ＩＯ－ＰＣＩソフト仕様書

２０００．５／９：初版
２００２．２／２８：第2版　IOpciIOWrite、IopciMMWrite関数の
引数順序変更（☆２）
２００５．１０／１：第3版　Win2000/XPのインストール説明追加

株式会社 ファード

目　　次

1．ＩＯ−ＰＣＩボードのドライバインストール方法

1−1．Ｗｉｎｄｏｗｓ９５／９８／Meの場合

①パソコンの電源を切ってＩＯ−ＰＣＩボードをＰＣＩバスに挿入します。

②パソコンの電源を投入しＷｉｎｄｏｗｓが起動すると、
新しいデバイスとしてＩＯ−ＰＣＩボードが自動検出されます。

③ここで付属のドライバディスクからドライバをインストールします。

④ドライバをインストールしおわったらパソコンを再起動して下さい。

⑤再起動後、[コントロールパネル][システム]のデバイスマネージャに
”ＭＴＤ”としてＩＯ−ＰＣＩボードが表示されていればインストール 終了です。

1−2．ＷｉｎｄｏｗｓＮＴ４．０の場合

①添付ソフトのドライバーをHardDisk のディレクトリにコピーし、
ENABLE.BAT をマウスのダブルクリックで実行します。
（またはDOS 窓で実行します）。
所定のところにドライバファイルがコピーされます。

②パソコンの電源を切ってＩＯ−ＰＣＩボードをＰＣＩバスに挿入します。

③この後、PC を再立ち上げして下さい。

④立ち上がったら、コントロールパネルの「デバイス」を立ち上げ、
「ＩＯ＿ＰＣＩ」を探し、それを「開始」させて下さい。
「開始」できれば、ドライバはインストール完了です。

1−3．Ｗｉｎｄｏｗｓ２０００／ＸＰの場合

①パソコンの電源を切ってＩＯ−ＰＣＩボードをＰＣＩバスに挿入します。

② 立ち上げの途中で、ＰＣボードのドライバーのインストールを要求して
きますので、ＩＯｐｃｉ．ｉｎｆを指定してください。
後は、自動的にインストールをしてくれます。

③インストールが、終わりましたら＜コントロールパネル＞→＜システム＞→
＜ハードウエア＞→＜デバイスマネージャ＞に以下の表示があれば終了です。

Ｆｉｒｄ　ＩＯ−ＰＣＩ
　└ ＩＯ−ＰＣＩ

2. 添付ソフト

2-1. Windows95／98／Me

（1）デバイスドライバー・ファイル

¥driver¥ io_pci.inf（インストール情報）
io_pci.vxd（ドライバ本体）

（2）テストプロ実行ファイル

¥exe¥ Fio_pci.exe（テストプロ）
iopcisb.dll （関数ライブラリィ）

（3）関数ファイル（テストプロソースファイル・ディレクトリィ内）

¥Fio_pci¥ iopcisb.dll（ライブラリィ本体）
iopcisb.lib（コンパイル時の参照）
IOPCIapi.h （関数定義）

（4）テストプロ・ソースファイル

¥Fio_pci¥（ソース一式）
Visual C++で開発したものです。

2-2. WindowsNT

（1）デバイスドライバー・ファイル

¥Driver¥ IOpci.ini（インストール情報）
io_pci.sys（ドライバ本体）
ENABLE.bat（インストール実行バッチファイル）
REGINI.EXE（レジストリ登録）

（2）テストプロ実行ファイル

¥testexe¥ Fio_pci.exe（テストプロ）
iopcisbnt.dll（関数ライブラリィ）

（3）関数ファイル（テストプロソースファイル・ディレクトリィ内）

¥Fio_pciNT¥ iopcisbnt.dll（ライブラリィ本体）
iopcisbnt.lib（コンパイル時の参照）
IOPCIapi.h （関数定義）

（4）テストプロ・ソースファイル

¥Fio_pciNT¥（ソース一式）
Visual C++で開発したものです。

2-3. Windows2000／XP

（1）デバイスドライバー

¥Driver2k-xx¥IOpci.inf（インストール情報）
io_pci.sys（ドライバ本体）

（2）テストプロ実行ファイル

WindowsNTのファイルを使用してください。

（3）関数ファイル

WindowsNTのファイルを使用してください。

（4）テストプロ・ソースファイル

WindowsNTのファイルを使用してください。

3．開発について

3－1．開発環境

ウインドウズ９５／９８／Ｍｅ／ＮＴ４／２０００／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iopcisb.dll（95/98/Me）／
iopcisbnt.dll（NT/2000/XP）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
”iopcisb.lib”(95/98/Me)／”iopcisbnt.lib”(NT/2000/XP)を入力して設定します。
上記ファイルは、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。
また、定義用ファイルとして”IOPCIapi.h”を作成プログラムに
インクルードしてください。
両ファイルは、添付のテストプロソースのディレクトリィにあります。

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。

４－１．ボードのオープン

```
HANDLE   hVxD；          //ハンドル定義
hVxd=IOpciOpen(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。

４－２．ボードのクローズ

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

（注）アプリケーション終了時には、必ず実行する事。

４－３．外部ポートアクセス

（１）モード０の場合

①モードの設定

