



REX-5052

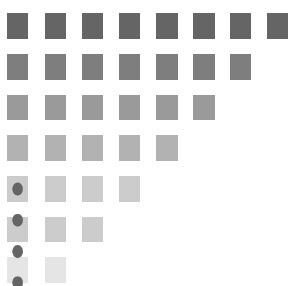
GPIB PC CARD

ユーザーズマニュアル

2002年8月

第4.0版

 **RATOC**
Systems, Inc.
ラトックシステム株式会社



第 1 章 GPIB PC Card 仕様 -----	1- 1
(1-1) REX-5052 の特徴	1- 1
(1-2) 添付品	1- 2
(1-3) REX-5052 の GPIB インターフェイス機能	1- 3
第 2 章 Windows95/98/Me 解説 -----	2- 1
(2-1) インストール	2- 1
(2-2) PC カード設定内容の確認	2- 7
(2-3) アンインストール	2- 9
(2-4) DLL ライブラリ関数仕様	2-11
(2-5) Visual C サンプルプログラム	2-45
(2-6) Visual BASIC サンプルプログラム	2-51
第 3 章 Windows2000/XP 解説 -----	3- 1
(3-1) インストール	3- 1
(3-2) PC カード設定内容の確認	3- 4
(3-3) アンインストール	3- 5
(3-4) DLL ライブラリ関数仕様	3- 7
(3-5) Visual C サンプルプログラム	3-42
(3-6) Visual BASIC サンプルプログラム	3-49
第 4 章 MS-DOS での使用 -----	4- 1
(4-1) イネーブラのインストール	4- 1
(4-2) GPBIOS	4-14
(4-3) MS-DOS 用 C 言語ライブラリ解説	4-37
(4-4) N88Basic での使用	4-70
第 5 章 Windows3.1 DLL ライブラリ関数仕様 -----	5- 1
第 6 章 WindowsNT4.0 解説 -----	6- 1
(6-1) インストール	6- 1
(6-2) DLL 関数仕様	6- 3
(6-3) サンプルプログラム解説	6-28
(6-4) 割込み制御の使用法	6-30
Appendix	
• GPIB とは	

発行 ラトックシステム株式会社
2002年8月30日 第4.0版 第1刷発行

製品に対するお問い合わせ

REX-5052 の技術的なご質問やご相談の窓口を用意していますのでご利用ください。

ラトックシステム株式会社
I&L サポートセンター
〒556-0012
大阪市浪速区敷津東 1-6-14 朝日なんばビル
TEL.06-6633-6741
FAX.06-6633-3553
<サポート受付時間>
月曜 - 金曜 (祝祭日は除く) AM 10:00 - PM 1:00
PM 2:00 - PM 5:00

また、インターネットのホームページでも受け付けています。

HomePage ⇨ <http://www.ratocsystems.com>

🔔 ご注意 🔔

- 本書の内容については、将来予告なしに変更することがあります。
- 本書の内容につきましては万全を期して作成しましたが、万一ご不審な点や誤りなどお気づきになられましたらご連絡願います。
- 本製品および本製品添付のマニュアルに記載されている会社名および製品名は、各社の商品または登録商標です。
- 運用の結果につきましては、責任を負いかねますので、予めご了承ください。

第1章 GPIB PC Card仕様

(1-1) REX-5052 の特徴

REX-5052 GPIB インターフェイスセットは、PCMCIA スロットを持つ DOS/V 機、NEC PC-9800 シリーズのために開発された GPIB 用のインターフェイス PC カードと、それを駆動するためのソフトウェアにより構成され、下記の様な特徴を持っています。

- バイナリ転送モードを持つコマンドを採用しました。メモリと GPIB の間で、高速一括転送が可能です。(最大 300Kbyte/Sec)
- PCMCIA インターフェイスに FPGA デバイスを採用し、高機能、高信頼性を実現しています。
- 強力なサポートソフトウェア、GPBIOS と、MS-C 用ライブラリ、Borland-C 用ライブラリ、Turbo-C 用ライブラリ、GPLIB が添付しております。GPBIOS は、DOS 上で、ソフトウェア割り込みによって使用できますので、アセンブラ言語等によるアプリケーションに最適です。
また、NEC PC-9800 シリーズ用に N88BASIC 用リンクを添付させていますので、N88BASIC 環境で HPL 言語ライクな制御を行うことができます。
- Windows3.1,Windows95/98/Me,Windows2000/XP,WindowsNT4.0 用の GPIB ダイナミックリンクライブラリ (GPLIB**.DLL) をご用意していますので、VisualBasic/VisualC++より GPIB 制御を簡単に行うことができます。(16bit,32bit 用をそれぞれ用意しています。)
- 富士通 FM シリーズで培った実績のある GPIB 回路と数多くの PCMCIA カードを供給した実績を組み合わせ、安定した性能を実現しています。ソフトウェアや、サポート体制の面でも、それらの実績を活かしています。

(1-2) 添付品

REX-5052 は PCMCIA スロットを持つパーソナルコンピュータ(DOS/V,PC-9800 シリーズ等)のための GPIB(IEEE488)インターフェイスセットで下記の製品より構成されています。開梱後、欠品がある場合には、すぐに御連絡ください。

REX - 5052 GPIBインターフェイスPCカード 1枚

パーソナルコンピュータ本体内の PCMCIA スロットに実装します。

添付ソフトウェア書き込み済ディスク(3.5", 1.44MB) 3枚

GPBIOS,GPLIB などの基本ソフトウェア、および応用プログラム例が書き込まれています。

REX - 5052ユーザーズマニュアル 1冊

本書のことで、REX-5052 を使用する上で必要な事項について述べてあります。

GPIB機器接続用ケーブル(1m長) 1本

保証書兼ご愛用者登録ハガキ 1枚

ご愛用者カードは保証書を切り離した後、必要事項を記入の上必ずご返送ください。ご返送頂けない場合、バージョンアップ等のサポートサービスは受けられませんのでご注意ください。



(1-3) REX-5052 の GPIB インターフェイス機能

GPIB には、下記の 10 種類のインターフェイス機能が定められています。そして、実際には、これらの機能のうち必要なものを選択して組合せて使用します。GPIB 用機器やコントローラ(パソコン)を選択する場合には、この機能コードをあらかじめ調べておく必要があります。その機能を持っているかどうかということ、どのレベルまでの機能を持っているかということは、SR0,C4 のような機能シンボルコードと 0~9 の数字の組み合わせで示され、0 は、その機能を持たないことを示します。

機能シンボル コード	インターフェイス 機能	機能
SH	ソースハンドシェイク	バス上のデータを送信する
AH	アクセプタハンドシェイク	バス上のデータを受信する
T	トーカー	SH機能を使って、他の装置にデータを送る
L	リスナ	AH機能を使って、他の装置からデータを受け取る
C	コントローラ	バス上にコマンドを送り出して、GPIB システムをコントロールする
DT	デバイストリガ	トリガコマンドを受信し、装置をトリガする
DC	デバイスクリア	クリアコマンドを受信し、装置をリセットする
PP	パラレルポール	コントローラのパラレルポールに回答する
SR	サービスリクエスト	コントローラに対し SRQ を送り出す
RL	リモートローカル	コントローラからの指令により装置のリモートとローカル状態とを切りかえる

GPIBでは、すべての機器がバスに対して、並列に接続されています。したがってバス上のデータは、L(リスナ)機能をもつ装置であれば同時に受信することができます。しかし送信(バス上へのデータの送り出し)は、必ずどれか一台のみしか行えません。

バス上でデータの衝突(同時に2台以上がトーカーとなる)が発生したり、受信データの指定などを行うために GPIB システムでは、コントローラ(C)機能が用意され各装置にはアドレスが割付けられます。通常のシステムでは、コントローラはバス上に1台のみ存在します。

[REX-5052 の機能表] パソコンをコントローラとしてのみ使用します。

機能	サブセット	内容
SH	SH1	ソースハンドシェイク機能を持つ
AH	AH1	アクセプタハンドシェイク機能を持つ
C	C1	コントローラ機能を持つ
	C2	コントローラインチャージ機能を持つ
	C3	リモートイネーブル機能を持つ
	C4	SRQ に対する応答機能を持つ
	C28	インターフェイスメッセージ送信機能を持つ
T	T8	基本的なトーカ機能を持つ MLA によってトーカ機能が解除される
L	L4	基本的なリスナ機能を持つ MTA によりリスナ機能が解除される
SR	SR0	システムコントローラとしてのみ動作しますので これらの機能はありません。
RL	RL0	
PP	PP0	
DC	DC0	
DT	DT0	

REX-5052GPIB インターフェイスセットは、パソコンをコントローラとして機能させるためのインターフェイスセットで、他のコントローラとの同居はできません。従って REX-5052 と同時に GPIB 上で使用できる機器は、下記の機能を持つ装置に限られます。

- ・ アドレス可能な装置であること。
- ・ コントローラ機能を持たない(C0)こと。

(ATN, IFC, REN ラインの管理機能を持たないこと)

また REX-5052 を実装し GPBIOS または GPLIB、GPLIB32 が動作中のパソコンは、すべてコントローラインチャージ(コントローラとしてバスの制御権を獲得している状態)ですので、GPIB関係のコマンドを実行していなくとも、他のコントローラとバス上での同居はできません。

第2章 Windows95/98/Me 解説

(2-1) インストール

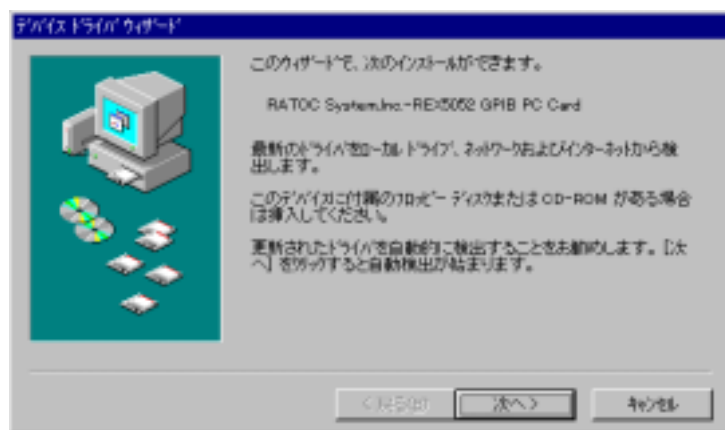
Windows95 OSR-2^(注1) のリリースにより現在 Windows95 のバージョンには、Windows95 OSR-2 と OSR-2 以前のバージョンがあります。「マイコンピュータ」を右クリックし「プロパティ」情報を表示することによりどちらのバージョンがインストールされているか調べることができます。システム情報が「Microsoft Windows95 4.00.950 a」の場合は OSR-2 以前のバージョンになり、OSR-2 の場合は「Microsoft Windows95 4.00.950 B」となります。ご利用の Windows95 が OSR-2 かそれ以前のバージョンによりインストールの方法が異なりますので注意してください。

(注1) OSR-2 (OEM Service Release 2) では FAT32、CardBus 等の新しい機能がサポートされています。

Windows95 OSR-2 でのインストール方法

[1] PC カードの挿入

PC カードをスロットに挿入すると、ハードウェアアウィザードが起動し右のデバイスドライバウィザードのインストールが表示されます。ここでは、次へを選択します。



[2] ドライバファイル場所の指定

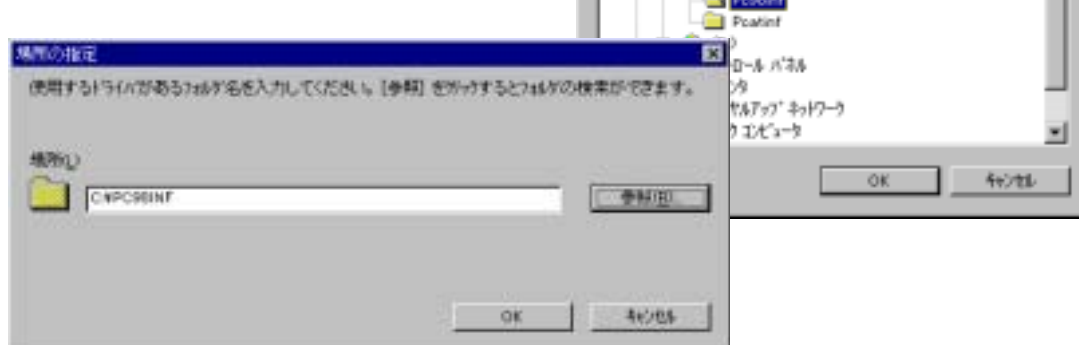
次にドライバファイル (INF ファイル) の場所を指定します。PC/AT にインストールされる場合は、

A:¥PCATINF

PC-98 にインストールされる場合は、

C:¥PC98INF

を設定し次に進んでください。



[3] インストールの完了

インストールが正常に完了した場合は、「このデバイス用に更新されたドライバが見つかりました。」のメッセージが表示されますので確認してください。

以上でインストールは完了です。



Windows95 (OSR-2 以前のバージョン) でのインストール方法

[1] PC カードの挿入

PC カードをスロットに挿入すると、右のハードウェアウィザードが起動します。ここでは「ハードウェアの製造元が提供するドライバ」選択し次に進みます。



[2] 配布ファイルコピー元の指定

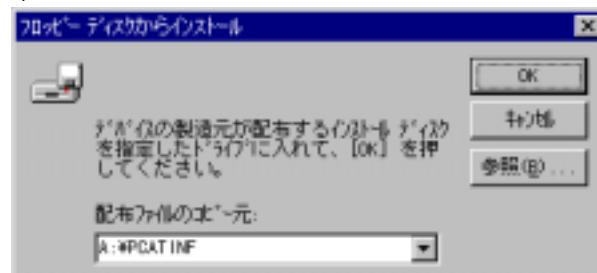
次にドライバーファイル(INF ファイル)の場所を指定します。PC/AT にインストールされる場合は、

A:¥PCATINF

PC-98 にインストールされる場合は、

C:¥PC98INF

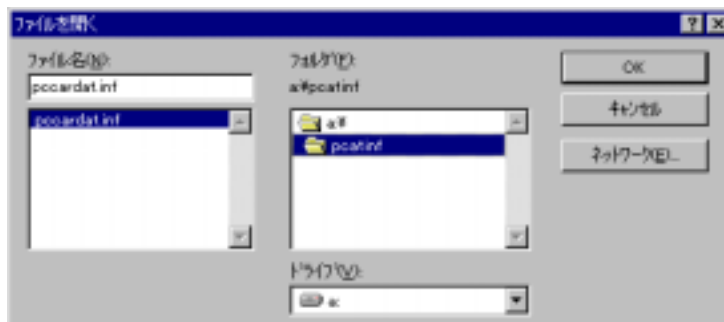
を設定し次に進んでください。



[3] インストールの完了

インストールが正常に行われるとビープ音で完了が通知され、ハードウェアウィザードは自動的に終了します。

以上でインストールは完了です。



Windows98 でのインストール方法

【1】PC カードの挿入

PC カードをスロットに挿入すると、ハードウェアウィザードが起動し右のデバイスドライバーウィザードのインストールが表示されます。ここでは、次へを押します。



ドライバの検索方法は「特定の場所にあるすべてのドライバの一覧を作成し、インストールするドライバを選択する。」を選択し、次へを押します。

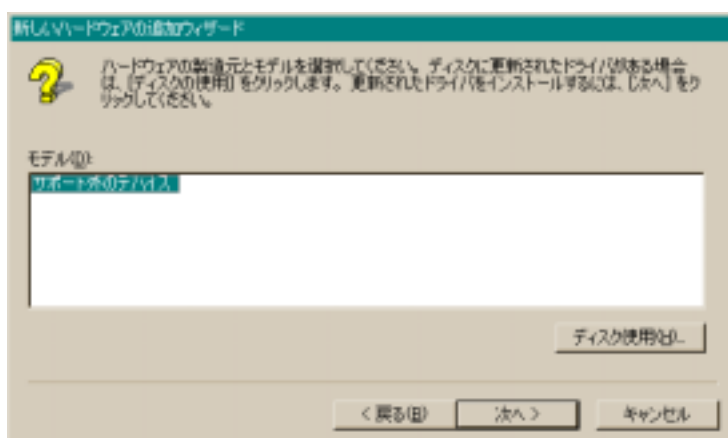


デバイスの種類は「その他のデバイス」または「Otherdevices」を選択し、次へを押します。



[2] ドライバファイル場所の指定

モデルの選択では「ディスク使用」を押します。



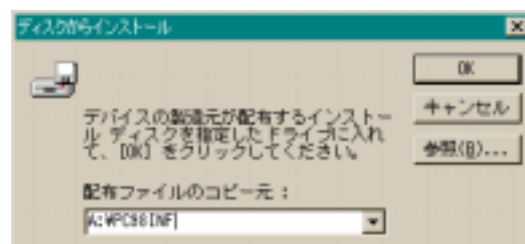
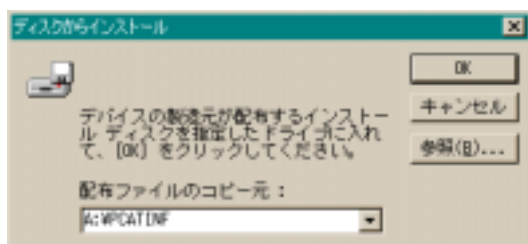
製品添付の Windows95/98/Me 用セットアップディスクをフロッピーディスクドライブに挿入し、次にドライバファイル(INF ファイル)の場所を指定します。PC/AT にインストールされる場合は、

☐ A:¥PCATINF

PC-98 にインストールされる場合は、

☐ C:¥PC98INF

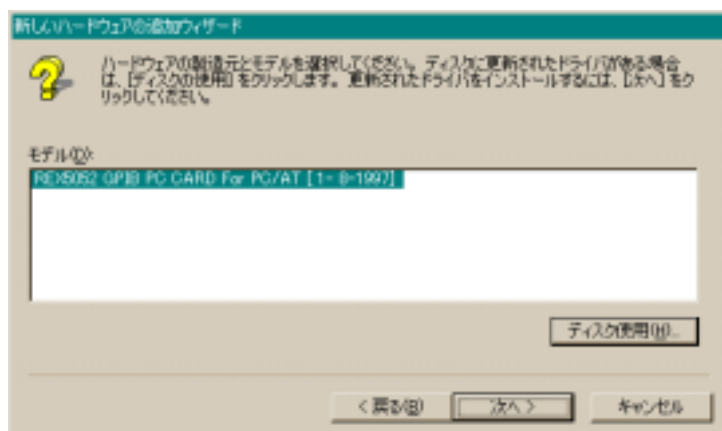
と入力し、OK を押します。



正しいモデル名「REX5052 GPIB PC CARD for PC/AT」が表示されたら、次へを押します。

(注意)

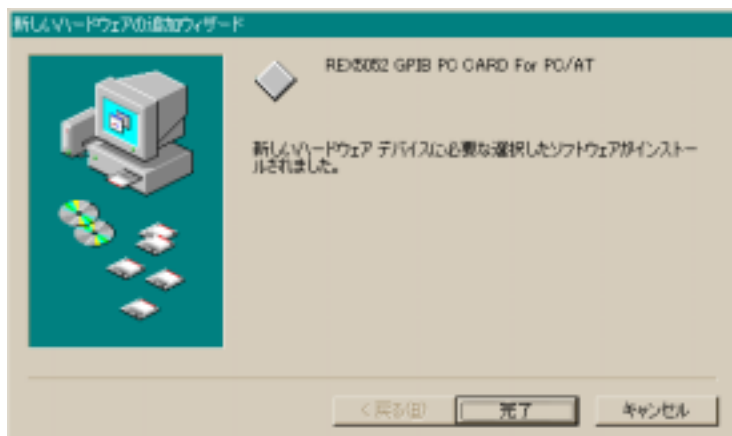
NEC PC9821 シリーズをご利用の場合は、for PC/AT の表示が for PC98 になっていることを確認します。



インストール準備が完了したら、次へを押します。



インストール完了が表示されたら、完了を押してハードウェアウィザードを終了します。



以上でインストールは完了です。

WindowsMe でのインストール方法


【1】PC カードの挿入

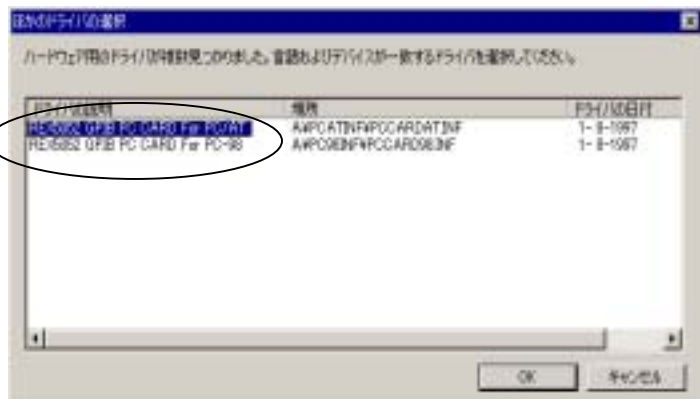
PC カードをスロットに挿入すると、右の新しいハードウェアの追加ウィザードが表示されますので、製品添付の Win95/98/Me 用セットアップディスクをフロッピーディスクドライブへ挿入してください。

次に、「適切なドライバを自動的に検索する（推奨）(A)」を選択し「次へ」ボタンを押します。

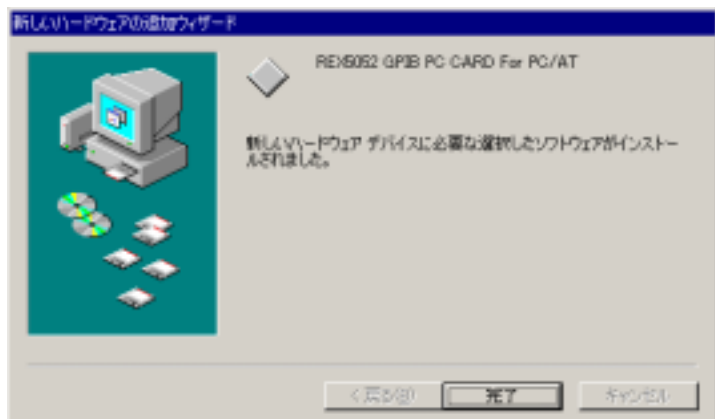


【2】ドライバーファイル場所の指定

右のように、セットアップ情報ファイル（inf ファイル）が、ディスク上から自動的に検索されますので、 「REX-5052 GPIB PC CARD For PC/AT」を選択し、「OK」ボタンを押します。



右の画面が表示されましたら、「完了」ボタンを押します。



以上で、REX-5052 インストールは完了です。

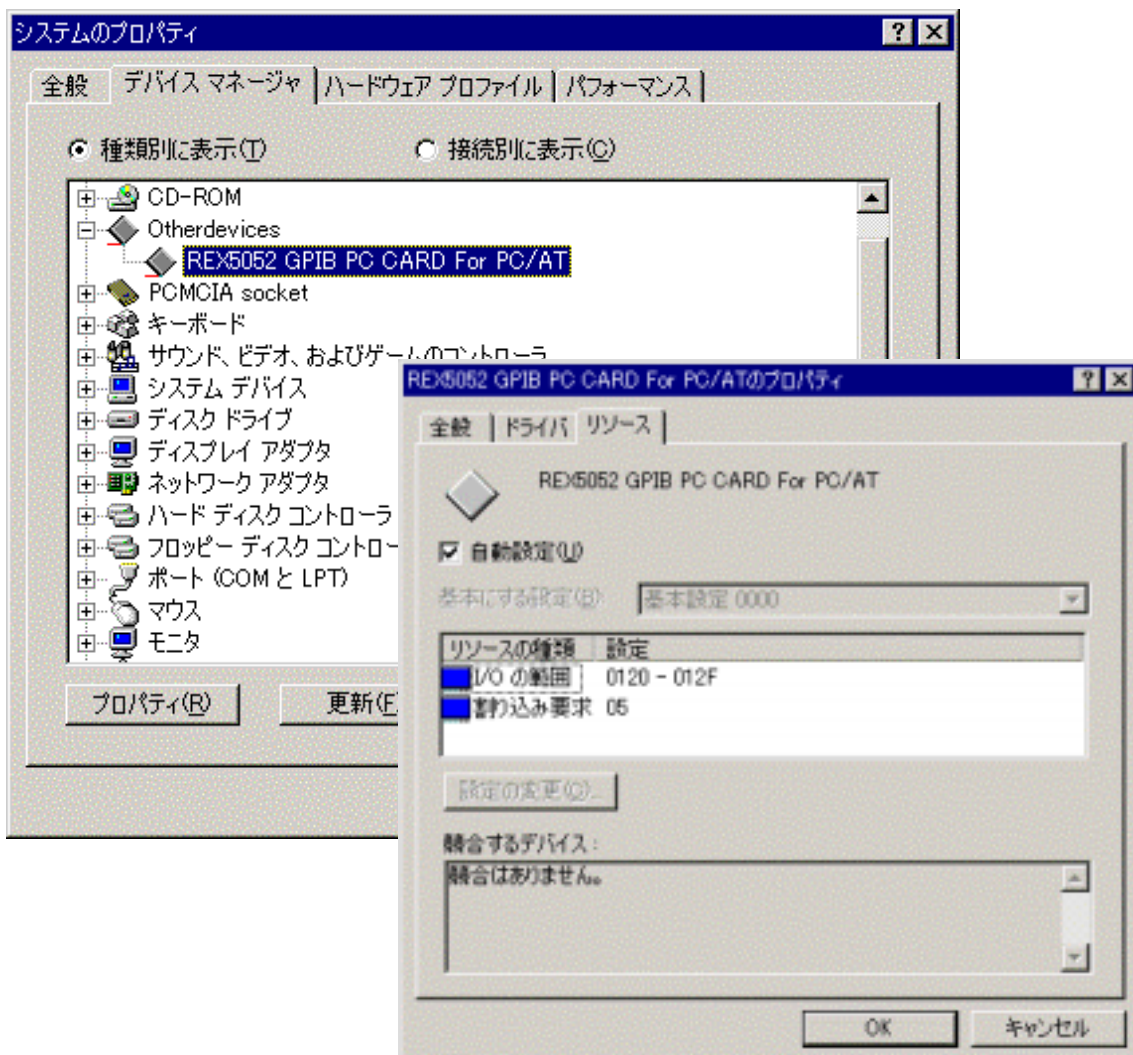
(2-2) PC カード設定内容の確認

システムプロパティの起動

コントロールパネルのシステムを起動し、デバイスマネージャのタブを選択します。カードの設定が正常に行われていれば、コンピュータのレジストリツリー「Otherdevices」の下に「REX5052 GPIB PC CARD For PC/AT(またはPC98)」が登録されます。

プロパティのリソースタブを選択して I/O ポートアドレスおよび IRQ の割り当てで競合していないことを確認してください。

競合がある場合は、次ページの「リソースの変更」を行い、空いているリソースに割り当ててください。

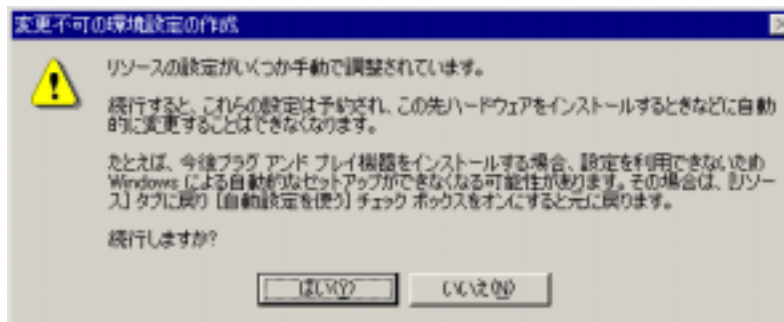


リソースの変更

リソースの変更は、自動設定チェックを外して、基本設定を別の設定に変更し、空いている I/O に割り当てます。割り込み要求は「設定の変更」ボタンを押して変更します。

手動設定を行うと、下図の「変更不可の環境設定の作成」ダイアログが表示されますが、続行「はい」を選択してください。

手動設定したリソースが他のデバイスと競合していなければ、「ピッポッ」というピープ音とともにシステムプロパティ画面に戻ります。もう一度、「REX5052 GPIB PC CARD For PC/AT」のプロパティを確認し、手動設定したリソースが他のデバイスと競合していないことを確認してください。

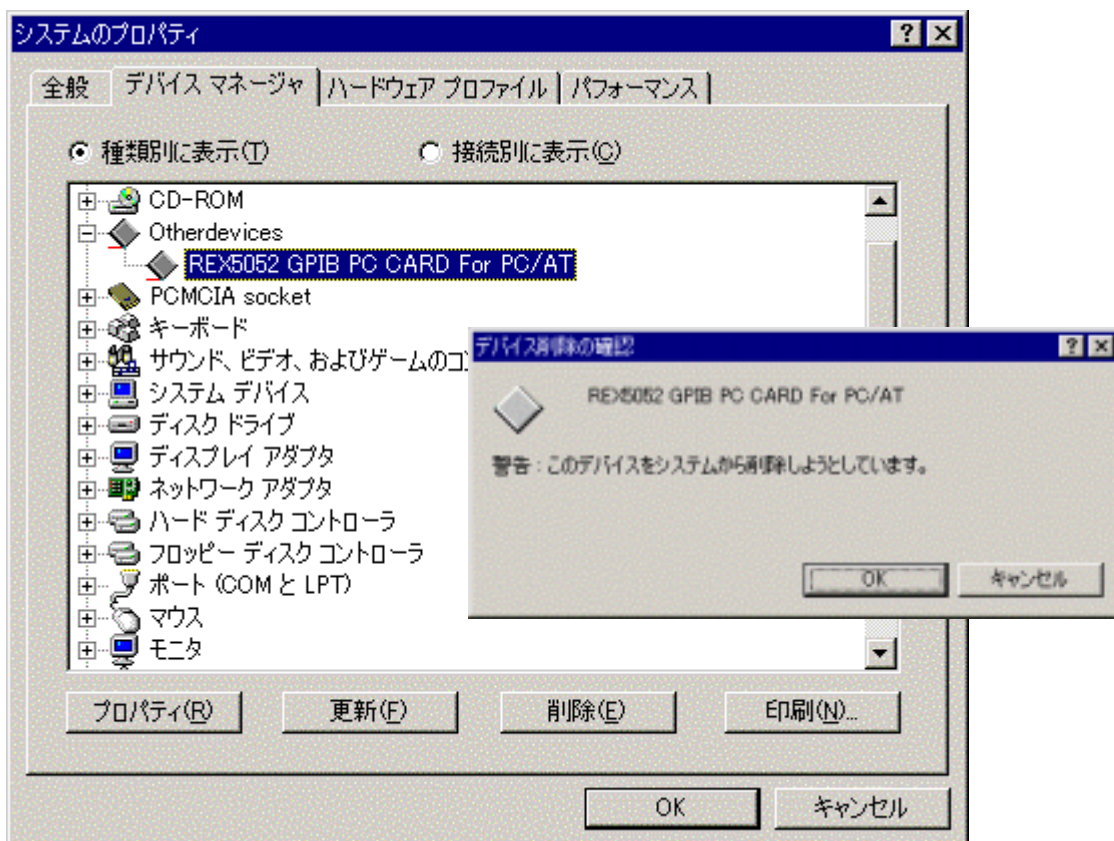


(2-3) アンインストール

カードが正しくインストールされなかった場合は以下の手順でカード情報の削除と INF ファイルの削除を行い、再度、(2-1)の方法でインストールを行ってください。

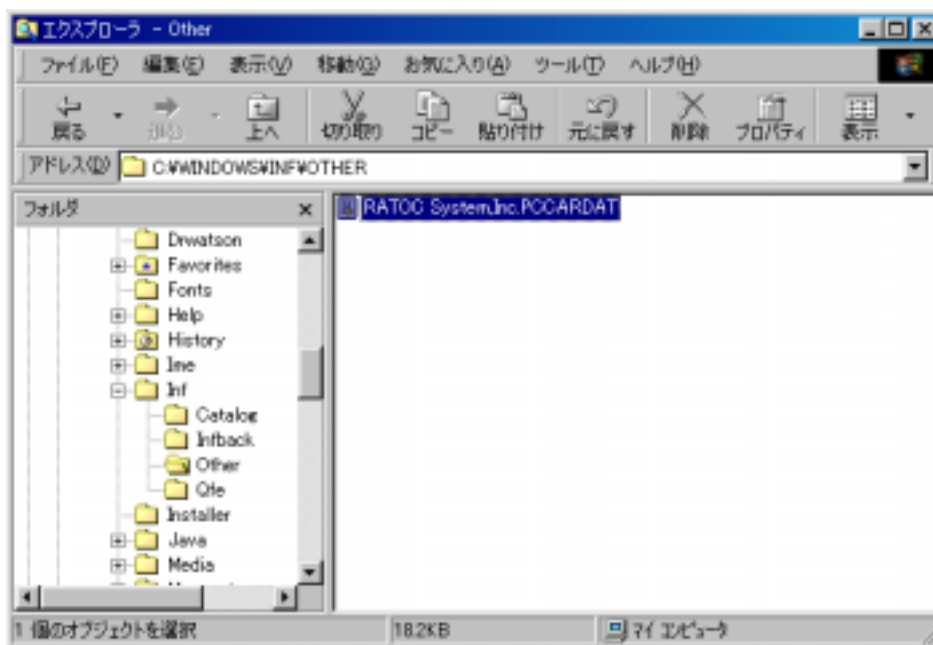
[1] カード情報の削除

コントロールパネルのシステムを起動し、「デバイスマネージャ」のタブを選択します。Otherdevices にある「REX5052 GPIB PC CARD For PC/AT(または PC-98)」を選択して「削除」ボタンを押すとデバイス削除の確認が表示されますので、「OK」を押してください。



[2] INF ファイルの削除

「エクスプローラ」を起動し、¥Windows¥Inf¥Other フォルダにある「RATOC System,Inc.PCCARDAT.INF」ファイルを削除してください。



◆※注意...◆

エクスプローラの設定が「全てのファイルを表示」になっていないとフォルダ「C:\WINDOWS\INF」は表示されません。設定の変更は、エクスプローラメニューの「表示」から「フォルダオプション」を選択して変更します。

(2-4) DLL ライブラリ関数仕様

サンプルプログラムから DLL でイクスポートされている関数を呼び出すためには、以下の3点を行う必要があります。

1. DLL 関数をインポート宣言(Visual C)および Declare 宣言(Visual BASIC)する
2. GPLIB32.LIB をプロジェクトに追加する(Visual C のみ必要)
3. アプリケーションの実行ディレクトリまたは WINDOWS¥SYSTEM に
GPLIB32.DLL ライブラリと VR5052D.VXD ドライバをコピーする

インポート宣言および Declare 宣言の方法については、サンプルプログラムヘッダーファイル GPLIB32.H およびモジュールファイル REX5052.BAS を参照してください。

◆関数仕様の記述について

本ソフトウェアを動作させるための個々のコマンドについて解説を行います。汎例を下記に示します。書式及び実行例は Visual C と Visual BASIC 両方を記述します。

gp_xxx(コマンド名)	機 能
書式	VC > Visual C での関数の記述 VB > Visual BASIC での関数の記述
関連	実行時に関連のあるパラメータ
実行例および動作	そのコマンドの実行例と GPIB 各信号線の動作を示します。

留意点

- すべての関数は INT 型の戻り値を返します。(VOID 型を除く)
- 戻り値は、0 の場合は正常終了です。それ以外はエラーコードです。
- 機器アドレスの指定は文字列で行ないます。(各コマンドの解説では書式の項目で "char *adrs" で示されています。)
このとき、トークン指定が必要なコマンドでは、文字列の先頭の機器アドレスがトークンアドレスとなります。
(例)リスナーアドレス 1,3,4,8 の場合 : adrs = "1,3,4,8"
全機器に対する場合 : adrs = "" (ヌル文字列)
- 引き数に関する注意
Visual BASIC で GPLIB32.DLL を呼び出す場合、値を渡す場合には、ByVal val1 As Integer になります。アドレスを渡す場合には、Val1 As Integer という構文になります。

◆関数一覧

関数	概要	頁
gp_cardinfo	カードのリソース情報を取得	2-13
gp_init	REX-5052 の初期化	2-14
gp_cli	IFC ラインを TRUE にする (約 10msec 間)	2-15
gp_ren	REN ラインを TRUE にする	2-16
gp_clr	デバイスクリアまたはセレクトッドデバイスクリアコマンド送出	2-17
gp_wrt	リスナアドレスで指定された機器にデータ送信	2-19
gp_red	指定した機器からデータをリード	2-21
gp_trg	リスナに指定された機器に対して GET 命令を送信	2-23
gp_strtodbl	8 バイトのデータを格納するメモリへの BYTE 型ポインタを double 型ポインタにキャストする	2-24
gp_strtoflt	4 バイトのデータを格納するメモリへの BYTE 型ポインタを float 型ポインタにキャストする	2-25
gp_tfrin	指定したトーカより指定バイト分データをバッファに格納	2-26
gp_tfrinit	gp_tfrins のトーカ指定を行う	2-28
gp_tfrins	gp_tfrinit で指定した機器から指定バイト数分のデータをバッファ領域内に直接読み込んで格納	2-29
gp_tfrend	gp_tfrinit で指定したトーカ指定の解除	2-29
gp_tfrout	指定した機器へ指定バイト分のデータを転送	2-30
gp_lcl	指定したリスナ機器をローカル状態に設定	2-31
gp_llo	GPIB 上の全機器のローカルスイッチを無効設定	2-33
gp_wtb	ATN ラインを TRUE にしてコマンド文字列を送信	2-34
gp_rds	シリアルポールを実行しステータスバイトを受信	2-35
gp_rds1	シリアルポールを実行しステータスバイトを受信	2-36
gp_wait	指定した時間プログラムの実行を停止	2-37
gp_wsrq	指定時間 SRQ を待つ (ステータスレジスタ 1 を見る)	2-38
gp_wsrqb	指定時間 SRQ を待つ (バスステータスを見る)	2-39
gp_delm	リスナ時トーカ時のデリミタを設定	2-40
gp_tmout	バスタイムアウトパラメータを設定	2-41
gp_setdelay	外部変数 delay_count のディレイ時間を変更	2-42
gp_count	送・受信データ (バイト) 数の取得	2-43
gp_myadr	設定された REX-5052 の GPIB アドレスを取得	2-44

gp_cardinfo

カードのリソース情報を取得

書式

- VC** ➤ int gp_cardinfo(LPWORD pSlotNo, LPWORD pIOBase, LPWORD plrqNo)
- pSlotNo** ➤ カードが挿入されているスロット番号を格納する変数のアドレス
 - IOBase** ➤ I/O リソース情報を格納する変数のアドレス
 - lrqNo** ➤ IRQ リソース情報を格納する変数のアドレス
- VB** ➤ Function gp_cardinfo (pSlotNo As Long, IOBase As Long, lrqNo As Long) As Long
- pSlotNo** ➤ カードが挿入されているスロット番号を格納する変数のアドレス
 - IOBase** ➤ I/O リソース情報を格納する変数のアドレス
 - lrqNo** ➤ IRQ リソース情報を格納する変数のアドレス

関連

なし

実行例および動作 **VC** ▼

```
WORD MyIONo;      // GPIB カード I/O ベースアドレス
WORD MyIrqNo;     // GPIB カード割り込み番号
WORD SlotNo;      // カードが挿入されているスロット番号

ret_val = gp_cardinfo( &SlotNo, &MyIONo, &MyIrqNo );
```

VB ▼

```
Dim UseSlotNo As Long 'カードが挿入されているスロット番号
Dim UseIOAdrs As Long ' GPIB カード I/O ベースアドレス
Dim UseIrqNo As Long ' GPIB カード割り込み番号

retval = gp_cardinfo(UseSlotNo, UseIOAdrs, UseIrqNo)
```

戻り値(10進数)

- 0 : リソース取得正常終了
- 1 : DEVICE I/O コントロールエラー
- 2 : カードサービスドライババージョンエラー
- 3 : GET_CARD_SERVICES_INFO ファクションコールサービスエラー
- 4 : GET_FIRST_TUPLE ファクションコールサービスエラー
- 5 : GET_TUPLE_DATA ファクションコールサービスエラー
- 6 : GET_CONFIG_INFO ファクションコールサービスエラー
- 7 : メモリアロケーションエラー
- 9 : GPIB PC カードが挿入されていない

gp_init

REX-5052 の初期化

書式

VC ➤ int gp_init
 (WORD GpAdrs, WORD IOBase, WORD IrqNo)
GpAdrs ➤ カードの GPIB 機器アドレス
IOBase ➤ I/O ベースアドレス
IrqNo ➤ 割り込み番号

VB ➤ Function gp_init
 (ByVal GpAdrs As Integer, ByVal IOBase As Long,
 ByVal IrqNo As Integer) As Long
GpAdrs ➤ カードの GPIB 機器アドレス
IOBase ➤ I/O ベースアドレス
IrqNo ➤ 割り込み番号

関連

なし

実行例および動作

VC ▼

```
WORD    GpAdrs;           // カードの GPIB 機器アドレス
WORD    IOBase;          // GPIB カード I/O ベースアドレス
WORD    IrqNo;           // GPIB カード 割り込み番号

ret_val = gp_init( GpAdrs, MyIONo, MyIrqNo );
```

VB ▼

```
Dim UseCardAdrs As Integer ' カードの GPIB 機器アドレス
Dim UseIOAdrs As Integer  ' GPIB カード I/O ベースアドレス
Dim UseIrqNo As Integer   ' GPIB カード 割り込み番号

retval = gp_init(UseCardAdrs, UseIOAdrs, UseIrqNo)
```

REX-5052 カード上の GPIB コントローラチップにソフトウェアリセットコマンドを送り、GPIB コントローラを初期化し、マイアドレスをセットします。また、本ライブラリで使用するパラメータを初期化します。

戻り値(10進数)

0 : 正常終了
 60 : デバイスが使用状態にない

gp_cli

IFC ラインを TRUE にする

書式 VC > int gp_cli(void)
 VB > Function gp_cli() As Long

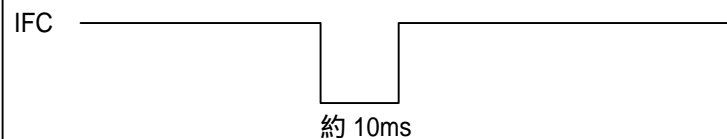
関連 なし

実行例および動作 VC ▾

```
int            gp_error;  
gp_error = gp_cli();
```

 VB ▾

```
Dim gp_error As Long  
gp_error = gp_cli()
```



IFC

約 10ms

REX-5052 カード上の LSI 及び、 GPIB に接続されている全ての機器の初期化を行うために、プログラムの先頭部で必ず一度は IFC コマンドの実行が必要です。必ず正常終了します。

戻り値(10進数) 常に 0 を返します。

gp_ren

REN ラインを TRUE にする

書式 VC > int gp_ren(void)
 VB > Function gp_ren() As Long

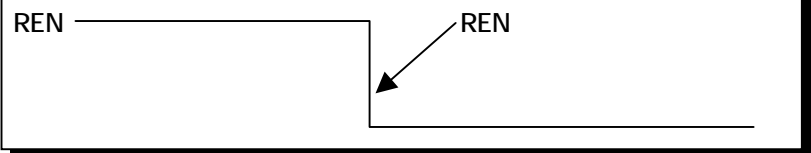
関連 なし

実行例および動作 VC ▼

```
int            gp_error;
gp_error = gp_ren();
```

VB ▼

```
Dim gp_error As Long
gp_error = gp_ren()
```



REN _____

REN コマンドの発行

LCL コマンド(LCL コマンドの項 実行例1を参照)が実行されるか、またはパソコンがリセットされるまでずっと True のままです。GPIB インターフェイスを持つ計測機器や装置は、REN ラインが True になるとリモート可能モードとなり、リモートモードを表示する LED などが点灯します。

REN ラインが False のままですと、GPIB 機器は正しく動作しませんので、プログラム先頭で必ず一度は REN コマンドの実行が必要です。

戻り値(10 進数) 常に 0 を返します。

gp_clr	デバイスクリアまたはセレクトッドデバイスクリアコマンド送出
--------	-------------------------------

書式 VC > int gp_clr(char *adrs)
 adrs > GPIB 機器アドレス
 VB > Function gp_clr(ByVal adrs As String) As Long
 adrs > GPIB 機器アドレス

関連 なし

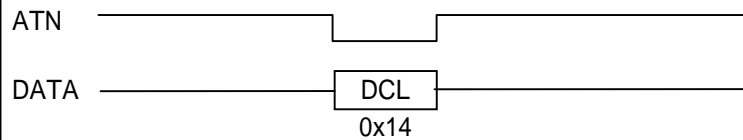
実行例および動作 実行例 1. 全機器に対する場合

VC ▼

```
char        *adrs = "";        // GPIB 機器アドレス
int         ret_val;
ret_val = gp_clr( adrs );
```

VB ▼

```
Dim UseGPIBAdrs As String * 12    ' GPIB 機器アドレス
retval = gp_clr(Str(UseGPIBAdrs))
```



GPIB 上の全機器に対してクリアコマンドを送り、全機器をリセットします。

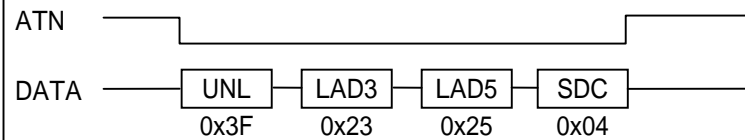
実行例 2. アドレス 3,5 の機器に対して、クリアコマンドを送る場合

VC ▼

```
char    *adrs = "3,5";           // GPIB 機器アドレス
int     ret_val;
ret_val = gp_clr ( adrs );
```

VB ▼

```
Dim UseGPIBAdrs As String * 12  ' GPIB 機器アドレス
UseGPIBAdrs = "3,5"             ' GPIB 機器アドレスをセット
retval = gp_clr(UseGPIBAdrs)
```



相手側機器の DC (DEVICE CLEAR) 機能が DC0 の場合は、このコマンドは無効です。また DC2 の場合は、実行例 2 の SDC コマンドは無効となりますので、実行例 1 を御使用ください。

戻り値 (10 進数)

0 : 正常終了
53 : GPIB バスタイムアウトエラー

gp_wrt

リスナアドレスで指定された機器にデータ送信

書式

VC ➤ int gp_wrt(char *adrs, char *buf)
 adrs ➤ GPIB 機器アドレス
 buf ➤ 送信文字列を格納するバッファアドレス

VB ➤ Function gp_wrt
 (ByVal adrs As String, ByVal buf As String) As Long
 adrs ➤ GPIB 機器アドレス
 buf ➤ 送信文字列を格納するバッファアドレス

関連

タイムアウト, トーカモードデリミタ

実行例および動作

実行例 1. シングルリスナアドレスの場合
 (トーカモードデリミタ = 0)

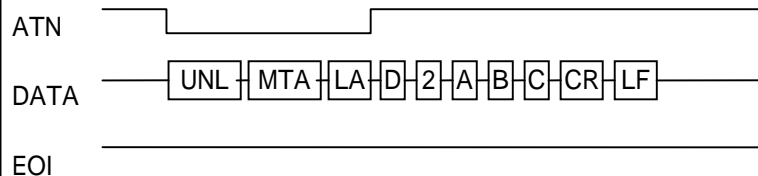
VC ▼

```
char    *adrs = "3";           // GPIB 機器アドレス
char    buf[128];
int     ret_val;
memset( buf,0x00,sizeof(buf) );
strcpy( buf,"D2ABC" );
ret_val = gp_wrt ( adrs , buf );
```

VB ▼

```
Dim UseGPIBAdrs As String * 12   ' GPIB 機器アドレス
Dim StrGPCom As String * 12     ' GPIB コマンド
StrGPCom = "D2ABC"
UseGPIBAdrs = "3"
retval = gp_wrt(UseGPIBAdrs, StrGPCom)
```

アドレス 3 の機器に "D2ABC" という文字列を送信します。



実行例 2. マルチリスナアドレスの場合
(トーカーモードデリミタ = 0x80)

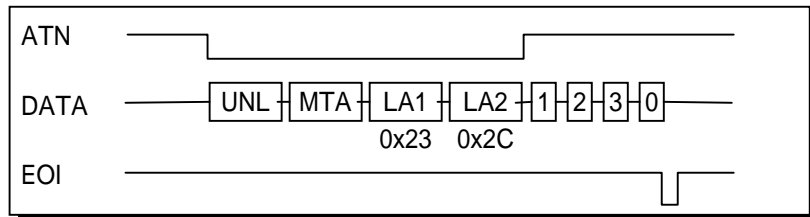
VC ▼

```
char *adrs = "3,12";           // GPIB機器アドレス
char buf[128];
int ret_val;
memset( buf,0x00,sizeof(buf) );
strcpy( buf,"1230" );
ret_val = gp_wrt ( adrs , buf );
```

VB ▼

```
Global UseGPIBAdrs As String * 12 ' GPIB機器アドレス
Global StrGPCom As String * 12 ' GPIBコマンド
StrGPCom = "1230"
UseGPIBAdrs = "3,12"
retval = gp_wrt(UseGPIBAdrs, StrGPCom)
```

アドレス 3,12 の機器に文字列を送信します。



戻り値(10進数) 0 : 正常終了
53 : GPIB バスタイムアウトエラー

gp_red

指定した機器からデータをリード

書式

VC > int gp_red(PSZ adrs, PSZ buf, size_t bufLen)
 adrs > GPIB 機器アドレス
 buf > 受信文字列を格納するバッファアドレス
 bufLen > バッファレングス

VB > Function gp_red
 (ByVal adrs As String, ByVal buf As String, ByVal bufLen As Long) As Long
 adrs > GPIB 機器アドレス
 buf > 受信文字列を格納するバッファアドレス
 bufLen > バッファレングス

注) バッファサイズは受信するバイト数より必ず1バイト以上多く取ってください。

関連

タイムアウト, リスナモードデリミタ

実行例および動作

実行例 1. 相手側機器の送信時デリミタが LF の場合

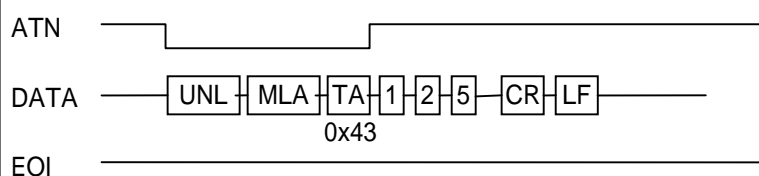
VC ▼

```
char *adrs = "3";           // GPIB機器アドレス
char buf[256];             // GPIB受信バッファ
int ret_val;
ret_val = gp_red( adrs , buf , sizeof(buf) );
```

VB ▼

```
Dim UseGPIBAdrs As String * 12 ' GPIB機器アドレス
Dim Buf As String * 64        ' GPIB受信バッファ
Buf = " " ' 必ず何らかの文字列をいれて初期化
UseGPIBAdrs = "3"
retval = gp_red( UseGPIBAdrs, Buf, 64 )
```

アドレス3の機器よりデータを受信し、文字配列buf内に格納します。



HP 社、横河電機、アドバンテスト等、ほとんどのメーカーが送信時デリミタとして CR,LF を使用していますので、リスナモードデリミタとしては 0x0a(LF)が一般的です。

実行例 2. リスナアドレス付の場合

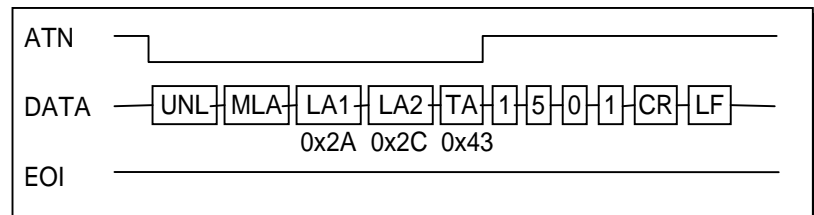
VC ▼

```
char *adrs="3,10,12";           // GPIB 機器アドレス
char buf[10];                   // GPIB 受信バッファ
int ret_val;
ret_val = gp_red( adrs, buf , sizeof(buf) );
```

VB ▼

```
Dim UseGPIBAdrs As String * 12 ' GPIB 機器アドレス
Dim Buf As String * 64         ' GPIB 受信バッファ
Buf = " "                      ' 必ず何らかの文字列で初期化
UseGPIBAdrs = "3,10,12"
retval = gp_red( UseGPIBAdrs, Buf, 64 )
```

アドレス3の機器よりデータを受信し、文字配列 buf 内に格納します。同時にアドレス 10,12 の機器にもデータが送られます。



(注意)

red コマンドは、相手側機器から出力される EOI を検出すると、その時点で読み込み動作を終了します。

戻り値(10進数)

0 : 正常終了
 53 : GPIB バスタイムアウトエラー
 61 : バッファオーバーフロー

gp_trg**リスナに指定された機器に対して GET 命令を送信**

書式

VC ➤ int gp_trg(char *adrs)
 adrs ➤ GPIB 機器アドレス
VB ➤ Function gp_trg(ByVal adrs As String) As Long
 adrs ➤ GPIB 機器アドレス

関連

タイムアウト

実行例および動作

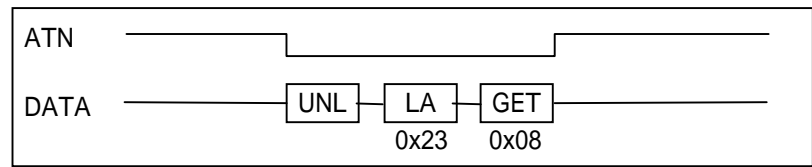
VC ▼

```
char *adrs = "3";           // GPIB 機器アドレス
int  ret_val;
ret_val = gp_trg ( adrs );
```

VB ▼

```
Dim UseGPIBAdrs As String * 12   ' GPIB 機器アドレス
UseGPIBAdrs = "3"
retval = gp_trg( UseGPIBAdrs )
```

アドレス 3 の機器に対して GET 命令を送信します。



戻り値(10進数)

