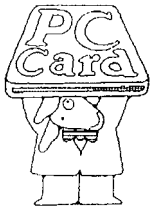
AXP-09-050322



PLUG MAGIC シリーズ GPIB アダプタ

AXP-GP02

取扱説明書

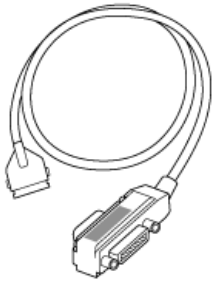
株式会社 アドテック システム サイエンス

すべて揃っていますか

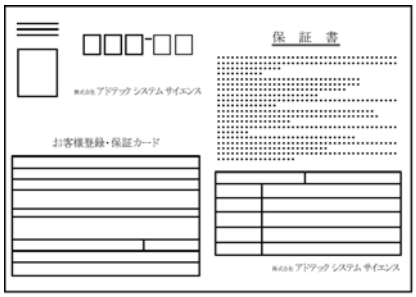
本体と次の付属品がすべて揃っているか確認してください。
万一、不足の品がありましたらお手数でもお買い上げの販売店もしくは当社までご連絡ください。

同梱品

□AXP-GP02 カード本体



□付属 GPIB ケーブル



□お客様登録カード･保証書

□サポートディスク

《おことわり》

(1) 本書の内容の一部または全部を無断で記載することは、禁止されております。
(2) 本製品の仕様および本書の内容は、将来予告なく変更することがあります。
(3) 本書の内容につきましては、万全を期して作成いたしましたが、万一ご不審な点やお気付きの点がございましたら、当社までご連絡ください。
(4) 本製品は、出荷の際十分な検査を行い万全を期しておりますが、万一ご使用中にご不審な点がございましたら、当社までご連絡ください。
(5) 本製品につきましては、保証書に明記された条件における保証期間中の修理をもって、当社の唯一の責任とさせていただきます。本製品を運用した結果の影響につきましては、(3)(4)項にかかわらず責任を負いかねます。
(6) 本文中にある会社名、商品名は各社の商標または登録商標です。

目次

はじめに

この度は、PULG MAGIC シリーズ GPIB アダプタ AXP-GP02 をお買い求めいただき、誠にありがとうございます。
本製品の性能を十分ご活用いただくため、本書を熟読され、正しい使用法で末永くご愛用いただきますようお願い申し上げます。

ー動作環境ー

■本製品は以下の動作環境でお使いください。

対応パソコン　PC Card Standard TYPEⅡ規格の PC カードスロットを持ったパソコン
DOS/V パソコン、NEC PC98 シリーズのいずれの機種も動作可能です。

ご使用前にそのパソコンにPCカードスロットがあるかをお確かめください。

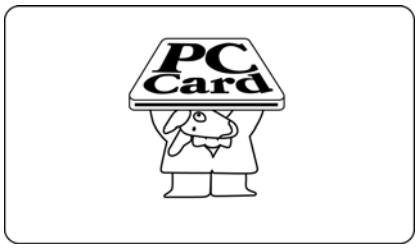
ー特　長ー

- 本製品AXP-GP02は、PC Card Standard TYPEⅡ規格に準拠したGPIBアダプタです。PCカードスロット装備の各種パソコンへ装着することにより、簡単にGPIBを利用できます。
- Windows 95/98/Me/2000/XPに対応。VisualC++、VisualBasic、Delphiに対応したサンプルプログラムにより、効率的にアプリケーションプログラムの開発をおこなうことができます。

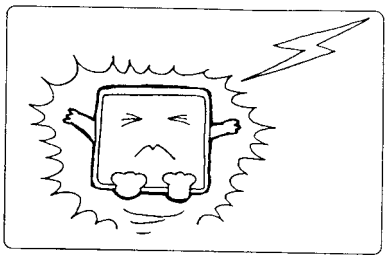
取り扱い上の注意

本製品は非常に精密な電子機器です。お取り扱いに際しては、次の事項を守ってご使用ください。

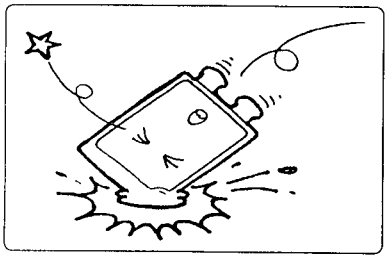
■ このカードは PC Card Standard 対応カードスロット以外では使用できません。



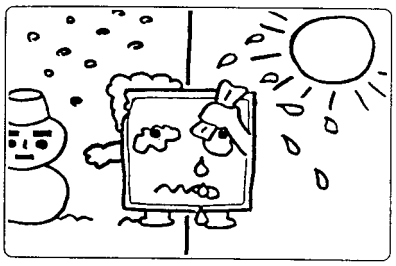
■ 静電気に弱いので、静電気の起きやすい場所等に放置しないでください。



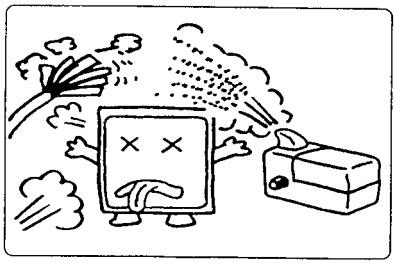
■ 本体に衝撃をあたえたり、落としたりしないでください



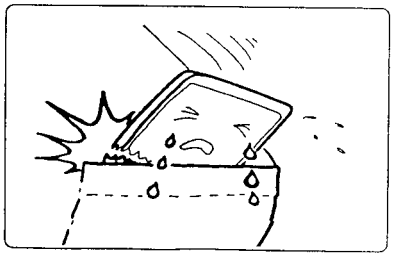
■ 直射日光の当たる場所や低温な場所での使用や保管は避けてください。



■ ほこりや湿気の多いところでの使用や保管はさけてください。



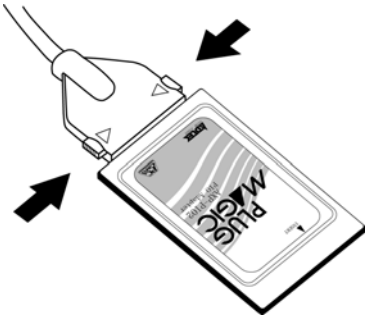
■ 折り曲げ厳禁。破損してしまったカードは修理できません。



1. カードの取り付け

1-1. ケーブルをカードに取り付ける

カードの上面（PLUG MAGICの文字が見える側）と接続ケーブルのカード側コネクタの上面（図のように△のマークのある側）を合わせ、矢印の方向に静かに差し込みます。ロッキング金具の「カチッ」という音がすることを確認します。

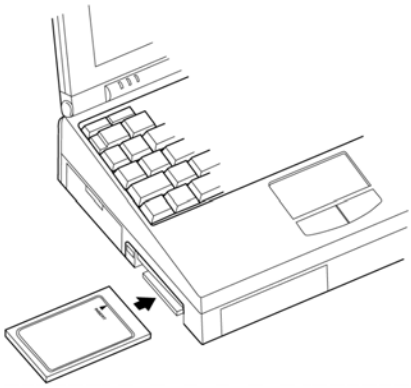


接続ケーブルを無理に曲げたり、コネクタとカードとの接続部に無理な力を加えると動作不良や故障の原因になります。

1-2. パソコンへカードを取り付ける

パソコンのカード・スロットにカードを差し込みます。

カードのインターフェース・コネクタ側をパソコンのPCカードスロットに静かに差し込みます。



PCカードTYPE Ⅰスロットには入りません。

PCカードは、誤挿入防止構造になっていますが、無理に差し込もうとすると、パソコンのPCカードスロットやPCカード本体の故障の原因となります。

パソコンの機種によっては、PCカードの裏面を上にし、実装するタイプがあります。ご注意ください。

■カードの取り出し方

PCカードをパソコンから取り出す時は、パソコンのカード・イジェクト・ボタンを押します。カードが少し飛び出します。飛び出した部分を持ち静かに引き抜きます。

ご使用ノートパソコンの取扱説明書カードスロットの項もお読みください。

■電源の ON/OFF 順序

電源を投入するときは、必ずパソコンの電源をONにしてから接続している装置の電源をONにしてください。
また、電源を切る時は、PCカードと接続している装置の電源をOFFにしからパソコンの電源をOFFにします。

2. ソフトウェアの組み込み

本製品をご使用になる前に、ソフトウェアの組み込み等の準備が必要です。

ソフトウェアは、サポートソフト（添付サポートディスクまたは弊社ホームページ http://www.adtek.co.jp/ からダウンロード）に収められています。
ここでは、サポートソフトを、フロッピーディスク（以下「サポートディスク」）にコピーして使用する場合について示しています。CD-R 等他のメディアをご使用の場合は、適宜読み替えて作業を進めてください。

2-1 カードの登録

以下は、初めて本カードをご使用いただくとき、もしくは登録の削除した場合の設定です。この設定は、１度おこなうと、次回から登録の削除をおこなわない限り有効です。

2-1-1 Windows95/98/Me の場合

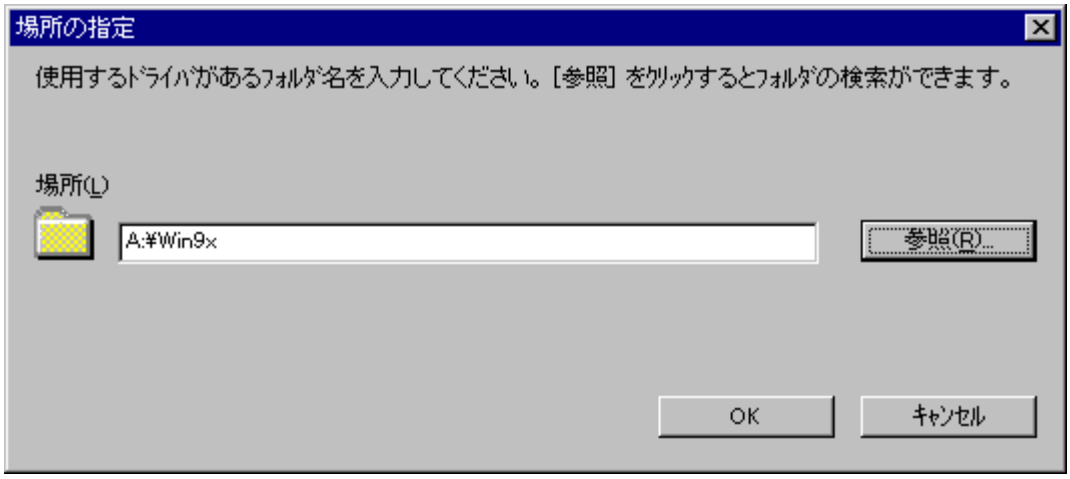
ここではAXP-GP02をWindows95で使用する場合を示します。Windows98/Meで使用する場合は、画面の指示に従って適宜読み替えてください。