```
IOpciMODE( hVxD,QMODE0 );   //   モード０設定
```

②ポートのデータの入力／出力方向の設定

```
IOpciMODE0DIR( hVxD,dir1,dir2,dir3,dir4 );
  // dir1～4 に入出力方向をセット
     //  入力：QDIRIN
     //  出力：QDIROUT
```

③ポートの入力

```
DWORD   data;
data=IOpciTPortRead( hVxD );  //TTL バッファの場合
data=IOpciDPortRead( hVxD );  // 差動／オプトバッファの場合（16ビット)
```

ただし、上記入出力方向の設定（TTL)で入力の設定がされていない場合
または、差動／オプトﾊﾞｯﾌｧで入力ﾊﾞｯﾌｧが、搭載されていない場合は、
無効です。

④ポートの出力

```
IOpciTPortWrite( hVxD,out_data,acs,posi );  //TTL バッファの場合
IOpciDPortWrite( hVxD,out_data,acs,posi );
        // 差動／オプトバッファの場合（16ビット)
```

acs：ｱｸｾｽﾓｰﾄﾞで、
QBACS：バイト、QWACS:ワード、QDWACS：ﾀﾞﾌﾞﾙﾜｰﾄﾞになる。

posi：ﾊﾞｲﾄ/ﾜｰﾄﾞの位置で、
ﾊﾞｲﾄｱｸｾｽの場合は、QBYTE0、QBYTE1、QBYTE2、QBYTE3になり
ﾜｰﾄﾞ　〃　QWORD0、QWORD1となります。
ﾀﾞﾌﾞﾙﾜｰﾄﾞの場合ダミーで０をいれてください。

ただし、上記入出力方向の設定（TTL)で出力の設定がされていない場合
または、差動／オプトﾊﾞｯﾌｧで出力ﾊﾞｯﾌｧが、搭載されていない場合は、
無効です。

（2）モード１の場合

①モードの設定

```
IOpciMODE( hVxD,QMODE1 );  //  モード１設定
```

②データモードの設定

```
IOpciM1DMode( hVxD,dmode); // データモードの設定
dmode=QDMOD0：32 ﾋﾞｯﾄｺﾝﾄﾛｰﾙ入力
        QDMOD1：      〃      出力
        QDMOD2：16 ﾋﾞｯﾄｺﾝﾄﾛｰﾙ出力（上位）、16 ﾋﾞｯﾄｺﾝﾄﾛｰﾙ入力（下位）
        QDMOD3：  〃  ﾓｰﾄﾞ 0  I/O（上位）、          〃
        QDMOD4：  〃  ｺﾝﾄﾛｰﾙ出力（上位）、16 ﾋﾞｯﾄﾓｰﾄﾞ 0  I/O（下位）
```

③ データ出力の場合

```
// データポインターを仮に  wrp  として
 IOpciINTClear( hVxD, QIOBFCLR ); // OBF 割り込みステータスクリア
 IOpciTportWrite( hVxD,*wrp,acs,posi );  wrp++;//第１データライト
for( ；；){
  if( IOpciINTRead( hVxD ) & QIOBFSTS ){  // OBF 割り込みチェック
        IOpciINTClear( hVxD, QIOBFCLR ); // OBF 割込ステータスクリア
      IOpciTportWrite( hVxD,*wrp,acs,posi ); wrp++;
       if( ﾃﾞｰﾀ終了 ) break;
    }
}
```

③ データ入力の場合

```
// データポインターを仮に  rdp  として
 IOpciINTClear( hVxD, QIIBFCLR ); // IBF 割り込みステータスクリア
for( ；；){
  if( IOpciINTRead( hVxD ) & QIIBFSTS ){  // OBF 割り込みチェック
        IOpciINTClear( hVxD, QIIBFCLR ); // IBF 割込ステータスクリア
      rdp=IOpciTportRead( hVxD ); rdp++;
       if( ﾃﾞｰﾀ終了 ) break;
    }
}
```

（注）１．モード１入出力は、割り込みマスクをあけて、割り込みにても行えます。
割り込みについては、後述します。
２．モード１入力は、STB 信号にてラッチされたものが入力されます。

（3）モード・パターンの場合

①モードの設定

```
IOpciMODE( hVxD,QMODEP );  //  モードP設定
```

②出力クロックの分周（パターン出力のみ有効）

```
IOpciPCLKDIV( hVxD,1 );   //  1／2の場合
```

③パターンモードの設定

```
IOpciPMODE( hVxD,repeat,i/o);
// repeat= QPREPT_ON(繰り返し)、QPREPT_OFF( 1回のみ)
// i/o= QPDIRIN(パターン入力)、QPDIROUT(パターン出力)
```

④パターンスタート、ストップアドレスの設定

```
IOpciPStartADR( hVxD,0x00000 );   //スタートアドレス
IOpciPStopADR( hVxD,0x10000 );    //ストップアドレス
```

⑤パターンスタート

```
IOpciPSTART( hVxD );
```

⑥パターンストップ

```
IOpciPSTOP(HANDLE ph);
```

//パターンストップは、繰り返しの場合使用します。

⑦パターンエンド

・ビジーステータスチェック

