HPCI-DIO5144

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I/F用ライブラリ

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

Windows 版デバイスドライバ の種別は

Windows XP, Windows Vista 32 ビット, Windows 7 32 ビット, Windows 8, Windows 10 32 ビットでは
hd5144a.sys
Windows Vista 64 ビット, Windows 7 64 ビット, Windows 8 64 ビット, Windows 10 64 ビットでは
hd5144b.sys です.

Windows 32 ビット 版ドライバI/F用ライブラリ として
hd5144.dll を使用します.

Windows 64 ビット 版 32 ビットアプリケーション用ドライバI/F用ライブラリ として
hd5144.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I/F用ライブラリ として
hd5144.dll(64 ビットアプリケーション用)を使用します.

株式会社 ハイバーテック
東京都江東区新大橋 1-8-11
大樹生命新大橋ビル 6F
TEL 03-3846-3801
FAX 03-3846-3773
sales@hivertec.co.jp

1.20 版 2019 年 4 月 18 日発行

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
1.6 ハードウェアの設定・取付け・接続 3
1.7 サンプルプログラム作成ツール 3
1.8 サンプルプログラムの実行 3
1.9 ユーザープログラムの作成 4
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 6
3.1 Windows 10, 8, 7, Vista(32 ビット)へのインストール 6
3.2 Windows 10, 8, 7, Vista(64 ビット)へのインストール 6
3.3 Windows XP へのインストール 6
3.4 アンインストール 6
3.3.1 Windows XP の場合 6
3.3.2 Windows 10, 8, 7, Vista の場合 6
4. ボードを複数枚利用する場合 7
4.1 ボードのデバイス番号の確認 7
4.2 デバイス番号の確認方法 7
4.3 DIO5144 ボードでのボード ID の使用 7
5. ドライバ I/F 用 DLL の使用方法 8
5.1 概　要 8
5.2 関数一覧 8
5.3 準　備 9
6. 制御概念 10
6.1 ボード（デバイス）認識用のデータ構造体 10
6.2 ボードアクセスの準備手順 12
6.3 関数の戻り値 13
7. 関数詳細 14
8. サンプルプログラム 20
8.1 サンプルプログラムの操作 20
8.1.1 ボード（デバイス）の選択 21
8.1.2 ボード上の操作と表示 21
8.1.3 設定 22
8.1.4 ラッチ機能 23
8.1.5 トランスファー接点機能 23
8.1.6 連続データ読込み機能 24
8.1.7 PWM 出力機能 25
8.1.8 カウンタ入力機能 25
8.2 サンプルプロジェクト 26

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
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)
Windows 8 の各エディション (以降 **Win8** と記します)
Windows 10 の各エディション (以降 **Win10** 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

[製品名]の標準添付ソフトウェアは Windows10, Windows 8.1, Windows 7 SP1 に対応しております.
マイクロソフト社 OS サポートのライフサイクル期間が終了した OS の対応については, マイクロソフト社 OS サポートのライフサイクル期間に確認したものであり, 本マニュアル発行時点での動作を保証するものではありません.

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よくお読みいただき, 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

[製品名] のサンプルプログラムは Microsoft Visual Studio のプロジェクト及びソースコードを添付しています.
マイクロソフト社製品サポートのライフサイクル期間が終了した Microsoft Visual Studio の各バージョンについては, マイクロソフト社製品サポートのライフサイクル期間に確認したものであり, 本マニュアル発行時点での動作を保証するものではありません.

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
⚠ 本製品に添付されるサンプルプログラムまたはマニュアル内のコード例は，本製品のソフトウェア・ボードの機能・動作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・エラー処理・例外処理・排他処理等は省略されています．実際にプログラムを作成する場合は，十分に安全対策等を考慮し，必要な処理を追加してください．

