Ｐ－ＰＣＩｅ４－ＬＶ
（PCI　Express 版）
ソフト説明書

株式会社　ファード

２００１．１１／２　初版
２０１１．１１／１１　第二版

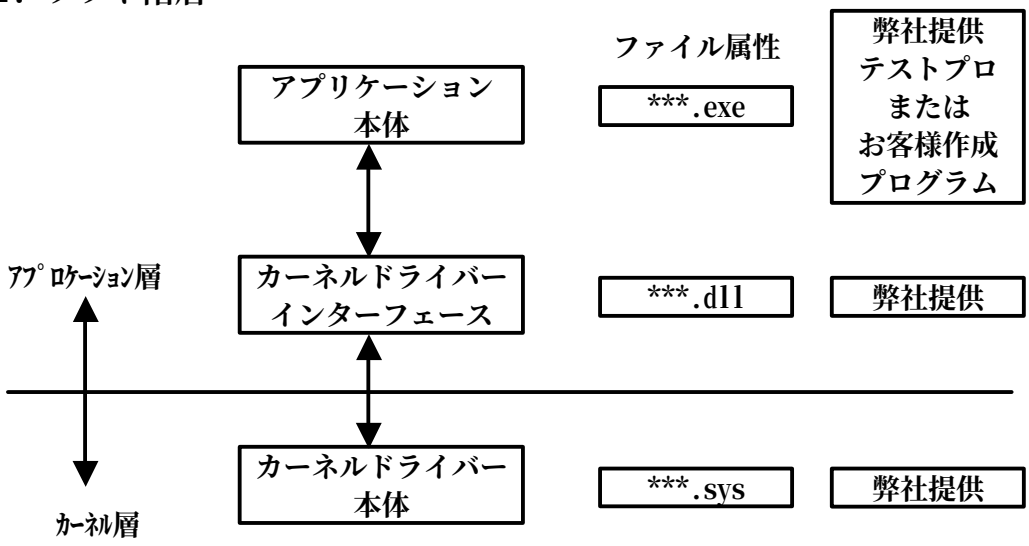
目　　次

１．概要

本説明書は、P－PCIe4－LV ボードを制御するアプリケーションソフトについて説明すろ。

添付ソフトに関しては、“ Windows　インストール方法の説明 ” を参照してください。

２．ソフト階層



(注)　***.sys は、弊社通常 PCI の “ P-PCI-LV ” ボードと同じになっていますが本ボード Express 版とは、内容が違うので、別扱いにしてください。

３．開発について

３－１．開発環境

Windows2000／XP で、Visual　C/C++を使用して
開発できます。

３－２．ライブラリのインストール

ライブラリは、DLL 形式で、添付ソフトの￥exe￥ppcilvdll.dll を
Windows の System ディレクトリィか、実行ファイルのあるディレクトリィに
コピーします。
（テストプログラム実行は、既に実行ファイルと共に同じディレクトリィに
存在していますので、コピーし直す必要は、ありません。）

３－３．ライブラリィのリンク

メニューの［ビルド］［設定］の”リンク”インデックスを選択して、
その中のオブジェクト／ライブラリィ　モジュールに
”ppcilvdll.lib”を入力して設定します。
” ppcilvdll.lib”ファイルは、Visual　C/C++のプロジェクトのある
ディレクトリィにコピーします。
ファイルは、添付の￥ppcilvtest ディレクトリィにあります。
また、定義用ファイルとして”ppcilvapi.h”を作成プログラムに
インクルードしてください。
ファイルは、添付の￥ppcilvtest ディレクトリィにあります。

３－４．ドライバ転送用のメモリ

マスタモード時、転送データ容量分のパソコンの物理メモリ（連続したエリア）を
確保しますので、大容量のデータを転送しようとすると、他のアプリケーションの
動作が遅くなったり、また確保、出来なかったりします。
大容量のデータを扱う場合、それなりのメモリを増設してください。

４．プログラム手順

実際に関数を使用してアプリケーションを作成する手順を説明します。
関数の説明も参照しながら読んでください。
一部変数の定義は、省略します。
また、複数枚使用時は、ハンドルを枚数分確保し、それぞれそのハンドルで
実行してください。
（注）当社“Ｐ－ＰＣＩ”ボードと関数は、同じ名前を使用しています。

４－１．ボードのオープン

（１）個別関数を使用する時

```
HANDLE  hVxD;          //ハンドル定義（複数枚使用時は、枚数分定義）
hVxd=PpciOpen(1);      // 引数(1)は、基板 ID でディップスイッチ下位４
                       //  ビットに対応するが、１枚搭載の場合
                       //  無視する。
```

（注）ボードのオープンは、開始時に一度行えばよい。

（２）マクロ関数を使用する時

```
HANDLE  hVxD;              //ハンドル定義
hVxD=PpciBlockInit(1);     //引数、戻り値は、PpciOpen と同じ。
```

