GPG-4301

GP-IB インタフェースモジュール用 Linux/RT ドライバ

Help for Linux

www.interface.co.jp

目　次

第1章　はじめに

1.1　概要

本ソフトウェアは、Linux アプリケーションから、弊社製品の制御を行うためのソフトウェアです。GPG-4301 は、弊社 GP-IB 製品を Linux アプリケーションから Lib をリンクし関数をコールすることにより制御します。

本ドキュメントは、Linux 上で GPG-4301 を使用するための情報を掲載しています。

1.2　特長

●SRQ 受信時のイベントおよびコールバック関数の呼び出しをサポートしています。
●ドライバの様々な動作設定(デリミタ、GP-IB アドレス等)をソフトウェアにより動的に変更/取得可能です。
●非コントローラとしての制御も可能です。
●非同期入出力のサポートにより、データ転送終了まで待たずに別の処理を同時に行うことができます。
●1 次アドレスのみだけでなく、2 次アドレスも使用可能です。
●データ送受信時のトーカ、リスナの指定を自動的に行うことができます。
●トーカ、リスナのみならず、デバイストリガ、デバイスクリア、IFC 受信など、多種多様な状態を検出することができます。
●1 枚だけでなく、GPG-4301 の対象デバイスと GPG-4304 の対象デバイスを混在にて最大 16 枚のデバイスを制御することができます。

第2章　製品仕様

2.1　基本仕様

組み込み方式	モジュール（ダイナミックロード/アンロード）
ソースコードの 取り扱い	ドライバモジュール一部公開 ライブラリソースコード非公開 共用モジュール公開
ビルドサポート	メイクファイル提供
Help	PDF 形式
最大デバイス枚数	16 枚
入出力チャンネル数	1 チャンネル
入出力形式	IEEE-488.1 規格(GP-IB)準拠 IEEE-488.2 対応
マイアドレス設定	・1 次アドレスのみ ・1 次アドレス、2 次アドレス 上記、2 つのアドレスモードをソフトウェアにて設定可能
送信/受信デリミタ	・デリミタ無し ・EOI のみ ・CR のみ ・CR+EOI ・LF のみ ・LF+EOI ・CRLF ・CRLF+EOR ・NULL(00h) ・任意の 1 文字 上記 10 種から任意選択可能
データ転送速度	1.1MB/s (使用環境および機器の速度に依存します。)

インタフェース・ファンクション	ファンクション	機能	内容
	C	C1	システム・コントローラ機能
		C2	IFC 送信、コントローラ・イン・チャージ機能
		C3	REN 送信
		C4	SRQ 応答
		C5	インタフェース・メッセージ送信、コントロール受け、コントロール渡し、自分自身へのコントロール渡し、パラレル・ポール、ハンドシェークに同期してコントロール
	SH	SH1	ソース・ハンドシェーク全機能
	AH	AH1	アクセプタ・ハンドシェーク全機能
	T	T6	基本的トーカ、シリアル・ポール、MLA によるトーカ解除
	TE	TE6	基本的拡張トーカ、シリアル・ポール、MSA によるトーカ解除
	L	L4	基本的リスナ、MTA によるリスナ解除
	LE	LE4	基本的拡張リスナ、MSA によるリスナ解除
	SR	SR1	サービス・リクエスト全機能
	RL	RL1	リモート・ローカル全機能
	PP	PP1	リモート・コンフィギュレーションによるパラレル・ポール
		PP2	ローカル・コンフィギュレーションによるパラレル・ポール
	DC	DC1	デバイス・クリア全機能
	DT	DT1	デバイス・トリガ全機能

※本バージョンではホットプラグに未対応です。
CardBus 製品の抜き差しやサスペンドは、本ドライバを取り外した状態で行ってください。

第3章　実行手順

3.1　デバイスを動かすまで

1.　インストール、ドライバモジュールの組み込み

Readme を参照してください。

2.　デバイス番号設定

デバイス番号を設定します。

```
#sh setup.sh
```

デバイス番号設定ユーティリティを起動させると、以下の画面が表示されます。

```
************************************************
 Setup Utility
------------------------------------------------
 Version: 1.30-07
------------------------------------------------
 Copyright 2003, 2005      Interface Corporation.
                             All rights reserved.
************************************************

Enter a model number of the product : GPG/GPH-
```

入力画面では、”4301”を打ち込んでリターンキーを押してください。

```
================================================
Ref.ID | Model      | RSW1 |    CH | Device No.
------------------------------------------------
     1 | PCI-4301   |    0 |       |     1
------------------------------------------------
     2 | PCI-4301   |    1 |       |     2
================================================
```

項目	内容
Ref.ID	そのデバイス番号の ID です。 次のメニューでデバイス番号の変更などを行う際に指定します。
Model	現在システムが認識している GPG-4301 が制御するデバイス一覧です。
RSW1	RSW1 設定値です。
Device No.	デバイス番号です。GPG-4301 では、RSW1 の設定値(CardBus 製品の場合は CardBusID)がデバイス番号として固定されます。

次にメニューを選択します。メニューに示されている番号を入力することでそれぞれの設定変更が可能です。

```
************** Command ***************
  1. Change the device number.
  2. Delete the device number.
  3. Load new device setting file.
  4. Run the initialization program.
  5. Run the CardBus ID setup utility.
 99. Exit the program.
**************************************
Enter the command number:
```

番号	項目	内容
1	デバイス番号の変更	デバイスのデバイス番号を変更します。GPG-4301 ではデバイス番号が固定(ロータリスイッチ(CardBus 製品の場合は CardBusID)設定値となります)であるため、変更することはできません。
2	デバイス番号（ノード）の削除	設定されたデバイス番号を削除します。ID を入力してください。ここでデバイス番号を削除した場合は、再度 dpg0101 を実行するとデバイス番号が設定されます。
3	別のソフト型式のデバイス番号設定	GPG-4301 以外の GPG-XXXX で制御するデバイス型式のデバイス番号設定に移ります。他のカテゴリ（例えば DIO（GPG-2000）など）と併用してシステムを構築している場合に使用してください。
4	初期設定ユーティリティの起動	初期設定プログラムは、ドライバソフトウェアが GP-IB デバイスを制御する際の各種パラメータの設定を行います。詳しくは『6.1 初期値設定プログラム』を参照してください。初期設定を行わない場合は、デフォルト値が使用されます。
5	CardBusID 設定ユーティリティの起動	CardBus 製品のボード ID 表示/設定を行います。『3.3 Card Bus ID設定ユーティリティについて』を参照してください。
99	プログラムの終了	デバイス番号設定ユーティリティを終了します。

※新しくデバイスをシステムに組み込んだ場合は、全てのデバイスに対してデバイス番号設定プログラムでデバイスノードを作成してください。
複数カテゴリのドライバ使用時にドライバのロード順番が前回と変わった場合には、ドライバのロード順番を、デバイス番号設定プログラムを実行した際の順番に戻して下さい。
やむをえず、ドライバのロード順番を変更する場合は、デバイス番号設定プログラムで再設定を行なって下さい。

3. プログラム作成

プログラムの作成を行います。エディタを起動し、下記のコードを記述します。1 次アドレスが 14 の YOKOGAWA7561 をターゲットとしています。「デバイスの初期化」→「IFC 送出」→「REN 信号有効」→「データ送信」→「データ受信」→「使用終了」といった、一連のデバイス制御を行います。
ここでは YOKOGAWA 7561 からの計測データを画面に表示します。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "pcigpib.h"

int  nInpDevAdrsTbl[2] = { 14, -1 };    // 入力機器アドレステーブル
char *szSendData = "F1ROSI100IT1";      // YOKOGAWA 7561 送信データ
char RecvBuf[100];                      // バッファ格納領域

int main(void)
{
    int nRet;                   // 関数戻り値
    unsigned long Len;          // バッファサイズ
```

```
    // デバイス番号 0 のデバイスを初期化
    nRet = PciGpibExInitBoard(0, 0);
    if (nRet) {
        printf("InitBoard ERROR:%d¥n", nRet);
        exit(0);                            // プログラム終了
    }

    // IFC を送出
    nRet = PciGpibExSetIfc(0, 1);
    if (nRet) {
        printf("SetIfc ERROR:%d¥n", nRet);
        exit(0);                            //プログラム終了
    }

    // REN 信号有効
    nRet = PciGpibExSetRen(0);
    if (nRet) {
        printf("SetRen ERROR:%d¥n", nRet);
        exit(0);                            //プログラム終了
    }

    // YOKOGAWA 7561 に対して次のモード設定を行います
    //
    // ・DCV / AUTO RANGE / インターバル 100msec / 積分時間 2.5msec
    //
    printf("YOKOGAWA 7561 : send data = %s¥n", szSendData);
    nRet = PciGpibExSendData(0, nInpDevAdrsTbl, strlen(szSendData), szSendData, 0);
    if (nRet) {
        printf("SendData ERROR:%d¥n", nRet);
        exit(0);                            //プログラム終了
    }

    // YOKOGAWA 7561 から測定データを受信します
    Len = sizeof(RecvBuf);
    nRet = PciGpibExRecvData(0, nInpDevAdrsTbl, &Len, RecvBuf, 0);
    if (nRet) {
        printf("RecvData ERROR:%d¥n", nRet);
        exit(0);                            //プログラム終了
    }

    // 受信文字列の表示
    printf("YOKOGAWA 7561 : receive data = %s¥n", RecvBuf);
    printf("YOKOGAWA 7561 : receive length = %ld¥n", Len);

    // デバイス番号 0 のデバイスを使用終了
    PciGpibExFinishBoard(0);

    return 0;
}
```

コードの記述が終われば gpibtest.c というファイル名で保存しておきます。

4. コンパイル

作成したプログラムをコンパイルします。下記のコマンドを実行してください。

```
#gcc -o gpibtest gpibtest.c -lgpg4301 -lpthread
```

5. 実行

コンパイルすることで実行ファイル gpibtest ができていますので、

```
#./gpibtest
```

と入力し、プログラムを実行してください。
「デバイスの初期化」→「IFC 送出」→「REN 信号有効」→「データ送信」→「データ受信」→「使用終了」といった、一連のデバイス制御を行います。受信したデータは画面上に表示されます。

より詳しいプログラム方法は、『3.2 制御手順』をご参照ください。

3.2　制御手順

3.2.1 コントローラとしての実行

デバイスを複数枚使用する場合は、デバイス上のロータリスイッチ(CardBus 製品の場合は CardBusID)設定値が重複しないように設定してからコンピュータに実装してください。複数のデバイスが存在する場合、デバイスを一意に識別するための番号となります。重複していた場合、本ドライバは正常に動作いたしません。
デバイスをコントローラとして使用する場合の基本的な制御の手順は以下の通りです（記述例は C 言語です)。

1. **デバイスの初期化**

   デバイスをPciGpibExInitBoard関数で初期化します。初期設定プログラムを実行して初期設定を行っている場合は、設定した値が使用されます。初期設定を行っていない場合は、デフォルト値(『2. デバイスの設定』参照)が使用されます。

```
int ret;

// デバイス番号 0 のデバイスを初期化します。
ret = PciGpibExInitBoard(0, 0);
if (ret != NORMAL_EXIT) {// オープンに失敗
    return 1;
}
```

   第 1 引数には、デバイス番号を指定します。デバイス番号は、ロータリスイッチ設定値と同じ値となります。複数のデバイスを使用する際は、デバイス番号が重ならないようにしてください。第 2 引数は予約となっています。0 を指定してください。
   初期化が正常終了後、指定したデバイス番号(この場合は 0)で以後の各関数を使用することができます。
   ※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

2. **デバイスの設定**

   PciGpibExSetConfig 関数で、デバイスのデリミタ等の各種設定を行います。この関数を実行しない場合は、初期値で設定されます。初期値については、初期設定プログラム(cgpibconf)で設定した値が使用されます。初期設定プログラムを実行していない場合は、デフォルト値が使用されます。
   また、この関数は初期化(PciGpibExInitBoard 関数)の実行後は何度でも実行できます。設定例を示します。各パラメータは、スペースで区切ってください。

```
int ret;

// 送信デリミタとして[EOI のみ]、受信デリミタとして[CRLF+EOI]を設定します
ret = PciGpibExSetConfig(0, "/SDELIM=EOI /RDELIM=CRLF+EOI");
if (ret != NORMAL_EXIT) {// 設定に失敗
    return 1;
}
```

PciGpibExSetConfig 関数で設定できる項目とデフォルト値(初期設定プログラムを実行しなかった場合の値)は以下のとおりです。

項目	デフォルト
送受信タイムアウト	100s
STB 応答時間	1s
事象変化検出タイムアウト	100s
送信デリミタ設定	CRLF+EOI
受信デリミタ設定	CRLF+EOI
非同期入出力	同期方式でのデータ送受信
マイアドレス設定	0
2 次アドレス設定	使用しません
IEEE-488 バスハンドシェークタイミング	350ns
パラレルポール実行時間設定	2μs
NRFD 信号ライン待ち設定	無効(データ送信時、NRFD ラインが無効になるのを待ちません)
バスコマンド送信タイムアウト	100s
コントローラ設定	システムコントローラ

※NRFD 信号ライン待ち設定、非同期入出力設定はこの関数でのみ設定ができます(初期設定プログラムでは設定できません)。

各パラメータの設定値を取得する場合は、PciGpibExGetConfig 関数を使用します。設定例を示します。

```
int ret;
unsigned long ulpGetPrm;

// 送受信タイムアウト値を取得します
ret = PciGpibExGetConfig(0, 0, &ulpGetPrm);
if (ret != NORMAL_EXIT) {// 取得に失敗
    return 1;
}
```

3. GP-IB インタフェースの初期化

コントローラとして使用する場合は、PciGpibExSetIfc 関数を使用して GP-IB バスの初期化を行います。続いて、PciGpibExSetRen 関数を使用して REN 信号を有効にします。

```
int ret;

// デバイス番号アクセス番号 0 のデバイスに対して、100μs 単位で IFC 送出時間を設定します
ret = PciGpibExSetIfc(0, 1);
if (ret != NORMAL_EXIT) {// IFC 送出に失敗
    return 1;
}

// REN 信号を送出します
ret = PciGpibExSetRen(0);
if (ret != NORMAL_EXIT) {// REN 信号の送出に失敗
    return 1;
}
```

4. データ送信

機器固有の制御コマンドは GP-IB バスのデータとして送信します。
PciGpibExSendData 関数で指定した機器に対してデータを送信します。コントローラとして本関数を実行する場合は、コントローラ・イン・チャージの状態にする必要があります。本関数使用前にあらかじめ、PciGpibExSetIfc 関数を実行し、デバイスをコントローラ・イン・チャージ状態にしてください。
同期入出力時には、タイムアウト発生もしくは何らかの要因により処理が終了した場合において、本関数の処理を終了します。
非同期入出力時にタイムアウトが発生すると、非同期入出力中のエラーとなります。

```
int ret;
int nAdrsTbl[2];
     ・
     ・
nAdrsTbl[0] = 1;     // 制御する測定機器のアドレス
nAdrsTbl[1] = -1;    // アドレステーブルの終端
                     // 機器アドレステーブルの終端には必ず -1 を代入してください
     ・
     ・
// アドレス 1 の機器に対して、”*RST”という 4 バイトのデータを送信します
ret = PciGpibExSendData(0, nAdrsTbl, 4, ”*RST”, 0);
if (ret != NORMAL_EXIT) {// データ送信に失敗
    return 1;
}
```

PciGpibExSendData 関数の第 1 引数には、デバイス番号を指定します。
第 2 引数には、データ送信先の機器の機器アドレスを格納しているアドレステーブルを指定します。
第 3 引数には、送信データ長をバイト単位で指定します（デリミタ長をあわせたデータ長 1Mbyte(1048576byte)までのデータ送信が可能です)。
第 4 引数には、送信データへのポインタを指定します。
第 5 引数には同期転送時には NULL を指定します。非同期転送の場合にはデータ送信完了時にこの引数に設定したコールバック関数がデリミタ送出後に呼び出されます。デリミタは関数内で自動的に付加されるので、あらかじめデータに付加しておく必要はありません。

●データ送信時の機器アドレスについて

機器アドレステーブルには、1 次、2 次アドレスを混在して複数台の機器を指定することができます。コントローラ時、機器の指定が必要な関数にて機器アドレステーブルを引数とする場合、以下のように int 型(整数型)の配列に機器のアドレスを設定して使用します。機器アドレスを指定する場合は、終端として必ず-1 を格納してください。
1 次アドレスのみを使用する機器 1 台だけを指定する場合は、下記のようになります。

配列の Index	格納する値
0	1 次アドレス
1	終端 (-1)

データ送信先の機器の 1 次アドレスが 2 の場合の指定例は、下記のようにプログラムを記述します。