0 : 正常終了
 53 : GPIB バスタイムアウトエラー

gp_strtodbl

8 バイトのデータを格納するメモリへの BYTE 型
ポインタを double 型ポインタにキャストする

書式

VC > void gp_strtodbl(BYTE *bPoint, double *val)
 bPoint > 8 バイトデータを格納するメモリへの BYTE
 型ポインタ
 val > キャストした double 型ポインタ

VB > Sub gp_strtodbl(bPoint As Any, val As Double)
 bPoint > 8 バイトデータを格納するメモリへの BYTE
 型アドレス
 val > キャストした double 型アドレス

関連

タイムアウト

実行例および動作

8 バイトのデータの格納するメモリへ BYTE 型ポインタを受けて
 その 8 バイトのデータを double 型実数に変換します。

VC では、直接キャスト可能であるため、使用する必要はありません。

VC ▼

```
byte  buf[8]; // 8 バイトデータを格納する BYTE 型ポインタ
double data; // キャストした double 型ポインタ
buf[0] = 0x1B;
buf[1] = 0xDE;
buf[2] = 0x83;
buf[3] = 0x42;
buf[4] = 0xCA;
buf[5] = 0XC0;
buf[6] = 0XF3;
buf[7] = 0x3F;
gp_strtodbl(buf,&data);
```

VB ▼

```
Dim ReadBuf(7) As Byte      ' 8 バイトデータを格納するメモリへのアドレス
Dim data As Double        ' キャストした double 型アドレス
buf(0) = &H1B
buf(1) = &HDE
buf(2) = &H83
buf(3) = &H42
buf(4) = &HCA
buf(5) = &HC0
buf(6) = &HF3
buf(7) = &H3F
aa strtodbl ReadBuf(0) data
```

戻り値 (10 進数)

なし

gp_strtoflt

4 バイトのデータを格納するメモリへの BYTE 型
ポインタを float 型ポインタにキャストする

書式

VC > void gp_strtoflt(BYTE *bPoint, float *val)
 bPoint > 4 バイトデータを格納するメモリへの BYTE 型
 ポインタ
 val > キャストした float 型ポインタ
 VB > Sub gp_strtoflt(bPoint As Any, val As Single)
 bPoint > 4 バイトデータを格納するメモリへの BYTE
 型アドレス
 val > キャストした float 型アドレス

関連

タイムアウト

実行例および動作

4 バイトのデータの格納するメモリへ BYTE 型ポインタを受けて
 その 4 バイトのデータを float 型実数に変換します。

VC では、直接キャスト可能であるため、使用する必要はありません。

VC ▼

```
byte  buf[4];      // 4 バイトデータを格納する BYTE 型ポインタ
float  data;      // キャストした float 型ポインタ
buf[0] = 0x52;
buf[1] = 0x06;
buf[2] = 0x9E;
buf[3] = 0x3F;

gp_strtoflt(buf,&data);
```

VB ▼

```
Dim ReadBuf(3) As Byte      ' 4 バイトデータを格納するメモリへのアドレス
Dim data As float          ' キャストした float 型アドレス

buf(0) = &H52
buf(1) = &H6
buf(2) = &H9E
buf(3) = &H3F

gp_strtoflt buf(0), data
```

戻り値(10 進数)

なし

gp_tfrin

指定したトーカーより指定バイト分データをバッファに格納

書式

VC > int gp_tfrin(char *adrs, int bytc, char *buf)

adrs > GPIB 機器アドレス

bytc > 受信バイト数

buf > 受信用配列領域

VB > Function gp_tfrin

(ByVal adrs As String, ByVal bytc As Long, ByVal buf As String) As Long

adrs > GPIB 機器アドレス

bytc > 受信バイト数

buf > 受信用配列領域

関連

タイムアウト

実行例および動作

- 画像処理装置や FFT アナライザなどでは、一度に1～数 Kb のデータを転送する機能を持っていますので、この tfrin を使用するとデータを1度に受信できます。
- 受信バイト数がバッファ変数の長さよりも大きい場合は、バッファ変数分のデータだけ受け取ります。但し受信動作は EOI が来るまで行い、バッファに入り切らない分は捨てられます。またその場合には戻り値として 61(BufferOverflow)を返します。
- 受信バイト数の指定は、整数型変数または符号無し整数型変数で行ってください。

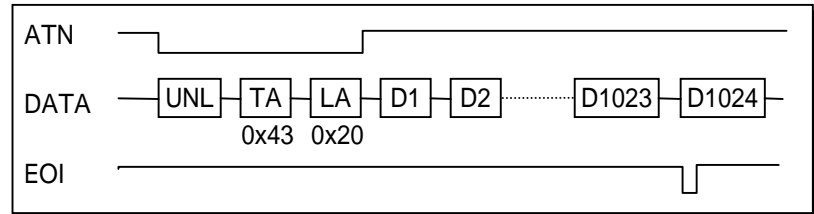
VC ▼

```
char *adrs = "3";           // GPIB機器アドレス
char buf[1025];           // GPIB受信バッファ
int bytc=1024;
int ret_val;
ret_val = gp_tfrin ( adrs, bytc, buf );
```

VB ▼

```
Dim UseGPIBAdrs As String * 12 ' GPIB機器アドレス
Dim Buf As String * 1025      ' GPIB受信バッファ
bytc = 1024
UseGPIBAdrs = "3"
retval = gp_tfrin( UseGPIBAdrs, bytc, buf )
```

トーカーアドレス 3 の機器から 1024 バイトのデータをバッファ変数内に読み込みます。リスナ指定が無い場合は、REN ラインを False にし、GPIB 上の全機器をローカル状態に戻します。



戻り値(10進数) 0 : 正常終了
 53 : GPIB バスタイムアウトエラー
 61 : バッファオーバーフロー

gp_tfrinit

gp_tfrins のトーク指定を行う

書式

VC > int gp_tfrinit(char *adrs)
 adrs > GPIB 機器アドレス

VB > Function gp_tfrinit
 (ByVal adrs As String) As Long
 adrs > GPIB 機器アドレス

関連

gp_tfrins(), gp_tfrend()を続けて呼び出してください。

実行例および動作

VC ▼

```
char *adrs = "3";           // GPIB 機器アドレス
char buf[1025];           // GPIB 受信バッファ
int bytc = 1024;
int ret_val;
ret_val = gp_tfrinit( adrs );
ret_val = gp_tfrins( bytc, buf );
gp_tfrend();
```

VB ▼

```
Global UseGPIBAdrs As String * 12 ' GPIB 機器アドレス
Global Buf As String * 1025      ' GPIB 受信バッファ
bytc = 1024
UseGPIBAdrs = "3"
retval = gp_tfrinit( UseGPIBAdrs )
retval = gp_tfrins(, bytc, buf )
gp_tfrend
```

戻り値(10進数)

0 : 正常終了
 53 : GPIB バスタイムアウトエラー

gp_tfrins	gp_tfrinit で指定した機器から指定バイト数分のデータをバッファ領域内に直接読み込んで格納
------------------	--

書式

VC > int gp_tfrins (unsigned int bytc, char *buf)
 bytc > 受信バイト数
 buf > 受信用配列領域

VB > Function gp_tfrins
 (ByVal bytc As Long, ByVal buf As String) As Long
 bytc > 受信バイト数
 buf > 受信用配列領域

関連 gp_tfrinit()を呼び出した後、gp_tfrins()を呼び出してください。

実行例および動作 *(前頁の gp_tfrinit を参照してください)*

指定バイト数分のデータをバッファ領域内に直接読み込んで格納します。読み込み動作は、指定されたバイト数分で終了するかまたは、EOIを検出した時点で終了します。

戻り値(10進数) 0 : 正常終了
 24 : EOIを受信して終了(正常終了)
 53 : GPIB バスタイムアウトエラー

gp_tfrend	gp_tfrinit で指定したトーカ指定の解除
------------------	---------------------------------

書式

VC > void gp_tfrend(void)
 VB > Sub gp_tfrend()

関連 gp_tfrinit(), gp_tfrins()を呼び出した後、gp_tfrend()を呼び出してください。

実行例および動作 *(前頁の gp_tfrinit を参照してください)*

戻り値(10進数) なし

gp_tfrount

指定した機器へ指定バイト分のデータを転送

書式

VC > int gp_tfrount(char *adrs, int bytc, char *buf)

adrs > GPIB 機器アドレス

bytc > 送信バイト数

buf > 送信用配列領域

VB > Function gp_tfrount

(ByVal adrs As String, ByVal bytc As Long, ByVal buf As String) As Long

adrs > GPIB 機器アドレス

bytc > 送信バイト数

buf > 送信用配列領域

関連

タイムアウト

実行例および動作

- 画像処理装置や FFT アナライザなどへ一度に数 KB のデータを送り込む場合にこの tfrount コマンドを使用します。
- 送信時デリミタとして、EOI が送られます。
- 送信バイト数の指定は、整数型変数または符号無し整数型変数で行ってください。

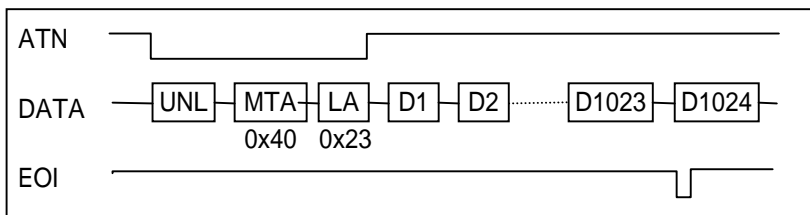
VC ▾

```
char *adrs = "3";           // GPIB 機器アドレス
char buf[1025];            // GPIB 送信バッファ
int bytc;
int ret_val;
bytc = 1024;
ret_val = gp_tfrount( adrs, bytc, buf );
```

VB ▾

```
Dim UseGPIBAdrs As String * 12 ' GPIB 機器アドレス
Dim buf As String * 1025      ' GPIB 送信バッファ
bytc = 1024
UseGPIBAdrs = "3"
retval = gp_tfrount( UseGPIBAdrs, bytc, buf )
```

リスナアドレス 3 の機器へ 1024 バイトのデータを送信します。



戻り値 (10 進数)

- 0 : 正常終了
- 2 : 送信データ設定エラー
- 53 : GPIB バスタイムアウトエラー

gp_lcl

指定したリスナ機器をローカル状態に設定

書式

VC > int gp_lcl(char *adrs)

adrs > GPIB 機器アドレス

VB > Function gp_lcl(ByVal adrs As String) As Long

adrs > GPIB 機器アドレス

関連

タイムアウト

実行例および動作 実行例 1. 全機器に対する場合

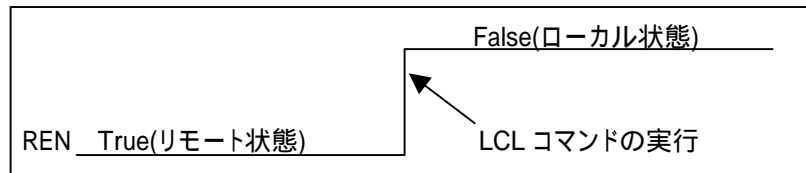
VC ▼

```
char *adrs = ""; // GPIB 機器アドレス
int ret_val;
ret_val = gp_lcl( adrs );
```

VB ▼

```
Dim UseGPIBAdrs As String * 12 ' GPIB 機器アドレス
retval = gp_lcl( Str(UseGPIBAdrs) ) '初期化していない文字列ですと
'先頭に 00h が入っています。
```

GPIB 上の全機器をローカルモードにします。



実行例 2. リスナアドレスの指定がある場合

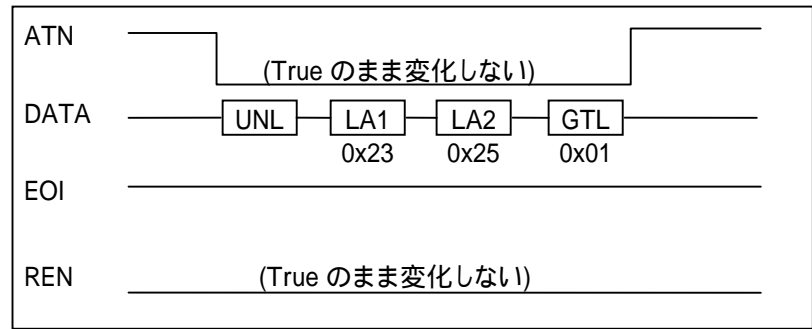
VC ▼

```
char *adrs = "3,5"; // GPIB機器アドレス
int ret_val;
ret_val = gp_lcl( adrs );
```

VB ▼

```
Dim UseGPIBAdrs As String * 12 ' GPIB機器アドレス
UseGPIBAdrs = "3,5" ' GPIB機器アドレスをセット
retval = gp_lcl( UseGPIBAdrs )
```

リスナアドレス 3,5 の機器に GTL (go to local) 命令を送りローカル状態に戻します。



戻り値 (10 進数)

- 0 : 正常終了
- 53 : GPIB バスタイムアウトエラー

gp_llo

GPIB 上の全機器のローカルスイッチを無効設定

書式 VC > int gp_llo(void)
 VB > Function gp_llo() As Long

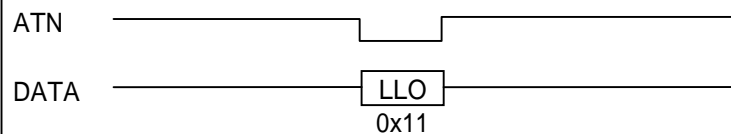
関連 なし

実行例および動作 VC ▾

```
int ret_val;
ret_val = gp_llo();
```

VB ▾

```
Dim retval As Long
retval = gp_llo()
```



- ATN ラインを True にし、LLO 命令を送信した後 ATN ラインを False にします。この命令を受信すると機器側ではパネル上の操作スイッチを無効にします。ただし機器のリモート状態もしくはローカル状態には、変化は生じません。
- 機器の LLO 状態を解除する場合は REN ラインを False にします。(LCL コマンドの実行)

戻り値(10 進数) 0 : 正常終了
 53 : GPIB バスタイムアウトエラー

gp_wtb**ATN ラインを TRUE にしてコマンド文字列を送信**

書式

VC > int gp_wtb(char *buf)

buf > 送信用配列領域

VB > Function gp_wtb(ByVal buf As String) As Long

buf > 送信用配列領域

関連

なし

実行例および動作

VC ▾

```
int ret_val;
char buf[256];
buf[0] = 0x3f;
buf[1] = 0x23;
buf[2] = 0x01;
buf[3] = 0x00;
ret_val = gp_wtb( buf );
```

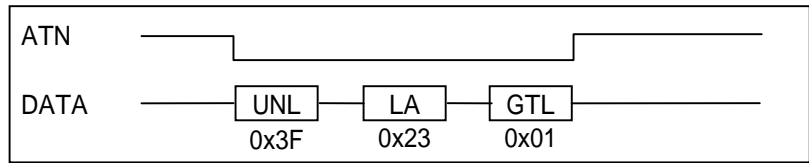
コマンド文字列の最後に、コマンド終了の buf[3] = 0x00 を記述する必要があります。

VB ▾

```
Dim buf As String * 64
buf = chr$(3f)+chr$(23)+chr$(01)+chr$(0)
retval = gp_wtb( buf )
```

コマンド文字列の最後に、コマンド終了の chr\$(0)を記述する必要があります。

LCL3 の実行と同様になります。



戻り値(10進数)

- 0 : 正常終了
- 2 : 送信データ設定エラー
- 53 : GPIB バスタイムアウトエラー

gp_rds

シリアルポールを実行しステータスバイトを受信

書式

VC > int gp_rds(PCHAR adrs, unsigned int *status)
 adrs > GPIB 機器アドレス
 status > GPIB 機器ステータスを返す変数への
 ポインタ

VB > Function gp_rds
 (ByVal adrs As String, status As Long) As Long
 adrs > GPIB 機器アドレス
 status > GPIB 機器ステータスを返す変数への
 メモリアドレス

関連

タイムアウト

実行例および動作

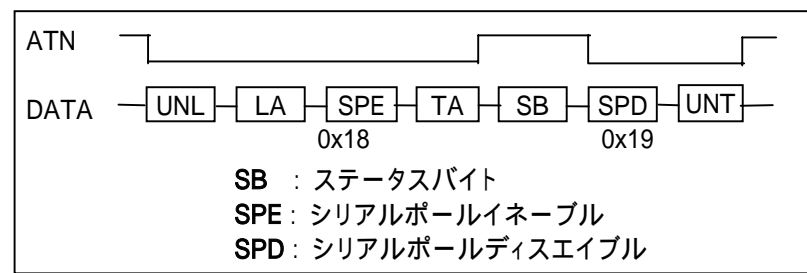
VC ▾

```
char           *adrs = "3";           // GPIB 機器アドレス
unsigned int   status;                // GPIB 機器ステータス
int            ret_val;
ret_val = gp_rds( adrs,&status );
```

VB ▾

```
Dim UseGPIBAdrs As String * 12   ' GPIB 機器アドレス
Dim status As Long                ' GPIB 機器ステータス
UseGPIBAdrs = "3"
retval = gp_rds( UseGPIBAdrs, status )
```

トーカーアドレス 3 の機器に対してシリアルポールを実行し、その機器のステータスバイトを読み込み変数 status に代入する。



SRQ を発信中の機器に対してこのコマンドを実行すると、SRQ ラインが False に復帰します。

戻り値 (10 進数)

0 : 正常終了
 53 : GPIB バスタイムアウトエラー

gp_rds1

シリアルポールを実行しステータスバイトを受信

(注意) gp_rds との違いは、最後に UNT コマンドを送出しない点です。

書式

VC > int gp_rds1(PCHAR adrs, unsigned int *status)
 adrs > GPIB 機器アドレス
 status > GPIB 機器ステータスを返す変数への
 ポインタ

VB > Function gp_rds1
 (ByVal adrs As String, status As Long) As Long
 adrs > GPIB 機器アドレス
 status > GPIB 機器ステータスを返す変数への
 メモリアドレス

タイムアウト

関連

実行例および動作

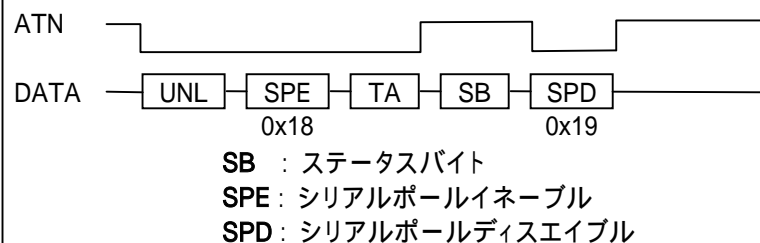
VC ▼

```
char           *adrs = "3";
unsigned int   status;
int            ret_val;
ret_val = gp_rds1( adrs,&status );
```

VB ▼

```
Dim UseGPIBAdrs As String * 12   ' GPIB機器アドレス
Dim status As Long               ' GPIB機器ステータス
UseGPIBAdrs = "3"
retval = gp_rds1( UseGPIBAdrs, status )
```

トーカーアドレス 3 の機器に対してシリアルポールを実行し、その機器のステータスバイトを読み込み変数 status に代入する。



SRQ を発信中の機器に対してこのコマンドを実行すると、SRQ ラインが False に復帰します。

戻り値(10進数)

0 : 正常終了
 53 : GPIB バスタイムアウトエラー

gp_wait

指定した時間プログラムの実行を停止

書式

VC > void gp_wait(unsigned int WaitSecTime)
WaitSecTime > 秒単位のウェイト時間

VB > Sub gp_wait
(ByVal WaitSecTime As Long)
WaitSecTime > 秒単位のウェイト時間

関連

なし

実行例および動作

- 1 WaitSecTime は約1秒です。
- 強制的にプログラムを停止させますのでマウスがきかなくなります。16bit 版からの互換性のために用意された関数です。

VC ▼

```
unsigned int  WaitSecTime = 10; // 待ち時間秒単位で指定
int          ret_val;
ret_val = gp_wait( WaitSecTime );
```

VB ▼

```
Dim WaitSecTime As Long ' 待ち時間秒単位で指定
WaitSecTime = 10
retval = gp_wait( WaitSecTime )
```

10 秒間、プログラムの実行を停止します。

戻り値(10 進数)

なし

gp_wsrq 指定時間 SRQ を待つ (ステータスレジスタ 1 を見る)

書式 VC > int gp_wsrq(unsigned int WaitSecTime)
 WaitMilliSecTime > ミリ秒単位のウェイト時間

 VB > Function gp_wsrq
 (ByVal WaitSecTime As Long) As Long
 WaitMilliSecTime > ミリ秒単位のウェイト時間

関連 なし

- 実行例および動作
- 1 WaitMilliSecTime は1ミリ秒です。
 - このコマンドによって SRQ ラインは変化しません。
 - 時間内に SRQ がなければ-1 を返します

VC ▾

```
unsigned int   WaitMilliSecTime = 10; // 待ち時間秒単位で指定
int           ret_val;
ret_val = gp_wsrq( WaitMilliSecTime );
```

VB ▾

```
Dim WaitMilliSecTime As Long      ' 待ち時間秒単位で指定
WaitSecTime = 10
retval = gp_wsrq( WaitMilliSecTime )
```

SRQ がくるまで 10 秒間待ちます。

戻り値(10進数) 0 :SRQ 正常受信
 -1 :タイムアウト

gp_wsrqb

指定時間 SRQ を待つ (バスステータスを見る)

書式

VC > int gp_wsrqb(int WaitSecTime)
WaitMilliSecTime > ミリ秒単位のウェイト時間

VB > Function gp_wsrqb
(ByVal WaitSecTime As Long) As Long
WaitMilliSecTime > ミリ秒単位のウェイト時間

関連

なし

実行例および動作

- 1WaitMilliSecTime は1ミリ秒です。
- このコマンドによって SRQ ラインは変化しません。
- 時間内に SRQ がなければ-1 を返します

VC ▼

```
unsigned int   WaitMilliSecTime = 10; //待ち時間秒単位で指定
int            ret_val;
ret_val=gp_wsrqb( WaitMilliSecTime );
```

VB ▼

```
Dim WaitSecTime As Long           ' 待ち時間秒単位で指定
WaitSecTime = 10
retval = gp_wsrqb( WaitMilliSecTime )
```

SRQ がくるまで 10 秒間待ちます。

戻り値(10 進数)

0:SRQ 正常受信
-1:タイムアウト

gp_delm

リスナ時トーカー時のデリミタを設定

書式

VC > int gp_delm(char *mode, unsigned int delm)

mode > (以下参照)

delm > (以下参照)

VB > Function gp_delm

(ByVal mode As String, ByVal delm As Long) As Long

mode > (以下参照)

delm > (以下参照)

関連

タイムアウト

実行例および動作

mode は "t", "l" のどれか一文字とし、次の意味を持ちます。

"t" : トーカー時の送信デリミタを指定します。

"l" : リスナ時の受信デリミタを指定します。

delm は 0 ~ 255 (0x00 ~ 0xff) の範囲の値で mode により次の意味をもちます。

"t" : デリミタコードは bit6 ~ bit0 の 7bit で設定します。

この時、bit7 を 1 にすると EOI を出力します。

delm = 0 とした場合は CR+LF が設定されます。

"l" : デリミタコードは bit7 ~ bit0 の 8bit で設定します。

変更されたデリミタは、次にこのコマンドによって変更されるまで有効です。

デフォルト状態では、トーカーモードデリミタは 0 (CR+LF) に、リスナモードデリミタは 0x0a (LF) に設定されています。

リスナモードデリミタとして LF を設定します。

VC ▼

```
char          *mode = "l";          // モード
unsigned int  delm = 0x0a;          // デリミタ
int           ret_val;
ret_val = gp_delm( mode, status );
```

VB ▼

```
Dim GPIBMode As String * 2          ' モード
Dim delm As Long                    ' デリミタ
GPIBMode = "l"
delm = &h0a
retval = gp_delm( GPIBMode, delm )
```

戻り値(10進数)

0 : 正常終了

53 : GPIB バスタイムアウトエラー

gp_tmout

バスタイムアウトパラメータを設定

書式

VC > int gp_tmout(unsigned int SecTime)
 SecTime > 秒単位のタイムアウト時間

VB > Function gp_tmout
 (ByVal SecTime As Long) As Long
 SecTime > 秒単位のタイムアウト時間

関連

なし

実行例および動作

- 1SecTime は1秒です。
- タイムアウトは1バイトのハンドシェイクに対し設定されます。
- デフォルト値は 10 秒です。
red/wrt 等のコマンド実行時のバスタイムアウトを 3 秒に設定します。

VC ▼

```
int ret_val;  
ret_val = gp_tmout( 3 );
```

VB ▼

```
Dim retval As Integer  
retval = gp_tmout( 3 )
```

戻り値(10進数)

0 : 正常終了
53 : GPIB バスタイムアウトエラー

gp_setdelay

外部変数 delay_count のディレイ時間を変更

書式

VC > int gp_setdelay(int DelayTime)
DelayTime > 0.8 μ sec 単位のディレイ時間

VB > Function gp_setdelay
(ByVal DelayTime As Long) As Long
DelayTime > 0.8 μ sec 単位のディレイ時間

関連

なし

実行例および動作

- デフォルトでは、 $625 \times 0.8 \mu \text{sec} = 500 \mu \text{sec}$ になっています。

VC ▼

```
int    ret_val;  
ret_val = gp_setdelay( 500 );
```

VB ▼

```
Dim retval As Integer  
retval = gp_setdelay( 500 )
```

戻り値(10進数)

ダミーで引数をそのまま返します。

gp_count

実際に送・受信したデータ数(バイト数)の取得

書式 VC > int gp_count(void)
 VB > Function gp_count() As Long

関連 なし

実行例および動作 VC ▼

```
int    ret_val;  
ret_val = gp_count();
```

 VB ▼

```
Dim retval As Integer  
retval = gp_count()
```

gp_red(), gp_tfrin(), gp_tfrins(), gp_wrt(), gp_tfrout()を実行後、gp_count()の呼び出しで実際に送・受信したデータ数(バイト数)を返します。

(注意)

gp_red ではデミリタをバッファ内に入れていないため1バイト少ない値を返します。

戻り値(10進数) 送信または受信バイト数を返します。

gp_myadr**設定された GPIB マイアドレスの値をリード**

書式

VC > int gp_myadr(void)

VB > Function gp_myadr() As Long

関連

なし

実行例および動作

互換性を確保する関数ですので、プログラムで新たに自分の機器アドレスを知る必要がない場合は実行する必要はありません。

VC ▼

```
int da;  
da = gp_myadr();
```

VB ▼

```
da = gp_myadr()
```

戻り値(10進数)

GPIB 機器アドレスを返します

(2-5) Visual C サンプルプログラム

Visual C 4.0 以上のバージョンで、本製品に添付されている“GPLIB32.DLL”のライブラリを使って REX-5052 を制御するアプリケーションを開発する場合は、サンプルプログラム“REX5052.C”を参考にしてください。

アプリケーションプログラムから“GPLIB32.DLL”を呼び出すためには、以下のインストレーションを行ってください。

- アプリケーションプログラムに“GPLIB32.H”ファイルをインクルードする。
- アプリケーションプログラムのプロジェクトファイルに GPLIB32.LIB を追加する。
- “C:\WINDOWS\SYSTEM”に、GPLIB32.DLL ライブラリと VR5052D.VXD ドライバーをコピーする。

(注意)

“GPLIB32.DLL”を呼び出しに必要となるインポート宣言、ライブラリ定数等の宣言を“GPLIB32.H”ヘッダーファイルで行っています。アプリケーション作成の際は“GPLIB32.H”ヘッダーファイルの内容を理解してください。

本製品には

★HP 社のデジタルマルチメータ (HP3478A) を制御するサンプルプログラムが添付されています。

次頁より、サンプルプログラムについて解説いたします。

★ HP3478A 制御プログラム

- HP3478Aの GPIBアドレスは3に設定しています。
- 接続計測器 HP3478A :
ヒューレットパッカード
デジタルマルチメータ



(操作方法)

最初に、機器側で設定されている GPIB 機器アドレスをエディットボックスに入力します。

イニシャライズボタンを押して REX-5052 の初期化を行います。

計測開始ボタンで 10 秒間バスラインをよみます。SRQ を調べ信号がきたときのバスラインの計測値を表示します。

■ サンプルプログラム抜粋

- **gp_cardinfo()**により REX-5052 のカードリソース情報を取得します。

```
LRESULT CALLBACK DlgProcHP3478A( HWND hDlg, UINT message, UINT wParam, LONG lParam)
{
    switch( message )
    {
        case WM_INITDIALOG:
        {
            BOOL Status;

            /* GPIB 機器アドレスのデフォルト値設定 */
            sprintf( szTmp, "%d", 3 );
            SetDlgItemText( hDlg, IDC_EDIT_GPIBADDRS, szTmp );

            /* スロットに挿入されている REX5052 GPIB カードのリソース情報を取得する */
            Status = gp_cardinfo( &SlotNo, &MyIOBase, &MyIrqNo );
            if ( Status == 0 )
            {
                /* リソース情報を表示する */
                sprintf( szTmp, "%x", MyIOBase );
                SetDlgItemText( hDlg, IDC_IOADRS, szTmp );
                return TRUE ;
            }
            SetDlgItemText( hDlg, IDC_IOADRS, "Fail Auto Detect" );
            return TRUE ;
        }
        case WM_COMMAND:
            switch( wParam )
            {
                case IDC_BUTTON_INIT:
                    if ( GetGPIBAdrs( hDlg ) != TRUE )
                        return TRUE ;
                    InitGPBiosForHP3478A( hDlg );
                    return TRUE ;
                case IDOK:
                    GetDataFromHP3478( hDlg );
                    return TRUE ;
                case IDCANCEL:
                    EndDialog( hDlg, TRUE );
                    return TRUE ;
                default:
                    return TRUE ;
            }
            break ;
    }
    return FALSE ;
}
```

- GPIB 機器アドレスを取得し GPIBAdrs にセットします。GPIB 機器アドレスが正しく設定されていない場合は、メッセージボックスを出します。

```
BOOL GetGPIBAdrs( HWND hDlg )
{
    UINT    cbText;          // アイテムコントロールが返してきたバイト数
    int i;

    // GPIB 機器アドレスアイテムボックスから入力文字列を取得
    cbText = GetDlgItemText( hDlg, IDC_EDIT_GPIBADRS, szTmp, 64 );
    if ( cbText > 2 || atoi( szTmp ) > 30 )
    {
        sprintf( szTmp, "GPIB 機器アドレスが正しく設定されていません", NULL );
        MessageBox( hDlg, szTmp, NULL, MB_OK|MB_ICONEXCLAMATION );
        return FALSE;
    }
    // GPIB 機器アドレスセット
    for ( i = 0; szTmp[i] != 0x00 ; i++ )
        HP3478GPIBAdrs[i] = szTmp[i] ;
    return TRUE;
}
```

- `gp_init()`で `GPLIB32.DLL` ライブラリを初期化し、`gp_clr()`で GPIB 機器に対してクリアコマンドを送り、機器をリセットします。

```
int InitGPBiosForHP3478A( HWND hDlg )
{
    int MyAdrs;

    if( gp_init( MyGPIBAdrs, MyIOBase, MyIrqNo ) != 0 )
    {
        sprintf( szTmp, "GP-IB DLL の初期化ができません", NULL );
        MessageBox( hDlg, szTmp, NULL, MB_OK|MB_ICONEXCLAMATION );
        return -1;
    }

    MyAdrs = gp_myadr();
    sprintf( szTmp, "%d", MyAdrs );
    SetDlgItemText( hDlg, IDS_MYGPIBADRS, szTmp );

    /* IFCラインを TRUE にする */
    gp_cli();

    /* RENラインを TRUE にする */
    gp_ren();

    /* デバ`イスクリアコマンド`送出 */
    if ( gp_clr( HP3478GPIBAdrs ) != 0 )
    {
        MessageBox( hDlg, "GP-IB デバ`イスクリアコマンド`失敗", NULL, MB_OK|MB_ICONEXCLAMATION );
        return -1;
    }

    /* HP3478A GPIB コマンド`送信 */
    if( gp_wrt( HP3478GPIBAdrs, "HOKM01" ) != 0 )
    {
        MessageBox( hDlg, "GP-IB デバ`イスクリアコマンド`失敗", NULL, MB_OK|MB_ICONEXCLAMATION );
        return -1;
    }

    return 0;
}
```


- HP3478A から受信したデータをダイアログ画面上に表示します。

```
void GetDataFromHP3478( HWND hDlg )
{
    INT          RcvBytes;
    INT          RetCode;

    /* 受信バッファクリア */
    memset( RcvData, 0x00, sizeof( RcvData ) );

    /* トリガコマンド実行 */
    gp_trg( HP3478GPIBAdrs );

    /* データ受信 */
    gp_wsrq( 10 );
    if( ( RetCode = gp_rds( HP3478GPIBAdrs, &GpStatus ) ) != 0 )
    {
        sprintf( szTmp, "ステータスエラー: %d", RetCode );
        SetDlgItemText( hDlg, IDC_DATA, szTmp );
    }

    /* GPIBバスからデータをリード */
    gp_red( HP3478GPIBAdrs, RcvData, sizeof( RcvData ) );

    /* CR, LF を加って表示 */
    RcvBytes = strlen( RcvData );
    RcvData[ RcvBytes - 2 ] = 0x00;
    SetDlgItemText( hDlg, IDC_DATA, RcvData );
}
```

(2-6) Visual BASIC サンプルプログラム

本製品には 32 ビットアプリケーション開発に必要となる API インターフェースを提供する DLL ライブラリ“GPLIB32.DLL”と“VR5052D.VXD”仮想デバイスドライバーが添付されています。32 ビットバージョン Visual BASIC で作成したアプリケーションは“GPLIB32.DLL”の API を呼び出しが必要になります。

Visual BASIC でアプリケーションを作成する場合、次の二つの内容について理解し、必要となる設定作業を行ってください。

Step.1 => 本製品添付ソフトのコピー

32 ビット版 DLL : GPLIB32.DLL 及び仮想デバイスドライバー : VR5052D.VXD を添付のフロッピーディスクからコピーします。

```
>COPY “北°-元ドライブ名”:¥Win95¥DI132¥GPLIB32.DLL “北°-先ドライブ名”:¥Windows¥System  
>COPY “北°-元ドライブ名”:¥Win95¥DI132¥VR5052D.VXD “北°-先ドライブ名”:¥Windows¥System
```

Step.2 => DLL ライブラリ関数の Declare 宣言

Visual BASIC から“GPLIB32.DLL”が提供する API 関数を呼び出すためにはモジュール定義ファイルで各 API 関数を Declare 宣言します。API 関数の Declare 宣言は、製品添付のサンプルプログラム“REX5052.BAS”からモジュール定義ファイルにコピーしてください。また、各 API 関数の仕様については「2-3. ライブラリ関数仕様」を参照してください。

◆ HP3478A 制御プログラム

本製品添付のサンプルプログラムは HP 社のデジタルマルチメータ (HP3478A) を制御するものです。

(注意点)

- HP3478Aの GPIBアドレスは3に設定しています。
- 接続計測器 HP3478A :
ヒューレットパッカード
デジタルマルチメータ



(操作方法)

最初に、機器側で設定されている GPIB 機器アドレス、REX-5052 アドレスをエディットボックスに入力します。

イニシャライズボタンを押して REX-5052 の初期化を行います。

計測開始ボタンで 10 秒間バスラインをよみます。SRQ を調べ信号がきたときのバスラインの計測値を表示します。

サンプルプログラム抜粋

- `gp_cardinfo()`により REX-5052 のカードリソース情報を取得します。

```
Private Sub Form_Load()
    Dim Ret As Long

    ' REX-5052 GPIB PC Card リソース情報取得
    Ret = gp_cardinfo(MySlotNo, MyIOAdrs, MyIrqNo)
    If Ret = 0 Then
        LabelSlotNo.Caption = Str(MySlotNo)
        LabelIOAdrs.Caption = " " + Hex(MyIOAdrs) + "h"
        LabelIrqNo.Caption = Str(MyIrqNo)
        CommandInit.Enabled = True
    Else
        LabelIOAdrs.Caption = "No Card !"
        LabelIrqNo.Caption = ""
        LabelSlotNo.Caption = ""
        CommandInit.Enabled = False
    End If

    ' 制御パラメータ初期設定
    CardGpAdrs = 0
    LabelCardGpAdrs.Caption = Str(CardGpAdrs)
    TextGpAdrs.Text = "3"
    CommandOK.Enabled = False
End Sub
```

- GPIB 機器アドレスを取得します。次に `gp_init()` で `GPLIB32.DLL` ライブラリを初期化し、`gp_clr()` で GPIB 機器に対してクリアコマンドを送り、機器をリセットします。

```
Private Sub CommandInit_Click()  
    Dim Dummy As Integer  
    Dim retval As Integer  
  
    ' HP3478A の GPIB アドレス設定値を取得  
    Call GetParam  
  
    LabelDataOut.Caption = ""  
  
    ' REX-5052 PC Card の初期化  
    retval = gp_init(CardGpAdrs, MyIOAdrs, MyIrqNo)  
    If retval <> 0 Then  
        Dummy = MsgBox("REX-5052 GPIB PC Card の初期化ができません.", vbCritical, "エラー  
")  
        Exit Sub  
    End If  
  
    retval = gp_cli()  
    retval = gp_ren()  
  
    retval = gp_clr(Str(GPIBAdrs))  
    If retval <> 0 Then  
        Dummy = MsgBox("デハ`イスクリアコマンド`送信エラー.", vbCritical, "エラー")  
        Exit Sub  
    End If  
  
    StrGPCom = "HOKM01"  
    retval = gp_wrt(Str(GPIBAdrs), StrGPCom)  
    If retval <> 0 Then  
        Dummy = MsgBox("HP3478A にデ`タ計測コマンド`送信エラー.", vbCritical, "エラー")  
        Exit Sub  
    End If  
    CommandOk.Enabled = True  
  
End Sub
```

- HP3478A から受信したデータをダイアログ画面上に表示します。

```
'  
' 計測データ受信と表示  
'  
Private Sub CommandOK_Click()  
    Dim retval As Integer  
    Dim status As Long  
    Dim Value As Double  
  
    Call GetParam  
  
    ' トリガコマンド実行  
    retval = gp_trg(Str(GPIBAdrs))  
  
    ' シリアルポート実行  
    status = 0  
    While status <> &H41  
        ' ステータスポート受信  
        retval = gp_rds(Str(GPIBAdrs), status)  
    Wend  
  
    ' HP3478A から計測データ受信  
    retval = gp_red(Str(GPIBAdrs), RcvData, 64)  
  
    ' データ表示  
    Value = Val(RcvData)  
    LabelDataOut.Caption = Format(Value, "0.0000") & " Volt"  
End Sub
```

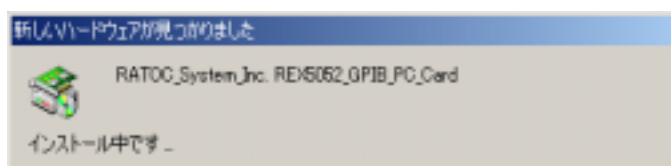
第3章 Windows2000/XP解説

(3-1) インストール

Windows2000 でのインストール方法

【1】PC カードの挿入

PC カードを挿入すると「ハードウェアウィザード」が起動し(右下画面)、インストールが開始されます。「RATOC_System_Inc. REX5052_GPIB_PC_Card」と表示されているかを確認し、以下の手順でインストールを行ってください。



「新しいハードウェアの検索ウィザードの開始」で「次へ(N)>」ボタンを押します。



「ハードウェアデバイスドライバのインストール」では「デバイスに最適なドライバを検索する(推奨)(S)」にチェックを入れて「次へ(N)>」ボタンを押します。

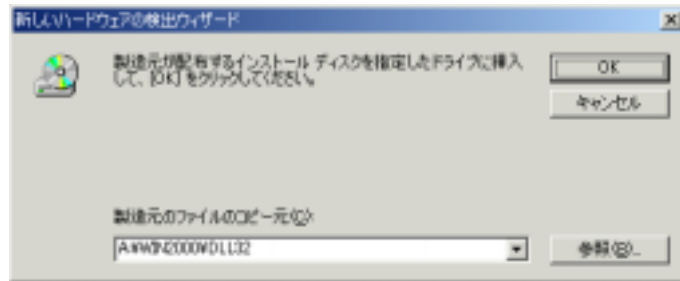


「ドライバファイルの特定」で「場所を指定(S)」にチェックを入れて「次へ(N)>」ボタンを押します。



[2] inf ファイル場所の指定

製品添付の Windows2000/XP 用ディスクをフロッピーディスクドライブに挿入します。製造元のファイルのコピー元(C)で inf ファイルの場所を指定し、「OK」ボタンを押します。



「ドライバファイルの検索」では、ディスク上より右画面のように inf ファイルが検索されますので「次へ(N)>」ボタンを押します。



「新しいハードウェアの検出ウィザードの完了」で「REX502.SYS for REX502 GPIB PC CARD」が表示されます。「完了」ボタンを押してください。

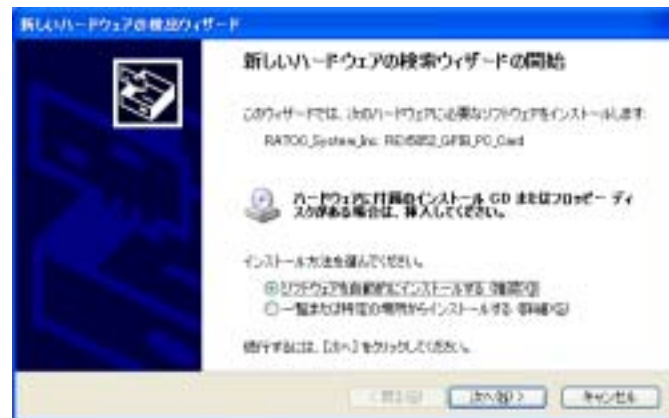
以上で、REX-502 のインストールは終了です。



WindowsXP でのインストール方法

PC カードを挿入すると「ハードウェアウィザード」が起動し、インストールが開始します。以下の手順でインストールを行って下さい。

製品添付の Windows2000/XP 用ディスクをフロッピーディスクドライブに挿入し、「新しいハードウェアの検索ウィザードの開始」で「ソフトウェアを自動的にインストールする(推奨)(I)」にチェックを入れて「次へ(N)>」ボタンを押して先へ進みます。



セットアップ情報ファイル(.inf ファイル)が、ディスク上から検索され、自動的にインストールが行われます。



「新しいハードウェアの検索ウィザードの完了」で「REX5052.SYS for REX5052 GPIB PC CARD」が表示されます。

「完了」ボタンを押してください。

以上で、REX-5052 のインストールは終了です。

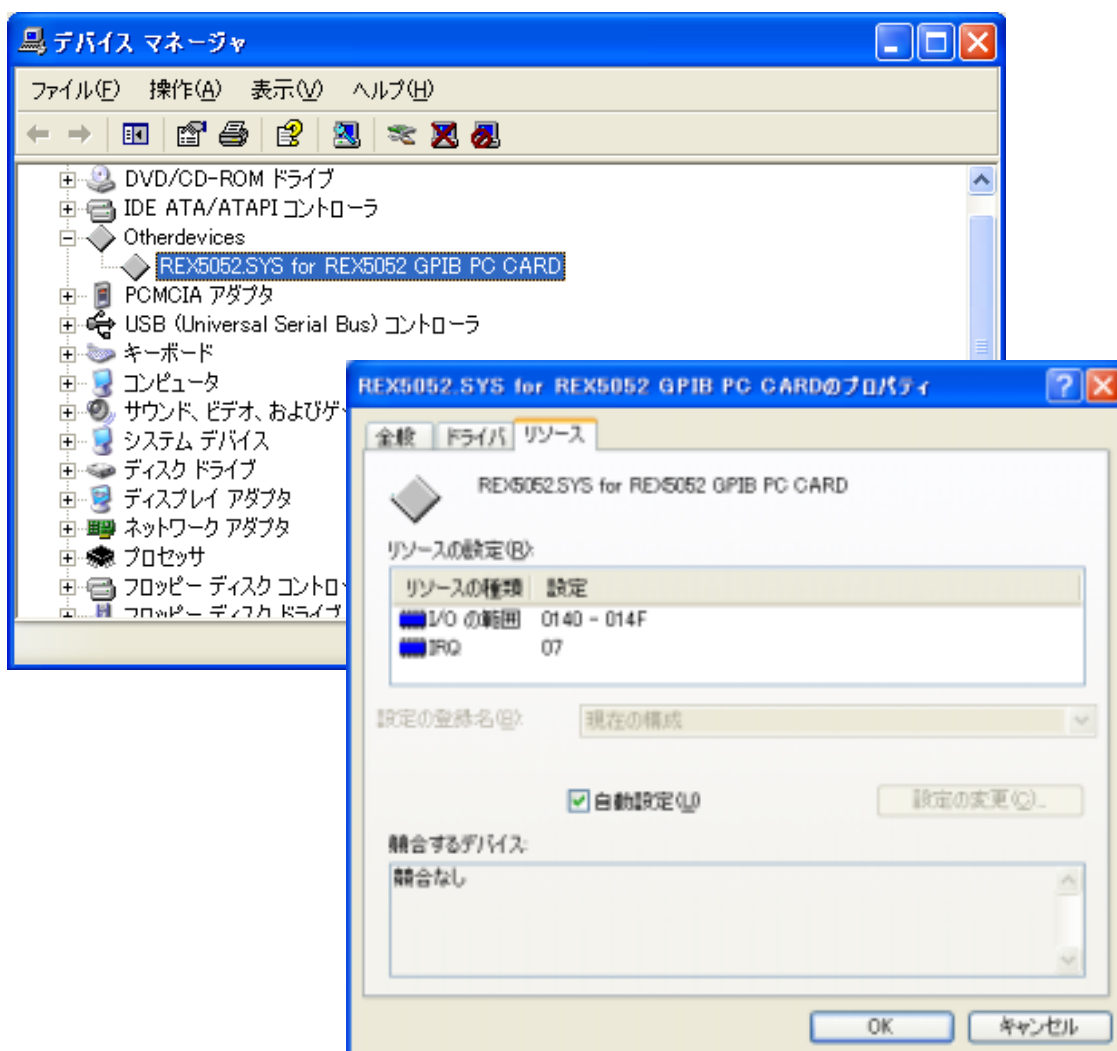


(3-2) PC カード設定内容の確認

Windows2000 および WindowsXP でのインストール確認

コントロールパネルのシステムを起動します。「システムのプロパティ」のハードウェアのタブから「デバイスマネージャ(D)」ボタンを押します。「OtherDevices」をクリックして新しく REX5052.SYS for REX-5052 GPIB PC CARD が追加されているのを確認してください。

また、「プロパティ」でリソースが正しく割当てられているかを確認してください。デバイスの競合が発生した場合は「自動設定(U)」のチェックを外し、競合が起こらない値に設定を変更してください。



(3-3) アンインストール

Windows2000 および WindowsXP でのアンインストール方法

インストールした内容を削除する方法について説明します。
削除は、

(1)デバイスの削除

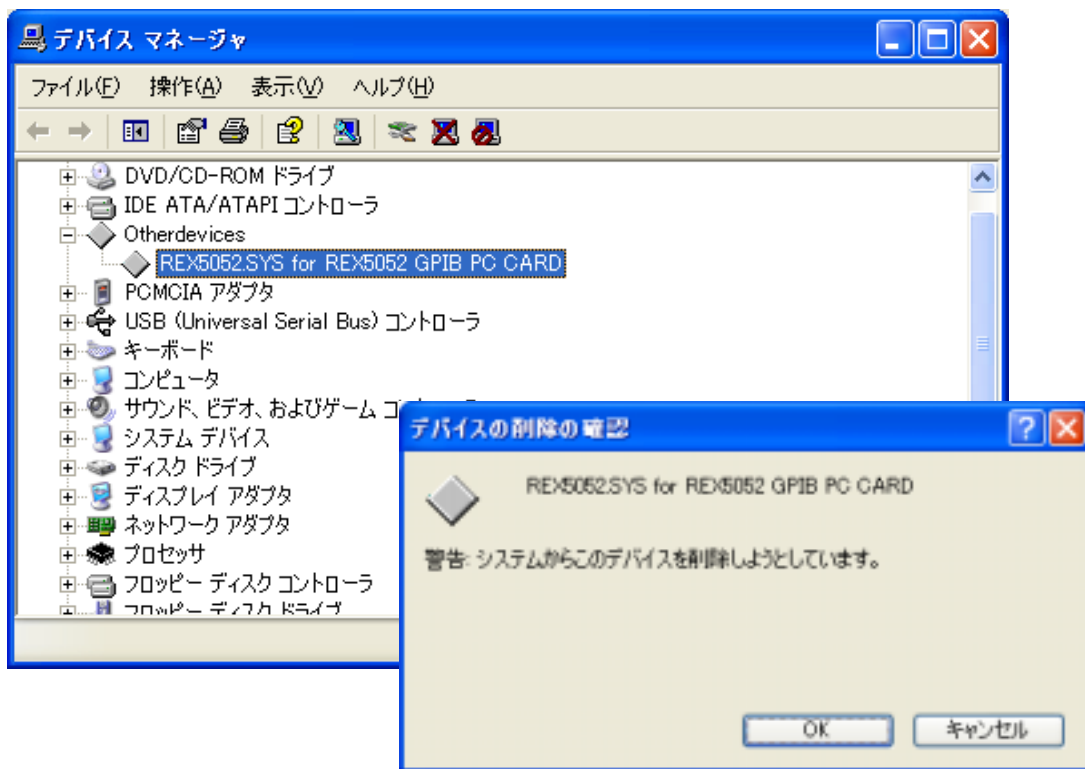
(2)INF ファイルの削除

の手順で行います。

【1】デバイスの削除

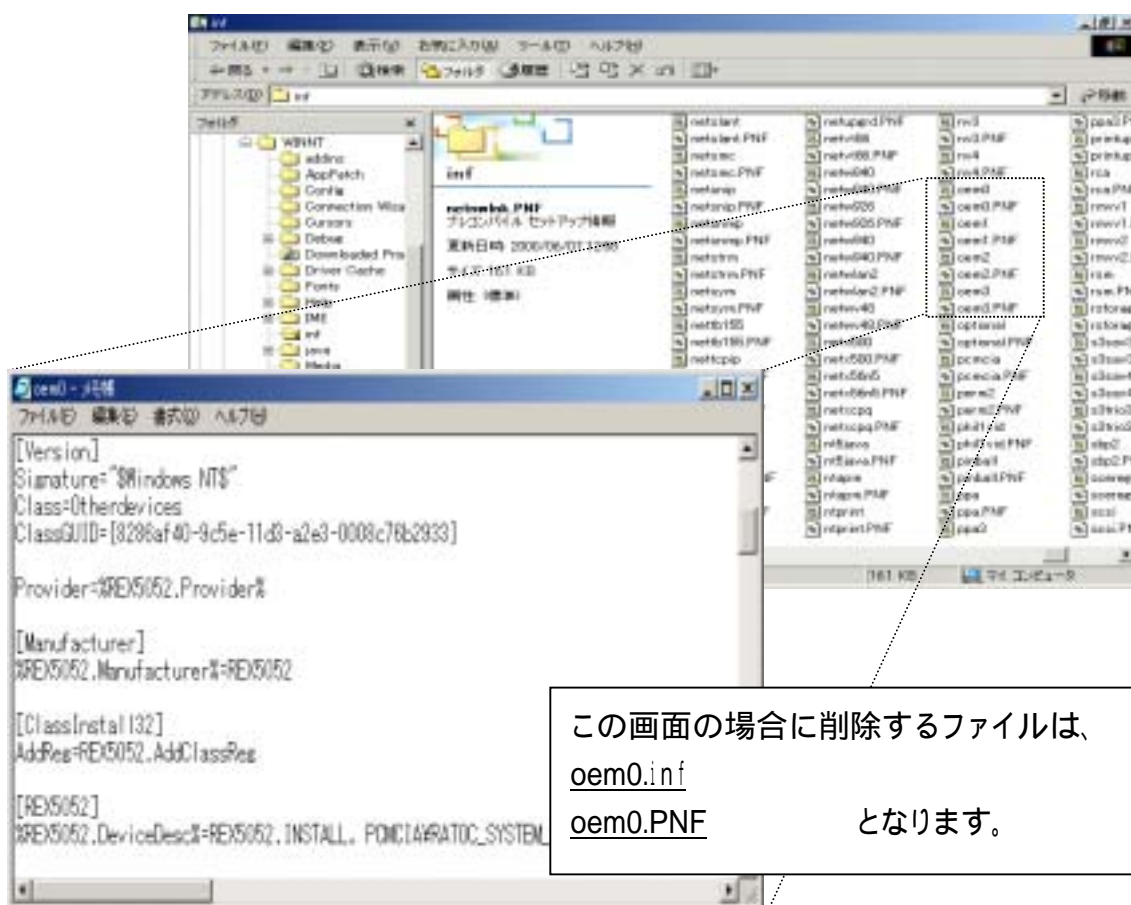
PC カードを挿入した状態で、コントロールパネルのシステムを起動します。「システムのプロパティ」のハードウェアのタブから「デバイスマネージャ(D)」ボタンを押します。「Otherdevices」をクリックして REX-5052.SYS for REX5052 GPIB PC CARD を表示させクリックします。

メニューバーの「操作(A)」 - 「削除(U)」を選択します。デバイスの削除の確認で「OK」ボタンを押し削除してください。



[2] INF ファイルの削除

エクスプローラからフォルダ「C:\WINNT\inf」を開き、oemX.inf ファイル(X=数字)を検索し、例えば oem0.inf が1つだけの場合は、oem0.infと拡張子のみ異なる oem0.PNF を削除してください。oemX.infが複数ある場合(oem0.inf, oem1.inf・・・)は、メモ帳などでそれぞれの inf ファイルを開いて、その内容の[Manufacturer]セクションが %REX5052, Manufacturer%=REX5052 となっているファイルと拡張子のみ異なる PNF ファイルを削除してください。



