Ｇ－ＰＣＩ３ソフト仕様書

１９９９．１１／２７　　第一版
２００５．１０／４　　第２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．Ｇ－ＰＣＩ３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Ｇ－ＰＣＩ３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Ｇ－ＰＣＩ３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終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P―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Ｇ－ＰＣＩ３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、
「Ｇ＿ＰＣＩ３」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Ｇ－ＰＣＩ３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、Ｇｐｃｉ３．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 Ｇ－ＰＣＩ３
　└ Ｇ－ＰＣＩ３

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　ｇ_ｐｃｉ３．ｉｎｆ（インストール情報）
ｇ_ｐｃｉ３．ｖｘｄ（ドライバ本体）

（2）テストプロ実行ファイル

￥SMPEXE￥　ｓａｍｐｌｅ．ｅｘｅ　（テストプロ）
Ｇｐｃｉ３ｓｂ．ｄｌｌ　（関数ライブラリィ）

（3）関数ファイル（テストプロ・ソースファイル内）

Ｇｐｃｉ３ｓｂ．ｄｌｌ（ライブラリィ本体）
Ｇｐｃｉ３ｓｂ．ｌｉｂ（コンパイル時の参照）
ｇｐｃｉ３ａｐｉ．ｈ　（関数定義）

（4）テストプロ・ソースファイル

￥ＳＡＭＰＬＥ－ｘ￥（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￥　Ｇｐｃｉ３．ｉｎｉ　（インストール情報）
ｇ_ｐｃｉ３．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（レジストリィ登録）

（2）テストプロ実行ファイル

￥ｅｘｅ￥　ｓａｍｐｌｅ．ｅｘｅ　（テストプロ）
Ｇｐｃｉ３ｓｂｎｔｃ．ｄｌｌ（関数ライブラリィ）

（3）関数ファイル（テストプロ・ソースファイル内）

Ｇｐｃｉ３ｓｂｎｔｃ．ｄｌｌ（ライブラリィ本体）
Ｇｐｃｉ３ｓｂｎｔｃ．ｌｉｂ（コンパイル時の参照）
Ｇｐｃｉ３ａｐｉｎｔｃ．ｈ　（関数定義）

（4）テストプロ・ソースファイル

￥ｓａｍｐｌｅ－ｎｔｃ－ｘ￥（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－Ｒｘｘ￥Ｇｐｃｉ３．ｉｎｆ（インストール情報）
ｇ_ｐｃｉ３．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
\sample-x\gpci3sb.dll（95/98/Me）、
\sample-ntc-x\gpci3sbntc.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
”gpci3sb.lib”(95/98/Me)、“gpci3sbntc.lib”（NT/2000･XP）を入力して
設定します。
”xxx.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。
また、定義用ファイルとして”gpci3api.h”(95/98/Me)、”gpci3apintc.h”(2000/XP)
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・ソースファイル・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。

４－１．ボードのオープン

```
HANDLE　hVxD；　　//ハンドル定義
hVxd=Gpci3Open( 0 );　// 引数(0)は、基板番号で０～になります。
                      // １枚搭載の場合は、無条件に０となります。
```

（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=Gpci3Close( hVxD );
```

（注）アプリケーション終了時には、必ず実行する事。

４－３．ＰＣＩ　Ｉ／Ｏアクセス

ＰＣＩのＩ／Ｏは、ＰＣＩ－ＦＰＧＡに割り当てられていて、マスタ、
ターゲット両モードでアクセスできます。

（１）ライト

```
Gpci3IOWrite( hVxD,offset,data ); //　３２ビットアクセスのみ。
```

（２）リード

```
data=Gpci3IORead( hVxD,offset,data ); //３２ビットアクセスのみ。
```

リードもライトも必要なデータに関しては、個別関数を用意していますので
そちらをお使いください。

４－４．ターゲットモード

（１）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）
このモードは、開発ＦＰＧＡへのメモリレジスタにアクセスするモードです。
ＰＣＩのメモリエリアは、開発用ＦＰＧＡ１、２に割り当てられています。