以下の関数についてのハンドルは、この"hVxD"を使用する。

４－２．ボードのクローズ

```
int  ret;
ret=PpciClose(hVxD);
```

（注）アプリケーション終了時には、必ず実行する事。

４－３．ターゲットモード

（共通）ターゲットモード設定（電源立ち上げ時は、このモードになっている）

```
PpciRunMode (hVxD,QTARGET);
```

（１）ＬＥＤの点灯／消灯を行う

```
PpciLEDOut(hVxD,LED_data)   //LED_data に LED 点灯/消灯データを入力
                            //  "1":点灯、"0"で消灯
```

（２）ディップスイッチをリードする。

```
BYTE  sw;
sw=PpciDipSwitchIn(hVxD);     //sw にスイッチ情報
                              //"1":ＯＮ、"0"：ＯＦＦ
```

（３）外部コネクタと入出力する。

①データの入力／出力の設定（バイト単位）

```
PpciTargetDIR(hVxD,dir1,dir2,dir3,dir4);
    //dir1～4 に入出力方向を入力
    //  入力：QBYTEIN
    //  出力：QBYTEOUT
```

②アクセス

・入力

```
DWORD  data;
data=PpciIOInput(hVxD);  //  出力モードになっているデータは、
                         //  ソフトでマスクして使用する。
```

・出力

```
PpciIOOutput(hVxd,out_data);  //  out_data に出力データを入力
```

４－４．マスタモード（マクロ関数を使用時）

（１）外部からの入力

①アプリケーションデータバッファ確保

```
char  *buf;
buf=(char*)malloc(data_size);  //ｃの関数
```

②実行

```
int  ret;
ret=PpciBlockRead(hVxD,buf,data_size,Timeout);
        //Timeout にタイマ監視（1mS 単位）を入力
```

アプリケーション開始時は、ボードをオープン（４－１参照）し、
アプリケーション終了時は、ボードをクローズ（４－２参照）する。

（２）外部へ出力

①アプリケーションデータエリア確保（データ作成）

```
char  *buf;
buf=(char*)malloc(data_size);  //ｃの関数
```

このエリアにデータを作成する。

②実行

```
int  ret;
ret=PpciBlockWrite(hVxD,buf,data_size,Timeout);
       //Timeout にタイマ監視（1mS 単位）を入力
```

アプリケーション開始時は、ボードをオープン（４－１参照）し、
アプリケーション終了時は、ボードをクローズ（４－２参照）する。

４－５．マスタモード（個別関数を使用する場合）

（共通）マスタモード設定

```
PpciRunMode（hVxD,QBUSMASTER);
```

（１）外部から入力

① 所有権の有無

・制御権をとる場合

```
int  ret;
ret=PpciSetPRV(hVxD,QPRVON);
//もし ret が QPRVOFF の場合制御権は、相手機器にあります。
```

・制御権をとらない場合（制御権を放棄する）

```
int  ret;
ret=PpciSetPRV(hVxD,QPRVOFF);
//ret は無条件に QPRVOFF になっています。
```

② 転送方向を設定

・制御権がある場合

```
PpciSetDIR(hVxd,QMSTIN);
```

・制御権がない場合

```
DWORD  status;
status=PpciStatusRead(hVxD);
      //status のビット 6 が"1"の場合入力が可能です。
```

③ドライバのバッファを確保

```
int  ret;
ret=PpciMemAlloc(hVxd, data_size) ;
      //data_size にバイト数を入力
      //ret が NULL の時、エラーで確保できません。
```

④アプリケーションバッファを確保

```
char  *buf;
buf=(char*)malloc(data_size);
```

⑤ 転送長を設定

```
PpciSetDataLENG(hVxD,data_size-1);
      //転送長は、data_size-1 を設定
```

⑥スタート

```
int  ret;
ret=PpciStart(hVxD);  //  ret=0 の時正常
                      //  エラーは、ライブラリィ説明を参照。
```

⑦ステータスリードし終了をチェックする。

```
int  status;
status=PpciStatusRead(hVxD);
//  status のビット 31="0"の時終了。
```

⑧ドライバのバッファをリードする。

```
PpciReadBuf(hVxD,buf,data_size);
```

以上で終わりですが、繰り返し同じ容量でしたら⑥より行い
容量が変わるときは、
ドライバ、アプリケーションバッファを解放（⑨）して③よりおこなう。

⑨バッファの解放（アプリケーション終了）

```
free(buf);              //アプリケーションバッファ解放
PpciMemFree(hVxD)；     //ドライババッファ解放
```

（２）外部へ出力

① 所有権の有無

・制御権をとる場合

```
int  ret;
ret=PpciSetPRV(hVxD,QPRVON);
//もし ret が QPRVOFF の場合制御権は、相手機器にあります。
```

・制御権をとらない場合（制御権を放棄する）

```
int  ret;
ret=PpciSetPRV(hVxD,QPRVOFF);
//ret は無条件に QPRVOFF になっています。
```

② 転送方向を設定

・制御権がある場合

```
PpciSetDIR(hVxd,QMSTOUT);
```

・制御権がない場合

```
DWORD  status;
status=PpciStatusRead(hVxD);
      //status のビット 6 が"0"の場合入力が可能です。
```

