン(ボード ID 無視)
5	hdio580_CloseDevice()	デバイスのクローズ
6	hdio580_ioswRead()	入出力数切替ポート読込(DIO548 のみ)
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
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
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
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
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
12	hdio580_intIdtStsRead()	割込入力信号 割込要因確認
13	hdio580_GetBoardCode()	ボード固有コードの取得
14	hdio580_rPortW()	オプションポート指定アドレスワード読込
	hdio580_wPortW()	オプションポート指定アドレスワード書込

表 5.2-1 関数一覧

5.3 準備

DLL を使用する手順を説明します。

(1) Visual C++ (5.0 以上)によるアプリケーションの構築

次のファイルをプロジェクトへ追加して下さい。

- hidio580.lib .. インポートライブラリ
- hidio580.h .. C アプリケーション構築用ヘッダーファイル

(注)1. DLL 関数は C 言語で作成されています。

2. ヘッダーファイル中の DLL 関数プロトタイプ宣言は次のように記述されています。

```
//-----  
// DLL 関数プロトタイプ宣言  
//-----  
#ifdef __cplusplus  
extern "C"  
{  
#endif  
    DLL 関数のプロトタイプ宣言  
#ifdef __cplusplus  
}  
#endif
```

これは、アプリケーションを C++コーディング(ファイル拡張子=.cpp)で作成する場合に備えての処理です。

3. 「#ifdef __cplusplus」の定義は"Visual C++"用です。

他言語で使用する場合には、明示的な宣言に変更できます。

```
(例)  
#define CPLUS 1  
.....  
#if CPLUS  
.....  
#endif
```

(2) Visual Basic 5.0/6.0 によるアプリケーションの構築

「hidio580.bas」を、プロジェクトの標準モジュールに追加してください。

このファイルに外部関数宣言(Declare 宣言)及びユーザー定義型宣言が記述されています。

(3) Visual Basic .NET によるアプリケーションの構築

「hidio580.vb」を、プロジェクトに追加してください。

このファイルに外部関数宣言(Declare 宣言)及びユーザー定義型宣言が記述されています。

(4) Visual C# .NET によるアプリケーションの構築

「hidio580.cs」を、プロジェクトに追加してください。

このファイルに外部関数宣言(DllImport)及びユーザー定義型宣言が記述されています。

6. 制御概念

DLL 及びデバイスドライバは複数の HPCI-DIO5xx ボードを制御することができます。

HPCI-DIO5xx 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。)

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 HPCDEVICEINFO 型構造体を用意します。

[Visual C++(C 言語/C++)]

```
typedef struct _HPCDEVICEINFO {
    DWORD nBusNumber;          /* バス番号 */
    DWORD nDeviceNumber;      /* デバイス番号 */
    DWORD dwIoPortAddress;    /* I/O ポートアドレス */
    DWORD dwIrqNo;            /* IRQ 番号 */
    DWORD dwNumber;           /* 管理番号 */
    DWORD dwBoardID;          /* ボード ID(0~15) */
} HPCDEVICEINFO, *PHPCDEVICEINFO
```

[Visual Basic 6.0]

```
Public Type HPCDEVICEINFO
    nBusNumber As Long 'バス番号
    nDeviceNumber As Long 'デバイス番号
    dwIoPortAddress As Long 'I/O ポートアドレス
    dwIrqNo As Long 'IRQ 番号
    dwNumber As Long '管理番号
    dwBoardID As Long 'ボード ID(0~15)
End Type
```

[Visual Basic .NET]

```
Public Structure HPCDEVICEINFO
    Dim nBusNumber As Integer 'バス番号
    Dim nDeviceNumber As Integer 'デバイス番号
    Dim dwIoPortAddress As Integer 'I/O ポートアドレス
    Dim dwIrqNo As Integer 'IRQ 番号
    Dim dwNumber As Integer '管理番号
    Dim dwBoardID As Integer 'ボード ID
End Structure
```

[Visual C# .NET]

```
public struct HPCDEVICEINFO
{
    /// <summary>
    /// バス番号
    /// </summary>
    public uint nBusNumber;

    /// <summary>
    /// デバイス番号
    /// </summary>
    public uint nDeviceNumber;

    /// <summary>
    /// I/O ポートアドレス
    /// </summary>
    public uint dwIoPortAddress;

    /// <summary>
    /// IRQ 番号
    /// </summary>
    public uint dwIrqNo;

    /// <summary>
    /// 管理番号(Windows 9x では無視)
    /// </summary>
    public uint dwNumber;

    /// <summary>
    /// ボード ID
    /// </summary>
    public uint dwBoardID;
}
```

注 1. 管理番号は Windows9Xでは使用されません. 常に「INVALID_HPC_NUMBER (-1)」が格納されています.

注 2. DIO580 ボードではボード ID は 15 に固定されています.

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

"HPCDEVICEINFO"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) ボード毎にデバイスオープン

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ハンドルを返してきます。

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。

- ◆ hdio580_OpenDeviceID()・・・ボードのオープン処理[ボード ID 参照]
- ◆ hdio580_OpenDevice()・・・ボードのオープン処理[ボード ID 無視]

(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

- ◆ hdio580_outpWriteB() または hdio580_outpWriteW()

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。

- ◆ hdio580_inpReadB() または hdio580_inpReadW()

(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。

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

6.3 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)であった場合には、この異常内容を読み込み、異常内容に対応した処理を行います。

(1) 異常内容の読み込み・・・GetLastError() 関数の起動

[C言語: Visual C++]

```
DWORD dErrorCode;
dErrorCode = GetLastError();
```

[Visual Basic 6.0]

```
Dim dErrorCode As Long
dErrorCode = GetLastError()
```

[Visual Basic .NET]

```
Dim dErrorCode As Integer
dErrorCode = GetLastError()
```

[Visual C# .NET]

```
uint dErrorCode;
HidIo580.dErrorCode = GetLastError();
```

(2) 異常内容の一覧

No	読 出 値 (値: C言語16進数表記)	異 常 内 容
1	NO_ERROR (0x00000000)	異常なし(デバイスなし)
2	ERROR_FILE_NOT_FOUND (0x00000002)	デバイスドライバが存在しない
3	ERROR_NOT_ENOUGH_MEMORY (0x00000008)	デバイス情報格納メモリが不足
4	ERROR_HPC_ALREADY_OPENED (0x20000001)	既にオープン済のデバイスをオープン
5	ERROR_HPC_ILLEGAL_DEVICE (0x20001000)	HPCI ボードと認められない
6	ERROR_INVALID_PARAMETER (0x00000087)	指定デバイス情報に該当するボード無し
7	ERROR_HPC_INVALID_HANDLE (0x20000002)	無効なデバイスハンドルを指定
8	ERROR_NOT_READY (0x00000021)	デバイスの入出力ポートが使用できない

表 6.3-1 異常内容一覧

(3) 異常発生時の確認項目

- ① NO_ERROR ◎異常は発生していません。
- ② ERROR_FILE_NOT_FOUND ◎デバイスドライバがインストールされていません。
◎デバイスドライバが所定のフォルダに格納されていません。
- ③ ERROR_NOT_ENOUGH_MEMORY... ◎アプリケーション用のメモリ不足です。
◇パソコン主記憶メモリの不足。
◎システムリソース(OS用メモリ)の不足です。
◇多数のアプリケーション起動。
◇1度に多数のウィンドウを開いた。
- ④ ERROR_HPC_ALREADY_OPENED .. ◎オープン済みデバイスに更にオープン指令を行いました。
◇オープンしたデバイスはクローズするまで使用します。(多重オープン禁止)
◎ボード2枚以上を使用する場合、オープンするデバイス情報を確認して下さい。
- ⑤ ERROR_HPC_ILLEGAL_DEVICE .. ◎デバイス情報は正常ですが、ボード機能が一致していません。
- ⑥ ERROR_INVALID_PARAMETER ... ◎デバイスオープン関数で指定したデバイス情報の内容を確認して下さい。
- ⑦ ERROR_HPC_INVALID_HANDLE .. ◎デバイスオープンで得られた"デバイスハンドル"ではありません。
◎このデバイスは既にクローズされています。
- ⑧ ERROR_NOT_READY ◎デバイス(ボード)内部の入出力ポートが認識できません。
(システムとの不整合等が考えられますので、弊社サポートまでお問い合わせください)

(2)	hdio580_GetDeviceInfo() デバイス情報の取得
-----	--

《機能》

現在パソコンに装着されているHPCI-DIO5xxボードのデバイス情報を取得します。
この結果、HPCDEVICEINFO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》

```
[ C言語: Visual C++ ]
DWORD WINAPI hdio580_GetDeviceInfo ( DWORD* nDeviceNumber, HPCDEVICEINFO* HpcDeviceInfo );

[ Visual Basic 6.0 ]
Declare Function hdio580_GetDeviceInfo Lib "hdio580.dll" _
    (ByRef nDeviceNumber As Long, _
    HpcDeviceInfo As HPCDEVICEINFO) As Long

[ Visual Basic .NET ]
Declare Function hdio580_GetDeviceInfo Lib "hdio580.dll" _
    ( ByRef pcnDevNum As Integer, _
    ByRef pHpcDevInfo As HPCDEVICEINFO) As Integer

[ Visual C# .NET ]
[DllImport("hdio580.dll")]
public static extern int hdio580_GetDeviceInfo(ref int pcnDevNum, ref HPCDEVICEINFO pHpcDevInfo);
```