```
if( IOpciPBRead( hVxD)==0 ) 終了;
```

・割り込みステータスチェック

```
IOpciINTClear( hVxD, QPENDCLR );   //割り込みクリア
for( ;; ){
if( IOpciINTRead( hVxD ) & QPENDSTS ) break;
}
```

（注）割り込みステータスリードでチェックする場合は、割り込みマスクをセットしてから、行ってください。
（マスクがクリアされている場合、ドライバで自動的にステータスをクリアします。）

上記の手順は、出力の場合パターンメモリにデータがあるとしてのものです。
パターンメモリのリード／ライトの方法は、後述します。

４－４．パターンメモリアクセス方法

（１）共通

①メモリ確保

```
BYTE *pmem;            // BYTE/WORD/DWORD
pmem=( BYTE *)memalloc( len );
// 確保するメモリの種類は、
// 8ﾋﾞｯﾄ：BYTE、16ﾋﾞｯﾄ：WORD、32ﾋﾞｯﾄ：DWORD
// にする。これは、出荷時のメモリ搭載によります。
```

（２）メモリライト（ﾊﾞｲﾄﾒﾓﾘとして説明）

①メモリブロックライト

```
IOpciMMBLKWR(  hVxD,pmem,0,len,QBACS);
// len は、最大
// 8ﾋﾞｯﾄ：512k/1M ﾊﾞｲﾄ、16ﾋﾞｯﾄ：1M/2Mﾊﾞｲﾄ、32ﾋﾞｯﾄ：2M/4Mﾊﾞｲﾄ
// 設定出来る。パターンとしては、どれでも 512k/1M パターンである。
```

（３）メモリリード（ﾊﾞｲﾄﾒﾓﾘとして説明）

①メモリブロックライト

```
IOpciMMBLKRD(  hVxD,pmem,0,len,QBACS);
```

４－５．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

```
IOpciINTEntry( hVxD,(DWORD)this->GetSafeHwnd() );
    //  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 CiopcisampView::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 & QINT1STS)!=0){//外部１割り込み
            cs+="INT1 割込、";
       }
       if((wParam & QINT2STS)!=0){//外部２割り込み
            cs+=" INT2 割込、";
       }
       if((wParam & QIIBFSTS)!=0){//IBF 立ち上がり割り込み
            cs+=" IBF 割込、";
       }
       if((wParam & QIOBFSTS)!=0){// OBF 立ち下がり割り込み
            cs+="OBF 割込、";
       }
       if((wParam & QPENDSTS)!=0){// パターンエンド割り込み
            cs+="PEND 割込、";
       }
       cs+="¥n 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);
}
```

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 IopcisampView)、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。
割り込み処理（ドライバ）で、割り込みステータスは、マスクがかかっていないビットについてはクリアされます。

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．定義定数

定義定数については、“ iopciapi.h ”のファイルに入っています。

5－1．構造体

・ＰＣＩコンフィグレジスタ情報用構造体

```
struct      IOPCI_CONFIG_DATA{
                WORD    DeviceID;
                WORD    BenderID;
                WORD    Status;
                WORD    Command;
                DWORD   ClassCode;
                BYTE    RevisionID;
                BYTE    Bist;
                BYTE    HeaderType;
                BYTE    LatencyTimer;
                BYTE    CacheLineSize;
                DWORD   BaseAddress1;
                DWORD   BaseAddress2;
                BYTE    MaxLatency;
                BYTE    MinGrant;
                BYTE    IntPin;
                BYTE    IntLine;
};
```

IOpciConfigRead 関数で使用する。

5－2．Ｉ／Ｏアドレスオフセット

```
#define  QTTLIO      0x00  //  外部データ入出力ポート（TTLﾊﾞｯﾌｧ）
#define  QDIFIO      0x04  //        〃              （差動／オプトﾊﾞｯﾌｧ）
#define  QMODE       0x08  //  モード設定
#define  QMOD0CNT    0x0c  //  モード０コントロール
#define  QMOD1CNT    0x10  //  モード１    〃
#define  QINTSET     0x14  //  割り込みマスク／クリア
#define  QTRGCNT     0x18  //  トリガコントロール
#define  QMODPCNT    0x1c  //  パターンコントロール
#define  QPTSTART    0x20  //  パターンスタート
#define  QPTSADR     0x24  //  パターンスタートアドレス
#define  QPTEADR     0x28  //  パターンエンドアドレス
#define  QDIPSW      0x2c  //  ディップスイッチリード
```

５－３．コントロール及び状態定義

（１）モード

```
#define  QMODE0      0    //モード０
#define  QMODE1      1    //モード１
#define  QMODEP      2    //パターンモード
```

（２）モード１のデータモード

```
#define  QDMOD0      0    //データモード０
#define  QDMOD1      1    //  〃      １
#define  QDMOD2      2    //  〃      ２
#define  QDMOD3      3    //  〃      ３
#define  QDMOD4      4    //  〃      ４
```

（３）データ入出力方向

```
#define  QDIRIN      0    //入力
#define  QDIROUT     1    //出力
```

（４）割り込みマスク