③ドライバのバッファを確保

```
int  ret;
ret=PpciMemAlloc(hVxd, data_size) ;
      //data_size にバイト数を入力
      //ret が NULL の時、エラーで確保できません。
```

④アプリケーションバッファを確保

```
char  *buf;
buf=(char*)malloc(data_size);
```

⑤ 転送長を設定

```
PpciSetDataLENG(hVxD,data_size-1);
    //転送長は、data_size-1 を設定
```

⑥データ作成

アプリケーションバッファ（buf)にデータを作成する。

⑦ドライバのバッファにライトする。

```
PpciWriteBuf(hVxD,buf,data_size);
```

⑧スタート

```
int  ret;
ret=PpciStart(hVxD);  //  ret=0 の時正常
                      //  エラーは、ライブラリィ説明を参照。
```

⑨ステータスリードし終了をチェックする。

```
int  status;
status=PpciStatusRead(hVxD);
//  status のビット 31="0"の時終了。
```

以上で終わりですが、繰り返し同じ容量でしたら⑥より行い
容量が変わるときは、ドライバ、アプリケーションバッファを
解放（⑩）して③よりおこなう。

⑩バッファの解放（アプリケーション終了）

```
free(buf);              //アプリケーションバッファ解放
PpciMemFree(hVxD)；     //ドライババッファ解放
```

４－６．割り込みについて

割り込みは、直接割り込みが入るのではなく、割り込みが入ると
ユーザーアプリケーションに対しメッセージを送ります。
ユーザーは、そのメッセージを処理します。

①割り込みの登録

```
PpciCallbackWND(hVxD,(DWORD)this->m_hWnd);
        //  this->m_hWnd  でそのクラスのウインドハンドルが得られる。
```

②メッセージ処理

```
LRESULT CPpcisampleView::WindowProc
        (UINT message, WPARAM wParam, LPARAM lParam)
{
    if(message==WM_USER_INT){     //  割り込みメッセージ番号
      if((DWORD)lParam==(DWORD)hVxD){   //ボード複数枚の識別で
                                        //ｵｰﾌﾟﾝ時のハンドルを使用する。
        CString     cs=" ";
        if((wParam&0x1)!=0){//正常終了
            cs+="正常終了、";
        }
        if((wParam&0x2)!=0){//強制終了
            cs+="強制終了、";
         }
        if((wParam&0x4)!=0){//開始要求
            cs+="開始要求";
         }
            cs+="の割り込みが入りました";
        MessageBox(cs);
      }
       if(lParam==(DWORD)hVxD1){  //次のボードをチェックする。
              上記同様ステータスチェックをする。
        }
        return CFormView::WindowProc(message, wParam, lParam);
}
```

上記は、添付テストプログラムより引用しました。
実際には、classWizard で作成するクラス、オブジェクト ID を選択し
（この場合 CPcisampleView)、メッセージを WindowProc を選択し、ダブルクリックに
より、上記下線の関数が生成されます。[コード編集]でその関数位置にジャンプし、
それ以外のコードを作成します。

message をチェックし、WM_USER_INT でしたら、割り込みが入っていて、wParam に割り込みステータスが、格納されています。
また、複数枚対応として、lParam にボードオープン時に獲得したハンドルが格納されていますので、それをチェックしどのボードから割り込みが入ったかを判断します。

割り込みステータスは、マスクが解除されているもののみ入っていて、マスクがされているビットについては、関知しません。
しかし、割り込み処理（ドライバ）で、割り込みステータスは、
クリアされますので、
マスクされている割り込みステータスは、破棄されます。

割り込みメッセージ番号は、定義（WM_USER_INT)していますが、
別定義にする場合の値は、”WM_USER+1”にしてください。

５．関数一覧

５－１．個別制御関数

No.	モード	関数名	概略内容
1	共通	PpciOpen	ﾄﾞﾗｲﾊﾞのｵｰﾌﾟﾝ
		PpciClose	〃　ｸﾛｰｽﾞ
		PpciSetRunMode	転送方式
		PpciStatusRead	ｽﾃｰﾀｽﾘｰﾄﾞ
2	ﾎﾟｰﾄ入出力	PpciTargetDIR	入出力方向
		PpciIOInput	データ入力
		PpciIOOutput	データ出力
		PpciLEDOut	ＬＥＤ出力
		PpciDipSwitchIn	ディップスイッチ・ﾘｰﾄﾞ
3	I/O 入出力	PpciIORead	Ｉ／Ｏリード
		PpciIOWrite	Ｉ／Ｏライト
4	ﾏｽﾀ転送	PpciMemAlloc	ﾄﾞﾗｲﾊﾞﾊﾞｯﾌｧの取得
		PpciMemFree	〃　　解放
		PpciSetPRV	制御権の設定
		PpciMasterDIR	転送方向
		PpciSetDataLENG	転送ﾊﾞｲﾄ数
		PpciStart	転送ｽﾀｰﾄ
		PpciStop	転送強制終了
		PpciINTMask	割り込みマスク
		PpciINTClear	割り込みクリア
		PpciCallbackWND	割り込み登録
		PpciReadBuf	ﾄﾞﾗｲﾊﾞﾊﾞｯﾌｧのﾘｰﾄﾞ
		PpciWriteBuf	〃　　ﾗｲﾄ
5	PCI ｺﾝﾌｨｸﾞ ﾚｼﾞｽﾀ	PpciConfigRead	全てのｺﾝﾌｨｸﾞﾚｼﾞ・ﾘｰﾄﾞ
		PpciGetIOAddress	I/O ﾍﾞｰｽｱﾄﾞﾚｽ・ﾘｰﾄﾞ
		PpciGetMEMAddress	ﾒﾓﾘ　　〃
		PpciGetRevID	ﾚﾋﾞｼﾞｮﾝ ID ﾘｰﾄﾞ