```
int nAdrs[2];

nAdrs[0] = 2;     // 機器アドレス
nAdrs[1] = -1;    // 終端
```

2 次アドレスを使用する機器 1 台だけを指定する場合は、下記のようになります。

配列の Index	格納する値
0	1 次アドレス
1	2 次アドレス
2	終端 (−1)

データ送信先の 1 台目の機器の 1 次アドレスが 2、2 台目の機器の 1 次アドレスが 7 の場合は、下記のようにプログラムを記述します。

```
int nAdrs[3];

nAdrs[0] = 2;   // 1 次アドレス
nAdrs[1] = 96;  // 2 次アドレス
nAdrs[2] = -1;  // 終端
```

1 次アドレスのみの機器を複数台指定する場合は、下記のようになります。

配列 の Index	格納する値
0	1 次アドレス
1	1 次アドレス
・・・	・・・
n-1	1 次アドレス
n	終端 (−1)

データ送信先の機器の 1 次アドレスが、1 台目 2、2 台目 7 の場合は、下記のようにプログラムを記述します。

```
int nAdrs[3];

nAdrs[0] = 2;    // 1 台目の機器の 1 次アドレス
nAdrs[1] = 7;    // 2 台目の機器の 1 次アドレス
nAdrs[2] = -1;   // 終端
```

本ドライバが送信したデータは 2 台の機器とも同じデータを同時に受信します。

2 次アドレスを使用する機器を複数台指定する場合は、下記のようになります。

配列 Index	格納値
0	1 次アドレス
1	2 次アドレス
2	1 次アドレス
3	2 次アドレス
・・・	・・・
n−2	1 次アドレス
n−1	2 次アドレス
n	終端(-1)

データ送信先の 1 台目の機器の 1 次アドレスが 2、2 次アドレスが 96 で、2 台目の機器の 1 次アドレスが 7、2 次アドレス 97 の場合は、下記のようになります。

```
int nAdrs[5];

nAdrs[0] = 2;   // 1 台目の機器の 1 次アドレス
nAdrs[1] = 96;  // 1 台目の機器の 2 次アドレス
nAdrs[2] = 7;   // 2 台目の機器の 1 次アドレス
nAdrs[3] = 97;  // 2 台目の機器の 2 次アドレス
nAdrs[4] = -1;  // 終端
```

本ドライバが送信したデータは 2 台の機器とも同じデータを同時に受信します。

5. データ受信

機器から送られてくるデータは PciGpibExRecvData 関数にて受信します。
PciGpibExRecvData 関数を実行することで受信データおよび受信データ長を取得することができます。コントローラとして本関数を実行する場合は、コントローラ・イン・チャージの状態にする必要があります。本関数使用前にあらかじめ、PciGpibExSetIfc 関数を実行し、デバイスをコントローラ・イン・チャージ状態にしてください。
同期入出力時には、タイムアウト発生もしくは何らかの要因により処理が終了した場合において、本関数の処理を終了します。
非同期入出力時にタイムアウトが発生すると、非同期入出力中のエラーとなります。

```
int ret;
char  szData[1024];
int   nAdrsTbl[2] = {2, -1};  // 機器アドレステーブル
unsigned long ulLen;  // 受信バッファ長
     ・
     ・
ulLen = 1024;       // 受信バッファのサイズ
ret = PciGpibExRecvData(0, nAdrsTbl, &ulLen, szData, 0);
if (ret != NORMAL_EXIT) {// 受信失敗時
    return 1;
}
```

PciGpibExRecvData 関数の第 1 引数には、デバイス番号を指定します。
第 2 引数には、データを送信してくる機器の機器アドレスを格納しているアドレステーブルを指定します。
第 3 引数には受信バッファ長をバイト単位で指定します。関数呼出し前に受信バッファ長にはあらかじめ受信バッファの長さを格納しておく必要があります。受信バッファ長のサイズはデリミタサイズも含めて充分余裕を持って指定してください(デリミタ長をあわせたデータ長 1Mbyte(1048576byte)までのデータが受信可能です)。これは機器から送られてくるデータ長が 10 バイトと判明しており、デリミタに CRLF を指定している場合、受信バッファサイズは最低 12 バイト必要であることを示します。関数の実行終了後、実際に受信したデータ長が格納されます。
第 4 引数には受信データを格納する変数へのポインタを指定します。
第 5 引数には同期転送時には NULL を指定します。非同期転送の場合にはデータ受信完了時(デリミタ受信後)にこの引数に設定したコールバックが呼び出されます。

●データ受信時の機器アドレスについて

1 台の機器からデータを受信する場合は、データ送信時のアドレス指定と同じ使い方になりますが、複数台の機器を指定する場合は、テーブルの先頭に指定した機器からのデータを、テーブルの次以降に指定した機器も受信します。1 次アドレスのみの機器を複数指定する場合の例を以下に示します。

```
int nAdrs[4];

nAdrs[0] = 2;  // 1 台目の機器の 1 次アドレス
               // この機器からのデータを受信します
nAdrs[1] = 7;  // 2 台目の機器の 1 次アドレス
               // 配列の先頭で指定したアドレス 2 の機器からのデータを受信します
nAdrs[2] = 9;  // 3 台目の機器の 1 次アドレス
               // 配列の先頭で指定したアドレス 2 の機器からのデータを受信します
nAdrs[3] = -1;  // 終端
```

上記の例では、アドレス 2 の機器がトーカになり、その他の機器(GP-IB デバイスを含む)がリスナとなってデータを受信します。

6. シリアルポーリング

シリアルポーリングは、SRQ が検出された場合に行います。SRQ を検出する方法は、以下の 2 つがあります。

●PciGpibExCheckSrq 関数で SRQ 信号が有効かどうかを確認する

PciGpibExCheckSrq 関数により、機器からの SRQ 信号が有効であるかを確認することができます。この関数を実行し、機器からの SRQ 信号が有効であれば、機器からのステータスバイト(STB)を受信するために、シリアルポーリング(PciGpibExExecSpoll 関数)を実行します。

```
int ret;
int npAdrsTbl[2]  = { 1, -1 };
int npStbTbl[2], npStbAdrs[2];

ret = PciGpibExCheckSrq( 0 );
if (ret == ACTIVE_SRQ) {// SRQ が検出された場合
    PciGpibExExecSpoll(0, npAdrsTbl, npStbTbl, npStbAdrs);
}
```

PciGpibExExecSpoll 関数の第 1 引数には、デバイス番号を指定します。
第 2 引数には、シリアルポールの対象の機器の機器アドレスを格納している機器アドレステーブルを指定します。送信時、受信時と同じように、アドレステーブルの終端には必ず-1 を格納してください。
第 3 引数には、シリアルポーリングを行い、受信した各機器の有効なステータス・バイトを格納するテーブルへのポインタを指定します。関数呼び出し後、終端には-1 が付加されます。
第 4 引数には、SRQ を送出し、有効なステータス・バイトを持っていた各機器のアドレスを格納するテーブルへのポインタを指定します。関数呼び出し後、終端には-1 が付加されます。
GP-IB 機器アドレスが、それぞれ 1、2、3 の機器が接続されているとし、アドレス 2 の機器から SRQ が発行された場合の処理は以下のようになります。

```
int ret;
int adrstbl[4] = {1, 2, 3, -1}; // 機器アドレスは1、2、3の3つを指定します
                                // 終端は必ず-1を指定してください
int stbtbl[4];  // 関数終了後にステータスバイトを格納する配列です
int stbadrs[4]; // 関数終了後に有効なステータスバイトを持っていた各機器のアドレスを
                // 格納する配列です

// アドレス1、2、3の機器に対してシリアルポーリングを実行します
ret = PciGpibExExecSpoll(0, adrstbl, stbtbl, stbadrs);
if (!ret) {// 関数正常終了時
    // アドレス2の機器からSRQが発行されていますので配列の要素は
    // stbtbl[0] = アドレス2の機器のステータスバイト
    // stbtbl[1] = -1 (終端)
    // stbadrs[0] = 2 (SRQを発行した機器のアドレス)
    // stbadrs[1] = -1 (終端)
    // となります
}
```

●SRQ コールバックイベントを使用する

PciGpibExSetSrqEvent 関数にて、SRQ 検出時に呼び出されるコールバック関数を登録し、PciGpibExWaitSrq 関数にて、SRQ が検出されるのを待ちます。

```
// コールバックルーチン
void SrqProc(int nBoardNo, unsigned long ulUser)
{
    int npAdrsTbl[2] = {1, -1};
    int npStbTbl[2], npStbAdrs[2];

    PciGpibExExecSpoll(0, npAdrsTbl, npStbTbl, npStbAdrs);
}

int main()
{
    int ret;

    ret = PciGpibExSetSrqEvent(0, SrqProc, 1);
    if (ret != NORMAL_EXIT) {// SRQイベントの登録に失敗
        return 1;
    }

    ret = PciGpibExWaitSrqEvent(0, 5000);
    if (ret != NORMAL_EXIT && ret != ERR_EVENT_WAIT_TIMEOUT) {// イベント待ちに失敗
        return 1;
    }
    ...
}
```

PciGpibExWaitSrqEvent 関数の第 2 引数で SRQ 受信を検出するまでの待ち時間を ms 単位で指定します。タイムアウト時間が経過すると、関数は SRQ を受信していない場合でも、制御を戻します。タイムアウト時間に 0 を指定した場合は、SRQ 受信の有無の状態を調べてから、すぐに制御を戻します。タイムアウトまでに SRQ が検出され、コールバック関数が呼ばれた場合、コールバック関数内でシリアルポーリングを行います。

7. 終了処理

デバイスを PciGpibExFinishBoard 関数で使用終了します。必ず本関数を実行して処理を終了してください。

```
int ret;

// デバイス番号 0 のデバイスでの処理を終了します。
ret = PciGpibExFinishBoard(0);
if (ret != NORMAL_EXIT) {// クローズに失敗
    return 1;
}
```

3.2.2 非コントローラとしての実行手順

デバイスを非コントローラとして使用する場合の基本的な制御の手順は以下の通りです。
（記述例はC言語です）

1. **デバイスの初期化**
   コントローラとしての実行手順と同じです。

2. **デバイスの設定**
   コントローラとしての実行手順と同じです。ただし、コントローラ設定には、非コントローラを指定してください。

```
int ret;

// 非コントローラとして使用します
ret = PciGpibExSetConfig(0, "/CONTROLLER=OFF");
if (ret != NORMAL_EXIT) {// 設定に失敗
    return 1;
}
```

3. **バス・ステータス確認**
   非コントローラで使用する場合、データの送信/受信を行うためにはデバイスがトーカ/リスナの状態になる必要があります。PciGpibExGetStatus 関数にてデバイスがどのような状態かを取得することができます。
   トーカの場合はデータの送信、リスナの場合は、データの受信ができます。

```
int ret;
unsigned int Status;
char ReceiveBuffer[100];
unsigned long Len = sizeof(ReceiveBuffer);

ret = PciGpibExGetStatus(0, &Status);
if (ret == NORMAL_EXIT) {// バス・ステータス正常取得
    if (Status & 0x01000000) {  // トーカ指定検出
        // トーカ指定済ならばデータ送信を行う
        PciGpibExSendData(0, NULL, 10, "0123456789", 0);
    }
    else if (Status & 0x02000000) {  // リスナ指定検出
        // リスナ指定済ならばデータ受信を行う
        PciGpibExRecvData(0, NULL, &Len, ReceiveBuffer, 0);
    }
}
```

4. データ送信

デバイスがトーカ指定されている場合には、GP-IB バスへデータ送信を実行することができます。PciGpibExSendData 関数でデータを送信します。基本的にはコントローラ時と同じですが、非コントローラ時は、第 2 引数のアドレステーブルに NULL を指定します。

```
int ret;
unsigned int Status;

ret = PciGpibExGetStatus(0, &Status);
if (ret == NORMAL_EXIT) {// バス・ステータス正常取得
    if (Status & 0x01000000) {  // トーカ指定検出
        // トーカ指定済ならばデータ送信を行う
        PciGpibExSendData(0, NULL, 10, "0123456789", 0);
    }
}
```

5. データ受信

デバイスがリスナ指定されている場合には、GP-IB バスからのデータ受信を実行することができます。PciGpibExRecvData 関数を実行することで受信データおよび受信データ長を取得することができます。基本的にはコントローラ時と同じですが、非コントローラ時は、第 2 引数のアドレステーブルに NULL を指定します。

```
int ret;
unsigned int Status;
char ReceiveBuffer[100];
unsigned long Len = sizeof(ReceiveBuffer);

ret = PciGpibExGetStatus(0, &Status);
if (ret == NORMAL_EXIT) {// バス・ステータス正常取得
    if (Status & 0x02000000) {  // リスナ指定検出
        // リスナ指定済ならばデータ受信を行う
        PciGpibExRecvData(0, NULL, &Len, ReceiveBuffer, 0);
    }
}
```

6. SRQ の発行

非コントローラの場合、PciGpibExSetSrq 関数により、ステータスバイトを設定し、SRQ 信号を送出することができます。また、PciGpibExCheckStb 関数により、コントローラにシリアルポールをされたか確認することができます。

```
int ret;

ret = PciGpibExSetSrq(0, 0x05);
if (ret != NORMAL_EXIT) {// SRQ の送出に失敗
    return 1;
}
          .
          .
ret = PciGpibExCheckStb(0);
if (ret == OK_SEND_STB){// ステータスバイトは通知済み
    ...
}
else if (ret == NOT_EXEC_SPOLL) {// まだシリアルポールが行われていない
    ...
}
```

SRQ 送出時のステータスバイトは、ビット 6 が 1 となります(2 進数で 01000000)ので、上記の例では、実際に送出されるステータスバイトは、0x05 とビット 6 を OR した値の 0x45 となります。

7. パラレルポール・コンフィギュレーション

パラレルポールが実行された場合に、DIO ラインのどのビットを使用して応答するのかをあらかじめ設定する必要があります。パラレルポールが実行された時の、ist の値(PciGpibSetIst の第 2 引数で指定した値)と、リモート/ローカル・コンフィギュレーションで指定されている Sense bit が等しいときに、指定している DIO ラインをアサートします。
PciGpibExSetIst 関数によりパラレルポールの応答極性を指定します。
PciGpibExSetPp2 関数によりパラレルポールの応答指定、応答極性、応答に使用する DIO ラインを指定します(ローカル・コンフィギュレーション)。

```
int ret;

// パラレルポールに対して肯定応答します
ret = PciGpibExSetIst(0, 1);
if (ret != NORMAL_EXIT) {// パラレルポール応答極性の設定に失敗
    return 1;
}

// パラレルポールに対して肯定応答に設定されている場合に
// DIO ライン 1 が ON になるように設定します
ret = PciGpibExSetPp2(0, 0x08);
if (ret != NORMAL_EXIT) {// パラレルポール応答条件の設定に失敗
    return 1;
}
```

例えば、PciGpibExSetIst 関数の第 2 引数に 1 をセットした場合、PciGpibExSetPp2 関数の第 2 引数のビット 3 が 1 にセットされていれば、パラレルポールが実行された場合に、PciGpibExSetPp2 関数の第 2 引数のビット 0～2 で指定される DIO ラインをアサートします。上記の例では、DIO ライン 1 をアサートします。

8. 終了処理

デバイスを PciGpibExFinishBoard 関数で使用終了します。必ず本関数を実行して処理を終了してください。コントローラとしての実行手順と同じです。

3.3 Card Bus ID 設定ユーティリティについて

Card Bus ID 設定ユーティリティは、複数枚の同一型式 Card Bus 製品を使用する為のものです。Card Bus 製品毎に異なる ID 番号を設定します。

【操作方法】

下記のコマンドを実行します。

```
#/usr/bin/dpg0101 -c
```

実行すると、現在挿入されている弊社 CardBus 製品の情報が表示されます。

```
============================================================
 Ref.ID | Bus | Dev | Func| Model                      | RSW1
------------------------------------------------------------
      1 |  2  |  0  |  0  | CBI-4302                   |   0
============================================================
```

項目	内容
Ref.ID	CardBus 製品のインデックス番号です。 メニューで CardBus 製品の選択を行う際に指定します。
Bus	CardBus 製品が挿入されているバス番号を示します。
Dev	CardBus 製品が挿入されているデバイス番号を示します。
Func	CardBus 製品が挿入されているデバイス番号を示します。
Model	CardBus 製品の型式を示します。
RSW1	設定されている ID 番号を示します。

ID 番号を変更したい場合はメニューから「1」を選択し、変更を行いたい CardBus 製品のインデックス番号を入力します。

```
************** Command ******************
  1. Change the board id number.
  2. Run the device number setup utility.
 99. Exit the program.
*****************************************
Enter the command number:1
```

次に CardBus ID 番号を入力しリターンキーを押します。CardBus ID 番号は 0～15 の値を入力してください。