```
#define  QINT1MSK    0x01    //外部ＩＮＴ１　割り込みマスク
#define  QINT2MSK    0x02    //外部ＩＮＴ２　　〃
#define  QIIBFMSK    0x04    //ＩＢＦ立ち上がり　〃
#define  QIOBFMSK    0x08    //ＯＢＦ立ち下がり　〃
#define  QPENDMSK    0x10    //パターンエンド　　〃
```

（５）割り込みステータスクリア

```
#define  QINT1CLR    0x01    //外部ＩＮＴ１　割り込みステータスクリア
#define  QINT2CLR    0x02    //外部ＩＮＴ２　　〃
#define  QIIBFCLR    0x04    //ＩＢＦ立ち上がり　〃
#define  QIOBFCLR    0x08    //ＯＢＦ立ち下がり　〃
#define  QPENDCLR    0x10    //パターンエンド　　〃
#define  QSIBFCLR    0x40    //ＩＢＦステータスクリア
#define  QSOBFCLR    0x80    //ＯＢＦ　　〃
```

（６）割り込みステータス

```
#define  QINT1STS    0x01    //外部ＩＮＴ１　割り込みステータス
#define  QINT2STS    0x02    //外部ＩＮＴ２　　〃
#define  QIIBFSTS    0x04    //ＩＢＦ立ち上がり　〃
#define  QIOBFSTS    0x08    //ＯＢＦ立ち下がり　〃
#define  QPENDSTS    0x10    //パターンエンド　　〃
#define  QSIBFSTS    0x40    //ＩＢＦステータスクリア
#define  QSOBFSTS    0x80    //ＯＢＦ　　〃
```

（７）メモリ、Ｉ／Ｏアクセスモード

```
#define  QBACS       0    // バイト　アクセス
#define  QWACS       1    // ワード　アクセス
#define  QDWACS      2    // ダブルワード　アクセス
```

（８）ライト位置

・バイトアクセス時

```
#define  QBYTE0      0   //第１バイト目（Ｄ０～７）
#define  QBYTE1      1   //第２　〃　　（Ｄ８～１５）
#define  QBYTE2      2   //第３　〃　　（Ｄ１６～２３）
#define  QBYTE3      3   //第４　〃　　（Ｄ２４～３１）
```

・ワードアクセス時

```
#define  QWORD0      0     //第１ワード目（DO～１５）
#define  QWORD1      2     //第２　〃　　（Ｄ１６～３２）
```

（９）トリガ極性

```
#define  QTRGPLUS    1    // + （正）
#define  QTRGMINUS   0    // - （負）
```

（１０）パターンコントロール

・パターンの繰り返し

```
#define  QPREPT_ON    1    // 繰り返し
#define  QPREPT_OFF   0    // １回のみ（繰り返しなし）
```

・パターン入出力

```
#define  QPDIRIN      0    // パターン入力
#define  QPDIROUT     1    //　　〃　出力
```

・パターン実行中