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

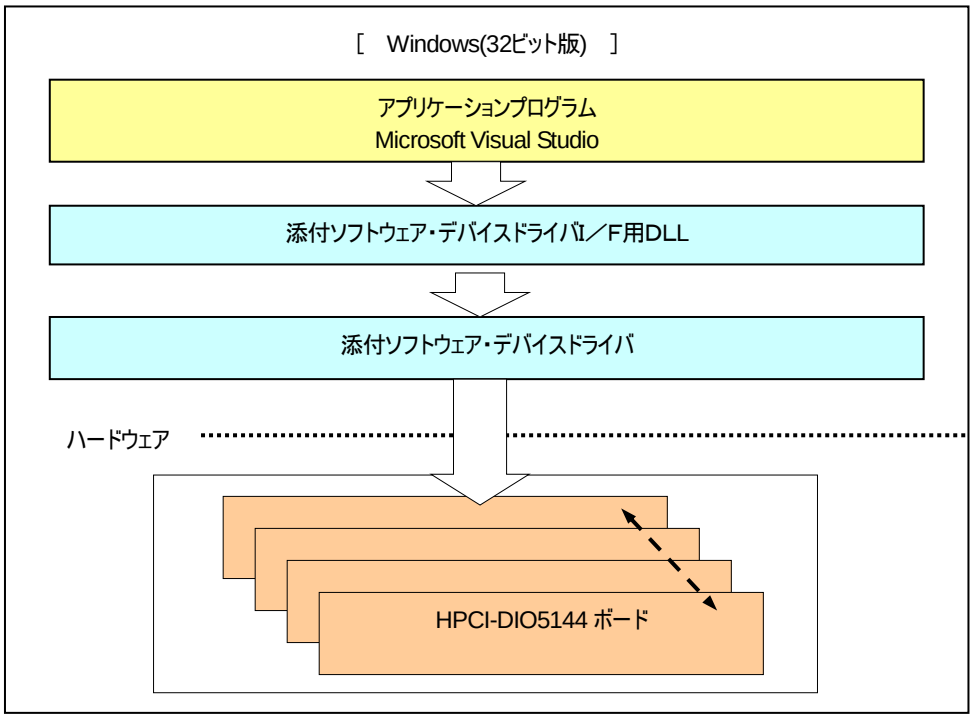


図 2.1-1 32 ビット版 OS 添付ソフトウェア関連図

2.1.2　64 ビット版 OS

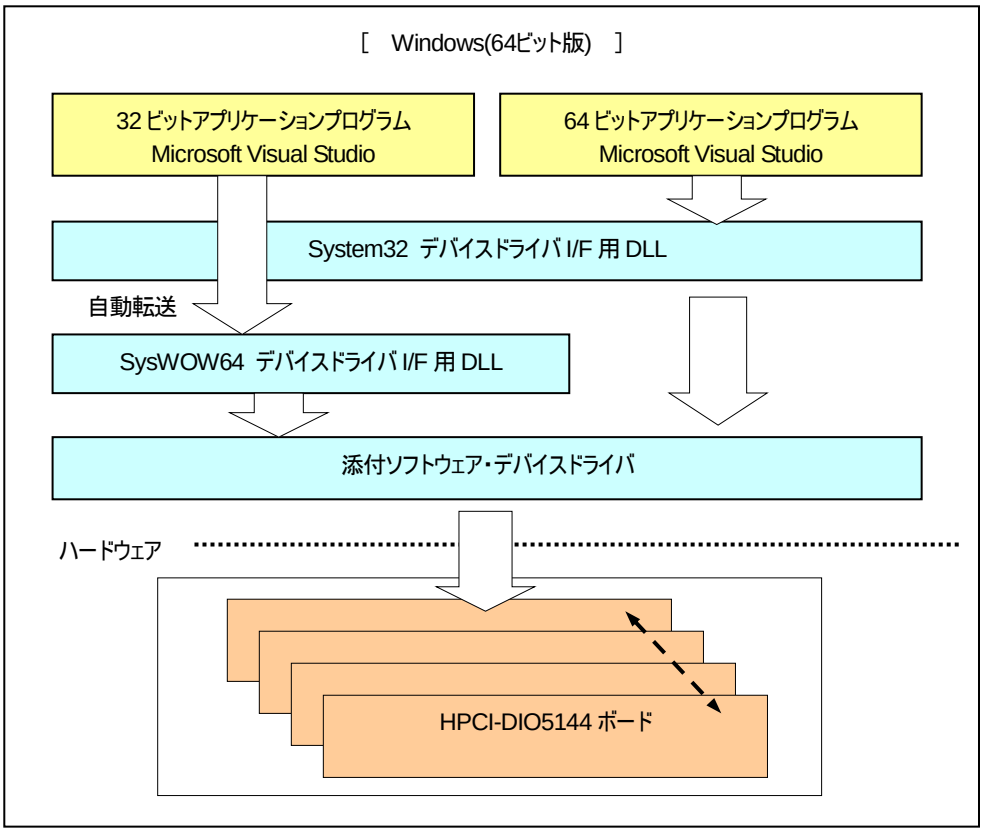


図 2.1-2　64 ビット版 OS　添付ソフトウェア関連図

2.2 添付ソフトウェアの内容

添付 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 Windows 10, 8, 7, Vista(32 ビット)へのインストール

(1) HPCI-DIO5144 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**:¥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-DIO5144 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 10, 8, 7, Vista(64 ビット)へのインストール

(1) HPCI-DIO5144 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**:¥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-DIO5144 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) HPCI-DIO5144 をパソコンの PCI バススロットに装着し，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**:¥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，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.

3.4 アンインストール

3.3.1 Windows XP の場合

Windows の「スタート」→「コントロールパネル」→「プログラムの追加と削除」
→「Windows ドライバ パッケージ Hivertec HPCI-DIO5144」を右クリックしアンインストールを行います.