Ⅰ. Windows95が起動したら、PCカードスロットにAXP-GP02を挿入してください。

Ⅱ. 挿入後、デバイスドライバウィザードが自動的に始まり以下の画面が表示されますので、［次へ］をクリックしてください。

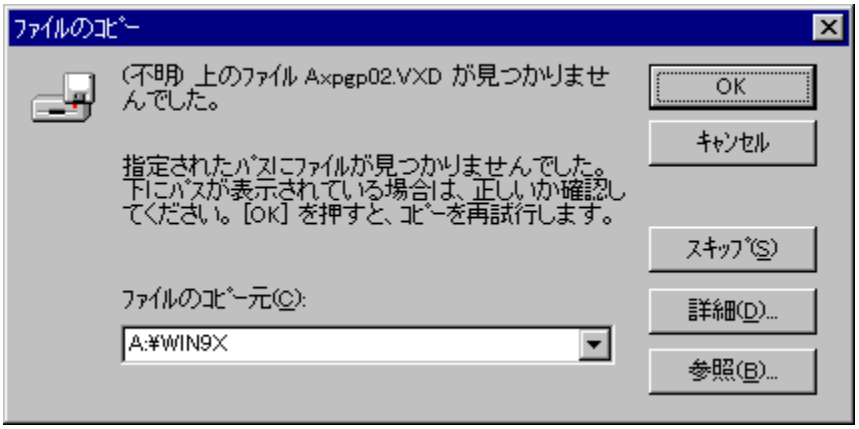
Ⅲ.　以下の画面が表示されますので、［場所の指定］をクリックしてください。

Ⅳ.　以下の画面が表示されますので、サポートディスクを指定ドライブに挿入し、Aドライブの場合は次のように入力し、［OK］をクリックしてください。



V.　AXP-GP02のドライバが見つかったことを確認して、［完了］をクリックしてください。

VI.　次の画面が表示されたら、以下のように入力し、［OK］をクリックしてください。



VII.　以上でインストール作業は終了です。

インストール作業が終了しますと、通常PCカードが挿入されたことを示す認識音（BEEP音）がでるとともに、タスクバー内のトレーに［PCカード（PCMCIA）の状態］のアイコンが表示されます。
また、カードを抜去すると、再び認識音とともにアイコンが消えることを確認してください。

■トラブルシューティング

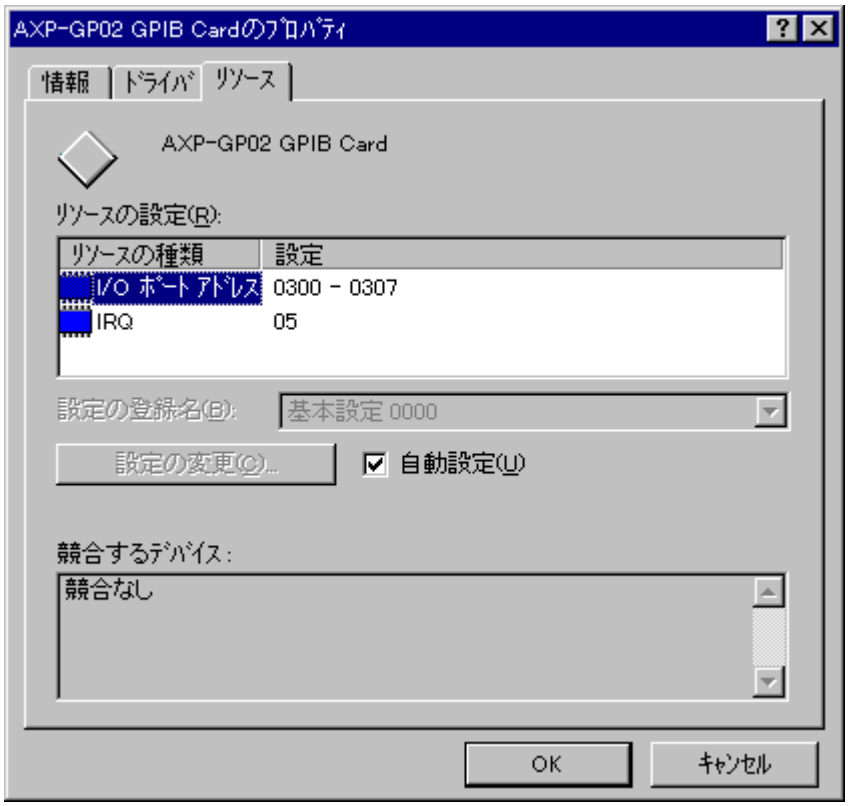
もしもアプリケーションが正常に動作しなかった場合、以下の手順でPCカードの状態を確認してください。

１：　［マイコンピュータ］アイコン、［コントロールパネル］アイコン、［システム］アイコンの順で、続けてダブルクリックします。

２：　［システムのプロパティ］が表示されます。以下は正常な場合です。

また、再度ハードウェアの検出によるドライバのインストールを行いたい場合は、図の中で［AXP-GP02 GPIB Card］を選択し、［削除(E)］をクリックします（詳細はWindows95の取扱説明書を参照してください）。

正常に設定されなかった場合、［AXP-GP02 GPIB Card］に［！］マークが表示されます。
この状態の原因は、おもにリソース（I/Oアドレスまたは割り込み）の競合が考えられます。
［AXP-GP02 GPIB Card］をダブルクリックし、詳細を確認してください。
もし、競合するデバイスがある場合、I/OアドレスとIRQの設定を変更して競合がなくなるようにしてください。



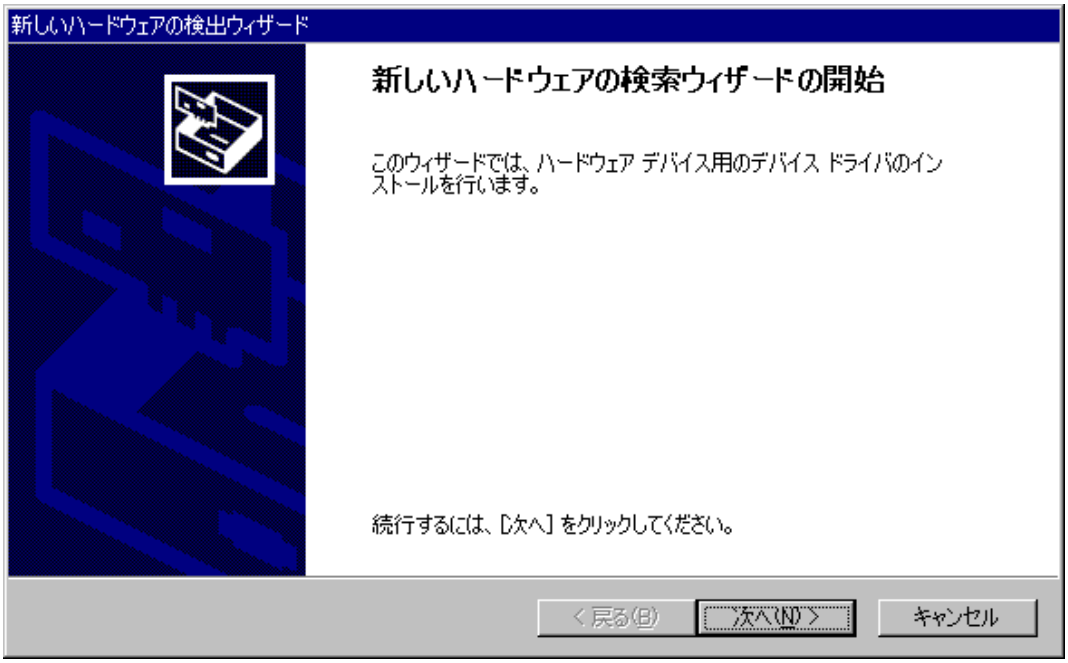
※ Windowsのバージョンによってインストールの画面が異なる場合があります。
その場合は画面の指示に従ってください。

2-1-2 Windows2000/XP の場合

ここではAXP-GP02をWindows2000で使用する場合を示します。WindowsXPで使用する場合は、画面の指示に従って適宜読み替えてください。

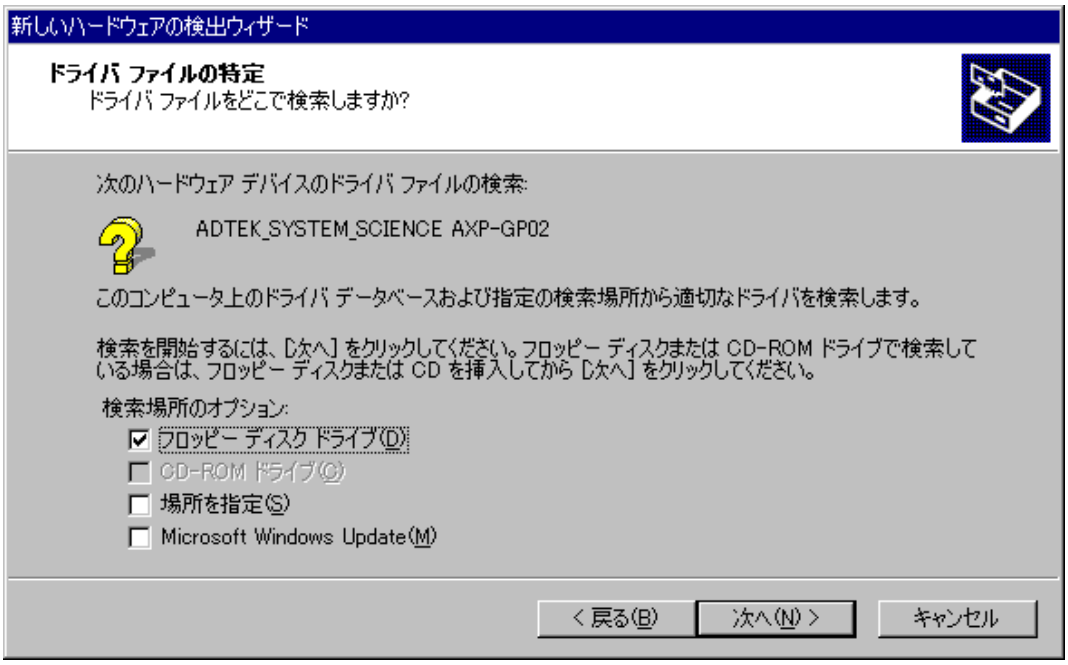
Ⅰ. Windows2000が起動したら「Administrator」でログオンし、PCカードスロットにAXP-GP02を挿入してください。