以上の操作でアンインストール完了です。

カードスロットより、REX-5052 を抜きパソコンを再起動してください。

◆*注意...*

エクスプローラの設定が「全てのファイルを表示」になっていないとフォルダ「C:\WINNT\INF」は表示されません。設定の変更は、エクスプローラメニューの「ツール」から「フォルダオプション」を選択し、表示タグ内の詳細設定で、すべてのファイルとフォルダを表示するに設定してください。

(3-4) DLL ライブラリ関数仕様

サンプルプログラムから DLL でイクスポートされている関数を呼び出すためには、以下 2 点を行う必要があります。

1. DLL 関数をインポート宣言(Visual C)および Declare 宣言(Visual BASIC)する
2. GPLIB2K.LIB をプロジェクトに追加する(Visual C のみ必要)

インポート宣言および Declare 宣言の方法については、サンプルプログラムヘッダーファイル GPLIB2K.H およびモジュールファイル REX5052.BAS を参照してください。

◆関数仕様の記述について

本ソフトウェアを動作させるための個々のコマンドについて解説を行います。汎例を下記に示します。書式及び実行例は Visual C と Visual BASIC 両方を記述します。

gp_xxx(コマンド名)	機 能
書式	VC > Visual C での関数の記述 VB > Visual BASIC での関数の記述
関連	実行時に関連のあるパラメータ

実行例および動作 そのコマンドの実行例と GPIB 各信号線の動作を示します。

留意点

- すべての関数は INT 型の戻り値を返します。(VOID 型を除く)
- 戻り値は、0 の場合は正常終了です。それ以外はエラーコードです。
- 機器アドレスの指定は文字列で行ないます。(各コマンドの解説では書式の項目で "char *adrs" で示されています。)
このとき、トーカ指定が必要なコマンドでは、文字列の先頭の機器アドレスがトーカアドレスとなります。
(例)リスナアドレス 1,3,4,8 の場合 : adrs = "1,3,4,8"
全機器に対する場合 : adrs = "" (ヌル文字列)
- 引き数に関する注意
Visual BASIC で GPLIB2K.DLL を呼び出す場合、値を渡す場合には、ByVal val1 As Integer になります。アドレスを渡す場合には、Val1 As Integer という構文になります。

◆関数一覧

関数	概要	頁
gp_cardinfo	カードのリソース情報を取得	3- 9
gp_init	REX-5052 の初期化	3-10
gp_cli	IFC ラインを TRUE にする (約 10msec 間)	3-11
gp_ren	REN ラインを TRUE にする	3-12
gp_clr	デバイスクリアまたはセレクトッドデバイスクリアコマンド送出	3-13
gp_wrt	リスナアドレスで指定された機器にデータ送信	3-15
gp_red	指定した機器からデータをリード	3-17
gp_trg	リスナに指定された機器に対して GET 命令を送信	3-19
gp_strtodbl	8 バイトのデータを格納するメモリへの BYTE 型ポインタを double 型ポインタにキャストする	3-20
gp_strtoflt	4 バイトのデータを格納するメモリへの BYTE 型ポインタを float 型ポインタにキャストする	3-21
gp_tfrin	指定したトーカより指定バイト分データをバッファに格納	3-22
gp_tfrinit	gp_tfrins のトーカ指定を行う	3-24
gp_tfrins	gp_tfrinit で指定した機器から指定バイト数分のデータをバッファ領域内に直接読み込んで格納	3-25
gp_tfrend	gp_tfrinit で指定したトーカ指定の解除	3-25
gp_tfrout	指定した機器へ指定バイト分のデータを転送	3-26
gp_lcl	指定したリスナ機器をローカル状態に設定	3-27
gp_llo	GPIB 上の全機器のローカルスイッチを無効設定	3-29
gp_wtb	ATN ラインを TRUE にしてコマンド文字列を送信	3-30
gp_rds	シリアルポールを実行しステータスバイトを受信	3-31
gp_rds1	シリアルポールを実行しステータスバイトを受信	3-32
gp_wait	指定した時間プログラムの実行を停止	3-33
gp_srq	シリアルポールウェア割り込み実行および割り込み解除を行う	3-34
gp_wsrq	指定時間 SRQ を待つ (ステータスレジスタ 1 を見る)	3-35
gp_wsrqb	指定時間 SRQ を待つ (バスステータスを見る)	3-36
gp_delm	リスナ時トーカ時のデリミタを設定	3-37
gp_tmout	バスタイムアウトパラメータを設定	3-38
gp_setdelay	外部変数 delay_count のデレイ時間を変更	3-39
gp_count	送・受信データ (バイト) 数の取得	3-40
gp_myadr	設定された REX-5052 の GPIB アドレスを取得	3-41

gp_cardinfo

カードのリソース情報を取得

書式

VC ➤ int gp_cardinfo
 (LPWORD pSlotNo, LPWORD pIOBase, LPWORD plrqNo)
 pSlotNo ➤ (Windows95/98 互換用) "0" を受け取る変数の
 アドレス
 pIOBase ➤ I/O リソース情報を格納する変数のアドレス
 plrqNo ➤ IRQ リソース情報を格納する変数のアドレス

VB ➤ Function gp_cardinfo
 (pSlotNo As Integer, pIOBase As Integer, plrqNo As
 Integer) As Long
 pSlotNo ➤ (Windows95/98 互換用) "0" を受け取る変数の
 アドレス
 pIOBase ➤ I/O リソース情報を格納する変数のアドレス
 plrqNo ➤ IRQ リソース情報を格納する変数のアドレス

関連

なし

実行例および動作

VC ▼

```
WORD SlotNo;           // カードスロット番号(常に0が返ります)
WORD MyIOno;           // GPIB カード I/O ベースアドレス
WORD MyIrqNo;          // GPIB カード割り込み番号

gp_error = gp_cardinfo( &SlotNo, &MyIOno, &MyIrqNo );
```

VB ▼

```
Dim SlotNo As Long      ' カードスロット番号(常に0が返ります)
Dim MyIOno As Long      ' GPIB カード I/O ベースアドレス
Dim MyIrqNo As Long     ' GPIB カード割り込み番号

retval = gp_cardinfo( SlotNo, MyIOno, MyIrqNo)
```

戻り値(10進数)

0 : 正常終了
 それ以外は、リソース取得エラーです

gp_init

REX-5052 の初期化

書式

VC ➤ int gp_init
 (WORD GpAdrs, WORD IOBase, WORD IrqNo)
GpAdrs ➤ カードの GPIB 機器アドレス
IOBase ➤ I/O ベースアドレス
IrqNo ➤ 割り込み番号

VB ➤ Function gp_init
 (ByVal GpAdrs As Integer, ByVal IOBase As Long,
 ByVal IrqNo As Integer) As Long
GpAdrs ➤ カードの GPIB 機器アドレス
IOBase ➤ I/O ベースアドレス
IrqNo ➤ 割り込み番号

関連

なし

実行例および動作

VC ▼

```
WORD    GpAdrs;    // カードの GPIB 機器アドレス
WORD    MyIONo;    // GPIBカードI/Oベースアドレス
WORD    MyIrqNo;   // GPIBカード割り込み番号
```

```
gp_error = gp_init( GpAdrs, MyIONo, MyIrqNo );
```

VB ▼

```
Dim GpAdrs As Integer    ' カードの GPIB 機器アドレス
Dim MyIONo As Integer    ' GPIBカードI/Oベースアドレス
Dim MyIrqNo As Integer   ' GPIBカード割り込み番号
```

```
retval = gp_init( GpAdrs, MyIONo, MyIrqNo )
```

REX-5052 カード上の GPIB コントローラチップにソフトウェアリセットコマンドを送り、GPIB コントローラを初期化し、マイアドレスをセットします。また、本ライブラリで使用するパラメータを初期化します。

戻り値(10進数)

0 : 正常終了
 -1 : カードコンフィグレーションエラー
 60 : デバイスが使用状態にない

gp_cli

IFC ラインを TRUE にする

書式 VC > int gp_cli(void)
 VB > Function gp_cli() As Long

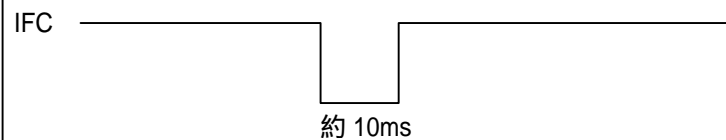
関連 なし

実行例および動作 VC ▼

```
int            gp_error;  
gp_error = gp_cli();
```

 VB ▼

```
Dim gp_error As Long  
gp_error = gp_cli()
```



REX-5052 カード上の LSI 及び、 GPIB に接続されている全ての機器の初期化を行うために、プログラムの先頭部で必ず一度は IFC コマンドの実行が必要です。必ず正常終了します。

戻り値(10進数) 常に 0 を返します。

gp_ren

REN ラインを TRUE にする

書式

VC > int gp_ren(void)

VB > Function gp_ren() As Long

関連

なし

実行例および動作

VC ▼

```
int      gp_error;
gp_error = gp_ren();
```

VB ▼

```
Dim gp_error As Long
gp_error = gp_ren()
```

LCL コマンド(LCL コマンドの項 実行例1を参照)が実行されるか、またはパソコンがリセットされるまでずっと True のままです。GPIB インターフェイスを持つ計測機器や装置は、REN ラインが True になるとリモート可能モードとなり、リモートモードを表示する LED などが点灯します。

REN ラインが False のままですと、GPIB 機器は正しく動作しませんので、プログラム先頭で必ず一度は REN コマンドの実行が必要です。

戻り値(10 進数)

常に 0 を返します。

gp_clr	デバイスクリアまたはセレクトドデバイスクリアコマンド送出
--------	------------------------------

書式 VC > int gp_clr(PCHAR adrs)
 adrs > GPIB 機器アドレス
 VB > Function gp_clr(ByVal adrs As String) As Long
 adrs > GPIB 機器アドレス

関連 なし

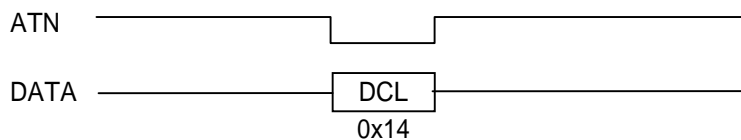
実行例および動作 実行例 1. 全機器に対する場合

VC ▼

```
char     *adrs = "";                    // GPIB 機器アドレス
int       ret_val;
ret_val = gp_clr( adrs );
```

VB ▼

```
Dim UseGPIBAdrs As String * 12        ' GPIB 機器アドレス
retval = gp_clr(Str(UseGPIBAdrs))
```



GPIB上の全機器に対してクリアコマンドを送り、全機器をリセットします。

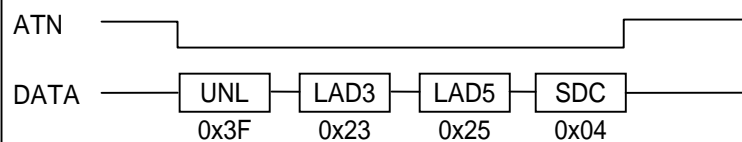
実行例 2. アドレス 3,5 の機器に対して、クリアコマンドを送る場合

VC ▼

```
char    *adrs = "3,5";           // GPIB機器アドレス
int     ret_val;
ret_val = gp_clr ( adrs );
```

VB ▼

```
Dim UseGPIBAdrs As String * 12  ' GPIB機器アドレス
UseGPIBAdrs = "3,5"             ' GPIB機器アドレスをセット
retval = gp_clr(UseGPIBAdrs)
```



相手側機器の DC (DEVICE CLEAR) 機能が DC0 の場合は、このコマンドは無効です。また DC2 の場合は、実行例2の SDC コマンドは無効となりますので、実行例1を御使用ください。

戻り値(10進数)

0 : 正常終了
53 : GPIB バスタイムアウトエラー

gp_wrt

リスナアドレスで指定された機器にデータ送信

書式

VC ➤ int gp_wrt(PCHAR adrs, PCHAR buf)
 adrs ➤ GPIB 機器アドレス
 buf ➤ 送信文字列を格納するバッファアドレス

VB ➤ Function gp_wrt
 (ByVal adrs As String, ByVal buf As String) As Long
 adrs ➤ GPIB 機器アドレス
 buf ➤ 送信文字列を格納するバッファアドレス

関連

タイムアウト, トーカモードデリミタ

実行例および動作

実行例 1. シングルリスナアドレスの場合
 (トーカモードデリミタ = 0)

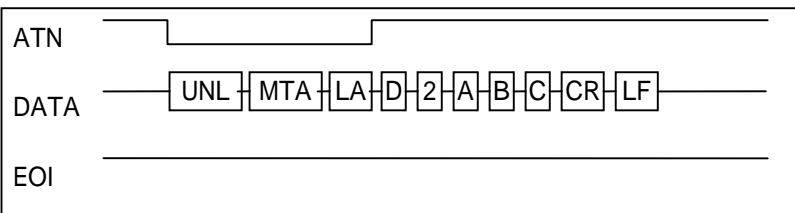
VC ▼

```
char *adrs = "3";           // GPIB 機器アドレス
char buf[128];
int ret_val;
memset( buf,0x00,sizeof(buf) );
strcpy( buf,"D2ABC" );
ret_val = gp_wrt ( adrs , buf );
```

VB ▼

```
Dim UseGPIBAdrs As String * 12 ' GPIB 機器アドレス
Dim StrGPCom As String * 12 ' GPIB コマンド
StrGPCom = "D2ABC"
UseGPIBAdrs = "3"
retval = gp_wrt(UseGPIBAdrs, StrGPCom)
```

アドレス 3 の機器に "D2ABC" という文字列を送信します。



実行例 2. マルチリスナアドレスの場合
(トーカーモードデリミタ = 0x80)

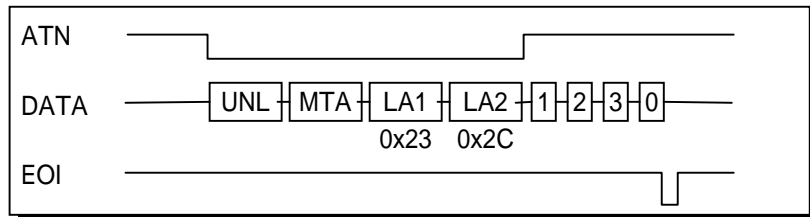
VC ▼

```
char    *adrs = "3,12";           // GPIB機器アドレス
char    buf[128];
int     ret_val;
memset( buf,0x00,sizeof(buf) );
strcpy( buf,"1230" );
ret_val = gp_wrt ( adrs , buf );
```

VB ▼

```
Dim UseGPIBAdrs As String * 12    ' GPIB機器アドレス
Dim StrGPCom As String * 12      ' GPIBコマンド
StrGPCom = "1230"
UseGPIBAdrs = "3,12"
retval = gp_wrt(UseGPIBAdrs, StrGPCom)
```

アドレス 3,12 の機器に文字列を送信します。



戻り値(10進数)

- 0 : 正常終了
- 2 : 送信データ設定エラー
- 53 : GPIB バスタイムアウトエラー

gp_red

指定した機器からデータをリード

書式

VC > int gp_red(PCHAR adrs, PCHAR buf, int bufLen)

adrs > GPIB 機器アドレス

buf > 受信文字列を格納するバッファアドレス

bufLen > バッファレングス

VB > Function gp_red

(ByVal adrs As String, ByVal buf As String, ByVal bufLen As Long) As Long

adrs > GPIB 機器アドレス

buf > 受信文字列を格納するバッファアドレス

bufLen > バッファレングス

注) バッファサイズは受信するバイト数より必ず1バイト以上多く取ってください。

関連

タイムアウト, リスナモードデリミタ

実行例および動作

実行例 1. 相手側機器の送信時デリミタが LF の場合

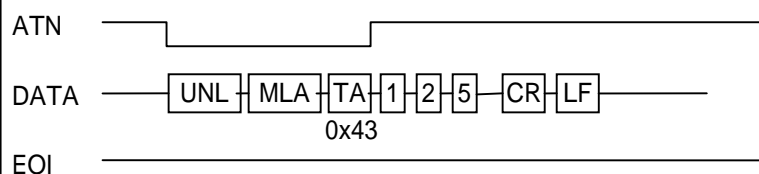
VC ▼

```
char *adrs = "3";           // GPIB機器アドレス
char buf[256];             // GPIB受信バッファ
int ret_val;
ret_val = gp_red( adrs , buf , sizeof(buf) );
```

VB ▼

```
Dim UseGPIBAdrs As String * 12 ' GPIB機器アドレス
Dim Buf As String * 64        ' GPIB受信バッファ
Buf = " " ' 必ず何らかの文字列をいれて初期化
UseGPIBAdrs = "3"
retval = gp_red( UseGPIBAdrs, Buf, 64 )
```

アドレス3の機器よりデータを受信し、文字配列buf内に格納します。



HP 社、横河電機、アドバンテスト等、ほとんどのメーカーが送信時デリミタとして CR,LF を使用していますので、リスナモードデリミタとしては 0x0a(LF)が一般的です。

実行例 2. リスナアドレス付の場合

VC ▼

```

char    *adrs = "3,10,12";           // GPIB 機器アドレス
char    buf[10];                     // GPIB 受信バッファ
int      ret_val;
ret_val = gp_red( adrs, buf , sizeof(buf) );

```

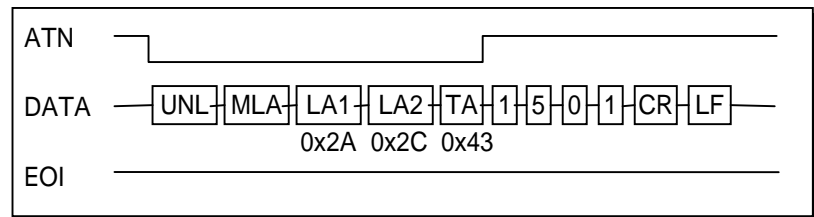
VB ▼

```

Dim UseGPIBAdrs As String * 12      ' GPIB 機器アドレス
Dim Buf As String * 64              ' GPIB 受信バッファ
Buf = " "                          ' 必ず何らかの文字列で初期化
UseGPIBAdrs = "3,10,12"
retval = gp_red( UseGPIBAdrs, Buf, 64 )

```

アドレス3の機器よりデータを受信し、文字配列 buf 内に格納します。同時にアドレス 10,12 の機器にもデータが送られます。



(注意)

red コマンドは、相手側機器から出力される EOI を検出すると、その時点で読み込み動作を終了します。

戻り値(10進数)

0 : 正常終了
53 : GPIB バスタイムアウトエラー
61 : バッファオーバーフロー

gp_trg リスナに指定された機器に対して GET 命令を送信

書式 **VC** > int gp_trg(PCHAR adrs)
 adrs > GPIB 機器アドレス
VB > Function gp_trg(ByVal adrs As String) As Long
 adrs > GPIB 機器アドレス

関連 タイムアウト

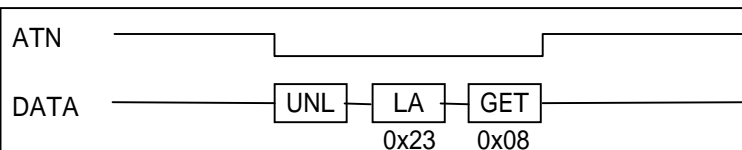
実行例および動作 **VC** ▼

```
char *adrs = "3";           // GPIB機器アドレス
int ret_val;
ret_val = gp_trg ( adrs );
```

VB ▼

```
Dim UseGPIBAdrs As String * 12 ' GPIB機器アドレス
UseGPIBAdrs = "3"
retval = gp_trg( UseGPIBAdrs )
```

アドレス 3 の機器に対して GET 命令を送信します。



戻り値(10進数) 0 : 正常終了
53 : GPIB バスタイムアウトエラー

gp_strtodbl

8 バイトのデータを格納するメモリへの BYTE 型
ポインタを double 型ポインタにキャストする

書式

VC > void gp_strtodbl(BYTE *bPoint, double *val)
 bPoint > 8 バイトデータを格納するメモリへの BYTE
 型ポインタ
 val > キャストした double 型ポインタ

VB > Sub gp_strtodbl(bPoint As Any, val As Double)
 bPoint > 8 バイトデータを格納するメモリへの BYTE
 型アドレス
 val > キャストした double 型アドレス

関連

タイムアウト

実行例および動作

8 バイトのデータの格納するメモリへ BYTE 型ポインタを受けて
 その 8 バイトのデータを double 型実数に変換します。

VC では、直接キャスト可能であるため、使用する必要はありません。

VC ▼

```
byte    buf[8];    // 8 バイトデータを格納する BYTE 型ポインタ
double  data;     // キャストした double 型ポインタ
buf[0] = 0x1B;
buf[1] = 0xDE;
buf[2] = 0x83;
buf[3] = 0x42;
buf[4] = 0xCA;
buf[5] = 0XC0;
buf[6] = 0XF3;
buf[7] = 0x3F;
gp_strtodbl(buf,&data);
```

VB ▼

```
Dim ReadBuf(7) As Byte    ' 8 バイトデータを格納するメモリへのアドレス
Dim data As Double       ' キャストした double 型アドレス
buf(0) = &H1B
buf(1) = &HDE
buf(2) = &H83
buf(3) = &H42
buf(4) = &HCA
buf(5) = &HC0
buf(6) = &HF3
buf(7) = &H3F
aa strtodbl ReadBuf(0) data
```

戻り値 (10 進数)

なし

gp_strtoflt

4 バイトのデータを格納するメモリへの BYTE 型
ポインタを float 型ポインタにキャストする

書式

VC > void gp_strtoflt(BYTE *bPoint, float *val)
 bPoint > 4 バイトデータを格納するメモリへの BYTE 型
 ポインタ
 val > キャストした float 型ポインタ

VB > Sub gp_strtoflt(**bPoint** As Any, **val** As Single)
 bPoint > 4 バイトデータを格納するメモリへの BYTE
 型アドレス
 val > キャストした float 型アドレス

関連

タイムアウト

実行例および動作

4 バイトのデータの格納するメモリへ BYTE 型ポインタを受けて
 その 4 バイトのデータを float 型実数に変換します。

VC では、直接キャスト可能であるため、使用する必要はありません。

VC ▼

```
byte    buf[4];    // 4 バイトデータを格納する BYTE 型ポインタ
float   data;     // キャストした float 型ポインタ
buf[0] = 0x52;
buf[1] = 0x06;
buf[2] = 0x9E;
buf[3] = 0x3F;

gp_strtoflt(buf,&data);
```

VB ▼

```
Dim ReadBuf(3) As Byte    ' 4 バイトデータを格納するメモリへのアドレス
Dim data As float        ' キャストした float 型アドレス

buf(0) = &H52
buf(1) = &H6
buf(2) = &H9E
buf(3) = &H3F

gp_strtoflt buf(0), data
```

戻り値(10 進数)

なし

gp_tfrin**指定したトーカーより指定バイト分データをバッファに格納**

書式

VC > int gp_tfrin(PCHAR adrs, int bufLen, PCHAR buf)

adrs > GPIB 機器アドレス

bufLen > バッファレングス

buf > 受信用配列領域

VB > Function gp_tfrin

(ByVal adrs As String, ByVal bufLen As Long, ByVal buf As String) As Long

adrs > GPIB 機器アドレス

bufLen > バッファレングス

buf > 受信用配列領域

関連

タイムアウト

実行例および動作

- 画像処理装置や FFT アナライザなどでは、一度に1～数 Kb のデータを転送する機能を持っていますので、この tfrin を使用するとデータを1度に受信できます。
- 受信バイト数がバッファ変数の長さよりも大きい場合は、バッファ変数分のデータだけ受け取ります。但し受信動作は EOI が来るまで行い、バッファに入り切らない分は捨てられます。またその場合には戻り値として 61(BufferOverflow)を返します。
- 受信バイト数の指定は、整数型変数または符号無し整数型変数で行ってください。

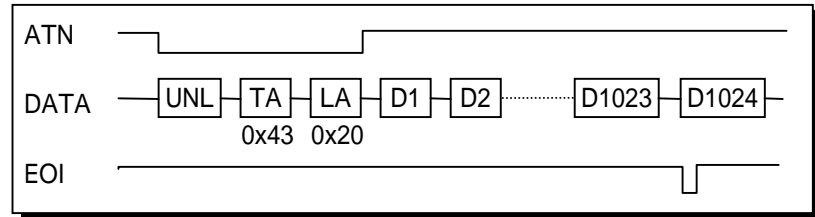
VC ▼

```
char *adrs = "3";           // GPIB機器アドレス
char buf[1025];           // GPIB受信バッファ
int bytc=1024;
int ret_val;
ret_val = gp_tfrin ( adrs, bytc, buf );
```

VB ▼

```
Dim UseGPIBAdrs As String * 12 ' GPIB機器アドレス
Dim Buf As String * 1025 ' GPIB受信バッファ
bytc = 1024
UseGPIBAdrs = "3"
retval = gp_tfrin( UseGPIBAdrs, bytc, buf )
```

トーカーアドレス 3 の機器から 1024 バイトのデータをバッファ変数内に読み込みます。リスナ指定が無い場合は、REN ラインを False にし、GPIB 上の全機器をローカル状態に戻します。



戻り値(10進数) 0 : 正常終了
 53 : GPIB バスタイムアウトエラー
 61 : バッファオーバーフロー

gp_tfrinit

gp_tfrins のトーク指定を行う

書式

VC > int gp_tfrinit(PCHAR adrs)
 adrs > GPIB 機器アドレス

VB > Function gp_tfrinit
 (ByVal adrs As String) As Long
 adrs > GPIB 機器アドレス

関連

gp_tfrins(), gp_tfrend()を続けて呼び出してください。

実行例および動作

VC ▼

```
char    *adrs = "3";           // GPIB機器アドレス
char    buf[1025];           // GPIB受信バッファ
int     bytc = 1024;
int     ret_val;
ret_val = gp_tfrinit( adrs );
ret_val = gp_tfrins( bytc, buf );
gp_tfrend();
```

VB ▼

```
Global UseGPIBAdrs As String * 12 ' GPIB機器アドレス
Global Buf As String * 1025      ' GPIB受信バッファ
bytc = 1024
UseGPIBAdrs = "3"

retval = gp_tfrinit( UseGPIBAdrs )
retval = gp_tfrins( bytc, buf )
gp_tfrend
```

戻り値(10進数)

0 : 正常終了
 53 : GPIB バスタイムアウトエラー

gp_tfrins	gp_tfrinit で指定した機器から指定バイト数分のデータをバッファ領域内に直接読み込んで格納
------------------	--

書式

VC > int gp_tfrins (int bufLen, PCHAR buf)
 bufLen > バッファレングス
 buf > 受信用配列領域

VB > Function gp_tfrins
 (ByVal bufLen As Long, ByVal buf As String) As Long
 bufLen > バッファレングス
 buf > 受信用配列領域

関連 gp_tfrinit()を呼び出した後、gp_tfrins()を呼び出してください。

実行例および動作 (前頁の gp_tfrinit を参照してください)

指定バイト数分のデータをバッファ領域内に直接読み込んで格納します。読み込み動作は、指定されたバイト数分で終了するかまたは、EOIを検出した時点で終了します。

戻り値(10進数) 0 : 正常終了
 24 : EOIを受信して終了(正常終了)
 53 : GPIB バスタイムアウトエラー

gp_tfrend	gp_tfrinit で指定したトーカ指定の解除
------------------	---------------------------------

書式

VC > void gp_tfrend(void)
VB > Sub gp_tfrend()

関連 gp_tfrinit(), gp_tfrins()を呼び出した後、gp_tfrend()を呼び出してください。

実行例および動作 (前頁の gp_tfrinit を参照してください)

戻り値(10進数) なし

gp_tfrount

指定した機器へ指定バイト分のデータを転送

書式

VC > int gp_tfrount(PCHAR adrs, int bufLen, PCHAR buf)

adrs > GPIB 機器アドレス

bufLen > バッファレングス

buf > 送信用配列領域

VB > Function gp_tfrount

(ByVal adrs As String, ByVal bufLen As Long, ByVal buf As String) As Long

adrs > GPIB 機器アドレス

bufLen > バッファレングス

buf > 送信用配列領域

関連

タイムアウト

実行例および動作

- 画像処理装置や FFT アナライザなどへ一度に数 KB のデータを送り込む場合にこの tfrount コマンドを使用します。
- 送信時デリミタとして、EOI が送られます。
- 送信バイト数の指定は、整数型変数または符号無し整数型変数で行ってください。

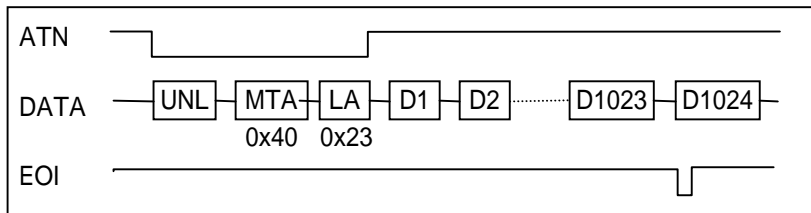
VC ▼

```
char *adrs = "3";           // GPIB 機器アドレス
char buf[1025];            // GPIB 送信バッファ
int bytc;
int ret_val;
bytc = 1024;
ret_val = gp_tfrount( adrs, bytc, buf );
```

VB ▼

```
Dim UseGPIBAdrs As String * 12 ' GPIB 機器アドレス
Dim buf As String * 1025      ' GPIB 送信バッファ
bytc = 1024
UseGPIBAdrs = "3"
retval = gp_tfrount( UseGPIBAdrs, bytc, buf )
```

リスナアドレス 3 の機器へ 1024 バイトのデータを送信します。



戻り値 (10 進数)

- 0 : 正常終了
- 2 : 送信データ設定エラー
- 53 : GPIB バスタイムアウトエラー

gp_lcl

指定したリスナ機器をローカル状態に設定

書式

VC > int gp_lcl(PCHAR adrs)

adrs > GPIB 機器アドレス

VB > Function gp_lcl(ByVal adrs As String) As Long

adrs > GPIB 機器アドレス

関連

タイムアウト

実行例および動作 実行例 1. 全機器に対する場合

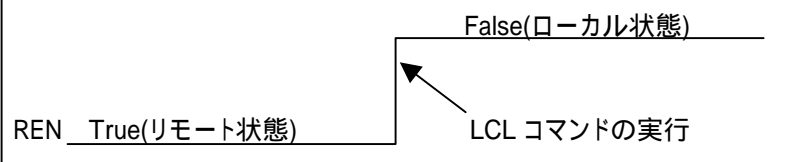
VC ▼

```
char *adrs = ""; // GPIB 機器アドレス
int ret_val;
ret_val = gp_lcl( adrs );
```

VB ▼

```
Dim UseGPIBAdrs As String * 12 ' GPIB 機器アドレス
retval = gp_lcl( Str(UseGPIBAdrs) ) ' 初期化していない文字列ですと
' 先頭に 00h が入っています。
```

GPIB 上の全機器をローカルモードにします。



実行例 2. リスナアドレスの指定がある場合

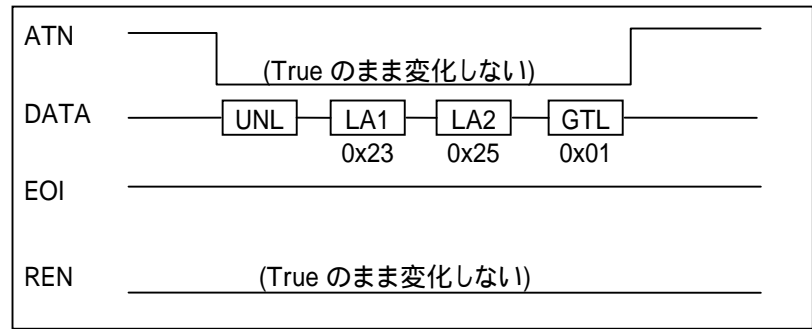
VC ▼

```
char *adrs = "3,5";           // GPIB機器アドレス
int ret_val;
ret_val = gp_lcl( adrs );
```

VB ▼

```
Dim UseGPIBAdrs As String * 12 ' GPIB機器アドレス
UseGPIBAdrs = "3,5"           ' GPIB機器アドレスをセット
retval = gp_lcl( UseGPIBAdrs )
```

リスナアドレス 3,5 の機器に GTL (go to local) 命令を送りローカル状態に戻します。



戻り値 (10 進数)

- 0 : 正常終了
- 53 : GPIB バスタイムアウトエラー

gp_llo

GPIB 上の全機器のローカルスイッチを無効設定

書式

VC > int gp_llo(void)

VB > Function gp_llo() As Long

関連

なし

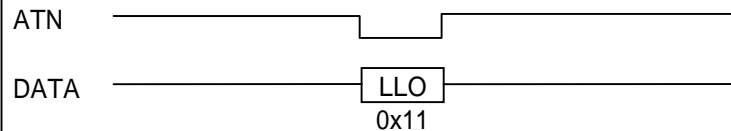
実行例および動作

VC ▼

```
int ret_val;
ret_val = gp_llo();
```

VB ▼

```
Dim retval As Long
retval = gp_llo()
```



- ATNラインを True にし、LLO 命令を送信した後 ATN ラインを False にします。この命令を受信すると機器側ではパネル上の操作スイッチを無効にします。ただし機器のリモート状態もしくはローカル状態には、変化は生じません。
- 機器の LLO 状態を解除する場合は REN ラインを False にします。(LCL コマンドの実行)

戻り値(10 進数)

0 : 正常終了

53 : GPIB バスタイムアウトエラー

gp_wtb	ATN ラインを TRUE にしてコマンド文字列を送信
---------------	------------------------------------

書式 VC > int gp_wtb(PCHAR buf)
 buf > 送信用配列領域

 VB > Function gp_wtb(ByVal buf As String) As Long
 buf > 送信用配列領域

関連 なし

実行例および動作 VC ▾

```
int ret_val;
char buf[256];
buf[0] = 0x3f;
buf[1] = 0x23;
buf[2] = 0x01;
buf[3] = 0x00;
ret_val = gp_wtb( buf );
```

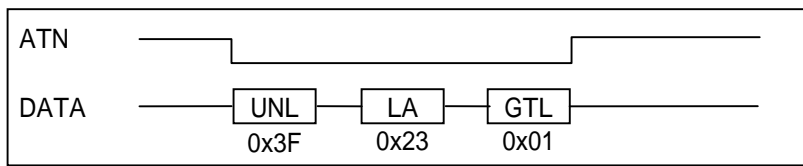
コマンド文字列の最後に、コマンド終了の buf[3] = 0x00 を記述する必要があります。

VB ▾

```
Dim buf As String * 64
buf = chr$(3f)+chr$(23)+chr$(01)+chr$(0)
retval = gp_wtb( buf )
```

コマンド文字列の最後に、コマンド終了の chr\$(0)を記述する必要があります。

LCL3 の実行と同様になります。



戻り値 (10 進数) 0 : 正常終了
 2 : 送信データ設定エラー
 53 : GPIB バスタイムアウトエラー

gp_rds

シリアルポールを実行しステータスバイトを受信

書式

VC > int gp_rds(PCHAR adrs, PCHAR pstatus_byte)
 adrs > GPIB 機器アドレス
 pstatus_byte > GPIB 機器ステータスを返す変数
 へのポインタ

VB > Function gp_rds
 (ByVal adrs As String, status_byte As Long) As Long
 adrs > GPIB 機器アドレス
 pstatus_byte > GPIB 機器ステータスを返す変数
 へのメモリアドレス

タイムアウト

関連

実行例および動作

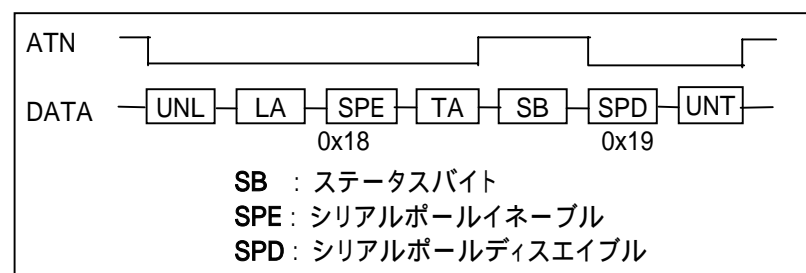
VC ▾

```
char      *adrs = "3";           // GPIB機器アドレス
unsigned int  status;           // GPIB機器ステータス
int        ret_val;
ret_val = gp_rds( adrs,&status );
```

VB ▾

```
Dim UseGPIBAdrs As String * 12 ' GPIB機器アドレス
Dim status As Long             ' GPIB機器ステータス
UseGPIBAdrs = "3"
retval = gp_rds( UseGPIBAdrs, status )
```

トーカーアドレス 3 の機器に対してシリアルポールを実行し、その機器のステータスバイトを読み込み変数 status に代入する。



SRQ を発信中の機器に対してこのコマンドを実行すると、SRQ ラインが False に復帰します。

戻り値 (10 進数)

0 : 正常終了
 53 : GPIB バスタイムアウトエラー

gp_rds1

シリアルポールを実行しステータスバイトを受信

(注意) gp_rds との違いは、最後に UNT コマンドを送出しない点です。

書式

VC > int gp_rds1(PCHAR adrs, PCHAR pstatus_byte)
 adrs > GPIB 機器アドレス
 pstatus_byte > GPIB 機器ステータスを返す変数へのポインタ

VB > Function gp_rds1
 (ByVal adrs As String, status_byte As Long) As Long
 adrs > GPIB 機器アドレス
 pstatus_byte > GPIB 機器ステータスを返す変数へのメモリアドレス

タイムアウト

関連

実行例および動作

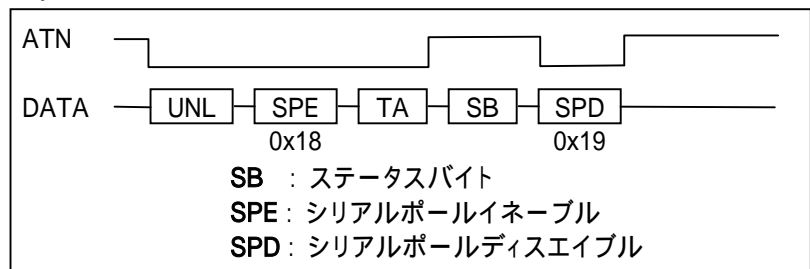
VC ▼

```
char          *adrs = "3";
unsigned int   status;
int           ret_val;
ret_val = gp_rds1( adrs,&status );
```

VB ▼

```
Dim UseGPIBAdrs As String * 12      ' GPIB機器アドレス
Dim status As Long                  ' GPIB機器ステータス
UseGPIBAdrs = "3"
retval = gp_rds1( UseGPIBAdrs, status )
```

トーカーアドレス 3 の機器に対してシリアルポールを実行し、その機器のステータスバイトを読み込み変数 status に代入する。



SRQ を発信中の機器に対してこのコマンドを実行すると、SRQ ラインが False に復帰します。

戻り値(10進数)

0 : 正常終了
 53 : GPIB バスタイムアウトエラー

gp_wait

指定した時間プログラムの実行を停止

書式

VC > void gp_wait(int WaitSecTime)
 WaitSecTime > 秒単位のウェイト時間

VB > Sub gp_wait
 (ByVal WaitSecTime As Long)
 WaitSecTime > 秒単位のウェイト時間

関連

なし

実行例および動作

- 1 WaitSecTime は約1秒です。
- 強制的にプログラムを停止させますのでマウスがきかなくなります。16bit 版からの互換性のために用意された関数です。

VC ▼

```
unsigned int  WaitSecTime = 10;    // 待ち時間秒単位で指定
int          ret_val;
ret_val = gp_wait( WaitSecTime );
```

VB ▼

```
Dim WaitSecTime As Long           ' 待ち時間秒単位で指定
WaitSecTime = 10
retval = gp_wait( WaitSecTime )
```

10 秒間、プログラムの実行を停止します。

戻り値(10 進数)

なし

gp_srq シリアルポートハードウェア割り込み実行および割り込み解除を行う

書式

VC > int gp_srq(HWND hWnd, INT SrqMode)

hWnd > ウィンドウのハンドル

SrqMode > 割り込み実行・解除フラグ

 実行フラグ: ENABLE_SRQ_INTERRUPT

 解除フラグ: DISABLE_SRQ_INTERRUPT

VB > Function gp_srq
(ByVal hWnd As Long, ByVal SrqMode As Long) As Long

hWnd > ウィンドウのハンドル

SrqMode > 割り込み実行・解除フラグ

 実行フラグ: ENABLE_SRQ_INTERRUPT

 実行フラグ: DISABLE_SRQ_INTERRUPT

関連 なし

実行例および動作 **VC** ▾

```
int        ret_val;
ret_val = gp_srq( hWnd, ENABLE_SRQ_INTERRUPT );
```

第2引数に「ENABLE_SRQ_INTERRUPT」を指定することにより、シリアルポート割り込みを実行します。

VB ▾

```
Dim OleHandle As Long                    ' MBOX5052.OCX ハンドル
Dim retval As Integer
' OLE のウィンドウハンドル取得
OleHandle = MBOX5052.GetMboxWnd
retval = gp_srq( olehandle, DISABLE_SRQ_INTERRUPT )
```

第2引数に「DISABLE_SRQ_INTERRUPT」を指定することにより、シリアルポート割り込みを解除します。

戻り値(10進数)

0 : 正常終了

-1 : モード設定エラー

-2 : 開始エラー

-4 : 開始エラー

gp_wsrq 指定時間 SRQ を待つ (ステータスレジスタ 1 を見る)

書式 VC > int gp_wsrq(int WaitSecTime)
WaitSecTime > 秒単位のウェイト時間
VB > Function gp_wsrq
(ByVal WaitSecTime As Long) As Long
WaitSecTime > 秒単位のウェイト時間

関連 なし

- 実行例および動作
- 1WaitSecTime は1秒です。
 - このコマンドによって SRQ ラインは変化しません。
 - 時間内に SRQ がなければ-1 を返します

VC ▼

```
unsigned int WaitSecTime = 10; // 待ち時間秒単位で指定
int ret_val;
ret_val = gp_wsrq( WaitSecTime );
```

VB ▼

```
Dim WaitSecTime As Long ' 待ち時間秒単位で指定
WaitSecTime = 10
retval = gp_wsrq( WaitSecTime )
```

SRQ がくるまで 10 秒間待ちます。

戻り値(10 進数) 0 :SRQ 正常受信
-1 :タイムアウト

gp_wsrqb	指定時間 SRQ を待つ (バスステータスを見る)
----------	-----------------------------

書式

VC > int gp_wsrqb(int WaitSecTime)
 WaitSecTime > 秒単位のウェイト時間

VB > Function gp_wsrqb
 (ByVal WaitSecTime As Long) As Long
 WaitSecTime > 秒単位のウェイト時間

関連 なし

- 実行例および動作
- 1WaitSecTime は1秒です。
 - このコマンドによって SRQ ラインは変化しません。
 - 時間内に SRQ がなければ-1 を返します

VC ▼

```
unsigned int  WaitSecTime = 10;    //待ち時間秒単位で指定
int           ret_val;
ret_val=gp_wsrqb( WaitSecTime );
```

VB ▼

```
Dim WaitSecTime As Long           ' 待ち時間秒単位で指定
WaitSecTime = 10
retval = gp_wsrqb( WaitSecTime )
```

SRQ がくるまで 10 秒間待ちます。

戻り値(10 進数) 0: SRQ 正常受信
 -1: タイムアウト

gp_delm

リスナ時トーカー時のデリミタを設定

書式

VC > int gp_delm(char *mode, unsigned int delm)

mode > (以下参照)

delm > (以下参照)

VB > Function gp_delm

(ByVal mode As String, ByVal delm As Long) As Long

mode > (以下参照)

delm > (以下参照)

関連

タイムアウト

実行例および動作

mode は "t", "l" のどれか一文字とし、次の意味を持ちます。

"t" : トーカー時の送信デリミタを指定します。

"l" : リスナ時の受信デリミタを指定します。

delm は 0 ~ 255 (0x00 ~ 0xff) の範囲の値で mode により次の意味をもちます。

"t" : デリミタコードは bit6 ~ bit0 の 7bit で設定します。

この時、bit7 を 1 にすると EOI を出力します。

delm = 0 とした場合は CR+LF が設定されます。

"l" : デリミタコードは bit7 ~ bit0 の 8bit で設定します。

変更されたデリミタは、次にこのコマンドによって変更されるまで有効です。

デフォルト状態では、トーカーモードデリミタは 0 (CR+LF) に、リスナモードデリミタは 0x0a (LF) に設定されています。

リスナモードデリミタとして LF を設定します。

VC ▼

```
char          *mode = "l";           // モード
unsigned int  delm = 0x0a;           // デリミタ
int           ret_val;
ret_val = gp_delm( mode, status );
```

VB ▼

```
Dim GPIBMode As String * 2           ' モード
Dim delm As Long                     ' デリミタ
GPIBMode = "l"
delm = &h0a
retval = gp_delm( GPIBMode, delm )
```

戻り値(10進数)

0 : 正常終了

53 : GPIB バスタイムアウトエラー

gp_tmout

バスタイムアウトパラメータを設定

書式

VC > int gp_tmout(int SecTime)
 SecTime > 秒単位のタイムアウト時間
VB > Function gp_tmout
 (ByVal **SecTime** As Long) As Long
 SecTime > 秒単位のタイムアウト時間

関連

なし

実行例および動作

- 1SecTime は1秒です。
- タイムアウトは1バイトのハンドシェイクに対し設定されます。
- デフォルト値は 10 秒です。
red/wrt 等のコマンド実行時のバスタイムアウトを 3 秒に設定します。

VC ▼

```
int    ret_val;
ret_val = gp_tmout( 3 );
```

VB ▼

```
Dim retval As Integer
retval = gp_tmout( 3 )
```

戻り値(10進数)

0 : 正常終了
 53 : GPIB バスタイムアウトエラー

gp_setdelay

外部変数 delay_count のディレイ時間を変更

書式

VC > int gp_setdelay(int DelayTime)
DelayTime > 0.8 μ sec 単位のディレイ時間

VB > Function gp_setdelay
(ByVal DelayTime As Long) As Long
DelayTime > 0.8 μ sec 単位のディレイ時間

関連

なし

実行例および動作

- ATNラインを TRUE または FALSE にする時のディレイ時間を変更します。
- コマンドデータを送信時にタイムアウトエラーになる場合に調整します。
- デフォルトは、0 μ sec になっています。

VC ▼

```
int    ret_val;  
ret_val = gp_setdelay( 500 );
```

VB ▼

```
Dim retval As Integer  
retval = gp_setdelay( 500 )
```

戻り値(10進数)

ダミーで引数をそのまま返します。

gp_count

実際に送・受信したデータ数(バイト数)の取

書式 VC > int gp_count(void)
 VB > Function gp_count() As Long

関連 なし

実行例および動作 VC ▾

```
int        ret_val;  
ret_val = gp_count();
```

 VB ▾

```
Dim retval As Integer  
retval = gp_count()
```

gp_red(), gp_tfrin(), gp_tfrins(), gp_wrt(), gp_tfrout()を実行後、gp_count()の呼び出しで実際に送・受信したデータ数(バイト数)を返します。

(注意)

gp_red ではデミリタをバッファ内に入れていないため1バイト少ない値を返します。

戻り値(10進数) 送信または受信バイト数を返します。

gp_myadr

設定された GPIB マイアドレスの値をリード

書式

VC > int gp_myadr(void)

VB > Function gp_myadr() As Long

関連

なし

実行例および動作

互換性を確保する関数ですので、プログラムで新たに自分の機器アドレスを知る必要がない場合は実行する必要はありません。

VC ▼

```
int    da;
da = gp_myadr();
```

VB ▼

```
da = gp_myadr()
```

戻り値(10進数)

GPIB 機器アドレスを返します

(3-5) Visual C サンプルプログラム

Visual C++ 5.0 および 6.0 で、本製品に添付されている“GPLIB2K.DLL”のライブラリを使って REX-5052 を制御するアプリケーションを開発する場合は、サンプルプログラム“HP3478A.C”を参考にしてください。

アプリケーションプログラムから“GPLIB2K.DLL”を呼び出すためには、以下の手順を行ってください。

- アプリケーションプログラムに“GPLIB2K.H”ファイルをインクルードする。
- アプリケーションプログラムのプロジェクトファイルに GPLIB2K.LIB を追加する。

(注意)

“GPLIB2K.DLL”を呼び出しに必要となるインポート宣言、ライブラリ定数等の宣言を“GPLIB2K.H”ヘッダーファイルで行っています。アプリケーション作成の際は“GPLIB2K.H”ヘッダーファイルの内容を理解してください。

本製品(Windows2000/XP 用 FD)にはヒューレットパッカード社製 HP3478A(マルチメーター)を使用した以下の2つのサンプルプログラムを添付しております。2つのサンプルプログラムは、同様の結果が得られます。

- ・ 割り込みを使用せずに SRQ が来るのをポーリングしデータを取得するプログラム (ポーリングモード)
- ・ SRQ の検知に割り込みを使用し、データを取得するプログラム (割り込みモード)

次頁より、サンプルプログラムについて解説いたします。

Windows95/98/Me で作成したアプリケーションを Windows2000/XP で使用する場合

基本的には Windows2000/XP 用ヘッダファイル GPLIB2K.H とライブラリファイル GPLIB2K.LIB を新規プロジェクトに追加し、Windows95/98 で作成したソースファイルにインクルード後、コンパイルすることによって使用可能になります。

但し、以下の関数については変数型が異なりますのでご注意ください。

```
int gp_rds( PCHAR, PCHAR ) ...Page3-31 参照
```

```
int gp_rds1( PCHAR, PCHAR ) ...Page3-32 参照
```

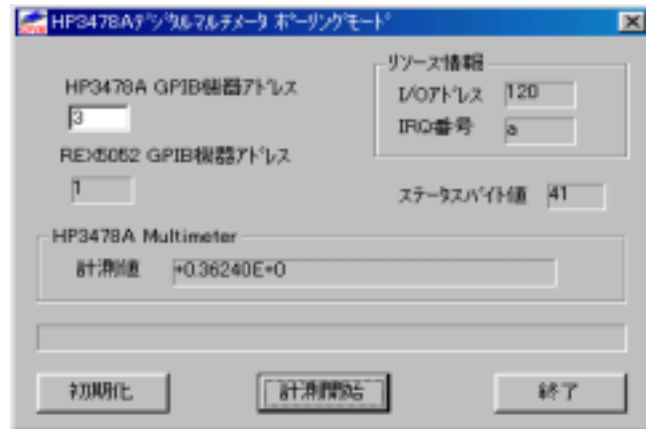
また、Windows2000/XP の GPLIB2K.DLL では、割り込みをご使用頂けるようになっているため以下の関数を追加しております。

```
int gp_srq( HWND, INT ) ...Page3-34 参照
```

HP3478A 制御プログラム

・ 割り込みを使用せずに SRQ が来るのをポーリングしデータを取得するプログラム（ポーリングモード）

- ・ HP3478Aの GPIB アドレスは3に設定しています。
- ・ 接続計測器 HP3478A :
ヒューレットパッカー
デジタルマルチメータ



（操作方法）

最初に、機器側で設定されている GPIB 機器アドレスをエディットボックスに入力します。

初期化ボタンを押して GPIB2K.DLL の初期化を行います。

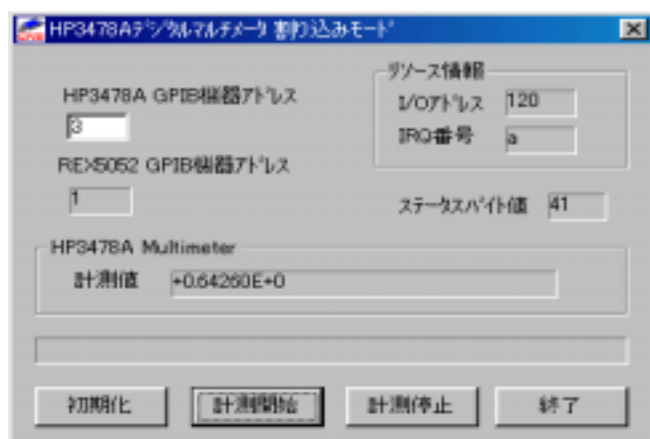
計測開始ボタンで 10 秒間バスラインをよみます。SRQ を調べ信号がきたときのバスラインの計測値を表示します。

（注意）

ポーリングモードでは、CPU を独占し SRQ が来るのをポーリングしているため、他の Windows の作業を行うことができなくなります。

・ SRQ の検知に割り込みを使用し、データを取得するプログラム（割り込みモード）

- ・ HP3478Aの GPIB アドレスは3に設定しています。
- ・ 接続計測器 HP3478A :
ヒューレットパッカー
デジタルマルチメータ



（操作方法）

最初に、機器側で設定されている GPIB 機器アドレスをエディットボックスに入力します。

初期化ボタンを押して GPIB2K.DLL の初期化を行います。

計測開始ボタンで、SRQ 待ちになり、信号がきたとき計測値を表示します。

計測停止ボタンで、SRQ 待ちの状態を止めます。

■ サンプルプログラム抜粋

- gp_cardinfo()により REX-5052 のカードリソース情報を取得します。

(ポーリングモード / 割り込みモード)

```
BOOL Dlg_OnInitDialog (HWND hwnd, HWND hwndFocus, LPARAM lParam)
{
    INT    Status;

    // 自分のカードのリソース情報を取得する...必ずしも呼び出す必要はない
    Status = gp_cardinfo( 0, &MyIOBase, &MyIrqNo );
    if ( Status != 0 )
    {
        sprintf( szBuf, "%s", ERROR );
        SetDlgItemText( hwnd, IDS_IOBASE, szBuf );
        SetDlgItemText( hwnd, IDS_IRQNO, szBuf );
        sprintf( szBuf, "GetMyCardResource エラー [ERROR:%d]", Status );
        SetDlgItemText( hwnd, IDS_STATUS, szBuf );
        return TRUE ;
    }
    // リソース情報を表示する
    sprintf( szBuf, "%x", MyIOBase );
    SetDlgItemText( hwnd, IDS_IOBASE, szBuf );
    sprintf( szBuf, "%x", MyIrqNo );
    SetDlgItemText( hwnd, IDS_IRQNO, szBuf );

    // REX5052 の GP-IB の機器アドレス
    REX5052GPIBAdrs = 1;
    SetDlgItemInt( hwnd, IDS_5052GPIBADRS, REX5052GPIBAdrs, FALSE );
    // HP3478A の GP-IB の機器アドレス
    SetDlgItemText( hwnd, IDE_3478GPIBADRS, "3" );
    EnableWindow( GetDlgItem(hwnd, IDOK), FALSE );

    return TRUE;
}
```