《引数》

- ◆ **DWORD* nDeviceNumber**
情報を取得するボードの最大枚数が格納されたDWORD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。
- ◆ **HPCDEVICEINFO* HpcDeviceInfo**
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HPCDEVICEINFO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 .. 「6. 3 デバイスドライバの異常報告(P10)」を参照して下さい。

《呼び出し例》 パソコンにHPCI-DIO5xxが2枚装着されていることを想定します。

[C言語: Visual C++]

```
DWORD      ret;           // 関数の戻り値
DWORD      count = 2;     // 最大枚数は2
HPCDEVICEINFO* HpcDeviceInfo[2]; // 2枚のHPCI-DIO5xxのデバイス情報が
                                // セットされるべきエリア

ret = hdio580_GetDeviceInfo(
    &count,           // countのアドレスを渡す。
    &HpcDeviceInfo[0]); // 配列の先頭アドレスを渡す。
```

[Visual Basic 6.0]

```
Dim ret          As Long      ' 関数の戻り値
Dim count        As Long      ' ボード枚数
Dim HpcDeviceInfo(2) As HPCDEVICEINFO ' 2枚のHPCI-DIO5xxのデバイス情報が
                                ' セットされるべきエリア

count = 2        ' 最大枚数は2
ret = hdio580_GetDeviceInfo( _
    count, _     ' countのアドレスを渡す。
    HpcDeviceInfo(0) ) ' 配列の先頭アドレスを渡す。
```

[Visual Basic .NET]

```
Dim ret          As Integer    ' 関数の戻り値
Dim count        As Integer    ' ボード枚数
Dim HpcDeviceInfo(2) As HPCDEVICEINFO ' 2枚のHPCI-DIO5xxのデバイス情報が
                                ' セットされるべきエリア

count = 2        ' 最大枚数は2
ret = hdio580_GetDeviceInfo( _
    count, _     ' countのアドレスを渡す。
    HpcDeviceInfo(0) ) ' 配列の先頭アドレスを渡す。
```

[Visual C# .NET]

```
uint          ret;           // 関数の戻り値
uint          count = 2;     // 最大枚数は2
// 2枚のHPCI-DIO5xxのデバイス情報がセットされるべきエリア
Hidio580.HPCDEVICEINFO[] HpcDevInfo = new Hidio580.HPCDEVICEINFO[2];
ret = Hidio580.hdio580_GetDeviceInfo(
    ref count,           // countのアドレスを渡す。
    ref 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》

渡したデバイス情報を持つHPCI-DIO548ボードをオープンし、他のHPCI-DIO5xxボード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HPCI-DIO548ボードにアクセスするためのハンドルとなります。
HPCI-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》

```
[ C言語: Visual C++ ]
    DWORD WINAPI hdio580_OpenDeviceID( HPCDEVICEINFO* HpcDeviceInfo );
[ Visual Basic 6.0 ]
    Declare Function hdio580_OpenDeviceID Lib "hdio580.dll" _
        (HpcDeviceInfo As HPCDEVICEINFO) As Long
[ Visual Basic .NET ]
    Declare Function hdio580_OpenDeviceID Lib "hdio580.dll" _
        (ByRef HpcDeviceInfo As HPCDEVICEINFO) As Integer

[ Visual C# .NET ]
    [DllImport("hdio580.dll")]
    public static extern int hdio580_OpenDeviceID(ref HPCDEVICEINFO pHpcDevInfo);
```

《引数》

- ◆ HPCDEVICEINFO* HpcDeviceInfo
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》 デバイスハンドル値

INVALID_HANDLE_VALUE (= -1) : オープン失敗
「6. 3 デバイスドライバの異常報告(P10)」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 : 0~15(ボード上のジャンパ設定値)
- 下位ワード: オープン順の番号: 1~16

【ヒント】 上位ワードの”ボードID”をデバイスハンドル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。

```
VC++ [ hDevID &= 0xffff0000; ]
VB    [ hDevID = hDevID And &HFFFFFF0000 ]
```

《呼び出し例》

パソコンにHPCI-DIO5xxが2枚装着されていることを想定します。

デバイス情報格納エリアとしてHPCDEVICEINFO型の配列 HpcDeviceInfo[2]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。

[C言語: Visual C++]

```
DWORD          hDevID[2];          //デバイスハンドル取得エリア
```

```
hDevID[0] = hdio580_OpenDeviceID(  
            &HpcDeviceInfo[0]); //1番目のデバイス情報
```

```
hDevID[1] = hdio580_OpenDeviceID(  
            &HpcDeviceInfo[1]); //2番目のデバイス情報
```

[Visual Basic 6.0]

```
Dim hDevID(2) As Long           'デバイスハンドル取得エリア
```

```
hDevID(0) = hdio580_OpenDeviceID(_  
            HpcDeviceInfo(0)) '1番目のデバイス情報
```

```
hDevID(1) = hdio580_OpenDeviceID(_  
            HpcDeviceInfo(1)) '2番目のデバイス情報
```

[Visual Basic .NET]

```
Dim hDevID(2) As Integer       'デバイスハンドル取得エリア
```

```
hDevID(0) = hdio580_OpenDeviceID(_  
            HpcDeviceInfo(0)) '1番目のデバイス情報
```

```
hDevID(1) = hdio580_OpenDeviceID(_  
            HpcDeviceInfo(1)) '2番目のデバイス情報
```

[Visual C# .NET]

```
uint[] hDevID = new uint[2];    //デバイスハンドル取得エリア
```

```
hDevID[0] = Hidio580.hdio580_OpenDeviceID(  
            ref HpcDeviceInfo[0]); //1番目のデバイス情報
```

```
hDevID[1] = Hidio580.hdio580_OpenDeviceID(  
            ref HpcDeviceInfo[1]); //2番目のデバイス情報
```

(4)	hdio580_OpenDevice() デバイスのオープン(ポートID無視)
-----	---

《機能》

指定したデバイス情報を持つHPCI-DIO5xxボードをオープンし、他のHPCI-DIO5xxボード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HPCI-DIO5xxボードにアクセスするためのハンドルとなります。

《書式》

```
[ C言語: Visual C++ ]
    DWORD WINAPI hdio580_OpenDevice( HPCDEVICEINFO* HpcDeviceInfo );
[ Visual Basic 6.0 ]
    Declare Function hdio580_OpenDevice Lib "hdio580.dll" _
        (HpcDeviceInfo As HPCDEVICEINFO) As Long
[ Visual Basic .NET ]
    Declare Function hdio580_OpenDevice Lib "hdio580.dll" _
        (ByRef HpcDeviceInfo As HPCDEVICEINFO) As Integer
[ Visual C# .NET ]
    [DllImport("hdio580.dll")]
    public static extern int hdio580_OpenDevice(ref HPCDEVICEINFO pHpcDevInfo);
```

《引数》

◆ HPCDEVICEINFO* HpcDeviceInfo
 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》 デバイスハンドル値

INVALID_HANDLE_VALUE (= -1) : オープン失敗
 「6.3 デバイスドライバの異常報告(P10)」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: オープン順の番号: 1~16

《呼び出し例》

パソコンにHPCI-DIO5xxが2枚装着されていることを想定します。デバイス情報格納エリアとしてHPCDEVICEINFO型の配列HpcDeviceInfo[2]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。

```
[ C言語: Visual C++ ]
    DWORD          hDevID[2];           //デバイスハンドル取得エリア
    hDevID[0] = hdio580_OpenDevice( _
        &HpcDeviceInfo[0] ); //1番目のデバイス情報
    hDevID[1] = hdio580_OpenDevice( _
        &HpcDeviceInfo[1] ); //2番目のデバイス情報
[ Visual Basic 6.0 ]
    Dim hDevID(2) As Long              'デバイスハンドル取得エリア

    hDevID(0) = hdio580_OpenDevice( _
        HpcDeviceInfo(0) )           '1番目のデバイス情報
    hDevID(1) = hdio580_OpenDevice( _
        HpcDeviceInfo(1) )           '2番目のデバイス情報
[ Visual Basic .NET ]
    Dim hDevID(2) As Integer          'デバイスハンドル取得エリア

    hDevID(0) = hdio580_OpenDevice( _
        HpcDeviceInfo(0) )           '1番目のデバイス情報
    hDevID(1) = hdio580_OpenDevice( _
        HpcDeviceInfo(1) )           '2番目のデバイス情報
[ Visual C# .NET ]
    uint[] hDevID = new uint[2];     //デバイスハンドル取得エリア

    hDevID[0] = Hdio580.Hdio580_OpenDevice(
        ref HpcDeviceInfo[0] ); //1番目のデバイス情報
    hDevID[1] = Hdio580.Hdio580_OpenDevice(
        ref HpcDeviceInfo[1] ); //2番目のデバイス情報
```


(6)	hdio580_ioswRead() 入出力数切替ポート読込
-----	---------------------------------------

《機能》

デバイスハンドルで指定されたHPCI-DIO548の、入出力数設定スイッチ(ジャンパ設定)内容を読み込み、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》