Ⅱ. 挿入後、新しいハードウェアの検出ウィザードが自動的に始まり以下の画面が表示されますので、［次へ］をクリックしてください。

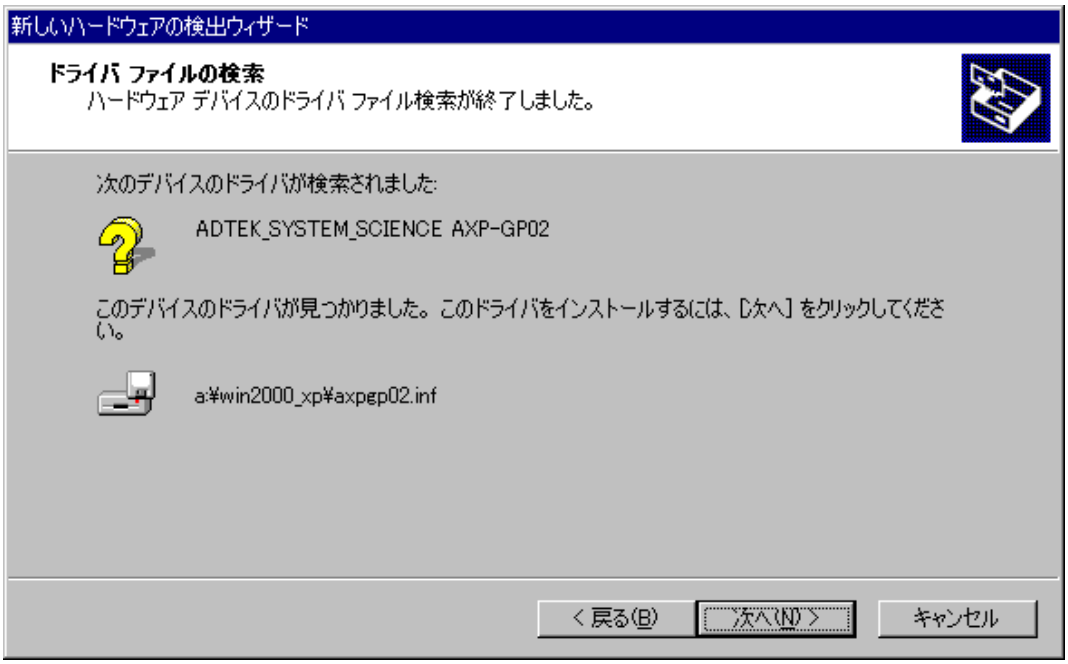


Ⅲ.　以下の画面が表示されますので、［デバイスに最適なドライバを検索する（推奨）(S)］を選択し、［次へ］をクリックしてください。

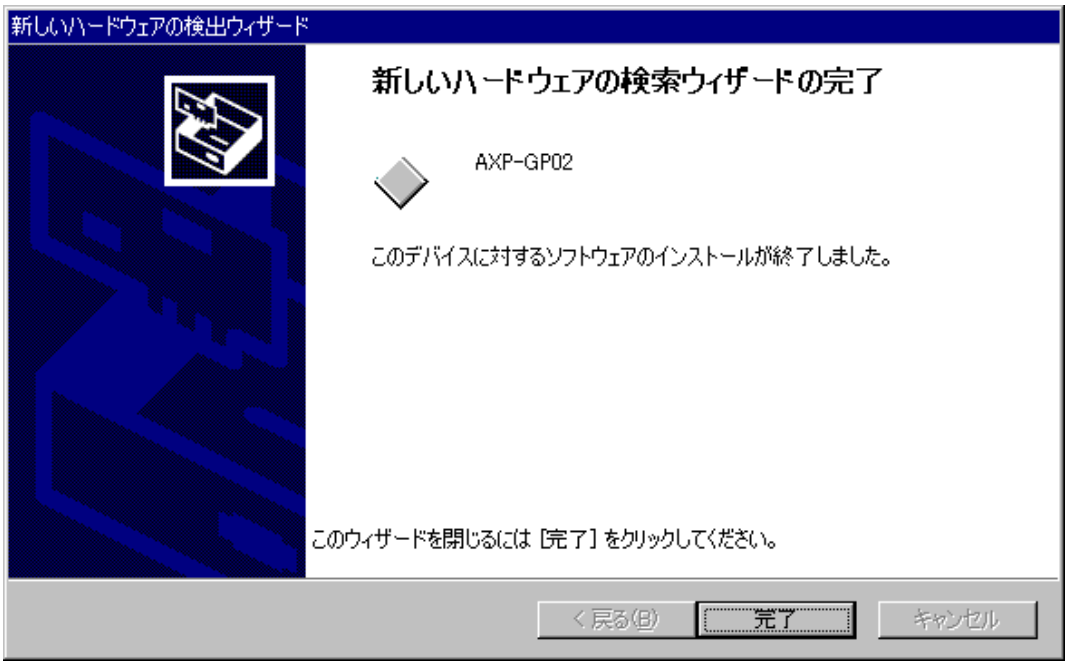
Ⅳ.　以下の画面が表示されますので、サポートディスクを指定ドライブに挿入し、［フロッピーディスク ドライブ(D)］を選択し、［次へ］をクリックしてください。



V.　AXP-GP02のドライバが見つかったことを確認して、［次へ］をクリックしてください。



VI.　［完了］をクリックしてください。



VII.　以上でインストール作業は終了です。

インストール作業が終了しますと、通常PCカードが挿入されたことを示す認識音（BEEP音）がでるとともに、タスクバー内のトレーに［ハードウェアの取り外しまたは取り出し］のアイコンが表示されます。
また、カードを抜去すると、再び認識音とともにアイコンが消えることを確認してください。

2-2 ライブラリのインストール

カードの登録、登録の確認が終了したら、アプリケーションプログラム開発に必要なファイルをインストールします。

・ＤＬＬ

Windows95/98/Me/2000/XP用ドライバを制御するDLLファイルは自動的にコピーされませんので、お使いの環境に合わせてファイルをコピーしてください。

※ Windows95/98/Meをお使いの場合はWin9xフォルダ内のDLLファイルを、Windows2000/XPをお使いの場合はWin2000_XPフォルダ内のDLLファイルをコピーしてください。

一般的には、「アプリケーションと同じディレクトリ」か、
Windows 95/98/Meでは「%WinDir%¥System　（例　C:¥Windows¥System）」、
Windows2000/XPでは　「%WinDir%¥System32（例　C:¥Winnt¥System32）」となります。

・動作チェックプログラム

axpgp02.exeは製品の動作確認用アプリケーションです。

カード自体の動作検証及びドライバ・DLL が正しくセットアップされ、正常に機能しているかどうかをご確認ください。

・サンプル

Sampleディレクトリ内にはVC/VB/Delphi用のサンプルソースがございます。
DLL内の関数のインターフェースは各関数のヘッダー、注釈及びAPIリファレンスを参照してください。

＜ラッパー関数＞

axpgp02w.* のように「w」が付いているファイルには、DLL内の関数を簡単にコールするためのラッパー（Wrapper）関数が定義されています（DLLのロード／アンロード関数も含まれています）。

※ また、VCにはスタティックリンクのためのインポートライブラリも含まれます。

＜ラッパー関数使用例＞

axpgp02s.* のように「s」が付いているファイルには、ラッパー関数を使用してデバイスを制御する例が記されています。

当サンプルソースは各開発環境にてすぐに実行してお試しいただけます。詳しくは各ディレクトリ内のbuildxx.txtの例をご覧ください。

3. アプリケーションの作成

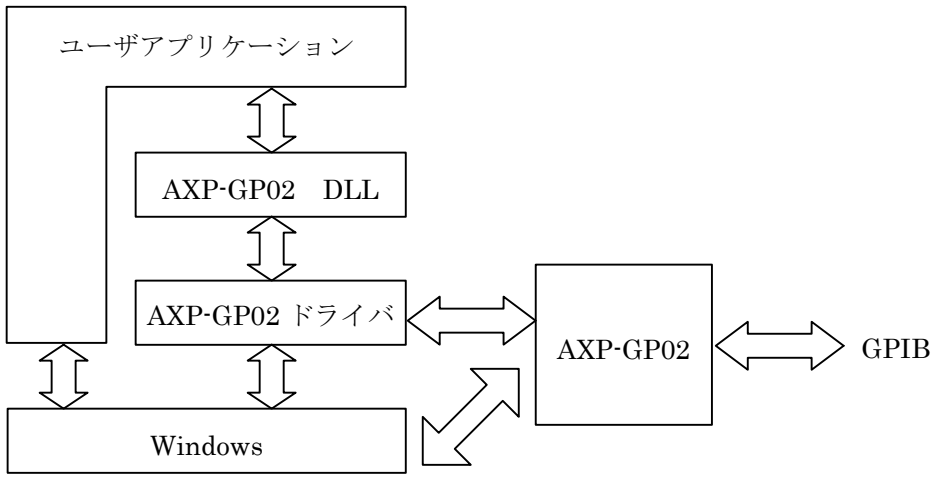
アプリケーションプログラムの作成方法を解説します。
AXP-GP02 は、Windows95/98/Me 用デバイスドライバ（VxD）、Windows2000/XP 用デバイスドライバ（SYS）、専用ライブラリ（DLL）等が付属しています。
これらのファイルは、サポートソフトに収めてあります。デバイスドライバ以外のファイルは、作業環境に合わせてコピーしてご使用ください。
また、サポートソフトには、デバイスドライバのアクセス方法や、実際に動作するサンプルプログラムのソースコードも含まれています。
アプリケーションプログラム作成の際に、参考にしてください。

3-1 サポートソフトの内容

```
¥
├ Win9x                    Windows95/98/Me用32bitドライバ
│  ├ axpgp02.inf              インストールファイル
│  ├ axpgp02.vxd              ドライバ
│  └ axpgp02.dll              DLL
├ Win2000_XP               Windows2000/XP用32bitドライバ
│  ├ axpgp02.inf              インストールファイル
│  ├ axpgp02.sys              ドライバ
│  └ axpgp02.dll              DLL
├ Sample                   Windows95/98/Me/2000/XP用サンプルソース
│  ├ Vc                       VCサンプルソース
│  │  ├ gp02def.h                パラメータ定義ファイル
│  │  ├ axpgp02.h                DLL定義ヘッダ
│  │  ├ axpgp02w.h               ラッパー関数ヘッダ
│  │  ├ axpgp02w.c               ラッパー関数
│  │  ├ axpgp02s.c               ラッパー関数使用例
│  │  ├ buildvc.txt              サンプルソース構築例
│  │  └ Lib
│  │     ├ Win9x
│  │     │  └ axpgp02.lib       インポートライブラリ
│  │     └ Win2000_XP
│  │        └ axpgp02.lib       インポートライブラリ
│  ├ Vb                       VBサンプルソース
│  │  ├ axpgp02d.bas             DLL定義
│  │  ├ axpgp02w.bas             ラッパー関数
│  │  ├ axpgp02s.frm             ラッパー関数使用例
│  │  └ buildvb.txt              サンプルソース構築例
│  └ Delphi                   Delphiサンプルソース
│     ├ axpgp02w.pas            DLL定義/ラッパー関数
│     ├ axpgp02s.pas            ラッパー関数使用例
│     ├ axpgp02s.dfm            フォームファイル
│     └ builddp.txt             サンプルソース構築例
├ axpgp02.hlp/cnt           ヘルプファイル
├ axpgp02.exe               動作チェックアプリケーション
├ axpgp02ap.txt             API仕様書
└ readme.txt                リードミー
```

3-2 ソフトウェア構成

AXP-GP02のソフトウェア構成は以下のようになっています。



・ドライバ

axpgp02.vxd／axpgp02.sys

AXP-GP02を制御するWindow95/98/Me用、Windows2000/XP用デバイスドライバです。
WindowsでGPIBを使用するには、このドライバが組み込まれていなければなりません。
アプリケーションプログラムは、DLLを介してドライバの機能を利用することができます（アプリケーションからドライバに直接アクセスすることはできません）。
このドライバは、WindowsがAXP-GP02がコンピュータのPCカードスロットに挿入されたことを認識した時点でメモリ上にロードされ､AXP-GP02がPCカードスロットから取り除かれた時点で、アンロードされます。

・ＤＬＬ（ダイナミックリンクライブラリ）

axpgp02.dll

GPIBアプリケーションを作成するためのライブラリです。
このDLLを介して、ドライバにアクセスします。

・インポートライブラリ

axpgp02.lib

C/C++言語でGPIBアプリケーションを作成し、DLLを呼び出すためにリンクするライブラリファイルです。
このファイルを通常のライブラリのようにリンクすることによって、DLLのAPIを呼び出すことができます。インポートライブラリの詳細については、Windowsのプログラミング解説書等をご覧ください。

・ユーティリティ

axpgp02.exe

GPIBの動作チェックを行うユーティリティです。
このプログラムを使用して、AXP-GP02の動作確認が行えます（要 axpgp02.dll）。

・ヘッダーファイル

axpgp02w.h、axpgp02.h、gp02def.h（C/C++言語用）
axpgp02w.bas、axpgp02d.bas（VisualBasic用）
axpgp02w.pas（Delphi用）

C/C++言語、VisualBasic、Delphiでプログラミングするときに必要なファイルです。
C/C++言語の場合はソースファイルの先頭でインクルードします。
VisualBasic、Delphiの場合はアプリケーションプロジェクトにファイルを追加します。

・サンプルプログラム

C言語、VisualBasic、Delphiそれぞれのサンプルプログラムです。
AXP-GP02用GPIBアプリケーション作成の際、参考にしてください。
サンプルプログラムの構築方法は同ディレクトリ内のbuildxx.txtをご参照ください。