3.3.2 Windows 10, 8, 7, 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-DIO5144」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

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．

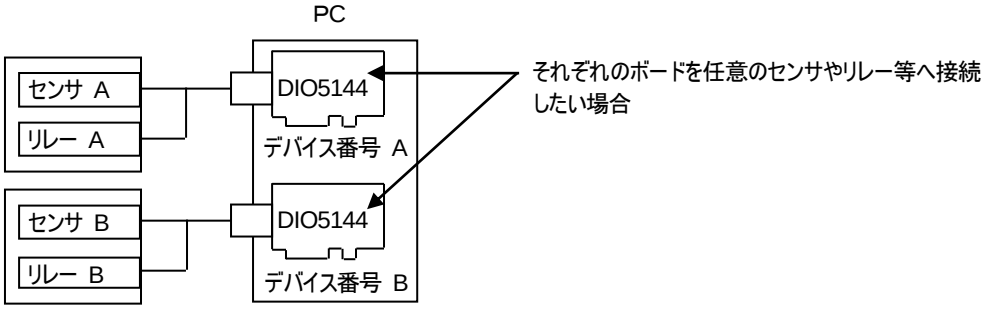


図 4.1-1 ボードを複数枚使用する場合

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．

「 ¥sample¥vc¥Release¥spd51440.exe」

このプログラムは，DIO5144 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，DIO5144 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，「7. 関数詳細 (2)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．

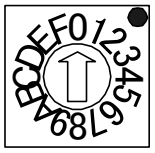


図 4.3-1 HPCI-DIO5144 ボード ID ロータリーディップ SW

5. ドライバ I/F 用 DLL の使用方法

5.1 概　要

ドライバ I/F 用 DLL は， Windows において，HPCI-DIO5144 ボードの制御を行うための関数群です.

各関数は"Visual C++6.0 以上"，"Visual Basic 6.0/ .NET"，"Visual C#"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-DIO5144 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 WindowsXP，Vista, 7, 8 , 10 (32 ビット版)用に hd5144a.sys ，
Windows Vista, 7, 8 , 10 (64 ビット版)用に hd5144b.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 6 種類 8 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_GetDeviceInfo()	デバイス情報の取得
3	hdio5144_OpenDevice()	デバイスのオープン(ボード ID 無視)
4	hdio5144_CloseDevice()	デバイスのクローズ
5	hdio5144_rPortB()	ポートバイト読込
	hdio5144_rPortW()	ポートワード読込
6	hdio5144_wPortB()	ポートバイト書込
	hdio5144_wPortW()	ポートワード書込

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.

・hd5144.lib ・・ インポートライブラリ

・hd5144.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

「hd5144.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

「hd5144.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

「hd5144.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-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.)

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#]

```
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
    /// </summary>
    public uint dwNumber;

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

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-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.

- ◆ 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.

- ◆ hdio5144_CloseDevice() ・・・ボードのクローズ処理(1 枚分)

6.3 関数の戻り値

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，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		
		VC++ DOS	VB VB.NET	
1	NO_ERROR	0x000	&H0	正常 異常は発生していません
2	NOT_FOUND	0x001	&H1	デバイスドライバが存在しない ◎デバイスドライバがインストールされていない ◎デバイスドライバが所定のフォルダに格納されていない
3	ALREADY_OPENED	0x002	&H2	既にオープン済のデバイスをオープン ◎オープン済みデバイスに更にオープン指令 ◇オープンしたデバイスはクローズするまで使用 （多重のオープンは禁止） ◎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	INSUFFICIENT_MEMORY	0x004	&H4	デバイス情報格納メモリが不足 ◎アプリケーション用のメモリ不足 ◇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 ◇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
5	INVALID_HANDLE	0x008	&H8	無効なデバイスハンドルを指定 ◎デバイスオープンで得られた"デバイスハンドル"の不使用 ◎このデバイスは既にクローズされている
6	NOT_READY	0x010	&H10	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	0x020	&H20	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_ADDRESS	0x040	&H40	不正なベースアドレス（HPCボード） ◎指定したベースアドレスの確認
9	ILLEGAL_PARAM	0x100	&H100	関数の引数の値が異常 ◇速度倍率設定値の範囲は2～4095． ◇その他引数の設定値を確認（マニュアル照合）

7. 関数詳細

関数説明文中で 16 進数の表記は C 言語記述で行っています.
Visual Basic でご利用の場合には, C 言語 16 進数表記を次のように読み替えて下さい.

C言語(C#)	Visual Basic
0x00000000	&H0
0x20001000	&H20001000

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

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

《書式》
[C言語: Visual C++]

```
DWORD WINAPI hdio5144_GetDeviceCount (DWORD* count);
```

[Visual Basic 6.0]

```
Declare Function hdio5144_GetDeviceCount Lib "hd5144.dll" _
                                        （ByRef count As Long）As Long
```

[Visual Basic .NET]

```
Declare Function hdio5144_GetDeviceCount Lib "hd5144.dll" _
                                        (ByRef count As Short) As Integer
```

[Visual C#.NET]

```
[DllImport ("hd5144.dll")] public static extern uint hdio5144_GetDeviceCount(ref unit count);
```

《引数》count ・・ 取得したCPDボードの枚数

《呼び出し例》
[C言語: Visual C++]

```
DWORD  count;        // ボードの枚数
DWORD  ret;          // 関数の戻り値
ret = hdio5144_GetDeviceCount( &count );
```

[Visual Basic 6.0]