```
Gpci3AccessCnt(hVxD,QTARGMODE);
```

（２）メモリ（レジスタ）ライト

```
Gpci3MMWrite( hVxD,offset,data ); //　３２ビットアクセスのみ
```

（３）メモリ（レジスタ）リード

```
data=Gpci3MMRead( hVxD,offset ); //　３２ビットアクセスのみ
```

４－５．マスタモード

（共通１）開発ＦＰＧＡ側のマスタースタートを実行し、待ち状態にする。
開発ＦＰＧＡ側のスタート時の操作は、ＦＰＧＡの設計によるので
ここでは、省略します。

（共通２）マスタモード設定

```
Gpci3AccessCnt（ hVxD,QMSTMODE );
```

（１）マスタリード（　開発ＦＰＧＡ１／２→ＰＣＩ－ＦＰＧＡ）

① 転送方向を設定

```
Gpci3MasterDIR( hVxd,QDIRIN );
```

②ドライバのバッファを確保

```
int   ret;
ret=Gpci3MemAlloc( 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。
```

③アプリケーションバッファを確保

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

④　転送長を設定

```
Gpci3DataLENG( hVxD,data_size );
        //転送長は、data_size を設定
```

⑤スタート

```
int   ret;
ret=Gpci3Start( hVxD );   //   ret=0 の時正常
                          //   エラーは、ライブラリィ説明を参照。
```

⑥ステータスリードし終了をチェックする。

```
BYTE   status;
status=Gpci3StatusRead( hVxD );
//   status のビット 7="0"の時終了。
// 割り込みでも終了を検知する事ができます。
// 割り込みについては、割り込みの説明の項を参照のこと。
```

⑦ドライバのバッファをリードする。

```
Gpci3ReadBuff( hVxD,buf,data_size );
```

⑧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
Gpci3MemFree( hVxD )；    //ドライババッファ解放
```

（2）マスタライト（ＰＣＩ－ＦＰＧＡ→開発ＦＰＧＡ１、２）

① 転送方向を設定

```
Gpci3MasterDIR( hVxd,QDIROUT );
```

②ドライバのバッファを確保

```
int   ret;
ret=Gpci3MemAlloc( 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。
```

③アプリケーションバッファを確保

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

④転送長を設定

```
Gpci3DataLENG( hVxD,data_size );
      //転送長は、data_size を設定
```

⑤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。

⑥ドライバのバッファにライトする。

```
Gpci3WriteBuff( hVxD,buf,data_size );
```

⑦スタート

```
int   ret;
ret=Gpci3Start( hVxD );   //   ret=0 の時正常
                          //   エラーは、ライブラリィ説明を参照。
```

⑧ステータスリードし終了をチェックする。

```
BYTE   status;
status=Gpci3StatusRead( hVxD );
//   status のビット 7="0"の時終了。
//割り込みにて、終了が検知できます。
```

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
Gpci3MemFree( 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

```
Gpci3INTEntry( 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 Cgpci3sampView::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 & QENDSTS)!=0){//正常終了
          cs+="正常終了、";
      }
      if((wParam & QSTOPSTS)!=0){//強制終了
          cs+="強制終了、";
      }
      if((wParam & QREQSTS)!=0){//開始要求
          cs+="開始要求、";
      }
      if((wParam & QIO1STS)!=0){// I/O 1
          cs+="IODA1 割込、";
      }
      if((wParam & QIO0STS)!=0){//  I/O 0
          cs+=" IODA0 割込、";
      }

      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);
}
```

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 Gpci3sampView)、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、“ gpci3api.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     GPCI3_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;
};
```

Gpci3ConfigRead 関数で使用する。

５－２．エラーコード

```
//スタート時エラーステータス
#define   ERROR_START1   -1   //  マスタモードになっていない。
#define   ERROR_START2   -2   //  転送バイト数を設定していない。
#define   ERROR_START3   -3   //  転送バッファを取得あいていない。
#define   ERROR_START4   -4   //  転送バッファより転送バイト数が大きい
```

５－３．Ｉ／Ｏアドレスオフセット