・ドキュメントファイル

readme.txt

プログラミングには直接使用しませんが、マニュアルの補足事項、およびユーティリティの解説等が記述されていますので、必ずお読みくださるようお願いいたします。
また、ソフトウェアバージョンアップ時には、これらのファイルに重要な情報が記述されることがありますので、ご注意ください。

axpgp02ap.txt

axpgp02.dllのAPIの仕様を解説しています。ユーザーがプログラミングする場合に参照してください。

なお、これらのドキュメントファイルは、テキストファイルです。
テキストエディター、またはWindows付属のメモ帳等でご覧になれます。

4. プログラミングについて

4-1 必要なファイル

C/C++言語、VisualBasic、Delphiを使用して、AXP-GP02アプリケーションを作成するには、以下のファイルが必要です。

・ヘッダーファイル

関数の型宣言や、定数のマクロ定義、ラッパー関数定義を行っているファイルです。
プログラムソース中でインクルード（C/C++言語）、またはプロジェクトへの追加（VisualBasic、Delphi）を必ず行わなければなりません。
C/C++言語　：　axpgp02.h、gp02def.h、axpgp02w.h
VisualBasic　：　axpgp02d.bas、axpgp02w.bas
Delphi　　：　axpgp02w.pas

・インポートライブラリ

C/C++言語で、DLL関数を呼び出すためにリンクするライブラリです。
VisualBasic、Delphiの場合は必要ありません。
C/C++言語　：　axpgp02.lib

・ＤＬＬ

AXP-GP02のAPIを提供するDLLです。
お使いの環境に合わせてファイルをコピーしてください。
C/C++、VisualBasic、Delphi　：　axpgp02.dll

4-2 プログラミング

4-2-1 C 言語の場合

AXP-GP02は、Microsoft社 VisualC++4.0以降のC言語処理系に対応しています。
VisualC++、C言語プログラミングについては、それぞれのマニュアル、解説書等をご覧ください。

1) ヘッダーのインクルード

C言語用ヘッダーファイル（axpgp02.h、axpgp02w.h、gp02def.h）をインクルードします。
これらのファイルには、axpgp02.dllの各関数のプロトタイプ宣言やラッパー関数の定義、また、定数のデファイン等が記述してあります。
動作モード、GPIBコマンドの値、デリミタモードの指定、エラーコード等は、即値ではなくデファインされた値をご使用ください。
C言語の場合、DLLのロード方法には2通りあります。

・Libファイルを使用する場合
axpgp02.hとgp02def.hをインクルードし、プロジェクトにLibファイルをリンクしてください。

・ラッパー関数を使用する場合
axpgp02.h、axpgp02w.h、gp02def.hをインクルードし、axpgp02w.cをプロジェクトに加えてください。

2) 初期化関数の呼び出し

プログラムの先頭で、Gp02Create関数を呼び出してください。
次に、この関数の返り値を判定し、エラーが発生していないかを確認してください。
もし、Falseの値を返した場合は、AXP-GP02は正常に動作できないことを示しています。

3) プログラム作成

サンプルプログラム等を参考にして、アプリケーションを作成してください。

4) コンパイル・リンク

プログラムができたら、コンパイルを行います。Libファイルを使用する場合は 1) を参照してください。

5) 実行

プログラムが完成したら、AXP-GP02をPCカードスロットに挿入してあることを確認し、実行してください。
サンプルプログラムの構築方法についてはbuildvc.txtを参照してください。
このときWindowsが、AXP-GP02を認識しているかを確認してください。

4-2-2 VisualBasic の場合

AXP-GP02は、Microsoft社 VisualBasic4.0以降に対応しています。
VisualBasic、Basicプログラミングについては、それぞれのマニュアル、解説書等をご覧ください。

1) ソースファイルをプロジェクトに追加

VisualBasic用ソースファイル（axpgp02d.bas、axpgp02w.bas）をプロジェクトに追加します。
このファイルには、axpgp02.dllの各関数の宣言やラッパー関数の定義、グローバル定数の宣言が記述してあります。

2) 初期化関数の呼び出し

プログラムの最初で、Gp02Create関数及びGp02Initを呼び出してください（Form_Load関数内など）。
次に、この関数の返り値を判定し、エラーが発生していないかを確認してください。
もし、Falseの値を返した場合は、AXP-GP02は正常に動作できないことを示しています。

3) プログラム作成

サンプルプロジェクト等を参考にして、アプリケーションを作成してください。

4) 実行

プログラムが完成したら、AXP-GP02をPCカードスロットに挿入してあることを確認し、実行してください（VisualBasicの場合、axpgp02.libは使用しません）。
サンプルプログラムの構築方法についてはbuildvb.txtを参照してください。
このときWindowsが、AXP-GP02を認識しているかを確認してください。

4-2-3 Delphi の場合

AXP-GP02は、Borland社 Delphi2.0以降に対応しています。
Delphiプログラミングについては、それぞれのマニュアル、解説書等をご覧ください。

1）ユニットをプロジェクトに追加

Delphi用ユニット（axpgp02w.pas）をプロジェクトに追加します。
このファイルには、axpgp02.dllの各関数の宣言やラッパー関数の定義、グローバル定数の宣言が記述してあります。

2）初期化関数の呼び出し

プログラムの最初で、Gp02Create関数及びGp02Initを呼び出してください。
次に、これらの関数の返り値を判定し、エラーが発生していないかを確認してください。
もし、Falseの値を返した場合は、AXP-GP02は正常に動作できないことを示しています。

3）プログラム作成

サンプルプロジェクト等を参考にして、アプリケーションを作成してください。

4）実行

プログラムが完成したら、AXP-GP02をPCカードスロットに挿入してあることを確認し、実行してください（Delphiの場合、axpgp02.libは使用しません）。
サンプルプログラムの構築方法についてはbuilddp.txtを参照してください。
このときWindowsが、AXP-GP02を認識しているかを確認してください。

5. 関数リファレンス

5-1 関数一覧

axpgp02.dllがサポートするAPIは以下のとおりです。

【初期化ＡＰＩ】

Gp02GetVersion	バージョン情報取得
Gp02Create	デバイスの使用を宣言する
Gp02Close	デバイスの開放
Gp02GetResource	リソース情報取得

【ＧＰ－ＩＢ ＡＰＩ】

Gp02Init	初期化
Gp02SetTime	タイムアウト待ち時間設定
Gp02SetEos	デリミタとEOIを設定します。
Gp02SendCmd	GP-IBコマンド送信（コントローラ専用）
Gp02Ifc	IFCの送信（コントローラ専用）
Gp02Ren	RENラインの設定（コントローラ専用）
Gp02Dcl	DCLの送信（コントローラ専用）
Gp02Sdc	SDCの送信（コントローラ専用）
Gp02SendDataCN	複数の指定機器へのデータ送信（コントローラ専用）
Gp02SendDataC	指定機器へのデータ転送（コントローラ専用）
Gp02SendDataS	データ送信（スレーブ専用）
GP02RecvDataC	指定機器からのデータ受信（コントローラ専用）
GP02RecvDataS	データ受信（スレーブ専用）
Gp02SetSrqLine	SRQラインとステータスバイト出力（スレーブ専用）
Gp02GetSrqLine	SRQラインの状態判定（コントローラ専用）
Gp02SPoll	シリアルポール実行（コントローラ専用）
Gp02PPoll	パラレルポール実行（コントローラ専用）
Gp02PPollConfig	パラレルポール構成（コントローラ専用）
Gp02PpollUnConfig	パラレルポール構成解除（コントローラ専用）
Gp02PPollUnConfigAll	パラレルポール全構成解除（コントローラ専用）
Gp02PPollRespModeSet	パラレルポール応答設定（スレーブ専用）
Gp02TransDataN	複数機器間データ転送（コントローラ専用）
Gp02TransData	機器間データ転送（コントローラ専用）

【その他ＡＰＩ】

Gp02GetLastError	エラーコード取得
Gp02GetStatus	GP-IBステータスの取得

※ ヘッダーファイルの記述で、マニュアルまたはreadme.txtに説明のない関数、定数は非公開となっておりますので、アプリケーションプログラム作成時にご使用にならないようお願いいたします。

5-2 関数リファレンス

はじめに、関数リファレンスの見方を項目ごとに解説します。

機能　：　APIの機能を示します。
形式　：　C/C++言語、VisualBasic、Delphiでの呼び出し方です。
引数　：　関数の引数の意味を説明します。
　　引数の値は、呼び出す関数によって条件がつく場合があります。
　　条件を満たさない値を引数として関数に与えると、エラーを示す関数ステータスが返ります（GPERR_PARAMETER）。
返値　：　関数が返す値についての説明です。
　　値は即値ではなく、gp02def.h、axpgp02d.bas、axpgp02w.pasファイルで定義された値です。
解説　：　各関数の説明です。
注意　：　特に注意すべき事項です。

【初期化ＡＰＩ】

Ｇｐ０２Ｇｅｔ Ｖｅｒｓｉｏｎ

■機能

バージョン情報取得

■形式

・Visual C++

```
BOOL Gp02GetVersion(PDWORD pdwDllVersion,
                    PDWORD pdwDriverVersion);
```

・Visual Basic

```
Function Gp02GetVersion(pdwDllVersion As Long,
                        pdwDriverVersion As Long) As Long
```

・Delphi

```
function Gp02GetVersion(var pdwDllVersion: DWORD;
                        var pdwDriverVersion: DWORD): Boolean;
```

■入力

pdwDllVersion

DLLのバージョン番号を格納する領域へのポインタ。
NULL可。

pdwDriverVersion

デバイスドライバのバージョン番号を格納する領域へのポインタ。
NULL可。

■出力

*pdwDllVersion

DLLのバージョン番号。

*pdwDriverVersion

デバイスドライバのバージョン番号。

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。
TRUE　正常終了。

■解説

DLLとデバイスドライバのバージョン番号を取得します。

上位16Bit　メジャーバージョン
下位16Bit　マイナーバージョン

■エラー

FALSE

WindowsのGetLastError() をコールしてください。所定のフォルダにドライバが無い等の理由でデバイスドライバが組み込まれていない可能性があります。

Ｇｐ０２Ｃｒｅａｔｅ

■機能

デバイスの使用を宣言

■形式

・Visual C++

```
BOOL Gp02Create(PWORD pwLogSocket);
```

・Visual Basic

```
Function Gp02Create(pwLogSocket As Integer) As Long
```

・Delphi

```
function Gp02Create(var pwLogSocket: WORD): Boolean;
```

■入力

*pwLogSocket

使用したいデバイスの論理ソケット番号を指定します。
ここで指定する番号はAXP-GP02カードの論理ソケット番号を設定します。
論理ソケット番号は0から始まります。
またGP02_SOCKET_AUTOを指定した場合、デバイスが存在している論理ソケットを探します。すでにアプリケーションにより使用されているデバイスはスキップします。
以後、アプリケーションはこの値でデバイスを識別します。
NULLは不可。

■出力

*pwLogSocket

使用可能なデバイスが見つかった場合は、その論理ソケット番号を格納します。見つからなかった場合、この値は未定義です。

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。
TRUE　正常終了。

■解説

ソケットに存在しているであろうデバイスを、アプリケーションが使用することをデバイスドライバに通知します。
ソケットにデバイスが存在していない場合はエラーとなります。
ソケット内のデバイスがすでに他のアプリケーションで使用されている場合もエラーとなります。これにより、1つのデバイスは単一のアプリケーションから排他的に使用されます。
Gp02GetVersion以外のAPIを呼び出す前に必ずこのAPIを呼び出してください。

■エラー

GP02_ERR_SYSTEM

WindowsのGetLastError() をコールしてください。

GP02_ERR_NO_DEVICE(*)

使用可能なデバイスがありません（GP02_SOCKET_AUTOを指定した場合）。

GP02_ERR_INVALID_SOCKET(*)