```
[ C言語: Visual C++ ]
  DWORD WINAPI hdio580_ioswRead (DWORD hDevID, BYTE* swin);

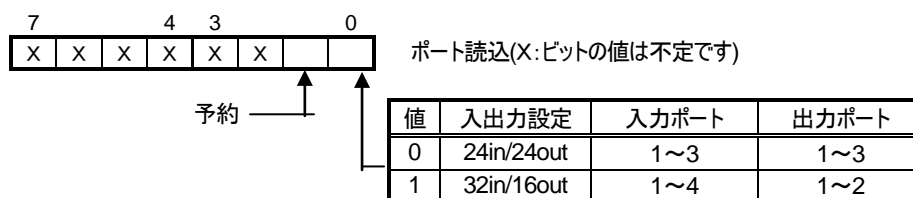
[ Visual Basic 6.0 ]
  Declare Function hdio580_ioswRead Lib "hdio580.dll" _
    (ByVal hDevID As Long, ByRef swin As Byte) As Long

[ Visual Basic .NET ]
  Declare Function hdio580_ioswRead Lib "hdio580.dll" _
    (ByVal hDevID As Integer, ByRef swin As Byte) As Integer

[ Visual C# .NET ]
  [DllImport("hdio580.dll")] public static extern int hdio580_ioswRead(int hDevID, ref byte bytin);
```

《引数》

- ◆ DWORD hDevID・・・対象デバイスのデバイスハンドル.
- ◆ 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



※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 .. 「6. 3 デバイスドライバの異常報告(P10)」を参照して下さい。

《呼び出し例》

[C言語: Visual C++]

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

```
ret = hdio580_ioswRead ( hDevID, //デバイスハンドル  
                        &swin ); //設定データ0ならば24in/24out  
                                //設定データ1ならば32in/16out
```

[Visual Basic 6.0]

```
Dim ret As Long '関数の戻り値  
Dim swin As Byte '読込結果の格納エリア
```

```
ret = hdio580_ioswRead ( hDevID, _ 'デバイスハンドル  
                        swin ) '設定データ0ならば24in/24out  
                              '設定データ1ならば32in/16out
```

[Visual Basic .NET]

```
Dim ret As Integer '関数の戻り値  
Dim swin As Byte '読込結果の格納エリア
```

```
ret = hdio580_ioswRead ( hDevID, _ 'デバイスハンドル  
                        swin ) '設定データ0ならば24in/24out  
                              '設定データ1ならば32in/16out
```

[Visual C# .NET]

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

```
ret = Hdio580.hdio580_ioswRead ( hDevID, //デバイスハンドル  
                                ref swin ); //設定データ0ならば24in/24out  
                                           //設定データ1ならば32in/16out
```

(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》

デバイスハンドルで指定されたHPCI-DIO5xxの、portnoで指定された入力ポートから
 入力ポートバイト読込 … 1バイトを読み込み、指定したエリアに格納します。
 入力ポートワード読込 … 2バイトを読み込み、指定したエリアに格納します。

《書式》

[C言語: Visual C++]

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

[Visual Basic 6.0]

```
Declare Function hdio580_inpReadB Lib "hdio580. DLL"
    (ByVal hDevID As Long, ByVal port As Integer,
     ByRef bytin As Byte) As Long
Declare Function hdio580_inpReadW Lib "hdio580. DLL"
    (ByVal hDevID As Long, ByVal port As Integer,
     ByRef wrdin As Integer) As Long
```

[Visual Basic .NET]

```
Declare Function hdio580_inpReadB Lib "hdio580. DLL"
    (ByVal hDevID As Integer, ByVal port As Short,
     ByRef bytin As Byte) As Long
Declare Function hdio580_inpReadW Lib "hdio580. DLL"
    (ByVal hDevID As Integer, ByVal port As Short,
     ByRef wrdin As Short) As Long
```

[Visual C# .NET]

```
[DllImport("hdio580.dll")]
public static extern int hdio580_inpReadB(int hDevID, short PortNo, ref byte byInp);
[DllImport("hdio580.dll")]
public static extern int hdio580_inpReadW(int hDevID, short PortNo, ref short wInp);
```

《引数》

- ◆ DWORD hDevID…対象デバイスのデバイスハンドル
- ◆ WORD port …ポート番号指定 0~4 (ポート番号-1)

ポート番号		5	4	3	2	1
入力信号番号		40 - 33	32 - 25	24 - 17	16 - 9	8 - 1
読込	バイト	port=4	port=3	port=2	port=1	port=0
	ワード	注1.未搭載ポートを読み込む時、入力値は不定		上位バイト port=0 下位バイト		
		注2.port=4上位バイト入力値は不定		上位バイト port=1 下位バイト		
		上位バイト port=2 下位バイト				
		上位バイト port=3 下位バイト				
		port=4				
		← DIO548 (24in/24out) →				
		← DIO548 (32in/16out) →				
		← DIO580 (40in/40out) →				

- ◆ 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 … 「6.3 デバイスドライバの異常報告(P10)」を参照して下さい。

《呼び出し例》

```
[ C言語: Visual C++ ]
    DWORD ret;           //関数の戻り値
    BYTE  bytin;         //格納先
    ret = hdio580_inpReadB( hDevID, 0, &bytin ); //ポート1をバイト入力

[ Visual Basic 6.0 ]
    Dim ret      As Long      '関数の戻り値
    Dim wrdin    As Integer   '格納先
    ret = hdio580_inpReadW( hDevID, 1, wrdin )   'ポート2をワード入力

[ Visual Basic .NET ]
    Dim ret      As Integer   '関数の戻り値
    Dim wrdin    As Short     '格納先
    ret = hdio580_inpReadW( hDevID, 1, wrdin )   'ポート2をワード入力

[ Visual C# .NET ]
    uint ret;           //関数の戻り値
    byte  bytin;        //格納先
    ret = Hidio580.hdio580_inpReadB( hDevID, 0, ref bytin ); //ポート1をバイト入力
```

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

《機能》

デバイスハンドルで指定されたHPCI-DIO5xxの、portnoで指定された出力ポートから
出力ポートバイト読込 ... 1バイトを読み込み、
出力ポートワード読込 ... 2バイトを読み込み、指定したエリアに格納します。
デバイスハンドルで指定されたHPCI-DIO5xxの、portno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。

《書式》

[C言語: Visual C++]

```
DWORD WINAPI hdio580_outpReadB (DWORD hDevID, WORD portno, BYTE* bytin);
DWORD WINAPI hdio580_outpReadW (DWORD hDevID, WORD portno, WORD* wrdin);

DWORD WINAPI hdio580_outpWriteB (DWORD hDevID, WORD portno, BYTE bytdt);
DWORD WINAPI hdio580_outpWriteW (DWORD hDevID, WORD portno, WORD wrddt);
```

[Visual Basic 6.0]

```
Declare Function hdio580_outpReadB Lib "hdio580.dll" _
    (ByVal hDevID As Long, ByVal portno As Integer, _
    ByRef bytin As Byte) As Long
Declare Function hdio580_outpReadW Lib "hdio580.dll" _
    (ByVal hDevID As Long, ByVal portno As Integer, _
    ByRef wrdin As Integer) As Long
Declare Function hdio580_outpWriteB Lib "hdio580.dll" _
    (ByVal hDevID As Long, ByVal portno As Integer, _
    ByVal bytdt As Byte) As Long
Declare Function hdio580_outpWriteW Lib "hdio580.dll" _
    (ByVal hDevID As Long, ByVal portno As Integer, _
    ByVal wrddt As Integer) As Long
```

[Visual Basic .NET]

```
Declare Function hdio580_outpReadB Lib "hdio580.dll" _
    (ByVal hDevID As Integer, ByVal portno As Short, _
    ByRef bytin As Byte) As Long
Declare Function hdio580_outpReadW Lib "hdio580.dll" _
    (ByVal hDevID As Integer, ByVal portno As Short, _
    ByRef wrdin As Short) As Long
Declare Function hdio580_outpWriteB Lib "hdio580.dll" _
    (ByVal hDevID As Integer, ByVal portno As Short, _
    ByVal bytdt As Byte) As Long
Declare Function hdio580_outpWriteW Lib "hdio580.dll" _
    (ByVal hDevID As Integer, ByVal portno As Short, _
    ByVal wrddt As Short) As Long
```

[Visual C# .NET]