```
Dim count   As Long
Dim ret     As Long
ret = hdio5144_GetDeviceCount( count )
```

[Visual Basic .NET]

```
Dim count   As Integer
Dim ret     As Integer
ret = hdio5144_GetDeviceCount( count )
```

[Visual C# .NET]

```
uint  count;
uint  ret;
ret = Hd5144.hdio5144_GetDeviceCount( ref count );
```

(2) hdio5144_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-DIO5144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 hdio5144_GetDeviceInfo (DWORD* count, HPCDEVICEINFO* HpcDeviceInfo );
```

[Visual Basic 6.0]

```
Declare Function hdio5144_GetDeviceInfo Lib "hd5144.dll" _
                    (ByRef count As Long, HpcDeviceInfo As HPCDEVICEINFO) As Long
```

[Visual Basic .NET]

```
Declare Function hdio5144_GetDeviceInfo Lib "hd5144.dll" _
                    ( ByRef count As Integer, ByRef pHpcDevInfo As HPCDEVICEINFO) As Integer
```

[Visual C# .NET]

```
[DllImport("hd5144.dll")]
public static extern uint hdio5144_GetDeviceInfo(ref int count, ref HPCDEVICEINFO pHpcDevInfo);
```

《引数》

count　・・ 取得したCPDボードの枚数

HpcDeviceInfo ・・ 取得するデバイス情報

《呼び出し例》 パソコンにHPCI-DIO5144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-DIO5144のデバイス情報格納配列
ret = hdio5144_GetDeviceInfo( &count,  &HpcDeviceInfo[0] );
```

[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-DIO5144のデバイス情報格納配列
count = 2                                       ' 最大枚数は2
ret = hdio5144_GetDeviceInfo( count, HpcDeviceInfo(0) )
```

[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-DIO5144のデバイス情報格納配列
count = 2                                       ' 最大枚数は2
ret = hdio5144_GetDeviceInfo( count, HpcDeviceInfo(0) )
```

[Visual C# .NET]

```
uint    ret;                    // 関数の戻り値
uint    count = 2;              // 最大枚数は2
// 2枚のHPCI-DIO5144のデバイス情報格納配列
Hd5144.HPCDEVICEINFO[] HpcDevInfo = new Hd5144.HPCDEVICEINFO[2];
 ret = Hd5144.hdio5144_GetDeviceInfo( ref count, ref HpcDeviceInfo[0] );
```

(3) 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HPCI-DIO5144ボードをオープンし，他のHPCI-DIO5144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-DIO5144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 hdio5144_OpenDevice( PHPCDEVICEINFO*  HpcDeviceInfo );
```

[Visual Basic 6.0]

```
Declare Function hdio5144_OpenDevice Lib "hd5144.dll" _
                    (HpcDeviceInfo As HPCDEVICEINFO) As Long
```

[Visual Basic .NET]

```
Declare Function hdio5144_OpenDevice Lib "hd5144.dll" _
                    (ByRef HpcDeviceInfo As HPCDEVICEINFO) As Integer
```

[Visual C# .NET]

```
 [DllImport("hd5144.dll")]
public static extern uint hdio5144_OpenDevice(ref HPCDEVICEINFO pHpcDevInfo);
```

《引数》

HpcDeviceInfo ‥ デバイスの情報

《呼び出し例》

パソコンにHPCI-DIO5144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hdio5144_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    ret;                    // 関数の戻り値
DWORD    hDevID[2]               //デバイスハンドル取得エリア
ret = hdio5144_OpenDevice( &hDevID[0] ,&HpcDeviceInfo[0] );        //1枚目のボードをオープン
ret = hdio5144_OpenDevice( &hDevID[1], &HpcDeviceInfo[1] );        //2枚目のボードをオープン
```

[Visual Basic 6.0]

```
Dim    ret                 As Long                 ' 関数の戻り値
Dim    hDevID(2)           As Long                 'デバイスハンドル取得エリア
ret = hdio5144_OpenDevice( hDevID(0), HpcDeviceInfo(0) )     '1枚目のボードをオープン
ret = hdio5144_OpenDevice( hDevID(1), HpcDeviceInfo(1) )     '2枚目のボードをオープン
```

[Visual Basic .NET]

```
Dim ret             As Integer                  ' 関数の戻り値
Dim hDevID(2)    As Integer                     'デバイスハンドル取得エリア
ret = hdio5144_OpenDevice( hDevID(0), HpcDeviceInfo(0) )    '1枚目のボードをオープン
ret = hdio5144_OpenDevice( hDevID(1), HpcDeviceInfo(1) )    '2枚目のボードをオープン
```

[Visual C# .NET]

```
uint     ret;                              // 関数の戻り値
uint[]   hDevID = new    uint[2];         //デバイスハンドル取得エリア
ret = Hd5144. hdio5144_OpenDevice( ref hDevID[0], ref HpcDeviceInfo[0] );     //1枚目のボードをオープン
ret = Hd5144. hdio5144_OpenDevice( ref hDevID[1], ref HpcDeviceInfo[1] );     //2枚目のボードをオープン
```

（4）	**hdio5144_CloseDevice()　　デバイスのクローズ**

《機能》

デバイスハンドルで指定されたHPCI-DIO5144ボードをクローズします.
以降, このデバイスハンドルは無効となります.

《書式》

[C言語: Visual C++]