```
#define  QPTBUSY      1    //パターンビジー
```

6．関数一覧

ファイル“ｉｏｐｃｉｓｂ．ｄｌｌ”でＤＬＬ形式で供給

No.	関数名	概略内容
1	IopciOpen	ボードのオープン（使用開始）
2	IOpciClose	ボードのクローズ（使用終了）
3	IopciMode	I/O モードの設定
4	IOpciMode0DIR	I/O モード０のデータ方向設定
5	IOpciTPortRead	バッファＴＴＬポートのリード
6	IOpciTPortWrite	〃 のライト
7	IOpciDPortRead	バッファ差動／オプト・ポートのリード
8	IOpciDPortWrite	〃 のライト
9	IOpciM1DMode	I/O モード１のデータモードの設定
10	IOpciTRG1W	トリガ１のパルス幅設定
11	IOpciTRG1P	〃 パルス極性の設定
12	IOpciTRG1OUT	〃 パルス出力
13	IOpciTRG2W	トリガ２のパルス幅設定
14	IOpciTRG2P	〃 パルス極性の設定
15	IOpciTRG2OUT	〃 パルス出力
16	IOpciPCLKDIV	出力パターンクロックの分周
17	IOpciPMODE	入出力パターンモードの設定
18	IOpciPSTART	〃 スタート
19	IOpciPSTOP	〃 ストップ
20	IOpciPBRead	〃 ビジー・リード
21	IOpciPStartADR	〃 スタートアドレス設定
22	IOpciPStopADR	〃 ストップ 〃
23	IOpciDIPSWRead	ディップスイッチのリード
24	IOpciINTMask	割り込みマスクの設定
25	IOpciINTClear	割り込みステータスクリア
26	IOpciINTRead	〃 リード
27	IOpciINTEntry	割り込み処理登録
28	IopciConfigRead	PCI コンフィグ・レジスタ リード
29	IOpciIORead	PCI I/O リード
30	IOpciIOWrite	〃 ライト
31	IOpciMMRead	PCI メモリ リード
32	IOpciMMWrite	〃 ライト
33	IOpciMMBLKRD	〃 ブロック リード
34	IOpciMMBLKWR	〃 ブロック ライト
35	IopciBoardNum	使用枚数リード

7．関数説明

関数 1	ボードのオープン
ﾌﾟﾛﾄﾀｲﾌﾟ	HANDLE　IOpciOpen(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。 DIP-SW　“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。 エラー：ボードが無い、または既にｵｰﾌﾟﾝされている時は、 INVALID_HANDLE_VALUE（-1）が、戻される。
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 2	ボードのクローズ
ﾌﾟﾛﾄﾀｲﾌﾟ	int IOpciClose(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ルをクローズする。（クローズ）
備 考	

関数 3	I/0モードの設定
ﾌﾟﾛﾄﾀｲﾌﾟ	void IOpciMode(HANDLE ph,BYTE p1)
引 数	HANDLE ph：ボードオープン時の戻り値 BYTE p1：ポートのモード値 QMODE0：モード０（単純入出力） QMODE1：モード１（ハンドシェークによる入出力） QMODEP：パターン入出力モード
戻り値	なし
内 容	ポートの動作モードを設定する。 詳細は、ハード仕様書を参照してください。
備 考	パターンモードは、オプションとしてメモリ及び発振器が必要です。

関数 4	I/0モード0のデータ方向設定
ﾌﾟﾛﾄﾀｲﾌﾟ	void IOpciMode0DIR(HANDLE ph,BYTE p1,BYTE p2 ,BYTE p3,BYTE p4)
引 数	HANDLE ph：ボードオープン時の戻り値 BYTE p1 ：第1バイト目の方向（最下位バイト） BYTE p2 ：第2バイト目の方向 BYTE p3 ：第3バイト目の方向 BYTE p4 ：第4バイト目の方向 （最上位バイト） （設定値） QDIRIN ：入力モード QDIROUT ：出力モード
戻り値	なし。
内 容	モード0時のＴＴＬポートの入出力の設定を行う。
備 考	差動／オプトバッファを使用した場合、出荷時固定となり 設定は、無効である。

関数 5	バッファＴＴＬポートのリード
ﾌﾟﾛﾄﾀｲﾌﾟ	DWORD IOpciTPortRead(HANDLE ph)
引 数	HANDLE ph：ボードオープン時の戻り値
戻り値	DWORD：ポートの 32 ビットデータ入力。 モード０入力モード：ポート直接入力データ。 モード１入力モード：ラッチされた入力データ。
内 容	バッファＴＴＬポートのデータ入力を行う。 出力に指定されたポートについては、無効です。
備 考	

関数 6	バッファＴＴＬポートのライト
ﾌﾟﾛﾄﾀｲﾌﾟ	void IOpciTPortWrite(HANDLE ph,DWORD p1,BYTE p2,BYTE p3)
引 数	HANDLE ph：ボードオープン時の戻り値 DWORD p1：出力データ。 BYTE p2:：アクセスモード QBACS：バイト アクセス QWACS：ワード 〃 QDWACS：ダブルワード 〃 BYTE p3：アクセス位置 バイトアクセス時：QBYTE0、QBYTE1、QBYTE2、QBYTE3 ワード 〃 ：QWORD0、QWORD1
戻り値	なし。