```
[DllImport("hdio580.dll")]
public static extern int hdio580_outpReadB(int hDevID, short PortNo, ref byte byOutp);
[DllImport("hdio580.dll")]
public static extern int hdio580_outpReadW(int hDevID, short PortNo, ref short wOutp);
[DllImport("hdio580.dll")]
public static extern int hdio580_outpWriteB(int hDevID, short PortNo, byte byData);
[DllImport("hdio580.dll")]
public static extern int hdio580_outpWriteW(int hDevID, short PortNo, short wData);
```

《引数》

- ◆ DWORD hDevID...対象デバイスのデバイスハンドル
- ◆ WORD port ...ポート番号指定 0~4 (ポート番号-1)

ポート番号	5	4	3	2	1	
入力信号番号	40 - 33	32 - 25	24 - 17	16 - 9	8 - 1	
読 込 ・ 書 込	バイト	port=4	port=3	port=2	port=1	port=0
	ワード	注1.未搭載ポートを読み込む時、入力値は不定			上位バイト port=0 下位バイト	
		注2.port=4上位バイト入力値は不定		上位バイト port=1 下位バイト		
		上位バイト port=2 下位バイト				
上位バイト port=3 下位バイト						
port=4		← DIO548 (32in/16out) →				
		← DIO548 (24in/24out) →				
		← DIO580 (40in/40out) →				

- ◆ 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
- ◆ BYTE bytdt ...書込む1バイトデータ
- ◆ WORD wrddt ...書込む2バイトデータ

《戻り値》 処理結果

- 1:成功
- 0:失敗 .. 「6. 3 デバイスドライバの異常報告(P10)」を参照して下さい。

《呼び出し例》

[C言語: Visual C++]

```

DWORD ret; //関数の戻り値
BYTE bytin; //格納先
ret = hdio580_outpReadB( hDevID, 0, &bytin ); //ポート1をバイト入力
if(ret) {
    bytin &= 0x0f; //ポート1上位4ビットoff
    ret = hdio580_outpWriteB( hDevID, 0, bytin ); //ポート1へバイト出力
}
    
```

[Visual Basic 6.0]

```

Dim ret As Long
Dim wrdin As Integer
ret = hdio580_outpReadW( hDevID, 1, wrdin ) 'ポート2をワード入力
If ret<>0 Then
    wrdin = wrdin Or &HF000 'ポート1上位4ビットon
    ret = hdio580_outpWrite( hDevID, 1, wrdin ) 'ポート1へワード出力
End If
    
```

[Visual Basic .NET]

```

Dim ret As Integer
Dim wrdin As Short
ret = hdio580_outpReadW( hDevID, 1, wrdin ) 'ポート2をワード入力
If ret<>0 Then
    wrdin = wrdin Or &HF000 'ポート1上位4ビットon
    ret = hdio580_outpWrite( hDevID, 1, wrdin ) 'ポート1へワード出力
End If
    
```

[Visual C# .NET]

```

uint ret; //関数の戻り値
byte bytin; //格納先
ret = Hdio580.hdio580_outpReadB( hDevID, 0, ref bytin ); //ポート1をバイト入力
if(ret) {
    bytin &= 0x0f; //ポート1上位4ビットoff
    ret = Hdio580.hdio580_outpWriteB( hDevID, 0, bytin ); //ポート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》

```
[ C言語: Visual C++ ]
DWORD hdio580_intFilterWrite( DWORD hDevID, BYTE fil_data );
DWORD hdio580_intFilterRead ( DWORD hDevID, BYTE* fil_data );

[ Visual Basic 6.0 ]
Declare Function hdio580_intFilterWrite Lib "hdio580.dll" _
    (ByVal hDevID As Long, ByVal fil_data As Byte) As Long
Declare Function hdio580_intFilterRead Lib "hdio580.dll" _
    (ByVal hDevID As Long, ByRef fil_data As Byte) As Long

[ Visual Basic .NET ]
Declare Function hdio580_intFilterWrite Lib "hdio580.dll" _
    (ByVal hDevID As Integer, ByVal fil_data As Byte) As Integer
Declare Function hdio580_intFilterRead Lib "hdio580.dll" _
    (ByVal hDevID As Integer, ByRef fil_data As Byte) As Integer

[ Visual C# .NET ]
[DllImport("hdio580.dll")]
public static extern int hdio580_intFilterWrite(int hDevID, byte byData);
[DllImport("hdio580.dll")]
public static extern int hdio580_intFilterRead(int hDevID, ref byte byData);
```

《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)
DI580	IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1	・・・書込
	IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1	・・・読込
DO580	0	0	0	0	0	0	0	IN1	・・・書込
	0	0	0	0	0	0	0	IN1	・・・読込

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 .. 「6. 3 デバイスドライバの異常報告(P10)」を参照して下さい。

《呼び出し例》

[C言語: Visual C++]

```
DWORD   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 */
```

[Visual Basic 6.0]

```
Dim ret      As Long
Dim fil_data As Byte

ret = hdio580_intFilterWrite( _
        hDevID, _      ' デバイス ID
        &H3 )         ' 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
```

[Visual Basic .NET]

```
Dim ret      As Integer
Dim fil_data As Byte

ret = hdio580_intFilterWrite( _
        hDevID, _      ' デバイス ID
        &H3 )         ' 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
```

[Visual C# .NET]

```
uint   ret;           /* 関数の戻り値 */
byte   fil_data;
ret = Hidio580.hdio580_intFilterWrite(
        hDevID,         /* デバイス ID */
        0x03 );        /* IN2,IN1:フィルタ設定 */
ret = Hidio580.hdio580_intFilterRead(
        hDevID,         /* デバイス ID */
        ref 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》

```
[ C言語: Visual C++ ]
  DWORD hdio580_intEdgeWrite( DWORD hDevID, BYTE edge_data );
  DWORD hdio580_intEdgeRead ( DWORD hDevID, BYTE* edge_data );

[ Visual Basic 6.0 ]
  Declare Function hdio580_intEdgeWrite Lib "hdio580.dll" _
    (ByVal hDevID As Long, ByVal edge_data As Byte) As Long
  Declare Function hdio580_intEdgeRead Lib "hdio580.dll" _
    (ByVal hDevID As Long, ByRef edge_data As Byte) As Long

[ Visual Basic .NET ]
  Declare Function hdio580_intEdgeWrite Lib "hdio580.dll" _
    (ByVal hDevID As Integer, ByVal edge_data As Byte) As Integer
  Declare Function hdio580_intEdgeRead Lib "hdio580.dll" _
    (ByVal hDevID As Integer, ByRef edge_data As Byte) As Integer

[ Visual C# .NET ]
  [DllImport("hdio580.dll")]
  public static extern int hdio580_intEdgeWrite(int hDevID, byte byData);
  [DllImport("hdio580.dll")]
  public static extern int hdio580_intEdgeRead(int hDevID, ref byte byData);
```

《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)
DI580	IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1	・・・書込
	IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1	・・・読込
DO580	0	0	0	0	0	0	0	IN1	・・・書込
	0	0	0	0	0	0	0	IN1	・・・読込

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 .. 「6. 3 デバイスドライバの異常報告(P10)」を参照して下さい。

《呼び出し例》

[C言語: Visual C++]

```
DWORD    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 */
```

[Visual Basic 6.0]

```
Dim ret      As Long
Dim edge_data As Byte
ret = hdio580_intEdgeWrite( _
        hDevID, _      ' デバイス ID
        &H1 )         ' IN1:1→0
ret = hdio580_intEdgeRead( _
        hDevID, _      ' デバイス ID
        edge_data )   ' 格納先のアドレス
```

[Visual Basic .NET]

```
Dim ret      As Integer
Dim edge_data As Byte

ret = hdio580_intEdgeWrite( _
        hDevID, _      ' デバイス ID
        &H1 )         ' IN1:1→0
ret = hdio580_intEdgeRead( _
        hDevID, _      ' デバイス ID
        edge_data )   ' 格納先のアドレス
```

[Visual C# .NET]

```
uint    ret;          /* 関数の戻り値 */
byte    edge_data;
ret = Hidio580.hdio580_intEdgeWrite(
        hDevID,        /* デバイス ID */
        0x01 );       /* IN1:1→0 */
ret = Hidio580.hdio580_intEdgeRead(
        hDevID,        /* デバイス ID */
        ref edge_data ); /* 格納先のアドレス */
```

(11)	hdio580_intlDtselWrite() 割込入力信号 要因選択
	hdio580_intlDtselRead() 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み、指定した 1 バイトのエリアに格納します。

《書式》

[C言語: Visual C++]

```
DWORD hdio580_intlDtselWrite( DWORD hDevID, BYTE sel_data );
DWORD hdio580_intlDtselRead ( DWORD hDevID, BYTE* sel_data );
```

[Visual Basic 6.0]

```
Declare Function hdio580_intlDtselWrite Lib "hdio580.dll" _
    (ByVal hDevID As Long, ByVal sel_data As Byte) As Long
Declare Function hdio580_intlDtselRead Lib "hdio580.dll" _
    (ByVal hDevID As Long, ByRef sel_data As Byte) As Long
```

[Visual Basic .NET]