```
DWORD WINAPI hdio5144_CloseDevice( DWORD   hDevID );
```

[Visual Basic 6.0]

```
Declare Function hdio5144_CloseDevice Lib "hd5144.dll" (ByVal hDevID As Long) As Long
```

[Visual Basic .NET]

```
Declare Function hdio5144_CloseDevice Lib "hd5144.dll" (ByVal hDevID As Integer) As Integer
```

[Visual C# .NET]

```
[DllImport("hd5144.dll")] public static extern uint hdio5144_CloseDevice(int hDevID);
```

《引数》

hDevID ・・ クローズするボードのデバイスハンドル.

《呼び出し例》

[C言語: Visual C++]

```
DWORD ret;                    //関数の戻り値
ret = hdio5144_CloseDevice( hDevID );
```

[Visual Basic 6.0]

```
Dim ret As Long               '関数の戻り値
ret = hdio5144_CloseDevice( hDevID )
```

[Visual Basic .NET]

```
Dim ret As Integer            '関数の戻り値
ret = hdio5144_CloseDevice( hDevID )
```

[Visual C# .NET]

```
uint    ret;                  //関数の戻り値
ret = Hd5144.hdio5144_CloseDevice( hDevID );
```

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

《機能》

デバイスハンドルで指定されたHPCI-DIO5144の，portで指定されたアドレスのポートから

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》

［ C言語: Visual C++ ］

```
DWORD WINAPI hdio5144_rPortB (DWORD hDevID, WORD port, BYTE* bytin);
DWORD WINAPI hdio5144_rPortW (DWORD hDevID, WORD port, WORD* wrdin);
```

［ Visual Basic 6.0 ］

```
Declare Function hdio5144_rPortB Lib "hd5144.dll"
                        (ByVal hDevID As Long, ByVal port As Integer,
                         ByRef bytin As Byte) As Long
Declare Function hdio5144_rPortW Lib "hd5144.dll"
                        (ByVal hDevID As Long, ByVal port As Integer,
                         ByRef wrdin As Integer) As Long
```

［ Visual Basic .NET ］

```
Declare Function hdio5144_rPortB Lib "hd5144.dll"
                        (ByVal hDevID As Integer, ByVal port As Short,
                         ByRef bytin As Byte) As Long
Declare Function hdio5144_rPortW Lib "hd5144.dll"
                        (ByVal hDevID As Integer, ByVal port As Short,
                         ByRef wrdin As Short) As Long
```

［ Visual C# .NET ］

```
 [DllImport("hd5144.dll")]
public static extern uint hdio5144_rPortB(int hDevID, short port, ref byte byInp);
 [DllImport("hd5144.dll")]
public static extern uint hdio5144_irPortW(int hDevID, short port, ref short wInp);
```

《引数》

hDevID・・・ 対象デバイスのデバイスハンドル

port　・・・ ポートアドレス

bytin　・・・ 読込んだデータ

wrdin　・・・ 読込んだデータ

《呼び出し例》

［ C言語: Visual C++ ］

```
DWORD ret;                  //関数の戻り値
BYTE   bytin;               //格納先
 ret = hdio5144_rPortB( hDevID , 0x35, &bytin );     //ラッチ信号ステータスポートをバイト読込み
```

［ Visual Basic 6.0 ］

```
Dim ret          As Long        '関数の戻り値
Dim bytin        As Byte        '格納先
 ret = hdio5144_rPortB( hDevID, &H35, bytin )        'ラッチ信号ステータスポートをバイト読込み
```

［ Visual Basic .NET ］

```
Dim ret          As Integer        '関数の戻り値
Dim wrdin         As Short         '格納先
ret = hdio5144_rPortW( hDevID, &H35, wrdin )        'ラッチ信号ステータスポートをワード読込み
```

［ Visual C# .NET ］

```
uint    ret;                //関数の戻り値
byte   bytin;               //格納先
ret = Hd5144.hdio5144_rPortW( hDevID , &H35, ref wrdin );     //ラッチ信号ステータスポートをワード読込み
```

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

《機能》

デバイスハンドルで指定されたHPCI-DIO5144の，portで指定されたアドレスのポートへ

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》

[C言語: Visual C++]

```
DWORD WINAPI hdio5144_outpWriteB (DWORD hDevID, WORD portno, BYTE    bytdt);
DWORD WINAPI hdio5144_outpWriteW (DWORD hDevID, WORD portno, WORD    wrddt);
```

[Visual Basic 6.0]

```
Declare Function hdio5144_outpWriteB Lib "hd5144.dll" _
                        (ByVal hDevID As Long, ByVal portno As Integer, _
                         ByVal bytdt As Byte) As Long
Declare Function hdio5144_outpWriteW Lib "hd5144.dll" _
                        (ByVal hDevID As Long, ByVal portno As Integer, _
                         ByVal wrddt As Integer) As Long
```

[Visual Basic .NET]