```
Enter Ref.ID:
Enter the board id number (0-15)
If you want to cancel this operation, enter -1.
```

「99」を選択することでユーティリティを終了します。

※変更した ID 番号をシステムに認識させるには、ドライバの再起動が必要です。
※設定した ID 番号がわかるように番号を記したシールを Card Bus 製品に貼ることをお勧めします。

第4章　リファレンス

4.1　関数一覧

No	関数名	機能
1	PciGpibExInitBoard	指定されたデバイスを初期化します。
2	PciGpibExFinishBoard	指定されたデバイスの使用を終了します。
3	PciGpibExGetInfo	指定されたデバイスの情報を取得します。
4	PciGpibExSetConfig	ライブラリの動作パラメータを設定/変更します。
5	PciGpibExGetConfig	ライブラリの動作パラメータを取得します。
6	PciGpibExGetStatus	現在のバス・ステータスを取得します。
7	PciGpibExClrStatus	現在のバス・ステータスのクリアを行います。
8	PciGpibExGetBusLine	GP-IB バスラインの状態を取得します。
9	PciGpibExSendData	バスにデータの送信を行います。
10	PciGpibExRecvData	バスからデータの受信を行います。
11	PciGpibExCheckStb	コントローラからシリアルポーリングされたかを確認します。
12	PciGpibExSetSrq	SRQ の送出(サービス要求)を行います。
13	PciGpibExSetIst	パラレル・ポール応答フラグを設定します。
14	PciGpibExSetPp2	パラレル・ポール応答モード(pp2)を設定します。
15	PciGpibExSetIfc	指定された時間、IFC の送出を行います。
16	PciGpibExSetRen	REN ラインをセットします。
17	PciGpibExResetRen	REN ラインをリセットします。
18	PciGpibExSetRemote	指定された機器をリモートモードに設定します。
19	PciGpibExSetRwls	指定された機器をリモート・ロックアウト状態に設定します。
20	PciGpibExReSysCtrl	システムコントローラの要求または解除を行います。
21	PciGpibExExecTrigger	指定された機器に対してトリガを行います。
22	PciGpibExExecDevClear	全ての機器に対してデバイスクリアを行います。
23	PciGpibExExecSdc	指定された機器に対してデバイスクリアを行います。
24	PciGpibExSetLocal	指定された機器をローカルモードに設定します。
25	PciGpibExSetLlo	全ての機器をローカル・ロックアウト状態に設定します。
26	PciGpibExExecPassCtrl	指定された機器に対してパス・コントロールを行います。
27	PciGpibExExecFindListener	バス上に接続されているリスナを探します。
28	PciGpibExExecDevReset	指定された機器を完全にリセットします。
29	PciGpibExGoStandby	コマンドモードからデータモードへ遷移させます。
30	PciGpibExGoActCtrller	データモードからコマンドモードへ遷移させます。
31	PciGpibExExecSpoll	指定された機器をシリアルポーリングします。
32	PciGpibExCheckSrq	SRQ 受信の有効/無効を確認します。
33	PciGpibExClearSrq	SRQ 受信フラグのクリアを行います。
34	PciGpibExEnableSrq	SRQ 受信の許可を行います。
35	PciGpibExDisableSrq	SRQ 受信の禁止を行います。
36	PciGpibExExecPpoll	パラレルポーリングを行います。
37	PciGpibExCfgPpoll	指定された機器に対してパラレルポール応答条件を設定します。
38	PciGpibExUnCfgPpoll	パラレル・ポール応答条件を解除します。
39	PciGpibExWriteBusCmd	バスコマンドの送出を行います。
40	PciGpibExSendFile	ファイルから読み込んだデータをバスに送信します。
41	PciGpibExRecvFile	バスから受信したデータをファイルに書き込みます。
42	PciGpibExSetSignal	事象変化検出条件の設定を行います。
43	PciGpibExWaitSignal	事象変化検出を待ちます。
44	PciGpibExSetSrqEvent	指定番号のデバイスからの SRQ コールバック関数を登録します。
45	PciGpibExWaitSrqEvent	指定番号のデバイスからの SRQ コールバック関数を待ちます。
46	PciGpibExKillSrqEvent	指定番号のデバイスからの SRQ コールバック関数を解除します。
47	pOnSrqEvent	SRQ イベントが発生した時にコールされるコールバック関数です。

4.2　関数個別説明

1.PciGpibExInitBoard

指定されたデバイスを初期化します。

```
int PciGpibExInitBoard(
  int   nBoardNo,   // デバイス番号
  int   nReserved   // 予約
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
「PCI-432101」の場合は「0」固定となります。
デバイス番号が不明の場合は、/proc ファイルシステムを参照する事により確認します。

#cat /proc/driver/gpib/cp4301
cp4301 info:1.0
0: PCI/LPC-432101(bid=0)　bid=0 がデバイス番号です。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

nReserved

将来拡張用です。0 を設定してください。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_DRV_RETURN
・ERR_NO_BOARD

備考

初期設定プログラムを実行している場合は、初期設定値でデバイスが初期化されますが、初期設定プログラムを実行していない場合は、デフォルト値が使用されます。

使用例

```
int ret;

ret = PciGpibExInitBoard(0, 0);
```

デバイス番号 0 のデバイスを初期化します。

参照

3.2 制御手順
3.2.1 コントローラとしての実行『1. デバイスの初期化』
3.2.2 非コントローラとしての実行『1. デバイスの初期化』

2. PciGpibExFinishBoard

指定されたデバイスの使用を終了します。

```
int PciGpibExFinishBoard(
    int    nBoardNo    // デバイス番号
);
```

パラメータ

nBoardNo

使用を終了するデバイスのデバイス番号を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_DRV_RETURN

使用例

```
int ret;
int bno = 0;

ret = PciGpibExInitBoard(bno, 0);
if (ret) {
    printf("ERROR:%d¥n", ret);
    exit(1);
}
...
PciGpibExFinishBoard(bno);
```

デバイス番号 0 のデバイスの使用を終了します。

参照

3.2 制御手順
3.2.1 コントローラとしての実行『7. 終了処理』
3.2.2 非コントローラとしての実行『8. 終了処理』

3.PciGpibExGetInfo

指定デバイスの情報を取得します。

```
int PciGpibExGetInfo(
  int             nBoardNo,   // デバイス番号
  int             nPrmNo,     // デバイス情報を取得するためのパラメータ番号
  unsigned long*  ulpGetPrm   // デバイス情報を格納する変数へのポインタ
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

nPrmNo

デバイス情報を取得するためのパラメータ番号を指定します。
取得結果は ulpGetPrm が指す領域に格納されます。

値	項目	内容
0	レジスタベースアドレス取得	デバイスのレジスタベースアドレスを取得します。
1	割り込み番号取得	デバイスの使用割り込み番号を取得します。
2	動作モード取得	デバイスの初期化時の動作モードを取得します。 0:コントローラ 1:非コントローラ
3	デバイスタイプ	デバイスの型式番号を取得します。 4301:PCI-4301 4302:PCI-4302、CPZ-4302、CPZ-4302P、CTP-4302、CTP-4302P、CBI-4302 4321:CBI-432101、CBI-432101WA, CSI-432101、LPC-432101、PEX-432101、PEX-H432101P
4	RSW1 設定値	デバイスの RSW1 設定値を取得します。
5	サブシステム ID	デバイスのサブシステム ID を取得します。 1h:PCI 製品 101h:CompactPCI 製品 201h:CBI-4302 2881h:CBI-432101、CBI-432101WA, CSI-432101 2081h:LPC-432101 2C81h:PEX-432101、PEX-H432101P

ulpGetPrm

nPrmNo 変数で指定した各種情報を格納する変数へのポインタを指定します。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_INP_PARAM
・ERR_DRV_RETURN

使用例

```
int ret;
unsigned long ulpGetPrm;

ret = PciGpibExGetInfo(0, 2, &ulpGetPrm);
if (!ret) {
    if (!ulpGetPrm) printf("SystemController¥n");
    else            printf("Not SystemController¥n");
}
```

デバイス番号 0 のデバイスのデバイス情報を取得します。

4. PciGpibExSetConfig

ライブラリの動作パラメータを設定/変更します。

```
int PciGpibExSetConfig(
  int    nBoardNo,   // デバイス番号
  char*  p           // 設定情報文字列へのポインタ
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

p

設定情報文字列を指定します。
指定デバイス番号のデバイスに対する設定に関する各種情報を含んだ文字列を指定してください。指定のなかった項目に対しては、初期値が設定されます。
設定情報の各パラメータを以下に示します。各パラメータは、スペースで区切ってください。
各パラメータ文字列の中にはスペースを入れないようにして下さい。

指定文字列	項目	設定可能値	内容
“/TMO=”	送受信 タイムアウト	“1”～”65535”	送受信時に使用するタイムアウト時間を 100ms 単位で設定します。デフォルトは初期設定プログラムでの設定値が使用されます。
“/SRT=”	STB 応答時間	“1”～”65535”	シリアルポール時のステータスバイト応答時間を 100ms 単位で設定します。デフォルトは初期設定プログラムでの設定値が使用されます。
“/STM=”	事象変化検出 タイムアウト	“0”～”65535”	事象変化検出(PciGpibExSetSignal 関数、PciGpibExWaitSignal 関数)でのタイムアウト時間を 100ms 単位で設定します。0 を指定すると、タイムアウトは発生しません。デフォルトは、初期設定プログラムでの設定値が使用されます。
“/SDELIM=”	送信デリミタ設定	“NO” “EOI” “CR” “CR+EOI” “LF” “LF+EOI” “CRLF” “CRLF+EOI” “NULL” “!x”	送信時に使用するデリミタを設定します。 デフォルトは、初期設定プログラムでの設定値が使用されます。 “NO” : デリミタなし “EOI” : EOI “CR” : CR “CR+EOI” : CR+EOI “LF” : LF “LF+EOI” : LF+EOI “CRLF” : CRLF “CRLF+EOI” : CRLF+EOI “NULL” : NULL “!x” : 感嘆符に続く 1 文字がデリミタ

“/RDELIM=”	受信デリミタ設定	“NO” “EOI” “CR” “CR+EOI” “LF” “LF+EOI” “CRLF” “CRLF+EOI” “NULL” “!x”	受信時に使用するデリミタを設定します。 デフォルトは、初期設定プログラムでの設定値が使用されます。 “NO” : デリミタなし “EOI” : EOI “CR” : CR “CR+EOI” : CR+EOI “LF” : LF “LF+EOI” : LF+EOI “CRLF” : CRLF “CRLF+EOI” : CRLF+EOI “NULL” : NULL “!x” : 感嘆符に続く 1 文字がデリミタ
“/ASYNC=”	非同期入出力設定	“ON” “OFF”	“ON” : データの送受信において、非同期による入出力を行います。 “OFF” : データの送受信において、同期しての入出力を行います。(デフォルト)
“/MA=”	マイアドレス設定	“0”～”30”	デバイス自身のマイアドレス(1 次アドレス)を設定/変更します。デフォルトは、初期設定プログラムでの設定値が使用されます。
“/SA=”	2 次アドレス設定	“96”～”126”	2 次アドレス可能化/不可能化を設定します。デフォルトは、初期設定プログラムでの設定値が使用されます。 “96”～”126”以外の値を設定すると、2 次アドレスを使用しません。
“/HSTMG=”	IEEE-488 バスハンドシェークタイミング設定	“0”～”2”	IEEE-488 バスハンドシェークタイミング T1 タイミングの設定を行います。デフォルトは、初期設定プログラムでの設定値が使用されます。 “0” : T1 に正常タイミング(2μs) “1” : T1 に高速タイミング(500ns) “2” : T1 に超高速タイミング(350ns)
“/PPETM=”	パラレルポール実行時間設定	“0” “1”	パラレルポール実行時間 T6 の設定を行います。デフォルトは、初期設定プログラムでの設定値が使用されます。 “0” : T6 を 2μs(デフォルト) “1” : T6 を 10μs
“/NRFDWAIT=”	NRFD 信号ライン待ち有効/無効設定	“ON” “OFF”	マスタモードデータ送信の最終デリミタ送信後に GP-IB バスの NRFD 信号ラインが無効になるまで待つか待たないかの設定を行います。 待ち時間には事象変化検出タイムアウトの時間が適用されます。 “ON” : NRFD 信号ラインが無効になるまで待ちます。 “OFF” : NRFD 信号ラインの待ちは無効となります。(デフォルト)

“/CMDTMO=”	バスコマンド送信タイムアウト	“1”～”65535”	GP-IB バスコマンド送信時に使用するタイムアウト時間を 100ms 単位で設定します。デフォルトは、初期設定プログラムでの設定値が使用されます。
“/CONTROLLER=”	コントローラ設定	“ON” “OFF”	デバイス初期化直後のみ有効なオプション文字列です。 デバイスをコントローラとして使用するか、非コントローラとして使用するかを設定します。デフォルトは、初期設定プログラムでの設定値が使用されます。 “ON” ： コントローラとして使用します。 “OFF” ： 非コントローラとして使用します。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

- NORMAL_EXIT
- ERR_BRD_NO
- ERR_INP_PARAM
- ERR_NOT_SET
- ERR_DRV_RETURN

備考

各種タイムアウト時間には 0 を指定しないようにして下さい。

使用例

```
int ret;

ret = PciGpibExSetConfig(0, ”/SDELIM=CRLF /RDELIM=EOI”);
```

デバイス番号 0 のデバイスを送信デリミタ CRLF、受信デリミタ EOI として、設定します。

参照

3.2.制御手順
3.2.1. コントローラとしての実行手順『2. デバイスの設定』

5.PciGpibExGetConfig

ライブラリの動作パラメータを取得します。

```
int PciGpibExGetConfig(
  int             nBoardNo,    // デバイス番号
  int             nPrmNo,      // 動作パラメータを取得するためのパラメータ番号
  unsigned long*  ulpGetPrm    // 動作パラメータを格納する変数へのポインタ
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

nPrmNo

動作パラメータを取得するためのパラメータ番号を指定します。

値	項目	内容
0	送受信タイムアウト取得	送受信時に使用するタイムアウト時間を 100ms 単位で取得します。
1	STB 応答時間取得	シリアルポール時のステータスバイトの応答時間を 100ms 単位で得ます。
2	事象変化検出タイムアウト取得	事象変化検出時に使用するタイムアウト時間を 100ms 単位で得ます。
3	送信デリミタ設定取得	送信時に使用するデリミタを取得します。 0 ： デリミタなし 1 ： EOI 2 ： CR 3 ： CR+EOI 4 ： LF 5 ： LF+EOI 6 ： CRLF 7 ： CRLF+EOI 8 ： NULL 9 ： 任意の文字
4	受信デリミタ設定取得	受信時に使用するデリミタを取得します。 0 ： デリミタなし 1 ： EOI 2 ： CR 3 ： CR+EOI 4 ： LF 5 ： LF+EOI 6 ： CRLF 7 ： CRLF+EOI 8 ： NULL 9 ： 任意の文字
5	非同期入出力設定取得	0 ： 非同期入出力は行いません。 1 ： 非同期入出力を行います。

6	マイアドレス値取得	デバイス自身のマイアドレス(1 次アドレス)を取得します。0～30 の値が返されます。
7	2 次アドレス値取得	2 次アドレスの使用/値を取得します。 60h～7Eh　：　2 次アドレス値 FFh　　：　2 次アドレスは使用しません。
8	IEEE-488 バスハンドシェーク タイミング設定値取得	IEEE-488 バスハンドシェークタイミング設定値を取得します。 0　：　T1 は正常タイミング(2μs) 1　：　T1 は高速タイミング(500ns) 2　：　T1 は超高速タイミング(350ns)
9	パラレルポール実行時間 設定値取得	パラレルポール実行時間設定値を取得します。 0　：　T6 は 2μs 1　：　T6 は 10μs

ulpGetPrm

取得結果を格納する変数へのポインタを指定します。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_INP_PARAM
・ERR_DRV_RETURN

使用例

```
int ret;
unsigned long config;

ret = PciGpibExGetConfig(0, 0, &config);
if (!ret) {
    printf("Send/Receive Timeout=%ld¥n", config);
}
```

送受信タイムアウトの設定値を取得します。

参照

3.2 制御手順
3.2.1 コントローラとしての実行『2. デバイスの設定』
3.2.2 非コントローラとしての実行 『2. デバイスの設定』

6.PciGpibExGetStatus

現在のバス・ステータスを取得します。