- `gp_init()`で `GPLIB32.DLL` ライブラリを初期化し、`gp_clr()`で GPIB 機器に対してクリアコマンドを送り、機器をリセットします。

(ポーリングモード / 割り込みモード)

```
void Cmd_OnCmdGpInit ( HWND hwnd )
{
    INT          GpStatus;
    CHAR  szCommand[] = "HOKM01";

    // REX-5052 初期化
    GpStatus = gp_init( REX5052GPIBAdrs, MyIOBase, MyIrqNo );
    if( GpStatus != 0 )
    {
        sprintf( szBuf, "gp_init()初期化エラー [ERROR:%d]", GpStatus );
        SetDlgItemText( hwnd, IDS_STATUS, szBuf );
        return;
    }

    // IFC ラインを TRUE にする
    gp_cli();

    // REN ラインを TRUE にする
    gp_ren();

    // HP3478A で設定されている GPIB 機器アドレス取得
    GetDlgItemText( hwnd, IDE_3478GPIBADRS, szHP3478A, sizeof(szHP3478A) );
    GpStatus = gp_tmout(3);
    if ( GpStatus != 0 )
    {
        sprintf( szBuf, "gp_tmout()エラー [ERROR:%d]", GpStatus );
        SetDlgItemText( hwnd, IDS_STATUS, szBuf );
        return;
    }

    // テーブルクリアコマンド送出
    GpStatus = gp_clr( szHP3478A );
    if ( GpStatus != 0 )
    {
        sprintf( szBuf, "gp_clr()エラー [ERROR:%d]", GpStatus );
        SetDlgItemText( hwnd, IDS_STATUS, szBuf );
        return;
    }

    // GPIB 上の全機器のローカルスイッチを無効に設定
    gp_llc();

    // HP3478A GPIB コマンド送信
    GpStatus = gp_wrt( szHP3478A, szCommand );
    if ( GpStatus != 0 )
    {
        sprintf( szBuf, "gp_wrt()エラー [ERROR:%d]", GpStatus );
        SetDlgItemText( hwnd, IDS_STATUS, szBuf );
        return;
    }
    SetDlgItemText( hwnd, IDS_STATUS, "初期化正常終了" );
    EnableWindow( GetDlgItem(hwnd, IDOK), TRUE );
}
```

- HP3478A から受信したデータをダイアログ画面上に表示します。

(ポーリングモード)

```
void Cmd_OnCmdOK ( HWND hwnd )
{
    INT  GpStatus;           // GP-IB の Status byte
    char  RcvData[256];     // 受信バッファ
    char  StatusByte[16];

    // 表示クリア
    SetDlgItemText( hwnd, IDS_STATUS, "" );
    SetDlgItemText( hwnd, IDS_SBYTE, "" );
    SetDlgItemText( hwnd, IDS_READVAL, "" );

    // HP3478A で設定されている GPIB 機器アドレス取得
    GetDlgItemText( hwnd, IDE_3478GPIBADRS, szHP3478A, sizeof(szHP3478A) );

    // トリガコマンド実行
    GpStatus = gp_trg( szHP3478A );
    if ( GpStatus != 0 )
    {
        sprintf( szBuf, "gp_trg()エラー [ERROR:%d]", GpStatus );
        SetDlgItemText( hwnd, IDS_STATUS, szBuf );
        return;
    }

    // 指定時間 SRQ を待つ
    GpStatus = gp_wsrq( 10 );
    if ( GpStatus != 0 )
    {
        sprintf( szBuf, "gp_wsrq()エラー [ERROR:%d]", GpStatus );
        SetDlgItemText( hwnd, IDS_STATUS, szBuf );
        return;
    }

    // シリアルポートを実行してデータを受信
    GpStatus = gp_rds( szHP3478A, StatusByte );
    if( GpStatus != 0 )
    {
        sprintf( szBuf, "ステータスポート gp_rds()エラー [ERROR:%d]", GpStatus );
        SetDlgItemText( hwnd, IDS_STATUS, szBuf );
        return;
    }

    sprintf( szBuf, "%x", StatusByte[0] );
    SetDlgItemText( hwnd, IDS_SBYTE, szBuf );

    // GPIB アドレスからデータをリード
    memset( RcvData, 0x00, sizeof(RcvData) );
    GpStatus = gp_red( szHP3478A, RcvData, sizeof(RcvData) );
    if( GpStatus != 0 )
    {
        sprintf( szBuf, "gp_red()エラー [ERROR:%d]", GpStatus );
        SetDlgItemText( hwnd, IDS_STATUS, szBuf );
        return;
    }

    SetDlgItemText( hwnd, IDS_READVAL, RcvData );
}
```

- `gp_srq()` で HP3478A からの SRQ 検知に割り込みを使用し、データをダイアログ画面上に表示します。

(割り込みモード)

```
void Cmd_OnCmdOK ( HWND hwnd )
{
    INT GpStatus;                // GP-IB の Status byte

    // 表示クリア
    SetDlgItemText( hwnd, IDS_STATUS, "" );
    SetDlgItemText( hwnd, IDS_SBYTE, "" );
    SetDlgItemText( hwnd, IDS_READVAL, "" );

    // HP3478A で設定されている GPIB 機器アドレス取得
    GetDlgItemText( hwnd, IDE_3478GPIBADRS, szHP3478A, sizeof(szHP3478A) );

    // シリアル割り込み実行
    GpStatus = gp_srq( hwnd, ENABLE_SRQ_INTERRUPT );
    if ( GpStatus != 0 )
    {
        sprintf( szBuf, "gp_srq()...エラー [ERROR:%d]", GpStatus );
        SetDlgItemText( hwnd, IDS_STATUS, szBuf );
        return;
    }

    // トリガポート実行
    GpStatus = gp_trg( szHP3478A );
    if ( GpStatus != 0 )
    {
        sprintf( szBuf, "gp_trg()エラー [ERROR:%d]", GpStatus );
        SetDlgItemText( hwnd, IDS_STATUS, szBuf );
        return;
    }
}
```

```
voidDlg_OnUserDefineMessage (HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    INT      GpStatus;                // GP-IB の Status byte
    char     RcvData[256];           // 受信バッファ
    char     StatusByte[16];

    switch ( wParam )
    {
    case EVENT_INTERRUPT:
        // シリアルポートを実行しステータスバイトを受信
        GpStatus = gp_rds( szHP3478A, StatusByte );
        if( GpStatus != 0 )
        {
            sprintf( szBuf, "ステータスバイトリード gp_rds()エラー [ERROR:%d]",
GpStatus );
            SetDlgItemText( hwnd, IDS_STATUS, szBuf );
            return;
        }
        sprintf( szBuf, "%x", StatusByte[0] );
        SetDlgItemText( hwnd, IDS_SBYTE, szBuf );

        // GPIBバスからデータをリード
        memset( RcvData, 0x00, sizeof(RcvData) );
        GpStatus = gp_red( szHP3478A, RcvData, sizeof(RcvData) );
        if( GpStatus != 0 )
        {
            sprintf( szBuf, "gp_red()エラー [ERROR:%d]", GpStatus );
            SetDlgItemText( hwnd, IDS_STATUS, szBuf );
            return;
        }
        SetDlgItemText( hwnd, IDS_READVAL, RcvData );
    break;
    case STOP_INTERRUPT:           // IOリクエスト停止コマンドでシグナルされるイベント発生
        sprintf( szBuf, "SRQ イベント監視を停止しました。" );
        SetDlgItemText( hwnd, IDS_STATUS, szBuf );
    break;
    case ERROR_INTERRUPT:         // その他エラー
    break;
    }
}
```

(3-6) Visual BASIC サンプルプログラム

Visual BASIC 5.0 および 6.0 で、本製品に添付されている“GPLIB2K.DLL”のライブラリを使って REX-5052 を制御するアプリケーションを開発する場合は、サンプルプログラムを参考にしてください。

アプリケーションプログラムから“GPLIB2K.DLL”を呼び出すためには、以下の手順を行ってください。

- DLL ライブラリ関数の Declare 宣言
- OLE カスタムコントロール[MBOX]の登録（割り込みモード使用時）
- OLE カスタムコントロール[MBOX]の追加（割り込みモード使用時）
- フォームに MBOX(OCX)を貼り付ける（割り込みモード使用時）

本製品(Windows2000/XP 用 FD)にはヒューレットパッカード社製 HP3478A(マルチメーター)を使用した以下の2つのサンプルプログラムを添付しております。

2つのサンプルプログラムは、同様の結果が得られます。

- ・ 割り込みを使用せずに SRQ が来るのをポーリングしデータを取得するプログラム（ポーリングモード）
- ・ SRQ の検知に割り込みを使用し、データを取得するプログラム（割り込みモード）

次頁より、アプリケーションプログラムから“GPLIB2K.DLL”を呼び出す方法及びサンプルプログラムについて解説いたします。

Windows95/98/Me で作成したアプリケーションを Windows2000/XP で使用する場合

基本的にはモジュールファイルで DLL 関数の参照宣言を行い、コンパイルすることによって使用可能になります。

また、Windows2000/XP の GPLIB2K.DLL では、割り込みをご使用頂けるようになっているため以下の関数を追加しております。Windows95/98 で作成されたアプリケーションから割り込みを使用するには多少の変更点が必要になります。

gp_srq() ……Page3-34 参照

Step.1 => DLL ライブラリ関数の Declare 宣言

Visual BASIC から“GPLIB2K.DLL”が提供する API 関数を呼び出すためにはモジュール定義ファイルで各 API 関数を Declare 宣言します。API 関数の Declare 宣言は、製品添付のサンプルプログラム“REX5052.BAS”からモジュール定義ファイルにコピーしてください。また、各 API 関数の仕様については「3-4. DLL ライブラリ関数仕様」を参照してください。

Step.2 => OLE カスタムコントロール[MBOX]の登録（割り込みモード使用時）

割り込み制御を行う場合は割り込みハンドラから送られてくるユーザ定義メッセージを Visual BASIC 側のアプリケーションで受け取るために、本製品に添付されている OLE カスタムコントロール MBOX（MBOX5052.OCX）を使用します。

本製品添付の OCX “MBOX5052.OCX”を VB で使用するためには、VB の CD-ROM に添付されているツール“REGSVR32.EXE”を使って OCX のレジストリ登録を行います。“REGSVR32.EXE”は 32 ビットコンソールアプリケーションですので、Windows の DOS BOX から実行します。尚、“REGSVR32.EXE”は VB の CD-ROM に添付されています。

OCX をレジストリ登録するときは、下記構文で実行します。

```
>REGSVR32 “ドライブ名”:¥WINNT¥SYSTEM¥Mbox5052.ocx
```

OCX をレジストリ登録から削除するときは、“/U”を付けて下記構文で実行します。

```
>REGSVR32 /U “ドライブ名”:¥WINNT¥SYSTEM¥Mbox5052.ocx
```



登録成功メッセージ

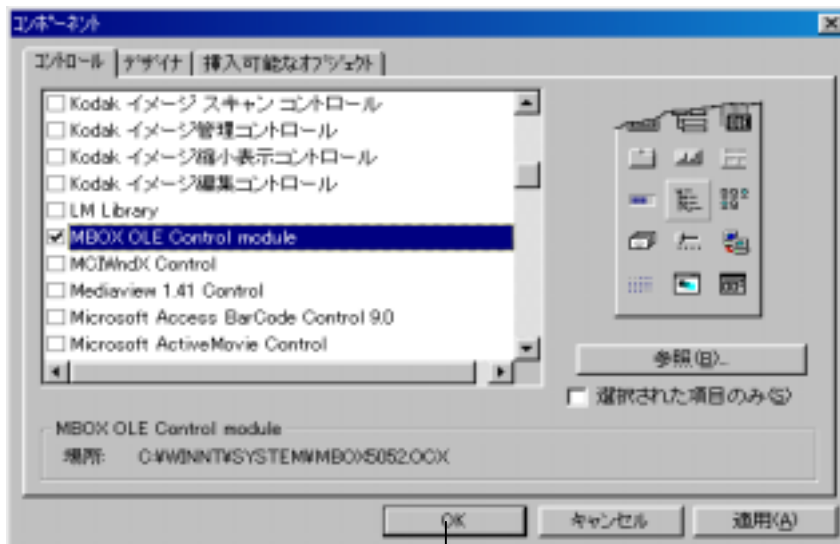


登録削除成功メッセージ

Step.3 => OLE カスタムコントロール[MBOX]の追加（割り込みモード使用時）

VB5.0/6.0 の場合、VB デザインメニューの「プロジェクト」の「コンポーネント」を起動し、利用可能なコントロールから「MBOX OLE Control module」をチェックします。VB ツールバーに MBOX が追加されます。

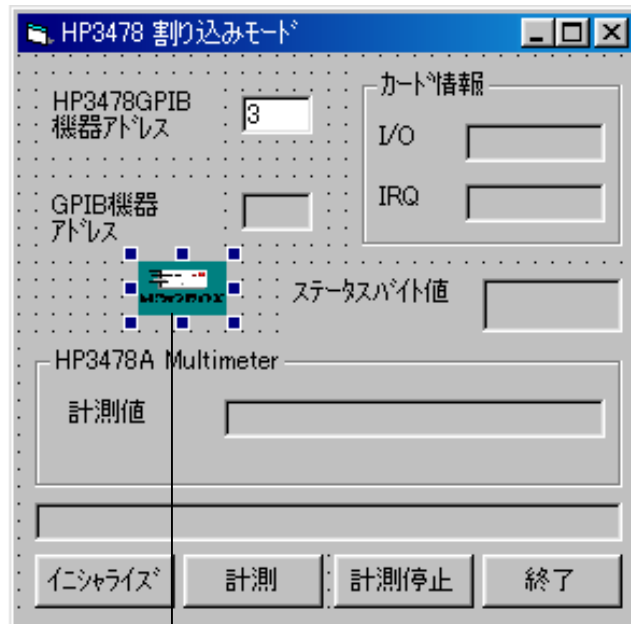
➤ VB5.0/6.0 の場合



Step.4 => フォームに MBOX(OCX)を貼り付ける (割り込みモード使用時)

フォームを作成し、割り込みハンドラが割り込み起動元プログラムに送るユーザ定義メッセージを受け取るための MBOX(MBOX5052.OCX)を貼り付けます。これにより、割り込みが発生すると MBOX がサービスするプロシージャ

MBOX5052_OnMsgPost(ByVal wParam As Integer, ByVal lParam As Long) が呼び出されます。この中で、割り込み通知に同期した処理を記述します。



```

Project1 - HP3478 Q-T?
MBOX5052
OnMsgPost

Private Sub MBOX5052_OnMsgPost(ByVal wParam As Integer, ByVal lParam As Long)
    Dim Code As Long

    ' システムコールを実行しステータスバイトを受信
    Status = sp_rds(GpAdrs, Code)
    If Status <> 0 Then
        ERROR.Text = "sp_rds()エラー : " & Status
        Exit Sub
    End If
    SBYTE.Text = Hex(Code)

    ' GPIBからカートにデータを送る
    szBuf = String(256, &H0)
    Status = sp_red(GpAdrs, szBuf, Len(szBuf))
    If Status <> 0 Then
        ERROR.Text = "sp_red()エラー : " & Status
        Exit Sub
    End If
    READVAL.Text = szBuf
End Sub

```

HP3478A 制御プログラム

・ 割り込みを使用せずに SRQ が来るのをポーリングしデータを取得するプログラム（ポーリングモード）

- ・ HP3478AのGPIBアドレスは3に設定しています。
- ・ 接続計測器 HP3478A :
ヒューレットパッカー
デジタルマルチメータ



（操作方法）

最初に、機器側で設定されている GPIB 機器アドレスをエディットボックスに入力します。

初期化ボタンを押して REX-5052 の初期化を行います。

計測開始ボタンで 10 秒間バスラインをよみます。SRQ を調べ信号がきたときのバスラインの計測値を表示します。

（注意）

ポーリングモードでは、CPU を独占し SRQ が来るのをポーリングしているため、他の Windows の作業を行うことができなくなります。

・ SRQ の検知に割り込みを使用し、データを取得するプログラム

（割り込みモード）

- ・ HP3478AのGPIBアドレスは3に設定しています。
- ・ 接続計測器 HP3478A :
ヒューレットパッカー
デジタルマルチメータ



（操作方法）

最初に、機器側で設定されている GPIB 機器アドレスをエディットボックスに入力します。

初期化ボタンを押して REX-5052 の初期化を行います。

計測開始ボタンで、SRQ 待ちになり、信号がきたとき計測値を表示します。

計測停止ボタンで、SRQ 待ちの状態を止めます。

■ サンプルプログラム抜粋

- gp_cardinfo()により REX-5052 のカードリソース情報を取得します。

(ポーリングモード / 割り込みモード)

```
Private Sub Form_Load()

    OK.Enabled = False
    ' GPIB 機器アドレスを 3 に設定する
    GpAdrs = 3
    GpibAdrs.Text = 3
    ' Rex5052 の GPIB アドレスのデフォルト値設定
    MyGpibAdrs = 1
    MyAdrs.Text = MyGpibAdrs

    ' スロットに挿入されている REX5052 GPIB カードのリソース情報を取得する
    Status = gp_cardinfo(0, MyIOBase, MyIrqNo)
    If (Status = 0) Then
        ' リソース情報を表示する
        IO.Text = Hex(MyIOBase)
        Irq.Text = Str(MyIrqNo)
    Else
        IO.Text = "Fail"
        Irq.Text = "Fail"
        ERROR.Text = "gp_cardinfo()エラー : " & Status
        INIT.Enabled = False
        Exit Sub
    End If
End Sub
```

- **gp_init()**で REX-5052 を初期化し、**gp_clr()**で GPIB 機器に対してクリアコマンドを送り、機器をリセットします。

(ポーリングモード / 割り込みモード)

```
Private Sub INIT_Click()

    GpAdrs = GpibAdrs.Text
    ' REX-5052 初期化
    Status = gp_init(MyGpibAdrs, MyIOBase, MyIrqNo)
    If Status <> 0 Then
        ERROR.Text = "gp_init()エラー：" & Status
        Exit Sub
    End If
    ' IFCラインを TRUE にする
    Status = gp_cli()
    If Status <> 0 Then
        ERROR.Text = "gp_cli()エラー：" & Status
        Exit Sub
    End If
    ' RENラインを TRUE にする
    Status = gp_ren()
    If Status <> 0 Then
        ERROR.Text = "gp_ren()エラー：" & Status
        Exit Sub
    End If

    ' デバイスカリアコマンド送出
    Status = gp_clr(GpAdrs)
    If Status <> 0 Then
        ERROR.Text = "gp_clr()エラー：" & Status
        Exit Sub
    End If
    Status = gp_llo()
    If Status <> 0 Then
        ERROR.Text = "gp_llo()エラー：" & Status
        Exit Sub
    End If
    ' HP3478A GPIB コマンド送信
    Status = gp_wrt(GpAdrs, "HOKM01")
    If Status <> 0 Then
        ERROR.Text = "gp_wrt()エラー：" & Status
        Exit Sub
    End If
    ERROR.Text = "初期化正常終了"
    OK.Enabled = True

End Sub
```

- HP3478A から受信したデータをダイアログ画面上に表示します。

(ポーリングモード)

```
Private Sub OK_Click()  
    Dim Code As Long  
  
    ' トリガコマンド実行  
    Status = gp_trg(GpAdrs)  
    If Status <> 0 Then  
        ERROR.Text = "gp_trg()エラー : " & Status  
        Exit Sub  
    End If  
  
    ' 指定時間SRQを待つ  
    Status = gp_wsrq(10)  
    If Status <> 0 Then  
        ERROR.Text = "gp_wsrq()エラー : " & Status  
        Exit Sub  
    End If  
    ' シリアルポートを実行しステータスバイトを受信  
    Status = gp_rds(GpAdrs, Code)  
    If Status <> 0 Then  
        ERROR.Text = "gp_rds()エラー : " & Status  
        Exit Sub  
    End If  
    SBYTE.Text = Hex(Code)  
  
    ' GPIBバスからデータをリード  
    szBuf = String(256, &H0)  
    Status = gp_red(GpAdrs, szBuf, Len(szBuf))  
    If Status <> 0 Then  
        ERROR.Text = "gp_red()エラー : " & Status  
        Exit Sub  
    End If  
  
    READVAL.Text = szBuf  
End Sub
```

- `gp_srq()` で HP3478A からの SRQ 検知に割り込みを使用し、割り込みが発生すると `gp_red()` によりデータを読み取ります。

(割り込みモード)

```
Private Sub OK_Click()  
    Dim OleHandle As Integer          ' MBOX5052.OCX ハンドル  
  
    'OLE のウィンドウハンドル取得  
    OleHandle = MBOX5052.GetMboxWnd  
    If (OleHandle = 0) Then  
        MsgBox "OLE のハンドルが取得できません。", vbOKOnly + vbCritical, "エラー"  
        Exit Sub  
    End If  
    ' シリアルポート割り込み実行  
    Status = gp_srq(OleHandle, ENABLE_SRQ_INTERRUPT)  
    If Status <> 0 Then  
        ERROR.Text = "gp_srq()エラー : " & Status  
        Exit Sub  
    End If  
  
    ' トリガポート実行  
    Status = gp_trg(GpAdrs)  
    If Status <> 0 Then  
        ERROR.Text = "gp_trg()エラー : " & Status  
        Exit Sub  
    End If  
  
End Sub
```

```
Private Sub MBOX5052_OnMsgPost(ByVal wParam As Integer, ByVal lParam As Long)  
    Dim Code As Integer  
  
    ' シリアルポートを実行しステータスバイトを受信  
    Status = gp_rds(GpAdrs, Code)  
    If Status <> 0 Then  
        ERROR.Text = "gp_rds()エラー : " & Status  
        Exit Sub  
    End If  
    SBYTE.Text = Hex(Code)  
  
    ' GPIBバスからデータをリード  
    szBuf = String(256, &H0)  
    Status = gp_red(GpAdrs, szBuf, Len(szBuf))  
    If Status <> 0 Then  
        ERROR.Text = "gp_red()エラー : " & Status  
        Exit Sub  
    End If  
  
    READVAL.Text = szBuf  
End Sub
```

(空白ページ)

第4章 MS-DOSでの使用

(4-1) イネーブラのインストール

PCカードを使用するためには、PCカードをイネーブルするという作業が必要になります。PCカードのイネーブルを行うために、イネーブラのインストールを行う必要があります。DOS/Vをお使いの場合は、カードサービス対応イネーブラとポイントイネーブラを用意していますので、最初にどちらを使用するか選択してください。

◆ DOS/V 版カードサービス対応イネーブラのインストール

添付フロッピーディスクからハードディスクにカードサービス対応イネーブラをコピーしてください。

REXGPCS.EXE はデバイスドライバー形式ですので、CONFIG.SYS に登録して使います。

```
C:>COPY A:%PCMCIA%DOSV%REXGPCS.EXE C:%CARD
```

◆ DOS/V 版ポイントイネーブラのインストール

添付フロッピーディスクからハードディスクにポイントイネーブラをコピーしてください。

REXGP365.EXE は、DOS プロンプトから実行します。

```
C:>COPY A:%PCMCIA%DOSV%REXGP365.EXE C:%CARD
```

◆ PC-98 版カードサービス対応イネーブラのインストール

添付フロッピーディスクからハードディスクに PC-98 用カードサービス対応イネーブラをコピーしてください。

REXGPCS.EXE はデバイスドライバー形式ですので、CONFIG.SYS に登録して使います。

```
A:>COPY C:%PCMCIA%PC98%REXGPCS.EXE A:%CARD
```

㊦ カードイネーブラとは... ㊦

パソコンのスロットに挿入した直後はメモリーカードとして認識されており、I/O カードとしての動作はしていません。このメモリーカードの中には、PCカードをI/Oカードにコンフィギュレーションするために必要な情報(カード属性情報)が書き込まれています。

PCカードをI/Oカードとして機能させるためには、コンフィギュレーションソフト「イネーブラ」が必要となります。イネーブラは、PCカードのカード属性情報を読み込んだ後、その情報に基づいてPCカードを所定のI/Oカードにコンフィギュレーションします。イネーブラによるコンフィギュレーションが正常に行なわれて、はじめてPCカードはI/Oカードとして使える状態になります。

☞ DOS/V 版対応カードサービスについて... ☞

カードサービスはパソコン本体に添付しているソフトウェアでソケットサービス(SS)・カードサービス(CS)・リソースマネージャ・コモンイネーブラ等のドライバがセットになっています。本製品は PCMCIA Release 2.0 以降の下記カードサービスに対応しています。

CSバージョン識別名	SS,CSドライバ名	搭載パソコン機種
IBM 版 PlayAtWill 2.xx / 3.xx	IBMDSS01.SYS, IBMDOSCS.SYS	IBM ThinkPad
IBM 版 PCMCIA 2.00 相当	IBMDSS01.SYS, IBMDOSCS.SYS	IBM ThinkPad Panacom PRONOTE jet
IBM PCMCIA 1.07 相当	IBMDSS02.SYS, IBMDOSCS.SYS	IBM ThinkPad
SystemSoft 版 CardSoft PCMCIA2.01 相当 v4.1x PCMCIA2.10 相当 v2.0x	SS365SL.EXE,SSCIRRUS.EXE,SSD BOOK.EXE, SSVADEM.EXE, CS.EXE,CSALLOC.EXE	SOTEC WINBook, IDEXON NT66CL2, DELL Latitude
SystemSoft 版 CardSoft PCMCIA2.0 相当 v2.0x	SSVLSI.EXE・CS.EXE,CSALLOC.E XE	COMPAQ CONTURA AERO 4/25,4/33C
Phoenix Technologies 版 CARD Manager Plus PCMCIA2.00 相当 v1.0 PCMCIA2.1 相当 v2.2x	PCCMSS.EXE, PCMCS.EXE	FUJITSU FMV Note, TOSHIBA DynaBook
DATABOOK 版 CardTalk	SOCKET.SYS, CTALKCCS.EXE, CARDTALK.SYS	MDT Arowana

注1) PCMCIA ドライバとして、Phoenix Technologies の PCMCU が提供されている機種(Olivetti QUADERUNO 33/J)では動作しません。IBM PC-DOSS J6.1/V,6.3/V または、PlayAtWill 等のカードサービスを別途お買い求めになるか、PCMCU を登録しないで本製品添付のポイントイネーブラを使ってイネーブルしてください。

注2) DATABOOK CardTalk v2.20.12,v2.20.12 はソケットサービスしかサポートしてませんので動作しません(PCiN P-NOTE,AT&T WaveNote,MDT, Arowana の発売初期の機種)。カードサービス版の CardTalk を入手してください。

(4-1-1) DOS/V 版カードサービス対応イネーブラを使用する場合

最初に、カードサービスのインストールが完了しているか確認してください。カードサービスのインストール方法については、パソコン側のマニュアル記載内容に従ってください。カードサービスのインストールが完了していれば、本製品添付のカードサービス版イネーブラをカードサービスの後に追加するだけです。次頁以降に CONFIG.SYS の登録例を示します。CONFIG.SYS の内容はお使いの機種によってまちまちですので、登録例の通りに修正する必要はありません。

オプション仕様

```
DEVICE=C:\CARD\REXGPCS.EXE [/<オプション>][ ] ... [ ]
```

オプション	説明
/P = x	I/O ベースアドレス x を 16 進表記で指定 カードに割り当てる I/O ベースアドレスを 16 進表記で指定します。何も指定しない場合は 300h にアドレスを割り当てます。
/I = n	割り込み番号 n を 10 進表記で指定 指定可能な割り込み番号は、5,7,10,11,12,15 になります。何も指定しない場合は、割り込みは使用しません。
/T = s	ビープ音の有無 s を ON/OFF で指定 カードをイネーブルした時に出すビープ音の有無を指定します。省略した場合、または“ON”が指定された場合はビープ音を出し“OFF”の場合はビープ音を出しません。
/S = n	スロット番号 n を指定 “1”でソケット 0 固定。“2”でソケット 1 固定になります。 省略した場合はスロットを順に調べてイネーブルします。 (省略の場合はオートサーチ)

【登録が正常に行えなかった場合】

イネープラの登録が正常に行えた場合には"カードサービスへのクライアント登録を完了しました"とのメッセージが出力されます。登録に失敗した場合には下記のメッセージが出力されます。

- 1) " GPIBカードイネープラのクライアント登録ができません。 "
- 2) "カ - ドサ - ビスが常駐していません。 "
- 3) "無効なカ - ドサ - ビスが常駐しています。 "
- 4) "有効なロジカルソケットが見つかりません。 "

- 2)のメッセージの場合には、カードサービスプログラムをインストールしてから再度ドライバの登録を行ってください。
- 1), 3), 4)のメッセージの場合には、カードサービスの登録部分に問題があると思われるので弊社ユーザサポートまでお問い合わせください。

【登録が正常に行われた後のカードイネーブル】

REX-5052 のカードイネープラは、デバイスドライバとして主記憶上に常駐し、カードが挿入された時にカードサービスから呼び出されます(コールバック)。その時に、PCカードより情報を読み出し、REX-5052 カードであればカードのイネーブル処理を行います。その時にBEEPがONであれば以下のようにBEEP音を発生させます。

- | | |
|-----------------------------------|----------|
| 1) 正常にカードのイネーブルが行われた | BEEP音が1回 |
| 2) 正常にカードのイネーブルが行われなかった | BEEP音が3回 |
| 3) 既にカードサービスなどでカードのイネーブルが行われている場合 | BEEP音が5回 |

- 1)及び3)のケースだとカードは正常に使用することが出来ます。ただし3)の場合はREXGPCS.EXEに対しオプションで設定したI/Oアドレス、割り込み番号が無効となっていますので、弊社にて用意したカードサービスに対する問い合わせのライブラリを使用してI/Oの番地など情報を求める必要があります。ただしGPBIO\$またはGPLIBを使用している場合はその必要がありません。
- 2)の場合には、カードをイネーブル出来ない状態ですので、カードは使用することができません。この場合は以下のことが考えられます。
 - a) 指定したI/Oアドレスに既に他のカード(ポート)が存在している。
 - b) 指定した割り込み番号は既に他のカード(ポート)が使用している。
 - c) 指定したソケット番号と逆のスロットにカードを挿入した
オプションの引数を変えて、再度パソコンを立ち上げなおしてください。

CONFIG.SYS 記述例1 => IBM カードサービス"PlayAtWill"の場合

```
DEVICE=C:\WINDOWS\EMM386.EXE RAM X=C800-CFFF (1)
.....
DEVICEHIGH=C:\EZPLAY\SSDPCIC1.SYS (2)
DEVICEHIGH=C:\EZPLAY\IBMDOSCS.SYS (3)
DEVICEHIGH=C:\EZPLAY\RMUDOSAT.SYS /SH=1 /NS=1 /MA=C800-CFFF (4)
.....
DEVICEHIGH=C:\EZPLAY\AUTODRV.SYS (5)
.....
DEVICE=C:\CARD\REXGPCS.EXE /P=300 /I=5 /S=0 /T=ON (6)
```

【解説】

- (1) 拡張メモリマネージャが [C800 ~ CFFF] のメモリウィンドウセグメントを使用しないように指定しています。
- (2) ソケットサービスを起動しています。
ソケットサービスファイル名はインストール時に選択したマシーンにより異なります。
- (3) カードサービスを起動しています。
- (4) リソースマップユーティリティに対しカードサービスが [C800 ~ CFFF] のメモリウィンドウセグメントを使用するように指定しています。
- (5) カードサービス標準イネーブラを起動しています。
- (6) 本製品添付のカードサービス版イネーブラを起動しています。
I/O ベースアドレスを 300h、割り込みを 5、ソケットを Sokect0 固定、ピープ音を ON となるように指定しています。

CONFIG.SYS 記述例 2 => COMPAQ カードサービスの場合

```

DEVICE=C:\DOS\EMM386.EXE 1024 X=D000-DFFF (1)
DEVICE=C:\CPQDOS\SSVLSI.EXE (2)
DEVICE=C:\CPQDOS\CS.EXE (3)
DEVICE=C:\CPQDOS\CSALLOC.EXE (4)
INSTALL=C:\CPQDOS\CARDID.EXE C:\CPQDOS\CARDID.INI (5)
.....
DEVICE=C:\CARD\REXGPCS.EXE /P=300 (6)

```

【解説】

- (1) 拡張メモリマネージャが[D000～DFFF]のメモリウィンドウセグメントを使用しないように指定しています。
- (2) ソケットサービスを起動しています。
- (3) カードサービスを起動しています。
- (4) リソースマネージャを起動しています。
- (5) カードサービス添付の標準イネーブラを起動しています。
- (6) 本製品添付のカードサービス版イネーブラを起動しています。
I/O ベースアドレスを 300h に割り当て、割り込みは使用しません。

CONFIG.SYS 記述例 3 => TOSHIBA カードサービスの場合

```

DEVICE=C:\DOS\EMM386.EXE RAM P0=D000 P1=D400 P2=D800 P3=DC00
I=B000-B7FF X=C800-C8FF (1)
.....
DEVICE=C:\PCPLUS3\CNFIGMAN.EXE /DEFAULT (2)
DEVICE=C:\PCPLUS3\PCMSS.EXE (3)
DEVICE=C:\PCPLUS3\PCMCS.EXE
DEVICE=C:\PCPLUS3\PCMRMAN.SYS
DEVICE=C:\PCPLUS3\PCMSCD.EXE (4)
.....
DEVICE=C:\CARD\REXGPCS.EXE /P=300 /I=5 (5)

```

【解説】

- (1) 拡張メモリマネージャが[C800～C8FF]のメモリウィンドウセグメントを使用しないように指定しています。(1行に記述してください)
- (2) ソケットサービスを起動しています。
- (3) カードサービスを起動しています。
- (4) カードサービス添付の標準イネーブラを起動しています。
- (5) 本製品添付のカードサービス版イネーブラを起動しています。
I/O ベースアドレスを 300h、割り込みを 5 となるように指定しています。

(4-1-2) DOS/V 版ポイントイネーブラを使用する場合

カードサービスが提供されていない機種で GPIB カードをイネーブルすることができます。また、カードサービス等のドライバをメモリーに常駐させるとコンベンショナルメモリーの空き領域が不足して不都合が生じることがあります。このような場合、ポイントイネーブラを使ってカードのイネーブルを行います。

ポイントイネーブラは、パソコン本体のメモリーウィンドウを通してカードの情報を読み出します。EMM386.EXE が CONFIG.SYS に組み込まれている場合には、</X=>オプションで[DF000h ~ DFFFFh]の 4K バイトのメモリーウィンドウを確保してください。

オプション仕様

```
C:¥CARD>REXGP365.EXE [/<オプション>] [ ] ... [ ]
```

オプション	説明
/P = x	I/O ベースアドレス x を 16 進表記で指定 カードに割り当てる I/O ベースアドレスを 16 進表記で指定します。何も指定しない場合は 300h にアドレスを割り当てます。
/I = n	割り込み番号 n を 10 進表記で指定 指定可能な割り込み番号は、5,7,10,11,12,15 になります。何も指定しない場合は、割り込みは使用しません。
/S = n	スロット番号 n を指定 /S オプションを省略した場合はスロットを順に調べてイネーブルします。スロットを指定する場合は、/S=1 か/S=2 を追加します。
/T = s	ビープ音の有無 s を ON/OFF で指定 カードをイネーブルした時に出すビープ音の有無を指定します。省略した場合、または“ON”が指定された場合はビープ音を出し“OFF”の場合はビープ音を出しません。
/MEM = x	使用するメモリーウィンドウセグメントアドレスを 10 進表記で指定 指定しないときは DF00 から 4K バイトを使います。 EMM386.EXE の</X=>オプションでイクスクルードしたメモリーウィンドウの範囲と一致するようにしてください。

[登録が正常に行えなかった場合]

イネーブラの登録が正常に行えた場合には

```
"REX5052 GPIB PC Card Point Enabler V *.* (9*****)"
"For PCIC Intel 82365SL"
"(C) Copyright RATOC System Inc. Osaka city,Japan"
```

とのメッセージが出力され、ビーブ音が1度なります(ただしBEEPオプションでOFFにした場合はBEEPは鳴りません)。登録に失敗した場合には3回のビーブ音でエラーを知らせ、下記のメッセージが出力されます。

- 1) "カードサービスを登録しないでください。"
- 2) "コマンドオプションの指定書式が不正です。"
- 3) "メモリウィンドウが取得できません(GPIBカード挿入確認)。"
- 4) "製品情報タプルが取得できません。"
- 5) "GPIBカードが挿入されていません。"
- 6) "コンフィギュレーションタプルが取得できません。"
- 7) "IRQが取得できません。"
- 8) "GPIBカードのコンフィギュレーションができません。"

- 1)の場合、既にカードサービスがインストールされていますので、カードサービス版のイネーブラを使用するか、カードサービスをConfig.sysより削除するかを選択してください。
- 2)の場合、コマンドラインオプションをもう一度見直してください。
- 3), 4)の場合EMM386のXオプションとイネーブラのMEMオプションの値が合っているかどうか確認してください。
- 5)の場合は、GPIBカードをスロットに正しく挿入し、またスロットオプションをオートスキャンにしてもう一度パソコンを起動してください。
- 7), 8)の場合カードをイネーブル出来ない状態ですので、カードは使用することができません。

この場合は以下のことが考えられます。

- a) 指定したI/Oアドレスに既に他のカード(ポート)が存在している。
- b) 指定した割り込み番号は既に他のカード(ポート)が使用している。
- c) 指定したソケット番号と逆のスロットにカードを挿入した。
オプションの引数を変えて、再度パソコンを立ち上げなおしてください。

(4-1-3) PC-98 版カードサービス対応イネーブラを使用する場合

最初に、カードサービスのインストールが完了しているか確認してください。カードサービスのインストール方法については、パソコン側のマニュアル記載内容に従ってください。カードサービスのインストールが完了していれば、本製品添付のカードサービス版イネーブラをカードサービスの後に追加するだけです。次頁以降に CONFIG.SYS の登録例を示します。

オプション仕様

```
DEVICE=A:¥CARD¥REXGPCS.EXE [/<オプション>] [ ] … [ ]
```

オプション	説明
/P = x	I/O ベースアドレス x を 16 進表記で指定 カードに割り当てる I/O ベースアドレスを 16 進表記で指定します。何も指定しない場合は 0D0h にアドレスを割り当てます。
/I = n	割り込み番号 n を 10 進表記で指定 何も指定しない場合は、割り込みは使用しません。 指定可能な割り込み番号は、3,5,6,10,12,13 になります。
/T = s	ビープ音の有無 s を ON/OFF で指定 カードをイネーブルした時に出すビープ音の有無を指定します。省略した場合、または“ON”が指定された場合はビープ音を出し“OFF”の場合はビープ音を出しません。
/S = n	スロット番号 n を指定 /S オプションを省略した場合はスロットを順に調べてイネーブルします。スロットを指定する場合は、/S=1 か /S=2 を追加します。

☞ PC-9800 シリーズ版対応カードサービスについて... ☞

PC-9800 シリーズで初期の機種では注 1)のソケットサービスしか提供されておらず、本製品添付のイネーブラを使ってカードをイネーブルすることはできません。別売版カードサービスを入手してください。PC-9800 シリーズ対応カードサービスと搭載機種は下表の通りです。

CSバージョン識別名	SS,CSドライバ名	搭載パソコン機種
別売版カードサービス SystemSoftCardSSoft2.10 Version2.06	SSMECIA.SYS, CS.EXE	PC-9821 Ne PC-9801 NX/C,P,NS/A,NL/R
標準カードサービス SystemSoftCardSSoft2.10	SSDRV.EXE, CS.EXE	PC-9821 Np,Ns,Ne2,Nd,Ld Nf,Nm,Lt,Ne3,Nd2 PC-9801 NL/A

注1) ソケットサービス NEC SocketService 2.00 Version 1.00

[登録が正常に行えなかった場合]

イネーブラの登録が正常に行えた場合には"カードサービスへのクライアント登録を完了しました"とのメッセージが出力されます。登録に失敗した場合には下記のメッセージが出力されます。

- 1) " GPIBカードイネーブラのクライアント登録ができません。 "
 - 2) "カ - ドサ - ビスが常駐していません。 "
 - 3) "無効なカ - ドサ - ビスが常駐しています。 "
 - 4) "有効なロジカルソケットが見つかりません。 "
- 2)のメッセージの場合には、カードサービスプログラムをインストールしてから再度ドライバの登録を行ってください。
 - 1), 3), 4)のメッセージの場合には、カードサービスの登録部分に問題があると思われるので弊社ユーザサポートまでお問い合わせください。

[登録が正常に行われた後のカードイネーブル]

REX-5052 のカードイネーブラは、デバイスドライバとして主記憶上に常駐し、カードが挿入された時にカードサービスから呼び出されます(コールバック)。その時に、PC カードより情報を読み出し、REX-5052 カードであればカードのイネーブル処理を行います。

その時に BEEP が ON であれば以下のように BEEP 音を発生させます。

1) 正常にカードのイネーブルが行われた	BEEP 音が1回
2) 正常にカードのイネーブルが行われなかった	BEEP 音が3回
3) 既にカードサービスなどでカードのイネーブルが行われている場合	BEEP 音が5回

- 1)及び3)のケースだとカードは正常に使用することが出来ます。ただし3)の場合は REXGPCS.EXE に対しオプションで設定した I/O アドレス、割り込み番号が無効となっていますので、弊社にて用意したカードサービスに対する問い合わせのライブラリを使用して I/O の番地など情報を求める必要があります。ただし GPBIOS または GPLIB を使用している場合はその必要がありません。
- 2)の場合には、カードをイネーブル出来ない状態ですので、カードは使用することができません。

この場合は以下のことが考えられます。

- a) 指定した I/O アドレスに既に他のカード(ポート)が存在している。
- b) 指定した割り込み番号は既に他のカード(ポート)が使用している。
- c) 指定したソケット番号と逆のスロットにカードを挿入した。
オプションの引数を変えて、再度パソコンを立ち上げなおしてください。

CONFIG.SYS 記述例 4 => NEC 添付のカードサービス

- PC-9821 Np,Ns,Ne2,Nd,Ld,Nf,Nm,Lt,Ne3,Nd2
- PC-9801 NL/A

```

DEVICE=A:¥DOS¥HIMEM.SYS
DEVICE=A:¥DOS¥EMM386.EXE /UMB /E=DC00-DFFF          (1)
.....
DEVICE=A:¥DOS¥SSDRV.SYS                              (2)
DEVICE=A:¥DOS¥CS.EXE                                 (3)
DEVICE=A:¥DOS¥CSALLOC.EXE A:¥DOS¥CSALLOC.INI        (4)
INSTALL=A:¥DOS¥CARDID.EXE A:¥DOS¥CARDID.INI         (5)
.....
DEVICE=A:¥CARD¥REXGPCS.EXE /P=D0 /I=3                (6)

```

【解説】

- (1) 拡張メモリマネージャが[DC00 ~ DFFF]のメモリウィンドウセグメントを使用しないように指定しています。
- (2) ソケットサービスを起動しています。
- (3) カードサービスを起動しています。
- (4) リソースマネージャがCSALLOC.INIを参照するようにして起動しています。
- (5) カードサービス添付の標準イネーブラを起動しています。
- (6) 本製品添付のカードサービス版イネーブラを起動しています。
カードにI/O ベースアドレス D0h・IRQ3 を割り当てます。

- PC-9821 Ne
- PC-9801 NX/C,P,NS/A,NL/R

```

DEVICE=A:¥DOS¥HIMEM.SYS
DEVICE=A:¥DOS¥EMM386.EXE /UMB /E=DA00-DBFF          (1)
.....
DEVICE=A:¥DOS¥SSMECIA.SYS                            (2)
DEVICE=A:¥DOS¥CS.EXE                                 (3)
DEVICE=A:¥DOS¥CSALLOC.EXE A:¥DOS¥CSALLOC.INI        (4)
INSATLL=A:¥DOS¥CARDID.EXE A:¥DOS¥CARDID.INI         (5)
.....
DEVICE=A:¥CARD¥REXGPCS.EXE /P=D0                    (6)

```

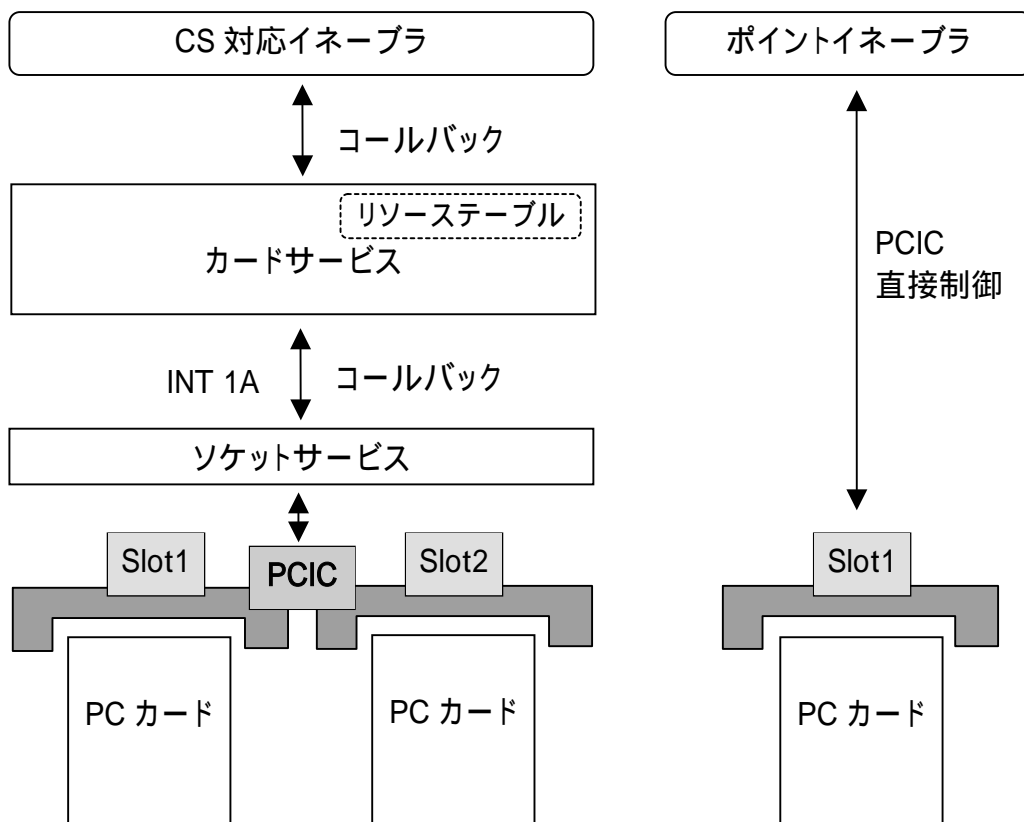
【解説】

- (1) ~ (5) 上記解説参照
- (6) 本製品添付のカードサービス版イネーブラを起動しています。
I/O ベースアドレスを D0h に割り当て、割り込みは使用しません。

☞ カードサービス対応イネーブラとポイントイネーブラ ☞

カードサービス(CS)対応イネーブラは起動された時点で、CSのファンクションセットである GetCardServiceInfo により、CS が常駐しているかチェックします。CS が常駐していれば、イネーブラは CS のファンクションセット RegisterClient により、カードが抜き差しされた時 CS がイネーブラを呼び出すために必要なコールバック情報を登録しメモリに常駐します。PC カードが挿入または抜き取られると、CS は登録されたコールバック情報をもとに全てのイネーブラに抜き差しの通知を行います。CS は、複数の PC カードが使用する I/O アドレス・IRQ のリソースをリソース管理テーブルで管理します。同時に、上記のカード抜き差しの監視を行います。図で示すようにカードが挿入されるとそれを検出してイネーブラに通知します。イネーブラは CS からの通知を受けて自分のカードかどうか調べます。自分のカードの時は、CS に対し必要な I/O アドレスおよび IRQ を割り当ててくれるようにリソースの要求とイネーブルの要求を発行します。この要求を受けて CS は要求されたリソースが他で使われていなければ、ソケットサービス(SS)と呼ばれる低レベルのファンクションセットを呼び出してリソースを確保しカードのイネーブルを行います。

ポイントイネーブラは、PC Card Interface Controller(PCIC)を直接制御してカードをイネーブルします。カードの抜き差しの管理は行いません。



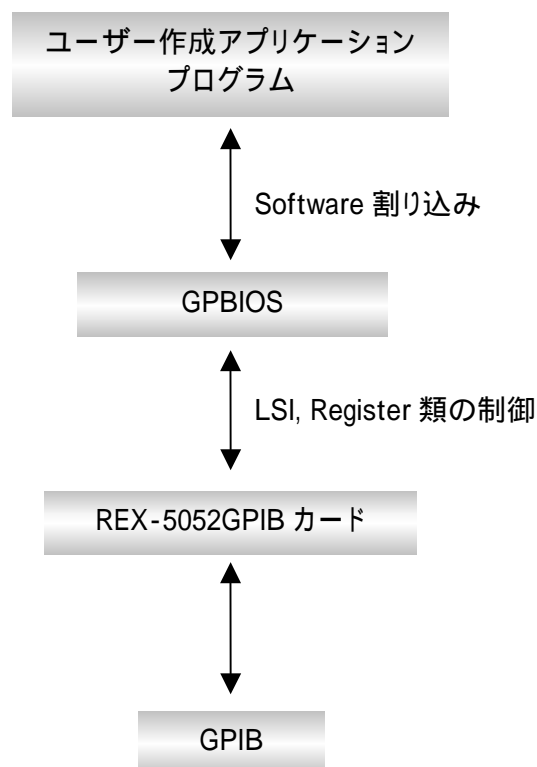
(4-2) GPBIOS

GPBIOS は、REX-5052 カードのハードウェアを直接制御し、GPIB のバスプロトコルを実現するための入出力ルーチンで、上位のアプリケーションプログラムよりコールすることにより、GPIB を駆動することができます。GPIB を駆動する上での個々の信号線のすべてはこの GPBIOS が実行しますので、アプリケーションプログラムの作成に際して、プログラムの負担を軽減します。

また、DOS 上の C 言語ライブラリ、N88Basic 用リンクは、全て GPBIOS をコールしています。そのため、REX-5052 を C 言語ライブラリで使用する場合、また N88Basic 用リンクを使用する場合には、必ず GPBIOS をロードする必要があります。

(4-2)では、この GPBIOS の単独の使用方法について述べてあります。

GPBIOS を単独で使用する場合



(4-2-1) GPBIOS の使用方法

GPBIOS は、レジスタインターフェイスと RCB インターフェイスを持ち、リクエストによりどちらのインターフェイスを使用するかが決められています。

◆RCB インターフェイス

RCB インターフェイスでは、メモリ上の RCB(RequestControlBlock)と呼ばれるパラメータ受渡し用の領域を使用して、GPBIOS を呼び出します。RCB の先頭には、呼び出す機能を指定するためのリクエストコードをセットし、2 バイト目以降には、トーカーアドレス、リスナーアドレスの順にセットします。それ以降は、各機能固有のパラメータをセットする領域となります。

RCB の標準構成を下記に示します。

内容	Size
リクエストコード	1Byte
トーカーアドレス	1Byte
リスナーアドレス	15Byte
固有パラメータ	0 ~ 8Byte

GPBIOS を呼び出すためには、RCB 内に必要なリクエストコード、アドレス、パラメータ類をセットした後、RCB の先頭オフセットアドレスを DX に、セグメントアドレスを DS に入れ、ソフトウェアインタラプト 242(INT242)を実行します。

GPBIOS からの復帰時には、CarryFlag でエラーの有無が示され、Carry=1 の場合には異常終了(エラー発生)を、Carry=0 の場合には正常終了(エラーなし)を示します。また、異常終了時には DL レジスタ内にエラーの意味を示す、エラーコードが格納されています。

RCB インターフェイスでは、AX,DX,CarryFlag を除くすべてのレジスタの値は保存されます。

◆レジスタインターフェイス

レジスタインターフェイスでは、RCB を使用せずに、CPU 内の特定のレジスタ内にリクエストコードやパラメータを入れて、GPBIOS を呼び出します。従って、受け渡すパラメータの数が少ない場合に使用されます。

レジスタインターフェイス使用時には、下記の様に AH 内にリクエストコードを、DL 内にパラメータをセットした後、ソフトウェアインタラプトの 243(INT243)を実行します。

レジスタ	内容
AH	リクエストコード
DL	パラメータ

ただし、INIT コマンドのみ、DX 及び AL にパラメータがセットされます。

GPBIOS からの復帰時には、CarryFlag でエラーの有無が示されます。Carry=1 の場合には異常終了(エラー発生)を、Carry=0 の場合には正常終了(エラーなし)を示します。また異常終了時には DL レジスタ内にエラーの意味を示すエラーコードが格納されています。

レジスタインターフェイスでは、AX,DX,CarryFlag を除くすべてのレジスタの値は保存されます。

◆ GPBIOS 機能一覧 RCB インターフェイス

名称	リクエストコード	内 容
LCL	0	GPIB 上の機器をローカルモードに戻す。
CLR	1	リスナを指定し、SDC コマンドを送る。または、GPIB 上の全機器に DCL コマンドを送る。
TRG	2	リスナを指定し、GET コマンドを送る。
WTB	3	ATN を True にし、DataByte を GPIB 上に出力する。出力後、ATN を False にする。
WRT	4	リスナを指定し、データを送信する
RED	5	トーカを指定し、からのデータを受信する。
TFI	6	トーカを指定し、トーカからのデータを受信する。 デリミタチェックはしない。
TFO	7	リスナを指定し、データを送信する。
SRQ	8	SRQ 検出用のフラグエリアの登録