内 容	バッファ TTL ポートにデータを出力する。 入力に指定されたポートにつては、無効である。
備 考	アクセス位置が何れであっても、データは、下位詰めで 設定する。

関数 7	バッファ差動／オプト・ポートのリード
ﾌﾟﾛﾄﾀｲﾌﾟ	DWORD IOpciDPortRead(HANDLE ph)
引 数	HANDLE ph：ボードオープン時の戻り値
戻り値	DWORD：データ１６ビット入力
内 容	バッファ差動／オプトポートの入力を行う。 出力のポートについては、無効である。
備 考	

関数 8	バッファ差動／オプト・ポートのライト
ﾌﾟﾛﾄﾀｲﾌﾟ	void IOpciDPortWrite(HANDLE ph,DWORD p1,BYTE p2,BYTE p3)
引 数	HANDLE ph：ボードオープン時の戻り値 DWORD p1：出力データ BYTE p2：アクセスモード QBACS：バイトアクセス QWACS：ワード 〃 BYTE p3：アクセス位置 QBYTE0：第１バイト目 QBYTE2：第２バイト目 バイトアクセス時のみ有効。
戻り値	なし。
内 容	バッファ差動／オプトポートのデータを出力する。 入力ポートに付いては、無効です。
備 考	アクセス位置が何れであってもデータは、下位詰めで 設定する。

関数　9	I/0モード１のデータモードの設定
ﾌﾟﾛﾄﾀｲﾌﾟ	void　IOpciM1DMode(HANDLE ph,BYTE p1)
引　数	HANDLE　ph：ボードオープン時の戻り値 BYTE　p1：データモード値 QDMOD0：32ﾋﾞｯﾄｺﾝﾄﾛｰﾙ入力 QDMOD1：　〃　出力 QDMOD2：16ﾋﾞｯﾄｺﾝﾄﾛｰﾙ出力（上位)、16ﾋﾞｯﾄｺﾝﾄﾛｰﾙ入力（下位） QDMOD3：　〃　ﾓｰﾄﾞ 0　I/O（上位)、　〃 QDMOD4：　〃　ｺﾝﾄﾛｰﾙ出力（上位)、16ﾋﾞｯﾄﾓｰﾄﾞ 0　I/O（下位） （注）上記ﾓｰﾄﾞは、バッファにより多少違います。 詳細は、ハード仕様書を参照してください。 なお、ここでは全てＴＴＬについて、説明している。
戻り値	なし。
内　容	モード１のデータモードの設定をおこなう。 モード０、１の混在モードもある。
備　考	

関数 10	トリガ１のパルス幅設定
ﾌﾟﾛﾄﾀｲﾌﾟ	void IOpciTRG1W(HANDLE ph,BYTE p1)
引 数	HANDLE ph：ボードオープン時の戻り値 BYTE p1：パルス幅 設定値：０～１５ パルス幅＝（設定値＋１）＊１０ｕＳ
戻り値	なし。
内 容	トリガ１のパルス幅の設定を行う。 １０～１６０ｕＳまで設定できる。
備 考	

関数 11	トリガ１のパルス極性の設定
ﾌﾟﾛﾄﾀｲﾌﾟ	void IOpciTRG1P(HANDLE ph,BYTE p1)
引 数	HANDLE ph：ボードオープン時の戻り値 BYTE p1：パルス極性 QTRGPLUS ：正極性 QTRGMINUS：負極性
戻り値	なし。
内 容	トリガ１のパルス極性を設定する。 設定すると、正極性の場合“０”で負極性の場合“１”に現在のレベルになる。
備 考	

関数 12	トリガ1のパルス出力
ﾌﾟﾛﾄﾀｲﾌﾟ	void IOpciTRG1OUT(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	トリガ1パルスを出力する。
備 考	

関数 13	トリガ2のパルス幅設定
ﾌﾟﾛﾄﾀｲﾌﾟ	void IOpciTRG2W(HANDLE ph,BYTE p1)
引 数	HANDLE ph：ボードオープン時の戻り値 BYTE p1：パルス幅 設定値：０～１５ パルス幅＝（設定値＋１）＊１０ｕＳ
戻り値	なし。
内 容	トリガ2のパルス幅の設定を行う。 １０～１６０ｕＳまで設定できる。
備 考	

関数 14	トリガ２のパルス極性の設定
ﾌﾟﾛﾄﾀｲﾌﾟ	void IOpciTRG2P(HANDLE ph,BYTE p1)
引 数	HANDLE ph：ボードオープン時の戻り値 BYTE p1：パルス極性 QTRGPLUS ：正極性 QTRGMINUS：負極性
戻り値	なし。
内 容	トリガ２のパルス極性を設定する。 設定すると、正極性の場合“０”で負極性の場合“１”に現在のレベルはなる。
備 考	

関数 15	トリガ2のパルス出力
ﾌﾟﾛﾄﾀｲﾌﾟ	void IOpciTRG2OUT(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	トリガ2パルスを出力する。
備 考	

関数　16	出力パターンクロックの分周
ﾌﾟﾛﾄﾀｲﾌﾟ	void　IOpciPCLKDIV(HANDLE ph,BYTE p1)
引　数	HANDLE　ph：ボードオープン時の戻り値 BYTE　p1：クロックの分周値 設定値：０～３ 分周値＝発振器の周波数／（２＾p1） 1/1、1/2、1/4、1/8 分周出来る。
戻り値	なし。
内　容	パターン出力時のクロックの分周を行う。
備　考	パターンオプションが無いときは、無効です。

関数 17	入出力パターンモードの設定
ﾌﾟﾛﾄﾀｲﾌﾟ	void IOpciPMODE(HANDLE ph,BYTE p1,BYTE p2)
引 数	HANDLE ph：ボードオープン時の戻り値 BYTE p1：パターン繰り返しモード QPREPT_ON：繰り返しON QPREPT_OFF： 〃 OFF（1回のみ行う） BYTE p2：パターン入出力 QPDIRIN :パターン入力モード QPDIROUT:パターン出力モード
戻り値	なし。
内 容	パターンの実行モードを設定する。 ・繰り返し ・入出力
備 考	パターンオプションが無いときは、無効です。

関数 18	入出力パターンスタート