```
int PciGpibExGetStatus(
  int            nBoardNo,    // デバイス番号
  unsigned int*  unpStatus    // バス・ステータスを格納する変数へのポインタ
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

unpStatus

現在のバス・ステータスを格納する変数へのポインタを指定します。
バス・ステータスは以下のビット情報で通知されます。

ビット	内容
31	IFC 受信を検出
30	SRQ 受信を検出
29	シリアルポール終了を検出
28	デバイストリガ受信を検出
27	デバイスクリア受信を検出
26	データ受信を検出
25	リスナ指定を検出
24	トーカ指定を検出
23	入出力完了を検出
22	リモート状態を検出
21	ロックアウト状態を検出
20	デリミタを検出
19	コントローラ・イン・チャージ (CIC) 検出
18	ATN 信号アクティブ検出
17	非同期入出力中に送信エラーを検出
16～0	予約

各ビットとも 1：検出 0：未検出
未使用のビットは 0 となります。

※各ビットにて該当モード以外の状態にて検出がされることがありますが、これは問題ありません。各ビットは該当状態のときのみチェックを行うようにしてください。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_DRV_RETURN

備考

・bit30(SRQ 受信)のクリアは、シリアル・ポーリングを行うと自動クリアされます。
・bit26(データ受信)のクリアは、受信関数呼び出しによるデータ受信時に自動的に行われます。
・bit25(リスナ指定)のクリアは、リスナ指定が解除された時に自動的に行われます。
・bit24(トーカ指定)のクリアは、トーカ指定が解除された時に自動的に行われます。
・bit23(入出力完了)のクリアは、データ送受信が開始すると自動的に行われます。
・bit22(リモート状態)のクリアは、ローカル状態になった時に自動クリアされます。
・bit21(ロックアウト状態)のクリアは、ロックアウト解除状態時に自動クリアされます。
・bit19(コントローラ・イン・チャージ検出)のクリアは、非コントローラ・イン・チャージ状態になった時に自動的に行われます。
・bit18(ATN 信号アクティブ検出)のクリアは、ATN 信号が非アクティブの状態になった時に、自動的に行われます。
・弊社 GP-IB インタフェースはデータ受信後、RFD ホールドオフ状態となりますので、以降、データ受信検出ビット(bit26)が有効とならない場合があります。
非コントローラの状態でデータを受信する場合など、リスナ指定検出後にデータ受信関数を実行するようにして下さい。

使用例

```
int ret;
unsigned int status;

ret = PciGpibExGetStatus(0, &status);
if(!ret){
    printf("Status : %x¥n",status);
}
```

デバイス番号 0 のデバイスのバス・ステータスを取得します。

参照

3.2 制御手順
3.2.2 非コントローラとしての実行 『2. バス・ステータス確認』

7.PciGpibExClrStatus

現在のバス・ステータス情報のクリアを行います。

```
int PciGpibExClrStatus(
  int           nBoardNo,   // デバイス番号
  unsigned int  unClrStat   // クリア・ステータス
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

unClrStat

クリア・ステータス情報を指定します。
クリア・ステータスは以下のビットパターンで指定します。

ビット	内容
31	IFC 受信を検出
30	SRQ 受信を検出
29	シリアルポール終了を検出
28	デバイストリガ受信を検出
27	デバイスクリア受信を検出
26	データ受信を検出
25	リスナ指定を検出
24	トーカ指定を検出
23	入出力完了を検出
22	リモート状態を検出
21	ロックアウト状態を検出
20	デリミタを検出
19	コントローラ・イン・チャージ (CIC) 検出
18	ATN 信号アクティブ検出
17	非同期入出力中に送信エラーを検出
16～0	予約

各ビットとも 1：検出クリア 0：クリアしない
未使用のビットには 0 を設定しておいてください。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_DRV_RETURN

使用例

```
int ret;

ret = PciGpibExClrStatus(0, 0x02000000);
```

デバイス番号 0 のシリアルポール終了検出をクリアします。

8. PciGpibExGetBusLine

GP-IB バスラインの状態を取得します。

```
int PciGpibExGetBusLine(
  int             nBoardNo,       // デバイス番号
  unsigned int*   unpGetBusLine   // バスライン状態を格納する変数へのポインタ
  int             nReserved       // 予約
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

unpGetBusLine

現在の GP-IB バスラインの状態を格納する変数へのポインタを指定します。
バスライン状態は以下のビットパターンで格納されます。

ビット	内容	
15	1：ATN 信号アサート	0：ATN 信号デアサート
14	1：DAV 信号アサート	0：DAV 信号デアサート
13	1：NDAC 信号アサート	0：NDAC 信号デアサート
12	1：NRFD 信号アサート	0：NRFD 信号デアサート
11	1：EOI 信号アサート	0：EOI 信号デアサート
10	1：SRQ 信号アサート	0：SRQ 信号デアサート
9	1：IFC 信号アサート	0：IFC 信号デアサート
8	1：REN 信号アサート	0：REN 信号デアサート
7～0	予約	

各信号の状態は、バスの入出力方向に依存します。
自分自身が出力している信号の場合は、その状態がリードバックされます。
未使用、予約のビットは不定値となります。

nReserved

予約パラメータです。0 を指定してください。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_DRV_RETURN

使用例

```
int ret;
unsigned int status;

ret = PciGpibExGetBusLine(0, &status, 0);
```

デバイス番号 0 のデバイスより現在の GP-IB バスラインの状態を取得します。

9.PciGpibExSendData

指定バイト数のデータを送信します。

```
int PciGpibExSendData(
  int             nBoardNo,    // デバイス番号
  int*            npAdrsTbl,   // 機器のアドレステーブルへのポインタ
  unsigned long   ulLength,    // 送信データ長
  void*           pvBuffer,    // 送信データ領域へのポインタ
  void*           pvFunc       // コールバック関数へのポインタ
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

npAdrsTbl

機器のアドレステーブルへのポインタを指定します。アドレステーブルへの終端には必ず－1を設定してください。
※非コントローラモードの時は NULL を設定します。

ulLength

送信データ長を指定します。

pvBuffer

送信データが格納されている領域へのポインタを指定します。

pvFunc

非同期動作終了時に実行させるコールバック関数へのポインタを指定します。
同期動作時には NULL を格納して本関数を呼び出してください。
コールバック関数の書式は void pvFunc(void);です。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_SEND_BUS_CMD
・ERR_DATA_SEND
・ERR_TRANSFER_TIMEOUT
・ERR_IFC_TRANS_EXIT
・ERR_NOW_BUS_OCCUPATION
・ERR_NOT_USE_NOSYS
・ERR_WAIT_SIGNAL_TMO
・ERR_DRV_RETURN

備考

・コントローラとして本関数を実行する場合は、コントローラ・イン・チャージの状態にする必要があります。本関数使用前にあらかじめ、PciGpibExSetIfc 関数を実行し、デバイスをコントローラ・イン・チャージの状態としてください。
・同期入出力時には、タイムアウト発生もしくは何らかの要因により処理が終了した場合において、本関数の処理を終了します。転送を強制的に中断することはできません。
・送信データ長としてデリミタ長をあわせて 1MByte をこえるデータは送信できません。
・デリミタは関数内で自動的に付加されるので、予めデータに付加しておく必要はありません。
・コールバック関数はデリミタ送出後に実行されます。

使用例

```
int ret;
int npAdrsTbl[2] = { 2, -1 };

ret = PciGpibExSendData(0, npAdrsTbl, 10, "0123456789", 0);
```

デバイス番号 0 のデバイスに対して、指定した機器への文字列"0123456789"のデータ送信を実行します(コントローラモード時)。

```
ret = PciGpibExSendData(0, NULL, 10, "0123456789", 0);
```

デバイス番号 0 のデバイスに対して、文字列"0123456789"のデータ送信を実行します(非コントローラモード時)。

参照

3.2 制御手順
3.2.1 コントローラとしての実行『4. データ送信』
3.2.2 非コントローラとしての実行『4. データ送信』

10. PciGpibExRecvData

指定デリミタを含め、データを受信します。

```
int PciGpibExRecvData(
  int             nBoardNo,     // デバイス番号
  int*            npAdrsTbl,    // 機器のアドレステーブルへのポインタ
  unsigned long*  ulpLength,    // 受信データ長を格納する変数へのポインタ
  void*           pvBuffer      // 受信データ領域へのポインタ
  void*           pvFunc        // コールバック関数へのポインタ
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

npAdrsTbl

機器のアドレステーブルへのポインタを指定します。アドレステーブルへの終端には必ず－1を設定します。※非コントローラモードの時は NULL を設定します。

ulpLength

受信バッファサイズが格納されている変数へのポインタを指定します。
希望するバッファサイズを設定して本関数を呼び出します。本関数呼び出し後、実際に受信したデータ長が格納されます。(同期入出力時)
※受信データ長としてデリミタ長をあわせて 1MByte をこえるデータは受信できません。データ長：0 の指定も不可です。

pvBuffer

受信データを格納する領域へのポインタを指定します。

pvFunc

非同期動作終了時に実行させるコールバック関数へのポインタを指定します。
同期動作時には NULL を格納して本関数を呼び出してください。
コールバック関数の書式は void pvFunc(void);です。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

- NORMAL_EXIT
- OK_EOI_DETECT
- OK_RECV_DATA_CNT
- ERR_BRD_NO
- ERR_SEND_BUS_CMD
- ERR_DATA_RECV
- ERR_TRANSFER_TIMEOUT
- ERR_IFC_TRANS_EXIT
- ERR_NOW_BUS_OCCUPATION
- ERR_NOT_USE_NOSYS
- ERR_DRV_RETURN

備考

・コントローラとして本関数を実行する場合は、コントローラ・イン・チャージの状態にする必要があります。本関数使用前にあらかじめ、PciGpibExSetIfc 関数を実行し、デバイスをコントローラ・イン・チャージの状態にしてください。
・受信バッファサイズ(ulpLength)の指定はデリミタサイズも含めて充分余裕を持って指定してください。これは、機器から送られてくるデータ長が 10 バイトと判明しており、デリミタに CRLF を指定している場合、受信バッファサイズは最低 12 バイト必要であることを示します。
※受信データ長としてデリミタ長をあわせて 1MByte をこえるデータは受信できません。データ長：0 の指定も不可です。
・同期入出力時には、タイムアウト発生もしくは何らかの要因により処理が終了した場合において、本関数の処理を終了します。転送を強制的に中断することはできません。
・コールバック関数はデリミタ受信後に実行されます。

使用例

```
int ret;
int npAdrsTbl[2] = { 2, -1 };
unsigned long len;
char RecvBuf[100];

len = sizeof(RecvBuf);
ret = PciGpibExRecvData(0, npAdrsTbl, &len, RecvBuf, 0);
```

デバイス番号 0 のデバイスに対して、指定した機器からのデータ受信の実行を行います。
(コントローラモード時)

```
len = sizeof(RecvBuf);
ret = PciGpibExRecvData(0, NULL, &len, RecvBuf, 0);
```

デバイス番号 0 のデバイスに対して、データ受信の実行を行います。
(非コントローラモード時)

参照

3.2 制御手順
3.2.1 コントローラとしての実行『5. データ受信』
3.2.2 非コントローラとしての実行『5. データ受信』

11. PciGpibExCheckStb

SRQ 信号送出後にコントローラからシリアル・ポールが行われて設定したステータス・バイトが通知済であるかどうかを確認します。

```
int PciGpibExCheckStb(
  int   nBoardNo    // デバイス番号
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。

- OK_SEND_STB
- NOT_EXEC_SPOLL
- ERR_BRD_NO
- ERR_DRV_RETURN

使用例

```
int ret;

ret = PciGpibExCheckStb( 0 );
if( ret > 0 ) printf("Status : %d¥n", ret);
```

デバイス番号 0 のデバイスに対して、シリアル・ポーリングが行われているかどうかを確認します。

12. PciGpibExSetSrq

ステータス・バイトを設定し、SRQ 信号を送出します。

```
int PciGpibExSetSrq(
  int   nBoardNo,   // デバイス番号
  int   nStb        // ステータスバイト値
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

nStb

ステータス・バイト値を指定します。
設定できるステータス・バイト情報は、bit6 を除く 7 ビットの情報です(bit6 は自動的に 1 に設定されます)。

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
○	×	○	○	○	○	○	○

この各ビット情報は特定のある事象状態をコントローラへ通知するために用いられます。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_PARAM_NO
・ERR_DRV_RETURN

使用例

```
int ret;

ret = PciGpibExSetSrq(0, 0x01);
```

デバイス番号 0 のデバイスのステータス・バイト値を 1 と設定し、SRQ を送出します。

参照

3.2 制御手順
3.2.2 非コントローラとしての実行『6. SRQの発行』

13. PciGpibExSetIst

パラレルポール応答モードを指定します。

```
int PciGpibExSetIst(
  int   nBoardNo,   // デバイス番号
  int   nMode       // パラレルポール応答モード
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

nMode

パラレルポール応答モードを指定します。

値	内容
1	[ist - Individual Status]ビットをセットします。
0	[ist - Individual Status]ビットをクリアします。

パラレルポール応答の論理はコントローラからの指示により決まります。

備考

本関数は非コントローラ状態でのみ有効です。
パラレルポール・コンフィギュレーション時の応答フラグを設定します。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_PARAM_NO
・ERR_DRV_RETURN

使用例

```
int ret;

ret = PciGpibExSetIst( 0, 1 );
```

デバイス番号 0 のデバイスに対して、ist ビットをセット(パラレル・ポール応答を肯定)します。

参照

3.2 制御手順
3.2.2 非コントローラとしての実行手順『7. パラレルポール・コンフィギュレーション』

14. PciGpibExSetPp2

ローカルコンフィグレーションにおける、パラレルポール応答モード(pp2)を指定します。

```
int PciGpibExSetPp2(
  int    nBoardNo,   // デバイス番号
  int    nMode       // パラレルポール応答モード
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

nMode

パラレルポール応答モードを指定します。
指定方法は、bit4 が応答指定、bit3 が応答極性、bit2～bit0 の 3 ビットで DIO ラインを指定します。

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0	0	0	U	S	p3	p2	p1

p3	p2	p1	DIO ライン 1～8 の指定
0	0	0	DIO1 で応答
0	0	1	DIO2 で応答
0	1	0	DIO3 で応答
0	1	1	DIO4 で応答
1	0	0	DIO5 で応答
1	0	1	DIO6 で応答
1	1	0	DIO7 で応答
1	1	1	DIO8 で応答

S	応答極性の指定
0	応答ラインは 0 をアサートする
1	応答ラインは 1 をアサートする

パラレルポール時、ist と S が一致するなら、bit2～bit0 で指定した DIO ラインをアサートします。

U	パラレルポール応答の指定
0	パラレルポールに応答する
1	パラレルポールに応答しない

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_PARAM_NO
・ERR_DRV_RETURN

備考

本関数は非コントローラモードでのみ有効です。
ローカル・コンフィギュレーション時の応答モードを設定します。

使用例

```
int ret;

ret = PciGpibExSetPp2( 0, 9 );
```

デバイス番号 0 のデバイスに対して、パラレル・ポール時に、ist が 1 ならば、DIO ライン 2 をアサートするよう設定します。

参照

3.2 制御手順
3.2.2 非コントローラとしての実行手順『7. パラレルポール・コンフィギュレーション』

15. PciGpibExSetIfc

指定された時間、IFC 信号の送出を行います。

```
int PciGpibExSetIfc(
  int   nBoardNo,   // デバイス番号
  int   nTime       // IFC 送出時間
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

nTime

100μs 単位で IFC 送出時間を設定します。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

- NORMAL_EXIT
- ERR_BRD_NO
- ERR_PARAM_NO
- ERR_SEND_BUS_CMD
- ERR_NOT_SYS_CONTROLLER
- ERR_DRV_RETURN

備考

本関数はシステムコントローラでのみ使用できます。

使用例

```
int ret;

ret = PciGpibExSetIfc(0, 1);
```

デバイス番号 0 のデバイスから IFC を 100μs 送出します。

参照

3.2 制御手順
3.2.1 コントローラとしての実行手順『3. GP-IBインタフェースの初期化』

16. PciGpibExSetRen

REN 信号を有効にします。

```
int PciGpibExSetRen(
  int   nBoardNo    // デバイス番号
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_SEND_BUS_CMD
・ERR_NOT_SYS_CONTROLLER
・ERR_DRV_RETURN

備考

本関数はシステムコントローラでのみ使用できます。
システムコントローラとして IFC 送出に引き続いて REN 信号を有効にすることで機器をリモート状態に設定します

使用例

```
int ret;

ret = PciGpibExSetRen(0);
```

デバイス番号 0 のデバイスの REN 信号を有効にします。

参照

3.2 制御手順
3.2.1 コントローラとしての実行『3. GP-IBインタフェースの初期化』

17. PciGpibExResetRen

REN 信号を無効にします。

```
int PciGpibExResetRen(
  int   nBoardNo   // デバイス番号
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_SEND_BUS_CMD
・ERR_NOT_SYS_CONTROLLER
・ERR_DRV_RETURN

備考

本関数はシステムコントローラでのみ使用できます。
すべてのデバイスをローカル状態(フロント・パネル制御可)の状態にします。

使用例

