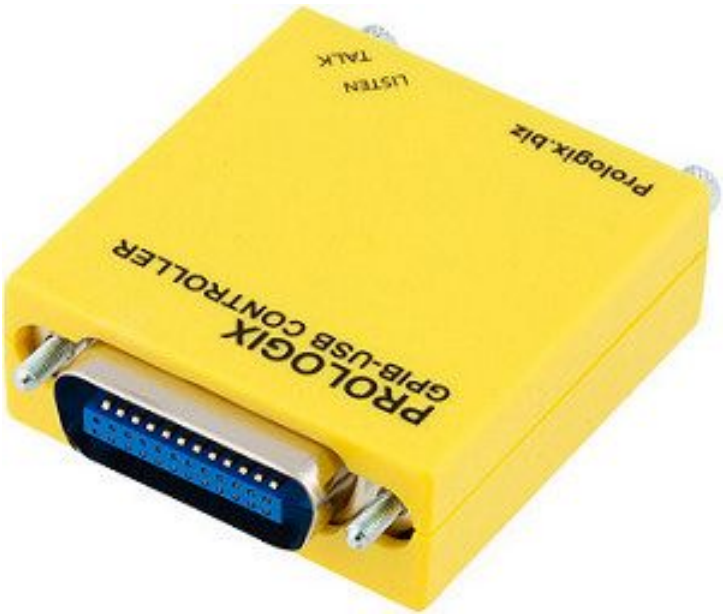
PROLOGIX

GPIB-USB コントローラ



ユーザーマニュアル

バージョン 6.107

2013 年 5 月 14 日

PROLOGIX.BIZ

目　次

変更履歴

2013年5月14日	++llo コマンド記述を追加。
2011年4月18日	++lon コマンド記述を追加。
2009年9月14日	++savecfg コマンド記述を追加。 read_tmo_ms の最大値を 3000 ms に修正。
2009年4月21日	++status コマンド記述を追加。セカンダリーアドレスに含めるために++addr, ++spoll と ++trg コマンド記述を更新。 バイナリ・データ通信の節を追加。
2008年3月1日	++trg コマンド記述を更新。
2007年12月22日	USBコマンドターミネータ記述とread_tmo_msコマンド記述を更新。
2007年7月9日	typo in ++modeコマンドを修正。*Mike Schulz に感謝!*
2007年7月7日	変更履歴を追加。 他のGPIB 文献と一致するようにPERIPHERALモードをDEVICEモードに名称変更。
2007年7月5日	最初のバージョン。

1. はじめに

Prologix GPIB-USB コントローラは、USB ポートを持つ全てのコンピュータを GPIB コントローラあるいはデバイスに変換します。

コントローラモードでは、Prologix GPIB-USB コントローラは、オシロスコープ、ロジックアナライザ、スペクトルアナライザ等の GPIB を使用可能なインスツルメントを遠隔制御できます。

デバイスモードでは、Prologix GPIB-USB コントローラは、コンピュータをインスツルメントのフロントパネルからデータとスクリーン描画のダウンロードのための GPIB 周辺機器に変換します。

両モードでは、Prologix GPIB-USB コントローラは、ホストコンピュータから受信した高レベルのコマンドを解釈し、適当な低レベルの GPIB プロトコルハンドシェーキングを実行します。

2. インストール

Prologix GPIB-USB コントローラを、これらの簡単なステップでインストールできます:

1.FTDI ウェブサイト(www.ftdichip.com)から FT245R チップのドライバーをダウンロードします。
2.Prologix GPIB-USB コントローラを USB A-B ケーブルを用いてコンピュータに接続します。
3.www.ftdichip.com/Documents/InstallGuides.htm の取扱説明書に従いドライバーをインストールします。
4.直接あるいは GPIB ケーブルを用いて、インスツルメントの GPIB コネクタにコントローラを差し込みます。

2 種類のドライバーが使用可能です:仮想 COM Port (VCP)ドライバーと Direct(D2XX)ドライバーです。Prologix GPIB-USB コンピュータを標準シリアル(RS-232)デバイスとして通信できるように、VCP ドライバーはホストコンピュータの標準シリアルポートをエミュレートします。D2XX ドライバーは、DLL インターフェイスによりコントローラへの直接アクセスを可能にします。

3. ファームウェアのアップグレード

Prologix GPIB-USB コントローラファームウェアは、アップグレード可能な分野です。最新のファームウェアとアップグレードインストールが prologix.biz で利用可能です。