レジスタインターフェイス

名称	リクエストコード	内 容
INIT	0	GPBIOS のイニシャライズ処理を行なう
CLI	1	IFC ラインを 1ms の期間 True にする
REN	2	REN ラインを True にする
LLO	3	GPIB 上の機器に LLO コマンドを送信する
RDS	4	指定した機器にシリアルポールを行ない、ステータスバイトを読み込む
TMO	5	バスタイムアウトパラメータをセットする
TDL	6	トーカモード時のデリミタをセットする
LDL	7	リスナモード時のデリミタをセットする
SRQ	8	SRQ 割り込みを許可する
WAIT	10	指定時間動作を停止する
WSRQ	11	指定時間 SRQ を待つ
ADRS	13	REX-5052 の機器アドレスを得る

* レジスタインターフェイスのリクエストコード 9,12 は REX-5052 の場合ありません。リクエストをしても Carry=0 で無動作で帰ります。

◆ エラーコード一覧表

GPBIOS は、エラー発生時、CarryFlag を 1 にし、DL 内に下記のエラーコードをセットして、動作を終了し、呼び出し元に復帰します。

エラーコード	内 容
2	リクエストコードのエラー
53	GPIB バスタイムアウトエラー
60	デバイスが使用可能な状態にない
61	バッファオーバーフロー
90	バウンダリエラー

(注意)10 進数で表記してあります。

(4-2-2) GPBIOS のロード

GPBIOS は下記のロードオプションを備えています。

オプション	説明
/I x x	割り込み番号の指定。 REX-5052 をイネーブルした時の割り込み番号(必ず同一の番号を指定してください。)
/U	GPBIOS のアンロード GPBIOS を使用しなくなった場合にメモリ常駐を解除します。
/H	GPBIOS のオプションを表示します。

◆ロード手順

1. まず、REX-5052PC カードをイネーブルします。
2. DOS 起動後 GPBIOS.COM を起動します。

```
C:¥>GPBIOS /I5
```

以上により GPBIOS がメモリに常駐し INT F2H,INT F3H のベクタが設定されます。

(4-2-3) 各コマンドの呼び出し**◆ RCB インターフェイスの注意事項**

RCB インターフェイスでは、(3-2-1)に示すような RCB と呼ばれるメモリエリアを使用します。RCB 内にセットする値は、すべてバイナリ値ですが、トーカアドレス、リスナアドレスを RCB 内にセットする場合には下記の様式に従う必要があります。

◆ トーカアドレスの設定

トーカを指定する必要があるコマンドを使用する場合にセットします。
 REX-5052 のマイトーカアドレスは設定する必要はありません。
 トーカ指定が不要なコマンドの場合にはダミーとなります。
 トーカアドレス値は、0～1E(h)の間の値をバイナリでセットします。

◆ リスナアドレスの設定

リスナを指定する必要があるコマンドを使用する場合にセットします。
 REX-5052 のマイリスナアドレスは設定する必要はありません。
 リスナ指定が不要な場合には、必ず LAG エリアの先頭(RCB の先頭から+2番地)エリアに"00"をセットしておいてください。
 リスナアドレス値は、0～1E(h)の間の値をバイナリでセットしますが、その際必ず MSB(bit7)を 1 にしてください。
 (例)

リスナアドレス	RCB にセットする値
0	80(h)
1	81(h)
1c(h)	8C(h)

最後のリスナアドレスの次の LAG エリアには必ず"00"をセットしておく必要があります。残りの LAG エリアはダミーとなります。

例えば、リスナアドレスが2ヶの場合、RCB の LAG エリア内には下記の様にセットする必要があります。

オフセット	内 容
+3	リスナアドレス 1
+4	リスナアドレス 2
+5	"00"
・	以降ダミー
・	
+17	

LCL (go to local)

機能 リスナに指定した機器に GTL(go to local)命令を与え、ローカル状態に戻します。リスナアドレスの指定がない場合には REN ラインを False にします。

RCB 構成

offset	内 容
+0	“00”リクエスト番号=“00”
+1	“00”ダミー
+2	リスナアドレス 1
+3	リスナアドレス 2
+4	リスナアドレス 3
⋮	⋮
⋮	⋮
⋮	⋮
+13	リスナアドレス 12
+14	リスナアドレス 13
+15	リスナアドレス 14
+16	リスナアドレス 15

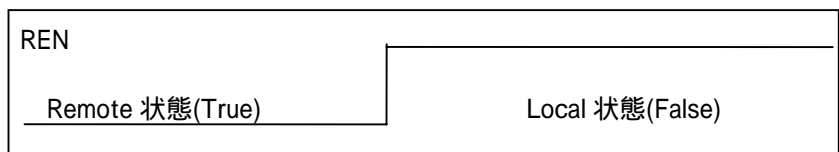
リスナアドレスの指定がない場合は、オフセット+2 のリスナアドレス 1 のエリアに“00”をセットします。

リターンコード CF=0 ➤正常終了
CF=1 ➤異常終了

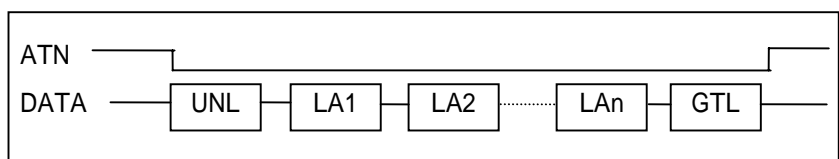
エラーコード DL=53 ➤バスタイムアウトエラー

動作

実行例-1) アドレス指定のない場合、REN ラインを False にします。



実行例-2) アドレス指定がある場合、ATN を True にし、UNL,LAG,GTL を送出し、ATN を False に戻します。



CLR (device clear)

機能 リスナに指定した機器に SDC(Selected Device Clear)命令を与えます。リスナ指定が無い場合は、全機器に対するコマンド DCL(Device Clear)命令を発行します。

RCB 構成

offset	内 容
+0	“00”リクエスト番号=“01”
+1	“00”ダミー
+2	リスナアドレス 1
.	.
.	.
.	.
+15	リスナアドレス 14
+16	リスナアドレス 15

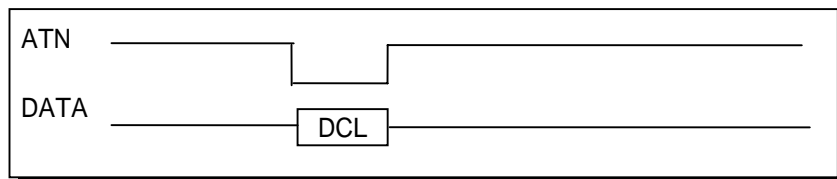
リスナアドレスの指定がない場合は、オフセット+2 のリスナアドレス 1 のエリアに”00”をセットします。

リターンコード CF=0 ➤ 正常終了
 CF=1 ➤ 異常終了

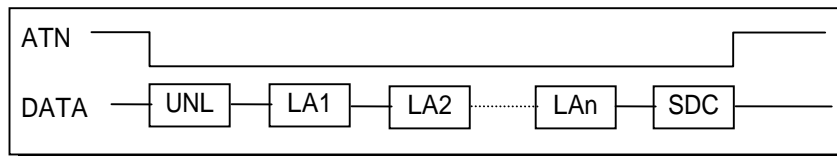
エラーコード DL=53 ➤ バスタイムアウトエラー

動作

実行例-1) アドレス指定のない場合、ATN を True にし DCL 命令を送出した後、ATN を False に戻します。



実行例-2) アドレス指定がある場合、ATN を True にし、UNL,LAG,SDC を送出し、ATN を False に戻します。



TRG (device trigger)

機能 リスナに指定した機器に GET(Group Execute Trigger)命令を与えます。

RCB 構成

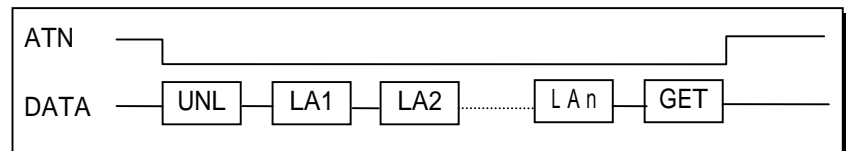
offset	内 容
+0	“00”リクエスト番号=“02”
+1	“00”ダミー
+2	リスナアドレス 1
.	.
.	.
.	.
+15	リスナアドレス 14
+16	リスナアドレス 15

リターンコード CF=0 ➤正常終了
CF=1 ➤異常終了

エラーコード DL=53 ➤バスタイムアウトエラー

動作 ATN を True にし、UNL,LAG,GET コマンドを送出し、ATN を False に戻します。

実行例



WTB (write byte)

機能 ATN ラインを True にし、与えられたデータ列を送信します。送信後 ATN ラインを False に戻します。特殊なコマンドや機器アドレスを出力する場合に使用します。

RCB 構成

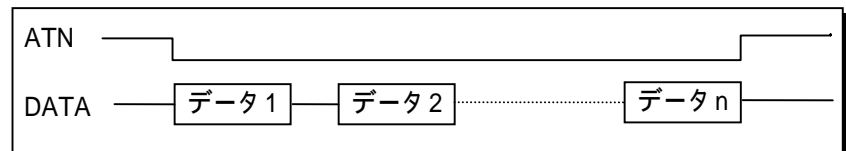
offset	内 容
+0	"00"リクエスト番号="03"
+1	"00"ダミー
+2	リスナアドレス 1
.	.
.	.
.	.
+15	リスナアドレス 14
+16	リスナアドレス 15
+17	送信バイトカウント
+18	送信 DATA1
.	.
.	.
.	.
+17+n	送信 DATA _n

リターンコード CF=0 ➤正常終了
CF=1 ➤異常終了

エラーコード DL=53 ➤バスタイムアウトエラー

動作 ATN を True にして、与えられたデータ列を順に送り出した後、ATN を False に戻します。

実行例



WRT (write)

機能 リスナアドレスを指定して、データ列を送り出します。また、データ列の最後に、トーカーモードデリミタによって指定されたデリミタを出力します。

RCB 構成

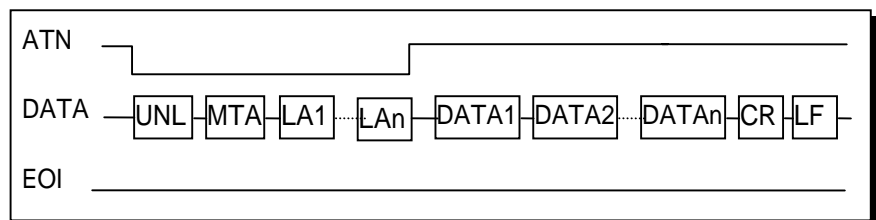
offset	内 容
+0	“00”リクエスト番号=”04”
+1	“00”ダミー
+2	リスナアドレス 1
.	.
.	.
.	.
+15	リスナアドレス 14
+16	リスナアドレス 15
+17	送信バイトカウント(L)
+18	送信バイトカウント(H)
+19	送信バッファオフセット(L)
+20	送信バッファオフセット(H)
+21	送信バッファセグメント(L)
+22	送信バッファセグメント(H)

リターンコード CF=0 ➤正常終了
CF=1 ➤異常終了

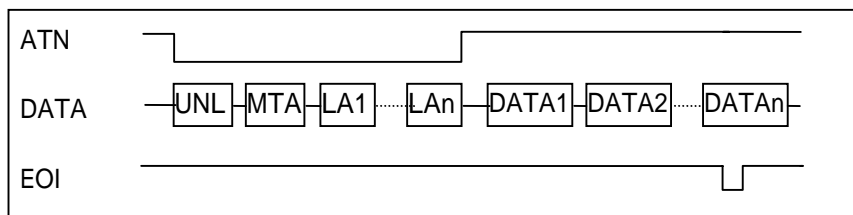
エラーコード DL=53 ➤バスタイムアウトエラー

動作 ATN を True にして、UNL,MTA,LAG を送出します。その後、ATNを False に戻し、与えられたデータ列を送出します。また、データ列の最後に、トーカーモードデリミタによって指定されたデリミタを出力します。トーカーモードデリミタについては、TDL を参照してください。

実行例-1) トーカーモードデリミタ=0(CR+LF)の場合



実行例-2) トーカモードデリミタ=80(h)(EOI のみ)の場合



RED (read)

機能 トーカを指定し、トーカが送出するデータ列を受信します。他にリスナがある場合は、複数のリスナ指定ができます。データの受信は、リスナモードデリミタの検出、あるいは EOI の検出によって終了します。

RCB 構成

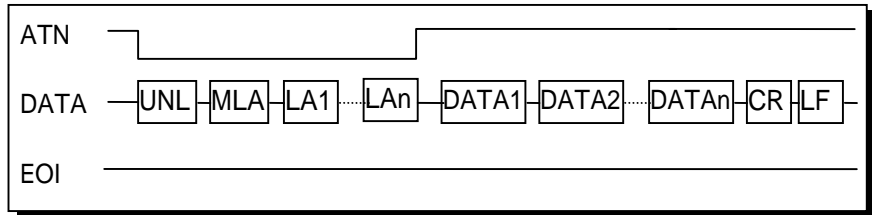
offset	内 容
+0	“00”リクエスト番号=“05”
+1	トーカアドレス
+2	リスナアドレス 1
.	.
.	.
.	.
+15	リスナアドレス 14
+16	リスナアドレス 15
+17	受信バイトカウンタ(L)
+18	受信バイトカウンタ(H)
+19	受信バッファオフセット(L)
+20	受信バッファオフセット(H)
+21	受信バッファセグメント(L)
+22	受信バッファセグメント(H)

リターンコード CF=0 ➤ 正常終了
CF=1 ➤ 異常終了

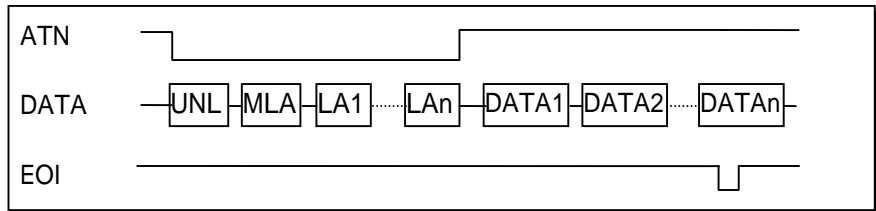
エラーコード DL=53 ➤ バスタイムアウトエラー
DL=61 ➤ バッファオーバーフロー
受信バッファが Full(受信バイトカウンタを超えた場合)でもデリミタまたは EOI を検出しなかった場合にセットされます。

動作 ATNを True にして、UNL,MLA,TA,LAG を送じます。その後、ATN を False に戻し、トーカーより送られてくるデータ列を受信します。また、データ受信動作はリスナモードデリミタの検出または、EOI の検出で終了します。リスナモードデリミタについては、LDL を参照してください。

実行例-1) リスナモードデリミタ=0a(h)(LF)の場合



実行例-2) EOI を検出した場合



ただし、RED 動作は、リスナモードデリミタがどのように設定されていようと EOI の検出により受信動作を終了します。

TFI (transfer in)

機能 トーカを指定し、トーカが送出するデータ列を受信します。他にリスナがある場合は、複数のリスナ指定ができます。データの受信は、EOI の検出によって終了します。

RCB 構成

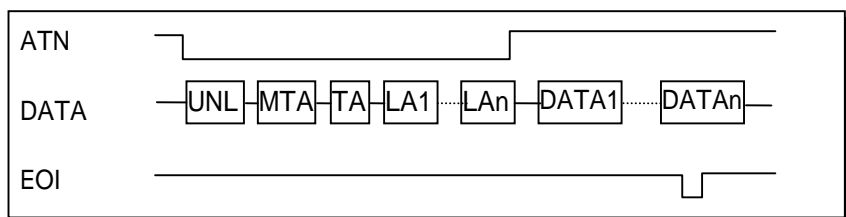
offset	内 容
+0	“00”リクエスト番号=”06”
+1	トーカアドレス
+2	リスナアドレス 1
.	.
.	.
.	.
+15	リスナアドレス 14
+16	リスナアドレス 15
+17	受信バイトカウンタ(L)
+18	受信バイトカウンタ(H)
+19	受信バッファオフセット(L)
+20	受信バッファオフセット(H)
+21	受信バッファセグメント(L)
+22	受信バッファセグメント(H)

リターンコード CF=0 ➤正常終了
 CF=1 ➤異常終了

エラーコード DL=53 ➤バスタイムアウトエラー
 DL=61 ➤バッファオーバーフロー
 受信バッファが Full(受信バイトカウンタを超えた場合)でも EOI を検出しなかった場合にセットされます。

動作 ATN を True にして、UNL,MLA,TA,LAG を送出します。その後、ATN を False に戻し、トーカより送られてくるデータ列を受信します。また、データ受信動作は、EOI の検出で終了します。バイナリモードの転送ですのでデリミタのチェックは行いません。

実行例



TFO (transfer out)

機能 リスナアドレスを指定して、データ列を送り出します。また、データ列の最後に、トークモードデリミタによって指定されたデリミタを出力します。

RCB 構成

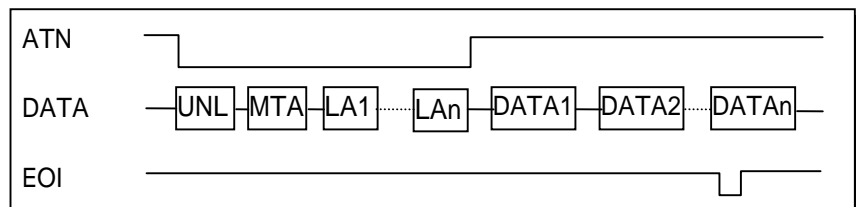
offset	内 容
+0	"00"リクエスト番号="07"
+1	"00"ダミー
+2	リスナアドレス 1
.	.
.	.
.	.
+15	リスナアドレス 14
+16	リスナアドレス 15
+17	送信バイトカウント(L)
+18	送信バイトカウント(H)
+19	送信バッファオフセット(L)
+20	送信バッファオフセット(H)
+21	送信バッファセグメント(L)
+22	送信バッファセグメント(H)

リターンコード CF=0 ➤正常終了
CF=1 ➤異常終了

エラーコード DL=53 ➤バスタイムアウトエラー

動作 ATN を True にして、UNL,MTA,LAG を送出します。その後、ATNを False に戻し、与えられたデータ列を送出します。また、データ列の最後バイトの送信と同時に EOI を出力します。

実行例



SROF (set SRQ flag address)

機能 SRQ 割り込みを検出したことを通知する為の2バイトのフラグエリアのアドレスを登録します。

RCB 構成

offset	内 容
+0	"00"リクエスト番号="08"
+1	"00"ダミー
+2	"00"ダミー
+3	"03"(03 固定)
+4	フラグエリアオフセット(L)
+5	フラグエリアオフセット(H)
+6	フラグエリアオフセット(L)
+7	フラグエリアオフセット(H)
+8	"00"
.	.
.	.
+17	"00"

リターンコード CF=0 ➤常に正常終了

動作 GPIB 機器からの SRQ を検出すると登録された FlagArea の内容(2Byte 値)に"1"を加えます。

◆ レジスタインターフェイス

INIT (initialize)

機能 GPBIOS 内部のパラメータエリアの初期設定、REX-5052 上の GPIB コントローラチップのイニシャライズを行ないます。GPBIOS を使用する場合には、必ず最初にこのコマンドを実行させる必要があります。

入力パラメータ

パラメータ	内 容
AH:00	リクエスト番号"00"
AL: x x	REX-5052 GPIB マイアドレス
DX: x x x x	REX-5052 PC カードの I/O アドレス

リターンコード CF=0 ➤正常終了
CF=1 ➤異常終了

エラーコード DL=60 ➤DX で指定された I/O アドレスに REX-5052 カードが実装されていないことを示します。

動作 DX で指定された I/O アドレスにアクセスを行ない、REX-5052 カードが実装されているかどうかをチェックします。チェックの結果、正常であれば、GPBIOS 内部のパラメータ類を初期設定します。各デフォルト値を下記に示します。

- マイアドレス AL で指定された値
- バスタイムアウト 10 秒
- トーカモードデリミタ CR+LF EOI なし
- リスナモードデリミタ LF または EOI

CLI (clear interface)

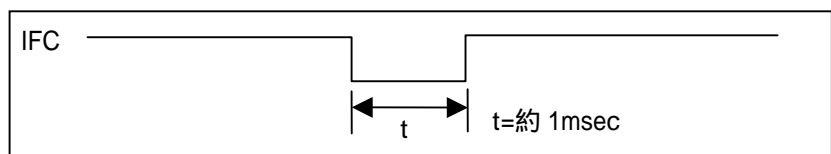
機能 IFC ラインを一定期間(約 1mSec)True にします。

入力パラメータ

パラメータ	内 容
AH:01	リクエスト番号"01"

リターンコード CF=0 ➤常に正常終了

動作

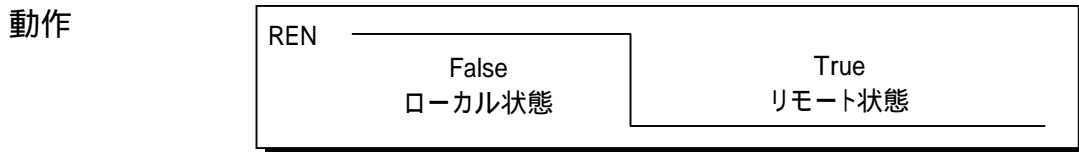


REN (remote enable)

機能 REN ラインを True("L")にします。

入力パラメータ	パラメータ	内 容
	AH:02	リクエスト番号"02"

リターンコード CF=0 ➤常に正常終了



LLO (local lock out)

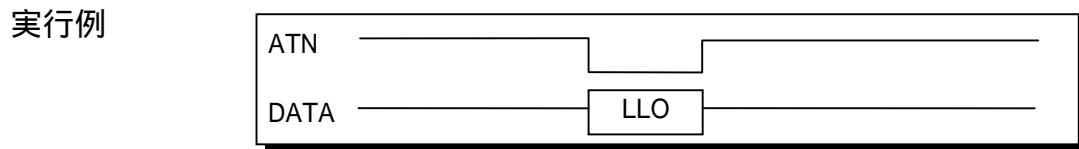
機能 GPIB 上の全機器に LLO(LocalLockOut)コマンドを送信します。

入力パラメータ	パラメータ	内 容
	AH:03	リクエスト番号"03"

リターンコード CF=0 ➤正常終了
CF=0 ➤異常終了

エラーコード DL=53 ➤バスタイムアウトエラー

動作 ATN を True にし、LLO コマンドを送出した後、ATN を False にします。



RDS (read status byte)

機能 GPIB 上の機器に対して、シリアルポールを行ない、指定したトーカーが出力するステータスバイトを受信します。

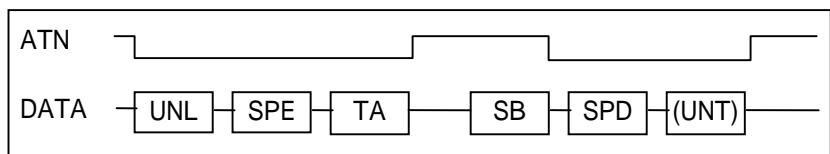
入力パラメータ

パラメータ	内 容
AH:04	リクエスト番号"04"
DL: トーカーアドレス	ステータスバイトを送出するように指定する機器のアドレス。bit7 は次の意味を持ちます。 bit7=0 SPD に続いて UNT を送る。 bit7=1 SPD に続いて UNT を送らない。

リターンコード CF=0 ➤正常終了 この時、DL 内には受信したステータスバイトが入っています。
CF=0 ➤異常終了

エラーコード DL=53 ➤バスタイムアウトエラー

動作 ATN を True にした後、UNL, SPE, TA を送信します。その後、ATN を False に戻し、指定したトーカーから送り出されてくるステータスバイトを受信します。受信後の動作は入力パラメータの DL の値によって異なります。
DL の bit7 が"0"の場合は ATN を True にし、SPD, UNT を送出した後、ATN を False に戻します。
DL の bit7 が"1"の場合は ATN を True にし、SPD だけを送出した後、ATN を False に戻します。

実行例

TMO (time out parameter)

機能 バスタイムアウトの監視のパラメータの値をセットします。単位は秒です。

入力パラメータ

パラメータ	内 容
AH:05	リクエスト番号"05"
DL:01 ~ FF	タイムアウトパラメータ(1 ~ 255 秒)。 0 の場合は無限。

リターンコード CF=0 ➤常に正常終了

動作 GPBIOS 内部のタイムアウトパラメータエリアに DL の値をセットします。GPIB では、データバス上の受け渡しは、すべてハンドシェイクと呼ばれる手順で管理されています。コントローラ (REX-5052) と、リスナ、トーカーとの間で、データ(コマンドやアドレスも含む)の受け渡しを行なう際に、規定時間内にハンドシェイクが終了しない(リスナやトーカーが応答しない)ことをバスタイムアウトエラーと呼びます。この規定時間の長さを決定するのが、本コマンドでセットするタイムアウトパラメータです。

TDL (time mode delimiter)

機能 バスタイムアウトの監視のパラメータの値をセットします。単位は秒です。

入力パラメータ

パラメータ	内 容
AH:06	リクエスト番号"06"
DL:パラメータ	

リターンコード CF=0 ➤常に正常終了

パラメータの意味 パラメータの意味を下記に示します。このパラメータはWRTコマンドの実行時に参照されます。

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
EOI	Code6	Code5	Code4	Code3	Code2	Code1	Code0

➤EOIbit 1:EOIを出力する
 0:EOIを出力しない
 ➤コード bit・デリミタとして使用するコード

EOI bit=0, Codebit=0 の場合(DL=0)はデリミタとして CR+LF が送信されます。

EOI bit=1, Codebit=0 の場合(DL=80(h))は、最後のデータ出力と同時に EOI を True にします。CR+LF は付加されません。Codebit が 0 以外の場合には、Codebit で指定するコードがデリミタとして送り出されます。その時、同時に EOI が出力されるかどうかは、EOIbit の指定に従います。

LDL (listener mode delimiter)

機能 リスナモード(受信時)のデリミタを指定するためのパラメータをセットします。

入力パラメータ

パラメータ	内 容
AH:07	リクエスト番号"07"
DL:パラメータ	

リターンコード CF=0 ➤常に正常終了

パラメータの意味 GPBIOS は、RED コマンドの実行時に、このパラメータを参照し、このパラメータと同一の値を持つキャラクタを受信文字列中に発見すると、読みこみ動作を終了します。この他、EOI の検出によっても読み込み動作は終了します。初期値は 0a(h)(LF)が指定されています。

SRQ (SRQ interrupt enable)

機能 REX-5052 上の LSI に対し、SRQ 受信時、メイン CPU に対する割り込み要求発生の許可、不許可を設定します。

入力パラメータ

パラメータ	内 容
AH:08	リクエスト番号"08"
DL:パラメータ	00:割り込み不許可 01:割り込み許可

リターンコード CF=0 ➤常に正常終了
SRQ 割り込みを使用する場合には、必ず本コマンドにより、SRQ 割り込み要求発生を許可しておく必要があります。

WAIT (wait)

機能 指定した時間、実行を停止します。単位は秒です。

入力パラメータ

パラメータ	内 容
AH:0A(h)	リクエスト番号"0A"
DX:1 ~ FFFF(h)	停止時間(1 ~ 65535 秒)

動作

指定された時間、GPBIOS 内でダミーループを実行します。時間の計測は、ソフトタイマにより行ないます。また、基本的に互換性を保つ関数ですので、正確なタイマを必要とする場合には、他の方法をご検討ください。

WSRQ (wait service request)

機能

指定された時間だけ SRQ を監視します。時間の単位は 0.1 秒です。WSRQ を実行する前に、SRQ により SRQ 割り込みを許可する必要があります。また、WSRQ の実行により SRQ ラインは変化しません。

入力パラメータ

パラメータ	内 容
AH:0B(h)	リクエスト番号"0B"
DX:1 ~ FFFF(h)	監視時間(0.1 ~ 6553.5 秒)

リターンコード

DX=0000(0) ➤SRQ があった。
 FFFF(-1) ➤SRQ がなかった。

(4-3) MS-DOS 用 C 言語ライブラリ解説

(4-3-1) 関数仕様

本ライブラリは、アプリケーションからの関数コールにより GPBIOS を呼び出します。そのため必ず GPBIOS をロードしてからアプリケーションを実行してください。

◆ 関数仕様の記述について

本ソフトウェアを動作させるための個々のコマンドについて解説を行います。汎例を下記に示します。

gp_xxx (コマンド名)	機 能
書式	関数の記述
機能	そのコマンドの機能
引数	関数の引数
関連	実行時に関連のあるパラメータ
実行例および動作	そのコマンドの実行例と GPIB 各信号線の動作を示します。

留意点

- ライブラリを使用する場合は、必ず GPLIB.H をインクルードしてください。
- すべての関数は INT 型の戻り値を返します。
- 戻り値は、"0" の場合は正常終了です。それ以外はエラーコードです。機器アドレスの指定は文字列で行ないます(各コマンドの解説では書式の項目で "char *adrs" で示されています。)。このとき、トークン指定が必要なコマンドでは、文字列の先頭の機器アドレスがトークンアドレスとなります。

(例) リスナアドレス 1,3,4,8 の場合 adrs="1,3,4,8"
 全機器に対する場合 adrs="" (ヌル文字列)

◆ C 言語 GPIB 用関数一覧

関数	機能
gp_init(port, id)	GPBIO を初期化する。
gp_cli()	GPIB の IFC ラインを一定期間 true にする。
gp_ren()	GPIB の REN ラインを true にする。
gp_clr(adrs)	デバイスクリアコマンドを送出する。
gp_trg(adrs)	デバイストリガコマンドを送出する。
gp_wrt(adrs, buf)	GPIB 上にデータを出力する。
gp_red(adrs, buf)	GPIB 上のデータを読み込む
gp_tfrin(adrs, bytc, buf)	バッファメモリ上に GPIB 上のデータを読み込む。 バッファ上のデータを GPIB 上に送り出す。
gp_tfi(adrs, bytc, ofs, seg)	
gp_tfrout(adrs, bytc, buf)	
gp_tfrou(adrs, bytc, buf)	
gp_tfo(adrs, bytc, ofs, seg)	
gp_lcl(adrs)	指定された機器をローカルモードにする。
gp_llo()	GPIB 上の全機器のローカルスイッチを無効にする。
gp_wtb(buf)	ATN ラインを true にしてデータを送出する。
gp_srq(sw)	SRQ によるハードウェア割り込みを制御する。
gp_fnsrq(&flag)	SRQ 割り込み時に使用するフラグ変数のアドレスを登録
gp_rds(adrs)	シリアルポールを実行し、ステータスバイトを読み込む。
gp_wsrq(flag, time)	指定された時間だけ SRQ がくるのを待つ。
gp_delm(mode, delm)	red, wrt コマンドのデリミタを指定する。
gp_reds1(adrs)	シリアルポールを実行し、ステータスバイトを読み込む
gp_tmout(time)	データ送受信時のバス・タイムアウトを指定する。
gp_myadr()	カードにセットされたアドレスを読み取る
ongperr(func)	エラー発生時の処理関数を登録する
gp_wait(time)	指定された時間だけプログラムの実行を停止する。
gp_csinfo(pIObase, pIrqNo)	リソース情報の取得

gp_init

GP-BIOS を初期化する

書式 int gp_init (int port, int id);

機能 REX-5052 上の GPIB コントローラチップを初期化します。

引数 int port; ➤ カードに割り当てられた I/O ベースアドレス
 int id; ➤ カードの GPIB 機器アドレス

関連 なし

解説 REX-5052 カード上の GPIB コントローラチップにソフトウェアリセットコマンドを送り、GPIB コントローラチップを初期化し、GPIB 機器アドレスをセットし、本ライブラリで使用するパラメータを初期化します。
また GPBIOS のチェックを行いません。GPBIOS が起動されていない場合、戻り値として-1 を返し、errno に 91 をセットします。ただし、gp_init()では、後述の_gp_err の機能は利用できないため、if 文等の条件文を使用してください。

例

```
If( _gp_init(0x0120,0x00) ==-1 ){  
    fprintf( stderr, "GPBIOS error!!%n" );  
    exit( 1 );  
}
```

*すでにREXシリーズのGPIBカードをご使用になっているユーザの方は以下のことにご注意ください。
他のカードではハードウェアに GPIB 機器アドレスの SW がありますが、REX-5052 には SW がありません。そのため本コマンドで引数として GPIB 機器アドレスを渡しています。

gp_cli**IFC ラインを一定期間 TRUE にする**

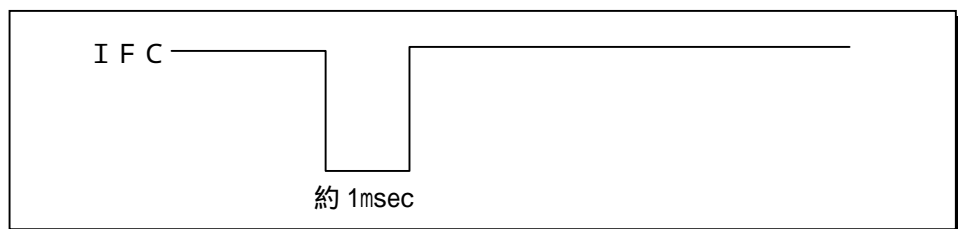
書式 int gp_cli (void);

機能 I F C ラインを約 1ms の間”True”にします。

引数 なし

関連 なし

解説 gp_cli();



REX-5052 カード上の LSI 及び、GPIB に接続されている全ての機器の初期化を行うために、プログラムの先頭部で必ず一度は IFC コマンドの実行が必要です。

gp_ren

REN ラインを TRUE にする

書式 int gp_ren (void);

機能 REN ラインを"True"にします。

引数 なし

関連 なし

解説 gp_ren();

The diagram shows a horizontal line labeled 'REN' on the left. A vertical line descends from this point, then a horizontal line extends to the right, ending in an arrowhead pointing back to the vertical line. Above this horizontal line is the text 'REN コマンドの実行'.

LCL コマンド (LCL コマンドの項 実行例 1 を参照) が実行されるか、PC カードがスロットから抜かれるか、または PC がリセットされるまでずっと True のままです。

GPIB インターフェイスを持つ計測機器や装置は、REN ラインが True になるとリモート可能モードとなり、リモートモードを表示する LED などが点灯します。

REN ラインが False のままですと、GPIB 機器は正しく動作しませんので、プログラム先頭で必ず一度は REN コマンドの実行が必要です。

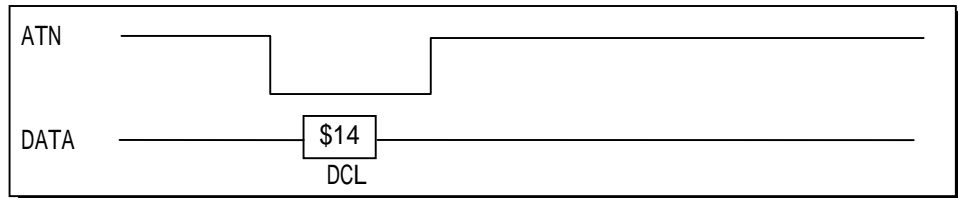
gp_clr**デバイスクリアコマンドを送出する**書式 `int gp_clr (char *adrs);`

機能 デバイスクリアコマンド、またはセレクトッドデバイスクリアコマンド(SDC)を GPIB 上に送り出し、相手側機器をリセットします。

引数 `char *adrs;`

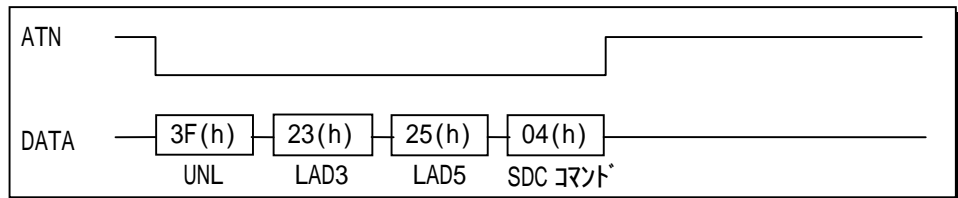
関連 タイムアウト

解説 実行例 1. 全機器に対する場合

`gp_clr("");`

GPIB 上の全機器に対してクリアコマンドを送り、全機器をリセットします。

実行例 2. アドレス 3,5 の機器に対して、クリアコマンドを送る場合

`gp_clr("3,5");`

相手側機器の DC(Device Clear)機能が DC0 の場合は、このコマンドは無効です。また DC2 の場合は、実行例 2 の SDC コマンドは無効となりますので、実行例 1 を御使用ください。

gp_trg**デバイストリガコマンドを送出する**

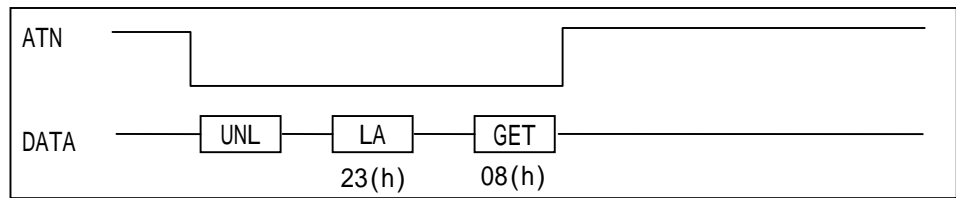
書式 int gp_trg (char *adrs);

機能 リスナに指定された機器に対して GET(トリガ)命令を送信します。

引数 char *adrs;

関連 タイムアウト

解説 gp_trg("3");
 アドレス 3 の機器に対して GET 命令を送信します。



gp_wrt

GPIB上にデータを送信する

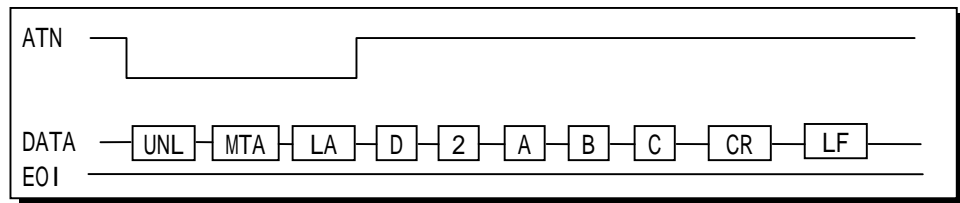
書式 `int gp_wrt (char *adrs, char *buf);`

機能 リスナアドレスによって指定された機器へデータを送信します。

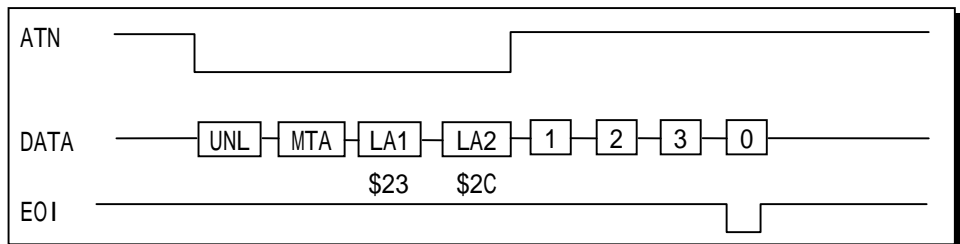
引数 `char *adrs;`
`char *buf;`

関連 タイムアウト、トーカモードデリミタ

解説 実行例 1. シングルリスナアドレスの場合
(トーカモードデリミタ=0)
`gp_wrt("3", "D2ABC");`
アドレス 3 の機器に "D2ABC" という文字列を送信します。



実行例 2. マルチリスナアドレスの場合
(トーカモードデリミタ=0x80)
`gp_wrt("3,12", "1230");`
アドレス 3,12 の機器に文字列を送信します。



gp_red

GPIB上のデータを読み込む

書式 `int gp_red (char *adrs, char *buf);`

機能 トークアドレスで指定した機器よりデータを受信し、バッファ領域内に格納します。同時にリスナアドレスを指定すると、その機器にもデータが送られます。

引数 `char *adrs;`

`char *buf;`

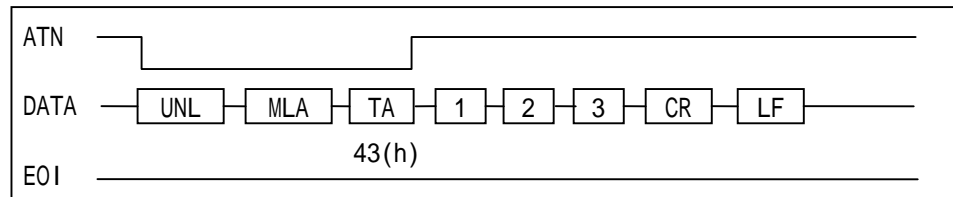
注) バッファサイズは受信するバイト数より必ず1バイト以上多く取ってください。

関連 タイムアウト, リスナモードデリミタ

解説 実行例 1. 相手側機器の送信時デリミタが LF の場合

```
char buf[30];
gp_red( "3", buf );
```

アドレス3の機器よりデータを受信し、文字配列 buf 内に格納します。

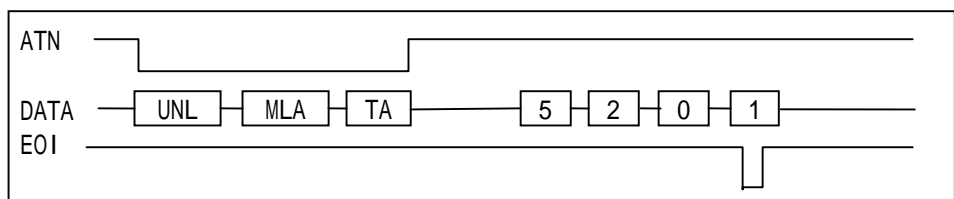


HP社、横河電機、アドバンテスト等、ほとんどのメーカーが送信時デリミタとして CR, LF を使用していますので、リスナモードデリミタとしては 0x0a(LF) が一般的です。

実行例 2. 相手側機器の送信時デリミタが EOI の場合

```
char a[10];
gp_red( "3", a );
```

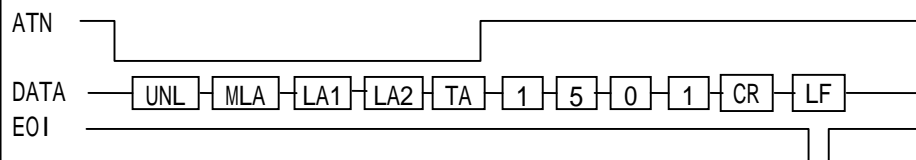
アドレス3の機器よりデータを受信し、文字配列 a 内に格納します。



実行例 3. リスナアドレス付の場合

```
char c[10];  
gp_red( "3,10,12", c );
```

アドレス 3 の機器よりデータを受信し、文字配列 c 内に格納します。
同時にアドレス 10,12 の機器にもデータが送られます。



注) red コマンドは、相手側機器から出力される EOI を検出するとその時点で読み込み動作を終了します。

gp_tfrin バッファメモリ上に GPIB 上のデータを読み込む

書式 `int gp_tfrin (char *adrs, unsigned int bytc, char *buf);`

機能 指定したトーカアドレスの機器より指定バイト数分のデータをバッファ領域内に直接読み込んで格納します。読み込み動作は、指定されたバイト数分で終了するか、EOI を検出した時点で終了します。

引数 `char *adrs;`
 `unsigned int bytc; >受信バイトカウント`
 `char *buf; >受信用配列領域`

関連 タイムアウト

解説 ● 画像処理装置や FFT アナライザなどでは、一度に 1 ~ 数 KB のデータを転送する機能を持っていますので、この tfrin を使用するとデータを 1 度に受信することができます。

 ● バッファ領域はデータセグメント(DS)内にあるものとします。もし、他のセグメント内に単独でバッファ領域を取る場合は、gp_tfi を使用してください。

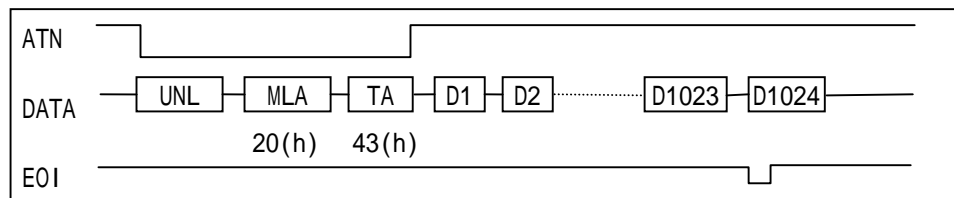
 ● 受信バイト数がバッファ変数の長さよりも大きい場合は、バッファ変数分のデータだけ受け取ります。

 ● 受信バイト数の指定は、0 から 65,535 の数値定数または、整数型変数で行ってください。

実行例

```
char buf[1025];
gp_tfrin( "3", 1024, buf );
```

トーカアドレス 3 の機器から 1024 バイトのデータをバッファ変数内に読み込みます。



gp_tfi**バッファメモリ上にGPIB上のデータを読み込む**

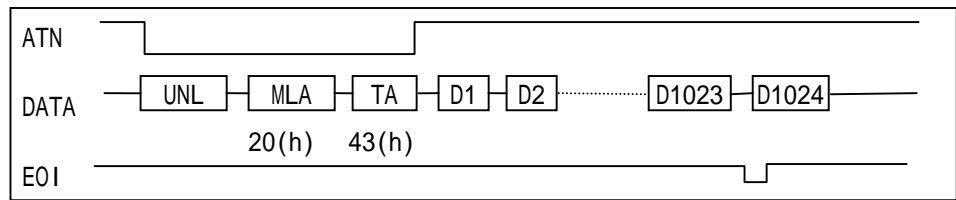
書式 int gp_tfi (char *adrs, unsigned int bytc,
 unsigned int ofs, unsigned int seg);

機能 指定したトーカアドレスの機器より指定バイト数分のデータをバッファ領域内に直接読み込んで格納します。読み込み動作は、指定されたバイト数分で終了するかまたは、EOI を検出した時点で終了します。

引数 char *adrs;
 unsigned int bytc; ➤受信バイトカウント
 unsigned int ofs; ➤バッファ領域へのオフセット
 unsigned int seg; ➤バッファ領域へのセグメント

関連 タイムアウト

解説 gp_tfi("3", 1024, buffofs, buffseg);
 トーカアドレス 3 の機器から 1024 バイトのデータをバッファ領域内に読み込みます。



受信バイト数の指定は、0 から 65,535 の数値定数または、整数型変数で行ってください。

gp_tfROUT, gp_tFRou バッファ上のデータを GPIB 上に送り出す

書式 int gp_tfROUT (char *adrs, unsigned int bytc, char *buf);
 int gp_tFRou (char *adrs, unsigned int bytc, char *buf);

機能 指定したリスナアドレスの機器へ指定のバイト数分のデータをバッファ変数内より直接転送します。送信（転送）動作は、指定のバイト数分を送り終わると終了します。

引数 char *adrs;
 unsigned int bytc; >送信バイトカウント
 char *buf; >送信用配列領域

関連 タイムアウト

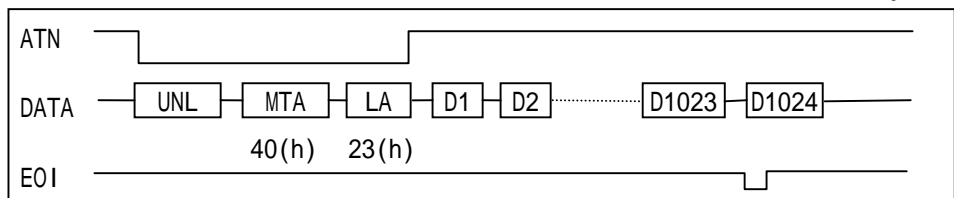
解説

- 画像処理装置や FFT アナライザなどへ一度に数 KB のデータを送り込む場合にこの tfROUT コマンドを使用します。
- 送信時デリミタとして、EOI が送られます。
- バッファ領域はデータセグメント(DS)内にあるものとします。もし、他のセグメント内に単独でバッファ領域を取る場合は、gp_tFO を使用してください。
- 送信バイト数の指定は、0 から 65,535 の数値定数または、整数型変数で行ってください。
- REX-5052 では、ソフトウェアにより送信されるため、タイムアウトは"gp_tmout"で設定されている値となります。

実行例

```
char buf[1025];
gp_tfROUT( "3", 1024, buf );
```

リスナアドレス 3 の機器へ 1024 バイトのデータを送信します。



gp_lcl

指定された機器をローカルモードにする

書式 `int gp_lcl (char *adrs);`

機能 指定したリスナアドレスの機器をローカル状態に戻します。リスナアドレスの指定が無い場合は、REN ラインを False にし、GPIB 上の全機器をローカル状態に戻します。

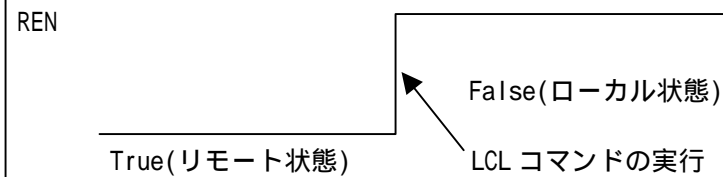
引数 char *adrs;

関連 タイムアウト

解説 実行例 1. リスナアドレスの無い場合

```
gp_lcl( "" );
```

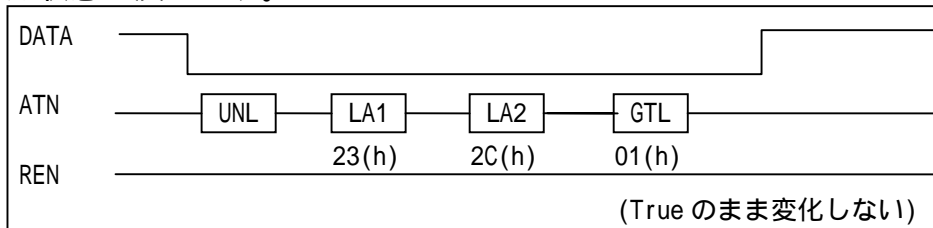
GPIB 上の全機器をローカルモードにします。



実行例 2. リスナアドレスの指定がある場合

```
gp_lcl( "3,12" );
```

リスナアドレス 3,12 の機器に GTL(go to local)命令を送りローカル状態に戻します。



gp_ll0 GPIB 上の全機器のローカルスイッチを無効にする

書式 int gp_ll0 (void);

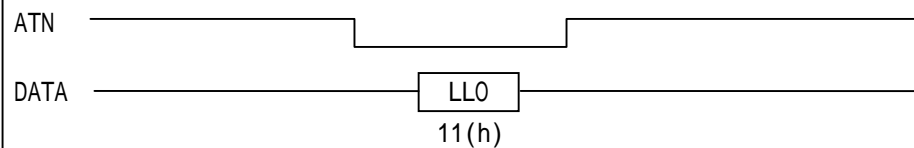
機能 GPIB 上の全機器のローカルスイッチを無効にします。

引数 なし

関連 タイムアウト

解説 実行例

```
gp_ll0( );
```



- ATN ラインを True にし、LL0 命令を送信した後 ATN ラインを False にします。この命令を受信すると機器側ではパネル上の操作スイッチを無効にします。ただし機器のリモート状態もしくはローカル状態には、変化は生じません。
- 機器の LL0 状態を解除する場合は REN ラインを False にします。(LCL コマンドの実行)

gp_wtb

ATN ラインを True にしてデータを送出する

書式 `int gp_wtb (char *buf);`

機能 ATN ラインを True にしてコマンド文字列を送信した後、ATN ラインを False にします。

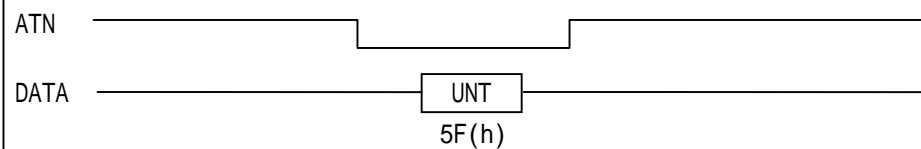
引数 `char *buf;`

関連 タイムアウト

解説 実行例 1.

```
gp_wtb( "%x5f" );
```

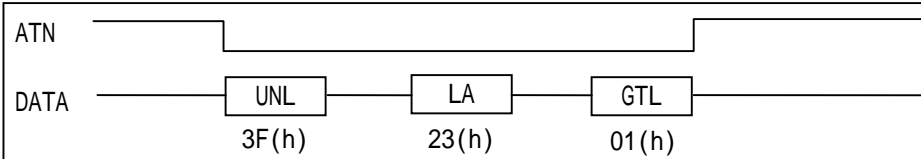
アントーク命令(UNT)をバス上に送り出し、トーカーに設定されている機器のトーカーモードを解除します。



実行例 2.

```
char *buf;
buf = "%x3f%x23%x01";
gp_wtb( buf );
```

LCL3 の実行と同様になります。



gp_srq SRQ によるハードウェア割り込みを制御する

書式 `int gp_srq (int sw);`

機能 SRQ によるハードウェア割り込みの許可、不許可を設定します。

引数 `int sw;`

- `sw = 0` で割り込み不許可。`sw = 1` で許可となります。常に正常終了します。
- 現バージョンでは、SRQ 割り込み時に `gp_fnsrq()` によって指定したフラグに値をセットすることができます。

関連 なし

解説 実行例

```
gp_srq( 1 );
```

SRQ によるハードウェア割り込みを許可します。

gp_fnsrq SRQ 割り込み時に使用するフラグ変数のアドレスを登録

書式 `int gp_fnsrq (unsigned int *flag);`

機能 割り込み時に使用する変数の登録を行ないます。

引数 `unsigned int *flag;`
SRQ 割り込みが発生した場合、現在の `flag` の内容を+1 し、割り込みをクリアします。現在の `flag` の内容が `0xffff` の場合は、`flag` を 1 にします。