```
int ret;

ret = PciGpibExResetRen(0);
```

デバイス番号 0 のデバイスの REN 信号を無効にします。

18. PciGpibExSetRemote

指定された機器をリモートモードにします。

```
int PciGpibExSetRemote (
  int   nBoardNo,    // デバイス番号
  int*  npAdrsTbl    // 機器のアドレステーブルへのポインタ
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

npAdrsTbl

リモートモードに設定する機器の機器アドレスを格納しているアドレステーブルへのポインタを指定します。アドレステーブルへの終端には必ず－1 を設定します。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_SEND_BUS_CMD
・ERR_NOT_USE_NOSYS
・ERR_NOT_SYS_CONTROLLER
・ERR_DRV_RETURN

備考

本関数はシステムコントローラでのみ使用できます。
機器を GP-IB による制御可能状態とし，フロントパネルからの制御を禁止します。

使用例

```
int ret;
int npAdrsTbl[2] = { 2, -1 };

ret = PciGpibExSetRemote(0, npAdrsTbl);
```

デバイス番号 0 のデバイスに対して npAdrsTbl に記述した機器をリモート状態に設定します。

19. PciGpibExSetRwls

指定した機器をリモート・ロックアウト状態にします。

```
int PciGpibExSetRwls(
  int   nBoardNo,   // デバイス番号
  int*  npAdrsTbl   // 機器のアドレステーブルへのポインタ
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

npAdrsTbl

リモート・ロックアウト状態にさせる機器の機器アドレスを格納しているアドレステーブルへのポインタを指定します。
アドレステーブルへの終端には必ず－1 を設定します。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_SEND_BUS_CMD
・ERR_NOT_USE_NOSYS
・ERR_NOT_SYS_CONTROLLER
・ERR_DRV_RETURN

備考

本関数はシステムコントローラでのみ使用できます。

使用例

```
int ret;
int npAdrsTbl[2] = { 2, -1 };

ret = PciGpibExSetRwls(0, npAdrsTbl);
```

デバイス番号 0 のデバイスから指定した機器(アドレス 2)に対してリモート・ロックアウトの設定を行います。

20. PciGpibExReSysCtrl

システム・コントローラの要求または解除を行います。

```
int PciGpibExReSysCtrl(
  int    nBoardNo,   // デバイス番号
  int    nMode       // モード設定
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

nMode

システム・コントローラの要求または解除のモードを指定します。下記の値を指定してください。

値	内容
0	システム・コントローラの解除を行います。
1	システム・コントローラの要求を行います。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_PARAM_NO
・ERR_SEND_BUS_CMD
・ERR_DRV_RETURN

備考

本関数はコントローラ・イン・チャージでのみ使用できます。

使用例

```
int ret;

ret = PciGpibExReSysCtrl(0, 1);
```

デバイス番号 0 のデバイスからシステム・コントローラの要求を行います。

21. PciGpibExExecTrigger

指定された機器に対してトリガ設定を行います。

```
int PciGpibExExecTrigger(
  int   nBoardNo,   // デバイス番号
  int*  npAdrsTbl   // 機器のアドレステーブルへのポインタ
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

npAdrsTbl

デバイストリガ機能を動作させる機器の機器アドレスを格納しているアドレステーブルへのポインタを指定します。
アドレステーブルへの終端には必ず－1 を設定します。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_SEND_BUS_CMD
・ERR_NOT_USE_NOSYS
・ERR_DRV_RETURN

備考

本関数はコントローラ・イン・チャージでのみ使用できます。

使用例

```
int ret;
int npAdrsTbl[2] = { 2, -1 };

ret = PciGpibExExecTrigger(0, npAdrsTbl);
```

デバイス番号 0 のデバイスから npAdrsTbl に記述した機器に対してトリガ設定を行います。

22. PciGpibExExecDevClear

全ての機器に対してデバイスクリア機能(装置の初期化)を動作させます。

```
int PciGpibExExecDevClear(
  int   nBoardNo   // デバイス番号
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_SEND_BUS_CMD
・ERR_NOT_USE_NOSYS
・ERR_DRV_RETURN

備考

本関数はコントローラ・イン・チャージでのみ使用できます。

使用例

```
int ret;

ret = PciGpibExExecDevClear( 0 );
```

デバイス番号 0 のデバイスからデバイスクリアの送出を行います。

23. PciGpibExExecSdc

指定された機器に対してデバイスクリア機能(装置の初期化)を動作させます。

```
int PciGpibExExecSdc(
  int   nBoardNo,   // デバイス番号
  int*  npAdrsTbl   // 機器のアドレステーブルへのポインタ
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

npAdrsTbl

デバイスクリア機能を動作させる機器の機器アドレスを格納しているアドレステーブルへのポインタを指定します。アドレステーブルへの終端には必ず－1 を設定します。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_SEND_BUS_CMD
・ERR_NOT_USE_NOSYS
・ERR_DRV_RETURN

備考

本関数はコントローラ・イン・チャージでのみ使用できます。

使用例

```
int ret;

int npAdrsTbl[2] = {2,-1};
ret = PciGpibExExecSdc(0, npAdrsTbl);
```

デバイス番号 0 のデバイスから指定した機器(アドレス番号 2)に対してセレクテッドデバイスクリアの送出を行います。

24. PciGpibExSetLocal

指定したデバイスをローカル状態にします（フロント・パネル制御可）。

```
int PciGpibExSetLocal(
  int   nBoardNo,   // デバイス番号
  int*  npAdrsTbl   // 機器のアドレステーブルへのポインタ
);
```

パラメータ

ulEventFactor

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

npAdrsTbl

ローカル状態にする機器の機器アドレスを格納しているアドレステーブルへのポインタを指定します。アドレステーブルへの終端には必ず－1 を設定します。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_SEND_BUS_CMD
・ERR_NOT_USE_NOSYS
・ERR_DRV_RETURN

備考

本関数はコントローラ・イン・チャージでのみ使用できます。

使用例

```
int ret;
int npAdrsTbl[2] = {2, -1};

ret = PciGpibExSetLocal(0, npAdrsTbl);
```

デバイス番号 0 のデバイスから指定した機器(アドレス 2)に対してローカル状態の設定を行います。

25. PciGpibExSetLlo

全ての機器をローカル・ロックアウト状態(リモート状態時のローカルボタンの入力を禁止)にします。

```
int PciGpibExSetLlo(
  int    nBoardNo    // デバイス番号
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_SEND_BUS_CMD
・ERR_NOT_USE_NOSYS
・ERR_DRV_RETURN

備考

本関数はコントローラ・イン・チャージでのみ使用できます。
機器によっては、ローカルロックアウト状態を解除するためには、電源を再投入しなければならないものもあります。

使用例

```
int ret;

ret = PciGpibExSetLlo(0);
```

デバイス番号 0 のデバイスからローカル・ロックアウトの送出を行います。

26. PciGpibExExecPassCtrl

指定した機器に対してパス・コントロールを行います。

```
int PciGpibExExecPassCtrl(
  int    nBoardNo,    // デバイス番号
  int*   npAdrsTbl    // 機器のアドレステーブルへのポインタ
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

npAdrsTbl

パス・コントロールを行う機器の機器アドレスを格納しているアドレステーブルへのポインタを指定します。アドレステーブルへの終端には必ず－1 を設定します。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_SEND_BUS_CMD
・ERR_NOT_USE_NOSYS
・ERR_DRV_RETURN

備考

本関数はコントローラ・イン・チャージでのみ使用できます。
本関数を実行することにより、相手にコントローラ機能があるかどうかに関係なく、本デバイスはコントローラ・イン・チャージが解除されます。

使用例

```
int ret;
int npAdrsTbl[2] = { 2, -1 };

ret = PciGpibExExecPassCtrl(0, npAdrsTbl);
```

デバイス番号 0 のデバイスから指定した機器(アドレス 2)に対してパス・コントロールを実行します。

27. PciGpibExExecFindListener

バス上に接続されているリスナ(デバイス)を探します。(IEEE-488. 2 関数)

```
int PciGpibExExecFindListener (
  int    nBoardNo,        // デバイス番号
  int*   npAdrsTbl,       // 探索する機器のアドレスを格納したテーブルへのポインタ
  int*   npFindAdrsTbl,   // 見つかった機器のアドレスを格納するテーブルへのポインタ
  int*   npFindCnt        // 見つかった機器の総数を格納する変数へのポインタ
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

npAdrsTbl

探索する機器のアドレスを格納したテーブルへのポインタを指定します。
接続されているか確認したいリスナ(デバイス)の GP-IB アドレスを格納したテーブルへのポインタを指定します。アドレステーブルへの終端には必ず－1 を設定します。

npFindAdrsTbl

見つかった機器のアドレスを格納するテーブルへのポインタを指定します。
本 API 実行後、npAdrsTbl 引数にて設定した機器が見つかった場合、その機器のアドレスが格納されます。
この引数に指定するテーブルのサイズは、npAdrsTbl 引数と同じか、それ以上の格納領域を確保しておく必要があります。
本 API 呼び出し後、自動で終端に-1 が付加されます。

npFindCnt

見つかった機器の総数を格納する変数へのポインタを指定します。
本 API 実行後、見つかった機器の総数が格納されます。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

- NOT_FOUND_LISTNER
- NORMAL_EXIT
- ERR_BRD_NO
- ERR_SEND_BUS_CMD
- ERR_NOT_USE_NOSYS
- ERR_DRV_RETURN

備考

本関数はコントローラ・イン・チャージでのみ使用できます。
npFindAdrsTbl 引数(機器アドレス探索結果テーブル)のサイズは、npAdrsTbl 引数(探索機器アドレス格納テーブル)と同一もしくはそれ以上のサイズを確保してください。

使用例

```
int ret;
int npAdrsTbl[3] = { 2, 7, -1 };
int npFindAdrsTbl[3];
int nFindCnt, i;

ret = PciGpibExExecFindListener(0, npAdrsTbl, npFindAdrsTbl, &nFindCnt);

if (!ret) {
    if (nFindCnt > 0) {
        printf("Find Count:%d¥n",nFindCnt);
        for (i = 0; i < nFindCnt; i++) printf("Find Address:%d¥n",npFindAdrsTbl[i]);
    }
}
```

デバイス番号 0 のデバイスから npAdrsTbl'に格納されているアドレスの探索を実行し、見つかった機器アドレスとその総数を取得します。

28. PciGpibExExecDevReset

指定された機器に完全にリセットします。(IEEE-488.2 関数)

```
int PciGpibExExecDevReset(
  int   nBoardNo,    // デバイス番号
  int*  npAdrsTbl    // リセットを行う機器アドレスのテーブルへのポインタ
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

npAdrsTbl

リセットを行う機器の機器アドレスを格納しているアドレステーブルへのポインタを指定します。アドレステーブルへの終端には必ず－1 を設定します。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_SEND_BUS_CMD
・ERR_DATA_SEND
・ERR_TRANSFER_TIMEOUT
・ERR_IFC_TRANS_EXIT
・ERR_NOW_BUS_OCCUPATION
・ERR_NOT_SYS_CONTROLLER
・ERR_DRV_RETURN

備考

本関数は次の処理は行います。

1. バスの初期化
IFC、次に REN がアクティブになります。その結果、全てのデバイスはアドレスされていない状態になり、GP-IB インタフェースデバイスがコントローラ・イン・チャージ (CIC) になります。

2. メッセージ交換初期化
DCL バスコマンドが全ての接続されたデバイスに送られます。その結果として全ての IEEE-488.2 規格にあったデバイスは次の Reset (RST) メッセージを受信できるようになります。

3. デバイス初期化
*RST メッセージが'npAdrsTbl'引数にて指定されるアドレスのデバイスに送られます。
その結果として、デバイス特有の機能が初期化されます。

使用例

```
int ret;
int npAdrsTbl[2] = { 2, -1 };

ret = PciGpibExExecDevReset(0, npAdrsTbl);
```

デバイス番号 0 のデバイスから指定した機器(アドレス 2)に対してリセットの実行を行います。

29. PciGpibExGoStandby

デバイスの動作状態をコマンドモードからデータモードへ遷移させます。

```
int PciGpibExGoStandby(
  int   nBoardNo,    // デバイス番号
  int   nMode        // データモード
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

nMode

データモードを指定します。下記の値を指定してください。

値	内容
0	通常のデータモードに遷移します。
1	シャドウ・ハンドシェークモードに遷移します。 本モード指定時、機器間のデータ転送を監視し、デリミタ検出時には自動でコマンドモードに遷移します。デリミタ指定時には、受信デリミタの設定が使用されます。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_PARAM_NO
・ERR_SEND_BUS_CMD
・ERR_NOT_USE_NOSYS
・ERR_DRV_RETURN

備考

・本関数はコントローラ・イン・チャージでのみ使用できます。
・本関数でシャドウ・ハンドシェークモードを使用することにより、自分自身が関与しない機器間でのデータ転送を行うことができます。また機器間でのデータ転送において、デリミタ検出時に自動でコマンドモードに戻ることができます。
・本関数を使用してシャドウ・ハンドシェークを行う場合、デリミタ指定は必ず行うようにしてください。

使用例

```
int ret;

ret = PciGpibExGoStandby(0, 0);
```

デバイス番号 0 のデバイスの動作モードを通常のデータモードに遷移させます。

30. PciGpibExGoActCtrller

デバイスの動作状態をデータモードからコマンドモードへ遷移させます。

```
int PciGpibExGoActCtrller(
  int   nBoardNo,    // デバイス番号
  int   nMode        // コマンドモードへの遷移方法
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

nMode

コマンドモードへの遷移方法を指定します。下記の値を指定してください。

値	内容
0	非同期にてコマンドモードへ遷移します。
1	同期してコマンドモードへ遷移します。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 戻り値一覧』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_PARAM_NO
・ERR_SEND_BUS_CMD
・ERR_NOT_USE_NOSYS
・ERR_DRV_RETURN

備考

・本関数はコントローラ・イン・チャージでのみ使用できます。
・本関数は PciGpibExGoStandby 関数で通常のデータモードに遷移後にアプリケーションプログラムでコマンドモードにするために使用します。本関数はデータ転送が完了しているかどうかに関係無く、ただちにコマンドモードに遷移しますので、実行のタイミングによっては転送中のデータが失われる可能性があります。

使用例

```
int ret;

ret = PciGpibExGoActCtrller( 0, 0 );
```

デバイス番号 0 のデバイスの動作モードをデータモードからコマンドモードへ非同期にて遷移させます。

31. PciGpibExExecSpoll

指定された機器に対してシリアル・ポーリングを行い、ステータス・バイトの受信を行います。

```
int PciGpibExExecSpoll(
  int     nBoardNo,    // デバイス番号
  int*    npAdrsTbl,   // シリアル・ポールを行う機器のアドレステーブルへのポインタ
  int*    npStbTbl,    // ステータス・バイト格納テーブルへのポインタ
  int*    npStbAdrs    // SRQ 送出機器アドレス格納テーブルへのポインタ
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

npAdrsTbl

シリアル・ポールの対象の機器の機器アドレスを格納しているアドレステーブルへのポインタを指定します。アドレステーブルへの終端には必ず－1 を設定します。

npStbTbl

シリアル・ポーリングを行い受信した各機器の有効なステータス・バイトを格納するテーブルへのポインタを指定します。本関数呼び出し後、終端に－1 が付加されます。

npStbAdrs

SRQ を送出し、有効なステータス・バイトを持っていた各機器のアドレスを格納するテーブルへのポインタを指定します。本関数呼び出し後、終端に－1 が付加されます。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

- NORMAL_EXIT
- ERR_BRD_NO
- ERR_SEND_BUS_CMD
- ERR_NO_ACTIVE_SRQ
- ERR_RECV_STB_TIMEOUT
- ERR_NOT_USE_NOSYS
- ERR_DRV_RETURN

備考

- 本関数はコントローラ・イン・チャージでのみ使用できます。
- 本関数を呼び出す際は、PciGpibExCheckSrq 関数もしくは PciGpibExWaitSignal 関数にて、機器よりの SRQ を受け付けていることを確認してから呼び出してください。
- アドレステーブルに複数の機器アドレスを指定することにより、複数の機器に対してシリアル・ポーリングを行うことが可能です。この時、ステータス・バイト格納テーブルと SRQ 送出機器アドレス格納テーブルには、機器の数+終端を格納することができるサイズが必要です。

使用例

```
int ret;
int npAdrsTbl[2]  = { 2, -1 };
int npStbTbl[2], npStbAdrs[2];

ret = PciGpibExExecSpoll(0, npAdrsTbl, npStbTbl, npStbAdrs);
if (!ret) {
      printf("ADRESS : %d,STB : %x¥n", npStbAdrs[0], npStbTbl[0]);
}
```