```
// I/O address offset
#define   QMODEIO      0x00    // 動作モードレジスタ
#define   QMSTADRIO    0x04    //  ＰＣＩ転送アドレスレジスタ
#define   QMSTLENIO    0x08    //  ＰＣＩ転送バイト数レジスタ
#define   QMSTCTLIO    0x0c    //  マスタコントロールレジスタ
#define   QIODAIO      0x10    //  汎用Ｉ／Ｏレジスタ
#define   QINTSETIO    0x14    //  割り込みマスク、クリアレジスタ
#define   QINTSTSIO    0x18    //  割り込みステータスレジスタ
#define   QMSTSTSIO    0x1c    //   マスタ転送ステータスレジスタ
```

５－４．動作モード用定数

```
//開発 FPGA 供給クロック選択（Gpci3Clock で使用）
  #define QPCICLK  0  // PCI クロック（33MHz）選択
  #define QEXTCLK  1  //  外部（ボード上）発振器選択
//データ方向選択（Gpci3IODADIR、Gpci3MasterDIR で使用）
  #define QDIRIN    0  // 入力
  #define QDIROUT  1  // 出力
//マスタ／ターゲットアクセス選択（Gpci3AccessCnt で使用）
  #define QTRGMODE  0  //  ターゲットモードアクセス
  #define QMSTMODE  1  //  マスタモードアクセス
```

５－５．割り込み関連コード

（１）割り込みマスク

```
  #define QIO0MASK    0x01 //  IODA0 割り込みマスク
  #define QIO1MASK    0x02 //  IODA1     〃
  #define QENDMASK    0x10 //  正常終了  〃
  #define QSTOPMASK   0x20 //  強制終了  〃
  #define QREQMASK    0x40 //  開始要求（REQ）  〃
```

（２）割り込みステータスクリアビット

```
  #define QIO0CLR    0x01 //  IODA0 割り込みステータスクリア
  #define QIO1CLR    0x02 //  IODA1     〃
  #define QENDCLR    0x10 //  正常終了  〃
  #define QSTOPCLR   0x20 //  強制終了  〃
  #define QREQCLR    0x40 //  開始要求（REQ）  〃
```

（３）割り込みステータスビット