ﾌﾟﾛﾄﾀｲﾌﾟ	void IOpciPSTART(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	パターンを開始する。
備 考	パターンオプションが無いときは、無効です。

関数 19	入出力パターンストップ
ﾌﾟﾛﾄﾀｲﾌﾟ	void IOpciPSTOP(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	現在実行しているパターン入出力を、強制停止させる。
備 考	パターンオプションが無いときは、無効です。

関数 20	入出力パターンビジー・リード
プロトタイプ	BYTE IOpciPBRead(HANDLE ph)
引 数	HANDLE ph：ボードオープン時の戻り値
戻り値	BYTE：:パターン実行中フラグリード。 NULL ：停止 QPTBUSY：実行中（ビジー）
内 容	パターン実行中フラグをリードする。
備 考	パターンオプションが無いときは、無効です。

関数 21	入出力パターンスタートアドレス設定
ﾌﾟﾛﾄﾀｲﾌﾟ	void IOpciPStartADR(HANDLE ph,DWORD p1)
引 数	HANDLE ph :ボードオープン時の戻り値 DWORD p1 :パターンスタートアドレス **設定範囲：**0x00000～0xfffff（**1**Ｍ**パターン**）
戻り値	**なし。**
内 容	**開始パターンアドレスを設定する。**
備 考	**パターンオプションが無いときは、無効です。**

関数 22	入出力パターンストップアドレス設定
ﾌﾟﾛﾄﾀｲﾌﾟ	void IOpciPStopADR(HANDLE ph,DWORD p1)
引 数	HANDLE ph：ボードオープン時の戻り値 BYTE p1：エンドパターンアドレス 設定値：0x00000～0xfffff（1M パターン）
戻り値	なし。
内 容	終了パターンアドレスを設定する。
備 考	パターンオプションが無いときは、無効です。

関数 23	ディップスイッチのリード
ﾌﾟﾛﾄﾀｲﾌﾟ	BYTE IOpciDIPSWRead(HANDLE ph)
引 数	HANDLE ph：ボードオープン時の戻り値
戻り値	BYTE :ディップスイッチ値 ON：”1” OFF：”0”
内 容	ボードのディップスイッチをリードする。 注）スイッチの１～４は、ボード複数枚使用時の識別値として 使用されます。（パワーON時のみ）
備 考	

関数 24	割り込みマスクの設定
ﾌﾟﾛﾄﾀｲﾌﾟ	void IOpciINTMask(HANDLE ph,BYTE p1)
引 数	HANDLE ph：ボードオープン時の戻り値 BYTE p1：マスク設定データ （マスクデータ） QINT1MSK：外部 INT1 割り込みマスク QINT2MSK： 〃 INT2 〃 QIIBFMSK：IBF 立ち上がり 〃 QIOBFMSK：OBF 立ち下がり 〃 QPENDMSK：パターンエンド 〃
戻り値	なし。
内 容	割り込みをマスク（禁止）する。値が無い時は、クリア（許可）する。 割り込みのマスクは、割り込みステータスに影響を与えない。
備 考	

関数 25	割り込みステータスクリア
ﾌﾟﾛﾄﾀｲﾌﾟ	void IOpciINTClear(HANDLE ph,BYTE p1)
引 数	HANDLE ph：ボードオープン時の戻り値 BYTE p1：ステータスクリアデータ （クリアデータ） QINT1CLR：外部 INT1 割り込みクリア QINT2CLR： 〃 INT2 〃 QIIBFCLR：IBF 立ち上がり 〃 QIOBFCLR：OBF 立ち下がり 〃 QPENDCLR：パターンエンド 〃 QSIBFCLR：IBF ステータスクリア QSOBFCLR：OBF 〃
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。 クリアされたステータスは、メッセージにより渡される。 （注）OBF、IBF はステータスであり、割り込みはこのステータスの エッジにより作成される。
備 考	

関数 26	割り込みステータスリード
プロトタイプ	BYTE IOpciINTRead(HANDLE ph)
引 数	HANDLE ph：ボードオープン時の戻り値
戻り値	BYTE: 割り込みステータスデータ QINT1STS：外部 INT1 割り込みステータス QINT2STS： 〃INT2 〃 QIIBFSTS：IBF 立ち上がり 〃 QIOBFSTS：OBF 立ち下がり 〃 QPENDSTS：パターンエンド 〃 QSIBFSTS：IBF ステータス（割り込みではない。） QSOBFSTS：OBF 〃 （ 〃 ）
内 容	割り込みステータスデータをリードする。
備 考	

関数 27	割り込み処理登録
ﾌﾟﾛﾄﾀｲﾌﾟ	void IOpciINTEntry(HANDLE ph,DWORD p1)
引 数	HANDLE ph：ボードオープン時の戻り値（ハンドル） DWORD p1：アプリケーションのウインドウハンドル
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	複数枚、１アプリケーションで使用する場合、ウインドウハンドルは、 同じになるので、メッセージを送るときパラメータとして ボードハンドル（ph）も送りますので、それによりどのボードかの 判断をする。

関数 28	PCI コンフィグ・レジスタ リード
ﾌﾟﾛﾄﾀｲﾌﾟ	void IOpciConfigRead(HANDLE ph,struct IOPCI_CONFIG_DATA *p1)
引 数	HANDLE ph：ボードオープン時の戻り値 IOPCI_CONFIG_DATA *p1：PCI コンフィグレジスタ構造体ポインター 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の内容を IOPCI_CONFIG_DATA 構造体内に格納する。