関連 なし

解説 実行例

```
gp_fnsrq( &flag );  
flag = 0;  
gp_srq( 1 );  
while( !flag ) /* 割り込み待ち */  
    ;  
    .  
    .  
    .  
  
/* SRQ の処理 */
```

gp_rds	シリアルポールを実行し、ステータスバイトを読み込む
--------	---------------------------

書式 int gp_rds (char *adrs);

機能 GPIB 上の機器に対してシリアルポールを実行し、機器からのステータスバイトを受信します。

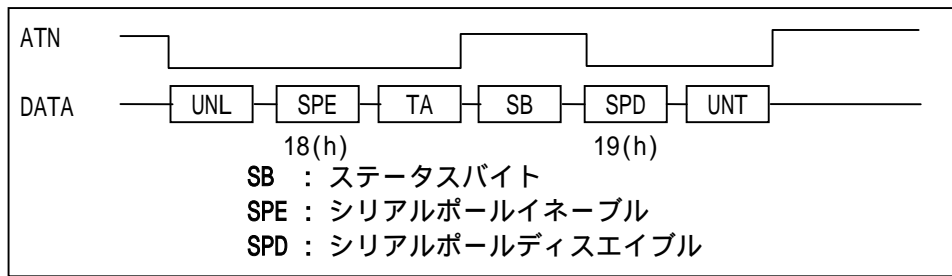
引数 char *adrs;

関連 タイムアウト

解説 実行例

```
int s, gp_rds();
s = gp_rds( "3" );
```

トーカーアドレス 3 の機器に対してシリアルポールを実行し、その機器のステータスバイトを読み込み変数 s に代入する。



SRQ を発信中の機器に対してこのコマンドを実行すると、SRQ ラインが False に復帰します。

gp_wsrq

指定され時間だけ SRQ がくるのを待つ

書式 `int gp_wsrq (int flag, unsigned int time);`

機能 指定したパラメータで決まる時間だけ、SRQ がくるのを待ちます。時間内に SRQ がくれば 0 を、SRQ がなければ-1 を返します。

引数 `int flag;`
`unsigned int time;`

- flag は SRQ 割り込み許可フラグで、1 に設定する。
- time は 100ms です。
- time は整数定数または整数型変数で、0 から 65,535 まで指定できます。
- このコマンドによって SRQ ラインは変化しません。

関連 なし

解説 実行例

```
int s, gp_wsrq();  
s = gp_wsrq( 100 );
```

SRQ がくるまで 10 秒間待ちます。戻り値として 10 秒以内に SRQ があれば 0 を、なければ-1 を、持ちます。

gp_delm

red, wrt コマンドのデリミタを指定する

書式 int gp_delm (char *mode, int delm);

機能 リスナ時またはトーカー時のデリミタを設定します。

引数 char *mode;
 int delm;

- mode は “t”, “l” のどれか一文字とし、次の様な意味を持ちます。
“t” ---- トーカー時の送信デリミタを指定します。
“l” ---- リスナ時の受信デリミタを指定します。
- delm は 0 ~ 255 (0x00 ~ 0xff) の範囲の値で mode により次の様な意味を持ちます。
“t” ---- デリミタコードは bit6 ~ bit0 の 7bit で設定します。
 この時、bit7 を 1 にすると EOI を出力します。
“l” ---- デリミタコードは bit7 ~ bit0 の 8bit で設定します。
 delm=0 とした場合は CR+LF が設定されます。
- 変更されたデリミタは、次にこのコマンドによって変更されるまで有効です。

関連 なし

解説 実行例 1.

```
gp_delm( "l", 10 );
```

リスナモードデリミタとして LF を設定します。

実行例 2.

```
gp_delm( "t", 0x8a );
```

トーカーモードデリミタとして LF を送信し同時に EOI を True にします。

gp_rds1 シリアルポールを実行し、ステータスバイトを読み込む

書式 `int gp_rds1 (char *adrs);`

機能 GPIB 上の機器に対してシリアルポールを実行し、機器からのステータスバイトを受信します。gp_rds との違いは、最後に UNT を出力しない点です。

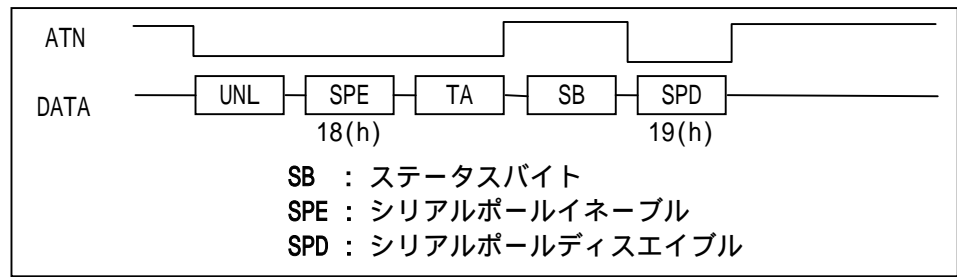
引数 `char *adrs;`

関連 タイムアウト

解説 実行例

```
int s, gp_rds1();
s = gp_rds1( "3" );
```

トークアドレス 3 の機器に対してシリアルポールを実行し、その機器のステータスバイトを読み込み変数 s に代入する。



SRQ を発信中の機器に対してこのコマンドを実行すると、SRQ ラインが False に復帰します。

gp_tmout **データ送受信時のバス・タイムアウトを指定する**

書式 int gp_tmout (int time);

機能 バスタイムアウトパラメータを設定します。

引数 int time;
● 1 time は 1 秒です。
● time は整数定数または整数型変数で、0 から 255 まで指定できます。
● タイムアウトは 1 バイトごとに設定されます。
● デフォルト値は 10 秒です。

関連 なし

解説 実行例

```
gp_tmout( 3 );
```

red コマンド実行時のバスタイムアウトを 3 秒に設定します。

gp_myadr **カードにセットされたアドレスを読み取る**

書式 int gp_myadr (void);

機能 バスタイムアウトパラメータを設定します。

引数 なし

関連 なし

解説 実行例

```
int da, gp_myadr();  
da=gp_myadr();
```

REX-5052 のカードのにセットしたアドレスの値を読み取り、変数 da に代入します。この値は、gp_init でセットしたカードの GPIB 機器アドレスの値です。これは他の REX カードの GPIB との互換性のために存在しています。

注) REX-5052 では実行する必要はありません。

ongperr**エラー発生時の処理関数を登録する**

書式 void ongperr (func);

機能 エラー発生時の処理関数を登録します。

引数 _gp_err = 2;

```
void           ongperr();
```

```
extern int     _gp_err;
```

```
int           func();
```

- 各関数でエラーが発生した場合、グローバル変数_gp_err = 2 としておくと、ongperr()で登録されている関数を実行します。
- また、処理関数の戻り値が、そのままエラーが発生した関数の戻り値となります。

関連 なし

解説 実行例

```
int gpiberr();
gp_wrt("3", "adcb");
_gp_err=2;
ongperr(gpiberr);
```

エラー処理関数として、gpiberr()を登録します。

例

```
A>program
error #53 - GPIB bus timeout error.
```

gp_wait**プログラムの実行を停止する**

書式 `int gp_wait (unsigned int time);`

機能 指定したパラメータで決まる時間だけ、プログラムの実行を停止します。

引数 `unsigned int time;`
● `time` は 1 秒です。
● `time` は整数定数または整数型変数で、0 から 65,535 まで指定できます。

関連 なし

解説 実行例

```
gp_wait( 10 );
```

10 秒間、プログラムの実行を停止します。

gp_csinfo**リソース情報の取得**

書式 `int gp_csinfo(unsigned int *pIObase, unsigned int *pIrqNo)`

機能 カードサービスを呼び出してカードに割り当てられているアドレス、IRQ 番号を取得

引数 `unsigned int *pIObase` : アドレス格納アドレス
`unsigned int *pIrqNo` : IRQ 番号格納アドレス

関連 なし

解説 実行例

```
if( gp_csinfo( &MyAdrs, &MyIrqNo ) != 0 )  
{  
    printf( "CardService Call Error!!\n" );  
    exit( 1 );  
}
```

リソース情報を自動取得します。

◆ 外部変数

_gp_err**エラーが発生した場合の処理を設定する**

機能 GPIB 関数で、エラーが発生した場合の処理を設定します。

解説 ユーザーが `_gp_err` に以下の値を設定することにより、エラー発生時の処理を指定することができます。

- `_gp_err=0`(デフォルト値)
エラーが発生した場合、各関数は戻り値として -1 を返し、グローバル変数 "errorno" にエラーコードを返します。

コード	内 容
2	リクエストコードのエラー
53	GPIB バスタイムアウトエラー
60	デバイスが使用可能な状態にない
61	バッファオーバーフロー
90	バウンダリエラー

(エラーコードは 10 進数で表記しています)

- `_gp_err=1`
エラーが発生した場合、そのエラー番号とエラー内容を表示し、プログラムを終了します。
- `_gp_err=2`
エラーが発生した場合、`ongperr()` で登録された関数を実行します。また、グローバル変数 "errorno" にエラーコードを返し、エラー処理関数の戻り値が、そのままエラーが発生した関数の戻り値となります。
- 上記以外の値は動作不定。

_gp_icnt**データ受信時の有効な受信バイト数を返す**

機能 `gp_red, gp_tfrin, gp_tfi` 実行後の実際に受信したデータ数(バイト数)を返します。

解説 データ受信時の有効な受信バイト数を返します。ただし、エラーが発生した場合は前回の値が残ります。

(4-3-2) MS C での応用プログラム例

◆応用例 1 (添付ディスク中の"HP3478A.C")

HP 社のボルトメータ 3478A(HP3478A)を使って電圧測定を行います。

- 注意点
- ・ REX-5052のI/OベースアドレスはREXGPCSで指定した値をカードサービスから取得します。
 - ・ HP3478Aの GPIB アドレスは3に設定します。REX-5052は0とする。
 - ・ 接続計測器 HP3478A : ヒューレットパッカード デジタルマルチメータ

```
/*
 * REX-5052 GPIB PC Card Sample Program
 */
#include <stdio.h>
#include <conio.h>
#include <memory.h>
#include <string.h>

#include "gplib.h"          /* Gplib.Hは必ずインクルードしてください */

void main( void )
{
    int      Status;
    char     ReadBuf[128];
    unsigned int MyAdrs;      /* REX5052が使用するI/Oアドレス */
    unsigned int MyIrqNo;    /* REX5052が使用する割り込み番号 */

    _gp_err = 1;

    /*
     * カードサービス使用時I/Oアドレスを自動取得します
     * ホットケーブル使用時は自動取得できません
     */
    if ( gp_csinfo( &MyAdrs, &MyIrqNo ) != 0 )
    {
        printf( "CardService Call Error!!\n" );
        goto MAIN_EXIT;
    }

    /* GPBIOS初期化 */

    if( gp_init( MyAdrs, 0 ) == -1 )
    {
        printf( "GPBIOS Error!!\n" );
        goto MAIN_EXIT;
    }

    /* GPIBに接続されている全ての機器を初期化 */
    gp_cli();

    /* GPIB接続機器をリモート指定 */
    gp_ren();
}
```

```
/* HP3478A にデバ`イスクリアコマンド`を送信 */
gp_clr( "3" );

/* HP3478A に電圧測定コマンド`送信 */
gp_wrt( "3","HOKM01" );

while( 1 )
{
    /* デバ`イストラ`イガコマンド`送信 */
    gp_trg( "3" );

    /* シリアルポ`-ルを実行し HP3478A のステ`-タスをリ`-ドして `ステ`-レ`化` `ットが立つまで待つ */
    Status = 0;
    while( Status != 0x41 )
    {
        Status = gp_rds( "3" );
        /* printf( "Status = %x`n", Status ); */
    }

    /* 計測`ステ`-タスをリ`-ド */
    memset(ReadBuf,0x00,sizeof(ReadBuf));
    gp_red( "3", ReadBuf );
    printf( "HP3478A : %s`n", ReadBuf );
}
MAIN_EXIT:
printf( "End of program`n" );
}
```

◆応用例 2 (添付ディスク中の"YEW2553.C")

YEW2553 に出力電圧値を設定しその出力した電圧を HP3478A で計測を行います。

- 注意点
- ・ REX-5052のI/OベースアドレスはREXGPCSで指定した値をカードサービスから取得します。
 - ・ HP3478AのGPIBアドレスは3、YEW2553は4、REX5052は0に設定します。
 - ・ 接続計測器 YEW2553で発生した電圧をHP3478Aで計測
 HP3478A : ヒューレットパッカード デジタルマルチメータ
 YEW2553 : 横河電機 標準電圧発生器

```

/*
 * REX-5052 GPIB PC Card Sample Program
 */

#include <stdio.h>
#include <conio.h>
#include <memory.h>
#include <string.h>

#include "gplib.h"          /* Gplib.Hは必ずインクルードしてください */

void main( void )
{
    int      Volt;
    int      Status;
    char     ReadBuf[128];
    char     WriteBuf[128];
    char     InBuf[128];
    unsigned int MyAdrs;      /* REX5052が使用するI/Oアドレス */
    unsigned int MyIrqNo;    /* REX5052が使用する割り込み番号 */

    _gp_err = 1;

    /*
     * カードサービス使用時I/Oアドレスを自動取得します
     * ホットケーブル使用時は自動取得できません
     */
    if ( gp_csinfo( &MyAdrs, &MyIrqNo ) != 0 )
    {
        printf( "CardService Call Error!!\n" );
        goto MAIN_EXIT;
    }

    /* GPBIOS初期化 */
    if( gp_init ( MyAdrs, 0 ) == -1 )
    {
        printf( "GPBIOS Error!!\n" );
        goto MAIN_EXIT;
    }
}

```



```
/* GPIB に接続されている全ての機器を初期化 */
gp_cli();

/* GPIB 接続機器をリモート指定 */
gp_ren();

/* HP3478A と YEW2553 にテラバイトを送信 */
gp_clr( "3,4" );

/* HP3478A に電圧測定コマンド送信 */
gp_wrt( "3","HOKM01");

while( 1 )
{
    /* 出力電圧値を入力設定 */
    printf( "YEW2553 出力電圧入力(単位 mV):" );
    scanf ( "%d", &Volt );
    sprintf( InBuf, "%05d", Volt );

    /* YEW2553 電圧出力コマンドデータの設定
     * V3:10Vレンジ設定
     * P0:極性+
     * 00:出力値
     * Ex.10V 電圧出力時のコマンド:"V3P0S0000000" */
    memset( WriteBuf, 0x00, sizeof( WriteBuf ) );
    strcpy( WriteBuf, "V3P0");
    strcat( WriteBuf, InBuf );
    strcat( WriteBuf, "00" );
    printf( "YEW2553 電圧出力コマンド=[%s]¥n", WriteBuf );

    /* YEW2553 の出力電圧設定 */
    gp_wrt( "4", WriteBuf );

    /* YEW2553 テラバイトコマンド送信 */
    gp_trg( "4" );

    /* YEW2553 出力値 */
    gp_wrt( "4", "01" );
    gp_trg( "4" );

    /* HP3478A サンプリング開始 */
    gp_trg( "3" );

    /* シリアルポートを実行し HP3478A のステータスをリードして出力データが立つまで待つ */
    Status = 0;
    while( Status != 0x41 )
        Status = gp_rds( "3" );

    /* HP3478A よりデータ入力 */
    memset( ReadBuf, 0x00, sizeof(ReadBuf) );
    gp_red( "3", ReadBuf );
    printf( "HP3478A 測定電圧 = %s¥n", ReadBuf );
}
MAIN_EXIT:
    printf( "End of program¥n" );
}
```

(4-3-3) Turbo C , Borland C での応用プログラム例

◆応用例 1 (添付ディスク中の"HP3478A.C")

HP 社のボルトメータ 3478A(HP3478A)を使って電圧測定を行います。

- 注意点
- ・ REX-5052のI/OベースアドレスはREXGPCSで指定した値をカードサービスから取得します。
 - ・ HP3478AのGPIBアドレスは3に設定します。REX-5052は0とする。
 - ・ 接続計測器 HP3478A : ヒューレットパッカード デジタルマルチメータ

```

/*
 * REX-5052 GPIB PC Card Sample Program
 */
#include <stdio.h>
#include <process.h>
#include <stdlib.h>
#include "gplib.h"
void main( void )
{
    int      i, Status;
    char     ReadBuf[128];
    unsigned int MyAdrs;           /* REX5052が使用するI/Oアドレス */
    unsigned int MyIrqNo;        /* REX5052が使用する割り込み番号 */

    _gp_err = 1;                /* GPIBがエラー発生した場合プログラム終了 */

    if( gp_csinfo( &MyAdrs, &MyIrqNo ) != 0 ){          /* I/Oアドレス自動取得 */
        printf( "CardService Call Error!!\n" );
        exit( 1 );
    }
    if( gp_init( MyAdrs, 0 ) == -1 ) {                  /* GPBIOS初期化 */
        printf( "GPBIOS Error!!\n" );
        exit( 1 );
    }
    gp_cli();                                          /* GPIB機器初期化 */
    gp_ren();                                          /* GPIB接続機器をリモート指定 */
    gp_clr( "3" );                                    /* HP3478Aにデジタルスクリーンを送信 */
    gp_wrt( "3", "HOKM01" );                          /* HP3478Aに電圧測定コマンド送信 */
    gp_srq( 1 );                                       /* SRQ割り込み許可 */
    for(i=0; i<10; i++){
        gp_trg( "3" );                                /* デジタルトリガコマンド送信 */
        Status = gp_wsrq( 1, 30 );                    /* SRQ信号待ち */
        if( Status != 0 ){
            printf( "SRQ Error!\n" );
            exit( 1 );
        }
        Status = gp_rds( "3" );                        /* シリアルポート実行 */
        if( Status & 0x40 == 1 ){
            printf( "No SRQ from HP3478A!\n" );
            exit( 1 );
        }
        gp_red( "3", ReadBuf );                       /* 計測データをリード */
        printf( "HP3478A : %s\n", ReadBuf );
    }
}

```

◆応用例 2 (添付ディスク中の"YEW2553.C")

YEW2553 に出力電圧値を設定しその出力した電圧を HP3478A で計測を行います。

注意点 ・ REX-5052のI/OベースアドレスはREXGPCSで指定した値をカードサービスから取得します。

・ HP3478Aの GPIBアドレスは3、YEW2553は4、REX5052は0に設定します。

・ 接続計測器 YEW2553で発生した電圧をHP3478Aで計測

HP3478A : ヒューレットパッカード デジタルマルチメータ

YEW2553 : 横河電機 標準電圧発生器

```

/*
 * REX-5052 GPIB PC Card Sample Program
 */
#include <stdio.h>
#include <process.h>
#include <stdlib.h>
#include <conio.h>
#include <memory.h>
#include <string.h>
#include "gplib.h"
void main( void )
{
    int      i, Volt, Status;
    char     ReadBuf[128];
    char     WriteBuf[128];
    char     InBuf[128];
    unsigned int MyAdrs;           /* REX5052が使用するI/Oアドレス */
    unsigned int MyIrqNo;        /* REX5052が使用する割り込み番号 */

    _gp_err = 1;                 /* GPIBエラー発生した場合プログラム終了 */
    if ( gp_csinfo( &MyAdrs, &MyIrqNo ) != 0 ){ /* I/Oアドレス自動取得 */
        printf( "CardService Call Error!!\n" );
        exit( 1 );
    }
    if( gp_init( MyAdrs, 0 ) == -1 ){ /* GPBIOS初期化 */
        printf( "GPBIOS Error!!\n" );
        exit( 1 );
    }
    gp_cli();                    /* GPIB機器初期化 */
    gp_ren();                    /* GPIB接続機器をリモート指定 */
    gp_clr( "3,4" );             /* GPIB接続機器にデフォルトコマンドを送信 */
    gp_wrt( "3","HOKM01" );     /* HP3478Aに電圧測定コマンド送信 */
    for(i=0; i<10; i++){
        /*
         * 出力電圧値を入力設定
         */
        printf( "YEW2553出力電圧入力(単位 mV):" );
        scanf ( "%d", &Volt );
        sprintf( InBuf, "S%05d", Volt );
    }
}

```

```
/*
 * YEW2553電圧出力コマンドデータの設定
 */
memset( WriteBuf, 0x00, sizeof( WriteBuf ) );

strcpy( WriteBuf, "V3P0");
strcat( WriteBuf, InBuf );
strcat( WriteBuf, "00" );

gp_wrt( "4", WriteBuf );      /* YEW2553の出力電圧設定 */
gp_trg( "4" );              /* YEW2553トリガコマンド送信 */
gp_wrt( "4", "01" );        /* YEW2553出力オン */
gp_trg( "4" );
gp_trg( "3" );              /* HP3478Aサンプリング開始 */

Status = 0;
while( Status != 0x41 )      /* シリアルポートを実行しデータレディまで待つ */
    Status = gp_rds( "3" );

memset( ReadBuf, 0x00, sizeof(ReadBuf) );
gp_red( "3", ReadBuf );      /* HP3478Aよりデータ入力 */
printf( "HP3478A 測定電圧 = %s\n", ReadBuf );
}
}
```

(4-4) N88Basic での使用

◆ N88Basic モードでのメモリ確保

GPBIOS.COM は、必ず N88BASIC.EXE を実行する前に実行させなければなりません。この順序を間違えると、GPBIOS 部が、N88Basic インタプリタによって破壊され、REX-5052 は動作しませんので、御注意ください。

また GPBIOS を呼ぶリンカである GLN88.O をロードする前に、必ず CLEAR 文及び DEF SEG 文により、機械語領域を確保してください。

GLN88.O は、BASIC 起動後、BLOAD コマンドを使用して機械語セグメント内にロードします。ただし、BASIC 起動前には必ず GPBIOS.COM が実行されていなければなりません。

作成する BASIC によるアプリケーションプログラムの先頭で必ず下記の宣言を行い機械語領域を確保してください。

```
10 CLEAR &H800
20 DEF SEG=SEGPTR(2)
30 BLOAD "GLN88.O"
40 GPSTART=0
```

10・・・機械語セグメントエリアの確保

値は&H3FF 以上の値が必要です。

&H800 の場合は、機械語エリアとして&H800(2K)バイト確保されます。

20・・・機械語セグメントの宣言。

30・・・GLN88.O のロードとインストール

40・・・CALL 文での機械語スタート番地の定義

以上の BASIC プログラムを実行させることにより、リンカ(GLN88.O)は起動されます。

◆ N88Basic 用リンカ GLN88.O のコマンド

(4-4-1) コマンドの概要

GLN88.O に対するコマンドは、コマンドを含む文字列を CALL 文の引数として GLN88.O に引き渡すことにより実行されます。具体的には下記の様にコマンド文字列を文字変数内に格納し、また戻り値とともに CALL 文の引数とします。

例 - 1)

```
G$="CLR3":CALL GPSTART(STS%,G$)
```

例 - 2)

```
G$="RED3;" +STRRING$(20,32):CALL GPSTART(STS%,G$)
```

例 - 3)

```
AD=3  
T$="HOKM01"  
G$="WRT"+STR$(AD)+";"+T$  
CALL GPSTART(STS%,G$)
```

コマンドは、英字 3 文字より構成されています。英字は、大文字、小文字どちらでも受け付けます。

コマンドは、"!"(SHIFT+¥)で区切るにより、ひとつの引数内に複数のコマンドを入れることができます。GLN88.O は、連続してこれらのコマンドを実行します。

また STS%は戻り値を表し、0 であれば、エラーなし、それ以外であればエラーが発生したことを示します。

```
G$="CLI &H0D0!REN!CLR3"  
CALL GPSTART(STS%,G$)
```

◆コマンド一覧表

コマンド	機能	使用例	HPL 言語の例
WRT	GPIB 上にデータを出力する。	GP\$="WRT3;FIR4":call gpstart (sts%,GP\$)アドレス3の機器にFIR4という文字列データを送り出す。	wrt 703,A\$
RED	GPIB 上のデータを読み込む。	GP\$="RED 3;" +STRING\$(20,32) call gpstart(sts%,GP\$)アドレス3の機器よりデータを受け取り文字変数 GP\$内の一部 STRING\$(20,32)に代入する。	red 703,A\$
TRG	デバイストリガコマンドを送出する。	GP\$="TRG3" call gpstart(sts%,GP\$)アドレス3の機器にトリガコマンドを送る。	trg 703
REN	GPIB の REN ラインを "True"にする。	GP\$="REN" call gpstart(sts%,GP\$)	ren 7
CLR	デバイスクリアコマンドを送出する。	GP\$="CLR3,12" call gpstart(sts%,GP\$)アドレス3,12の機器にデバイスクリアコマンドを送る。	clr 70312
LCL	指定された機器をローカルモードにする	GP\$="LCL 3" call gpstart (sts%,GP\$)アドレス3の機器をローカルモードにする。	lcl 703
LLO	GPIB 上の全機器のローカルスイッチを無効にする。	GP\$="LLO" call gpstart(sts%,GP\$)	llo 7
CLI	GPIB の IFC ラインを一定期間 "TRUE"にする。	GP\$="CLI &H0D0,&H0" call gpstart(sts%,GP\$)	clr 7
WTB	ATN ラインを "TRUE" にしてデータを送出する。	GP\$="WTB"+CHR\$(&H5F) call gpstart(sts%,GP\$) UNTLK コマンドを送出する。	
RDS	シリアルポートを実行しステータスバイトを読み込む。	GP\$="RDS 3" call gpstart(sts%,GP\$)アドレス3の機器に対してシリアルポートを実行し読み込んだステータスバイトの値をステータスバッファに格納する。	a=rds(703)
RDS1	(RDS1 では最後に UNT コマンドを送出しません)		
TFI	バッファメモリ上に GPIB 上のデータを読み込む	G\$="TFI3;&H20,&H800,&H7000":call gpstart(sts%,G\$)	
TFO	バッファメモリ上のデータを GPIB 上に送る。	G\$="TFO3;&H20,&H800,&H7000":call gpstart(sts%,G\$)	
SRQ	コントローラとして SRQ 割込みの受付を可能にする。	GP\$="SRQ1" call gpstart(sts%,GP\$)	
TMO	バスタイムアウトパラメータを設定する。	G\$="TMO 10" call gpstart(sts%,G\$)	
TDL	トーカーモードでのデリミタを設定する。	G\$="TDL &H8A" call gpstart(sts%,G\$)	
LDL	リスナモードでのデリミタを設定する。	G\$="LDL&H0A" call gpstart(sts%,G\$)	
MYAD	カードの GPIB 機器アドレスを読み取る。	G\$="MYAD" call gpstart(sts%,G\$)	
WAIT	指定された時間ウェイトする。	G\$="WAIT 30" call gpstart(sts%,G\$)	
WSRQ	指定された時間 SRQ を待つ。	G\$="WSRQ 100" call gpstart(sts%,G\$)	

*sts%は戻り値でノーエラーの場合は"0"です。

コマンド一般形式を下記に示します。

TYPE1

パラメータ部のないコマンド

"LLO","REN"

TYPE2

パラメータ部のあるコマンド

"CLI &H0D0,0", "TMO 10", "TDL &H8A"

パラメータ部は 10 進文字列または、&H で始まる 16 進文字列であることが必要です。

TYPE3

アドレス指定のあるコマンド

"CLR 3,4,5, &H1B"

アドレスは 10 進文字列または、&H で始まる 16 進文字列であることが必要です。アドレスとアドレスの区切りは","を使用します。

TYPE4

アドレス指定と、データ部のあるコマンド

"WRT 3,&H0B;ABCD123"

データ部

データ部は文字列(文字列変数内の値)であることが必要です。アドレス部とデータ部の区切りは";"を使用します。

TYPE5

アドレス指定とパラメータ部のあるコマンド

"TFI 3;1024,&H800,&H5000"

アドレス指定とパラメータ部の区切りは";"を使用します。

パラメータ部とパラメータ部の区切りは","を使用します。

コマンドの連結

複数のコマンドを連結(最大長 255 文字以内なら、連結するコマンド数、種類に制限はありません)して、一度にまとめて、連続実行させることができます。

(例)

```
G$="CLI &H0D0,0;REN;CLR 3"  
CALL GPSTART(STS%,G$)  
G$="WRT 3;H0KM00;TRG 3;RED 3"  
CALL GPSTART(STS%,G$)
```

各コマンドの区切りは";"を使用します。

◆関数仕様の記述について

本ソフトウェアを動作させるためのコマンド(引数となる文字列)の個々について解説を行います。汎例を下記に示します。

xxx(コマンド名)	機能
書式	関数の記述
関連	実行時に関連のあるパラメータ
実行例および動作	そのコマンドの実行例と GPIB 各信号線の動作を示します。

なお書式中の TADn はトーカーアドレスを、LADn はリスナアドレスを示します。ただしアドレス値は、必ず 1 ~ 15 の整数値でなければなりません。[]は省略可能を示します。

WRT

リスナアドレスで指定された機器にデータ送信

書式 WRT LAD1[,LAD2]---[,LAD15];データ文字列

関連 タイムアウト, トーカモードデリミタ

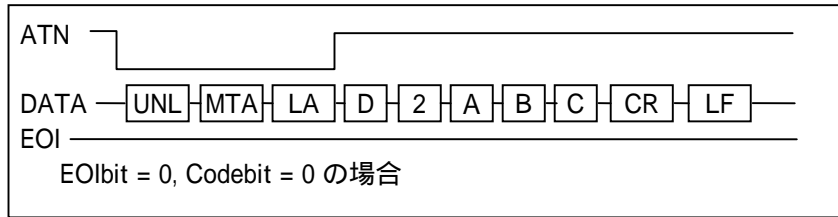
実行例および動作 [LAD1] ~ [LADn]で指定した機器(最大14個まで)に対してデータ文字列を送信し、続いてデリミタを出力する。

- トーカモードデリミタ=0 の場合
デリミタとして CR+LF コードを出力します。
- トーカモードデリミタの EOIbit が 1 で、他の 7bit(Codebit) が"0"の場合
デリミタ文字列中最後のデータバイト出力と同時に EOI を True にします。デリミタはなしです。
- トーカモードデリミタの EOIbit が"0"でコード bit が"0"以外の場合
デリミタとして、コード bit で指定されたコードを送信します。
- トーカモードデリミタの EOIbit が"1"でコード bit が"0"以外の場合
デリミタとしてコード bit で指定されたコードを送信し、同時に EOI を True にします。

実行例 1. シングリスナアドレスの場合
(トーカモードデリミタ = 0)

```
100 G$="WRT 3;D2ABC"
110 call gpstart(sts%,G$)
```

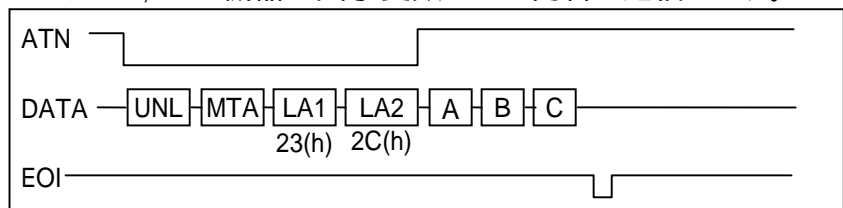
アドレス3の機器に"D2ABC"という文字列を送信します。



実行例 2. マルチリスナアドレスの場合

```
100 A$="ABC"
110 G$="WRT 3,12;"+A$
120 call gpstart(sts%,G$)
```

アドレス 3,12 の機器に文字変数 A \$ の内容を送信します。



RED

指定したトーカーよりデータ受信し変数に格納

書式 RED TAD1[,LAD1]---[,LAD15];ダミー文字列

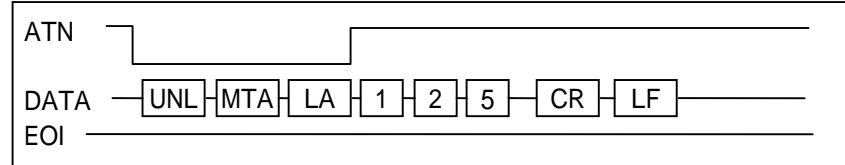
関連 タイムアウト, リスナモードデリミタコード

実行例および動作 トーカーアドレス[TAD]で指定された機器よりデータ文字列を受信し、ダミー文字列エリアに格納します。もしダミー文字列エリアよりも読み込んだ文字列の方が長い場合には、あふれ分は無視されます(ダミー文字列エリアには格納されません。)。これは N88BASIC から引渡された文字変数の長さを CALL 関数内部で変更できないためです。必ずトーカーから送られて来る文字列の長さと同様またはそれ以上の長さのダミー文字にしてください。

実行例 1. 相手側機器の送信時デリミタが LF の場合

```
100 GP$="RED 3;"+STRING$(20,32)
110 call gpstart(sts%,GP$)
```

アドレス 3 の機器よりデータを受信し、変数内に格納します。

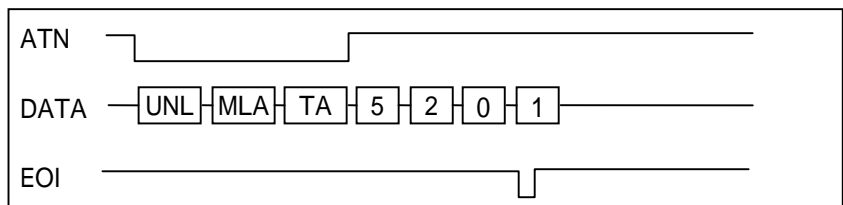


- HP 社、横河電機、アドバンテスト等、ほとんどのメーカーが送信時データとして CR,LF を使用していますので、デリミタの設定は 10(10 進数)が一般的です。
- 読み込み動作は、デリミタの検出または、EOI を検出すると終了します。

実行例 2. 相手側機器の送信時デリミタが EOI の場合

```
100 GP$="RED 3;"+STRING$(20,32)
110 call gpstart(sts%,GP$)
```

アドレス 3 の機器よりデータを受信し、変数内に格納します。



注) RED コマンドは、相手側機器から出力される EOI を検出するとその時点で読み込み動作を終了します。

TRG	リスナに指定された機器に対して GET 命令を送信
------------	----------------------------------

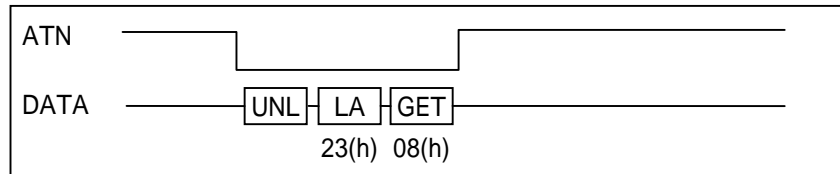
書式 TRG LAD1 [,LAD2] --- [LADn]

関連 タイムアウト

実行例および動作 ATN ラインを True にし、UNL コマンドに続いて、リスナアドレス、TRG コマンド 08(h)を送信します。

```
100 G$="TRG 3"
110 call gpstart(sts%,G$)
```

アドレス 3 の機器に対して GET 命令を送信します。



注) GPIB 用機器は、一般的に GET 命令を受信すると動作(測定など)を開始します。

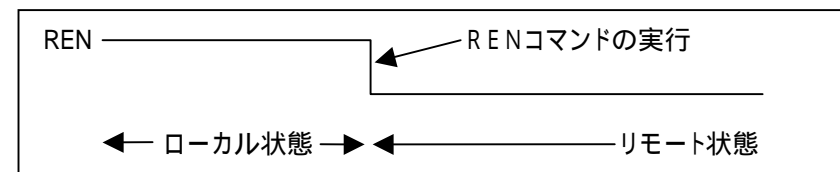
REN	REN ラインを True にする
------------	--------------------------

書式 REN

関連 なし

実行例および動

```
100 G$="REN"
110 call gpstart (sts%,G$)
```



PC 本体がリセットされるか、リスナアドレスなしの LCL コマンドが実行されるまでずっと True のままです。

CLR	DeviceClear または SelectedDeviceClear 送出
------------	---

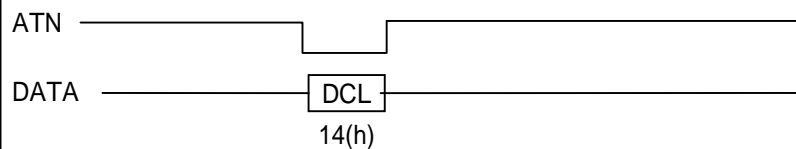
書式 CLR [LAD1][,LAD2]----[,LADn]

関連 タイムアウト

実行例および動作 アドレス指定がない場合は、ATN ラインを True にし、DCL コマンドを送信した後、ATN ラインを False にします。アドレス指定がある場合は、ATN ラインを True にして、UNL,LA,SDC コマンドを送信した後、ATN ラインを False にします。このコマンドを受信すると、機器はリセット状態になります。

実行例 1. 全機器に対する場合

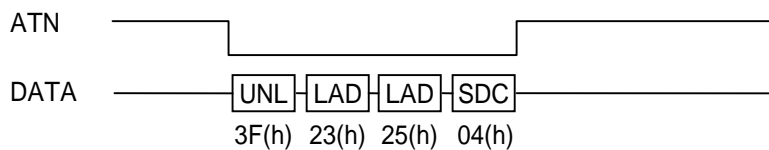
```
100 GP$="CLR"
110 call gpstart(sts%,GP$)
```



GPIB 上の全機器に対してクリアコマンドを送り、全機器をリセットします。

実行例 2. アドレス 3,5 の機器に対して、クリアコマンドを送る

```
100 GP$="CLR 3,5"
110 call gpstart(sts%,GP$)
```



相手側機器の DC(Device Clear)機能が DC0 の場合は、このコマンドは無効です。また DC2 の場合は、実行例 2 の SDC コマンドは無効となりますので、実行例 1 をご使用ください。

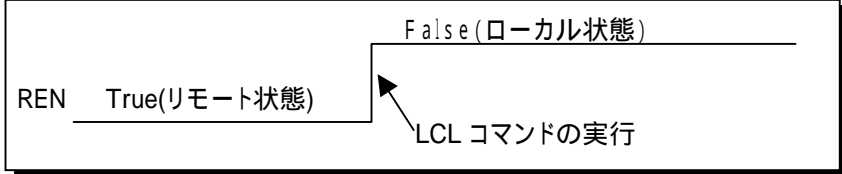
LCL 指定したリスナ機器をローカル状態に設定

書式 LCL [LAD1] [,LAD2] --- [LADn]

関連 タイムアウト

実行例および動作 実行例 1. リスナアドレスのない場合

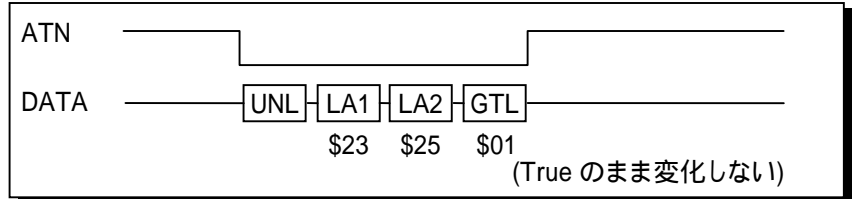
```
100 G$="LCL"
110 call gpstart(sts%,G$)
```



実行例 2. リスナアドレスの指定がある場合

```
100 G$="LCL 3,12"
110 call gpstart(sts%,G$)
```

リスナアドレス 3,5 の機器に GTL(go to local)命令を送りローカル状態に戻します。



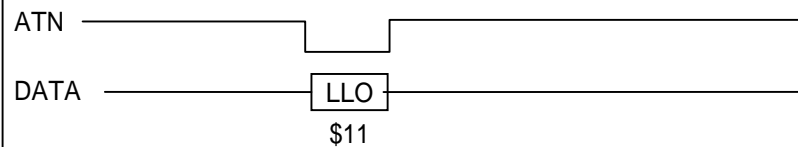
LLO GPIB 上の全機器のローカルスイッチを無効にする

書式 LLO

関連 タイムアウト

実行例および動作

```
100 G$="LLO"  
110 call gpstart(sts%,G$)
```



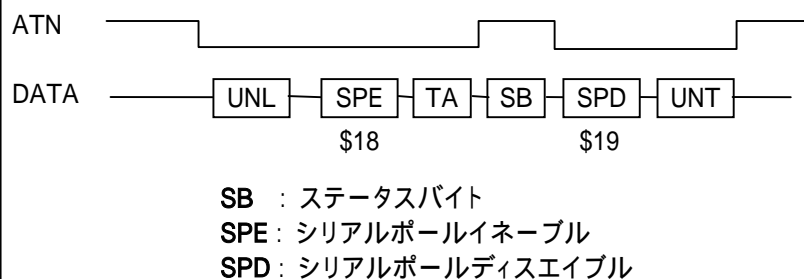
- ATN ラインを True にし、LLO 命令を送信した後 ATN ラインを False にします。この命令を受信すると機器側ではパネル上の操作スイッチを無効にします。ただし機器のリモート状態もしくはローカル状態には、変化は生じません。
- 機器の LLO 状態を解除する場合は REN ラインを False にします。(LCL コマンドの実行)

RDS	シリアルポールを実行しステータスバイトを受信
-----	------------------------

書式	RDS TAD
関連	タイムアウト

実行例および動作 ATN ラインを True にし、UNL,SPE,TA を送信した後、ATN ラインを False にします。その後、トーカーに指定した機器より送られてきたステータスバイトを受信しセーブエリアに格納します。そして再び、ATN ラインを True にし SPD コマンド、UNT を送信し、ATN ラインを False に戻します。通常このコマンドは機器より SRQ 割り込みが発せられた場合、その SRQ に対するサービスとして使用されます。
読みとったステータスバイトは PEEK 文で &H86 の値を読み出してください。

```
100 G$="RDS 2"
110 call gpstart(sts%,G$)
120 Stsbyte% = PEEK(&H86)
```



SRQ を発信中の機器に対してこのコマンドを実行すると、SRQ ラインが False に復帰します。

もし、RDS コマンドでシリアルポールがうまく実行できない場合は、RDS1 コマンドを使用してください。

RDS1**シリアルポールを実行しステータスバイトを受信**

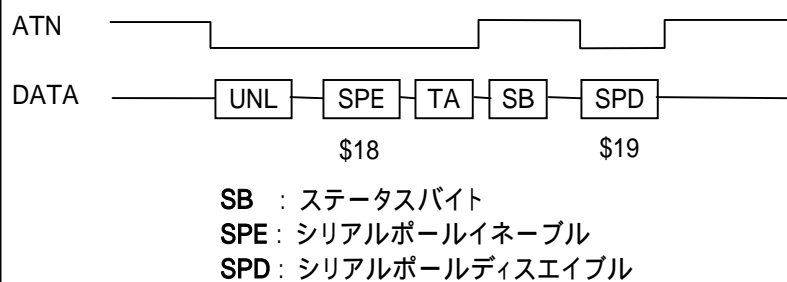
(注意)RDS との違いは、最後に UNT コマンドを送出しない点です。

書式 RDS1 TAD

関連 タイムアウト

実行例および動作 ATN ラインを True にし、UNL,SPE,TA を送信した後、ATN ラインを False にします。その後、トーカーに指定した機器より送られてきたステータスバイトを受信しセーブエリアに格納します。そして再び、ATN ラインを True にし SPD を送信し、ATN ラインを False に戻します。通常このコマンドは機器より SRQ 割り込みが発せられた場合、その SRQ に対するサービスとして使用されます。

```
100 G$="RDS1 2"
110 call gpstart(sts%,G$)
```



SRQ を発信中の機器に対してこのコマンドを実行すると、SRQ ラインが False に復帰します。

CLI

IFCラインを True にする

書式 CLI パラメータ1,パラメータ2

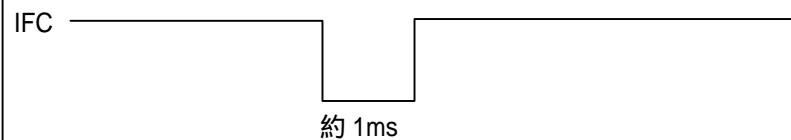
関連 なし

実行例および動作 パラメータ1はREX-5052カードのI/Oアドレスです。イネーブラで設定したI/Oアドレス値を10進文字列、または&Hで始まる16進文字列で記述します。

パラメータ2は GPIB 機器アドレス(マイアドレス)です。

```
100    G$="CLI &H0d0,7"  
110    CALL gpstart(sts%,G$)
```

REX-5052 カードの I/O アドレス(&H0d0)を GPBIOS に引渡し、マイアドレスをセットして、IFC ラインを約 1ms の間 True にします。



WTB**ATN ラインを True にしてコマンド文字列を送信**

書式 WTB データ1[,データ2]----[,データn]

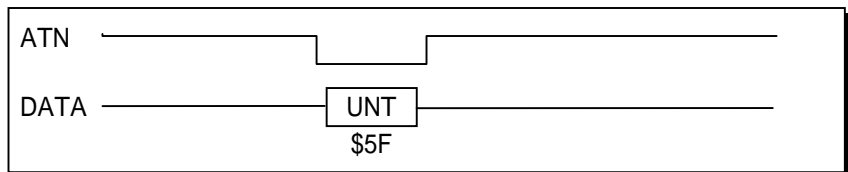
関連 タイムアウト

実行例および動作 ATN ラインを True にしてデータを送信した後、ATN ラインを False にします。

実行例 1.

```
100  G$="WTB &H5F"
110  call gpstart(sts%,G$)
```

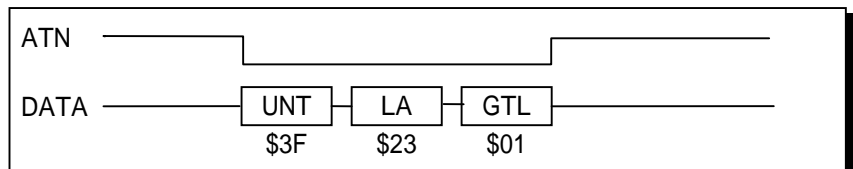
アントーク命令(UNT)をバス上に送り出し、トーカに設定されている機器のトーカモードを解除します。



実行例 2.

```
100  G$="WTB &H3F,&H23,1"
110  call gpstart(sts%,G$)
```

LCL 3 の実行と同様になります。



SRQ 機器からの割り込みの受け付けの許可、不許可を設定

書式 SRQ 1 または SRQ 0

関連 なし

実行例および動作 コントローラとして、機器からの SRQ 割り込み受け付けの許可、不許可を設定します。

実行例 1.

```
G$="SRQ1":call gpstart(sts%,G$)
```

SRQ 割り込みの受付を GPBIOS レベルで許可します。

実行例 2.

```
G$="SRQ0":call gpstart(sts%,G$)
```

SRQ 割り込みの受付を不許可(マスク)にします。
GPBIOS のデフォルト値は、この状態になっています。

TFI	トーカーより送られてくるデータをメモリに格納する
-----	--------------------------

書式 TFI TAD[*LAD1*]-*---*[*LAD15*];パラメータ 1,パラメータ 2,
パラメータ 3

関連 なし

実行例および動作 トーカーを指定してトーカーより送られてくるデータを受信し、指定されたセグメント、オフセットアドレスより始まるメモリ上に直接格納します。受信動作は EOI の検出または、指定バイトカウントになると終了します。

パラメータ 1~3 はそれぞれ下記の様な意味を持ちます。

パラメータ1 ---- 受信用バッファサイズ。

1~&H10000 の範囲の 10 進文字列、または&H で始まる 16 進文字列で記述します。トーカーより送られて来るデータを充分格納しうる大きさが必要です。単位はバイトです。

パラメータ2 ---- 受信用バイトの先頭オフセットアドレス。

0~&HFFFF の範囲の 10 進文字列、または&H で始まる 16 進文字列で記述します。

パラメータ3 ---- 受信用バイトの先頭セグメントアドレス。

0~&HFFFF の範囲の 10 進文字列、または&H で始まる 16 進文字列で記述します。

- 各パラメータの決定の際は下記に御注意ください。

転送アドレスの指定は 0~&HFFFF までしか行なうことができません。したがって、パラメータ 2,3 によって指定されたバッファの先頭アドレス(20bit アドレス)に、パラメータ 1 によって指定されたバッファサイズを加え、1 を引いた値(バッファの最終アドレス)の下位 16bit が&HFFFF を超えることはできません。GPBIOS 内ではこのチェックを行ない、&HFFFF を超える場合には、メモリバウンダリーエラーとして、エラーコード 90 を返し、異常終了します。

例

```
TFI 3;&H8000,&H100,&H7C40
```

```
    バッファの先頭アドレス  &H7C500
```

```
    バッファの終了アドレス  &H7C500+&H8000=&H84500
```

バッファの終了アドレスが&H7FFFF を超えてしまいますので、この場合はバウンダリーエラーとなります。

先頭オフセットアドレスとして&H100 セグメントアドレスとして&H7C40 を指定した場合には、可能な最大バッファサイズは&H7FFFF-&H7C500=&H3600 バイトとなります。

実行例 1.

```
G$="TFI 3;600,&H800,&H"+HEX$(VARPTR(SYSTEM.1))
call gpstart(sts%,G$)
```

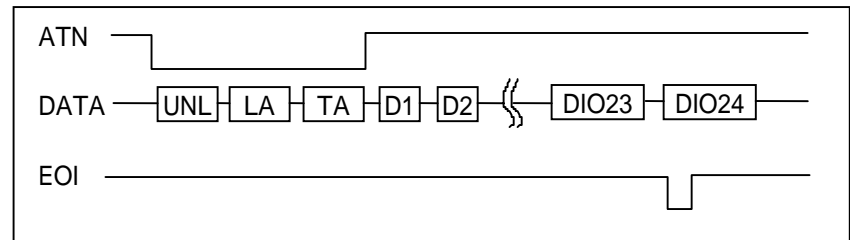
機械語セグメントエリアの先頭からオフセットアドレス&H800 より始まり、大きさが 600 バイトのバッファエリア内に、トーカアドレス 3 の機器より送られて来るデータを格納します。

この場合は、機械語セグメントエリアの大きさを 3K バイト以上、CLEAR 文で確保しておく必要があります。

実行例 2.

```
10 DIM A%(1000)
20 FOR I=0 TO 1000
30             A%(I)=0
40 NEXT
50 G$="TFI 3;2002,&H"+HEX$(VARPTR(A%(0),0)
           +",&H"+HEX$(VARPTR(A%(0),1)
60 call gpstart(sts%,G$)
```

整数型配列変数の中に直接、受信データを格納します



TFO

バッファメモリ内のデータを送信する

書式 TFO LAD1[,LAD2]---[,LAD15];パラメータ 1,パラメータ 2,
パラメータ 3

関連 なし

実行例および動作 リスナを指定し、指定されたセグメント、オフセットアドレスより始まるバッファメモリ内のデータを指定されたバイト数分送信します。最終データの出力と同時に EOI を True にします。パラメータ 1----送信するバイト数を、1~&HFFFF の範囲の 10 進文字列、または &H で始まる 16 進文字列で記述します。送信バッファの最終アドレスとの関係で制限されます。パラメータ 2 , パラメータ 3----TFI と同様です。パラメータ 1,2,3 は、TFI と同様な制限があります。例えば、パラメータ 2,3 をそれぞれ&H100,&H7C40 と指定した場合には &H7FFFF-&H7C500=&H3600 となり、13824 バイトまでしか送信することができません。

実行例 1.

```
G$="TFO 1;512,&H800,&H"+HEX$(VARPTR(SYSTEM.1))
call gpstart(sts%,G$)
```

機械語セグメントエリアの先頭からオフセットアドレス&H800 より始まるバッファメモリ内のデータを 512 バイト送信します。

実行例 2.

```
10 DIM DT0%(360)
20 PY=3.14159/180
30 FOR PX =0 TO 359
40     DT0%(PX)=100 * SIN(PX * PY)
50 NEXT
60 G$"TFO 3;720,&H"+HEX$(VARPTR(DT0%(0),0))
    +",&H"+HEX$(VARPTR(DT0%(0),1))
70 call gpstart(sts%,G$)
```

整数型配列変数内のデータ(sin 波の 1 周期分)を送信します。

TMO

バスタイムアウトパラメータを設定

書式 TMO パラメータ 1

関連 なし

実行例および動作 パラメータ1は、バスのモニタを行なう時間を 0～255 の範囲の 10 進文字列、または&H で始まる 16 進文字列で記述します。単位は秒です。

GPBIOS は、トーカー、リスナーとのハンドシェイクの応答時間を常に監視しています。コマンドの送信時、データの送受信時に、この設定時間以内に応答がない場合にはバスタイムアウトエラーとなります。

GPBIOS のデフォルト値は 10 秒に設定されています。

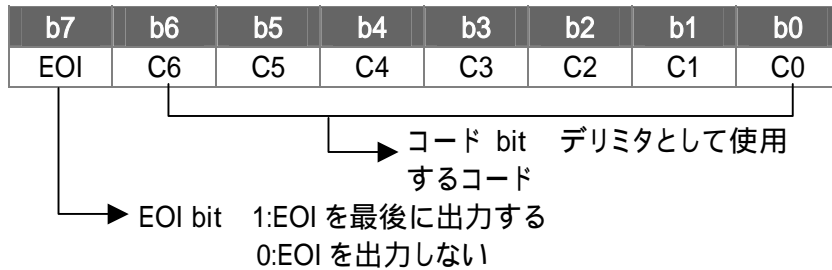
TDL

トーカーモードでのデリミタの設定

書式 TDL パラメータ 1

関連 なし

実行例および動作 パラメータ 1 は 0 ~ 255 の範囲の値を、10 進文字列または &H で始まる 16 進文字列で記述します。値の意味を下記に示します。



- EOIbit=0、コード bit=0 の場合はデリミタとして CR+LF が送信されます。
- EOIbit=1、コード bit=0 の場合は、最後のデータ出力と共に EOI が出力されます。
- GPBIOS のデフォルト値は、デリミタとして CR+LF を使用し、EOI は出力しません。