５－２．マクロ関数（マスタモードのみ）

No.	モード	関数名	概略内容
1	ﾏｽﾀ転送	PpciBlockInit	ﾌﾞﾛｯｸﾓｰﾄﾞ初期化
		PpciBlockRead	ﾌﾞﾛｯｸﾘｰﾄﾞ
		PpciBlockWrite	ﾌﾞﾛｯｸﾗｲﾄ

（注）アプリケーション終了時は、個別関数の PpciClose を使用してください。

６．構造体（”ppcilvapi.h”）

・ＰＣＩコンフィグレジスタ情報用構造体

```
struct    PPCI_CONFIG_DATA{
              WORD    DeviceID;
              WORD    BenderID;
              WORD    Status;
              WORD    Command;
              DWORD   ClassCode;
              BYTE    RevisionID;
              BYTE    Bist;
              BYTE    HeaderType;
              BYTE    LatencyTimer;
              BYTE    CacheLineSize;
              DWORD   BaseAddress1;          //  I/O
              DWORD   BaseAddress2;          //  メモリ
              BYTE    MaxLatency;
              BYTE    MinGrant;
              BYTE    IntPin;
              BYTE    IntLine;
};
```

PpciConfigRead 関数で使用する。

７．エラーコード（”ppcilvappi.h”）

```
#define    QERROR_END      0xffffffff
#define    QNORMAL_END     0x0
#define    QSTOP_END           0x1
#define    QERROR_CNN      0x2
#define    QERROR_BUF      0x4
#define    QTIME_OVER      0x8
```

８．定数

```
#define WM_USER_INT     WM_USER+1    //割り込み時、アプリケーションに渡す
                                     //メッセージ
#define QTARGET         0            //  ターゲットモード
#define QBUSMASTER      1            //  マスターモード
#define QBYTEIN         0            //  ターゲット時のデータ入力方向
#define QBYTEOUT        1            //        〃          出力方向
#define QMSTOUT         1            //  マスタ時のデータ出力方向
#define QMSTIN          0            //        〃        入力方向

#define QPRVON          0            //  制御権獲得
#define QPRVOFF         1            //  制御権解放

#define QSTARTMASK      0x0004       //  開始割り込みマスク
#define QSTOPMASK       0x0002       //  強制終了　〃
#define QENDMASK        0x0001       //  正常終了　〃
```

９．ステータス

PpciStatusRead 関数で得られるデータは、Ｈ／Ｗステータスです。

ビット

３１　マスタビジー　　"0"：停止（終了）、"1"：実行中
　　PpciStart 関数でスタート後、"0"になると
　　転送終了。

３０
～１９　空き

１８　開始要求割り込みステータス　"1"：開始要求割り込みあり
１７　強制終了　　〃　　"1"：強制終了　　〃
１６　正常終了　　〃　　"1"：正常終了　　〃

（注）割り込みステータスは、割り込みマスクをしていてもセットされる。

１５
～９　空き
８　動作モード　"0"：ターゲットモード、"1"：マスタモード
７　制御権　　"0"：制御権あり、"1"：制御権なし
６　マスタモード時の転送方向
　　・制御権ありの時　"0"：入力、"1"：出力
　　・制御権なしの時　"0"：出力、"1"：入力
５
～４　空き
３　ＲＥＱＵＥＳＴ状態　"1"：アクティブ
２　ＲＥＡＤＹ状態　　〃
１　ＥＮＡＢＬＥ状態　　〃
０　ＶＡＬＩＤ状態　　〃

ビット０～３は、マスタモード時ブロック転送コントロール信号である。

１０．ライブラリィ説明（ＤＬＬ形式で提供）

関数1.1	ボードのオープン	有効ﾓｰﾄﾞ	共通
ﾌﾟﾛﾄﾀｲﾌﾟ	HANDLE　Ppci0pen（BYTE　p1)		
引数	BYTE　p1:ディップスイッチ下位４ビット（Sw-No４～１） ボードが複数存在するときに識別用として、ディップスイッチを使用する。 １枚のみ使用時は、無条件に選択する。 DIPSW　"1"：ＯＮ "0"：ＯＦＦ		
戻り値	HANDLE　：オープンしたボードのハンドル番号 この HANDLE で、以後の関数で使用する。		
内容	ドライバをオープンし、使用できるようにする。		
備考			