```
Declare Function hdio580_intlDtselWrite Lib "hdio580.dll" _
    (ByVal hDevID As Integer, ByVal sel_data As Byte) As Integer
Declare Function hdio580_intlDtselRead Lib "hdio580.dll" _
    (ByVal hDevID As Integer, ByRef sel_data As Byte) As Integer
```

[Visual C# .NET]

```
[DllImport("hdio580.dll")] public static extern int hdio580_intlDtselWrite(int hDevID, byte byData);
[DllImport("hdio580.dll")] public static extern int hdio580_intlDtselRead(int hDevID, ref byte byData);
```

《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)
DI580	IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1	..書込
	IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1	..読込
DO580	0	0	0	0	0	0	0	IN1	..書込
	0	0	0	0	0	0	0	IN1	..読込

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 .. 「6.3 デバイスドライバの異常報告(P10)」を参照して下さい。

《呼び出し例》

[C言語: Visual C++]

```
DWORD    ret;          /* 関数の戻り値 */
BYTE     sel_data;
ret = hdio580_intlDtlSelWrite(
        hDevID,        /* デバイス ID */
        0x01 );       /* IN1:選択 */
ret = hdio580_intlDtlSelRead(
        hDevID,        /* デバイス ID */
        &sel_data );  /* 格納先のアドレス */
```

[Visual Basic 6.0]

```
Dim ret      As Long
Dim sel_data As Byte
ret = hdio580_intlDtlSelWrite( _
        hDevID, _      ' デバイス ID
        &H1 )         ' IN1:選択
ret = hdio580_intlDtlSelRead( _
        hDevID, _      ' デバイス ID
        sel_data )    ' 格納先のアドレス
```

[Visual Basic .NET]

```
Dim ret      As Integer
Dim sel_data As Byte

ret = hdio580_intlDtlSelWrite( _
        hDevID, _      ' デバイス ID
        &H1 )         ' IN1:選択
ret = hdio580_intlDtlSelRead( _
        hDevID, _      ' デバイス ID
        sel_data )    ' 格納先のアドレス
```

[Visual C# .NET]

```
uint    ret;          /* 関数の戻り値 */
byte    sel_data;
ret = Hidio580.hdio580_intlDtlSelWrite(
        hDevID,        /* デバイス ID */
        0x01 );       /* IN1:選択 */
ret = Hidio580.hdio580_intlDtlSelRead(
        hDevID,        /* デバイス ID */
        ref sel_data ); /* 格納先のアドレス */
```

(12)	hdio580_intlDtsRead() 割込入力信号 割込要因確認とクリア
------	---

《機能》

デバイスIDで指定されたDIO5xxの、割込入力信号を読み込み、読み込み後割込要因をクリアします。

《書式》

[C言語: Visual C++]

```
DWORD hdio580_intlDtsRead( DWORD hDevID, BYTE* idt_data );
```

[Visual Basic 6.0]

```
Declare Function hdio580_intlDtsRead Lib "hdio580.dll" _
    ( ByVal hDevID As Long, ByVal idt_data As Byte ) As Long
```

[Visual Basic .NET]

```
Declare Function hdio580_intlDtsRead Lib "hdio580.dll" _
    ( ByVal hDevID As Integer, ByVal idt_data As Byte ) As Integer
```

[Visual C# .NET]

```
[DllImport("hdio580.dll")] public static extern int hdio580_intlDtsRead(int hDevID, ref byte byData);
```

《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
DIO548	x	x	x	x	x	IN3	IN2	IN1
DI580	IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1
DO580	0	0	0	0	0	0	0	IN1

.. 読込(xは不定)

.. 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 .. 「6. 3 デバイスドライバの異常報告(P10)」を参照して下さい。

《呼び出し例》

[C言語: Visual C++]

```
DWORD ret; /* 関数の戻り値 */
BYTE idt_data;
ret = hdio580_intlDtsRead(
    hDevID, /* デバイスID */
    &idt_data); /* 格納先のアドレス */
```

[Visual Basic 6.0]

```
Dim ret As Long
Dim idt_data As Byte
ret = hdio580_intlDtsRead( _
    hDevID, _ ' デバイスID
    idt_data ) ' 格納先のアドレス
```

[Visual Basic .NET]

```
Dim ret As Integer
Dim idt_data As Byte
ret = hdio580_intlDtsRead( _
    hDevID, _ ' デバイスID
    idt_data ) ' 格納先のアドレス
```

[Visual C# .NET]

```
uint ret; /* 関数の戻り値 */
byte idt_data;
ret = Hdio580.Hdio580_intlDtsRead(
    hDevID, /* デバイスID */
    ref idt_data ); /* 格納先のアドレス */
```

(13)

hdio580_GetBoardCode () ボード固有コードの取得

《機能》

HPCI-DIO5xx ボードに対して、指定したボード固有コードの取得を行います。
これにより、HPCI-DIO580A、HPCI-DIO580、HPCI-DIO548の各ボードの区分が可能となります。

《書式》

```
[ C言語: Visual C++ ]
    DWORD WINAPI hdio580_GetBoardCode( DWORD hDevID, WORD* b_code );
[ Visual Basic 6.0 ]
    Declare Function hdio580_GetBoardCode Lib "hdio580.dll" _
        ( ByVal hDevID As Long, ByRef b_code As Integer ) As Long
[ Visual Basic .NET ]
    Declare Function hdio580_GetBoardCode Lib "hdio580.dll" _
        ( ByVal hDevID As Integer, ByRef b_code As Short ) As Integer
[ Visual C# .NET ]
    [DllImport("hdio580.dll")]
    public static extern int hdio580_GetBoardCode(int hDevID, ref short b_code);
```

《引数》

- ◆ DWORD hDevID ...対象デバイスのデバイスハンドル
- ◆ WORD b_code ...読出データ(ボード固有コード)の格納エリアアドレス
800xh: DIO580A, DI580, DO580,
FFFFh: DIO580, FF0xh: DIO548(xはボードID設定値)

《戻り値》 処理結果

- 1: 成功
- 0: 失敗 .. 「6. 3 デバイスドライバの異常報告(P10)」を参照して下さい。

《呼び出し例》

```
[ C言語: Visual C++ ]
    DWORD ret; //関数の戻り値
    WORD b_code; //ボード固有コード

    ret = hdio580_GetBoardCode( hDevID, &b_code );

[ Visual Basic 6.0 ]
    Dim ret As Long '関数の戻り値
    Dim b_code As Integer 'ボード固有コード

    ret = hdio580_GetBoardCode( hDevID, b_code )

[ Visual Basic .NET ]
    Dim ret As Integer '関数の戻り値
    Dim b_code As Short 'ボード固有コード

    ret = hdio580_GetBoardCode( hDevID, b_code )

[ Visual C# .NET ]
    uint ret; //関数の戻り値
    ushort b_code; //ボード固有コード

    ret = Hdio580.hdio580_GetBoardCode( hDevID, ref b_code );
```

(14)	hdio580_rPortW () hdio580_wPortW()	オプションポート指定アドレスワード読込 オプションポート指定アドレスワード書込
------	---	--

《機能》

デバイスハンドルで指定されたHPCI-DIO5xxの, adrsで指定されたオプションポートから2バイトを読み、指定したエリアに格納します。

デバイスハンドルで指定されたHPCI-DIO5xxの, adrsで指定されたオプションポートへ2バイトを書込みます。

《書式》

[C言語: Visual C++]

```
DWORD WINAPI hdio580_rPortW (DWORD hDevID, WORD adrs, WORD* wrddt;
DWORD WINAPI hdio580_wPortW (DWORD hDevID, WORD adrs, WORD wrddt);
```

[Visual Basic 6.0]

```
Declare Function hdio580_rPortW Lib "hdio580.dll" _
    (ByVal hDevID As Long, ByVal adrs As Integer, _
    ByRef wrddt As Integer) As Long
```

```
Declare Function hdio580_wPortW Lib "hdio580.dll" _
    (ByVal hDevID As Long, ByVal adrs As Integer, _
    ByVal wrddt As Integer) As Long
```

[Visual Basic .NET]

```
Declare Function hdio580_rPortW Lib "hdio580.dll" _
    (ByVal hDevID As Integer, ByVal adrs As Short, _
    ByRef wrddt As Short) As Long
```

```
Declare Function hdio580_wPortW Lib "hdio580.dll" _
    (ByVal hDevID As Integer, ByVal adrs As Short, _
    ByVal wrddt As Short) As Long
```

[Visual C# .NET]

```
[DllImport("hdio580.dll")]
public static extern int hdio580_rPortW(int hDevID, short PortAdr, ref short wData);
[DllImport("hdio580.dll")]
public static extern int hdio580_wPortW(int hDevID, short PortAdr, short wData);
```

《引数》

- ◆ DWORD hDevID・・・対象デバイスのデバイスハンドル
- ◆ WORD adrs ...オプションポートアドレス
- ◆ WORD* wrddt ...読込んだデータを格納する2バイトエリアのアドレス
- ◆ WORD wrddt ...書込む2バイトデータ

《戻り値》 処理結果

1: 成功

0: 失敗 .. 「6. 3 デバイスドライバの異常報告(P10)」を参照して下さい。

《呼び出し例》

[C言語: Visual C++]