実行例 1.

```
G$="TDL &H80":call gpstart(sts%,G$)
```

最後のデータバイト送信と同時に EOI を True にします。

実行例 2.

```
G$="TDL &H83":call gpstart(sts%,G$)
```

デリミタとして ETX(03)を送信し、デリミタの送信と同時に EOI を出力します。

LDL **リスナモードでのデリミタコードの設定**

書式 LDL パラメータ

関連 なし

実行例および動作 パラメータには、デリミタコードとして、使用する文字コードを 10 進文字列、または &H で始まる 16 進文字列によって設定します。

実行例 1.

```
G$="LDL &H0A":call gpstart(sts%,G$)
```

受信デリミタコードを "LF" に設定します。

実行例 2.

```
G$="LDL3":call gpstart(sts%,G$)
```

受信デリミタコードを "ETX" に設定します。

GPBIOS は、デフォルト値として受信デリミタコードを "LF" に設定しています。

MYAD **カード上に設定したアドレスの値を読み取る**

書式 MYAD

関連 なし

実行例および動作 GPBIOS 内で設定されているアドレスの値を読み取り、GLN88.0 内のワークエリア(オフセット &H1C0)に格納されます。
注) プログラムで新たに自分の機器アドレスを知る必要がない場合は、実行する必要はありません。

```
G$="MYAD":call gpstart(sts%,G$)  
A=PEEK(&H1C0)
```

WAIT**指定された時間ウェイトする**

書式 WAIT パラメータ

関連 なし

実行例および動作 パラメータで指定された時間(1 秒単位)だけウェイトします。このウェイトは、GPBIOS 内で、単純にウェイトしているだけなので、BASIC プログラムで使用する場合は注意してください。

```
G$="WAIT 30":call gpstart(sts%,G$)
```

WSRQ**指定時間 SRQ を待つ**

書式 WSRQ パラメータ

関連 なし

実行例および動作 SRQ が来るまで、指定された時間(1 秒単位)ウェイトします。実行終了後、SRQ が来ていれば、GLN88.0 内のワークエリア(オフセット&H1C2)の内容をクリアします。時間以内にSRQが来なければ、ワークエリアに&HFF をセットします。

```
G$="WSRQ 100":call gpstart(sts%,G$)  
S=PEEK(&H1C2)
```

(4-4-2) エラー

GLN88.O は、引き渡されたコマンドにエラーがあった場合や、実行中に何らかのエラーが発生した場合に、エラーコードを引数 STS% にセットし、エラーの発生を通知します。BASIC 側は、エラーコードに対応して処理をしてください。

エラーコード	内 容	意 味
2	文法に誤りがある	パラメータの記述、アドレス部の記述に誤りがある
5	関数または命令の呼び方に誤りがある	コマンドの記述に誤りがある
53	入出力装置にエラーが発生した	バスタイムアウトエラーが発生した
60	指定の装置は使用できない	CLI コマンドのパラメータで指定した I/O アドレスにカードがアドレスされていない
61	回線の入力バッファがあふれた	RED コマンドの受信バッファ用のダミー文字列の長さが短かすぎる
90	未定義エラーコードによるエラーが発生した	TFI, TFO コマンドでパラメータ 1, 2, 3 の設定が正しくなく、バウンダリーエラーを起こした

(4-4-3) BASIC による応用プログラム例

応用例 添付ディスク中の"GPDEMO1"

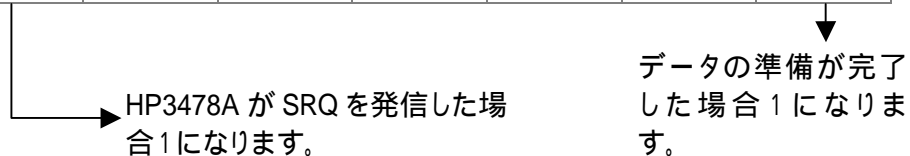
HP 社のデジタルマルチメータ 3478A を使用した例です。PC 本体内のインターバルタイマを利用し、3 秒間隔で 3478A にトリガ命令を送り、測定を行なわせます。

3478A は、トリガ命令を受信した後、データの送信準備が完了すると、SRQ 割り込みを発生します。

PC は、この SRQ 割り込みを検出した後、シリアルポールを行ない、ステータスバイトの値が正しければ、3478A からデータを読みこみ、CRT 上に表示します。

3478A のステータスバイトの内容を示します。

b7	b6	b5	b4	b3	b2	b1	b0
Power ON	SRQ	Call fail	フロントパネル SRQ	ハードウェアエラー	文法エラー	常に 0	Data ready



データレディ SRQ の場合はステータスバイトの値が"01000001"=&H41 となります。

```

10 ' GPIB N88 Basic Test Program
15 ' REX-5052 & HP3478A Test
20 CLEAR &H3FF
25 DEF SEG=SEGPTR(2)
26 BLOAD " gln88.o"
30 GPSTART=0
35 '
40 '
50 G$="cli&hd0,&h0"          'GPIB 起動 I/O アドレス=&h0d0,GPIB 機器アドレス=0
60 '
70 STS%=0
80 CALL GPSTART(STS%,G$)
90 G$="ren"
100 CALL GPSTART(STS%,G$)
110 G$="clr 3"
120 CALL GPSTART(STS%,G$)
130 G$="wrt 3;HOKM00"
140 CALL GPSTART(STS%,G$)
150 G$="trg 3"
160 CALL GPSTART(STS%,G$)
170 FOR I=0 TO 500:NEXT
180 G$="rds 3"
190 CALL GPSTART(STS%,G$)
200 G$="red 3;          "
210 CALL GPSTART(STS%,G$)
220 PRINT G$
230 GOTO 150

```

第5章 Windows3.1 DLL ライブラリ関数仕様

◆関数仕様の記述について

本ソフトウェアを動作させるための個々のコマンドについて解説を行います。汎例を下記に示します。書式及び実行例は Visual C と Visual Basic 両方を記述します。

gp_xxx(コマンド名)	機能
書式	VC > Visual C での関数の記述 VB > Visual BASIC での関数の記述
関連	実行時に関連のあるパラメータ

実行例および動作 そのコマンドの実行例と GPIB 各信号線の動作を示します。

留意点

すべての関数は INT 型の戻り値を返します。

戻り値は、0 の場合は正常終了です。それ以外はエラーコードです。

機器アドレスの指定は文字列で行ないます。(各コマンドの解説では書式の項目で "char far *adrs" で示されています。)

このとき、トーカ指定が必要なコマンドでは、文字列の先頭の機器アドレスがトーカアドレスとなります。

(例) リスナアドレス 1,3,4,8 の場合 : adrs = "1,3,4,8"

全機器に対する場合 : adrs = "" (ヌル文字列)

引き数に関する注意

GPIB.DLL を呼び出す場合 (DLL ルーチン呼び出す場合全てに当てはまります) には、必ずポインタを渡す引き数は FAR ポインタとすることが必須です。C 言語の場合には注意してください。Visual Basic の場合には、言語仕様として DLL を呼び出す場合には FAR ポインタとなりますので、特に注意は必要ありません。

また整数型変数で呼び出す場合には、値を渡す場合には、ByVal val1 As Integer アドレスを渡す場合には val1 As Integer という構文になります。

◆関数一覧(C言語の場合)

```

int FAR PASCAL gp_init( int , int );
int FAR PASCAL gp_cli();
int FAR PASCAL gp_ren();
int FAR PASCAL gp_clr( char far* );
int FAR PASCAL gp_lcl( char far* );
int FAR PASCAL gp_llo();
int FAR PASCAL gp_red( char far*, char far* );
int FAR PASCAL gp_wrt( char far*, char far* );
int FAR PASCAL gp_tfrin( char far*, unsigned int, char far* );
int FAR PASCAL gp_tfrout( char far*, unsigned int, char far* );
int FAR PASCAL gp_rds( char far*, unsigned int far* );
int FAR PASCAL gp_rds1( char far*, unsigned int far* );
int FAR PASCAL gp_trg( char far* );
int FAR PASCAL gp_wait( unsigned int );
int FAR PASCAL gp_wsrq( unsigned int );
int FAR PASCAL gp_wtb( char far* );
int FAR PASCAL gp_delm( char far*, unsigned int );
int FAR PASCAL gp_tmout( unsigned int );
int FAR PASCAL gp_myadr();

```

◆関数一覧(Visual Basic の場合)

```

Declare Function gp_init Lib "gplib.dll" (ByVal Val1 As Integer,ByVal Val2 As Integer) As Integer
Declare Function gp_cli Lib "gplib.dll" () As Integer
Declare Function gp_clr Lib "gplib.dll" (ByVal Str1 As String) As Integer
Declare Function gp_lcl Lib "gplib.dll" (ByVal Str1 As String) As Integer
Declare Function gp_llo Lib "gplib.dll" () As Integer
Declare Function gp_ren Lib "gplib.dll" () As Integer

Declare Function gp_red Lib "gplib.dll" (ByVal Str1 As String, ByVal Str2 As String) As Integer
Declare Function gp_wrt Lib "gplib.dll" (ByVal Str1 As String, ByVal Str2 As String) As Integer
Declare Function gp_tfrin Lib "gplib.dll" (ByVal Str1 As String, ByVal Str2 As String) As Integer
Declare Function gp_tfrout Lib "gplib.dll" (ByVal Str1 As String, ByVal Str2 As String) As Integer

Declare Function gp_rds Lib "gplib.dll" (ByVal Str1 As String) As Integer
Declare Function gp_rds1 Lib "gplib.dll" (ByVal Str1 As String) As Integer

Declare Function gp_delm Lib "gplib.dll" (ByVal Str1 As String, ByVal Val1 As Integer) As Integer
Declare Function gp_myadr Lib "gplib.dll" () As Integer
Declare Function gp_tmout Lib "gplib.dll" (ByVal Val1 As Integer) As Integer
Declare Function gp_trg Lib "gplib.dll" (ByVal Str1 As String) As Integer
Declare Function gp_wait Lib "gplib.dll" (ByVal Val1 As Integer) As Integer
Declare Function gp_wsrq Lib "gplib.dll" (ByVal Val1 As Integer) As Integer
Declare Function gp_wtb Lib "gplib.dll" (ByVal Str1 As String) As Integer

```

gp_init

REX-5052 を初期化

書式

VC ➤ int FAR PASCAL **gp_init**(int **port** , int **gpibid**);
port ➤ REX-5052 I/O Address
gpibid ➤ REX-5052 GPIB 機器 Address

VB ➤ Declare Function **gp_init** Lib "gplib.dll"
(ByVal **port** As Integer,ByVal **gpibid** As Integer) As Integer
port ➤ REX-5052 I/O Address
gpibid ➤ REX-5052 GPIB 機器 Address

関連

なし

実行例および動作

VC▼

```
int port;  
int gpibid;  
int gp_error;  
port=0x300;  
gpibid=0x00;  
gp_error=gp_init( port , gpibid );
```

VB▼

```
Global port As Integer ' GPIBカードI/Oベースアドレス  
Global gpibid As Integer ' GPIBカード機器アドレス  
gp_error = gp_init (port, gpibid)
```

REX-5052 カード上の GPIB コントローラチップにソフトウェアリセットコマンドを送り、GPIB コントローラチップを初期化し、マイアドレスをセットします。また、本ライブラリで使用するパラメータを初期化します。

REX-5052 が指定された I/O アドレスに存在していない場合には、戻り値として 91(10 進数)を返します。

gp_cli

IFC ラインを True にする

書式

VC ➤ int FAR PASCAL gp_cli(void);

VB ➤ Declare Function gp_cli Lib "gplib.dll" () As Integer

関連

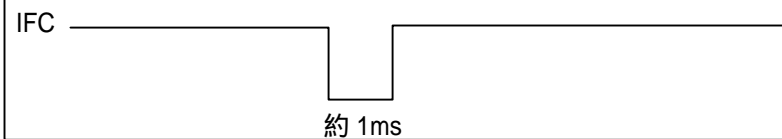
なし

実行例および動作 VC▼

```
int gp_error;
gp_error=gp_cli();
```

VB▼

```
Declare Function gp_cli Lib "gplib.dll" () As Integer
gp_error=gp_cli()
```



REX-5052 カード上の LSI 及び、 GPIB に接続されている全ての機器の初期化を行うために、プログラムの先頭部で必ず一度は IFC コマンドの実行が必要です。必ず正常終了します。

gp_ren

REN ラインを TRUE にする

書式 VC ➤ int FAR PASCAL APIENTRY gp_ren(void);
 VB ➤ Declare Function gp_ren Lib "gplib.dll" () As Integer

関連 なし


実行例および動作 VC▼

```
int gp_error;  
gp_error=gp_ren();
```

VB▼

```
Declare Function gp_ren Lib "gplib.dll" () As Integer  
gp_error=gp_ren()
```

```
REN _____
```



LCL コマンド(LCL コマンドの項 実行例1を参照)が実行されるか、またはPCがリセットされるまでずっと True のままです。 GPIB インターフェイスを持つ計測機器や装置は、REN ラインが True になるとリモート可能モードとなり、リモートモードを表示する LED などが点灯します。

REN ラインが False のままですと、GPIB 機器は正しく動作しませんので、プログラム先頭で必ず一度は REN コマンドの実行が必要です。

gp_clr

デバイスクリアまたはセレクトッドデバイスクリアコマンド送出

書式

VC > int FAR PASCAL gp_clr(char far*gp_adrs);
 gp_adrs > GPIB 機器アドレス

VB > Declare Function gp_clr Lib "gplib.dll"
 (ByVal gp_adrs As String) As Integer
 gp_adrs > GPIB 機器アドレス

関連

なし

実行例および動作

実行例 1. 全機器に対する場合

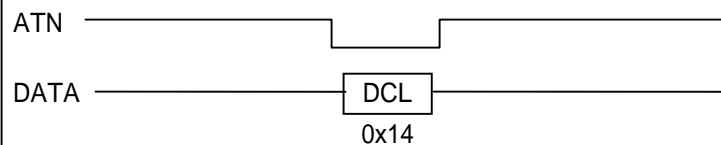
VC▼

```
char far *adrs;
int      ret_val;
adrs = "";
ret_val = gp_clr( adrs );
```

VB▼

Global UseGPIBAdrs As String * 12 'GPIB 機器アドレス

'何も代入していない文字列ですと先頭に NUL(0x00)が入っています。
 retval = gp_clr(Str(GPIBAdrs))



GPIB 上の全機器に対してクリアコマンドを送り、全機器をリセットします。

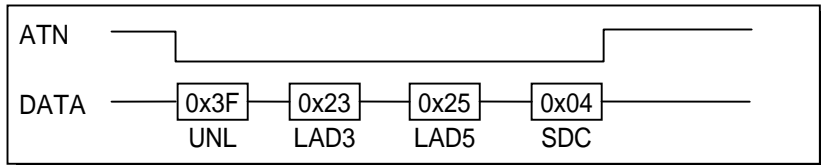
実行例 2. アドレス 3,5 の機器に対して、クリアコマンドを送る場合

VC▼

```
char far *adrs;
int      ret_val;
adrs = "3,5";
ret_val = gp_clr ( adrs );
```

VB▼

```
Global UseGPIBAdrs As String * 12 //GPIB 機器アドレス
GPIBAdrs="3,5" //GPIB 機器アドレスをセット
retval = gp_clr(GPIBAdrs)
```



相手側機器の DC (DEVICE CLEAR) 機能が DC0 の場合は、このコマンドは無効です。また DC2 の場合は、実行例 2 の SDC コマンドは無効となりますので、実行例 1 を御使用ください。

戻り値(10進数) 0 :正常終了
 53 :タイムアウト

gp_wrt

リスナアドレスで指定された機器にデータ送信

書式

VC > int FAR PASCAL gp_wrt
 (char far *adrs, char far *buf);
VB > Declare Function gp_wrt Lib "gplib.dll"
 (ByVal adrs As String, ByVal buf As String)
 As Integer

関連

タイムアウト、トーカモードデリミタ

実行例および動作

実行例 1. シングルリスナアドレスの場合
 (トーカモードデリミタ = 0)

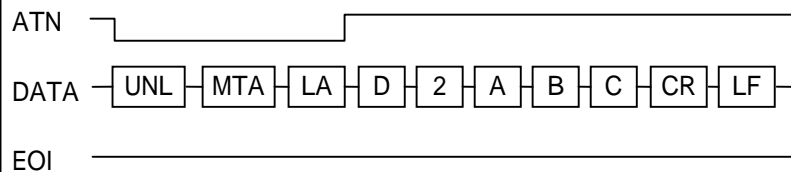
VC▼

```
char far *adrs;
char far *buf;
int      ret_val;
adrs = "3";
buf = "D2ABC";
ret_val = gp_wrt ( adrs , buf );
```

VB▼

```
Global GPIBAdrs As String * 12      ' GPIB機器アドレス
Global StrGCom As String * 12      ' GPIBコマンド
StrGCom = "D2ABC"
GPIBAdrs = "3"
retval = gp_wrt(GPIBAdrs, StrGCom)
```

アドレス 3 の機器に "D2ABC" という文字列を送信します。



実行例 2. マルチリスナアドレスの場合
(トーカモードデリミタ = 0x80)

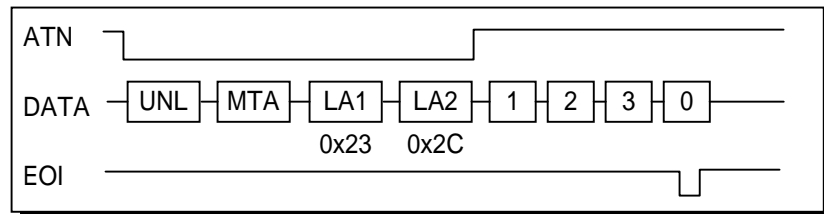
VC

```
char far *adrs;
char far *buf;
int ret_val;
adrs = "3";
buf = "1230";
ret_val = gp_wrt ( adrs, buf )
```

VB

```
Global GPIBAdrs As String * 12      ' GPIB 機器アドレス
Global StrGPCom As String * 12     ' GPIB コマンド
StrGPCom = "1230"
GPIBAdrs = "3,12"
retval = gp_wrt(GPIBAdrs, StrGPCom)
```

アドレス 3,12 の機器に文字列を送信します。



- 戻り値(10進数) 0 : 正常終了
 53 : タイムアウト

gp_red

指定したトーカーよりデータ受信しバッファに格納

書式

VC > int FAR PASCAL gp_red
 (char far *adrs, char far *buf);
VB > Declare Function gp_red Lib "gplib.dll"
 (ByVal adrs As String, ByVal buf As String)
 As Integer

関連

タイムアウト, リスナモードデリミタ

実行例および動作

実行例 1. 相手側機器の送信時デリミタがLFの場合

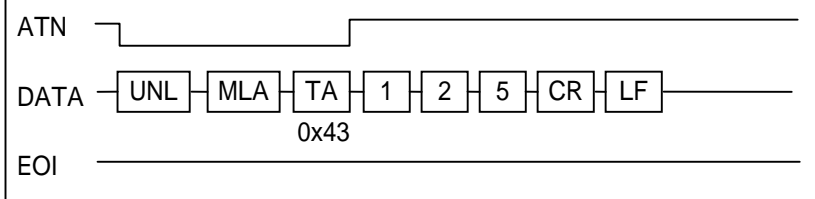
VC▼

```
char far buf[30];
char far *adrs;
int ret_val;
adrs = "3";
ret_val = gp_red( adrs , buf );
```

VB▼

```
Global GPIBAdrs As String * 12 ' GPIB 機器アドレス
Global Buf As String * 30 ' GPIB 受信バッファ
Buf = " " '必ず何らかの文字列をいれて初期化
GPIBAdrs = "3"
retval = gp_red(GPIBAdrs, buf)
```

アドレス3の機器よりデータを受信し、文字配列buf内に格納します。



HP社、横河電機、アドバンテスト等、ほとんどのメーカーが送信時デリミタとして CR,LF を使用していますので、リスナモードデリミタとしては 0x0a(LF)が一般的です。

実行例 2. リスナアドレス付の場合

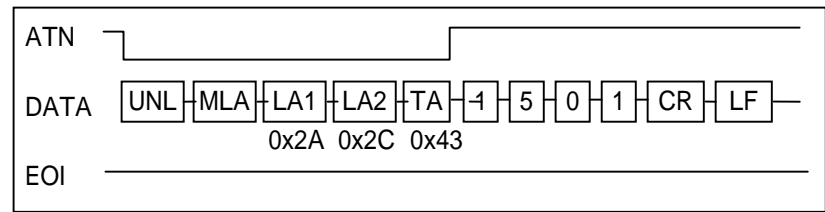
VC▼

```
char far buf[10];
char far *adrs="3,10,12";
int ret_val;
ret_val=gp_red( adrs, buf )
```

VB▼

```
Global GPIBAdrs As String * 12 ' GPIB機器アドレス
Global Buf As String * 30 ' GPIB受信バッファ
Buf = " " '必ず何らかの文字列で初期化
GPIBAdrs = "3,10,12"
retval = gp_red(GPIBAdrs, buf)
```

アドレス3の機器よりデータを受信し、文字配列buf内に格納します。同時にアドレス 10, 12 の機器にもデータが送られます。



(注意)

red コマンドは、相手側機器から出力される EOI を検出すると、その時点で読み込み動作を終了します。

gp_trg	リスナに指定された機器に対して GET 命令を送信
--------	---------------------------

書式

VC > int FAR PASCAL gp_trg(char far *adrs);
 VB > Declare Function gp_trg Lib "gplib.dll"
 (ByVal adrs As String) As Integer

関連 タイムアウト

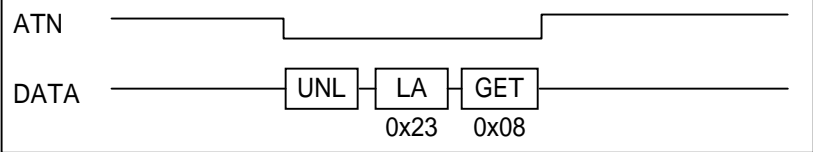
実行例および動作 VC▼

```
char far *adrs = "3";
int      ret_val;
ret_val = gp_trg ( adrs )
```

VB▼

```
Global GPIBAdrs As String * 12 ' GPIB 機器アドレス
GPIBAdrs = "3"
retval = gp_trg(GPIBAdrs)
```

アドレス3の機器に対して GET 命令を送信します。



gp_tfrin 指定したトーカーより指定バイト分データをバッファに格納

書式 **VC** > int FAR PASCAL gp_trg
(charfar *adrs, unsigned int bytc, charfar *buf);
VB > Declare Function gp_trg Lib "gplib.dll"
(ByVal adrs As String, ByVal bytc As Integer,
ByVal buf As String,) As Integer

関連 タイムアウト

- 実行例および動作
- 画像処理装置やFFTアナライザなどでは、一度に1～数Kbのデータを転送する機能を持っていますので、この tfrin を使用するとデータを一度に受信できます。
 - 受信バイト数がバッファ変数の長さよりも大きい場合は、バッファ変数分のデータだけ受け取ります。ただし受信動作は EOI が来るまで行い、バッファに入り切らない分は捨てられます。またその場合には戻り値として 61(BufferOverflow)を返します。
 - 受信バイト数の指定は、整数型変数または符号無し整数型変数で行ってください。

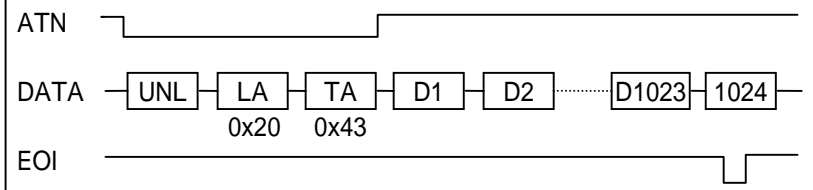
VC▼

```
char far buf[1025];
char far *adrs = "3";
unsigned int bytc = 1024;
int ret_val;
ret_val = gp_tfrin ( adrs, bytc, buf );
```

VB▼

```
Global GPIBAdrs As String * 12 ' GPIB 機器アドレス
Global Buf As String * 1025 ' GPIB 受信バッファ
bytc = 1024
GPIBAdrs = "3"
retval = gp_tfrin(GPIBAdrs, bytc, buf)
```

トーカーアドレス3の機器から 1024 バイトのデータをバッファ変数内に読み込みます。リスナ指定が無い場合は、REN ラインを False にし、GPIB 上の全機器をローカル状態に戻します。



gp_tfROUT

指定した機器へ指定バイト分のデータを転送

書式

VC > int FAR PASCAL gp_tfROUT
 (charfar *adrs, insigned int bytc, charfar *buf);
adrs > GPIB 機器アドレス
bytc > 送信バイトカウント
buf > 送信用配列領域

VB > Declare Function gp_tfROUT Lib "gplib.dll"
 (ByVal adrs As String,ByVal bytc As Integer,
 ByVal buf As String) As Integer
adrs > GPIB 機器アドレス
bytc > 送信バイトカウント
buf > 送信用配列領域

関連

タイムアウト

実行例および動作

- 画像処理装置やFFTアナライザなどへ一度に数KBのデータを送り込む場合にこの tfROUT コマンドを使用します。
- 送信時デリミタとして、EOI が送られます。
- 送信バイト数の指定は、整数型変数または符号無し整数型変数で行ってください。

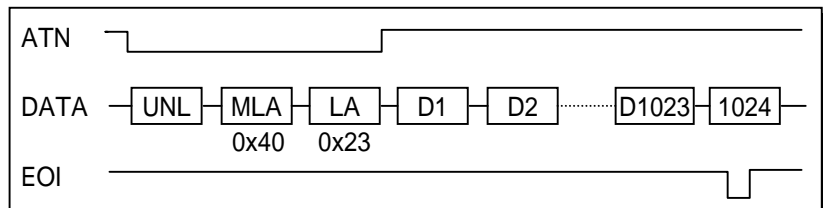
VC

```
char far buf[1025];
char far *adrs = "3";
unsigned int bytc;
int ret_val;
bytc = 1024;
ret_val= gp_tfROUT( adrs, bytc, buf );
```

VB

```
Global GPIBAdrs As String * 12 ' GPIB 機器アドレス
Global Buf As String * 1025 ' GPIB 受信バッファ
bytc = 1024
GPIBAdrs = "3"
retval = gp_tfROUT(GPIBAdrs, bytc, buf)
```

リスナアドレス 3 の機器へ 1024 バイトのデータを送信します。



gp_lcl **指定したリスナ機器をローカル状態に設定**

書式 **VC** > int FAR PASCAL gp_lcl(char far *adrs);
VB > Declare Function gp_lcl Lib "gplib.dll"
 (ByVal adrs As String) As Integer

関連 タイムアウト

実行例および動作 実行例 1. 全機器に対する場合

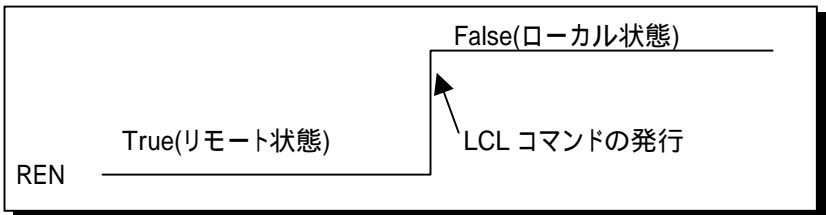
VC▼

```
char far *adrs;
int ret_val;
adrs = "";
ret_val=gp_lcl( adrs );
```

VB▼

```
Global UseGPIBAdrs As String * 12 ' GPIB 機器アドレス
retval = gp_lcl(Str(GPIBAdrs)) ' 初期化していない文字列ですと
                                ' 先頭に 00h が入っています。
```

GPIB 上の全機器をローカルモードにします。



実行例 2. リスナアドレスの指定がある場合

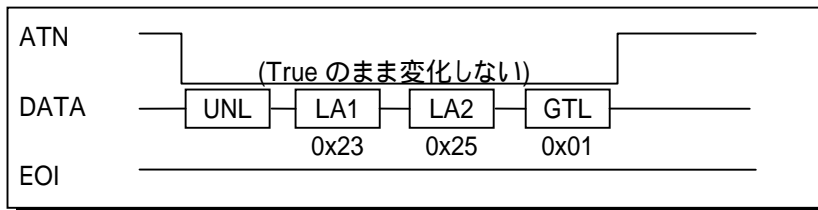
VC▼

```
char far *adrs;
int      ret_val;
adrs = "3,5";
ret_val=gp_lcl( adrs );
```

VB▼

```
Global UseGPIBAdrs As String * 12 'GPIB 機器アドレス
GPIBAdrs="3,5"                    'GPIB 機器アドレスをセット
retval = gp_lcl(GPIBAdrs)
```

リスナアドレス 3,5 の機器に GTL(go to local)命令を送りローカル状態に戻します。



gp_llo

GPIB 上の全機器のローカルスイッチを無効設定

書式 VC > int FAR PASCAL gp_llo(void);
 VB > Declare Function gp_llo Lib "gplib.dll" () As Integer

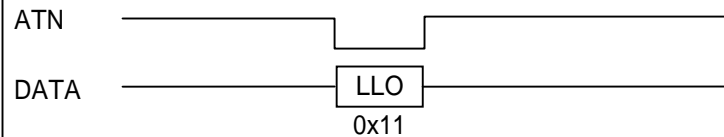
関連 なし

実行例および動作 VC▼

```
int gp_error;  
gp_error=gp_llo();
```

 VB▼

```
Declare Function gp_llo Lib "gplib.dll" () As Integer  
gp_error=gp_llo()
```



- ATN ラインを True にし、LLO 命令を送信した後 ATN ラインを False にします。この命令を受信すると機器側ではパネル上の操作スイッチを無効にします。ただし機器のリモート状態もしくはローカル状態には、変化は生じません。
- 機器の LLO 状態を解除する場合は REN ラインを False にします。(LCL コマンドの実行)

gp_wtb

ATN ラインを TRUE にしてコマンド文字列を送信

書式

VC > int FAR PASCAL gp_wtb(char far *buf);
 VB > Declare Function gp_wtb Lib "gplib.dll"
 (ByVal buf As String) As Integer

関連

タイムアウト

実行例および動作

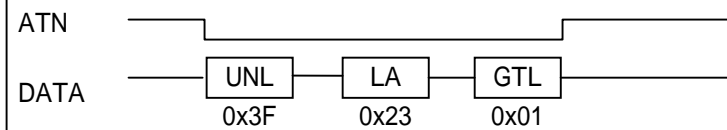
VC▼

```
char far buf[4];
buf[0] = 0x3f;
buf[1] = 0x23;
buf[2] = 0x01;
buf[3] = 0x00;
gp_wtb( buf );
```

VB▼

```
Gloval buf As String * 12
buf = chr$(3f)+chr$(23)+chr$(01)+chr$(0)
retval = gp_wtb(buf)
```

LCL3 の実行と同様になります。



gp_rds

シリアルポールを実行ステータスバイトを受信

書式

VC > int FAR PASCAL gp_rds

(char far *adrs, unsigned int far *status);

VB > Declare Function gp_rds Lib "gplib.dll"

(ByVal adrs As String, status As Integer) As Integer

関連

タイムアウト

実行例および動作

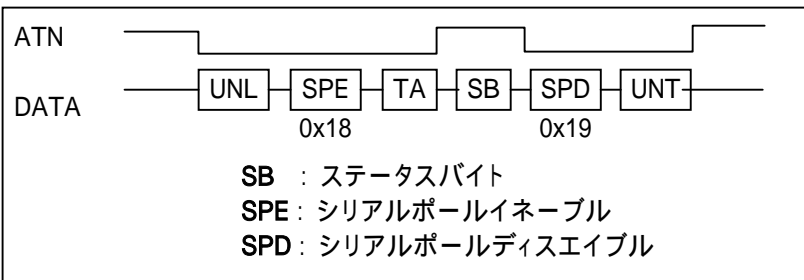
VC▼

```
char far *adrs = "3";
unsigned int far status;
int ret_val;
ret_val=gp_rds( adrs, &status );
```

VB▼

```
Global GPIBAdrs As String * 12      ' GPIB 機器アドレス
Global status As Integer            ' GPIB 機器ステータス
GPIBAdrs = "3"
retval = gp_rds(GPIBAdrs, status )
```

トーカーアドレス 3 の機器に対してシリアルポールを実行し、その機器のステータスバイトを読み込み変数 status に代入する。



SRQ を発信中の機器に対してこのコマンドを実行すると、SRQ ラインが False に復帰します。

gp_wait

指定した時間プログラムの実行を停止

書式

VC > int FAR PASCAL gp_wait(unsigned int tim);
VB > Declare Function gp_wait Lib "gplib.dll"
(ByVal tim As Integer) As Integer

関連

なし

実行例および動作

- 1time は約1秒です。
- 強制的にプログラムを停止させますのでマウスがきかなくなります。16bit 版からの互換性のために用意された関数です。

VC▼

```
unsigned int tim;  
int ret_val;  
tim = 10;  
ret_val=gp_wait( tim );
```

VB▼

```
Global tim As Integer      ' 待ち時間秒単位で指定  
tim = 10  
retval = gp_wait( tim )
```

10 秒間、プログラムの実行を停止します。

gp_wsrq

指定時間 SRQ を待つ

書式 VC > int FAR PASCAL gp_wsrq(unsigned int tim);
 VB > Declare Function gp_wsrq Lib "gplib.dll"
 (ByVal tim As Integer) As Integer

関連 なし

実行例および動作 ● 1time は1ミリ秒です。
 ● このコマンドによって SRQ ラインは変化しません。
 ● 時間内に SRQ がなければ-1 を返します

VC▼

```
unsigned int  tim;
int          ret_val;
tim = 10000;
ret_val=gp_wsrq( tim );
```

VB▼

```
Global tim As Integer        ' 待ち時間秒単位で指定
tim = 10000
retval = gp_wsrq( tim )
```

SRQ がくるまで 10 秒間待ちます。戻り値として 10 秒以内に SRQ があれば 0 を、なければ -1 を、返します。

gp_delm

リスナ時トーカー時のデリミタを設定

書式

VC > int FAR PASCAL gp_delm
 (char far *mode , unsigned int delm);
VB > Declare Function gp_delm Lib "gplib.dll"
 (ByVal mode As String, ByVal delm As Integer)
 As Integer

mode は "t", "l" のどれか一文字とし、次の意味を持ちます。

"t" : トーカー時の送信デリミタを指定します。

"l" : リスナ時の受信デリミタを指定します。

delm は 0 ~ 255 (0x00 ~ 0xff) の範囲の値で mode により次の意味をもちます。

"t" : デリミタコードは bit6 ~ bit0 の 7bit で設定します。

この時、bit7 を 1 にすると EOI を出力します。

delm = 0 とした場合は CR+LF が設定されます。

"l" : デリミタコードは bit7 ~ bit0 の 8bit で設定します。

変更されたデリミタは、次にこのコマンドによって変更されるまで有効です。デフォルト状態では、トーカーモードデリミタは 0 (CR+LF) に、リスナモードデリミタは 0x0a (LF) に設定されています。

関連

タイムアウト

実行例および動作

リスナモードデリミタとして LF を設定します。

VC ▽

```
char far *mode = "l";
unsigned int delm = 0x10;
int ret_val;
ret_val = gp_delm( mode, delm );
```

VB ▽

```
Global GPIBMode As String * 2          ' モード
Global delm As Integer                 ' デリミタ
GPIBMode = "l"
delm = &h0a
retval = gp_delm(GPIBMode, delm )
```

gp_tmout

バスタイムアウトパラメータを設定

書式

VC > int FAR PASCAL gp_tmout(unsigned int tim);
VB > Declare Function gp_tmout Lib "gplib.dll"
(ByVal tim As Integer) As Integer

関連

なし

実行例および動作

- 1time は 1 ミリ秒です。
- タイムアウトは 1 バイトのハンドシェイクに対し設定されます。
- デフォルト値は 10 秒です。
red/wrt 等のコマンド実行時のバスタイムアウトを 3 秒に設定します。

VC▼

```
unsigned int tim;  
int      ret_val;  
tim = 3000;  
ret_val=gp_tmout ( tim );
```

VB▼

```
Global tim As Integer      ' 待ち時間秒単位で指定  
tim = 3000  
retval = gp_tmout( tim )
```

gp_myadr

設定された GPIB マイアドレスの値をリード

書式 VC > int FAR PASCAL gp_myadr(void);
 VB > Declare Function gp_myadr Lib "gplib.dll" () As Integer

なし

関連

実行例および動作

REX-5052 のカード上の GPIB コントローラにセットされたアドレスの値を読み取り、変数 da に代入します。これにより、自分の GPIB 上の機器アドレスを知ることができます。
 互換性を確保する関数ですので、プログラムで新たに自分の機器アドレスを知る必要がない場合は実行する必要はありません。

VC▼

```
int da;
da=gp_myadr();
```

VB▼

```
da = gp_myadr()
```

◆関数の戻り値

エラーが発生した場合、各関数は戻り値としてエラーコードを返します。

エラーコード	意味
2	リクエストコードのフォーマットエラー
53	GPIB バスタイムアウトエラー
60	デバイスが使用状態にない
61	バッファオーバーフロー

(注意) 10進数で表記してあります。

(空白ページ)

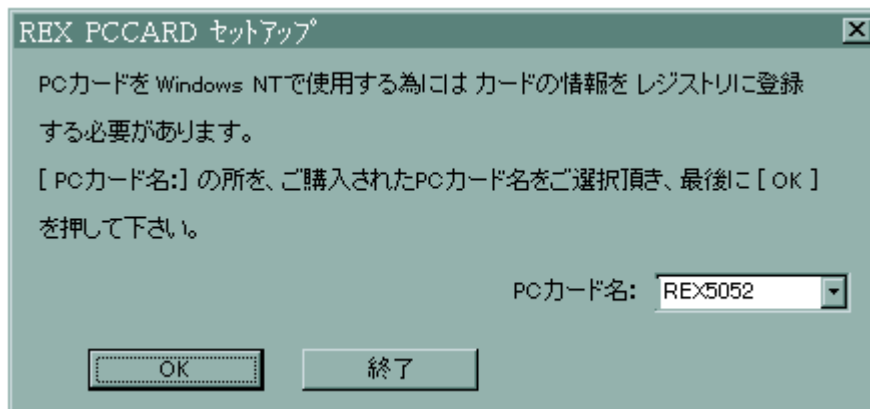
第6章 WindowsNT4.0 解説

(6-1) インストール

WindowsNT 上で REX-5052 を使用した計測システムを接続する場合は、本製品添付のセットアッププログラム (INISETUP.EXE) を実行します。

⇒ セットアッププログラムの実行

本製品に含まれるセットアッププログラム (INISETUP.EXE) を実行すると、下記のような画面が表示されます。使用されるカードを選択し [OK] を押してください。



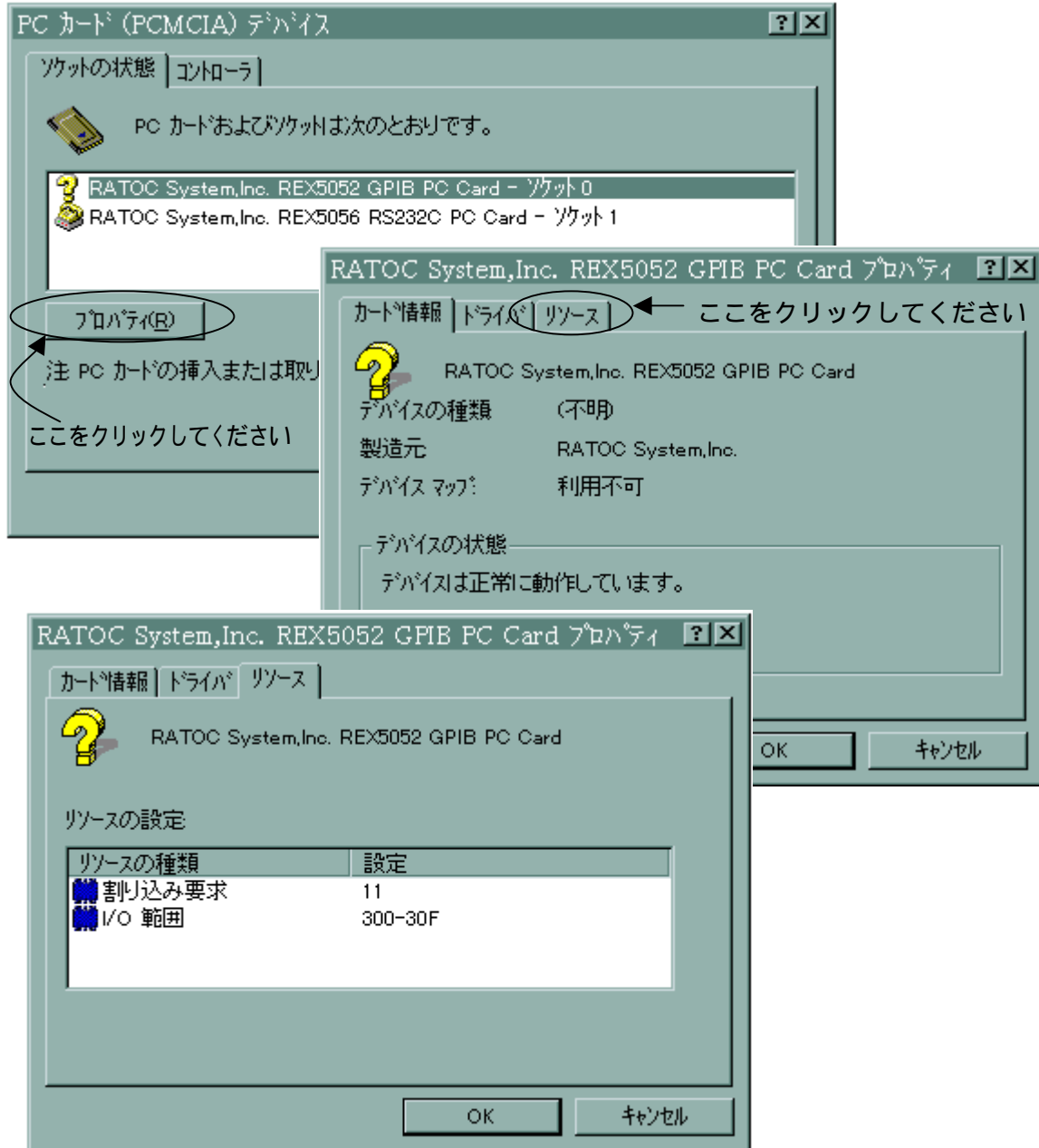
実行画面

このセットアッププログラムにより指定されたカードのドライバの登録と、ドライバ、DLL (ダイナミックライブラリ) のコピーが行われます。ドライバは %WinNT%\System32\Drivers へ、DLL は %WinNT%\System32 へコピーされます。

以上で設定完了です。WindowsNT を再起動し、REX5052 が使用可能な状態になったかどうかを確認します。

=> 再起動後の確認

再起動後リソースの取得ができていればセットアップは正常に行われています。(下図参照)



これらの画面はコントロールパネルの [PC カード] を選択することにより画面に表示されます。リソースが取得できていれば、セットアップ完了です。また、コントロールパネルの [デバイス] を起動することにより、Windows NT で現在使用できるドライバの一覧が表示されます。そこに REX-5052 のドライバも再起動により表示されます。

ドライバのロード、アンロード等の設定を行う場合はここで設定を行います。

(特に指定のない場合は何も行う必要はありません)

(6-2) DLL 関数仕様

サンプルプログラムから DLL でイクスポートされている関数を呼び出すためには、

1. DLL 関数をインポート宣言する

2. WindowsNT 用ライブラリ 5052lib.lib をプロジェクトに追加する

必要があります。

インポート宣言の方法については、サンプルプログラムヘッダーファイル

GpLib32.h (Declare.bas) を参照してください。

◆関数仕様の記述について

本ソフトウェアを動作させるための個々のコマンドについて解説を行います。汎例を下記に示します。書式及び実行例は Visual C と Visual Basic 両方を記述します。

gp_xxx (コマンド名)	機能
書式	VC > Visual C での関数の記述 VB > Visual BASIC での関数の記述
関連	実行時に関連のあるパラメータ
実行例および動作	そのコマンドの実行例と GPIB 各信号線の動作を示します。

留意点

- 戻り値は、0 の場合は正常終了です。それ以外はエラーコードです。
- 機器アドレスの指定は文字列で行ないます。(各コマンドの解説では書式の項目で"PSZ adrs"で示されています。)
このとき、トーカ指定が必要なコマンドでは、文字列の先頭の機器アドレスがトーカアドレスとなります。
(例)リスナアドレス 1,3,4,8 の場合 : adrs = "1,3,4,8"
全機器に対する場合 : adrs = "" (ヌル文字列)
- 引き数に関する注意
Visual Basic で 5052lib.DLL を呼び出す場合、値を渡す場合には、ByVal val1 As Long になります。アドレスを渡す場合には、Val1 As String という構文になります。

B 既に Windows95/98 でご使用のお客様へ B

WindowsNT のライブラリでは割り込みをご使用頂けるようになっているため
Windows95/98 で作成されたアプリケーションを使用するには多少の変更点が必要になります。DLL 関数の機能覧に (NT) マークのついている関数をご使用
の際には、引数が変わっておりますので、ご使用には注意が必要です。

◆関数一覧

関数	概要	頁
gp_init	REX-5052 を初期化 (NT)	6- 5
gp_cli	IFC ラインを TRUE にする	6- 6
gp_ren	REN ラインを TRUE にする	6- 7
gp_clr	デバイスクリアまたはセレクトッドデバイスクリアコマンド 送出	6- 8
gp_wrt	リスナアドレスで指定された機器にデータ送信	6-10
gp_red	指定したトーカよりデータを受信しバッファに格納 (NT)	6-12
gp_trg	リスナに指定された機器に対して GET 命令を送信	6-14
gp_tfrin	指定したトーカより指定バイト分データをバッファに格納	6-15
gp_tfROUT	指定した機器へ指定バイト分のデータを転送	6-16
gp_lcl	指定したリスナ機器をローカル状態に設定	6-17
gp_llo	GPIB 上の全機器のローカルスイッチを無効設定	6-19
gp_wtb	ATN ラインを TRUE にしてコマンド文字列を送信	6-20
gp_rds	シリアルポールを実行しステータスバイトを受信	6-21
gp_rds1	シリアルポールを実行しステータスバイトを受信	6-22
gp_wait	指定した時間プログラムの実行を停止	6-23
gp_wsrq	指定時間 SRQ を待つ	6-24
gp_delm	リスナ時トーカ時のデリミタを設定	6-25
gp_tmout	バスタイムアウトパラメータを設定	6-26
gp_myadr	設定された GPIB マイアドレスの値をリード	6-26
gp_srq	SRQ Interrupt Enable (NT)	6-27

gp_init

REX-5052 を初期化 (NT)

書式

VC > INT gp_init
 (INT port, INT gpibid, INT irq)
 port > (Windows95/98 互換用) "0"を指定します
 gpibid > REX-5052 GPIB 機器 Address
 irq > (Windows95/98 互換用) "0"を指定します

VB > Function gp_init
 (ByVal port As Long, ByVal gpibid As Long, ByVal irq
 As Long) As Long
 port > (Windows95/98 互換用) "0"を指定します
 gpibid > REX-5052 GPIB 機器 Address
 irq > (Windows95/98 互換用) "0"を指定します

関連

なし

実行例および動作

VC ▼

```
INT    gpibid;           // GPIB カード機器アドレス
INT    gp_error;
gpibid = 0x00;
gp_error = gp_init( 0 , gpibid , 0);
```

VB ▼

```
Global gpibid As Long    ' GPIB カード機器アドレス
gp_error = gp_init (0, gpibid, 0)
```

REX-5052 カード上の GPIB コントローラチップにソフトウェアリセットコマンドを送り、GPIB コントローラチップを初期化し、マイアドレスをセットします。また、本ライブラリで使用するパラメータを初期化します。REX-5052 がコンフィグレーションに失敗している場合には、戻り値として 60 (10 進数) を返します。

gp_cli

IFC ラインを TRUE にする

書式 VC > INT gp_cli(void)
VB > Function gp_cli() As Long

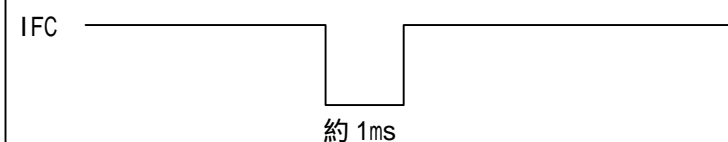
関連 なし

実行例および動作 VC ▾

```
INT    gp_error;  
gp_error = gp_cli();
```

VB ▾

```
Dim gp_error as Long  
gp_error = gp_cli()
```



REX-5052 カード上の LSI 及び、 GPIB に接続されている全ての機器の初期化を行うために、プログラムの先頭部で必ず一度は IFC コマンドの実行が必要です。必ず正常終了します。

gp_ren

REN ラインを TRUE にする

書式 VC ➤ INT gp_ren(void)
VB ➤ Function gp_ren() As Long

関連 なし

実行例および動作 VC ▼

```
INT gp_error;  
gp_error = gp_ren();
```

VB ▼

```
Dim gp_error as Long  
gp_error = gp_ren()
```

REN _____ REN コマンドの発行

LCL コマンド (LCL コマンドの項 実行例 1 を参照) が実行されるか、または PC がリセットされるまでずっと True のままです。 GPIB インターフェイスを持つ計測機器や装置は、REN ラインが True になるとリモート可能モードとなり、リモートモードを表示する LED などが点灯します。 REN ラインが False のままですと、GPIB 機器は正しく動作しませんので、プログラム先頭で必ず一度は REN コマンドの実行が必要です。

実行例 2. アドレス 3,5 の機器に対して、クリアコマンドを送る場合

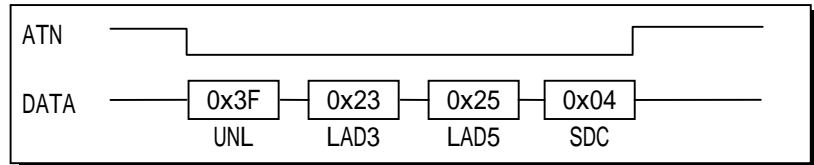
VC ▼

```
char    *adrs = "3,5";           // GPIB 機器アドレス
INT     ret_val;

ret_val = gp_clr ( adrs );
```

VB ▼

```
Global Adrs As String * 12      ' GPIB 機器アドレス
Adrs="3,5"                       ' GPIB 機器アドレスをセット
retval = gp_clr(GPIBAdrs)
```



相手側機器の DC (DEVICE CLEAR) 機能が DC0 の場合は、このコマンドは無効です。また DC2 の場合は、実行例 2 の SDC コマンドは無効となりますので、実行例 1 を御使用ください。

戻り値 (10 進数)

0 : 正常終了
53 : タイムアウト

gp_wrt**リスナアドレスで指定された機器にデータ送信**

書式

VC ➤ INT gp_wrt(PSZ adrs, PSZ buf)

VB ➤ Function gp_wrt

(ByVal adrs As String, ByVal buf As String) As Long

関連

タイムアウト, トーカモードデリミタ

実行例および動作

実行例 1. シングルリスナアドレスの場合
(トーカモードデリミタ=0)

VC ▼

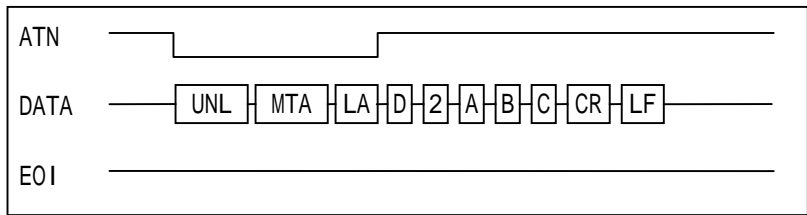
```
char    *adrs = "3";           // GPIB 機器アドレス
char    buf[20];
INT     ret_val;

strcpy(buf, "D2ABC");
ret_val = gp_wrt ( adrs , buf );
```

VB ▼

```
Global GPIBAdrs As String * 12   ' GPIB 機器アドレス
Global StrGPCom As String * 12   ' GPIB コマンド
StrGPCom = "D2ABC"
GPIBAdrs = "3"
retval = gp_wrt(GPIBAdrs, StrGPCom)
```

アドレス 3 の機器に "D2ABC" という文字列を送信します。



実行例 2. マルチリスナアドレスの場合
(トーカモードデリミタ = 0x80)

VC ▼

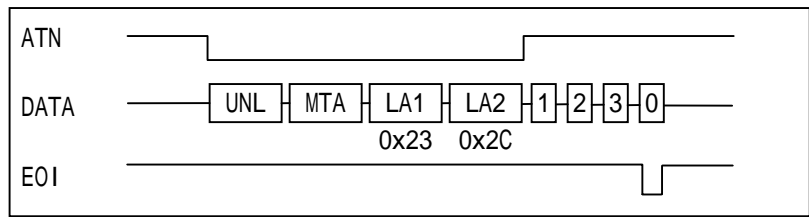
```
char *adrs = "3,12";           // GPIB 機器アドレス
char buf[12];
INT  ret_val;

strcpy(buf, "1230");
ret_val = gp_wrt ( adrs, buf );
```

VB ▼

```
Global GPIBAdrs As String * 12   ' GPIB 機器アドレス
Global StrGPCom As String * 12   ' GPIB コマンド
StrGPCom = "1230"
GPIBAdrs = "3,12"
retval = gp_wrt(GPIBAdrs, StrGPCom)
```

アドレス 3,12 の機器に文字列を送信します。



戻り値 (10 進数) 0 : 正常終了
 53 : タイムアウト

gp_red

指定したトーカーよりデータ受信しバッファに格納 (NT)

書式

VC > INT gp_red(PSZ adrs, PSZ buf , size_t size)