```
  #define QIO0STS     0x01 //  IODA0 割り込みステータス
  #define QIO1STS     0x02 //  IODA1     〃
  #define QENDSTS     0x10 //  正常終了  〃
  #define QSTOPSTS    0x20 //  強制終了  〃
  #define QREQSTS     0x40 //  開始要求（REQ）  〃
```

６．ステータス

Gpci3StatusRead 関数で得られるデータは、H／Wステータスです。

ビット

７　マスタビジー　　"0"：停止（終了）、"1"：実行中
　　Gpci3Start 関数でスタート後、"0"になると
　　転送終了。

６　空き（＝０）

５　ＡＣＫ　：ＲＥＱに対する応答（PCI-FPGA が出力）

４　ＲＥＱ　：転送要求　　（開発 FPGA　〃　）

３　ＥＮＢ＿Ｗ　：マスタライト時のイネーブル（開発 FPGA が出力）

２　ＶＬＤ＿Ｗ　：　　〃　　データ有効（PCI-FPGA　〃　）

１　ＥＮＢ＿Ｒ　：マスタリード時のイネーブル（PCI-FPGA　〃　）

０　ＶＬＤ＿Ｒ　：　　〃　　データ有効（開発 FPGA　〃　）

ビット０～５は、“１”でアクティブであり、
マスタモード時ブロック転送コントロール信号である。

7．関数一覧

“ｇｐｃｉ３ｓｂ．ｄｌｌ”でＤＬＬ形式で供給

No.	関数名	概略内容
1	Gpci3Open	ボードのオープン（使用開始）
2	Gpci3Close	ボードのクローズ（使用終了）
3	Gpci3Clock	供給クロック選択
4	Gpci3IODADIR	ポート IODA０～15 の入出力方向
5	Gpci3IODARead	〃　　のリード
6	Gpci3IODAWrite	〃　　のライト
7	Gpci3AccessCnt	マスタ／ターゲットのアクセス選択
8	Gpci3MemAlloc	マスタ転送時のドライババッファの確保
9	Gpci3MemFree	〃　　の解放
10	Gpci3MasterDIR	マスタ転送方向設定
11	Gpci3DataLENG	マスタ転送バイト数の設定
12	Gpci3Start	マスタ転送スタート
13	Gpci3Stop	マスタ転送ストップ（強制終了）
14	Gpci3INTMask	割り込みマスクの設定
15	Gpci3INTClear	割り込みステータス・クリア
16	Gpci3INTRead	〃　　リード
17	Gpci3INTEntry	割り込み登録
18	Gpci3ReadBUFF	ドライバのバッファ・リード
19	Gpci3WriteBUFF	〃　　ライト
20	Gpci3StatusRead	マスタステータス・リード
21	Gpci3ConfigRead	PCI コンフィグレジ・リード
22	Gpci3IORead	PCI-FPGA のレジスタリード（I/O マップ）
23	Gpci3IOWrite	〃　　〃　ライト（　〃　）
24	Gpci3MMRead	開発 FPGA のレジスタリード（メモリマップ）
25	Gpci3MMWrite	〃　　〃　ライト（　〃　）

８．関数説明

関数　1	ボードのオープン
ﾌﾟﾛﾄﾀｲﾌﾟ	HANDLE　Gpci3Open(　BYTE　p1　)
引　数	BYTE　p1：ボード番号 ボードが複数存在しているとき、0～15 を設定する。 １枚目が０で、次のボードから１，2…と番号付けする。 ボードが、１枚のみの場合、この番号は、無視される。
戻り値	HANDLE　：オープンしたボードのハンドル。 このハンドルで、以後の関数を使用する。 エラー時、以下の値を戻す。 INVALID_HANDLE_VALUE （ボードがない場合か、すでにオープンされている場合）
内　容	ボードをオープンし、使用できる状態にする。
備　考	

関数　2	ボードのクローズ
ﾌﾟﾛﾄﾀｲﾌﾟ	int　Gpci3Close(　HANDLE　ph　)
引　数	HANDLE　ph：ボードオープン時の戻り値（ハンドル）
戻り値	int：ステータス ０　　：複数枚使用時、全てボードは、クローズされている。 ０以外：　〃　　、他にオープンされているボードが存在する。 エラーは、ありません。 （オープンされていないボードをクローズにしてもエラーにしません。）
内　容	現在使用しているボードをクローズします。 そして、どれかのアプリケーションからのオープンを待つ。
備　考	複数アプリケーションで、同じボードを使用する場合、同時に 同じボードをオープンできません。 その場合、使用中のアプリケーションは、ボードをクローズして、 解放後、他のアプリケーションでオープンして使用してください。

関数　3	開発 FPGA へのクロック供給の選択
ﾌﾟﾛﾄﾀｲﾌﾟ	void　Gpci3Clock(　HANDLE　ph,　BYTE　p1　)