デバイス番号 0 のデバイスから npAdrsTbl に記述される各機器に対してシリアル・ポーリングを実行します。

参照

3.2 制御手順
3.2.1 コントローラとしての実行手順『6. シリアルポーリング』

32. PciGpibExCheckSrq

SRQ 信号の有効/無効を確認します。

```
int PciGpibExCheckSrq(
  int   nBoardNo   // デバイス番号
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NOT_ACTIVE_SRQ
・ACTIVE_SRQ
・ERR_BRD_NO
・ERR_NOT_USE_NOSYS
・ERR_DRV_RETURN

備考

・本関数はコントローラ・イン・チャージの状態にて使用できます。
・PciGpibExInitBoard 関数実行後は SRQ 受信割り込み許可状態となっています。
・本関数を実行後、機器からの SRQ 信号がアサート(有効)になっている場合には、機器からのステータスバイトを受信するために、必ず PciGpibExExecSpoll 関数を実行してください。

使用例

```
int ret;
int npAdrsTbl[2]  = { 2, -1 };
int npStbTbl[2], npStbAdrs[2];

ret = PciGpibExCheckSrq( 0 );
if (ret == ACTIVE_SRQ ) {
    ret = PciGpibExExecSpoll(0, npAdrsTbl, npStbTbl, npStbAdrs);
    if (!ret) {
        printf("ADRESS : %d,STB : %x¥n", npStbAdrs[0], npStbTbl[0]);
    }
}
```

デバイス番号 0 のデバイスに対する SRQ 受信を確認します。

参照

3.2 制御手順
3.2.1 コントローラとしての実行手順『6. シリアルポーリング』

33. PciGpibExClearSrq

SRQ 受信フラグが有効にも関わらず、シリアル・ポーリング実行後、SRQ 発信機器を検出できなかった場合において、本関数にて強制的にアクセスデバイスの SRQ 受信フラグのクリアを行います。

```
int PciGpibExClearSrq(
  int  nBoardNo  // デバイス番号
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_NOT_USE_NOSYS
・ERR_DRV_RETURN

備考

・本関数はコントローラ・イン・チャージでのみ使用できます。
・本関数はアクセスデバイス自身の SRQ 受信フラグのクリアを行いますが、これによって、GP-IB バスラインの SRQ 信号がデアサート(信号無効状態)にはなりません。
・意図しない GP-IB バスラインの SRQ 信号アサート状態を解除するためには、SRQ 信号をアサートした該当機器が SRQ 信号をデアサートさせるか、または全ての機器の GP-IB インタフェース機能を再初期化させる必要があります。

使用例

```
int ret;

ret = PciGpibExClearSrq( 0 );
```

デバイス番号 0 のデバイスに対する SRQ 受信フラグをクリアします。

34. PciGpibExEnableSrq

PciGpibExDisableSrq 関数実行後に SRQ 受信割り込みを許可したい場合に本関数を使用します。
本関数呼び出し後に SRQ 信号がアサートされた時、PciGpibExCheckSrq 関数にて SRQ 受信フラグが有効となります。

```
int PciGpibExEnableSrq(
  int   nBoardNo    // デバイス番号
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_NOT_USE_NOSYS
・ERR_DRV_RETURN

備考

本関数はコントローラ・イン・チャージでのみ使用できます。
PciGpibExInitBoard 関数実行後は SRQ 受信割り込み許可状態となっています。

使用例

```
int ret;

ret = PciGpibExEnableSrq( 0 );
```

デバイス番号 0 のデバイスに対して SRQ 受信割り込みを許可します。

35. PciGpibExDisableSrq

本関数を呼び出すことで、SRQ 受信割り込みが禁止されます。
本関数呼び出し後に SRQ 信号がアサートされても、PciGpibExCheckSrq 関数では SRQ 受信フラグは無効のままです。

```
int PciGpibExDisableSrq(
  int   nBoardNo    // デバイス番号
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_NOT_USE_NOSYS
・ERR_DRV_RETURN

備考

・本関数はコントローラ・イン・チャージでのみ使用できます。
・PciGpibExInitBoard 関数実行後は SRQ 受信割り込み許可状態となっています。
・本関数実行後は、SRQ 受信割込みは禁止となるため、PciGpibExCheckSrq 関数による SRQ 受信の確認はできません。ただし、本関数実行後に SRQ 信号がアサートされ、SRQ 信号アサート状態のままで PciGpibExEnableSrq 関数を呼び出しますと、SRQ 受信割込みが有効となるため、PciGpibExCheckSrq 関数にて SRQ 受信が有効となります。

使用例

```
int ret;

ret = PciGpibExDisableSrq( 0 );
```

デバイス番号 0 のデバイスに対して SRQ 受信割り込みを禁止します。

36. PciGpibExExecPpoll

パラレルポーリングを実行します。
パラレルポーリングを行うことで、複数の機器からの応答を DIO ラインから同時に得ます。

```
int PciGpibExExecPpoll(
  int   nBoardNo,   // デバイス番号
  int*  npPst       // パラレルポール・ステータスを格納する変数へのポインタ
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

npPst

パラレルポール・ステータスを格納する変数へのポインタを指定します。
パラレルポーリング時のステータスが格納されます。
この応答の真偽はアプリケーションプログラムにて判断する必要があります。
各機器毎の情報を得るために、事前に各機器に対してどのビットを使用するのか、また応答の極性をどうするのかについては、PciGpibExCfgPpoll 関数で指定する方法(リモートコンフィギュレーション)と機器のフロントパネルからの設定(ローカルコンフィギュレーション)の 2 通りがあります。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_SEND_BUS_CMD
・ERR_NOT_USE_NOSYS
・ERR_DRV_RETURN

備考

本関数はコントローラ・イン・チャージでのみ使用できます。

使用例

```
int ret;
int npPst;

ret = PciGpibExExecPpoll(0, &npPst);
if (!ret) {
      printf("Parallel Polling Status : %x¥n", npPst);
}
```

デバイス番号 0 のデバイスからパラレル・ポーリングを実行します。

37. PciGpibExCfgPpoll

パラレルポールに応答する機器に対して、応答する DIO ラインの割り当てと、その極性を指定します。(リモート・コンフィグレーション)

```
int PciGpibExCfgPpoll(
  int    nBoardNo,    // デバイス番号
  int*   npAdrsTbl,   // 機器のアドレステーブルへのポインタ
  int    nPpe         // パラレルポール応答指定
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

npAdrsTbl

パラレルポールに応答させる機器のアドレステーブルへのポインタを指定します。
アドレステーブルへの終端には必ず－1 を設定します。

nPpe

パラレルポール応答指定
指定方法は、bit3 が応答極性、bit2～bit0 の 3 ビットで DIO ラインを指定します。
複数の機器に対して同一の DIO ラインを指定することも可能ですが、ワイヤード OR 化されたビット情報からは、グループ識別は可能ですが機器を特定することはできません。

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0	0	0	0	S	p3	p2	p1

p3	p2	p1	DIO ライン 1～8 の指定
0	0	0	DIO1 で応答
0	0	1	DIO2 で応答
0	1	0	DIO3 で応答
0	1	1	DIO4 で応答
1	0	0	DIO5 で応答
1	0	1	DIO6 で応答
1	1	0	DIO7 で応答
1	1	1	DIO8 で応答

S	応答極性の指定
0	応答ラインは 0 をアサートします
1	応答ラインは 1 をアサートします

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_PARAM_NO
・ERR_SEND_BUS_CMD
・ERR_NOT_USE_NOSYS
・ERR_DRV_RETURN

使用例

```
int ret;
int npAdrsTbl[2] = { 2, -1 };
int nPpe;

nPpe = 0x08;
ret = PciGpibExCfgPpoll(0, npAdrsTbl, nPpe);
```

デバイス番号 0 のデバイスから、指定した機器に対してパラレルポール・コンフィギュレーションを行います。
パラレルポーリングの応答は DIO1 ラインで応答するように設定します。

38. PciGpibExUnCfgPpoll

リモート・コンフィギュレーションで、パラレルポール・コンフィグレーションされている機器に対して PPC(Parallel Poll Disable)を送出し、パラレルポール応答条件を解除します。

```
int PciGpibExUnCfgPpoll(
  int   nBoardNo,   // デバイス番号
  int*  npAdrsTbl   // 機器のアドレステーブルへのポインタ
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

npAdrsTbl

パラレルポールに応答解除させる機器の機器アドレスを格納しているアドレステーブルへのポインタを指定します。
アドレステーブルへの終端には必ず-1 を設定します。
また、アドレステーブルの先頭に-1 を設定した場合は、PPU(Parallel Poll Unconfigure)を送出してリモート・コンフィギュレーション機能を持つ全ての機器の応答条件を解除します。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_SEND_BUS_CMD
・ERR_NOT_USE_NOSYS
・ERR_DRV_RETURN

備考

本関数はコントローラ・イン・チャージでのみ使用できます。

使用例

```
int ret;
int npAdrsTbl[2] = { 2, -1 };

ret = PciGpibExUnCfgPpoll(0, npAdrsTbl);
```

デバイス番号 0 のデバイスから、パラレルポール応答条件解除を行います。

39. PciGpibExWriteBusCmd

機器に対してバス・コマンド(マルチライン・インタフェース・メッセージ)を発行します。

```
int PciGpibExWriteBusCmd(
  int   nBoardNo,   // デバイス番号
  int*  npCmdTbl    // バス･コマンドテーブルへのポインタ
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

npCmdTbl

バス上に送出されるコマンドを格納してあるテーブルへのポインタを指定します。
バス上に送出されるコマンドは一切加工されません。また、コマンドテーブルの終端には必ず-1 を設定してください。最大 255 のコマンドを送出できます。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_SEND_BUS_CMD
・ERR_NOT_USE_NOSYS
・ERR_DRV_RETURN

備考

・ 本関数はコントローラ・イン・チャージでのみ使用できます。
・ コマンドテーブルに設定されるデータは、そのままの形で GP-IB バスに送出されます。

使用例

```
int ret;
int npCmdTbl[2] = { 63, -1 };

ret = PciGpibExWriteBusCmd(0, npCmdTbl);
```

デバイス番号 0 のデバイスから npCmdTbl に記述されているバス・コマンド(ここでは UNL)を送出します。

40. PciGpibExSendFile

ファイルからデータを読み込み、GP-IB バス上に送信します。

```
int PciGpibExSendFile(
  int     nBoardNo,    // デバイス番号
  int*    npAdrsTbl,   // 機器のアドレステーブルへのポインタ
  char*   pFname       // ファイル名へのポインタ
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

npAdrsTbl

機器のアドレステーブルへのポインタを指定します。
アドレステーブルへの終端には必ず－1 を設定します。
※非コントローラモード時は NULL を設定します。

pFname

ファイル名へのポインタを指定します。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

- NORMAL_EXIT
- ERR_BRD_NO
- ERR_SEND_BUS_CMD
- ERR_DATA_SEND
- ERR_TRANSFER_TIMEOUT
- ERR_IFC_TRANS_EXIT
- ERR_NOW_BUS_OCCUPATION
- ERR_FILE_ACCESS
- ERR_NOT_USE_NOSYS
- ERR_WAIT_SIGNAL_TMO
- ERR_DRV_RETURN

備考

- コントローラとして本関数を実行する場合は、コントローラ・イン・チャージの状態にする必要があります。本関数使用前にあらかじめ、PciGpibExSetIfc 関数を実行し、デバイスをコントローラ・イン・チャージの状態としてください。
- ファイルの内容は加工されずにそのままの形でバスに送出されます。
- ファイルの最後には、送信デリミタが付加されます。そのため、CRLF 等のデリミタを指定していた場合、ファイルの最後に CRLF が付加されてしまいます。従ってファイル転送等に使用する場合においては、送信デリミタには必ず EOI のみを指定してください。
  ※デリミタ長をあわせて 1MByte をこえるファイル送信はできません。
- ファイルはバイナリデータとしてオープンされ読み込まれます。また、オープンされたファイルは読み込み終了後、ただちにクローズされます。
- 本関数では非同期入出力設定がされていても、非同期処理は行われず、同期して処理されます。

使用例

```
int ret;
int npAdrsTbl[2] = { 2, -1 };

ret = PciGpibExSendFile(0, npAdrsTbl, "test.dat");
```

デバイス番号 0 のデバイスに対して、ファイル名"test.dat"のファイルを読み込み、指定した機器に送出します(コントローラモード時)。

```
ret = PciGpibExSendFile(0, NULL, "test.dat");
```

デバイス番号 0 のデバイスに対して、ファイル名"test.dat"のファイルを読み込み、GP-IB バス上に送出します(非コントローラモード時)。

41.PciGpibExRecvFile

GP-IB バスから受信したデータをファイルに書き込みます。

```
int PciGpibExRecvFile(
  int             nlBoardNo,      // デバイス番号
  int*            npAdrsTbl,      // 機器のアドレステーブルへのポインタ
  unsigned long*  ulpLength       // 受信データ長を格納する変数へのポインタ
  char*           pFname          // ファイル名へのポインタ
);
```

パラメータ

nlBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

npAdrsTbl

機器のアドレステーブルへのポインタを指定します。アドレステーブルへの終端には必ず－1を設定します。
※非コントローラモードの時は NULL を設定します。

ulpLength

受信バッファサイズが格納されている変数へのポインタを指定します。
希望するバッファサイズを設定して本関数を呼び出します。
本関数呼び出し後、実際に受信したデータ長が格納されます。
※受信データ長としてデリミタ長をあわせて 1MByte をこえるデータは受信できません。データ長：0 の指定も不可です。

pFname

ファイル名へのポインタ

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

- NORMAL_EXIT
- OK_EOI_DETECT
- OK_RECV_DATA_CNT
- ERR_BRD_NO
- ERR_SEND_BUS_CMD
- ERR_DATA_RECV
- ERR_TRANSFER_TIMEOUT
- ERR_FILE_ACCESS
- ERR_NOT_USE_NOSYS
- ERR_IFC_TRANS_EXIT
- ERR_NOW_BUS_OCCUPATION
- ERR_DRV_RETURN

備考

・コントローラとして本関数を実行する場合は、コントローラ・イン・チャージの状態にする必要があります。本関数使用前にあらかじめ、PciGpibExSetIfc 関数を実行し、デバイスをコントローラ・イン・チャージの状態としてください。
・バスから読み込まれたデータに受信デリミタが含まれていた場合、その時点で受信処理を終了します。従ってファイル転送等に使用する場合において、ファイル中に CRLF 等のコードが含まれている場合には、受信デリミタには必ず EOI のみを指定してください。
・ファイルはバスから読み込まれたデータがそのままの形式で書き込まれます。
・また、オープンされたファイルは書き込み終了後、ただちにクローズされます。既にファイルが存在していた場合は、上書きされます。
・本関数では非同期入出力設定がされていても、非同期処理は行われず、同期して処理されます。
・受信バッファサイズ(ulpLength)の指定はデリミタサイズも含めて充分余裕を持って指定してください。これは、機器から送られてくるデータ長が 10 バイトと判明しており、デリミタに CRLF を指定している場合、受信バッファサイズは最低 12 バイト必要であることを示します。
※受信データ長としてデリミタ長をあわせて 1MByte をこえるデータ受信はできません。データ長：0 の指定も不可です。

使用例

```
int ret;
int npAdrsTbl[2] = { 2, -1 };
unsigned long len;

len = 65536;
ret = PciGpibExRecvFile(0, npAdrsTbl, &len, "test.dat");
```

デバイス番号 0 のデバイスに対して、データ受信の実行を行い、ファイル"test.dat"に受信したデータを書き込みます(コントローラモード時)。

```
ret = PciGpibExRecvFile(0,  NULL, &len, "test.dat");
```

デバイス番号 0 のデバイスに対して、データ受信の実行を行い、ファイル"test.dat"に受信したデータを書き込みます(非コントローラモード時)。

42. PciGpibExSetSignal

デバイスの各種バス・ステータスにおける事象変化の検出条件の設定を行います。

```
int PciGpibExSetSignal(
  int           nBoardNo,     // デバイス番号
  unsigned int  unSignal,     // 検出条件
  int           nDetect       // 検出フラグ
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

ulSignal

PciGpibExWaitSignal 関数での検出条件を以下のビット情報で指定します。

ビット	内容
31	IFC 受信検出(非コントローラ・イン・チャージ時のみ)
30	SRQ 受信検出(コントローラ・イン・チャージ時のみ)
29	シリアルポール終了検出(非コントローラ・イン・チャージ時のみ)
28	デバイストリガ受信検出(非コントローラ・イン・チャージ時のみ)
27	デバイスクリア受信検出(非コントローラ・イン・チャージ時のみ)
26	データ受信検出(非コントローラ・イン・チャージ時のみ)
25	リスナ指定検出(非コントローラ・イン・チャージ時のみ)
24	トーカ指定検出(非コントローラ・イン・チャージ時のみ)
23	入出力完了検出
22	リモート状態検出(非コントローラ・イン・チャージ時のみ)
21	ロックアウト状態検出(非コントローラ・イン・チャージ時のみ)
20	デリミタ検出(コントローラ・イン・チャージ時、シャドウハンドシェーク実行時のみ)
19	コントローラ・イン・チャージ (CIC) 検出(コントローラ・イン・チャージ時のみ)
18	ATN 信号アクティブ検出(コントローラ・イン・チャージ時のみ)
17～1	予約
0	1:タイムアウトあり、0:タイムアウト無し