```
DWORD ret; //関数の戻り値
BYTE wrddt; //格納先
ret = hdio580_rPortW( hDevID, 0, &wrddt ); //オプションポートアドレス0をワード読込
ret = hdio580_wPortW( hDevID, 0, wrddt ); //オプションポートアドレス0へワード書込
```

[Visual Basic 6.0]

```
Dim ret As Long '関数の戻り値
Dim wrddt As Integer '格納先
ret = hdio580_rPortW ( hDevID, 1, wrddt ) 'オプションポートアドレス1をワード読込
ret = hdio580_wPortW ( hDevID, 1, wrddt ) 'オプションポートアドレス1へワード書込
```

[Visual Basic .NET]

```
Dim ret As Integer '関数の戻り値
Dim wrdin As Short '格納先
ret = hdio580_rPortW ( hDevID, 1, wrddt ) 'オプションポートアドレス1をワード読込
ret = hdio580_wPortW ( hDevID, 1, wrddt ) 'オプションポートアドレス1へワード書込
```

[Visual C# .NET]

```
uint ret; //関数の戻り値
short wrddt; //格納先
ret = Hidio580.hdio580_rPortW ( hDevID, 0, ref wrddt ); //オプションポートアドレス0をワード読込
ret = Hidio580.hdio580_wPortW ( hDevID, 0, wrddt ); //オプションポートアドレス0へワード書込
```

8. DIO580, DIO548 用サンプルプログラム

ドライバ/IF用DLLの各関数の使用方法を解説する目的のサンプルプログラムを添付しています。
サンプルプログラムは次の5種類があり、ほぼ同一の画面表示と操作となっています。
以降のサンプルプログラム説明では、①の「Cコーディング」を用います。

- (1) Visual C++ 6.0・・・ C コーディング 【 spd58000.exe 】
- (2) Visual C++ 6.0・・・ (MFC) 【 spd58001.exe 】
- (3) Visual Basic 6.0 【 spd58002.exe 】
- (4) Visual Basic .NET2003 【 spd58003.exe 】
- (5) Visual C# .NET2003 【 spd58003.exe 】

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

◀ ご注意 ▶

- (1) Visual Basic サンプルは開発ツールとして「Visual Basic 6.0」がインストールされている必要があります。
- (2) 実行開始時に次のエラーメッセージが表示される場合には、サンプルプログラムは動作しません。

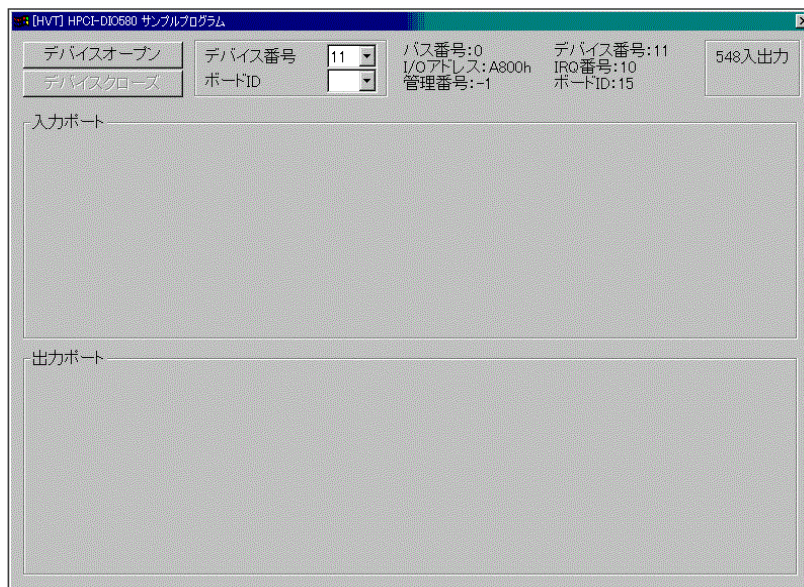


【 エラーメッセージの表示 】

- ※ HPCI-DIO5xxボードが未搭載。
- ※ デバイスドライバがインストールされていない。

8.1 サンプルプログラムの操作

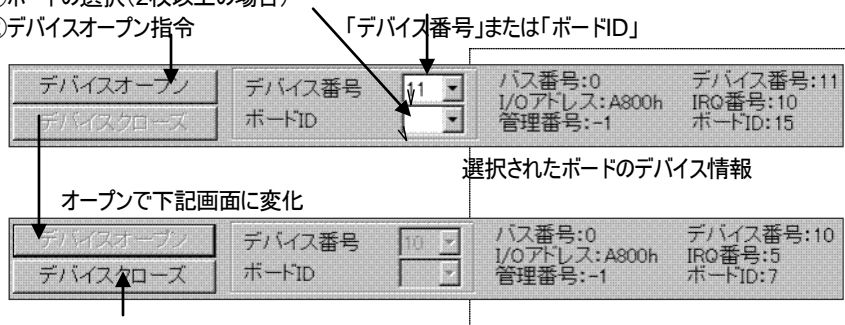
サンプルプログラムが起動され、1枚以上のボード(デバイス)が正常に認識される時、次の画面が表示されます。



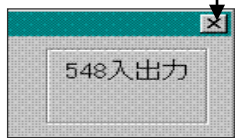
8.1.1 ボード(デバイス)の選択

サンプルプログラムでは、ボード上の動作操作開始・終了は次の手順に従います。

- ①ボードの選択(2枚以上の場合)
- ②デバイスオープン指令



- ③デバイスクローズ指令
- ④サンプルプログラムの終了(デバイスクローズ後)



(注)②デバイスオープンと③デバイスクローズ処理において、全出力ポート(1~5)には'0'を書込みます。(出力ポートのクリア)

8.1.2 ボード上の操作と表示

デバイスオープンを行いますと次の画面となります。



(注)上記画面はDIO548の画面です。(24in/24out)
入力・出力部のポート4と5はすべて読込値が'1'となっています。

8.2 サンプルプログラムの変更について

5種類のサンプルプログラムは、**Visual C++ 6.0**、**Visual Basic 6.0**、**Visual Basic .NET2002**、**Visual C# .NET2003** を使用しています。

サンプルプログラムを変更する場合には、上記開発環境がパソコンにインストールされている事が前提条件です。

個々のディレクトリ(フォルダ)には「実行可能ファイル(.exe)」を含めて、新規の実行可能ファイルを作成する為の全てのファイルが格納されています。

ファイル種類	開発環境・コーディング区分				
	Visual C++ (C)	Visual C++ (MFC)	Visual Basic 6.0	Visual Basic .NET	Visual C# .NET
実行可能ファイル	spd58000.exe	spd58001.exe	spd58002.exe	spd58003.exe	spd58004.exe
ソリューションファイル	—	—	—	spd58003.sln	spd58004.sln
プロジェクトファイル	spd58000.dsw spd58000.dsp spd58000.opt	spd58001.dsw spd58000.dsp spd58000.opt	spd58002.vbp	spd58003.vbproj spd58003.vbproj.user	spd58004.vbproj spd58004.vbproj.user
ドライバI/Fファイル	hidio580.dll hidio580.lib hidio580.h	hidio580.dll hidio580.lib hidio580.h	hidio580.dll	hidio580.dll	hidio580.dll
ソースプログラム	上記以外	上記以外	上記以外	上記以外	上記以外

8.2.1 プロジェクトファイル(ソリューションファイル)

- (1) Visual C++(C)版サンプルプログラムを変更する場合
プロジェクトワークスペースは、spd58000.dswを選択して下さい。
Visual C++ 6.0以上 がインストールされている必要があります。
- (2) Visual C++(MFC)版サンプルプログラムを変更する場合
プロジェクトワークスペースは、spd58001.dswを選択して下さい。
Visual C++ 6.0以上 がインストールされている必要があります。
- (3) Visual Basic版サンプルプログラムを変更する場合
プロジェクトファイルは、spd58002.vbpを選択して下さい。
Visual Basic 6.0 がインストールされている必要があります。
- (4) Visual Basic.NET 2002版サンプルプログラムを変更する場合
ソリューションファイルは、spd58003.slnを選択して下さい。
Visual Basic .NET2002以上 がインストールされている必要があります。
- (5) Visual C#.NET 2003版サンプルプログラムを変更する場合
ソリューションファイルは、spd58004.slnを選択して下さい。
Visual C# .NET2003以上 がインストールされている必要があります。
- (6) 開発環境のバージョンが異なる場合
Visual Basic 5.0, Visual C++ 5.0 等を使用している場合には、そのままプロジェクトファイルが使用出来ません。