```
Declare Function hdio5144_outpWriteB Lib "hd5144.dll" _
                        (ByVal hDevID As Integer, ByVal portno As Short, _
                         ByVal bytdt As Byte) As Long
Declare Function hdio5144_outpWriteW Lib "hd5144.dll" _
                        (ByVal hDevID As Integer, ByVal portno As Short, _
                         ByVal wrddt As Short) As Long
```

[Visual C# .NET]

```
 [DllImport("hd5144.dll")]
public static extern uint hdio5144_outpWriteB(int hDevID, short PortNo, byte byData);
 [DllImport("hd5144.dll")
public static extern uint hdio5144_outpWriteW(int hDevID, short PortNo, short wData);
```

《引数》

hDevID・・・対象デバイスのデバイスハンドル

port　・・・ポート番号指定　0～4 （ポート番号－1）

bytdt ・・・書込む1バイトデータ

wrddt ・・・書込む2バイトデータ

《呼び出し例》

[C言語: Visual C++]

```
DWORD ret;        //関数の戻り値
BYTE     bytdt;       //格納先
ret = hdio5144_wPortB( hDevID, 0x36, bytdt );    //ラッチ信号ステータスポートにバイト書込み
```

[Visual Basic 6.0]

```
Dim ret          As Long
Dim bytdt        As Byte
ret = hdio5144_wPortB( hDevID, &H36, bytdt )       'ラッチ信号ステータスポートにバイト書込み
```

[Visual Basic .NET]

```
Dim ret           As Integer
Dim wrddt        As Short
ret = hdio5144_wPortW( hDevID, &H36, wrddt )       'ラッチ信号ステータスポートにワード書込み
```

[Visual C# .NET]