1：検出許可
0：検出無効

複数の条件設定(OR 指定)ができます。未使用のビットには 0 を設定しておいてください。
タイムアウトを無しに設定した場合、PciGpibExWaitSignal 関数を呼び出し時、ステータスの状態を調べてから、すぐに処理を戻します。

nDetect

検出フラグを指定します。下記の値を指定してください。

値	内容
0	検出を無効にします。 指定された検出項目の状態はクリアされ、事象変化検出を行いません。
1	検出を有効にします。 指定された検出項目の設定を有効とし、事象変化検出を行います。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_PARAM_NO
・ERR_DRV_RETURN

備考

・弊社 GP-IB インタフェースはデータ受信後、RFD ホールドオフ状態となりますので、以降、データ受信検出ビット(bit26)が有効とならない場合があります。
非コントローラの状態でデータを受信する場合など、リスナ指定検出後にデータ受信関数を実行するようにして下さい。
・bit23(入出力完了)の検出条件はデータ転送(送信および受信)が完了した時点でセットされます。

使用例

```
int ret;

ret = PciGpibExSetSignal(0, 0x02000001, 1);
```

デバイス番号 0 のデバイスに対して、リスナ指定検出を許可、タイムアウト有りとして設定します。

43. PciGpibExWaitSignal

PciGpibExSetSignal 関数で設定した条件にてアクセスデバイスの各種バス・ステータスの事象変化を待ちます。

```
int PciGpibExWaitSignal(
  int            nBoardNo,    // デバイス番号
  unsigned int*  unpStatus    // 検出された条件を格納する変数へのポインタ
);
```

パラメータ

nBoardNo

デバイス番号(0～15)を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

unpStatus

検出された条件を格納する変数へのポインタを指定します。
検出条件は以下のビット情報で通知されます。

ビット	内容
31	IFC 受信検出(非コントローラ・イン・チャージ時のみ)
30	SRQ 受信検出(コントローラ・イン・チャージ時のみ)
29	シリアルポール終了検出(非コントローラ・イン・チャージ時のみ)
28	デバイストリガ受信検出(非コントローラ・イン・チャージ時のみ)
27	デバイスクリア受信検出(非コントローラ・イン・チャージ時のみ)
26	データ受信検出(非コントローラ・イン・チャージ時のみ)
25	リスナ指定検出(非コントローラ・イン・チャージ時のみ)
24	トーカ指定検出(非コントローラ・イン・チャージ時のみ)
23	入出力完了検出
22	リモート状態検出(非コントローラ・イン・チャージ時のみ)
21	ロックアウト状態検出(非コントローラ・イン・チャージ時のみ)
20	デリミタ検出(コントローラ・イン・チャージ時、シャドウハンドシェーク実行時のみ)
19	コントローラ・イン・チャージ (CIC) 検出(コントローラ・イン・チャージ時のみ)
18	ATN 信号アクティブ検出(コントローラ・イン・チャージ時のみ)
17～0	予約

1：検出
0：未検出

※各ビットにて該当モード以外の状態にて検出がされることがありますが、これは問題ありません。各ビットは該当状態のときのみチェックを行うようにしてください。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_NO_SET_SIGNAL
・ERR_WAIT_SIGNAL_TMO
・ERR_INVALID_EVENT
・ERR_WAIT_EVENT
・ERR_DEV_RETURN

備考

・弊社 GP-IB インタフェースはデータ受信後、RFD ホールドオフ状態となりますので、以降、データ受信検出ビット(bit26)が有効とならない場合があります。
非コントローラの状態でデータを受信する場合など、リスナ指定検出後にデータ受信関数を実行するようにして下さい。
・bit23(入出力完了)の検出条件はデータ転送(送信および受信)が完了した時点でセットされます。

使用例

```
int ret;
unsigned int status;

ret = PciGpibExSetSignal(0, 0x02000001, 1);
if(!ret){
    ret = PciGpibExWaitSignal(0, &status);
    if(!ret) printf("SIGNAL : %x¥n",status);
}
```

デバイス番号 0 のデバイスのリスナ指定検出を許可の事象変化検出を待ちます。

44. PciGpibExSetSrqEvent

指定番号のデバイスからの SRQ コールバックイベントを登録します。

```
int PciGpibExSetSrqEvent(
  int            nBoardNo,     // デバイス番号
  void*          pOnSrqProc,   // コールバック関数のアドレス
  usigned long   ulUser        // ユーザパラメータ
);
```

パラメータ

nBoardNo

SRQ コールバックイベントの登録を行うデバイスのデバイス番号（0～15）を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

pOnSrqProc

コールバック関数のアドレスを指定します。

ulUser

コールバック関数に渡す任意のデータを指定します。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 戻り値一覧』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_SET_EVENT
・ERR_DRV_RETURN

使用例

```
void pOnSrqProc(int nBoardNo, unsigned long ulUser){
    ...
}

int ret;

ret = PciGpibExSetSrqEvent( 0, pOnSrqProc, 0 );
```

デバイス番号 0 の GP-IB デバイスに対して SRQ コールバック関数 pOnSrqProc を登録します

参照

3.2 制御手順
3.2.1 コントローラとしての実行『6. シリアルポーリング』

45. PciGpibExWaitSrqEvent

指定番号のデバイスからの SRQ コールバックイベントを待ちます。

```
int PciGpibExWaitSrqEvent(
  int            nBoardNo,   // デバイス番号
  unsigned long  ulTimeOut   // タイムアウト時間
);
```

パラメータ

nBoardNo

SRQ コールバックイベントを待つデバイスのデバイス番号（0～15）を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

ulTimeOut

待ち時間を指定します。タイムアウト時間をミリ秒単位で指定します。
タイムアウト時間が経過すると、 関数は、オブジェクトの状態が非シグナル状態である場合でも、 制御を戻します。ulTimeout に 0 を指定した場合は、 オブジェクトの状態を調べてから、すぐに制御を戻します。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 戻り値一覧』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_INVALID_EVENT
・ERR_EVENT_WAIT_TIMEOUT
・ERR_WAIT_EVENT
・ERR_NOT_SET_EVENT
・ERR_DRV_RETURN

使用例

```
void pOnSrqProc(int nBoardNo, unsigned long ulUser){
    ...
}

int ret;

ret = PciGpibExSetSrqEvent( 0, pOnSrqProc, 0 );
if (!ret) {
    ret = PciGpibExWaitSrqEvent( 0, 5000 );
}
```

デバイス番号 0 の GP-IB デバイスの SRQ コールバックイベント待ちをタイムアウト 5 秒で設定します。

参照

3.2 制御手順
3.2.1 コントローラとしての実行『6. シリアルポーリング』

46. PciGpibExKillSrqEvent

指定番号のデバイスからの SRQ コールバックイベントの登録を解除します。

```
int PciGpibExKillSrqEvent(
  int   nBoardNo,   // デバイス番号
);
```

パラメータ

nBoardNo

nBoardNo SRQ コールバックイベントの登録解除を行うデバイスのデバイス番号（0～15）を指定します。
デバイスの RSW1 設定値となります。
※ソルコン CD 製品(PCI-432601)の場合、デバイス番号は 257 となります。

戻り値

この関数では下記の戻り値が返されます。詳細は『4.3 **戻り値一覧**』をご参照ください。
正常終了した場合は、NORMAL_EXIT が返されます。

・NORMAL_EXIT
・ERR_BRD_NO
・ERR_NOT_SET_EVENT
・ERR_DRV_RETURN

使用例

```
int ret;

ret = PciGpibExKillSrqEvent( 0 );
```

デバイス番号 0 の GP-IB デバイスの SRQ コールバックイベント登録を解除します。

参照

3.2 制御手順
3.2.1 コントローラとしての実行『6. シリアルポーリング』

47. pOnSrqEvent

PciGpibExSetSrqEvent 関数にて登録し、SRQ イベントが発生した時にコールされるコールバックルーチンです。

```
void pOnSrqEvent(
  int            nBoardNo,       // デバイス番号
  unsigned long  ulUser          // ユーザデータ
);
```

パラメータ

nBoardNo

イベントが発生したデバイス番号が格納されます。

ulUser

ユーザデータが格納されます。

戻り値

なし。

4.3 戻り値一覧

エラー識別子	値	意味
NOT_FOUND_LISTENER	11	リスナが 1 台も見つかりませんでした。
OK_SEND_STB	7	ステータス・バイトはコントローラへ通知済です。
NOT_EXEC_SPOLL	6	まだ、シリアル・ポールは行われていません。
NOT_ACTIVE_SRQ	5	SRQ 信号無効です。 ・PciGpibExCheckSrq 関数において、SRQ 割り込みを受け付けていないとき、この戻り値が通知されます。
ACTIVE_SRQ	4	SRQ 信号有効です。 ・PciGpibExCheckSrq 関数において、SRQ 割り込みを受け付けているとき、この戻り値が通知されます。
OK_EOI_DETECT	2	EOI を検出して終了しました。 ・データ受信時にデリミタとして EOI を検出して終了しました。
OK_RECV_DATA_CNT	1	指定された受信データ数に達して終了しました。 ・データ受信時に指定された受信データ数に達して終了しました。この状態ではデリミタの検出は行われていません。デリミタありの送受信処理を行っている場合には、引き続いてデータ受信処理を実行する必要があります。
NORMAL_EXIT	0	正常終了または指定されたデリミタを検出して終了しました。 ・データ受信時に指定された受信デリミタを検出して終了しました。 ・他の関数の場合には、処理を正常終了しました。
ERR_BRD_NO	-1	デバイス番号が違います。 ・コントロールパネルで指定されたデバイス番号以外のデバイスを指定しました。 ・すでに初期化済のデバイスに対して再初期化を行おうとしました。 ・または初期化が行われていないデバイスに対して制御を行おうとしました。 ・0～15 以外の番号を指定しました。
ERR_INP_PARAM	-2	入力パラメータに間違いがあります。 ・入力パラメータの値、範囲を確認してください。
ERR_PARAM_NO	-3	パラメータ番号に間違いがあります。 ・パラメータ番号の値、範囲を確認してください。
ERR_NOT_USE_NOSYS	-4	非コントローラ状態では使用できません。 ・スレーブモードにおいて、マスタモードでのみ使用可能な関数を呼び出しました。(デバイスが[CIC]状態で無い場合に、[CIC]状態でのみ使用可能な API を呼び出しました) ※CIC = Controller-in-Charge ： 当番コントローラ
ERR_NOT_USE_SYS	-5	コントローラ状態では使用できません。(コントローラ状態では使用できません) (デバイスが[CIC]状態の場合に、非[CIC]状態でのみ使用可能な API を呼び出しました) ※非 CIC = Not Controller-in-Charge ： 当番コントローラでは無い状態
ERR_SEND_BUS_CMD	-7	バスコマンドの送出に失敗しました。 ・コントローラ状態において、バスコマンドの送出に失敗しました。以下の要因が考えられます。 ・コントローラ状態において、PciGpibExSetIfc 関数を実行していない。 ・現在、他の機器および非同期入出力中により、GP-IB バスが占有状態である場合

		・非コントローラにおいて、バスコマンドを送出する関数を実行した場合 ・他のアプリケーションの負荷が重く、バスコマンド送出タイムアウトが発生した場合(この場合、初期値設定プログラムでタイムアウト時間を増やすようにしてください。)
ERR_NO_SET_SIGNAL	-8	検出する事象が設定されていません。 ・PciGpibExSetSignal 関数で事象変化検出設定を行わずに、PciGpibExWaitSignal 関数を実行しようとしました。
ERR_NO_ACTIVE_SRQ	-9	シリアル・ポーリングにも関わらず SRQ 送出元を検出できませんでした。
ERR_RECV_STB_TIMEOUT	-10	STB 受信時にタイムアウトが発生しました。 ・機器からのステータス・バイトが指定した時間内に受信できませんでした。 ・初期値設定プログラムの設定「STB 応答時間」を増やす必要があります。
ERR_DATA_RECV	-12	データ受信に失敗しました。 ・他の機器もしくは非同期転送中により、GP-IB バスが占有中である可能性があります。
ERR_DATA_SEND	-13	データ送信に失敗しました。 ・データ送信処理に失敗しました。以下の要因が考えられます。送信したデータが受信されなかった(この場合、ケーブル接続、全ての機器の電源が入っているか、ケーブル長は GP-IB の規格を守っているか等を確認してください。 ・他の機器もしくは非同期転送中により、GP-IB バスが占有中である場合
ERR_TRANSFER_TIMEOUT	-14	タイムアウトが発生しました。 ・送受信時に指定時間内にデータ転送が終了しませんでした。以下の要因が考えられます。 ・指定したデータ長に対するデータ転送時間がタイムアウト時間より長い場合(この場合は送受信タイムアウト時間を長くする必要があります。) ・送受信中に機器からの応答が何らかの要因でなくなった場合
ERR_WAIT_SIGNAL_TMO	-15	タイムアウトで終了しました。 ・PciGpibExWaitSignal 関数において、指定時間内に事象変化が検出できませんでした。 ・NRFD 信号ラインが無効になりませんでした。
ERR_IFC_TRANS_EXIT	-16	IFC 受信による強制終了。 ・送受信中に IFC を受信したため、送受信処理を強制終了しました。
ERR_NOT_CACS	-19	コントローラアクティブ状態に遷移できませんでした。
ERR_NOW_BUS_OCCUPATION	-20	現在、バスが占有状態となっています(非同期入出力中)。 ・送受信関数を実行しましたが、GP-IB バスが占有状態となっています。 ・非同期入出力の完了を待つようにしてください。
ERR_NOT_SET	-22	設定変更ができませんでした。 ・PciGpibExSetConfig 関数において、設定変更に失敗しました。以下の要因が考えられます。 ・非同期入出力中に PciGpibExSetConfig 関数を呼び出しました。
ERR_FILE_ACCESS	-30	ファイルをオープンできません、またはファイルの読み込み/書き込み中にエラーが発生しました。 以下の要因が考えられます。 ・存在しないファイルをオープンしようとしました。 ・ディスクの空き容量が足らない。 ・空きメモリ容量が極端に少ない(ハードディスクの空き容量に注意してください。)

		・共有ファイルをオープンしようとした、またはそのファイルが別のプロセスでオープンされている。
ERR_SET_EVENT	-40	コールバックイベントの登録に失敗しました。 ・コールバック関数の定義が正しくできていない。 ・指定したコールバック関数のアドレスが正しくない。
ERR_INVALID_EVENT	-42	イベントオブジェクトが無効です。
ERR_EVENT_WAIT_TIMEOUT	-43	イベント待ちでタイムアウトしました。
ERR_WAIT_EVENT	-44	イベント待ちでエラーが発生しました。
ERR_NOT_SET_EVENT	-45	イベントがまだ登録されていません。
ERR_NOT_SYS_CONTROLLER	-50	システムコントローラではありません。

ERR_DRV_RETURN	-100	カーネル(ドライバモジュール)からのエラーです。
		・プログラム上で perror 関数を実行し、詳細な内容を確認できます。
		例) char *pErr;
		ret = PciGpibExInitBoard(0, 0);
		if (ret == ERR_DRV_RETURN) {
		// perror 関数を使って、エラーメッセージを
		// 標準エラー出力に出力します
		perror(pErr);
		}

ERR_NOT_SUPPORT	-101	現在、使用しているデバイスではサポート外の関数です。
ERR_NOT_INIT_CALL	-994	デバイスのオープン時に何らかのエラーが発生しました。
ERR_NO_BOARD	-999	デバイスが存在しません。 以下の要因が考えられます。 ・指定したデバイス番号のデバイスが実装されていない ・デバイス自体に異常がある場合 ・自己診断プログラムを実行して、デバイスのチェックを行ってください。
ERR_USBIO_FAILED	-4000	USB デバイスの実行に失敗しました。 再起動を行なうか、DPG-0401 の「IfUsbDevicePowerCtl」関数を使用し、USB デバイスの電源を OFF→ON して下さい。 「IfUsbDevicePowerCtl」関数の使用方法は、DPG-0401 の Help を参照してください。
ERR_USB_TIMEOUT	-4001	USB デバイスとの通信がタイムアウトしました。 再起動を行なうか、DPG-0401 の「IfUsbDevicePowerCtl」関数を使用し、USB デバイスの電源を OFF→ON して下さい。 「IfUsbDevicePowerCtl」関数の使用方法は、DPG-0401 の Help を参照してください。
ERR_USBLIB_LOAD	-4002	ライブラリの読み込みに失敗しました。 デバイス番号が正しいことを確認してください。