関数 1.2	ドライバの終了	有効ﾓｰﾄﾞ	共通
ﾌﾟﾛﾄﾀｲﾌﾟ	int PpciClose(HANDLE　ph)		
引数	HANDLE　ph：ボードオープン時の戻り値		
戻り値	int　：エラーステータス ０　：正常終了 －１：オープンされていないのにクローズをした。		
内容	現在使用しているボードのハンドラをクローズする。（クローズ）		
備考			

関数 1.3	ボード動作モード	有効ﾓｰﾄﾞ	共通
ﾌﾟﾛﾄﾀｲﾌﾟ	void PpciSetRunMode（HANDLE ph, int pl）		
引数	HANDLE ph：ボードオープン時の戻り値 int pl :基板動作モード （基板動作モード） QTARGET （ターゲットモード：パラレルＩ／Ｏ） QBUSMASTER （マスタモード：ブロック転送）		
戻り値	なし		
内容	ボードの動作モードを設定する。 ターゲットモードは、単発アクセスで３２ビットパラレルアクセスを行うモードです。 バスマスタモードは、コントロール線を使用して、３２ビットバーストアクセスを行う。 その時、ボードはパソコンのメモリに対しバスマスタアクセスを行う。		
備考	転送方式に関しては、Ｈ／Ｗ説明書を参照してください。		

関数 1.4	ステータスリード	有効ﾓｰﾄﾞ	共通
ﾌﾟﾛﾄﾀｲﾌﾟ	DWORD PpciStatusRead(HANDLE　ph)		
引数	HANDLE　ph：ボードオープン時の戻り値		
戻り値	DWORD　：ステータス３２ビット ステータス内容は、”９．ステータス”を参照してください。		
内容	ボードの実行状態、設定状態をリードする。		
備考	ビットの内容に対しては、ステータス説明を参照してください。		

関数 2.1	データ方向	有効ﾓｰﾄﾞ	ﾀｰｹﾞｯﾄ
ﾌﾟﾛﾄﾀｲﾌﾟ	void PpciTargetDIR (HANDLE ph,int p1,int p2,int p3,int p4)		
引数	HANDLE ph：ボードオープン時の戻り値 int p1 ：第１バイト目の方向（最下位バイト） int p2 ：第２バイト目の方向 int p3 ：第３バイト目の方向 int p4 ：第４バイト目の方向 （最上位バイト） （設定値） QBYTEIN ：入力モード QBYTEOUT ：出力モード		
戻り値	なし		
内容	ターゲット（パラレルＩ／Ｏ）モード時の各データバイトの方向を設定する。		
備考			

関数 2.2	データ入力	有効ﾓｰﾄﾞ	ﾀｰｹﾞｯﾄ
ﾌﾟﾛﾄﾀｲﾌﾟ	DWORD PpciIOInput (HANDLE ph)		
引数	HANDLE ph：ボードオープン時の戻り値		
戻り値	DWORD ： ３２ビットデータ入力。		
内容	ターゲットモード時のデータ入力。 （注）データを入力にしているバイトのみ有効。 出力にしているバイトに対しては、出力データが入力される。		
備考			

関数 2.3	データ出力	有効ﾓｰﾄﾞ	ﾀｰｹﾞｯﾄ
ﾌﾟﾛﾄﾀｲﾌﾟ	void PpciIOOutput (HANDLE ph,DWORD p1)		
引数	HANDLE ph：ボードオープン時の戻り値 DWORD p1 ：出力データ３２ビット。		
戻り値	なし		
内容	ターゲット（ＰＩ／Ｏ）モード時のデータ出力をする。 入力モードに設定されているバイトのデータは、任意である。		
備考			

関数2.4	ＬＥＤデータ出力	有効ﾓｰﾄﾞ	ﾀｰｹﾞｯﾄ
ﾌﾟﾛﾄﾀｲﾌﾟ	void PpciLED0ut(HANDLE　ph,BYTE　p1)		
引数	HANDLE　ph：ボードオープン時の戻り値 BYTE　p1　：ＬＥＤデータ８ビット		
戻り値	なし		
内容	ＬＥＤを点灯／消灯する。 "1"で点灯。 "0"で消灯。		
備考	ターゲット（ＰＩ／Ｏ）モード時のみ有効で、バスマスタ（ブロック転送）時には、無効である。		

関数2.5	ディップスイッチ入力	有効ﾓｰﾄﾞ	ﾀｰｹﾞｯﾄ
ﾌﾟﾛﾄﾀｲﾌﾟ	BYTE PpciDipSwitchIn(HANDLE　ph)		
引数	HANDLE　ph：ボードオープン時の戻り値		
戻り値	BYTE　：　ディップスイッチデータ８ビット		
内容	ディップスイッチをリードする。 スイッチＯＮで"1"、ＯＦＦで"0"。		
備考	ターゲット（ＰＩ／Ｏ）モード時のみ有効で、バスマスタ（ブロック転送）時には、無効である。 （注）電源立ち上げ時、下位４ビットをボード識別番号として使用する。 それ以降では、使用は任意である。		