```
uint        ret;          //関数の戻り値
ushort    wrddt;       //格納先
ret = Hd5144.hdio5144_wPortW( hDevID, &H36, wrddt );         //ラッチ信号ステータスポートにワード書込み
```

8. 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.
サンプルプログラムは次の3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++ 2008・・ C　コーディング　　【 spd51440.exe 】
(2) Visual Basic 6.0　　【 spd51442.exe 】
(3) Visual Basic 2008　　【 spd51443.exe 】
(4) Visual C#　2008　　【 spd51444.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-DIO5144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.

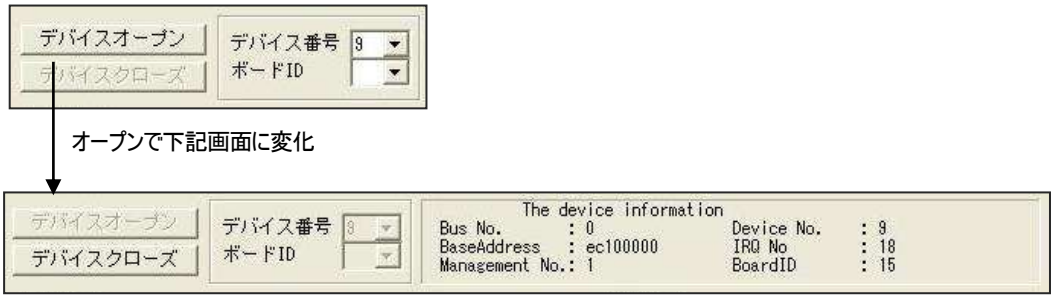
8.1.1　ボード(デバイス)の選択

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）

「デバイス番号」または「ボードID」を選択します。

②デバイスオープン指令

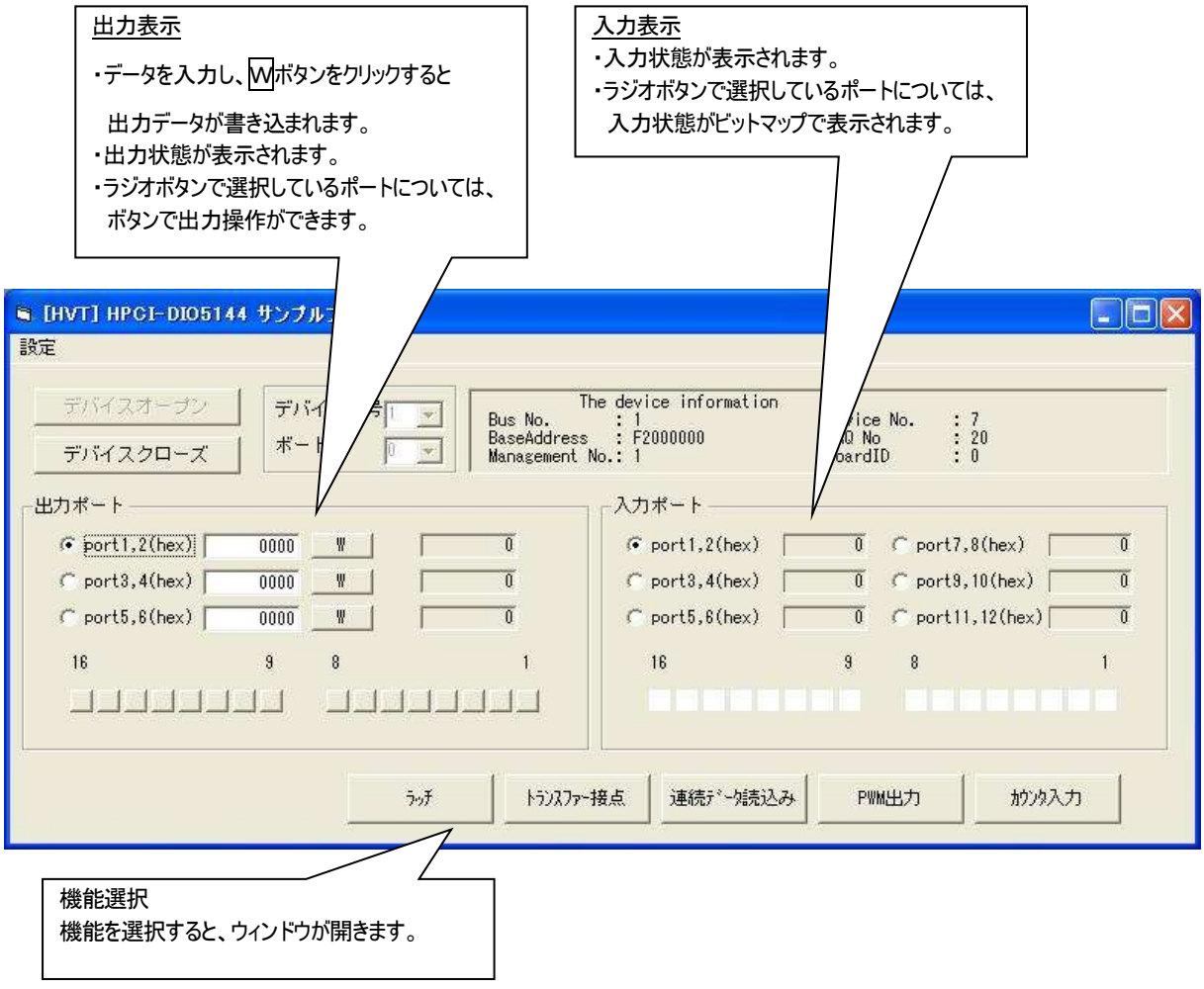


③デバイスクローズ指令

④サンプルプログラムの終了

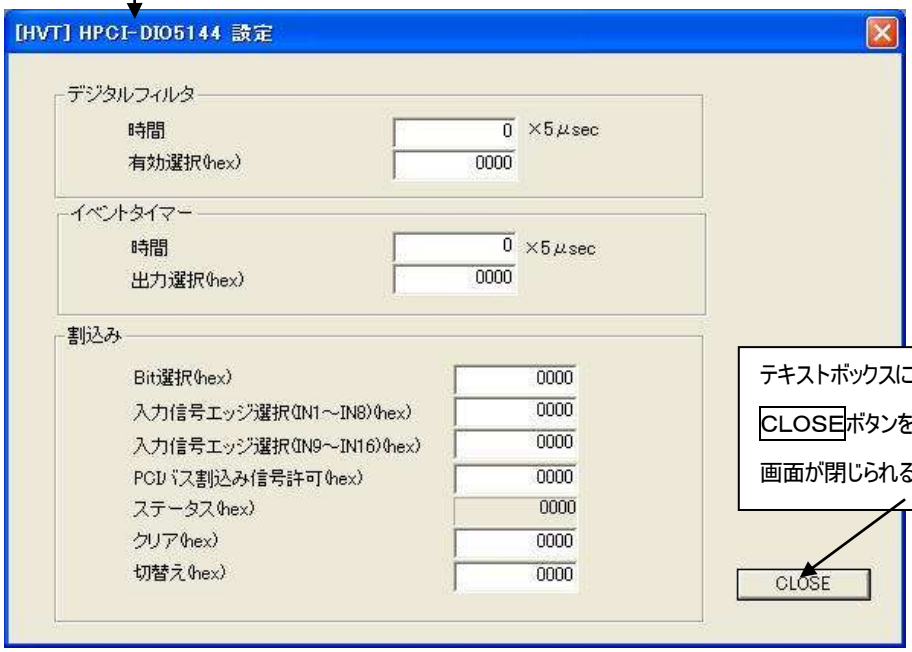
8.1.2　ボード上の操作と表示

デバイスオープンを行いますと次の画面となります．



8.1.3　設定

メニューの「設定」をクリックすると、設定画面が表示されます。



テキストボックスに値を入力して、

CLOSEボタンをクリックして下さい。

画面が閉じられる時に、設定値が書込まれます。

8.1.4 ラッチ機能

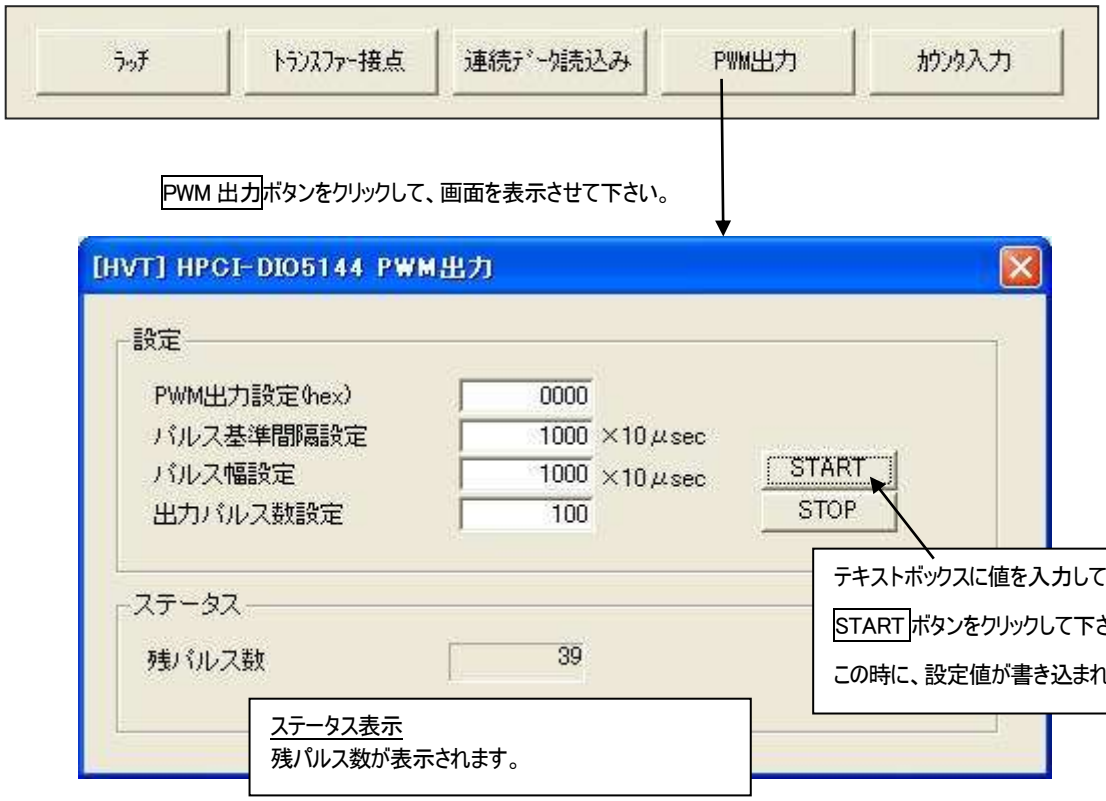
8.1.5 トランスファー接点機能

8.1.6　連続データ読込み機能

■データの読出しについて