4. ホストソフトウェア

多種多様なホストソフトウェアを、Prologix GPIB-USB との通信に使用できます:

ターミナルプログラム－HyperTerminal、Tera Term Pro や Minicom のような全てのターミナルエミュレーションプログラムを、それに接続されたコントローラとインスツルメントとの通信に使用できます。

カスタムアプリケーション ― シリアルポートへのアクセス(VCP ドライバーを使用する場合)を提供する、あるいは DLL へのインターフェース(D2XX ドライバーを使用する場合)を可能にする全てのプログラミング言語や環境を、カスタムアプリケーションの開発に使用できます。
National Instruments LabView や Agilent VEE のようなグラフィックプログラミング開発環境を、同様に使用できます。

EZGPIB ― 使いやすいプログラミング環境を、データ取得アプリケーション開発のために、Ulrich Bangert が開発しました。このツールへのリンクは、prologix.biz で見付けられます。

プロッターエミュレータ ―7470.exe のようなプロッターエミュレーションアプリケーション、PrintCapture と Plottergeist を GPIB-USB を用いてダウンロードされたスクリーンプロットの描写に使用できます。これらのツールの設定方法の詳細は、prologix.biz にあります。

5. 設 定

Prologix GPIB-USB コントローラを、次の方法の何れかで設定できます:

Prologix.exe ― Prologix.exe は,コントローラ設定のために、John Miles が開発したオープンソースツールです。このツールへのリンクを prologix.biz で見つけられます。

ターミナルプログラム ― HyperTerminal、Tera Term Pro や Minicom のような全てのターミナルミュレーションプログラムを、適当なコマンド(コマンド参照)の手動入力によるコントローラの設定に使用できます。ターミナルプログラムを用いて、USB ドライバーで作成された仮想 COM ポートをオープンします。
ボーレート、データビット、ストップビットとフローコントロールのようなシリアルポートパラメータは問題なく、全ての値にセットできます。入力されているコマンドを見るために、ターミナルプログラムの“Local Echo”を可能にすることができます。種々のターミナルプログラムの詳細設定ステップは、prologix.biz の FAQ か、ユーザマニュアルを参照して下さい。Prologix GPIB-USB コントローラは、不揮発メモリーに最新の構成設定を格納しています。これらの設定の、アドレス仕様はありません。GPIB バスに、異なる構成設定を必要とする多数のインスツルメントがある場合は、各インスツルメントと通信する前に設定を変更しなければなりません。

6. 動作モード

Prologix GPIB-USB コントローラを、CONTROLLER と DEVICE の 2 つのモードで作動できます。
++mode コマンド(コマンド参照)を用いて、2 つのモード間を切り替えられます。

6.1 コントローラモード

コントローラモードでは、GPIB-USB コントローラが、GPIB バスの Controller-In-Charge (CIC)として作動します。コントローラが、USB ターミネーターCR (ASCII 13)あるいは LF (ASCII 10)ーでターミネートされた USB ポート上のコマンドを受信すると、リスンのために、現在指定されているアドレス(++addr コマンド参照)の GPIB インスツルメントをアドレス指定し、受信データを伝えます。

Read-After-Write 機能が enable になっていると(++auto コマンド参照)、Prologix GPIB-USB コントローラは、その応答を読むために、コマンド送信後、トークのためにそのインスツルメントをアドレス指定します。GPIB 上のインスツルメントから受信した全てのデータを、USB でホストに送信します。Read-After-Write 機能は、インスツルメントとの交信を簡単にします。低レベルの GPIB プロトコル詳細を考慮しないで、コマンドを送信し応答を読めます。

Read-After-Write 機能が enable でない時は、Prologix GPIB-USB コントローラがトークするためのインスツルメントを自動的にアドレス指定しません。データを読むためには、++read コマンドを使用しなければなりません。

コントローラモードは、インスツルメントの遠隔制御とホストコンピュータからのプロットコマンドの送信によって、スクリーンプロットをダウンロードするのに使用します。

6.2 デバイスモード

デバイスモードでは、Prologix GPIB-USB コントローラは、GPIB バスの別の周辺機器として作動します。このモードでは、コントローラは、GPIB トーカあるいは GPIB リスナとしてのみ作動可能です。Prologix GPIB-USB コントローラは、このモードの間、Controller-In-Charge ではないので、GPIB コントローラからのコマンドの受信を想定しています。デバイスモードが enable の場合、Prologix GPIB-USB コントローラは、GPIB リスナとして自身を設定します。GPIB ポート上で、コントローラで受信した全てのデータは、バッファリングなしに USB ポートに伝えられます。