関数 3.1	ボードのＩ／Ｏレジスタのリード	有効ﾓｰﾄﾞ	共通
ﾌﾟﾛﾄﾀｲﾌﾟ	DWORD　PpciIORead(HANDLE　ph,DWORD p1　)		
引数	HANDLE　ph：ボードオープン時の戻り値 DWORD　p1：I/O ｱﾄﾞﾚｽのｵﾌｾｯﾄ（+0～）		
戻り値	DWORD　該当ｵﾌｾｯﾄのレジスタデータ３２ビット		
内容	ボードのレジスタをリードする。 ないレジスタをリードしたとき、データは保証されない。		
備考	レジスタのオフセットについては、ハード仕様書を参照してください。		

関数 3.2	ボードのＩ／Ｏレジスタのリード	有効ﾓｰﾄﾞ	共通
ﾌﾟﾛﾄﾀｲﾌﾟ	void PpciIOWrite（ HANDLE ph,DOWRD p1,DWORD p2 ）		
引数	HANDLE ph：ボードオープン時の戻り値 DWORD p1：I/O ｱﾄﾞﾚｽのｵﾌｾｯﾄ（+0～） DWORD p2:ﾗｲﾄﾃﾞｰﾀ		
戻り値	なし		
内容	データをレジスタにライトする。 ないレジスタのオフセットにライトしないでください、予期しない 動作が、おきることがあります。		
備考	レジスタのオフセットについては、ハード仕様書を参照してください。		

関数 4.1	ドライバの転送バッファ取得	有効ﾓｰﾄﾞ	ﾏｽﾀ
ﾌﾟﾛﾄﾀｲﾌﾟ	int PpciMemAlloc(HANDLE ph,DWORD p1)		
引数	HANDLE ph：ボードオープン時の戻り値 DWORD p1 ：取得するバイト数 バイト数は、４の倍数で最大２５６Ｍバイト設定できるが、 パソコンのメモリ容量により、取得出来ないことがある。 （バイト数が４の倍数でないとき、端数は切り捨てられる。）		
戻り値	int ：物理アドレス NULL の時取得失敗。		
内容	ドライバが、バスマスタ転送用に使用する物理メモリを取得する。		
備考			

関数 4.2	ドライバの転送バッファ解放	有効ﾓｰﾄﾞ	ﾏｽﾀ
ﾌﾟﾛﾄﾀｲﾌﾟ	void PpciMemFree(HANDLE　p1)		
引数	HANDLE　p1：初期化時の戻り値		
戻り値	なし		
内容	関数 PpciMemAlloc で取得したドライバのバッファを解放する。		
備考			

関数 4.3	制御権の設定	有効ﾓｰﾄﾞ	ﾏｽﾀ
ﾌﾟﾛﾄﾀｲﾌﾟ	int PpciSetPRV(HANDLE ph,int p1,int p2)		
引数	HANDLE ph：ボードオープン時の戻り値 int p1 ：制御権値 int p2 ：リトライ回数 （制御権値） QPRVON ：制御権あり QPRVOFF ：制御権なし （リトライ回数） QMASTER で制御権をとれなかった場合のリトライ回数。		
戻り値	int ：QPRVON/QPRVOFF QPRVON で制御権をとれなかった場合、QPRVOFF を戻す。		
内容	制御権を取得/解放する。 制御権とは、転送方向を決めることができる権利である。		
備考			

関数 4.4	データの転送方向	有効ﾓｰﾄﾞ	ﾏｽﾀ
ﾌﾟﾛﾄﾀｲﾌﾟ	void PpciMasterDIR(HANDLE ph,int p1)		
引数	HANDLE ph：ボードオープン時の戻り値 int p1 ：転送方向 （転送方向） QMSTIN ：入力 QMSTOUT ：出力		
戻り値	なし		
内容	ブロック転送の方向を設定する。 ただし、制御権を取得（QPRVON）していなければならない。 （QPRVOFF の場合、この関数は無効である。）		
備考			

関数 4.5	転送データバイト数	有効ﾓｰﾄﾞ	ﾏｽﾀ
ﾌﾟﾛﾄﾀｲﾌﾟ	void PpciSetDataLENG(HANDLE　ph,DWORD p1)		
引数	HANDLE　ph：ボードオープン時の戻り値 DWORD　p1：転送データ数（バイト数）の設定。 （転送データ数） 設定範囲は 4～256M バイト。 設定は４の倍数で、端数は切り捨てられる		
戻り値	なし		
内容	転送データバイト数で、PpciMemAlloc で取得したバイト数以下にする。		
備考			

関数 4.6	転送開始	有効ﾓｰﾄﾞ	ﾏｽﾀ
ﾌﾟﾛﾄﾀｲﾌﾟ	int PpciStart(HANDLE　ph)		
引数	HANDLE　ph：ボードオープン時の戻り値		
戻り値	int　：　-1=転送長をセットしていない。 -2=転送バッファを確保していない。 -3=転送バッファより転送長の方が長い。 0=正常終了		
内容	転送を開始する。 しかし、対抗機器が、スタートしていない時は、開始待ち状態になる。		
備考			