指定のデバイスは使用中です。

GP02_ERR_INVALID_ARGUMENT

pwLogSocketがNULLです。

このAPIが失敗した場合、Gp02GetLastError() のwLogSocketにはGP02_SOCKET_AUTOを指定してください。

Ｇｐ０２Ｃｌｏｓｅ

■機能

デバイスの開放

■形式

・Visual C++

```
BOOL Gp02Close(WORD wLogSocket);
```

・Visual Basic

```
Function Gp02Close(ByVal wLogSocket As Integer) As Long
```

・Delphi

```
function Gp02Close(wLogSocket: WORD): Boolean;
```

■入力

wLogSocket

開放するデバイスの論理ソケット番号を指定します。

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

アプリケーションがデバイスの使用を終了し、デバイスを他のアプリケーションに開放することをデバイスドライバに通知します。

アプリケーションを終了する前に必ずこのAPIを呼び出してください。

■エラー

GP02_ERR_SYSTEM

WindowsのGetLastError() をコールしてください。

GP02_ERR_INVALID_SOCKET(*)

無効な論理ソケット番号です。

またはデバイスがクリエイトされていません。

このAPIが失敗した場合、Gp02GetLastError() のwLogSocketにはGP02_SOCKET_AUTOを指定してください。

Ｇｐ０２ＧｅｔＲｅｓｏｕｒｃｅ

■機能

リソース情報取得

■形式

・Visual C++

```
#define GP02_MAX_MEM      9
#define GP02_MAX_IO       20
#define GP02_MAX_IRQ      7
#define GP02_MAX_DMA      7

typedef struct _GP02RESOURCE {
    DWORD   dwNumMemWindows;                // ﾒﾓﾘｳｲﾝﾄﾞｳ数 未使用
    DWORD   dwMemBase [GP02_MAX_MEM];       // ﾒﾓﾘｳｲﾝﾄﾞｳﾍﾞｰｽｱﾄﾞﾚｽ 未使用
    DWORD   dwMemLength [GP02_MAX_MEM];     // ﾒﾓﾘｳｲﾝﾄﾞｳ長 未使用
    DWORD   dwMemAttrib [GP02_MAX_MEM];     // ﾒﾓﾘｳｲﾝﾄﾞｳ属性 未使用
    DWORD   dwNumIOPorts;                   // I/Oﾎﾟｰﾄ数
    DWORD   dwIOPortBase [GP02_MAX_IO];     // I/Oﾎﾟｰﾄﾍﾞｰｽｱﾄﾞﾚｽ
    DWORD   dwIOPortLength [GP02_MAX_IO];   // I/Oﾎﾟｰﾄ長
    DWORD   dwNumIRQs;                      // IRQ数
    DWORD   dwIRQRegisters [GP02_MAX_IRQ];  // IRQﾘｽﾄ
    DWORD   dwIRQAttrib [GP02_MAX_IRQ];     // IRQ属性ﾘｽﾄ
    DWORD   dwNumDMAs;                      // DMAﾁｬﾈﾙ数 未使用
    DWORD   dwDMALst [GP02_MAX_DMA];        // DMAﾁｬﾈﾙﾘｽﾄ 未使用
    DWORD   dwDMAAttrib [GP02_MAX_DMA];     // DMAﾁｬﾈﾙ属性ﾘｽﾄ 未使用
    DWORD   dwReserved1 [3];                // 予約 未使用
} GP02RESOURCE, *PGP02R;

BOOL Gp02GetResource(WORD wLogSocket,
                     PGP02R pres);
```

・Visual Basic

```
Global Const GP02_MAX_MEM = 9
Global Const GP02_MAX_IO = 20
Global Const GP02_MAX_IRQ = 7
Global Const GP02_MAX_DMA = 7

Type GP02RESOURCE
    dwNumMemWindows As Long                          ' ﾒﾓﾘｳｲﾝﾄﾞｳ数              未使用
    dwMemBase(1 To GP02_MAX_MEM) As Long             ' ﾒﾓﾘｳｲﾝﾄﾞｳﾍﾞｰｽｱﾄﾞﾚｽ      未使用
    dwMemLength(1 To GP02_MAX_MEM) As Long           ' ﾒﾓﾘｳｲﾝﾄﾞｳ長             未使用
    dwMemAttrib(1 To GP02_MAX_MEM) As Long           ' ﾒﾓﾘｳｲﾝﾄﾞｳ属性           未使用
    dwNumIOPorts As Long                             'I/Oﾎﾟｰﾄ数
    dwIOPortBase(1 To GP02_MAX_IO) As Long           'I/Oﾎﾟｰﾄﾍﾞｰｽｱﾄﾞﾚｽ
    dwIOPortLength(1 To GP02_MAX_IO) As Long         'I/Oﾎﾟｰﾄ長
    dwNumIRQs As Long                                'IRQ数
    dwIRQRegisters(1 To GP02_MAX_IRQ) As Long        'IRQﾘｽﾄ
    dwIRQAttrib(1 To GP02_MAX_IRQ) As Long           'IRQ属性ﾘｽﾄ
    dwNumDMAs As Long                                'DMAﾁｬﾈﾙ数              未使用
    dwDMALst(1 To GP02_MAX_DMA) As Long              'DMAﾁｬﾈﾙﾘｽﾄ             未使用
    dwDMAAttrib(1 To GP02_MAX_DMA) As Long           'DMAﾁｬﾈﾙ属性ﾘｽﾄ         未使用
    dwReserved1(1 To 3) As Long                      ' 予約                   未使用
End Type

Function Gp02GetResource(ByVal wLogSocket As Integer,
                         pres As GP02RESOURCE) As Long
```

・Delphi

```
const
  GP02_MAX_MEM    =   9;
  GP02_MAX_IO     =  20;
  GP02_MAX_IRQ    =   7;
  GP02_MAX_DMA    =   7;

type
  TGP02RESOURCE = record
    dwNumMemWindows:                              DWORD;  // ﾒﾓﾘｳｲﾝﾄﾞｳ数(未使用)
    dwMemBase:     array [1..GP02_MAX_MEM] of DWORD;  // ﾒﾓﾘｳｲﾝﾄﾞｳﾍﾞｰｽｱﾄﾞﾚｽ(未使用)
    dwMemLength:   array [1..GP02_MAX_MEM] of DWORD;  // ﾒﾓﾘｳｲﾝﾄﾞｳ長(未使用)
    dwMemAttrib:   array [1..GP02_MAX_MEM] of DWORD;  // ﾒﾓﾘｳｲﾝﾄﾞｳ属性(未使用)
    dwNumIOPorts:                                 DWORD;  // I/Oﾎﾟｰﾄ数
    dwIOPortBase:  array [1..GP02_MAX_IO] of      DWORD;  // I/Oﾎﾟｰﾄﾍﾞｰｽｱﾄﾞﾚｽ
    dwIOPortLength: array [1..GP02_MAX_IO] of     DWORD;  // I/Oﾎﾟｰﾄ長
    dwNumIRQs:                                    DWORD;  // IRQ数
    dwIRQRegisters: array [1..GP02_MAX_IRQ] of    DWORD;  // IRQﾘｽﾄ
    dwIRQAttrib:   array [1..GP02_MAX_IRQ] of     DWORD;  // IRQ属性ﾘｽﾄ
    dwNumDMAs:                                    DWORD;  // DMAﾁｬﾈﾙ数(未使用)
    dwDMALst:      array [1..GP02_MAX_DMA] of DWORD;  // DMAﾁｬﾈﾙﾘｽﾄ(未使用)
    dwDMAAttrib:   array [1..GP02_MAX_DMA] of DWORD;  // DMAﾁｬﾈﾙ属性ﾘｽﾄ(未使用)
    dwReserved1:   array [1..3] of                DWORD;  //  予約(未使用)
  end;
  PGP02RESOURCE = ^TGP02RESOURCE;

function Gp02GetResource(wLogSocket: WORD;
                         pres: PGP02RESOURCE): Boolean;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

pres

リソース情報を格納する領域へのポインタ。

NULLは不可。

■出力

*pres

割り当てられているリソースの情報。

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

wLogSocketで指定されたデバイスに割り当てられているリソースを表示用に取得します。

通常このAPIは使用する必要はありません。

■エラー

GP02_ERR_SYSTEM

WindowsのGetLastError() をコールしてください。

GP02_ERR_INVALID_SOCKET(*)

無効な論理ソケット番号です。

またはデバイスがクリエイトされていません。

GP02_ERR_INVALID_ARGUMENT

presがNULLです。

Gp02Init

■機能

AXP-GP02の初期化

■形式

・Visual C++

```
BOOL Gp02Init(WORD wLogSocket,
              WORD wGpibAddr,
              WORD wGpibMode,
              DWORD dwFifoSize);
```

・Visual Basic

```
Function Gp02Init(ByVal wLogSocket As Integer,
                  ByVal wGpibAddr As Integer,
                  ByVal wGpibMode As Integer,
                  ByVal dwFifoSize As Long) As Long
```

・Delphi

```
function Gp02Init(wLogSocket: Word;
                  wGpibAddr: Word;
                  wGpibMode: Word;
                  dwFifoSize: DWord): Boolean;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

wGpibAddr

自分自身のGP-IBアドレス。

wGpibMode

自分自身のGP-IB動作モード。

MODE_SYSCON：コントローラとして動作

MODE_TLKLSN：スレーブとして動作

dwFifoSize

送受信バッファの大きさ。

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

wLogSocketで指定されたデバイスの初期化を行います。
送受信バッファは1ブロック（デリミタまたはEOIまで）の通信量以上の十分なサイズを指定してください。
システム・コントローラとして動作する場合はGp02Ifcコマンドを送出してください。

以下の処理が行われます。

- ドライバ内にバッファを確保してクリア
- デリミタを「CR+LF」にセット
- EOIを「使用する」にセット
- タイムアウトを「2秒（デフォルト値）」にセット
- TMS9914の初期化
- エラーコードやステータスのクリア

■エラー

GP02_ERR_SYSTEM
WindowsのGetLastError() をコールしてください。

GP02_ERR_BUSY
ドライバがビジー状態です（動作中）。

GP02_ERR_INVALID_ARGUMENT
GP-IBアドレスや動作モードが範囲外です。

GP02_ERR_INVALID_SOCKET(*)
無効な論理ソケット番号です。
またはデバイスがクリエイトされていません。

GP02_ERR_NOTINIT(*)
バッファの確保に失敗しました。

Ｇｐ０２Ｓｅｔ Ｔｉｍｅ

■機能

GP-IB処理のタイムアウト設定

■形式

・Visual C++

```
BOOL Gp02SetTime(WORD wLogSocket,
                 DWORD dwTimeoutCnt);
```

・Visual Basic

```
Function Gp02SetTime(ByVal wLogSocket As Integer,
                     dwTimeoutCnt As Long) As Long
```

・Delphi

```
function Gp02SetTime(wLogSocket: Word;
                     dwTimeoutCnt: DWord): Boolean;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

dwTimeoutCnt

タイムアウト値。単位はミリ秒で0～60000まで。

0を指定するとタイムアウトしません。

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

wLogSocketで指定されたデバイスのGP-IB処理のタイムアウトを設定します。

■エラー

GP02_ERR_SYSTEM

WindowsのGetLastError() をコールしてください。

GP02_ERR_BUSY

ドライバがビジー状態です（動作中）。

GP02_ERR_INVALID_SOCKET(*)

無効な論理ソケット番号です。

またはデバイスがクリエイトされていません。

GP02_ERR_INVALID_ARGUMENT(*)

タイムアウト値が範囲外です。

Ｇｐ０２ＳｅｔＥｏｓ

■機能

デリミタとEOIを設定します。

■形式

・Visual C++

```
BOOL Gp02SetEos(WORD wLogSocket,
                WORD wDlmMode,
                WORD wEoiMode);
```

・Visual Basic