USB 上のホストから受信した全てのデータは、GPIB コントローラが、トークのために Prologix GPIB-USB コントローラをアドレス指定するまでバッファします、その時になると、バッファしたデータを GPIB ポートに伝えます。コントローラは、1 つのコマンドだけをバッファリングできます。以前のものがまだ GPIB 上に送られていない場合は、USB 上で受信した次のコマンドは、以前にバッファリングしたものを上書きします。

デバイスモードは、プロッターエミュレーションソフトを用いて描画するために、インスツルメントのフロントパネルから、スクリーンプロットをダウンロードするのに使用します。

7. データ通信

コントローラとデバイスモードでは、USB ポート上で受信した文字は内部バッファーに集められ、USB 終了文字－CR (ASCII 13) あるいは LF (ASCII 10)－を受信した時、解釈します。もし CR、LF、ESC (ASCII 27)あるいは‘+’ (ASCII 43) 文字が USB データの一部であるなら、それに ESC 文字を先行して、回避しなければなりません。全ての回避されなかった USB データの LF、CR と ESC と‘+’文字は、廃棄されます。

前述したように、回避されなかったCRやLFは、USBターミネータとして作動します。CRやLFは除かれ、GPIB終了文字(++eosコマンドで指定される)が、インスツルメントへのデータ送信前に追加されます。

回避されなかった“++”文字で始まる全てのUSB入力は、コントローラコマンドとして解釈し、GPIB上に送信されません。

そのような設定の時は、ー++auto コマンドか++read コマンドを使用してー、インスツルメントから受信した文字はホストに送信されます。インスツルメントにデータを送信する時と違って、 文字の置換はありません。ファームウェアは、1文字のみをバッファリングします。FTDI USBドライバーは、約4KBの追加のバッファリングを提供します。デバイスがトーキングしていて、バッファー(ファームウェアとドライバーの)が一杯の場合、デバイスは、トークモードを続けますが、GPIBハンドシェーキングを使用しての更なるデータ送信は停止されます。

++eot_char コマンドを、GPIB EOI 信号確認の検出に使用できます。

7.1. バイナリーデータ通信

Prologix GPIB-USBコントローラは、GPIBが使用可能なインスツルメントとバイナリーデータの送受信が可能です。

インスツルメントからのバイナリーデータの受信には、特別な処置は必要ありません。インスツルメントから受信した全てのバイナリーデータは、PCにUSBで、ASCIIデータと同様に未修正で送信されます。インスツルメントからのバイナリーデータは、通常、CRやLF文字で終了しないので(通常、ASCIIデータの場合のように)、データの終わりを示すEOIを検出する++eot_enableコマンドの使用を希望することができます。より詳細は、++eot_enableコマンドのヘルプをご覧下さい。

インスツルメントへのバイナリーデータ送信時は、特別の注意をしなければなりません。次の文字の何れかがバイナリーデータに発生した場合ーCR (ASCII 13)、LF (ASCII 10)、ESC (ASCII 27)、‘+’ (ASCII 43) ーそれらにESC文字を先行することによって、それらを避けなければなりません。

例えば、次の(10進法)バイナリーデータの送信に対して:
00 01 02 13 03 10 04 27 05 43 06

それを、次のように避けなければなりません:
00 01 02 **27** 13 03 **27** 10 04 **27** 27 05 **27** 43 06

更に、CR やLF のようなGPIB終了文字をバイナリーデータに追加した場合、大部分のインスツルメントは混同します。そのような挙動を抑制するために、++eos 3コマンドを使用します。より詳細は、++eosコマンドのヘルプをご覧下さい。

8. コマンド

Prologix GPIB-USB コントローラは、その挙動を設定するいくつかのコマンドを提供します。
それらを、次の節で詳細に説明します。全てのコマンドは、“++”文字列で始まります。

コマンドは、CR か LF で終了しなければなりません。

8.1. addr

addr コマンドは、GPIB アドレスの設定、あるいは問合せに使用します。GPIB アドレスの意味は、コントローラの動作モードに依存します。コントローラモードでは、制御しているインスツルメントの GPIB アドレスを参照します。デバイスモードでは、Prologix GPIB-USB コントローラがエミュレートしている GPIB 周辺機器のアドレスです。