関数 4.7	転送強制終了	有効ﾓｰﾄﾞ	ﾏｽﾀ
ﾌﾟﾛﾄﾀｲﾌﾟ	void PpciStop(HANDLE　ph)		
引数	HANDLE　ph：ボードオープン時の戻り値		
戻り値	なし		
内容	PpciStart でスタート後、この関数を実行すると終了する。 割り込み（強制終了、正常終了）は、起こらない。		
備考			

関数 4.8	割り込みマスクの設定	有効ﾓｰﾄﾞ	ﾏｽﾀ
ﾌﾟﾛﾄﾀｲﾌﾟ	void PpciINTMask(HANDLE　ph,BYTE p1)		
引数	HANDLE　ph：ボードオープン時の戻り値 BYTE　p1：割り込みマスクデータ （マスクデータ） QENDMASK　：正常終了マスク QSTOPMASK　：強制終了マスク QSTARTMASK　：開始マスク マスクをセット（割り込み禁止）するときは、上記値を またクリア（割り込み許可）する時は、上記値を入れない。		
戻り値	なし		
内容	割り込みのマスク（禁止）する。値がないときは、マスクを クリア（許可）する。 割り込みのマスクをしても、ステータスには、影響を与えない。		
備考			

関数 4.9	割り込みクリア	有効ﾓｰﾄﾞ	ﾏｽﾀ
ﾌﾟﾛﾄﾀｲﾌﾟ	void PpciINTClear(HANDLE　ph)		
引数	HANDLE　ph：ボードオープン時の戻り値		
戻り値	なし		
内容	割り込みステータスをクリアする。 割り込みを使用している場合、割り込みが発生した時点でドライバにより　自動的にクリアされる。 割り込み要因は、メッセージにより渡される。		
備考			

関数 4.10　割り込み登録	有効ﾓｰﾄﾞ	ﾏｽﾀ
ﾌﾟﾛﾄﾀｲﾌﾟ	void　PpciCallbackWND(HANDLE　ph,DWORD　p1)	
引数	HANDLE　ph：ボードオープン時の戻り値 DWORD　p1：アプリケーションウインドハンドル ウインドハンドルは、（DWORD）this->m_hWnd で得られる。	
戻り値	なし	
内容	割り込み時、メッセージを送るウインドを登録する。	
備考		

関数4.11	ドライバのバッファをリード	有効ﾓｰﾄﾞ	ﾏｽﾀ
ﾌﾟﾛﾄﾀｲﾌﾟ	void PpciReadBuf(HANDLE ph,void *p1,DWORD p2)		
引数	HANDLE ph：ボードオープン時の戻り値 void *p1：アプリケーションバッファポインター DWORD p2：リードバイト数（４の倍数）		
戻り値	なし		
内容	ドライバのバッファからアプリケーションバッファにデータを リードする。		
備考			

関数 4.12	ドライバのバッファライト	有効ﾓｰﾄﾞ	ﾏｽﾀ
ﾌﾟﾛﾄﾀｲﾌﾟ	void PpciWriteBuf(HANDLE ph,void *p1,DWORD p2)		
引数	HANDLE ph：ボードオープン時の戻り値 void *p1：アプリケーションバッファポインター DWORD p2：ライトバイト数（４の倍数）		
戻り値	なし		
内容	アプリケーションバッファからドライバのバッファにライトする。		
備考			

関数5.1	ＰＣＩコンフィグレジスタリード	有効ﾓｰﾄﾞ	共通
ﾌﾟﾛﾄﾀｲﾌﾟ	void PpciConfigRead(HANDLE　ph,struct PPCI_CONFIG_DATA *p1)		
引数	HANDLE　ph：ボードオープン時の戻り値 PPCI_CONFIG_DATA *p1 ：ＰＣＩコンフィグレジスタ構造体のポインター 構造体の内容は、構造体の項を参照。		
戻り値	なし		
内容	現在オープンされているボードのＰＣＩコンフィグレジスタを全てリードし、PPCI_CONFIG_DATA 構造体内に格納する。		
備考			

関数 5.2	Ｉ／Ｏベースアドレスリード	有効ﾓｰﾄﾞ	共通
ﾌﾟﾛﾄﾀｲﾌﾟ	DWORD PpciGetIOAddress(HANDLE　ph)		
引数	HANDLE　ph：ボードオープン時の戻り値		
戻り値	DWORD　：Ｉ／Ｏアドレス値３２ビット 有効アドレスは、下位１６ビットである。 （ＩＮＴＥＬ系の３２ビットＣＰＵのＩ／Ｏアドレスが、 １６ビットの為）		
内容	ボードのＩ／Ｏを直接アクセスするためのアドレスで、このアドレスにオフセットを加算し、Ｃ言語または、アセンブラ言語のＩ／Ｏ命令で実行する。 (注)直接 I/O 命令でアクセスできるのは、Windows95/98 で WindowsNT/2000 の場合は、できません。 I/O のアクセスは、できるなら IopciIORead、IopciIOWrite の関数を使用してください。		
備考	このベースアドレスは、PpciConfigRead を実行し、PPCI_CONFIG_DATA 構造体からも取得できる。		