引　数	HANDLE　ph：ボードオープン時の戻り値（ハンドル） BYTE　p1：供給クロック選択値 （選択値）　QPCICLK　（PCI クロック 33MHz） QEXTCLK　（ボード上の発振器）
戻り値	なし
内　容	開発用 FPGA1、2 に供給するクロックを選択する。 このクロックは、回路信号 OCLK に PCI-FPGA より出力される。
備　考	ボード上の発振器は出荷時搭載されていませんので、発振器の選択は、注意してください。（クロックが供給されません。）

関数　4	IODA０～１５の入出力の選択
ﾌﾟﾛﾄﾀｲﾌﾟ	void　Gpci3IODADIR(　HANDLE　ph,BYTE　p1,BYTE　p2　)
引　数	HANDLE　ph：ボードオープン時の戻り値（ハンドル） BYTE　p1,p2：入出力方向選択値 p1：IODA0～７の入出力 p2：　〃　8～15　〃 （入出力選択値） QDIRIN　：入力方向 QDIROUT：出力方向
戻り値	なし
内　容	IODA0～15 をバイト単位で、入出力方向を設定する。
備　考	IODA0～15 の信号は、開発用 FPGA1,2 及び外部コネクタに接続されている。 また、IODA0,1 は、割り込み信号として利用できる。 その場合、IODA0～７は入力モードにする必要がある。

関数　5	ポート（　IODA０～15　）リード
ﾌﾟﾛﾄﾀｲﾌﾟ	WORD　Gpci3IODARead（　HANDLE　ph　）
引　数	HANDLE　ph：ボードオープン時の戻り値（ハンドル）
戻り値	WORD：リードデータ１６ビット 出力モードになっているポートについては、出力データと同じものがリードできる。
内　容	ポート（IODA0～15）をリードする。
備　考	

関数　6	**ポート（　IODA0～15　）ライト**
ﾌﾟﾛﾄﾀｲﾌﾟ	void　Gpci3IODAWrite(　HANDLE　ph,WORD　p1　)
引　数	HANDLE　ph：**ボードオープン時の戻り値（ハンドル）** WORD　p1：**ライトデータ１６ビット** **ポートが入力モードになっているバイトには、出力されませんが、内部出力ラッチには、データが残ります。** **ライト後、入力を出力モードにした場合、直前のライトデータが出力される。**
戻り値	**なし**
内　容	**ポート（IODA0～15）に、データをライトする。**
備　考	

関数　7	アクセスモードを設定する
ﾌﾟﾛﾄﾀｲﾌﾟ	void　Gpci3AccessCnt(　HANDLE　ph,BYTE　p1　)
引　数	HANDLE　ph：ボードオープン時の戻り値（ハンドル） BYTE　p1：アクセスモード （アクセスモード） QMSTMODE：マスタモード QTRGMODE：ターゲットモード マスタ　　：開発 FPGA1,2 とバーストモードでデータ転送を行う。 ターゲット：Gpci3MMRead、Gpci3MMWrite 関数を使用して 開発 FPGA1,2 のレジスタ（メモリマップ）を アクセスする。
戻り値	なし
内　容	マスタ／ターゲット・モードを選択する。
備　考	マスタアクセスもターゲットアクセスもデータバスを同じ （DAT0～31）バスを使用しているため、同時アクセスはできない。 （注）マスタモード時は、Gpci3MMRead、Gpci3MMWrite 関数は、 使用できません。

関数　8	ドライバのマスタ転送用バッファ取得（パソコンのメモリ）
ﾌﾟﾛﾄﾀｲﾌﾟ	int　Gpci3MemAlloc(　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：取得するバイト数 バイト数は、４の倍数で最大４Mバイト(SRAM 容量分)設定できる。 設定数が、４の倍数でない時、端数は切り捨てられる。
戻り値	int　：物理メモリアドレス NULL の時、取得失敗。
内　容	マスタ転送時、ドライバが使用する物理メモリを取得する。
備　考	取得バイト数は、転送バイト数以上設定してください。 転送バイト数以下だと、違うメモリを壊す恐れがあり、それにより パソコンが、異常動作する事もあります。

関数　9	ドライバのマスタ転送用バッファ解放（パソコンのメモリ）
ﾌﾟﾛﾄﾀｲﾌﾟ	void　Gpci3MemFree(　HANDLE　ph　)
引　数	HANDLE　ph：ボードオープン時の戻り値（ハンドル）
戻り値	なし
内　容	関数 Gpci3MemAlloc で取得したメモリを解放します。
備　考	アプリケーション終了時には、必ずこの関数を実行する必要がある。

関数１０	マスタ転送方向の設定
ﾌﾟﾛﾄﾀｲﾌﾟ	void　Gpci3MasterDIR(　HANDLE　ph,BYTE　p1　)
引　数	HANDLE　ph：ボードオープン時の戻り値（ハンドル） BYTE　p1：転送方向 （転送方向） QDIRIN　：入力 QDIROUT：出力
戻り値	なし