・FIFO にデータが入ると、プログラムはデータを読出して、保持しておきます。
・File Write ボタンがクリックされると、保持したデータをテキストファイル「sfifolog.txt」に書込みます。
・FIFO が FULL になった場合と保持データ数が2000を超えた場合には、Strobe 機能を無効にして、
　保持したデータをテキストファイルに「fifolog. txt」書込みます。

8.1.7 PWM 出力機能



8.1.8 カウンタ入力機能

8.2 サンプルプロジェクト

ドライバ関数の使用方法を解説する目的のサンプルプロジェクトを添付しています.
サンプルプログラムを使用する場合は, お客様の開発環境用PCにコピーして使用します.

≪ ご注意 ≫

(1) Visual C++ サンプルプロジェクトより, 実行ファイルを生成するには, 開発ツールとして Visual C++ 2008以降がインストールされている必要があります.
プロジェクトを開発環境より開き, ビルドすることで実行ファイルが生成されます.
実行環境に合わせて, プロジェクトの設定を適宜設定してください.
(Windows SDKバージョン, プラットフォームツールセットなど)

(2) Visual Basicサンプルプロジェクトより, 実行ファイルを生成するには, 開発ツールとしてVisual Basic 2008以降 がインストールされている必要があります.
プロジェクトを開発環境より開き, ビルドすることで実行ファイルが生成されます.
実行環境に合わせて, プロジェクトの設定を適宜設定してください.
(.NET Framework設定など)

(3) Visual C#サンプルプロジェクトより, 実行ファイルを生成するには, 開発ツールとして Visual C# 2008以降 がインストールされている必要があります.
プロジェクトを開発環境より開き, ビルドすることで実行ファイルが生成されます.
実行環境に合わせて, プロジェクトの設定を適宜設定してください.
(.NET Framework設定など)

■ 更 新 履 歴

版	更 新 内 容	備　考
第1版	1.新規作成	
第1.1版	1.Windows 8対応	
第1.2版	1. [1.5適合OS], [1.7サンプルプログラム作成ツール], [1.9ユーザープログラムの作成] :修正	
	2. Windows 10 に関する追記	
	3. [8.2 サンプルプロジェクト] 追記	
	4. 住所：ビル名の変更	