4.4 テストドライバ

本ソフトウェアには、デバイスがなくてもドライバを動かすことができるテストドライバ機能がついております。
テストドライバを使用する場合は、libgpg4301.so の代わりに libgpg4301t.so をリンクし、コンパイルしてください。
テストドライバ機能を使用するプログラム test.c をコンパイルする場合のコンパイル例を示します。

```
#gcc -o test test.c -lpthread -lgpg4301t
```

各関数は下記のように動作します。

関数	動作
PciGpibExInitBoard	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。
PciGpibExFinishBoard	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。
PciGpibExGetInfo	デバイスタイプに 4300、RSW1 番号に指定したデバイス番号が返されます。その他の値は 0 を返します。
PciGpibExSetConfig	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。 ただし、自身のアドレスは 1 次アドレスのみサポートしています。
PciGpibExGetConfig	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。 初期設定プログラムで設定した値、または PciGpibExSetConfig 関数で指定した値が取得できます。
PciGpibExSetIfc	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。
PciGpibExSetRen	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。 ドライバ内部で REN ラインが ON になった状態になります。
PciGpibExResetRen	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。 ドライバ内部で REN ラインが OFF になった状態になります。
PciGpibExSetRemote	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。
PciGpibExExecTrigger	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。
PciGpibExExecDevClear	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。
PciGpibExExecSdc	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。
PciGpibExSetLocal	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。
PciGpibExSetLlo	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。
PciGpibExSetRwls	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。
PciGpibExExecPassCtrl	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。 この関数を実行すると、ドライバ内部でコントローラ・イン・チャージ状態が解除されます。自分自身へのパスコントロールはサポートしていません。 この関数実行後、再度コントローラ・イン・チャージの状態にするには、PciGpibExSetIfc 関数をコールしてください。
PciGpibExExecFindListener	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。 探索アドレスに、自身のアドレスを指定している場合、自身のアドレスがリスナアドレスとして返されます。
PciGpibExExecDevReset	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。
PciGpibExReSysCtrl	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。
PciGpibExGoStandby	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。
PciGpibExGoActCtrller	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。
PciGpibExExecSpoll	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。 PciGpibExSetSrq 関数で SRQ の送出を行っている場合、ステータスバイトを読み取ることができます。
PciGpibExCheckSrq	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。 PciGpibExSetSrq 関数で SRQ の送出を行っているかどうかを確認することができます。
PciGpibExClearSrq	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。 PciGpibExSetSrq 関数で SRQ の送出を行っている場合、SRQ 受信フラグのクリアを行います。
PciGpibExEnableSrq	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。 PciGpibExSetSrq 関数を実行した際に、SRQ を受信できるようになります。 PciGpibExInitBoard 関数をコールした直後は、SRQ を受信できる状態になっています。
PciGpibExDisableSrq	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。 PciGpibExSetSrq 関数を実行しても、SRQ を受信しなくなります。

PciGpibExExecPpoll	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。 PciGpibExSetIst 関数で設定した極性と、PciGpibExSetPp2 関数(または、PciGpibExCfgPpool 関数)で設定した極性が等しい場合、設定した DIO ラインが ON とみなされます。ただし、PciGpibExSetPp2 関数では、パラレルポールに応答しないようにも設定できます。
PciGpibExCfgPpoll	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。自身のアドレスを指定した場合、自身のパラレルポール・コンフィギュレーションを行います。
PciGpibExUnCfgPpoll	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。自身のアドレスを指定した場合、自身のパラレルポール・コンフィギュレーションの解除を行います。
PciGpibExWriteBusCmd	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。
PciGpibExSetSignal	サポートしていません。
PciGpibExWaitSignal	サポートしていません。
PciGpibExGetStatus	サポートしていません。
PciGpibExClrStatus	サポートしていません。
PciGpibExGetBusLine	SRQ ラインと REN ラインのみ確認できます。
PciGpibExSendData	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。 下記のデータのみ意味を持ちます。 “*IDN?” “MEASure:VOLTage:DC?”
PciGpibExRecvData	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。 PciGpibExSendData 関数で、アドレスを指定していない場合か、自身のアドレスを指定している場合のみ、PciGpibExSendData 関数で送信されたデータにより、下記のデータを返します。 “*IDN?” → “INTERFACE:GPG-4301/4304” “MEASure:VOLTage:DC?” → ランダムな仮想電圧値
PciGpibExSendFile	サポートしていません。
PciGpibExRecvFile	サポートしていません。
PciGpibExCheckStb	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。 PciGpibExSetSrq 関数実行後、PciGpibExExecSpoll 関数を実行しているかどうかを確認することができます。
PciGpibExSetSrq	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。 自身に SRQ を送出することになります。
PciGpibExSetIst	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。
PciGpibExSetPp2	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。
PciGpibExSetSrqEvent	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。 PciGpibExSetSrq 関数実行時、設定したコールバック関数が呼び出されます。
PciGpibExWaitSrqEvent	サポートしていません。
PciGpibExKillSrqEvent	正常なパラメータで関数をコールすると NORMAL_EXIT を返します。

第5章　サンプルプログラム

サンプルプログラムのディレクトリのメイクファイルを実行してサンプルプログラムのコンパイルを行ってください。

```
#make
```

コンパイル後、各実行形式ファイルにて、サンプルプログラムを実行してください。

5.1　advr6451

内容	マルチメータから 10 回電圧取得（SRQ 未使用）
対象機器	マルチメータ（アドバンテスト社 R6451）
説明	マルチメータから 10 回ほど電圧値を取得し、表示します。 R6451 の GP-IB アドレスは 8 として作成されています。 電源を入れた後、[I/F]キーで、GP-IB アドレスを必ず確認して下さい

5.2　async

内容	マルチメータから 10 回電圧取得（非同期処理）
対象機器	マルチメータ（アジレントテクノロジー社(旧 HP 社) HP34401A）
説明	マルチメータから 10 回ほど電圧値を取得し、表示します。 HP34401A の GP-IB アドレスは 22 として作成されています。 非同期処理なので受信関数実行後、PciGpibExGetStatus 関数で受信完了するまで待ちます。

5.3　buscmd

内容	外部機器に対してデバイストリガ、デバイスクリアのバスコマンドを発行します。
対象機器	GP-IB アドレスが 22 の外部機器
説明	機器アドレス 22 の外部機器に対してデバイストリガ、デバイスクリアのバスコマンドを発行します。

5.4　creceive

内容	データを受信します。
対象機器	GP-IB アドレスが 2 の外部機器
説明	外部機器からデータを 1 回受信します。

5.5　csend

内容	データを送信します。
対象機器	GP-IB アドレスが 2 の外部機器
説明	外部機器に”*RST”コマンドを送信します。

5.6　hp3458a

内容	マルチメータから 10 回電圧取得（SRQ 未使用）
対象機器	マルチメータ（アジレントテクノロジー社(旧 HP 社) HP 3458A）
説明	マルチメータから 10 回ほど電圧値を取得し、表示します。 HP3458A の GP-IB アドレスは 22 として作成されています。

5.7 hp34401a

内容	マルチメータから 10 回電圧取得（SRQ 未使用）
対象機器	マルチメータ（アジレントテクノロジー社(旧 HP 社) HP 34401A）
説明	マルチメータから 10 回ほど電圧値を取得し、表示します。 HP34401A の GP-IB アドレスは 22 として作成されています。

5.8 hp54645d

内容	オシロスコープからワンショットサンプリングし、データを表示します。
対象機器	オシロスコープ（アジレントテクノロジー社(旧 HP 社) HP 54645D）
説明	オシロスコープからワンショットサンプリングし、100 ポイントのデータを取得し、表示します。 HP54645D の GP-IB アドレスは 7 として作成されています。

5.9 lreceive

内容	非コントローラ状態で外部からのデータ受信を行います。
対象機器	システムコントローラ状態の機器
説明	非コントローラ状態に変更し、データを受信します。 リスナに指定されるまで処理が戻りません。

5.10 spoll

内容	シリアルポールを実行します。
対象機器	GP-IB アドレスが 2 の外部機器
説明	SRQ 信号が有効になるまで待ち、シリアルポールを行います。

5.11 srq

内容	SRQ 信号を発行します。
対象機器	システムコントローラ状態の機器
説明	非コントローラ状態に変更し、SRQ 信号を発行します。 SRQ 信号を外部機器が受け取るまで処理が戻りません。

5.12 sync_no

内容	非コントローラ状態で外部からのデータ受信を行います。
対象機器	システムコントローラ状態の機器
説明	システムコントローラからの REN 信号、送信データの受信を行います。 システムコントローラからの REN 信号を受け取るか、タイムアウトが起きるまで処理が戻りません。

5.13 tsend

内容	非コントローラ状態で外部へデータ送信を行います。
対象機器	システムコントローラ状態の機器
説明	非コントローラ状態に変更し、データを送信します。 トーカに指定されるまで処理が戻りません。

5.14 yk7555

内容	マルチメータから 10 回電圧取得 (SRQ 未使用)。
対象機器	マルチメータ(横河電機社 Model 7555)
説明	マルチメータから 10 回ほど電圧値を取得し、表示します。 Model7555 の GP-IB アドレスは 1 として作成されています。 このプログラムは、シングルモードでサンプリングしています。

5.15 yk7561

内容	マルチメータから 10 回電圧取得 (SRQ 未使用)。
対象機器	マルチメータ(横河電機社 Model 7561)
説明	マルチメータから 10 回ほど電圧値を取得し、表示します。 Model7561 の GP-IB アドレスは 1 として作成されています。 このプログラムは、シングルモードでサンプリングしています。

第6章　ユーティリティ

6.1 初期値設定プログラム

初期設定プログラムは、ドライバソフトウェアが GP-IB デバイスを制御する際の各種パラメータの設定を行います。
また、このプログラムではアプリケーションがドライバモジュールを制御するために必要なデバイスノードファイルの作成も行っています。

1. 初期設定プログラムを実行する前にドライバモジュールを組み込みます。
   ドライバモジュールを組み込まないで初期設定プログラムを実行するとエラー(:-999)が返されます。
2. /usr/bin ディレクトリの「cgpibconf」実行ファイルを実行します。
3. デバイスの種類を選択します。
4. 次にデバイスの型式番号を入力します。型式番号を選択してください。
5. 次にデバイスロータリスイッチ(CardBus 製品の場合はCardBusID)設定値を10進数で入力してください。
6. 入力したデバイスのドライバモジュールのメジャー番号、マイナー番号を表示し、デバイスノードファイルを作成します。
7. 各種のパラメータ設定を保存するファイルを作成します。キーボードの「n」キーを押してください。
8. 設定ファイルが存在した場合は、その内容を表示します。
9. 現在の設定を変更するため 1～12 の項目を選択します。
   1：初期設定プログラムのデフォルトの設定を行います。
   2：コントローラモードの設定を行います。
   　コントローラモードには「System Controller」、「Not System Controller」のどちらかのモード設定を行います。(default：System Controller)
   3：1 次アドレスの設定を行います。00h～1Eh の範囲の 16 進数で設定します。(default：00h)
   4：2 次アドレスの設定を行います。60h～7Eh の範囲の 16 進数で設定します。使用しない場合は ffh を設定してください。(default：FFh)
   5：データ送受信時のタイムアウトを設定します。1～65535 の範囲で設定します。単位は 100 ミリ秒です。(default：1000　(100 秒))
   6：事象変化のタイムアウトを設定します。1～65535 の範囲で設定します。単位は 100 ミリ秒です。(default：1000　(100 秒))
   7：コマンド送信時のタイムアウトを設定します。1～65535 の範囲で設定します。単位は 100 ミリ秒です。(default：1000　(100 秒)
   8：ハンドシェイクタイミングの設定を行います。(Normal－正常：2us High speed－高速：500ns Very high speed － 超高速：350ns)(default：Very high speed)
   9：送信デリミタを設定します。(default：CRLF+EOI)
   10：受信デリミタを設定します。(default：CRLF+EOI)
   11：シリアルポーリング時のステータスバイト応答タイムアウト時間を設定します。
   1～65535 の範囲で設定します。単位は 100 ミリ秒です。(default：10　(1 秒))
   12：パラレルポーリング時の実行タイムアウト時間を設定します。
   　2μ秒もしくは 10μ秒で設定します。(default：2μ秒)
   終了する時は 99 を入力します。

10. 設定したパラメータを表示します。その下にデバイスのリソース情報なども表示します。

6.2 自己診断プログラム

自己診断プログラムは、動作不具合時の原因を判断するためのものです。
自己診断プログラムにより、以下のことが確認できます。
・デバイスドライバのインストールの有無
・コンピュータと GP-IB デバイスとのインタフェース動作(デバイスのレジスタアクセス、PCI バスマスタアクセス、割り込み信号)

自己診断プログラムが正常に動作すれば、システムの不具合の原因を以下の要因に絞ることができます。
・デバイスの GP-IB インタフェース部分の故障
・GP-IB ケーブルの破損
・接続している GP-IB 機器の動作不具合
・アプリケーションプログラム、ドライバモジュールの不具合

必要な機材
・GP-IB デバイス
・自己診断プログラム

※ソルコン CD 製品(PCI-432601)では、使用できません。

【操作手順】

1. 自己診断プログラムを実行する前にドライバモジュールを組み込むます。
2. /usr/bin ディレクトリの「cgpibdiag」実行ファイルを実行します。
3. 自己診断を行いたいデバイスのデバイス番号を入力してください。
   対象が「PCI-432101」の場合は「0」を指定してください。
   デバイス番号が不明の場合は、/proc ファイルシステムを参照する事により確認します。

   ```
   #cat /proc/driver/gpib/cp4301
   cp4301 info:1.0
   0: PCI/LPC-432101(bid=0)  bid=0 がデバイス番号になります。
   ```

4. 検査を行うデバイスを選択すると、デバイス型式、RSW1 番号、レジスタベースアドレス、IRQ 番号が表示されます。
5. 自動的に各項目の診断が行われ、結果が表示されます。
6. 自己診断結果はファイルに保存することができます。100 文字以内でファイル名を指定してください。診断結果が NG の場合は、診断結果をテキスト形式で保存してください。その診断結果とともに弊社 お客様相談センタまでお問い合わせください。

第7章　重要な情報

保証の内容と制限

弊社は本ドキュメントに含まれるソースプログラムの実行が中断しないこと、またはその実行に誤りが無いことを保証していません。
本製品の品質や使用に起因する、性能に起因するいかなるリスクも使用者が負うものとします。

弊社はドキュメント内の情報の正確さに万全を期しています。万一、誤記または誤植などがあった場合、弊社は予告無く改訂する場合があります。ドキュメントまたはドキュメント内の情報に起因するいかなる損害に対しても弊社は責任を負いません。
ドキュメント内の図や表は説明のためであり、ユーザ個別の応用事例により変化する場合があります。

著作権、知的所有権

弊社は本製品に含まれるおよび本製品に対する権利や知的所有権を保持しています。
本製品はコンピュータ ソフトウェア、映像/音声(例えば図、文章、写真など)を含んでいます。

医療機器/器具への適用における注意

弊社の製品は人命に関わるような状況下で使用される機器に用いられる事を目的として設計、製造された物では有りません。
弊社の製品は人体の検査などに使用するに適する信頼性を確保する事を意図された部品や検査機器と共に設計された物では有りません。
医療機器、治療器具などの本製品の適用により、製品の故障、ユーザ、設計者の過失などにより、損傷/損害を引き起こす場合が有ります。

複製の禁止

弊社の許可なく、本ドキュメントの全て、または一部に関わらず、複製、改変などを行うことはできません。

責任の制限

弊社は、弊社または再販売者の予見の有無にかかわらず発生したいかなる特別損害、偶発的損害、間接的な損害、重大な損害について、責任を負いません。

本製品(ハードウェア，ソフトウェア)のシステム組み込み、使用、ならびに本製品から得られる結果に関する一切のリスクについては、本製品の使用者に帰属するものとします。

本製品に含まれる不都合、あるいは本製品の供給(納期遅延)、性能もしくは使用に起因する付帯的損害もしくは間接的損害に対して、弊社に全面的に責がある場合でも、弊社はその製品に対する改良(有償サービスの利用)、代品交換までとし、製品の予防交換並びに、代金減額等、金銭面での賠償の責任は負わないものとします。

日本国内仕様です。

商標/登録商標

本書に掲載されている会社名、製品名は、それぞれ各社の商標または登録商標です。