内　容	マスタ転送時の方向を設定する。 出力：PCI-FPGA　→　開発 FPGA1、2 入力：開発 FPGA1、2　→　PCI-FPGA
備　考	マスタ転送中は、設定を変えないこと。

関数１１	マスタ転送バイト数の設定
ﾌﾟﾛﾄﾀｲﾌﾟ	void　Gpci3DataLENG(　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：転送バイト数 （転送バイト数） 設定範囲は、８～４Mバイト。 設定数は、４の倍数で端数は、切り捨てられる。
戻り値	なし
内　容	マスタ転送バイト数で、Gpci3MemAlloc で取得したバイト数 以下にする。
備　考	

関数１２	マスタ転送開始
ﾌﾟﾛﾄﾀｲﾌﾟ	int　Gpci3Start(　HANDLE　ph　)
引　数	HANDLE　ph：ボードオープン時の戻り値（ハンドル）
戻り値	int　：エラーステータス NULL：正常 ERROR_START1：マスタモードになっていない。 ERROR_START2：転送バイト数を設定していない。 ERROR_START3：転送バッファを取得していない。 ERROR_START4：転送バッファより転送バイト数が大きい。
内　容	マスタ転送を開始する。 しかし、開発 FPGA1/2 がスタートしていないと待ち状態となる。
備　考	上記エラー時、 スタートは、無視される。（　ERROR_START1～4　）

関数１３	マスタ転送強制終了
ﾌﾟﾛﾄﾀｲﾌﾟ	void　Gpci3Stop(　HANDLE　ph　)
引　数	HANDLE　ph：ボードオープン時の戻り値（ハンドル）
戻り値	なし
内　容	Gpci3Start でスタート後、この関数を実行すると終了する。 ただし、この終了では PCI-FPGA が出力しているコントロール線を非アクティブにします。 それにより、開発 FPGA1、2 もその信号で終了させないとマスタ転送のフェーズが合わなくなることがあります。 （非アクティブにするコントロール線） ・ nACK ・ nVLD_W／nENB_R
備　考	この関数の実行による PCI-FPGA の強制終了の割り込みは、起こりません。

関数１４	割り込みマスクの設定
ﾌﾟﾛﾄﾀｲﾌﾟ	void　Gpci3INTMask(　HANDLE　ph,BYTE　p1　)
引　数	HANDLE　ph：ボードオープン時の戻り値（ハンドル） BYTE　p1：マスク設定データ （マスクデータ） QENDMASK　：正常終了割り込みマスク（マスタ転送）（0x10） QSTOPMASK ：強制終了　〃　（　〃　）（0x20） QREQMASK　：nREQ　〃　（　〃　）（0x40） QIO0MASK　：IODA0　〃　（0x01） QIO1MASK　：IODA1　〃　（0x02）
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１５	割り込みステータスクリア
ﾌﾟﾛﾄﾀｲﾌﾟ	void　Gpci3INTClear(　HANDLE　ph,BYTE　p1　)
引　数	HANDLE　ph：ボードオープン時の戻り値（ハンドル） BYTE　p1：ステータスクリアデータ （クリアデータ） QENDCLR　：正常終了割り込みステータスクリア　（0x10） QSTOPCLR ：強制終了　〃　（0x20） QREQCLR　：nREQ　〃　（0x40） QIO0CLR　：IODA0　〃　（0x01） QIO1CLR　：IODA1　〃　（0x02）
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。
備　考	

関数１６	割り込みステータスリード
ﾌﾟﾛﾄﾀｲﾌﾟ	BYTE　Gpci3INTRead(　HANDLE　ph　)
引　数	HANDLE　ph：ボードオープン時の戻り値（ハンドル）
戻り値	BYTE：割り込みステータスデータ （ステータスビットデータ） QENDSTS　：正常終了割り込みステータスビットデータ　（0x10） QSTOPSTS ：強制終了　〃　（0x20） QREQSTS　：nREQ　〃　（0x40） QIO0STS　：IODA0　〃　（0x01） QIO1STS　：IODA1　〃　（0x02）
内　容	割り込みステータスデータをリードする。
備　考	

関数１７	割り込みの登録
ﾌﾟﾛﾄﾀｲﾌﾟ	void　Gpci3INTEntry(　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 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	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１８	ドライバのバッファのリード
ﾌﾟﾛﾄﾀｲﾌﾟ	void　Gpci3ReadBUFF(　HANDLE　ph,void　*p1,DWORD　p2, DWORD　p3　)