オプションのセカンダリーアドレスも指定できます。セカンダリーアドレスは、プライマリーアドレスとスペース文字で分離しなければなりません。有効なセカンダリーアドレス値は 96 から 126(10 進)です。96 のセカンダリーアドレスは、0 のセカンダリー GPIB アドレスに対応し、97 は、1 に対応するなど。セカンダリーアドレスの指定は、デバイスモードに影響しません。
このコマンドをパラメータなしで出すと、現在設定されているアドレス(指定されている場合、プライマリーとセカンダリー)を返します。

構文: ++addr [<PAD> [<SAD>]]
PAD (プライマリーアドレス) は 0 と 30 の間の 10 進値です。
SAD(セカンダリーアドレス)は、96 と 126 の間の 10 進値です。SAD はオプションです。

利用可能モード: コントローラ、デバイス

例:

```
++addr 5      ープライマリーアドレスを 5 にセットする
++addr        ー現在のアドレスを問い合わせる
++addr 9 96   ープライマリーアドレスを 9 に、セカンダリーアドレスを 0 にセットする
```

注記:
多くの HP-GL/2 プロッターの初期設定 GPIB アドレスは 5 です。

8.2. auto

Prologix GPIB-USB コントローラは、コマンドを送信後、それらの応答を読み込みトークするために、インスツルメントを自動的にアドレス指定するよう設定できます。

Read-After-Write と呼ばれる機能は、繰り返し読込コマンドを出さねばならぬことからユーザーを救います。このコマンドは、Read-After-Write 機能を使用可能、あるいは使用不可にします。

更に、auto コマンドは、トークあるいはリスンのために、現在指定されているアドレスのインスツルメントをアドレス指定します。++auto 0 は、リスンのためにインスツルメントをアドレスし、++auto 1 はトークのためにインスツルメントをアドレスします。
もし、このコマンドを引数なしで出した場合、read after-write 機能の現在の状態を返します。

構文: ++auto [0|1]

利用可能モード: コントローラ

注記:
いくつかのインスツルメントは、応答(しばしば non-query コマンドと呼ばれる)を生成しないコマンドを送信後、トークのためにアドレス指定すると、“Query Unterminated”あるいは“-420”エラーを生じます。事実上インスツルメントは、私は尋ねられていますが話すことはありませんと、言ってます。このエラーはしばしば良性で無視できます。さもなければ、インスツルメントの応答を読むために、++read コマンドを使用して下さい。例えば:

++auto 0ー read-after-write をオフにし、インスツルメントをリスンのためにアドレスします。
SET VOLT 1.0　　ー 非問い合わせコマンド
*idn?　　ー 問い合わせコマンド
++read eoi ー インスツルメントにより EOI を示されるまで読む
"HP54201A" ー インスツルメントからの応答

8.3. clr

このコマンドは、Selected Device Clear (SDC) メッセージを現在指定されている GPIB アドレスに送信します。特別なインスツルメントがこのメッセージにどのように応答するかについての詳細は、プログラミングマニュアルを参照して下さい。

構文: ++clr

利用可能なモード: コントローラ

8.4. eoi

このコマンドは、GPIB ポート上に送る全てのコマンドの最後の文字に、EOI 信号の宣言を使用可か使用不可にします。いくつかのインスツルメントは、コマンドの終わりを適切に検出するために、EOI 信号を宣言する必要があります。

構文: ++eoi [0|1]

利用可能なモード: コントローラ、デバイス

例:

++eoi 1	最後文字に EOI 宣言を利用可にします
++eoi 0	EOI 宣言を利用不可にします
++eoi	EOI が利用可か不可かを問合せます

8.5. eos

このコマンドは、GPIB 終了文字を指定します。ホストからのデータを USB 上で受信すると、全ての non-escaped LF、CR、ESC 文字を除去し、インスツルメントにデータを送信する前に、このコマンドで指定された GPIB ターミネータを追加します。このコマンドは、GPIB ポートで受信したインスツルメントからのデータに影響しません。

このコマンドが引数なしに出された場合、現在の設定を返します。

構文: ++eos [0|1|2|3]　ここで、: 0 － CR+LF、1 －CR、2－LF、3 －なし

利用可能なモード: コントローラ、デバイス

例:

++eos 0	インスツルメントコマンドに CR+LF を追加します
++eos 1	インスツルメントコマンドに CR を追加します
++eos 2	インスツルメントコマンドに LF を追加します
++eos 3	インスツルメントコマンドに何も追加しません
++eos	現在の EOS 状態を問合せます