```
Function Gp02SetEos(ByVal wLogSocket As Integer,
                    ByVal wDlmMode As Integer,
                    ByVal wEoiMode As Integer) As Long
```

・Delphi

```
function Gp02SetEos(wLogSocket: Word;
                    wDlmMode: Word;
                    wEoiMode: Word): Boolean;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

wDlmMode

デリミタを指定。

DELIM_NO　：デリミタを使用しない

DELIM_CRLF：CR+LF（デフォルト）

DELIM_CR　：CR

DELIM_LF　：LF

wEoiMode

EOIの使用または不使用を指定。

EOI_OFF：使用しない

EOI_ON　：使用する（デフォルト）

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

デリミタの種類とEOIの使用・不使用を指定します。

■エラー

GP02_ERR_SYSTEM

WindowsのGetLastError() をコールしてください。

GP02_ERR_BUSY

ドライバがビジー状態です（動作中）。

GP02_ERR_INVALID_ARGUMENT

デリミタまたはEOIの指定値が範囲外です。

GP02_ERR_INVALID_SOCKET(*)

無効な論理ソケット番号です。

またはデバイスがクリエイトされていません。

Ｇｐ０２ＳｅｎｄＣｍｄ

■機能

GP-IBコマンド送信（コントローラ専用）

■形式

・Visual C++

```
BOOL Gp02SendCmd(WORD wLogSocket,
                 PBYTE paSendBuf,
                 DWORD dwSendBytes);
```

・Visual Basic

```
Function Gp02SendCmd(ByVal wLogSocket As Integer,
                     ByRef paSendBuf As Byte,
                     ByVal dwSendBytes As Long) As Long
```

・Delphi

```
function Gp02SendCmd(wLogSocket: Word;
                     var paSendBuf: array of Byte;
                     dwSendBytes: DWord): Boolean;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

paSendBuf

送信コマンドバッファのアドレス。

dwSendBytes

送信すべきコマンドのバイト数。

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

wLogSocketで指定されたデバイスからコマンドを送信します。リスナが存在しない場合でもエラーにはなりません。

■エラー

GP02_ERR_SYSTEM

WindowsのGetLastError() をコールしてください。

GP02_ERR_BUSY

ドライバがビジー状態です（動作中）。

GP02_ERR_INVALID_ARGUMENT

Gp02Initコマンドで指定したバッファサイズ以上を送信しようとしています。
またはアドレスの指定エラーです。

GP02_ERR_MODE

自分自身がコントローラではありません。

GP02_ERR_INVALID_SOCKET(*)

無効な論理ソケット番号です。
またはデバイスがクリエイトされていません。

GP02_ERR_TIMEOUT(*)

タイムアウトしました。

■機能
　IFCの送信（コントローラ専用）
■形式
・Visual C++
BOOL Gp02Ifc(WORD wLogSocket);

・Visual Basic
Function Gp02Ifc(ByVal wLogSocket As Integer) As Long

・Delphi
function Gp02Ifc(wLogSocket: Word): Boolean;

■入力
　wLogSocket
　　デバイスの論理ソケット番号を指定。
■戻り値
　APIが正常終了したか、失敗したかを返します。
　　FALSE 失敗。
　　TRUE　正常終了。
■解説
　wLogSocketで指定された論理ソケットからIFCを出力します。
■エラー
　GP02_ERR_BUSY
　　ドライバがビジー状態です（動作中）。
　GP02_ERR_MODE
　　自分自身がコントローラではありません。
　GP02_ERR_INVALID_SOCKET(*)
　　無効な論理ソケット番号です。
　　またはデバイスがクリエイトされていません。
　GP02_ERR_TIMEOUT(*)
　　タイムアウトしました。

Ｇｐ０２Ｒｅｎ

■機能

RENラインの設定（コントローラ専用）

■形式

・Visual C++

```
BOOL Gp02Ren(WORD wLogSocket,
             WORD wRenLine);
```

・Visual Basic

```
Function Gp02Ren(ByVal wLogSocket As Integer,
                 ByVal wRenLine As Integer) As Long
```

・Delphi

```
function Gp02Ren(wLogSocket: Word;
                 wRenLine: Word): Boolean;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

wRenLine

1：真

0：偽

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

wLogSocketで指定された論理ソケットからRENを設定します。

■エラー

GP02_ERR_BUSY

ドライバがビジー状態です（動作中）。

GP02_ERR_INVALID_ARGUMENT

パラメータが不正です（wRenLine）。

GP02_ERR_MODE

自分自身がコントローラではありません。

GP02_ERR_INVALID_SOCKET(*)

無効な論理ソケット番号です。

またはデバイスがクリエイトされていません。

Ｇｐ０２Ｄｃｌ

■機能
　DCLの送信（コントローラ専用）
■形式
・Visual C++
BOOL Gp02Dcl(WORD wLogSocket);

・Visual Basic
Function Gp02Dcl(ByVal wLogSocket As Integer) As Long

・Delphi
function Gp02Dcl(wLogSocket: Word): Boolean;

■入力
　wLogSocket
　　デバイスの論理ソケット番号を指定。
■戻り値
　APIが正常終了したか、失敗したかを返します。
　　FALSE 失敗。
　　TRUE　正常終了。
■解説
　wLogSocketで指定された論理ソケットからDCLを出力します。
■エラー
　GP02_ERR_BUSY
　　ドライバがビジー状態です（動作中）。
　GP02_ERR_MODE
　　自分自身がコントローラではありません。
　GP02_ERR_INVALID_SOCKET(*)
　　無効な論理ソケット番号です。
　　またはデバイスがクリエイトされていません。
　GP02_ERR_TIMEOUT(*)
　　タイムアウトしました。

Ｇｐ０２Ｓｄｃ

■機能

SDCの送信（コントローラ専用）

■形式

・Visual C++

```
BOOL Gp02Sdc(WORD wLogSocket,
             WORD wLsnAdrs);
```

・Visual Basic

```
Function Gp02Dcl(ByVal wLogSocket As Integer,
                 ByVal wLsnAdrs As Integer) As Long
```

・Delphi

```
function Gp02Sdc(wLogSocket: Word;
                 wLsnAdrs: Word): Boolean;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

wLsnAdrs

リスナアドレス。

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

wLogSocketで指定された論理ソケットからSDCを出力します。リスナが存在しない場合でもエラーにはなりません。

■エラー

GP02_ERR_BUSY

ドライバがビジー状態です（動作中）。

GP02_ERR_INVALID_ARGUMENT

アドレスの指定エラーです。

GP02_ERR_MODE

自分自身がコントローラではありません。

GP02_ERR_INVALID_SOCKET(*)

無効な論理ソケット番号です。

またはデバイスがクリエイトされていません。

GP02_ERR_TIMEOUT(*)

タイムアウトしました。

Ｇｐ０２ＳｅｎｄＤａｔａＣＮ

■機能

複数の指定機器へのデータ送信（コントローラ専用）

■形式

・Visual C++

```
BOOL Gp02SendDataCN(WORD wLogSocket,
                    PWORD pwLsnAdrsList,
                    WORD wLsnAdrsCnt,
                    PBYTE pbSendBuff,
                    DWORD dwDataCnt);
```

・Visual Basic

```
Function Gp02SendDataCN(ByVal wLogSocket As Integer,
                        ByRef pwLsnAdrsList As Integer,
                        ByVal wLsnAdrsCnt As Integer,
                        ByRef pbSendBuff As Byte,
                        ByVal dwDataCnt As Long) As Long
```

・Delphi

```
function Gp02SendDataCN(wLogSocket: Word;
                        var pwLsnAdrsList: array of Word;
                        wLsnAdrsCnt: Word;
                        var pbSendBuff: array of Byte;
                        dwDataCnt: DWord): Boolean;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

pwLsnAdrsList

リスナアドレスリストのアドレス。

wLsnAdrsCnt

リスナアドレスの数。

pbSendBuff

送信データバッファのアドレス。

dwDataCnt

送信データ数。

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

wLogSocketで指定されたデバイスから複数のGP-IB機器に対してデータを送信します。
pbSendBuffのデータを全て送信後、自動的にGp02SetEosコマンドで指定したデリミタを送信し、EOIも（「使用する」に設定されていれば）出力します。

■エラー

GP02_ERR_BUSY
ドライバがビジー状態です（動作中）。

GP02_ERR_INVALID_ARGUMENT
Gp02Initコマンドで指定したバッファサイズ以上を送信しようとしています。
またはアドレスの指定エラーです。

GP02_ERR_MODE
自分自身がコントローラではありません。

GP02_ERR_INVALID_SOCKET(*)
無効な論理ソケット番号です。
またはデバイスがクリエイトされていません。

GP02_ERR_TIMEOUT(*)
タイムアウトしました。

GP02_ERR_NOLSN(*)
リスナが存在しません。
またはケーブルが接続されていません。

Ｇｐ０２ＳｅｎｄＤａｔａＣ

■機能

指定機器へのデータ送信（コントローラ専用）

■形式

・Visual C++

```
BOOL Gp02SendDataC(WORD wLogSocket,
                   WORD wLsnAdrs,
                   PBYTE pbSendBuff,
                   DWORD dwDataCnt);
```

・Visual Basic

```
Function Gp02SendDataC(ByVal wLogSocket As Integer,
                       ByVal wLsnAdrs As Integer,
                       ByRef pbSendBuff As Byte,
                       ByVal dwDataCnt As Long) As Long
```

・Delphi

```
function Gp02SendDataC(wLogSocket: Word;
                       wLsnAdrs: Word;
                       var pbSendBuff: array of Byte;
                       dwDataCnt: DWord): Boolean;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

wLsnAdrs

リスナアドレス。

pbSendBuff

送信データバッファのアドレス。

dwDataCnt

送信データ数。

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

wLogSocketで指定されたデバイスから指定のGP-IB機器に対してデータを送信します。

pbSendBuffのデータを全て送信後、自動的にGp02SetEosコマンドで指定したデリミタを送信し、EOIも（「使用する」に設定されていれば）出力します。

■エラー

GP02_ERR_BUSY

ドライバがビジー状態です（動作中）。

GP02_ERR_INVALID_ARGUMENT

Gp02Initコマンドで指定したバッファサイズ以上を送信しようとしています。
またはアドレスの指定エラーです。

GP02_ERR_MODE

自分自身がコントローラではありません。

GP02_ERR_INVALID_SOCKET(*)

無効な論理ソケット番号です。
またはデバイスがクリエイトされていません。

GP02_ERR_TIMEOUT(*)

タイムアウトしました。

GP02_ERR_NOLSN(*)

リスナが存在しません。
またはケーブルが接続されていません。

Ｇｐ０２ＳｅｎｄＤａｔａＳ

■機能

データ送信（スレーブ専用）

■形式

・Visual C++

```
BOOL Gp02SendDataS(WORD wLogSocket,
                   PBYTE pbSendBuff,
                   DWORD dwDataCnt);
```

・Visual Basic

```
Function Gp02SendDataS(ByVal wLogSocket As Integer,
                       ByRef pbSendBuff As Byte,
                       ByVal dwDataCnt As Long) As Long
```

・Delphi

```
function Gp02SendDataS(wLogSocket: Word;
                       var pbSendBuff: array of Byte;
                       dwDataCnt: DWord): Boolean;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

pbSendBuff

送信データバッファのアドレス。

dwDataCnt

送信データ数。

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

wLogSocketで指定されたデバイスからデータを送信します。

pbSendBuffのデータを全て送信後、自動的にGp02SetEosコマンドで指定したデリミタを送信し、EOIも（「使用する」に設定されていれば）出力します。

■エラー

GP02_ERR_BUSY

ドライバがビジー状態です（動作中）。

GP02_ERR_INVALID_ARGUMENT

Gp02Initコマンドで指定したバッファサイズ以上を送信しようとしています。

GP02_ERR_MODE

自分自身がスレーブではありません。

GP02_ERR_INVALID_SOCKET(*)