引　数	HANDLE　ph：ボードオープン時の戻り値（ハンドル） void　*p1：アプリケーションバッファのポインター DWORD　p2：ドライババッファのバイトオフセット（４の倍数） DWORD　p3：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リードする。
備　考	

関数１９	ドライバのバッファのライト
ﾌﾟﾛﾄﾀｲﾌﾟ	void　Gpci3WriteBUFF(　HANDLE　ｐｈ, void　*p1, DWORD　p2, DWORD　p3)
引　数	HANDLE　ph：ボードオープン時の戻り値（ハンドル） void　*p1：アプリケーションバッファのポインター DWORD　p2：ドライババッファのバイトオフセット（４の倍数） DWORD　p3：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へライトする。
備　考	

関数２０	マスタ転送ステータスリード
ﾌﾟﾛﾄﾀｲﾌﾟ	BYTE　Gpci3StatusRead(　HANDLE　ph　)
引　数	HANDLE　ph：ボードオープン時の戻り値（ハンドル）
戻り値	BYTE：ステータスデータ QMSTBUSY　：マスタビジーデータ（0x80） QACK　　：ACK データ　　（0x20） QREQ　　：REQ データ　　（0x10） QENBW　　：ENB_W データ　　（0x08） QVLDW　　：VLD_W　〃　　（0x04） QENBR　　：ENB_R　〃　　（0x02） QVLDR　　：VLD_R　〃　　（0x01） 上記いづれも“１”でアクティブ。
内　容	マスタ転送時のコントロール信号をリードする。 （注）QMSTBUSY は、動作中のステータスで終了すると “０”になる。
備　考	

関数２１	ＰＣＩコンフィグレジスタリード
ﾌﾟﾛﾄﾀｲﾌﾟ	void　Gpci3ConfigRead(HANDLE　ph, struct　GPCI3_CONFIG_DATA　*p1　)
引　数	HANDLE　ph：ボードオープン時の戻り値（ハンドル） GPCI3_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 GPCI3_CONFIG_DATA 構造体内に格納する。
備　考	

関数２２	ＰＣＩ　Ｉ／Ｏリード
ﾌﾟﾛﾄﾀｲﾌﾟ	DWORD　Gpci3IORead(　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：I/O アドレスオフセット 　オフセット値は、DWORD アクセスなので４の倍数で設定。
戻り値	Ｉ／Ｏリードデータ（３２ビット）
内　容	PCI-FPGA のＩ／Ｏレジスタをリードする。
備　考	PCI-FPGA はＩ／Ｏマップレジスタを、開発 FPGA1、2 は メモリマップレジスタを割り当てている。

関数２３	ＰＣＩ　Ｉ／Ｏライト
ﾌﾟﾛﾄﾀｲﾌﾟ	viod Gpci3IOWrite(HANDLE　ph,DWORD　p1,DWORD　p2　)
引　数	HANDLE　ph：ボードオープン時の戻り値（ハンドル） DWORD　p1：I/O アドレスオフセット DWORD　p2：ライトデータ（３２ビット） 　オフセット値は、DWORD アクセスなので４の倍数で設定。
戻り値	なし
内　容	PCI-FPGA のＩ／Ｏレジスタにライトする。
備　考	PCI-FPGA はＩ／Ｏマップレジスタを、開発 FPGA1、2 は メモリマップレジスタを割り当てている。

関数２４	ＰＣＩのメモリリード
ﾌﾟﾛﾄﾀｲﾌﾟ	DWORD　Gpci3MMRead(　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：メモリアドレスオフセット 　オフセット値は、DWORD アクセスなので４の倍数で設定。
戻り値	リードデータ（３２ビット）
内　容	開発ＦＰＧＡ１、２のメモリレジスタからリードする。 （注）ターゲットモードでのみ有効で、 マスタモード時は、不定である。
備　考	PCI-FPGA はＩ／Ｏマップレジスタを、開発 FPGA1、2 は メモリマップレジスタを割り当てている。

関数２５	ＰＣＩのメモリライト
ﾌﾟﾛﾄﾀｲﾌﾟ	void Gpci3MMWrite(HANDLE　ph,DWORD　p1,DWORD　p2)
引　数	HANDLE　ph：ボードオープン時の戻り値（ハンドル） DWORD　p1：メモリアドレスオフセット DWORD　p2：ライトデータ（３２ビット） 　オフセット値は、DWORD アクセスなので４の倍数で設定。
戻り値	なし
内　容	開発 FPGA1、2 のメモリレジスタにデータをライトする。 （注）ターゲットモード時のみ有効で、 　　マスタモード時は、なにもしない。
備　考	PCI-FPGA はＩ／Ｏマップレジスタを、開発 FPGA1、2 は メモリマップレジスタを割り当てている。