VB > Function gp_red

(ByVal adrs As String, ByVal buf As String,
ByVal size As Long) As Long

注)バッファサイズは受信するバイト数より必ず 1 バイト以上多く取ってください。

関連

タイムアウト, リスナモードデリミタ

実行例および動作

実行例 1. 相手側機器の送信時デリミタが LF の場合

VC ▼

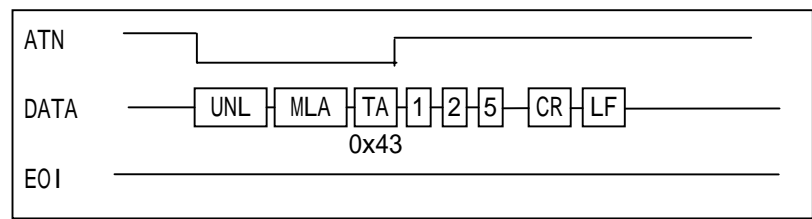
```
char    buf[30];                // GPIB 受信バッファ
char    *adrs = "3";           // GPIB 機器アドレス
INT     ret_val;

gp_red( adrs , buf ,30);
```

VB ▼

```
Global GPIBAdrs As String * 12   ' GPIB 機器アドレス
Global Buf As String * 30        ' GPIB 受信バッファ
GPIBAdrs = "3"
retval = gp_red(GPIBAdrs, buf, 30)
```

アドレス 3 の機器よりデータを受信し、文字配列 buf 内に格納します。



HP 社、横河電機、アドバンテスト等、ほとんどのメーカーが送信時デリミタとして CR, LF を使用していますので、リスナモードデリミタとしては 0x0a(LF)が一般的です。

実行例 2. リスナアドレス付の場合

VC ▼

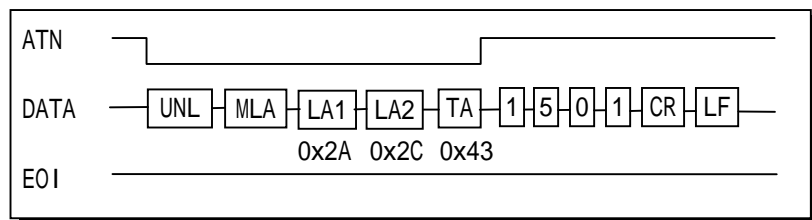
```
char    buf[10];                // GPIB 受信バッファ
char    *adrs="3,10,12";       // GPIB 機器アドレス
INT     ret_val;
ret_val = gp_red( adrs, buf ,10)
```

VB ▼

```
Global GPIBAdrs As String * 12  ' GPIB 機器アドレス
Global Buf As String * 30      ' GPIB 受信バッファ

GPIBAdrs = "3,10,12"
retval = gp_red(GPIBAdrs, buf, 30)
```

アドレス 3 の機器よりデータを受信し、文字配列 buf 内に格納します。同時にアドレス 10,12 の機器にもデータが送られます。



(注意)

red コマンドは、相手側機器から出力される EOI を検出すると、その時点で読み込み動作を終了します。

gp_trg**リスナに指定された機器に対して GET 命令を送信**

書式 VC > INT gp_trg(PSZ adrs)
 VB > Function gp_trg
 (ByVal adrs As String) As Long

関連 タイムアウト

実行例および動作

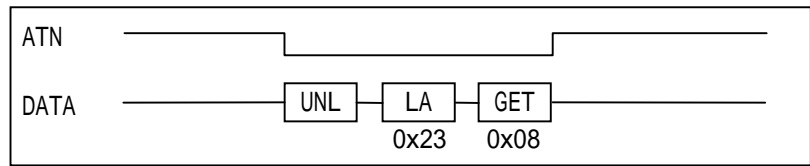
VC ▼

```
char    *adrs = "3";           // GPIB 機器アドレス
INT     ret_val;
ret_val = gp_trg ( adrs )
```

VB ▼

```
Global GPIBAdrs As String * 12 ' GPIB 機器アドレス
GPIBAdrs = "3"
retval = gp_trg(GPIBAdrs)
```

アドレス 3 の機器に対して GET 命令を送信します。



gp_tfrin

指定したトーカより指定バイト分データをバッファに格納

書式

VC > INT gp_tfrin(PSZ adrs, INT bytc, PSZ buf)

adrs > GPIB 機器アドレス

bytc > 受信バイトカウント

buf > 受信用配列領域

VB > Function gp_tfrin
(ByVal adrs As String, ByVal bytc As Long
ByVal buf As String) As Long

adrs > GPIB 機器アドレス

bytc > 受信バイトカウント

buf > 受信用配列領域

関連

タイムアウト

実行例および動作

- 画像処理装置や FFT アナライザなどでは、一度に 1 ~ 数 Kb のデータを転送する機能を持っていますので、この tfrin を使用するとデータを 1 度に受信できます。
- 受信バイト数がバッファ変数の長さよりも大きい場合は、バッファ変数分のデータだけ受け取ります。但し受信動作は EOI が来るまで行い、バッファに入り切らない分は捨てられます。またその場合には戻り値として 61(BufferOverflow) を返します。
- 受信バイト数の指定は、整数型変数または符号無し整数型変数で行ってください。

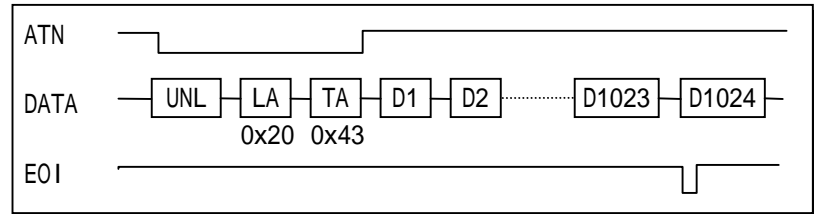
VC ▼

```
char    buf[1025];           // GPIB 受信バッファ
char    *adrs = "3";        // GPIB 機器アドレス
INT     bytc = 1024;
INT     ret_val;
ret_val = gp_tfrin ( adrs, bytc, buf );
```

VB ▼

```
Global GPIBAdrs As String * 12    ' GPIB 機器アドレス
Global Buf As String * 1025       ' GPIB 受信バッファ
bytc = 1024
GPIBAdrs = "3"
retval = gp_tfrin(GPIBAdrs, bytc, buf)
```

トーカーアドレス 3 の機器から 1024 バイトのデータをバッファ変数内に読み込みます。リスナ指定が無い場合は、REN ラインを False にし、 GPIB 上の全機器をローカル状態に戻します。

**gp_tfROUT****指定した機器へ指定バイト分のデータを転送**

書式

VC > int gp_tfROUT(PSZ adrs, INT bytc, PSZ buf)

- adrs > GPIB 機器アドレス
- bytc > 送信バイトカウント
- buf > 送信用配列領域

VB > Function gp_tfROUT

(ByVal adrs As String, ByVal bytc As Long

ByVal buf As String) As Long

- adrs > GPIB 機器アドレス
- bytc > 送信バイトカウント
- buf > 送信用配列領域

関連

タイムアウト

実行例および動作

- 画像処理装置や FFT アナライザなどへ一度に数 KB のデータを送り込む場合にこの tfROUT コマンドを使用します。
- 送信時デリミタとして、EOI が送られます。
- 送信バイト数の指定は、整数型変数または符号無し整数型変数で行ってください。

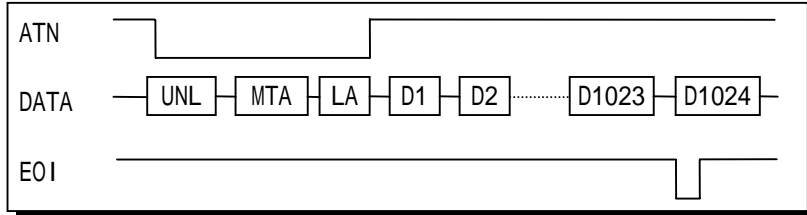
VC ▼

```
char    buf[1025];           // GPIB 受信バッファ
char    *adrs = "3";        // GPIB 機器アドレス
INT     bytc;
INT     ret_val;
bytc = 1024;
ret_val= gp_tfROUT( adrs, bytc, buf );
```

VB ▼

```
Global GPIBAdrs As String * 12 ' GPIB 機器アドレス
Global Buf As String * 1025 ' GPIB 受信バッファ
bytc = 1024
GPIBAdrs = "3"
retval = gp_tfrcout(GPIBAdrs, bytc, buf)
```

リスナアドレス 3 の機器へ 1024 バイトのデータを送信します。



gp_lcl 指定したリスナ機器をローカル状態に設定

書式

```
VC > int gp_lcl( PCHAR adrs )
VB > Function gp_lcl
    (ByVal adrs As String) As Long
```

関連

タイムアウト

実行例および動作

実行例 1. 全機器に対する場合

VC ▼

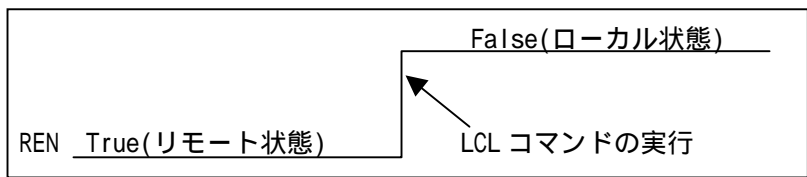
```
char *adrs = ""; // GPIB 機器アドレス
INT ret_val;

ret_val = gp_lcl( adrs );
```

VB ▼

```
Global UseGPIBAdrs As String * 12 ' GPIB 機器アドレス
retval = gp_lcl(Str(GPIBAdrs)) ' 初期化していない文字列ですと
    ' 先頭に 00h が入っています。
```

GPIB 上の全機器をローカルモードにします。



実行例 2. リスナアドレスの指定がある場合

VC ▼

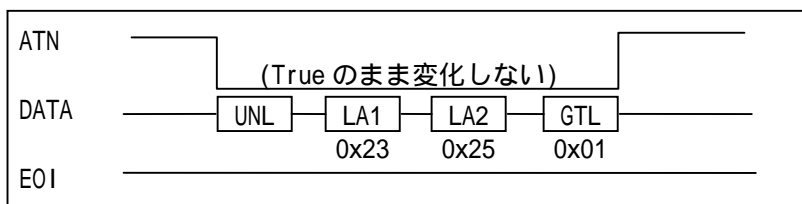
```
char    *adrs = "3,5";           // GPIB 機器アドレス
INT     ret_val;

ret_val = gp_lcl( adrs );
```

VB ▼

```
Global UseGPIBAdrs As String * 12 'GPIB 機器アドレス
GPIBAdrs="3,5"                    'GPIB 機器アドレスをセット
retval = gp_lcl(GPIBAdrs)
```

リスナアドレス 3,5 の機器に G T L (go to local)命令を送りローカル状態に戻します。



gp_ll0

GPIB 上の全機器のローカルスイッチを無効設定

書式 VC ➤ INT gp_ll0(void)
 VB ➤ Function gp_ll0() As Long

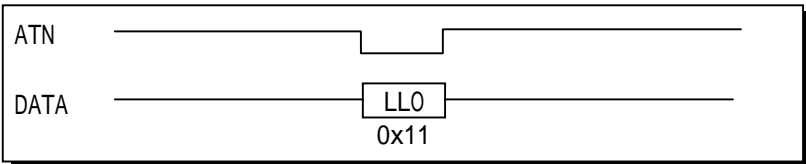
関連 なし

実行例および動作 VC ▼

```
INT gp_error;
gp_error=gp_ll0();
```

VB ▼

```
Dim gp_error As Long
gp_error=gp_ll0()
```



- ATN ラインを True にし、LL0 命令を送信した後 ATN ラインを False にします。この命令を受信すると機器側ではパネル上の操作スイッチを無効にします。ただし機器のリモート状態もしくはローカル状態には、変化は生じません。
- 機器の LL0 状態を解除する場合は REN ラインを False にします。(LCL コマンドの実行)

gp_wtb**ATN ラインを TRUE にしてコマンド文字列を送信**

書式

VC > INT gp_wtb(PSZ buf)

VB > Function gp_wtb
(ByVal buf As String) As Long

関連

なし

実行例および動作

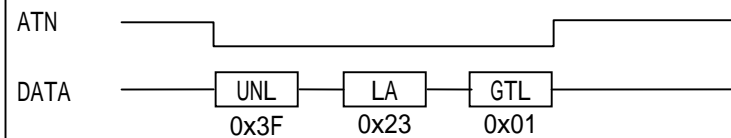
VC ▼

```
char buf[20];
buf[0]=0x3f;
buf[1]=0x23;
buf[2]=0x01;
buf[3]=0x00;
gp_wtb( buf );
```

VB ▼

```
Gloval buf As String * 12
buf = chr(&h3f)+chr(&h23)+chr(&h01) +chr(&h00)
retval = gp_wtb(buf)
```

LCL3 の実行と同様になります。



gp_rds

シリアルポールを実行しステータスバイトを受信

書式 VC > INT gp_rds(PSZ adrs, PUINT status)
 VB > Function gp_rds
 (ByVal adrs As String, status As Long) As Long

関連 タイムアウト

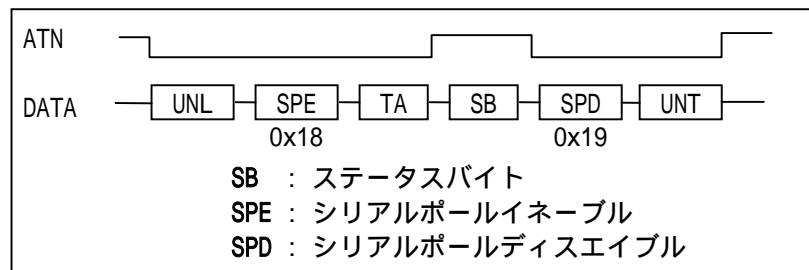
実行例および動作 VC ▼

```
char *adrs = "3";           // GPIB 機器アドレス
INT      status;           // GPIB 機器ステータス
INT      ret_val;
ret_val = gp_rds( adrs,&status );
```

VB ▼

```
Global GPIBAdrs As String * 12 ' GPIB 機器アドレス
Global status As Long          ' GPIB 機器ステータス
GPIBAdrs = "3"
retval = gp_rds(GPIBAdrs, status )
```

トーカーアドレス 3 の機器に対してシリアルポールを実行し、その機器のステータスバイトを読み込み変数 status に代入する。



SRQ を発信中の機器に対してこのコマンドを実行すると、SRQ ラインが False に復帰します。

gp_rds1

シリアルポールを実行しステータスバイトを受信

(注意) gp_rds との違いは、最後に UNT コマンドを送出しない点です。

書式

VC > INT gp_rds1(PSZ adrs, PUINT status)

VB > Function gp_rds1

(ByVal adrs As String, status As Long) As Long

関連

タイムアウト

実行例および動作

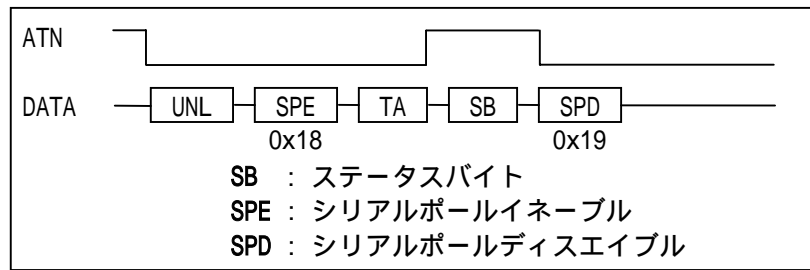
VC ▼

```
char *adrs = "3";           // GPIB 機器アドレス
INT status;                // GPIB 機器ステータス
INT ret_val;
ret_val = gp_rds1( adrs,&status );
```

VB ▼

```
Global GPIBAdrs As String * 12 ' GPIB 機器アドレス
Global status As Long          ' GPIB 機器ステータス
GPIBAdrs = "3"
retval = gp_rds1(GPIBAdrs, status )
```

トーカーアドレス 3 の機器に対してシリアルポールを実行し、その機器のステータスバイトを読み込み変数 status に代入する。



SRQ を発信中の機器に対してこのコマンドを実行すると、SRQ ラインが False に復帰します。

gp_wait

指定した時間プログラムの実行を停止

書式

VC > INT gp_wait(INT tim)
VB > Function gp_wait
(ByVal tim As Long) As Long

関連

なし

実行例および動作

- 1 time は約 1 秒です。
- 強制的にプログラムを停止させますのでマウスがきかなくなります。16bit 版からの互換性のために用意された関数です。

VC ▼

```
INT tim = 10;           // 待ち時間秒単位で指定
INT ret_val;

ret_val = gp_wait( tim );
```

VB ▼

```
Global tim As Long     ' 待ち時間秒単位で指定
tim = 10
retval = gp_wait( tim )
```

10 秒間、プログラムの実行を停止します。

gp_wsrq

指定時間 SRQ を待つ

書式

VC > INT gp_wsrq(INT tim)

VB > Function gp_wsrq
(ByVal tim As Long) As Long

関連

なし

実行例および動作

- 1 time は 1 ミリ秒です。
- このコマンドによって SRQ ラインは変化しません。
- 時間内に SRQ がなければ -1 を返します

VC ▼

```
INT    tim = 10000;    // 待ち時間秒単位で指定
INT    ret_val;

ret_val = gp_wsrq( tim );
```

VB ▼

```
Global tim As Long      ' 待ち時間秒単位で指定
tim = 10000
retval = gp_wsrq( tim )
```

SRQ がくるまで 10 秒間待ちます。戻り値として 10 秒以内に SRQ があれば 0 を、なければ -1 を、返します。

gp_delm

リスナ時トーカー時のデリミタを設定

書式

VC > INT gp_delm(PSZ mode , UINT delm)
 VB > Function gp_delm
 (ByVal mode As String, ByVal delm As Long)
 As Long

関連

タイムアウト

実行例および動作

mode は "t", "l" のどちらか一文字とし、次の意味を持ちます。
 "t" : トーカー時の送信デリミタを指定します。
 "l" : リスナ時の受信デリミタを指定します。
 delm は 0 ~ 255 (0x00 ~ 0xff) の範囲の値で mode により次の意味をもちます。
 "t" : デリミタコードは bit6 ~ bit0 の 7bit で設定します。
 この時、bit7 を 1 にすると EOI を出力します。
 delm = 0 とした場合は CR+LF が設定されます。
 "l" : デリミタコードは bit7 ~ bit0 の 8bit で設定します。
 変更されたデリミタは、次にこのコマンドによって変更されるまで有効です。
 デフォルト状態では、トーカーモードデリミタは 0 (CR+LF) に、リスナモードデリミタは 0x0a (LF) に設定されています。

リスナモードデリミタとして LF を設定します。

VC ▾

```
char *mode = "l";           // モード
UINT delm = 0x0a;          // デリミタ
INT ret_val;
ret_val = gp_delm( mode, delm);
```

VB ▾

```
Global GPIBMode As String * 2 ' モード
Global delm As Long           ' デリミタ
GPIBMode = "l"
delm = &h0a
retval = gp_delm(GPIBMode, delm )
```


gp_tmout**バスタイムアウトパラメータを設定**

書式

VC > INT gp_tmout(INT tim)
 VB > Function gp_tmout
 (ByVal tim As Long) As Long

関連

なし

実行例および動作

- 1time は 1 ミリ秒です。
- タイムアウトは 1 バイトのハンドシェイクに対し設定されます。
- デフォルト値は 10 秒です。
red/wrt 等のコマンド実行時のバスタイムアウトを 3 秒に設定します。

VC ▼

```
INT    tim = 3000;      // 待ち時間ミリ秒単位で指定
INT    ret_val;

ret_val=gp_tmout ( tim );
```

VB ▼

```
Global tim As Long      ' 待ち時間ミリ秒単位で指定
tim = 3000
retval = gp_tmout( tim )
```

gp_myadr**設定された GPIB マイアドレスの値をリード**

書式

VC > INT gp_myadr(void)
 VB > Function gp_myadr() As Long

関連

なし

実行例および動作

互換性を確保する関数ですので、プログラムで新たに自分の機器アドレスを知る必要がない場合は実行する必要はありません。

VC ▼

```
INT    da;
da = gp_myadr();
```

VB ▼

```
da = gp_myadr()
```

gp_srq

SRQ Interrupt Enable (NT)

書式	VC > INT gp_srq(HWND, INT) VB > Function gp_srq (ByVal Val1 As Long, ByVal Val2 As Long) As Long
関連	なし
実行例および動作	コントローラとして機器からの SRQ 割り込み受け付けの許可不許可を設定します。 詳しくはサンプルプログラムを参照してください。

◆関数の戻り値

エラーが発生した場合、各関数は戻り値としてエラーコードを返します。

エラーコード	意味
2	リクエストコードのフォーマットエラー
53	GPIBバスタイムアウトエラー
60	デバイスが使用状態にない
61	バッファオーバーフロー

(注意) 10進数で表記してあります。

(6-3) サンプルプログラム解説

本製品には

ヒューレットパカード社製 HP3478A (マルチメーター)

YOKOGAWA 社製 WT110E (デジタルパワーメーター)

を使用したサンプルプログラムが添付しております。

HP3478A サンプルプログラム解説 (Visual Basic , Visual C , MFC)

HP3478A のサンプルプログラムは、2つのモジュールで構成されています。

- ・ 割り込みを使用せずに SRQ が来るのをポーリングしデータを取得するプログラム
- ・ SRQ の検知に割り込みを使用し、データを取得するプログラム。

SRQ 制御を御使用される際のサンプルとして御活用ください。



SRQ 割り込みサンプルプログラム (画面は VC 版)



SRQ ポーリングサンプルプログラム (画面は VC 版)

WT110E サンプルプログラム解説 (MFC) Windows95 WindowsNT

- WT110E のサンプルプログラムは、2つのモジュールで構成されています。
- ・ 電流ピーク値を取得し最小ピーク電流、最大ピーク電流を取得するプログラム。
 - ・ WT110E の設定などを行うプログラム。

MFC を使用したアプリケーションを開発する際のサンプルとしてご利用ください



サンプルプログラム実行画面

(6-4) 割り込み制御の使用方法

ユーザー定義メッセージによる割り込みプログラム

gp_srq を実行することにより、割り込みを使用可能にすると、SRQ が起きた際に割り込み発生に同期したユーザー定義メッセージをアプリケーション側に送ります。Visual C、MFC の場合、ユーザー定義メッセージ用の処理をメッセージループ内に記述するといった形式で実現することができますが、Visual Basic ではユーザー定義メッセージを取得することができないため、本製品に添付されている OLE カスタムコントロール[MBOX]を使用することでユーザー定義メッセージを取得し、割り込み処理を実現します。

Visual C の場合

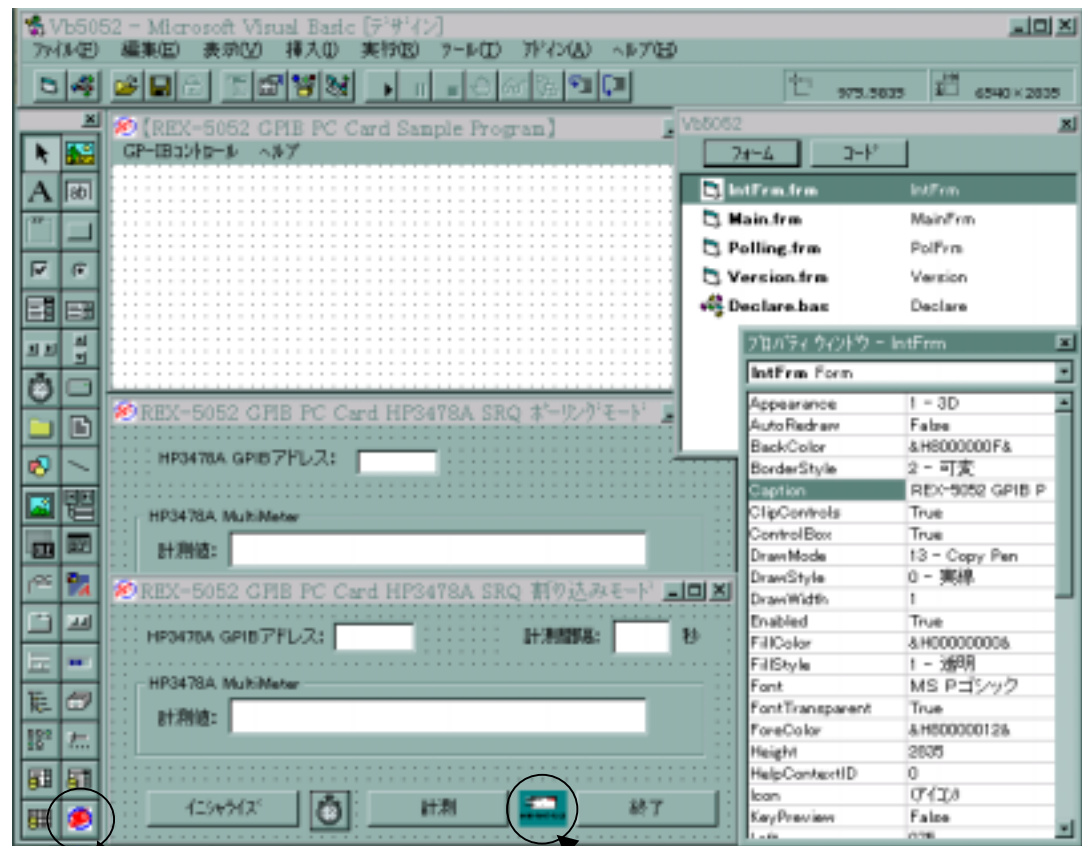
```
LRESULT CALLBACK DlgProcInt( HWND hDlg,.....
{
    switch( message ){
    case WM_INITDIALOG:
        return TRUE ;
    case WM_INTERRUPT_5052: // 割り込み発生!!
        gp_srq( hDlg, SRQ_DISABLE );
        return TRUE;
    case WM_COMMAND:
        switch( wParam ){
        case IDOK: // 割り込み開始
            gp_srq( hDlg, SRQ_ENABLE );
            gp_trg( HP3478GPIBAdrs );
            return TRUE ;
        default:
            return TRUE ;
        }
        break ;
    }
    return FALSE ;
}
```

MFC の場合

```
BEGIN_MESSAGE_MAP(CIntDlg, CDialog)
//{{AFX_MSG_MAP(CIntDlg)
ON_BN_CLICKED(IDC_INIT_BT_M, OnInitBtm)
//}}AFX_MSG_MAP
ON_MESSAGE(WM_INTERRUPT, IntMsg)
END_MESSAGE_MAP()

LRESULT Cmycls::IntMsg(WPARAM wParam, .....
{
    //割り込み発生!!
}
```

Visual Basic の場合



カスタムコントロール「MBOX OLE Control module」を追加
添付されている OLE カスタムコントロール「MBOX」を追加

割り込み処理記述



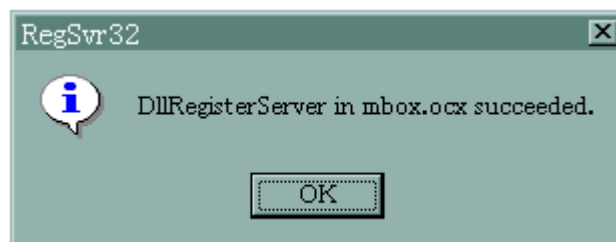
次に、OLE カスタムコントロール [MBOX] 使用するためのレジストリ登録方法について説明いたします。

OLE カスタムコントロール [MBOX] の登録 (Visual Basic 使用時)

本製品添付の OCX “MBOX.OCX”を Visual Basic で使用するためには、Visual Basic の CD-ROM に添付されているツール“REGSVR32.EXE”を使って OCX のレジストリ登録を行います。“REGSVR32.EXE”は 32 ビットコンソールアプリケーションですので、WindowsNT の DOS プロンプトから実行します。

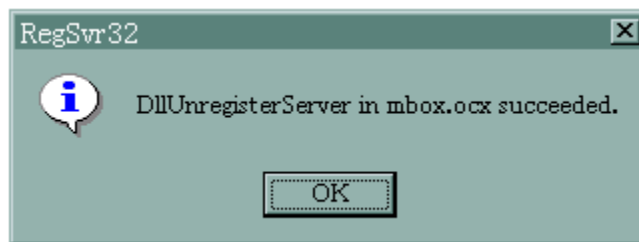
尚、“REGSVR32.EXE”は VB の CD-ROM に添付されています。OCX をレジストリ登録するときは、下記構文を実行します。

```
>REGSVR32 “ドライブ名”:%WinNT%System32%Mbox.ocx
```



OCX をレジストリ登録から削除するときは、“/U”を付けて下記構文を実行します。

```
>REGSVR32 /U “ドライブ名”:% WinNT%System32%Mbox.ocx
```

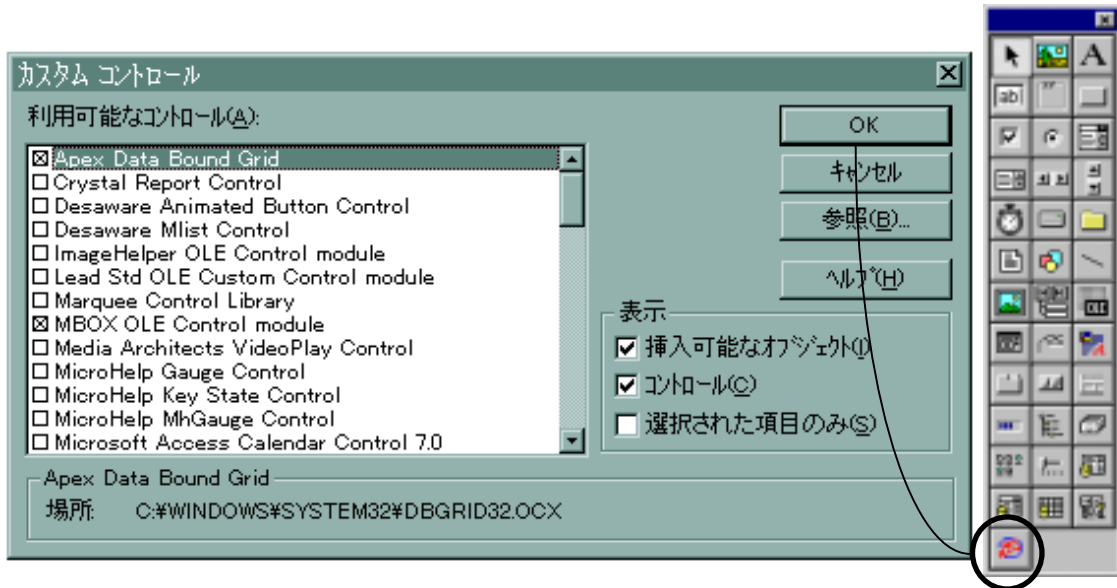


登録削除成功メッセージ

OLE カスタムコントロール [MBOX] の登録が終了しましたら、Visual Basic よりカスタムコントロールの追加を行う必要があります。次に、OLE カスタムコントロール [MBOX] の追加方法について説明いたします。

OLE カスタムコントロール [MBOX] の追加 (Visual Basic 使用時)

Visual Basic のカスタムコントロールに [MBOX] を追加します。
Visual Basic デザインメニューの「ツール」の「カスタムコントロール」
を起動し、利用可能なコントロールから「MBOX OLE Control module」
をチェックすれば Visual Basic ツールバーに MBOX が追加されます。



以上で、Visual Basic で割り込み処理を使用するための設定は終了です。

(空白ページ)

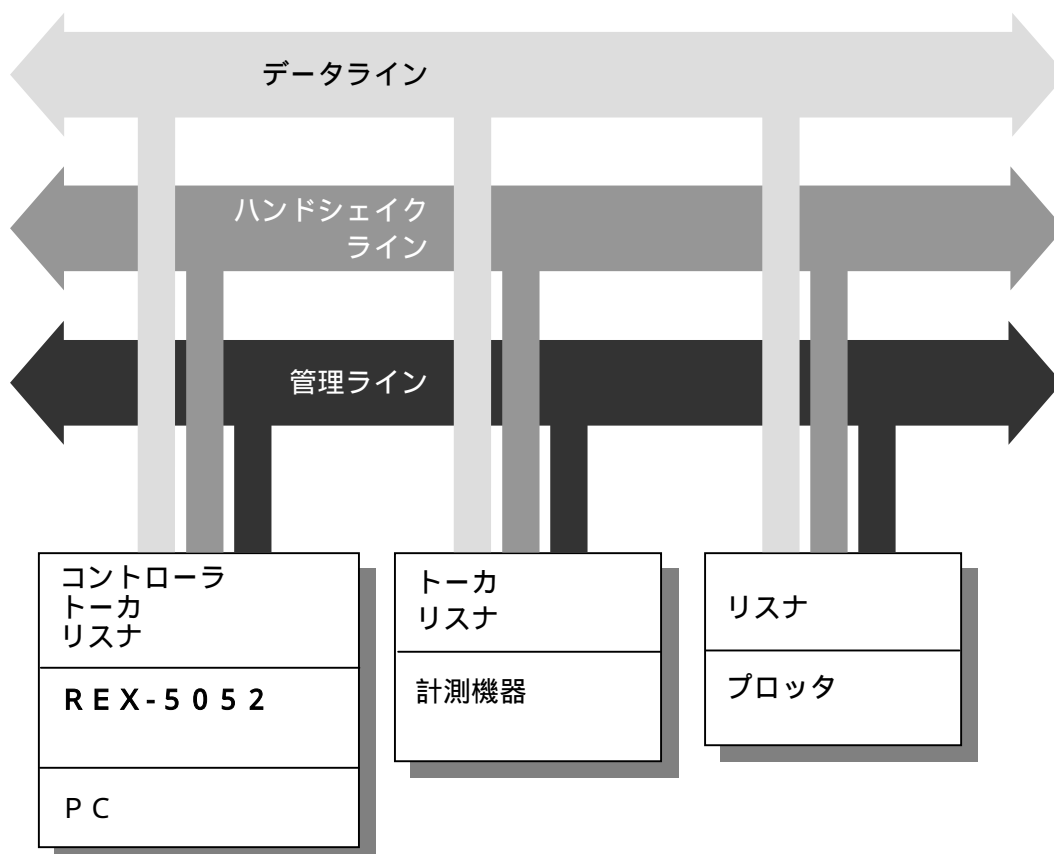
GPIB とは

GPIB は、General Purpose Interface Bus の略称で、計測器相互の制御やデータ伝送のための標準バスとして普及しています。今日ではプロトコルだけでなく、コネクタやケーブルなど全てを統一した仕様として、電気電子学会(IEEE)で公認され、一般的には IEEE488 や GPIB と呼ばれています。元々は、アメリカのヒューレット・パッカード社によって提案されたバスなので、HP-IB とも呼ばれています。

また、ヨーロッパ規格の IEC バスは、プロトコルや伝送方式、電気的条件等は IEEE488 と同一ですが、コネクタ(IEC は D-sub 25P)が異なりますので、REX-5052 と IEC バスをもつ計測器とを接続する場合は、別途 IEC-IEEE 変換コネクタを用意する必要があります。この IEC バスも GPIB と呼称されていますので、機器との接続の際にはご注意ください。

(1-1) インターフェイスバスの構成

<図 1>



GPIB バスの構成は、データラインが 8 本、ハンドシェイクラインが 3 本、管理ラインが 5 本の計 16 本となっています。図 1 は 24 本のバスライン（電線）に機器が並列接続されており、その内の 8 本のラインが GND レベルに接続されています。

GPIB のバスライン構成

データライン（8本）

データバスとして 8 本用意されていますので 8 ビットの平行データ（1 度に 1 バイト）の転送が行われます。

DIO1 ~ DIO8 (Input/Output)	双方向性のデータバス
-------------------------------	------------

ハンドシェイクライン（3本）

データの方向、伝送タイミングを制御します。

DAV (Data Valid)	True の時、トーカー、コントローラから送り出されたデータライン上のデータが有効であることを示します。
NRFD (Not Ready For Data)	True の時、リスナが BUSY であること、つまりデータ処理中であり次のデータを受け取る準備ができていないことを示します。
NDAC (Not Data Accepted)	True の時、リスナが受信動作を完了していないことを示します。

管理ライン（5本）

データの区別やインターフェイスの初期化などを管理します。

ATN (Attention)	True の時、データライン上のコントローラからのコマンドデータであることを示します。
REN (Remote Enable)	True の時、バス上の各機器をリモート状態にします。
IFC (Interface Clear)	一定期間 True となって、インターフェイスをクリアします。
EOI (End Of Identify)	True の時、データのデリミタ（区切）を示します。
SRQ (Service Request)	True の時、バス上の機器がコントローラに対してサービスリクエストを発信していることを示します。

GND ライン

各信号の GND レベルを設定するために使用します。

コネクタ端子	
1 2	シールド
2 4	ロジックグランド
2 3	ATN に対するグランド
2 2	SRQ に対するグランド
2 1	IFC に対するグランド
2 0	NDAC に対するグランド
1 9	NRFD に対するグランド
1 8	DAV に対するグランド

注) True は”L”レベルを示し、False は”H”レベルを示します。

(1-2) インターフェイスの機能

GPIB には、下記の 10 種類のインターフェイス機能が定められています。そして、実際には、これらの機能のうち必要なものを選択して組合せて使用します。GPIB 用機器やコントローラ(パソコン)を選択する場合には、この機能コードをあらかじめ調べておく必要があります。その機能を持っているかどうかということ、どのレベルまでの機能を持っているかということは、SR0,C4 のような機能シンボルコードと 0~9 の数字の組み合わせで示され、0 はその機能を持たないことを示します。

機能シンボル コード	インターフェイス 機能	機 能
SH	ソースハンドシェイク	バス上のデータを送信する
AH	アクセプタハンドシェイク	バス上のデータを受信する
T	トーカ	SH機能を使って、他の装置にデータを送る
L	リスナ	AH機能を使って、他の装置からデータを受け取る
C	コントローラ	バス上にコマンドを送り出して、GPIB システムをコントロールする
DT	デバイストリガ	トリガコマンドを受信し、装置をトリガする
DC	デバイスクリア	クリアコマンドを受信し、装置をリセットする
PP	パラレルポール	コントローラのパラレルポールに応答する
SR	サービスリクエスト	コントローラに対し SRQ を送り出す
RL	リモート・ローカル	コントローラからの指令により装置のリモートとローカル状態とを切りかえる

REX-5052 と GP-BIOS を使用した場合以下の表に示す機能を有します。

機 能	サブセット	内 容
SH	SH1	ソースハンドシェイク機能を持つ
AH	AH1	アクセプタハンドシェイク機能を持つ
C	C1	コントローラ機能を持つ
	C2	コントローラインチャージ機能を持つ
	C3	リモートイネーブル機能を持つ
	C4	SRQ に対する応答機能を持つ
	C28	インターフェイスメッセージ送信機能を持つ
T	T8	基本的なトーカ機能を持つ MLA によってトーカ機能が解除される
L	L4	基本的なリスナ機能を持つ MTA によりリスナ機能が解除される
SR	SR0	システムコントローラとしてのみ動作しますのでこれらの機能はありません。
RL	RL0	
PP	PP0	
DC	DC0	
DT	DT0	

図 1 の様に GPIB では、すべての機器がバスに対して、並列に接続されています。したがってバス上のデータは、L(リスナ)機能をもつ装置であれば同時に受信することができます。しかし送信(バス上へのデータの送り出し)は、必ずどれか一台のみしか行えません。

バス上でデータの衝突(同時に2台以上がトーカーとなる)が発生したり、受信データの指定などを行うために GPIB システムでは、コントローラ(C)機能が用意され各装置にはアドレスが割付けられます。通常のシステムでは、コントローラはバス上に1台のみ存在します。

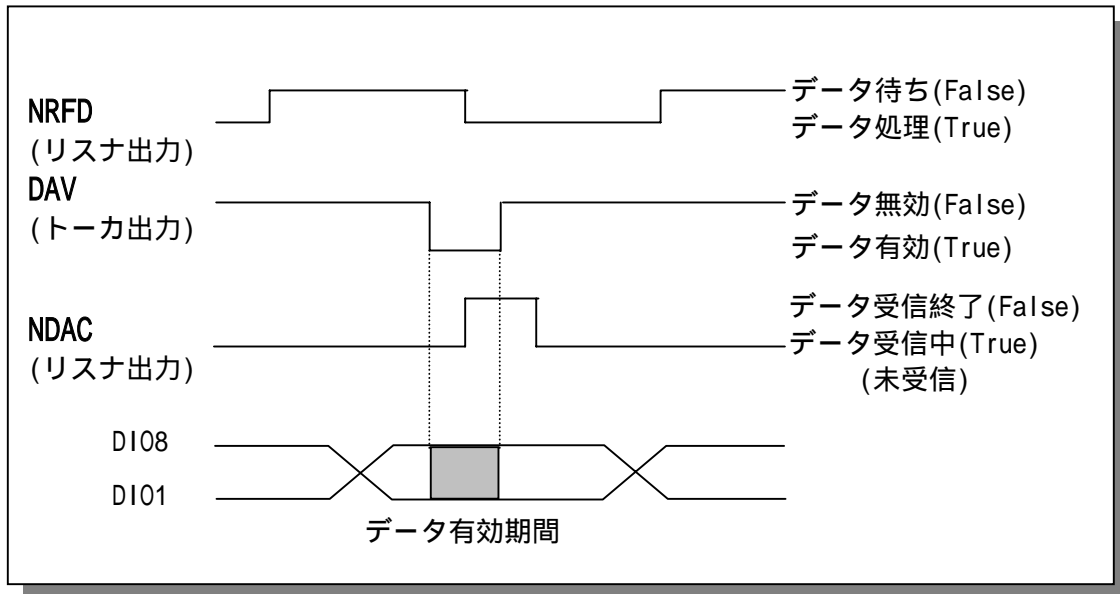
REX-5052 GPIB インターフェイスセットは、PC をコントローラとして機能させるためのインターフェイスセットで、他のコントローラとの同居はできません。従って REX-5052 と同時に GPIB 上で使用できる機器は、下記の機能を持つ装置に限られます。

- a) アドレス可能な装置であること。
- b) コントローラ機能を持たない(C0)こと。
(ATN, IFC, REN ラインの管理機能を持たないこと)

また、REX-5052 を実装し GP-BIOS が動作中の PC は、すべてコントローラインチャージ(コントローラとしてバスの制御権を獲得している状態)でありますので、GPIB 関係のコマンドを実行していなくとも、他のコントローラとバス上での同居はできません。

(1-3) ハンドシェイク機能

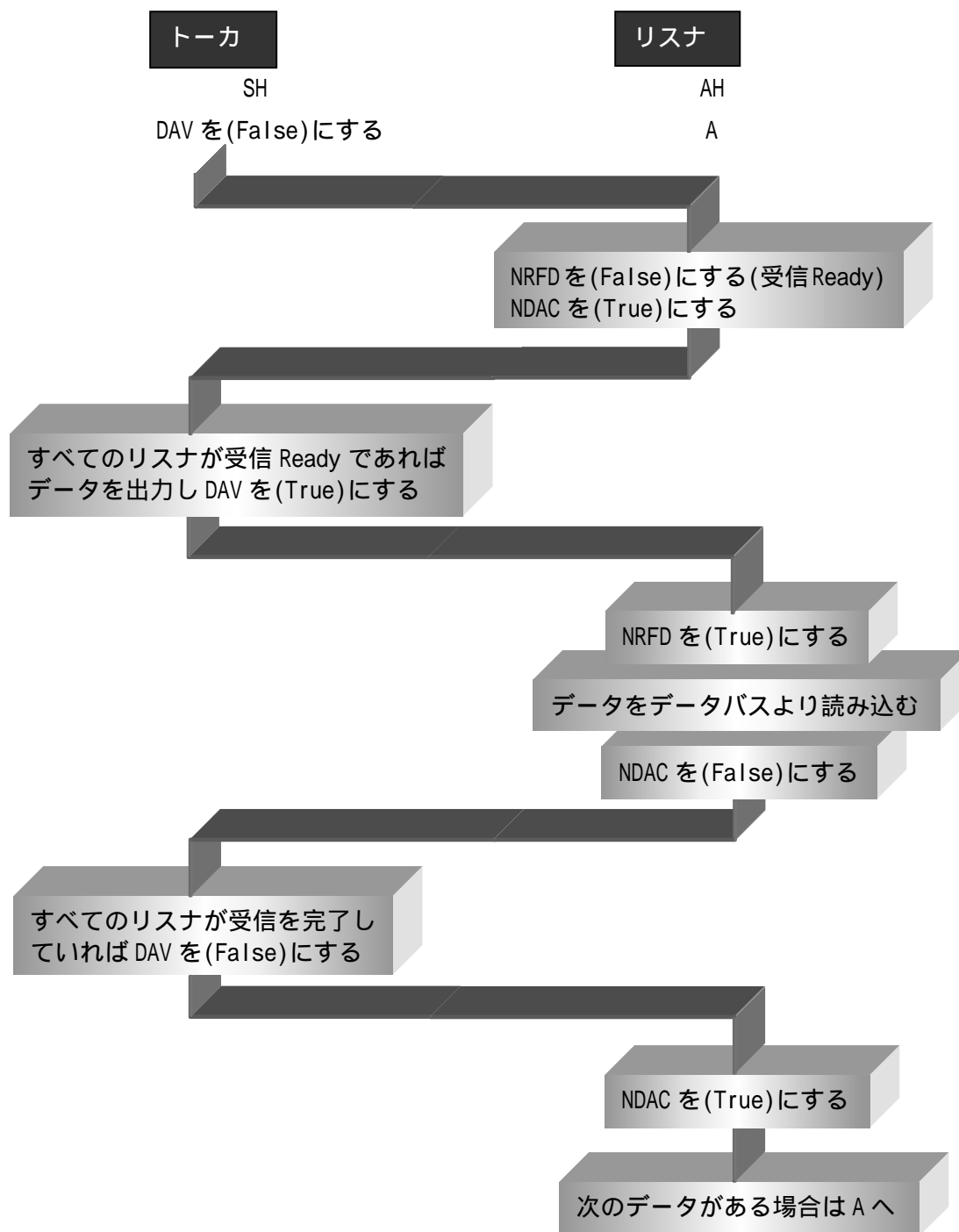
GPIB 上のデータ転送は、3本のハンドシェイクラインによって下記の要領で行われます。



DAV ラインは、バス上にその時点で SH を行うトーカのみが出力します。

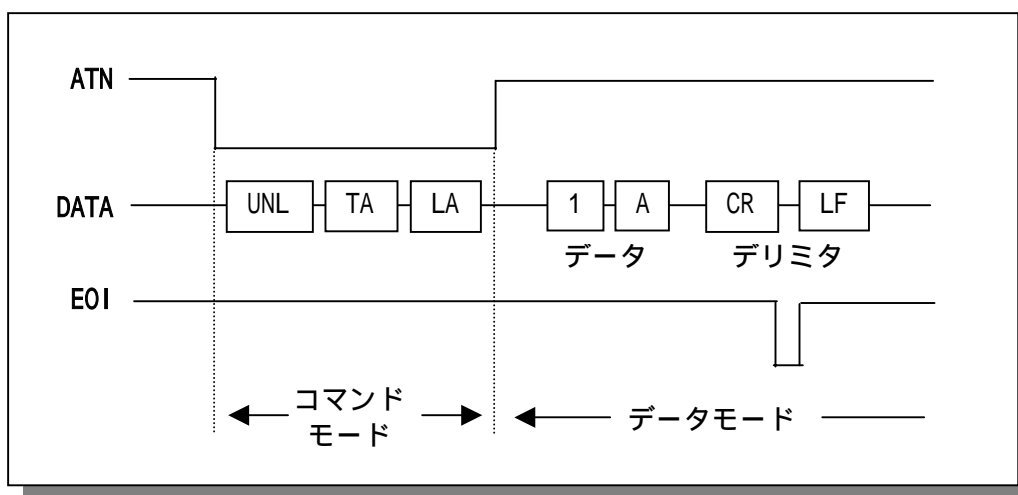
NRFD, NDAC は、バス上のすべてのリスナの出力の OR となります。つまりトーカは、バス上のすべてのリスナの NRFD 出力が False の状態であり、かつ同様にすべてのリスナ NDAC 出力が True の状態である場合のみ、DAV を True にします。
(データはそれ以前にデータバスに出力する)

ハンドシェイクのシーケンスを下記に示します。



(1-4) コマンド、データの送受信

GPIIB 上でトーカーとリスナが衝突しないようにコントローラが各機器にコマンドやアドレスを送り出してコントロールします。コマンドやアドレスもデータの種類としてデータバス上に送り出され、コントローラがトーカーとなり、各機器がリスナとなってハンドシェイクが行われます。そしてデータバス上のデータがコマンドやアドレスであることを示すために、ATN ラインが使用されます。



コントローラは各機器に対し、コマンドやアドレスを送出する場合に ATN ラインを “ True ” にして、コマンドおよびアドレスを送り出します。各機器は、コマンドモード (ATN が “ True ”) の間、コントローラから送られてくるコマンドやアドレスはすべて受信しなければなりません。

コマンドモードの終了は、ATN ラインが “ False ” に戻ることによって検知しますが、データモードのデータ列の終了はデータとして予め認定されている文字か EOI ラインを “ True ” にすることによって行います。このデリミタは、トーカーとリスナの間で予め認定しておく必要があります。

(1-5) SRQ 割り込みとシリアルポール

GPIIB に接続されている各機器は、SRQ ラインを “ True ” にすることにより、コントローラに対して割り込み要求 (サービスリクエスト) を出すことができます。(ただし機能仕様が SRQ の機器にはこの機能はありません。)

コントローラは、SRQ ラインが “ True ” になったことを検知し、バス上のどれかの機器がサービスを要求していることを確認します。そして、どの機器がどんな理由でサービスを要求しているか調べるためにポーリングを実行します。ポーリングには、一台ずつ呼び出して調べるシリアルポールと一度に八台ずつ調べるパラレルポールがあります。(REX-5052 ではシリアルポール機能のみサポートしています。)

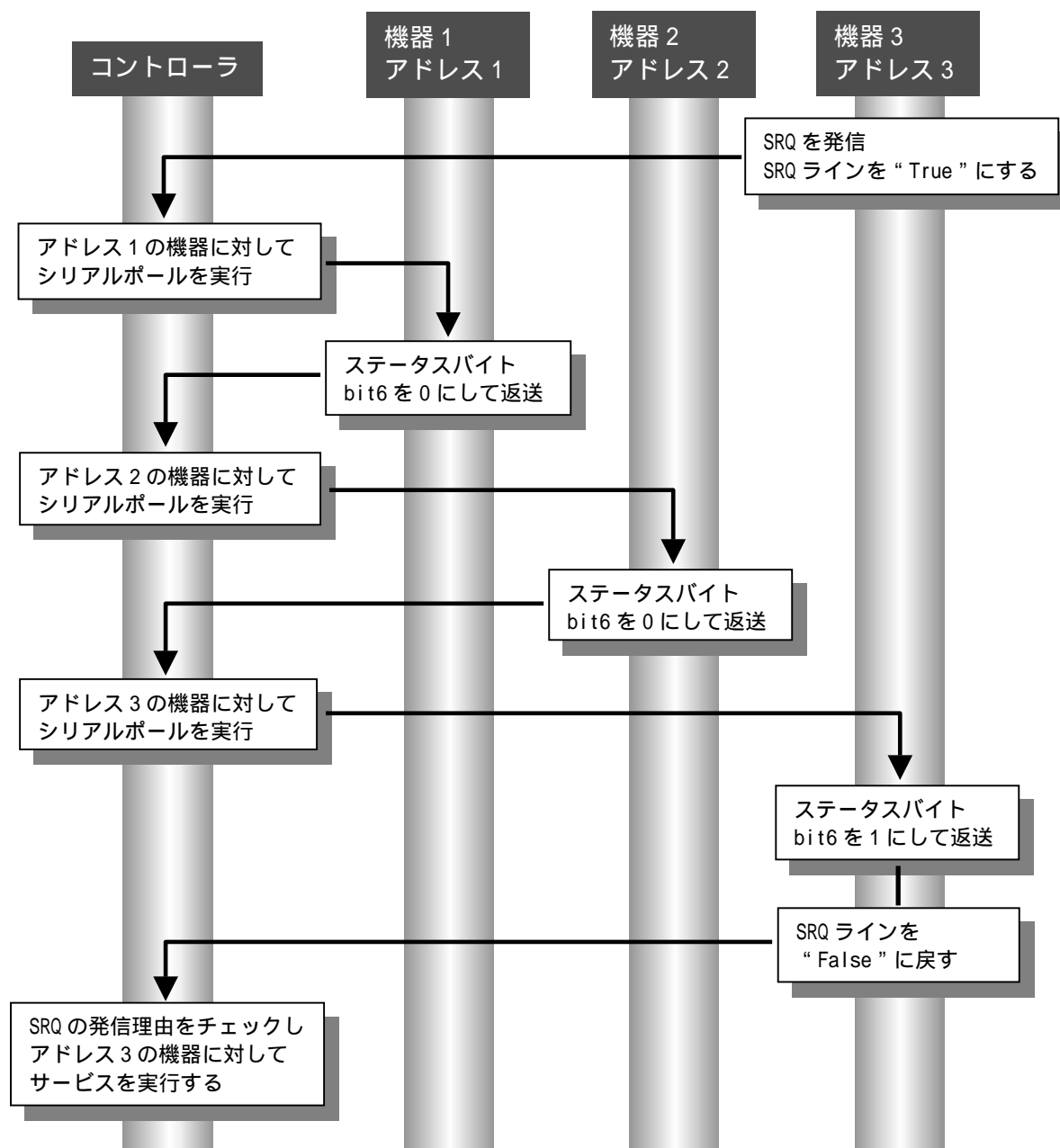
シリアルポールは、コントローラが SR 機能を持っている機器に対して一台ずつ順に SPE コマンドとその機器のアドレスを送り出すことによって開始されます。SPE コマンドと自分のアドレスを受信した機器は、ステータスバイトと呼ば

れる 1 バイトのデータをコントローラに対して送り返します。この時、自分が SRQ の発信元であれば、bit6 を “1” にし、発信理由を示すステータスを bit5 ~ bit0 にセットして送り返します。自分が発信元でなければ、bit6 を “0” にして送り返します。

ビット	b7	b6	b5	b4	b3	b2	b1	b0
意味	拡張用	SRQ 発信中						

← SRQ を発信している理由を示す。
(各機器により異なる) →

SRQ 割り込みとシリアルポールのシーケンスを下記に示します。



(空白ページ)