無効な論理ソケット番号です。

またはデバイスがクリエイトされていません。

GP02_ERR_TIMEOUT(*)

タイムアウトしました。

Ｇｐ０２ＲｅｃｖＤａｔａＣ

■機能

指定機器からのデータ受信（コントローラ専用）

■形式

・Visual C++

```
BOOL Gp02RecvDataC(WORD wLogSocket,
                   WORD wTlkAdrs,
                   PBYTE pbRecvBuff,
                   PDWORD pdwDataCnt);
```

・Visual Basic

```
Function Gp02RecvDataC(ByVal wLogSocket As Integer,
                       ByVal wTlkAdrs As Integer,
                       ByRef pbRecvBuff As Byte,
                       ByRef pdwDataCnt As Long) As Long
```

・Delphi

```
function Gp02RecvDataC(wLogSocket: Word;
                       wTlkAdrs: Word;
                       var pbRecvBuff: array of Byte;
                       var pdwDataCnt: DWord): Boolean;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

wTlkAdrs

トーカアドレス。

pbRecvBuff

受信データバッファのアドレス。

pdwDataCnt

受信データ数を指定する変数のアドレス。

■出力

pbRecvBuff

受信データ。

*pdwDataCnt

受信したバイト数。

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

wLogSocketで指定されたデバイスを使って、指定のGP-IB機器からデータを受信します。
*pdwDataCntで指定した分を受信するか、指定デリミタを受信した時点またはEOIを検出すると終了します。

■エラー

GP02_ERR_BUSY

ドライバがビジー状態です（動作中）。

GP02_ERR_INVALID_ARGUMENT

Gp02Init コマンドで指定したバッファサイズ以上を受信しようとしています。
またはアドレスの指定エラーです。

GP02_ERR_MODE

自分自身がコントローラではありません。

GP02_ERR_INVALID_SOCKET(*)

無効な論理ソケット番号です。
またはデバイスがクリエイトされていません。

GP02_ERR_TIMEOUT(*)

タイムアウトしました。

■機能

データ受信（スレーブ専用）

■形式

・Visual C++

```
BOOL Gp02RecvDataS(WORD wLogSocket,
                   PBYTE pbRecvBuff,
                   PDWORD pdwDataCnt);
```

・Visual Basic

```
Function Gp02RecvDataS(ByVal wLogSocket As Integer,
                       ByRef pbRecvBuff As Byte,
                       ByRef pdwDataCnt As Long) As Long
```

・Delphi

```
function Gp02RecvDataS(wLogSocket: Word;
                       var pbRecvBuff: array of Byte;
                       var pdwDataCnt: DWord): Boolean;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

pbRecvBuff

受信データバッファのアドレス。

pdwDataCnt

受信データ数を指定する変数のアドレス。

■出力

pbRecvBuff

受信データ。

*pdwDataCnt

受信したバイト数。

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

wLogSocketで指定されたデバイスからデータを受信します。

*pdwDataCntで指定した分を受信するか、指定デリミタを受信した時点またはEOIを検出すると終了します。

■エラー

GP02_ERR_BUSY

ドライバがビジー状態です（動作中）。

GP02_ERR_INVALID_ARGUMENT

Gp02Initコマンドで指定したバッファサイズ以上を受信しようとしています。

GP02_ERR_MODE

自分自身がスレーブではありません。

GP02_ERR_INVALID_SOCKET(*)

無効な論理ソケット番号です。

またはデバイスがクリエイトされていません。

GP02_ERR_TIMEOUT(*)

タイムアウトしました。

Ｇｐ０２ＳｅｔＳｒｑＬｉｎｅ

■機能

SRQラインとステータスバイト出力（スレーブ専用）

■形式

・Visual C++

```
BOOL Gp02SetSrqLine(WORD wLogSocket,
                    BYTE bStsByte);
```

・Visual Basic

```
Function Gp02SetSrqLine(ByVal wLogSocket As Integer,
                        ByVal bStsByte As Byte) As Long
```

・Delphi

```
function Gp02SetSrqLine(wLogSocket: Word;
                        bStsByte: Byte): Boolean;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

bStsByte

ステータスバイト値。

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

wLogSocketで指定されたデバイスのSRQラインを真にします。

このとき、bStsByteで設定したステータスバイトの第6ビットをセットして出力します。

■エラー

GP02_ERR_BUSY

ドライバがビジー状態です（動作中）。

GP02_ERR_MODE

自分自身がスレーブではありません。

GP02_ERR_INVALID_SOCKET(*)

無効な論理ソケット番号です。

またはデバイスがクリエイトされていません。

Ｇｐ０２ＧｅｔＳｒｑＬｉｎｅ

■機能

SRQラインの状態判定（コントローラ専用）

■形式

・Visual C++

```
BOOL Gp02GetSrqLine(WORD wLogSocket,
                    PWORD pwSrqLine);
```

・Visual Basic

```
Function Gp02GetSrqLine( ByVal wLogSocket As Integer,
                         ByRef pwSrqLine As Integer) As Long
```

・Delphi

```
function Gp02GetSrqLine(wLogSocket: Word;
                        var pwSrqLine: Word): Boolean;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

pwSrqLine

SRQラインの状態を格納する変数のアドレス。

■出力

*pwSrqLine

SRQライン状態。

真：1

偽：0

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

wLogSocketで指定されたデバイスのSRQライン状態を取得します。

SRQラインが真なら「1」、偽なら「0」がpwSrqLineにセットされます。

■エラー

GP02_ERR_BUSY

ドライバがビジー状態です（動作中）。

GP02_ERR_MODE

自分自身がコントローラではありません。

GP02_ERR_INVALID_SOCKET(*)

無効な論理ソケット番号です。

またはデバイスがクリエイトされていません。

■機能

シリアルポール実行（コントローラ専用）

■形式

・Visual C++

```
BOOL Gp02Spoll(WORD wLogSocket,
               WORD wPollAdrs,
               PBYTE pbStsByte);
```

・Visual Basic

```
Function Gp02Spoll(ByVal wLogSocket As Integer,
                   ByVal wPollAdrs As Integer,
                   ByRef pbStsByte As Byte) As Long
```

・Delphi

```
function Gp02Spoll(wLogSocket: Word;
                   wPollAdrs: Word;
                   var pbStsByte: Byte): Boolean;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

wPollAdrs

ポーリング相手のアドレス。

pbStsByte

ステータスバイトを格納する変数のアドレス。

■出力

*pbStsByte

ステータスバイト。

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

wLogSocketで指定されたデバイスから指定した機器に対して、シリアルポーリングを行います。

機器がサービスをリクエストしている場合は*pbStsByteにゼロ以外の値が返ります。

■エラー

GP02_ERR_BUSY

ドライバがビジー状態です（動作中）。

GP02_ERR_INVALID_ARGUMENT

アドレスの指定エラー。

GP02_ERR_MODE

自分自身がコントローラではありません。

GP02_ERR_INVALID_SOCKET(*)

無効な論理ソケット番号です。

またはデバイスがクリエイトされていません。

GP02_ERR_TIMEOUT(*)

タイムアウトしました。

■機能

パラレルポール実行（コントローラ専用）

■形式

・Visual C++

```
BOOL Gp02Ppoll(WORD wLogSocket,
                    PBYTE pbDioLine);
```

・Visual Basic

```
Function Gp02Ppoll(ByVal wLogSocket As Integer,
                        ByRef pbDioLine As Byte) As Long
```

・Delphi

```
function Gp02Ppoll(wLogSocket: Word;
                    var pbDioLine: Byte): Boolean;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

pbDioLine

DIOラインを格納する変数のアドレス。

■出力

*pbDioLine

DIOライン。

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

wLogSocketで指定されたデバイスからパラレルポールを実行し、DIOラインの状態を返します。

パラレルポールに対応していない機器がありますので注意してください。

この関数を呼び出す前にGp02PpollConfigを呼び、パラレルポーリング終了後にGp02PpollUConfigまたはGp02PpollUConfigAllを呼び出してください。

■エラー

GP02_ERR_BUSY

ドライバがビジー状態です（動作中）。

GP02_ERR_MODE

自分自身がコントローラではありません。

GP02_ERR_INVALID_SOCKET(*)

無効な論理ソケット番号です。

またはデバイスがクリエイトされていません。

■機能

パラレルポール構成（コントローラ専用）

■形式

・Visual C++

```
BOOL Gp02PpollConfig(WORD wLogSocket,
                     WORD wLsnAdrs,
                     BYTE bDioLine,
                     BYTE bSbit);
```

・Visual Basic

```
Function Gp02PpollConfig(ByVal wLogSocket As Integer,
                         ByVal wLsnAdrs As Integer,
                         ByVal bDioLine As Byte,
                         ByVal bSbit As Byte) As Long
```

・Delphi

```
function Gp02PpollConfig(wLogSocket: Word;
                         wLsnAdrs: Word;
                         bDioLine: Byte;
                         bSbit: Byte): Boolean;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

wLsnAdrs

パラレルポールする機器アドレス。

bDioLine

機器を割り付けるDIOライン番号（0～7のビット番号で指定）。

bSbit

センスビットの指定（0または1）。

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

wLogSocketで指定されたデバイスから指定した機器をDIOラインに割り付けます。

Gp02Ppoll関数を呼ぶ前にこの関数を呼び出してください。

■エラー

GP02_ERR_BUSY

ドライバがビジー状態です（動作中）。

GP02_ERR_INVALID_ARGUMENT

アドレス、DIOまたはSビットの指定エラー。

GP02_ERR_MODE

自分自身がコントローラではありません。

GP02_ERR_INVALID_SOCKET(*)

無効な論理ソケット番号です。

またはデバイスがクリエイトされていません。

GP02_ERR_TIMEOUT(*)

タイムアウトしました。

Ｇｐ０２ＰｐｏｌｌＵｎＣｏｎｆｉｇ

■機能

パラレルポール構成解除（コントローラ専用）

■形式

・Visual C++

```
BOOL Gp02PpollUnConfig(WORD wLogSocket,
                       WORD wLsnAdrs);
```

・Visual Basic

```
Function Gp02PpollUnConfig(ByVal wLogSocket As Integer,
                           ByVal wLsnAdrs As Integer) As Long
```

・Delphi

```
function Gp02PpollUnConfig(wLogSocket: Word;
                           wLsnAdrs: Word): Boolean;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

wLsnAdrs

パラレルポール構成解除する機器アドレス。

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

wLogSocketで指定されたデバイスから指定した機器1台のDIOライン割り付けを解除します。パラレルポールを行った後、本関数またはGpibPpollUConfigAll関数を呼び出してパラレルポールの構成を解除してください。

■エラー

GP02_ERR_BUSY

ドライバがビジー状態です（動作中）。

GP02_ERR_INVALID_ARGUMENT

アドレスの指定エラー。

GP02_ERR_MODE

自分自身がコントローラではありません。

GP02_ERR_INVALID_SOCKET(*)

無効な論理ソケット番号です。

またはデバイスがクリエイトされていません。

GP02_ERR_TIMEOUT(*)

タイムアウトしました。

Ｇｐ０２ＰｐｏｌｌＵｎＣｏｎｆｉｇＡｌｌ

■機能

パラレルポール全構成解除（コントローラ専用）

■形式

・Visual C++

```
BOOL Gp02PpollUnConfigAll(WORD wLogSocket);
```

・Visual Basic

```
Function Gp02PpollUnConfigAll(ByVal wLogSocket As Integer) As Long
```