8.6. eot_enable

このコマンドは、GPIB ポートから文字を読み込み中に、EOI が検出された時はいつでも、USB 出力へユーザーが指定した文字(eot_char 参照) の追加を利用可あるいは利用不可にします。

このコマンドを、引数なしに出した場合、eot_enable の現在の状態を返します。

構文: eot_enable [0|1]

利用可能なモード: コントローラ、デバイス

例:

++eot_enable 1	EOI 検出時、ユーザー指定文字を追加します
++eot_enable 0	EOI 検出時、文字を追加しません
++eot_enable	現在の eot_enable の状態を問合せます

8.7. eot_char

このコマンドは、eot_enable が 1 にセットされ、かつ EOI を検出した時に、USB 出力に追加される文字を指定します。

もし、引数なしでこのコマンドを出した場合、現在指定されている文字を返します。

構文: eot_char [<char>]　ここで、<char>は 256 未満の 10 進値

利用可能モード: コントローラ、デバイス

例:

++eot_char 42	EOI 検出時、* (ASCII 42)を追加します
++eot_char	現在設定されている eot_char を問合せます

8.8. ifc

このコマンドは、Prologix GPIB-USBコントローラをGPIBバス上のController-In-Chargeにして150マイクロセカンド、GPIB IFC 信号を宣言します。

構文: ++ifc

利用可能モード:コントローラ

8.9. llo

このコマンドは、現在アドレス指定されているインスツルメントのフロントパネル操作を使用禁止にします。

構文:++llo

利用可能モード: コントローラ

8.10. loc

このコマンドは現在アドレス指定されているインスツルメントのフロントパネル操作を使用にします。

構文: ++loc

利用可能モード: コントローラ

8.11. lon

このコマンドは現在指定されているアドレスに関係なくGPIB バス上の全ての通話に対して GPIB-USB をリスンとして設定します。この設定はリスンオンリーモードとしても知られています。このモードにおいてコントローラは受信だけでき、どんなデータも送信できません。

構文: ++lon[0|1]

利用可能モード: デバイス

例:

++lon1	リスンオンリーモード可能
++lon0	リスンオンリーモード禁止
++lon	リスンオンリーモード問合せ

8.12. mode

このコマンドは, Prologix GPIB-USB コントローラをコントローラまたはデバイスに設定します。

もし、引数なしにこのコマンドを出すと、現在のモードを返します。

構文: ++mode[0|1]ここで 1-コントローラ、0-デバイス

利用可能モード: コントローラ、デバイス

例:

++mode1	コントローラモードに切り替えます。
++mode0	デバイスモードに切り替えます。
++mode	現在のモードを問合せます。

8.13. read

このコマンドは、以下の時までインスツルメントからデータを読むのに使用できます

- EOI を検出するかタイムアウト終了になる、あるいは
- 指定された文字を読むかタイムアウト終了になる、あるいは
- タイムアウト終了になる

タイムアウトは read_tmo_ms コマンドでセットし、inter-character delay すなわち、最後の文字を読み込んでからの遅れに適用します。タイムアウトは、データを読み込んだ合計時間と混同してはなりません。

構文: ++read [eoi|<char>] ここで、<char>は 256 未満の 10 進値です

利用可能モード: コントローラ

例:

++read	タイムアウトまで読みます
++read eoi	EOI が検出されるかタイムアウトまで読みます
++read 10	LF (ASCII 10)を受信するかタイムアウトまで読みます

8.14. read_tmo_ms

このコマンドは、読込コマンドと spoll コマンドで使用するため、ミリ秒で、タイムアウト値を指定します。
タイムアウトを、1 と 3000 ミリ秒間の全ての値にセットできます。

構文: ++read_tmo_ms <time> ここで、<time>は 1 と 3000 の間の 10 進値

利用可能モード: コントローラ

8.15. rst

このコマンドは、コントローラのパワーオンリセットを実行します。プロセスは約 5 秒かかります。この時間中に USB で受信した全ての入力は無視されます。

構文: ++rst

利用可能モード: コントローラ、デバイス

8.16. savecfg

このコマンドは、EPROM の設定パラメータの自動保存を使用可にするか、使用不可にします。
使用可の場合、それらはアップデートされるといつでも、次の設定パラメータは保存されます-
mode, addr, auto, eoi, eos, eot_enable, eot_char と read_tmo_ms。

しかし、頻繁なアップデートは、結局は EPROM を消耗します。このコマンドは、EEPROM の消耗を減らすために、設定パラメータの自動保存を一時的に使用不可にするために使用します。
savecfg 設定自体は EPROM に保存されません。それは、常に使用可です(電源オンまたはリセット後)。