関数 5.3	メモリベースアドレスリード	有効ﾓｰﾄﾞ	共通
ﾌﾟﾛﾄﾀｲﾌﾟ	DWORD PpciGetMEMAddress(HANDLE　ph)		
引数	HANDLE　ph：ボードオープン時の戻り値		
戻り値	DWORD　：メモリベースアドレス値３２ビット		
内容	メモリベース物理アドレスで、参照用です。 使用できません。		
備考			

関数 5.4	ボードレビジョンリード	有効ﾓｰﾄﾞ	共通
ﾌﾟﾛﾄﾀｲﾌﾟ	BYTE PpciGetRevID(HANDLE　ph)		
引数	HANDLE　ph：ボードオープン時の戻り値		
戻り値	BYTE　：オープンされているボードのレビジョン番号		
内容	ボードのレビジョン番号リードで、管理用である。		
備考	PpciConfigRead でも得られる。		

関数 6.1	ブロック転送初期化（マクロ関数）	有効ﾓｰﾄﾞ	ﾏｽﾀ
ﾌﾟﾛﾄﾀｲﾌﾟ	HANDLE PpciBlockInit(BYTE　p1)		
引数	BYTE　p1:ディップスイッチ下位４ビット（Sw-No４～１） ボードが複数存在するときに識別用として、ディップスイッチを使用する。 １枚のみ使用時は、無条件に選択する。 DIPSW　"1"：ＯＮ "0"：ＯＦＦ		
戻り値	HANDLE　：オープンしたボードのハンドル番号		
内容	PpciOpen と PpciRUNMode、PpciSetPRV 関数で構成され、 マクロ関数 PpciBlockRead、PpciBlockWrite のみ使用時は、 初期化として実行する。		
備考	アプリケーション開始時に、初期化として実行する。		

関数 6.2	ブロック転送入力 （マクロ）	有効ﾓｰﾄﾞ	ﾏｽﾀ
ﾌﾟﾛﾄﾀｲﾌﾟ	int PpciBlockRead(HANDLE ph,void *p1,DWORD p2,DWORD p3)		
引数	HANDLE ph：ボードオープン時の戻り値 void *p1 ：データ入力バッファポインター DWORD p2 ：入力データバイト数 DWORD p3 ：タイムオーバー 入力バッファポインター：アプリ側の入力格納バッファのポインター 入力データバイト数：入力データバイト数で、４の倍数設定。 ４～２５６Mバイト設定出来る。 タイムオーバー：１ｍＳ単位で、スタート（待ちも含む）からの 時間管理タイマ。 "0"設定で、時間管理を行わない。		
戻り値	int ：エラー情報 QERROR_END：制御権がなくて、データ転送方向が逆である。 QERROR_BUF：ドライバのバッファが確保出来ない。 QTIME_OVER：タイムオーバー		
内容	ブロック転送入力で、マクロ関数である。 内部で実行している関数は、 ① 転送方向を設定（制御権が取得している時）。PpciMasterDIR ② ドライバのバッファを獲得。PpciMemAlloc ③ 転送長セット。PpciSetDataLENG ④ スタート。PpciStart ⑤ タイムアウトチェック及び ステータスで転送終了待ち ⑥ ドライバのバッファ解放。PpciMemFree		
備考	制御権が、取得出来なかった場合（PRVOFF）で、転送方向が、出力になっていた場合、エラーになり実行されない。		

関数 6.3	ブロック転送出力	有効ﾓｰﾄﾞ	ﾏｽﾀ
ﾌﾟﾛﾄﾀｲﾌﾟ	int PpciBlockWrite(HANDLE ph,void *p1,DWORD p2,int p3)		
引数	HANDLE ph：ボードオープン時の戻り値 DWORD *p1 ：データ出力バッファポインター DWORD p2 ：出力データバイト数 DWORD p3 ：タイムオーバー 出力バッファポインター：アプリ側の出力格納バッファのポインター 出力データバイト数：入力データバイト数で、４の倍数設定。 ４～２５６Mバイト設定出来る。 タイムオーバー：１ｍＳ単位で、スタート（待ちも含む）からの 時間管理タイマ。 "0"設定で、時間管理を行わない。		
戻り値	int ：エラー情報 QERROR_END：制御権がなくて、データ転送方向が逆である。 QERROR_BUF：ドライバのバッファが確保出来ない。 QTIME_OVER：タイムオーバー		
内容	ブロック転送出力で、マクロ関数である。 内部で実行している関数は、 ① 転送方向を設定（制御権が取得している時）。PpciMasterDIR ② ドライバのバッファを獲得。PpciMemAlloc ③ 転送長セット。PpciSetDataLENG ④ スタート。PpciStart ⑤ タイムアウトチェック及び ステータスで転送終了待ち ⑥ ドライバのバッファ解放。PpciMemFree		
備考	制御権が、取得出来なかった場合（PRVOFF）で、転送方向が、 入力になっていた場合エラーになり、実行されない。 また、この関数は終了ステータス待ちをしているので、終了かエラーに ならないと抜けて来ない。		