備 考	

関数 29	PCI I/0 リード
ﾌﾟﾛﾄﾀｲﾌﾟ	DWORD IOpciIORead(HANDLE ph,DWORD p1,BYTE p2)
引 数	HANDLE ph：ボードオープン時の戻り値 DWORD p1： I/O アドレスオフセット BYTE p2：アクセスモード QBACS：バイトアクセス QWACS：ワードアクセス QDWACS：ダブルワードアクセス
戻り値	DWORD：リードデータ （下位詰め）
内 容	このボードのＩ／Ｏをリードする。
備 考	

関数 30	PCI I/0 ライト
ﾌﾟﾛﾄﾀｲﾌﾟ	void IOpciIOWrite(HANDLE ph,DWORD p1,BYTE p2,DWORD p3)
引 数	HANDLE ph：ボードオープン時の戻り値 DWORD p1：I/O アドレスオフセット ☆2 BYTE p2：アクセスモード QBACS：バイトアクセス QWACS：ワードアクセス QDWACS：ダブルワードアクセス DWORD p3：ライトデータ（データは、下位詰め）
戻り値	なし。
内 容	このボードのI／Oにライトする。 アクセスモードにより、不要なバイト、ワードをアクセスしないように出来る。
備 考	

関数 31	PCI メモリ リード
ﾌﾟﾛﾄﾀｲﾌﾟ	DWORD IOpciMMRead(HANDLE ph,DWORD p1,BYTE p2)
引 数	HANDLE ph：ボードオープン時の戻り値 DWORD p1：メモリアドレスオフセット BYTE p2：アクセスモード QBACS：バイトアクセス QWACS：ワードアクセス QDWACS：ダブルワードアクセス
戻り値	DWORD：パターンメモリ・リードデータ （データは、下位詰め）
内 容	パターンメモリのデータをリードする。 アクセスモードは、 パターンメモリ8ビットの場合、バイトのみで 〃 １６ 〃 、バイト／ワードで 〃 ３２ 〃 、バイト／ワード／ダブルワードを 設定できる。
備 考	(注) オフセットは、ワードアクセスの場合ワード境界 ダブルワードアクセスの場合ダブルワード境界の設定をしてください。 ワード境界：２の倍数 ダブルワード境界：４の倍数

関数 32	PCI メモリ ライト
ﾌﾟﾛﾄﾀｲﾌﾟ	void IOpciMMWrite(HANDLE ph,DWORD p1,BYTE p2,DWORD p3)
引 数	HANDLE ph：ボードオープン時の戻り値 DWORD p1：メモリアドレスオフセット ☆2 BYTE p2：アクセスモード QBACS：バイトアクセス QWACS：ワードアクセス QDWACS：ダブルワードアクセス DWORD p3：ライトデータ（下位詰め）
戻り値	なし。
内 容	パターンメモリのデータをライトする。 アクセスモードは、 パターンメモリ8ビットの場合、バイトのみで 〃 16 〃 、バイト／ワードで 〃 32 〃 、バイト／ワード／ダブルワードを 設定できる。
備 考	（注）オフセットは、ワードアクセスの場合ワード境界 ダブルワードアクセスの場合ダブルワード境界の設定をしてください。 ワード境界：2の倍数 ダブルワード境界：4の倍数

関数 33	PCI メモリ ブロック リード
ﾌﾟﾛﾄﾀｲﾌﾟ	void IOpciMMBLKRD(HANDLE ph,void *p1,DWORD p2, DWORD p3,BYTE p4)
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のポインター DWORD p2：パターンメモリのバイトオフセット DWORD p3：リードバイト数 BYTE p3：アクセスモード QBACS：バイトアクセス QWACS：ワードアクセス QDWACS：ダブルワードアクセス
戻り値	なし。
内 容	パターンメモリのデータをブロックでリードする。 アクセスモードは、 パターンメモリ8ﾋﾞｯﾄの場合、バイトのみで 〃 １６ 〃 、バイト／ワードで 〃 ３２ 〃 、バイト／ワード／ダブルワードを 設定できる。
備 考	(注) オフセットは、ワードアクセスの場合ワード境界 ダブルワードアクセスの場合ダブルワード境界の設定をしてください。 ワード境界：２の倍数 ダブルワード境界：４の倍数

関数 34	PCI メモリ ブロック ライト
プロトタイプ	void IOpciMMBLKWR(HANDLE ph,void *p1,DWORD p2, DWORD p3,BYTE p4)
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のポインター DWORD p2：パターンメモリのバイトオフセット DWORD p3：ライトバイト数 BYTE p3：アクセスモード QBACS：バイトアクセス QWACS：ワードアクセス QDWACS：ダブルワードアクセス
戻り値	なし。
内 容	パターンメモリのデータをブロックでライトする。 アクセスモードは、 パターンメモリ8ビットの場合、バイトのみで 〃 １６ 〃 、バイト／ワードで 〃 ３２ 〃 、バイト／ワード／ダブルワードを 設定できる。
備 考	(注) オフセットは、ワードアクセスの場合ワード境界 ダブルワードアクセスの場合ダブルワード境界の設定をしてください。 ワード境界：２の倍数 ダブルワード境界：４の倍数

関数 35	使用枚数リード
ﾌﾟﾛﾄﾀｲﾌﾟ	DWORD IOpciBoardNum(HANDLE ph)
引 数	HANDLE ph：ボードオープン時の戻り値
戻り値	DWORD：“ＩＯ－ＰＣＩ”ボードの枚数。
内 容	現在パソコンに搭載されている“ＩＯ－ＰＣＩ”ボードの枚数を 取得する。
備 考	