構文: ++savecfg [0|1]

利用可能モード: コントローラ、デバイス

例:

++savecfg 1　EPROM の設定パラメータの保存を使用可にします。
++savecfg 0　EPROM の設定パラメータを使用不可にします。
++savecfg　　現在の設定を問合せます。

注記:

“++savecfg 1”コマンドは、パラメータの自動保存を使用可能にするだけでなく、全ての設定パラメータの現在値をすぐに保存します。

8.17. spoll

このコマンドは、指定されたアドレスのインスツルメントのシリアルポールを実行します。アドレスが指定されない場合、このコマンドは、現在アドレス指定されているインスツルメント(前の++addr コマンドでセットした)をシリアルポールします。このコマンドは、read_tmo_ms コマンドで指定されたタイムアウトを使用します。

構文: ++spoll [<PAD> [<SAD>]]

PAD(プライマリーアドレス)は、0 と 30 の間の 10 進値です。

SAD(セカンドリーアドレス)は、96 と 126 の間の 10 進値です。SAD はオプションです。

利用可能モード: コントローラ

例:

++spoll 5	プライマリーアドレス 5 のインスツルメントをシリアルポールします。
++spoll 9 96	プライマリーアドレス 9、セカンドリーアドレス 0 のインスツルメントをシリアルポールします。
++spoll	現在指定されているインスツルメントをシリアルポールします。

8.18. srq

このコマンドは、GPIB SRQ 信号の現在の状態を返します。このコマンドは、SRQ 信号がアクティブ状態であれば(Low)、‘1’を、信号がアクティブ状態でなければ(High)、‘0’を返します。

構文: ++srq

利 用可能モード: コントローラ

8.19. status

status コマンドは、GPIB コントローラでシリアルポールされた時、返されるデバイスステータスバイトを指定するのに使います。ステータスバイトの RQS ビット(ビット#6)をセットすると、SRQ 信号がアクティブ状態になります(低)。シリアルポール後、SRQ ラインをアクティブ状態でなくして、ステータスバイトを 0 にセットします。ステータスバイトは、電源を入れると 0 に初期化されます。

GPIB コントローラから DEVICE CLEAR (DCL) メッセージ、あるいは SELECTED DEVICE CLEAR (SDC) メッセージを受けると、SRQ もアクティブ状態でなくなり、ステータスバイトはクリアされます。

もし、このコマンドが何の引数もなく出されると、現在指定されているステータスバイトを返します。

構文: ++status [0-255]

利用可能モード: デバイス

例:

++status 48　48 とシリアルポールステータスバイトを指定します。ビット#6 をセットしてから、このコマンドが SRQ をアクティブにします。

++status　　現在のシリアルポールステータスバイトを問合せます。

8.20. trg

このコマンドは、指定されたアドレスのデバイスに Group Execute Trigger GPIB コマンドを出します。おそらく、15 アドレスまで指定できます。アドレスはスペースで区切らなければなりません。もし、アドレスを指定しない場合、Group Execute Trigger コマンドを、現在アドレス指定されたインスツルメントに出します(前の++addr コマンドでセットした)。Group Execute Trigger コマンドへの指定したインスツルメントの応答に対しては、プログラミングマニュアルを参照して下さい。

構文: ++trg [<PAD1> [<SAD1>] <PAD2> [SAD2] …<PAD15> [<SAD15>]]

利用可能モード: コントローラ

8.21. ver

このコマンドは、Prologix GPIB-USB コントローラのバージョン文字列を返します。

構文: ++ver

利用可能モード: コントローラ、デバイス

8.22. help

このコマンドは、利用可能な全コマンドの概要をプリントします。

構文: ++help

利用可能モード: コントローラ、デバイス

9. 仕様

サポート OS　　: Windows 98/2000/XP/Vista/7、 Mac OS 8/9/X, Linux、FreeBSD
サポート標準　　: IEEE 488.1、IEEE 488.2、USB 1.1、USB 2.0
サポートされない GPIB コマンド: PARALLEL POLL、PASS CONTROL
電源 : USB バスパワー、+5V, 100 mA (最大)
サポート USB ハブ : セルフパワーとバスパワーハブ
インジケータ　　:TALK、LISTEN
寸法　　:2.5 in. (L) x 2.5 in. (W) x 1.0 in. (H)
重量　　:3 oz

このマニュアルの包括的見直しといくつかの改善提案に対し、Gerry Glauser に感謝します!