・Delphi

```
function Gp02PpollUnConfigAll(wLogSocket: Word): Boolean;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

wLogSocketで指定されたデバイスから指定した全機器のDIOライン割り付けを解除します。

パラレルポールを行った後、本関数またはGpibPpollUConfig関数を呼び出してパラレルポールの構成を解除してください。

■エラー

GP02_ERR_BUSY

ドライバがビジー状態です（動作中）。

GP02_ERR_INVALID_ARGUMENT

アドレスの指定エラー。

GP02_ERR_MODE

自分自身がコントローラではありません。

GP02_ERR_INVALID_SOCKET(*)

無効な論理ソケット番号です。

またはデバイスがクリエイトされていません。

GP02_ERR_TIMEOUT(*)

タイムアウトしました。

Ｇｐ０２ＰｐｏｌｌＲｅｓｐＭｏｄｅＳｅｔ

■機能

パラレルポール応答設定（スレーブ専用）

■形式

・Visual C++

```
BOOL Gp02PpollRespModeSet(WORD wLogSocket,
                          WORD wPMode);
```

・Visual Basic

```
Function Gp02PpollRespModeSet(ByVal wLogSocket As Integer,
                              ByVal wPMode As Integer) As Long
```

・Delphi

```
function Gp02PpollRespModeSet(wLogSocket: Word;
                              wPMode: Word): Boolean;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

wPMode

パラレルポールへの応答方法。

0：パラレルポールに肯定的に応答する

1：パラレルポールに否定的に応答する

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

パラレルポールに対する応答の仕方を設定します。

コントローラから送信される、PPEコマンドのセンスビットと本関数で設定する値が一致した場合に、割り付けられているDIOラインを真にします。

■エラー

GP02_ERR_BUSY

ドライバがビジー状態です（動作中）。

GP02_ERR_INVALID_ARGUMENT

アドレスの指定エラー。

GP02_ERR_MODE

自分自身がスレーブではありません。

GP02_ERR_INVALID_SOCKET(*)

無効な論理ソケット番号です。

またはデバイスがクリエイトされていません。

GP02_ERR_TIMEOUT(*)

タイムアウトしました。

■機能

複数機器間データ転送（コントローラ専用）

■形式

・Visual C++

```
BOOL Gp02TransDataN(WORD wLogSocket,
                    WORD wTlkAdrs,
                    PWORD pwLsnAdrsList,
                    WORD wLsnAdrsCnt);
```

・Visual Basic

```
Function Gp02TransDataN(ByVal wLogSocket As Integer,
                        ByVal wTlkAdrs As Integer,
                        ByRef pwLsnAdrsList As Integer,
                        ByVal wLsnAdrsCnt As Integer) As Long
```

・Delphi

```
function Gp02TransDataN(wLogSocket: Word;
                        wTlkAdrs: Word;
                        var pwLsnAdrsList: array of Word;
                        wLsnAdrsCnt: Word): Boolean;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

wTlkAdrs

トーカアドレス。

pwLsnAdrsList

リスナアドレスリストのアドレス。

wLsnAdrsCnt

リスナアドレスの数。

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

wLogSocketで指定されたデバイスからGP-IB機器どうしでデータ転送を行います（リスナは複数して可能です）。

デリミタの指定は出来ず、トーカがEOIを送信したら終了します。

■エラー

GP02_ERR_BUSY

ドライバがビジー状態です（動作中）。

GP02_ERR_INVALID_ARGUMENT

アドレスの指定エラーです。

GP02_ERR_MODE

自分自身がコントローラではありません。

GP02_ERR_INVALID_SOCKET(*)

無効な論理ソケット番号です。
またはデバイスがクリエイトされていません。

GP02_ERR_TIMEOUT(*)

タイムアウトしました。

■機能

機器間データ転送（コントローラ専用）

■形式

・Visual C++

```
BOOL Gp02TransData(WORD wLogSocket,
                   WORD wTlkAdrs,
                   WORD wLsnAdrs);
```

・Visual Basic

```
Function Gp02TransData(ByVal wLogSocket As Integer,
                       ByVal wTlkAdrs As Integer,
                       ByVal wLsnAdrs As Integer) As Long
```

・Delphi

```
function Gp02TransData(wLogSocket: Word;
                       wTlkAdrs: Word;
                       wLsnAdrs: Word): Boolean;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

wTlkAdrs

トーカアドレス。

wLsnAdrsCnt

リスナアドレス。

■戻り値

APIが正常終了したか、失敗したかを返します。

FALSE 失敗。

TRUE　正常終了。

■解説

wLogSocketで指定されたデバイスからGP-IB機器どうしでデータ転送を行います（リスナは単数です）。

デリミタの指定は出来ず、トーカがEOIを送信したら終了します。

■エラー

GP02_ERR_BUSY

ドライバがビジー状態です（動作中）。

GP02_ERR_INVALID_ARGUMENT

アドレスの指定エラーです。

GP02_ERR_MODE

自分自身がコントローラではありません。

GP02_ERR_INVALID_SOCKET(*)

無効な論理ソケット番号です。
またはデバイスがクリエイトされていません。

GP02_ERR_TIMEOUT(*)

タイムアウトしました。

Ｇｐ０２ＧｅｔＬａｓｔＥｒｒｏｒ

■機能

エラーコード取得

■形式

・Visual C++

```
DWORD Gp02GetLastError(WORD wLogSocket);
```

・Visual Basic

```
Function Gp02GetLastError(ByVal wLogSocket As Integer) As Long
```

・Delphi

```
function Gp02GetLastError(wLogSocket: WORD): DWORD;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

■戻り値

エラーコード。

■解説

もっとも最近起こったエラーのコードを取得します。

Ｇｐ０２Ｇｅｔ Ｓｔａｔｕｓ

■機能

GP-IBステータスの取得

■形式

・Visual C++

```
DWORD Gp02GetStatus(WORD wLogSocket);
```

・Visual Basic

```
Function Gp02GetStatus(ByVal wLogSocket As Integer) As Long
```

・Delphi

```
function Gp02GetStatus(wLogSocket: WORD): DWORD;
```

■入力

wLogSocket

デバイスの論理ソケット番号を指定。

■戻り値

GP-IBステータス。

■解説

GP-IBステータスを取得します。

5-3 定数

＜エラーコード＞

```
#define GP02_SUCCESS                    0       // 異状なし（正常終了）
#define GP02_ERR_SYSTEM                 1
                                // WindowsのGetLastError() をコールしてください
                                // 正常にドライバがロードされていない可能性があります
#define GP02_ERR_NO_DEVICE              2       // 使用可能なデバイスがありません
#define GP02_ERR_IN_USE                 3       // 指定のデバイスは使用中です
#define GP02_ERR_INVALID_SOCKET         4       // 無効な論理ソケットです
#define GP02_ERR_RESOURCE               5       // リソースエラー
#define GP02_ERR_INVALID_ARGUMENT       6       // パラメータ不正
#define GP02_ERR_NOTINIT                7       // 初期化がされていません。
#define GP02_ERR_MODE                   8       // GPIBのモードエラーです。
#define GP02_ERR_TIMEOUT                9       // タイムアウトしました。
#define GP02_ERR_NOLSN                  10      // リスナ無し（ケーブル未接続）
#define GP02_ERR_BUSY                   11      // 実行中
#define GP02_ERR_ETC                    12      // その他のエラー
#define GP02_ERR_WRAPDLL                0xffff  // ラッパー関数内エラー
```

＜ＧＰＩＢパラメータ＞

```
// GPIBステータス取得
#define STS_IFC                 0x0001  // IFC
#define STS_DCAS                0x0002  // DeviceClear
#define STS_UNC                 0x0004  // Ppoll,TCT etc
#define STS_ERR                 0x0008  // Error
#define STS_GET                 0x0010  // GET
#define STS_SPAS                0x0020  // Spoll
#define STS_END                 0x0040  // END
#define STS_REM                 0x0100  // Remote
#define STS_LLO                 0x0200  // LLO
#define STS_TLK                 0x0400  // Talker
#define STS_LSN                 0x0800  // Listener
#define STS_ATN                 0x1000  // ATN
#define STS_SRQ                 0x2000  // SRQ
#define STS_REN                 0x4000  // REN

// GPIB関連パラメータ指定範囲
#define ADDR_MIN                0       // 最小GPIBアドレス
#define ADDR_MAX                30      // 最大　　〃
#define TIMEOUT_DEFAULT         2000L   // タイムアウトデフォルト
#define TIMEOUT_MAX             60000L  // タイムアウト最大値
```

```
// マルチラインメッセージ（GPIBコマンド）指定
#define LA_0                    0x0020      // Listener Addr
#define TA_0                    0x0040      // Talker Addr
#define UNL                     0x003f      // UNListen
#define UNT                     0x005F      // UNTalk
#define GTL                     0x0001      // Go To Local
#define SDC                     0x0004      // Select Device Clear
#define GET                     0x0008      // Group Execute Trigger
#define LLO                     0x0011      // Local Lock Out
#define DCL                     0x0014      // Device Clear
#define SPE                     0x0018      // Serial Poll Enable
#define SPD                     0x0019      // Serial Poll Disable
#define PPC                     0x0005      // Parallel Poll Config
#define PPE                     0x0060      // Parallel Poll Enable
#define PPZ                     0x0070      // Parallel Poll Disable
#define PPU                     0x0015      // Parallel Poll Unconfig
#define TCT                     0x0009      // Take Control (XXX)

// GPIB動作モード指定
#define MODE_SYSCON             1           // システムコントローラモード
#define MODE_TLKLSN             0           // トーカリスナモード
#define MODE_SLAVE              0           // スレーブモード
#define MODE_CON                2           // コントローラモード（使用不可）
#define MODE_MASTER             2           // マスタモード（使用不可）

// デリミタモード指定
#define DELIM_CRLF              1           // CR+LF（デフォルト）
#define DELIM_CR                2           // CR
#define DELIM_LF                3           // LF
#define DELIM_NO                0           // NOTHING

// EOIモード指定
#define EOI_ON                  1           // 付加（デフォルト）
#define EOI_OFF                 0           // なし
```

＜その他の定数＞

```
#define GP02_SOCKET_AUTO        ((WORD) ~0U)
#define GP02_MAX_SOCKETS        16          // サポートするボード枚数
```

6. 製品仕様

①インターフェース規格	:	PC Card Standard 準拠
②入出力形式	:	IEEE-488 (GPIB) 準拠
③チャンネル数	:	1 チャンネル
④インターフェイス機能	:	SH1、AH1、T5/TE5、L3/LE3、SR1/RL1、PP1/PP2、 DC1/DT1、C1、C2、C3、C4、C5
⑤GPIB コントローラ LSI	:	テキサスインスツルメンツ社製 TMS9914A 相当品
⑥データバッファ	:	75160／75161 相当品
⑦転送速度	:	40KByte/sec
⑧I/O アドレス	:	連続 8 アドレス
⑨割り込み	:	1 点
⑩電源	:	DC＋5V＋5% 最大 0.7A PC カードスロットより供給
⑪使用温度範囲	:	5 ～ 50℃
⑫保存温度範囲	:	0 ～ 75℃
⑬対応機種	:	PC カードスロットを搭載した IBM PC/AT 互換機 または NEC PC-98 シリーズ
⑭カード寸法	:	PC カード TypeⅡサイズ準拠 54×85.6×5 (mm)
⑮重量	:	約 20g

改訂履歴

発行年月日　　1997 年 07 月 28 日　　初版

発行年月日　　1999 年 08 月 05 日　　第 2 版

ドライババージョンアップ（API 仕様の変更）に伴い、
「5. 関数リファレンス」等の関連情報を修正

発行年月日　　2005 年 03 月 22 日　　第 3 版

お問い合わせに関する情報を修正

Windows2000/XP 対応の記述を追加

株式会社 アドテック システム サイエンス

技術的なお問い合わせはテクニカルサポートへ
E-mail　support@adtek.co.jp
FAX　(045)331-7770

インターネットホームページ　http://www.adtek.co.jp/