この場合には、新規のプロジェクトを作成し、このプロジェクトにサンプルプログラムを構成する各種のファイルを追加して下さい。
なお、開発言語の種類により、サンプルプログラムで使用している機能が使用できない事があります。
ソースプログラムを削除してご使用下さい。

[例] Visual C++ 5.0で"C++"サンプルをビルドする場合
stdafx.h ... #include <afxdtctl.h> の1行をコメントとします。

8.2.2 サンプルプログラムを構成する関数

サンプルプログラムを構成する関数を大別すると次のようになります。

分類	No	関 数 名	関 数 の 処 理 (ボタン対応)
画面表示の処理関数	1	WinMain()	Windows のメイン関数
	2	WndProc()	Windows から呼び出されて、メッセージキューからメッセージの引き渡しを受ける
	3	FormLoad1()	ウインドウを中央へ移動する, ハンドル取得
	4	FormLoad3()	コンボボックスの初期設定
	5	FormUnload()	ウインドウを閉じる
	6	CmbJoho_Click()	デバイス情報選択 & 表示(コンボボックスのクリック)
	7	BtnOpen2_Click()	デバイスオープンボタンのクリック
	8	BtnClose1_Click()	デバイスクローズボタンのクリック(クローズ前)
	9	BtnClose3_Click()	デバイスクローズボタンのクリック(クローズ後)
	10	ErrMsg()	エラーメッセージ出力(サンプルプログラムの終了)
ボード制御	11	FormLoad2()	デバイス情報取得
	12	BtnOpen1_Click()	デバイスオープンボタンのクリックでボード初期化
	13	BtnClose2_Click()	デバイスクローズボタンのクリックでデバイスクローズ
	14	BtnOut_Click()	出力ポートボタンのクリックで ON/OFF 切替え
	15	TmrInOut()	入力ポートと出力ポート表示(動作中軸:0.1 秒毎)

9. DI580 用サンプルプログラム

ドライバ\I/F用DLLの各関数の使用方法を解説する目的のサンプルプログラムを添付しています。

(1) Visual C++ 6.0・C コーディング 【 spdi5800.exe 】

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(*.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。

《 ご注意 》

- (1) Visual Basic サンプルは開発ツールとして「Visual Basic 6.0」がインストールされている必要があります。
- (2) 実行開始時に次のエラーメッセージが表示される場合には、サンプルプログラムは動作しません。



【 エラーメッセージの表示 】

- ※ HPCI-DIO5xxボードが未搭載。
- ※ デバイスドライバがインストールされていない。

9.1 サンプルプログラムの操作

サンプルプログラムが起動され、1枚以上のボード(デバイス)が正常に認識される時、次の画面が表示されます。



9.1.1 ボード(デバイス)の選択

サンプルプログラムでは、ボード上の動作操作開始・終了は次の手順に従います。

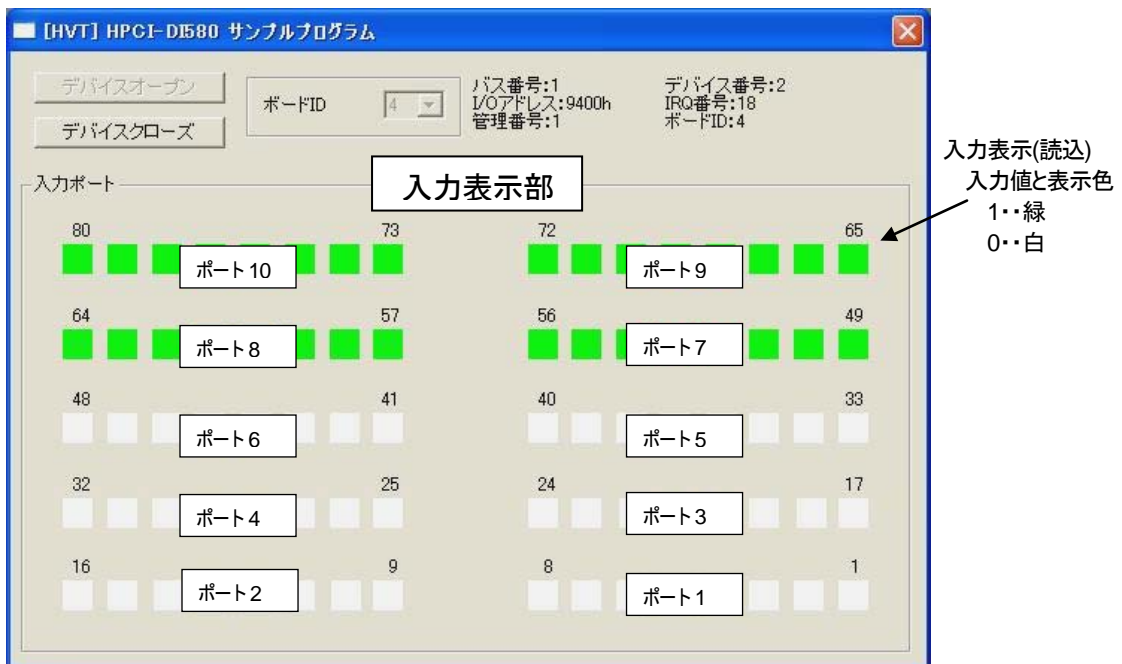
- ①ボードの選択(2枚以上の場合)
- ②デバイスオープン指令



- ③デバイスクローズ指令
- ④サンプルプログラムの終了(デバイスクローズ後)

9.1.2 ボード上の操作と表示

デバイスオープンを行いますと次の画面となります。



9.2 サンプルプログラムの変更について

サンプルプログラムは、Visual C++ 6.0 を使用しています。

サンプルプログラムを変更する場合には、上記開発環境がパソコンにインストールされている事が前提条件です。

個々のディレクトリ(フォルダ)には「実行可能ファイル(.exe)」を含めて、新規の実行可能ファイルを作成する為の全てのファイルが格納されています。

	開発環境・コーディング区分
ファイル種類	Visual C++ (C)
実行可能ファイル	spdi5800.exe
ソリューションファイル	—
プロジェクトファイル	spdi5800.dsw spdi5800.dsp spdi5800.opt
ドライバ\Fファイル	hidio580.dll hidio580.lib hidio580.h
ソースプログラム	上記以外

9.2.1 プロジェクトファイル(ソリューションファイル)

プロジェクトワークスペースは、spdi5800.dswを選択して下さい。

Visual C++ 6.0以上 がインストールされている必要があります。

■開発環境のバージョンが異なる場合

Visual Basic 5.0, Visual C++ 5.0 等を使用している場合には、そのままプロジェクトファイルが使用出来ません。

この場合には、新規のプロジェクトを作成し、このプロジェクトにサンプルプログラムを構成する各種のファイルを追加して下さい。

なお、開発言語の種類により、サンプルプログラムで使用している機能が使用できない事があります。

ソースプログラムを削除してご使用下さい。

[例] Visual C++ 5.0で"C++"サンプルをビルドする場合

stdafx.h ... #include <afxdtctl.h> の1行をコメントとします。

9.2.2 サンプルプログラムを構成する関数

サンプルプログラムを構成する関数を大別すると次のようになります。

分類	No	関 数 名	関 数 の 処 理 (ボタン対応)
画面表示の処理関数	1	WinMain()	Windows のメイン関数
	2	WndProc()	Windows から呼び出されて、メッセージキューからメッセージの引き渡しを受ける
	3	FormLoad()	ウインドウを中央へ移動する、ハンドル取得、コンボボックスの初期設定
	4	FormUnload()	ウインドウを閉じる
	5	CmbSelCng()	デバイス情報選択 & 表示(コンボボックスのクリック)
	6	BtnClose()	デバイスクローズボタンのクリック
	7	ErrMsg()	エラーメッセージ出力(サンプルプログラムの終了)
ボード制御	8	GetDevInfo()	デバイス情報取得
	9	BtnOpen()	デバイスオープンボタンのクリックでボード初期化
	10	DevClose()	デバイスクローズボタンのクリックでデバイスクローズ
	11	TmrIn()	入力ポート表示(動作中軸:0.1 秒毎)

10. DO580 用サンプルプログラム

ドライバ/F用DLLの各関数の使用方法を解説する目的のサンプルプログラムを添付しています。

(1) Visual C++ 6.0 C コーディング 【 spdo5800.exe 】

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(*.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。

《 ご注意 》

(1) Visual Basic サンプルは開発ツールとして「Visual Basic 6.0」がインストールされている必要があります。

(2) 実行開始時に次のエラーメッセージが表示される場合には、サンプルプログラムは動作しません。



【 エラーメッセージの表示 】

※ HPCI-DIO5xxボードが未搭載。

※ デバイスドライバがインストールされていない。

10.1 サンプルプログラムの操作

サンプルプログラムが起動され、1枚以上のボード(デバイス)が正常に認識される時、次の画面が表示されます。



10.1.1 ボード(デバイス)の選択

サンプルプログラムでは、ボード上の動作操作開始・終了は次の手順に従います。

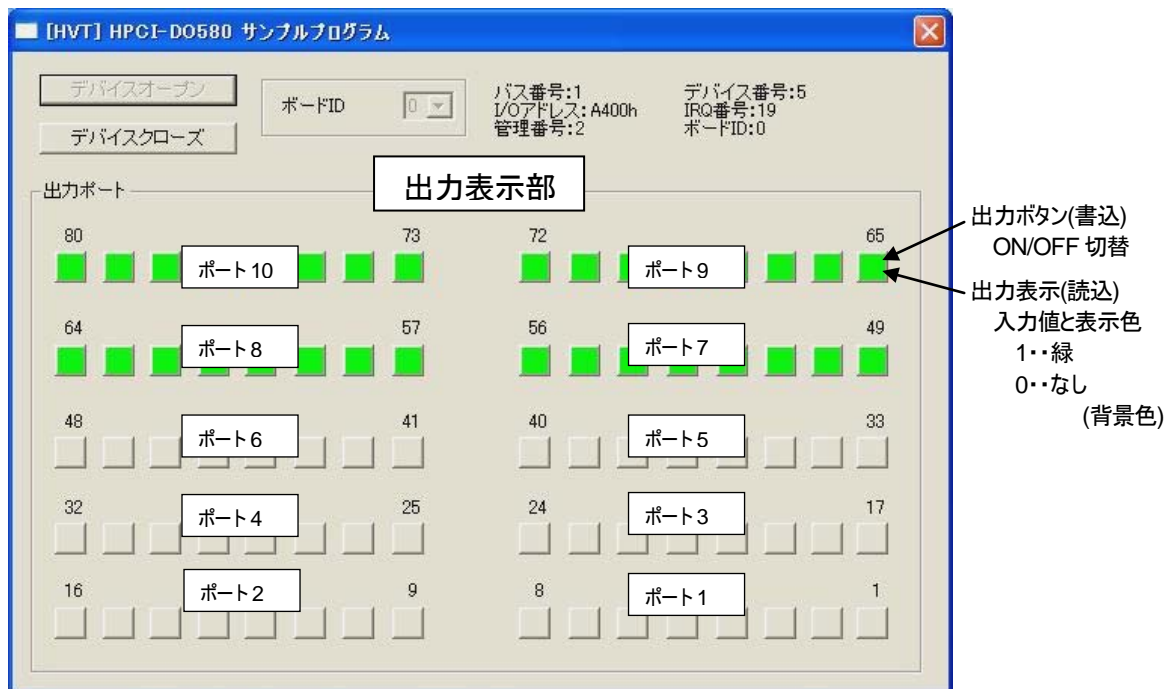
- ①ボードの選択(2枚以上の場合)
- ②デバイスオープン指令



- ③デバイスクローズ指令
- ④サンプルプログラムの終了(デバイスクローズ後)

10.1.2 ボード上の操作と表示

デバイスオープンを行いますと次の画面となります。



10.2 サンプルプログラムの変更について

サンプルプログラムは、Visual C++ 6.0 を使用しています。

サンプルプログラムを変更する場合には、上記開発環境がパソコンにインストールされている事が前提条件です。

個々のディレクトリ(フォルダ)には「実行可能ファイル(.exe)」を含めて、新規の実行可能ファイルを作成する為の全てのファイルが格納されています。

	開発環境・コーディング区分
ファイル種類	Visual C++ (C)
実行可能ファイル	spdo5800.exe
ソリューションファイル	—
プロジェクトファイル	spdo5800.dsw spdo5800.dsp spdo5800.opt
ドライバ\Fファイル	hidio580.dll hidio580.lib hidio580.h
ソースプログラム	上記以外

10.2.1 プロジェクトファイル(ソリューションファイル)

プロジェクトワークスペースは、spdo5800.dswを選択して下さい。

Visual C++ 6.0以上 がインストールされている必要があります。

■開発環境のバージョンが異なる場合

Visual Basic 5.0, Visual C++ 5.0 等を使用している場合には、そのままプロジェクトファイルが使用出来ません。

この場合には、新規のプロジェクトを作成し、このプロジェクトにサンプルプログラムを構成する各種のファイルを追加して下さい。

なお、開発言語の種類により、サンプルプログラムで使用している機能が使用できない事があります。

ソースプログラムを削除してご使用下さい。

[例] Visual C++ 5.0で"C++"サンプルをビルドする場合

stdafx.h ... #include <afxdtctl.h> の1行をコメントとします。

10.2.2 サンプルプログラムを構成する関数

サンプルプログラムを構成する関数を大別すると次のようになります。

分類	No	関 数 名	関 数 の 処 理 (ボタン対応)
画面表示の処理関数	1	WinMain()	Windows のメイン関数
	2	WndProc()	Windows から呼び出されて、メッセージキューからメッセージの引き渡しを受ける
	3	FormLoad	ウィンドウを中央へ移動する、ハンドル取得、コンボボックスの初期設定
	4	FormUnload()	ウィンドウを閉じる
	5	CmbSelCng()	デバイス情報選択 & 表示(コンボボックスのクリック)
	6	BtnClose()	デバイスクローズボタンのクリック
	7	ErrMsg()	エラーメッセージ出力(サンプルプログラムの終了)
ボード制御	8	GetDevInfo()	デバイス情報取得
	9	BtnOpen	デバイスオープンボタンのクリックでボード初期化
	10	DevClose()	デバイスクローズボタンのクリックでデバイスクローズ
	11	BtnOut()	出力ポートボタンのクリックで ON/OFF 切替え
	12	TmrOut()	出力ポート表示(動作中軸:0.1 秒毎)

■ 更新履歴

版	更新内容	備考
第1版	1. 新規作成	
第2版	<p>1. DIO548ボードにボードID(0~15まで任意に設定可)がサポートされました。ボードIDは、ボード上のジャンパで決定され、ソフトでこの値を指定したデバイス(ボード)指定が可能となりました。</p> <p>デバイス情報に"ボードID"の項目が追加されました。</p> <p>[C言語: Windows: Visual C++]</p> <pre>typedef struct _HPCDEVICEINFO { DWORD nBusNumber; /* バス番号 */ DWORD nDeviceNumber; /* デバイス番号(PCIスロット番号)*/ DWORD dwIoPortAddress; /* I/Oポートアドレス */ DWORD dwIrqNo; /* IRQ番号 */ DWORD dwNumber; /* 管理番号 */ ➡ DWORD dwBoardID; /* ボードID(0~15) */ } HPCDEVICEINFO, *PHPCDEVICEINFO</pre> <p>[Windows: Visual Basic]</p> <pre>Public Type HPCDEVICEINFO nBusNumber As Long 'バス番号 nDeviceNumber As Long 'デバイス番号 dwIoPortAddress As Long 'I/Oポートアドレス dwIrqNo As Long 'IRQ番号 dwNumber As Long '管理番号 ➡ dwBoardID As Long 'ボードID(0~15) End Type</pre> <p>2. ドライバ/IF用DLL関数の追加 ボードID追加による関数の追加, DIO548ボードの入出力数確認用関数の追加です。</p> <p>(1) デバイスのオープン(ボードID参照) hdio580_OpenDeviceID() (2) 入出力数切替ポート読込 hdio580_ioswRead()</p> <p>3. ドライバ/IF用DLL関数の引数と戻り値のデータ型</p> <p>(1) Visual C++ "BOOL"・"INT"の定義を"DWORD(unsigned long)"としました。 戻り値の値は従来通りです。</p> <p>(2) Visual Basic "Boolean" の定義を "Long"としました。戻り値の値は従来通りです。</p> <p>4. C言語用ヘッダーファイルの統一 ドライバ/IF用 DLL結合用ヘッダーファイルを1つとしました。</p> <p>5. サンプルプログラムの充実</p> <p>(1) サンプルプログラムに「C++(MFC)」コーディング例を追加しました。 (2) 開発環境のバージョンアップ "Visual C++(5.0以上)", "Visual Basic(5.0以上)" →"Visual C++(6.0以上)", "Visual Basic 6.0"</p>	<p>注. DIO580 ボードでは "15"固定となります。</p> <p>(1) ボードID 0~15 (2) DIO548ボード用</p>
第3版	1. WindowsXPに対応しました。	
第4版	<p>1. DIO580Aボードがリリースされました。HPCI-DIO580AではボードIDがサポートされます。</p> <p>2. Windows Vistaに対応しました。</p> <p>3. Visual Basic.NETのサンプルプログラムを追加しました。</p> <p>4. VC, C++(MFC), VB6.0の各サンプルプログラムの仕様を一部変更しました。</p>	
第4.1版	1. hdio580_intFilterWrite(), hdio580_intFilterRead(), hdio580_intEdgeWrite(), hdio580_intEdgeRead(), hdio580_intLdtSelWrite(), hdio580_intLdtSelRead(), hdio580_intLdtStsRead()の7種の関数を追加しました。	
第5版	<p>1. Windows 7に対応しました。</p> <p>2. 64ビットWindowsに対応しました。</p> <p>3. Windows Vista のインストール方法を変更しました。</p> <p>4. Visual C#のサンプルプログラムを追加しました。</p>	
第6版	<p>1. HPCI-DI580, HPCI-DO580 に対応しました。</p> <p>2. ドライバ/IF用 DLL 関数 hdio580_rPortW(), hdio580_wPortW を追加しました。</p>	