UV-11

(USB-GPIB Converter)

取扱説明書

REV1.0

20001/07/20

株式会社 計測技術研究所

本製品のお問い合わせ先

株)計測技術研究所
〒222-0033 横浜市都筑区茅ヶ崎南 2-12-2
TEL:045-948-0211(代)
FAX:045-948-0221
URL http://www.keisoku.co.jp/
E-mail pwsales@hq.keisoku.co.jp

梱包品

この度は UV-11 をお買い上げ頂き誠に有り難うございます。
UV-11 の梱包品は以下のようになっております内容を御確認ください。

- UV-11 本体 1(ゴム足 4 ヶ付き)
- USB ケーブル 1 本(1.8m)
- CD-ROM 1 枚
- Windows 用デバイスドライバー(添付 CD-ROM に収録)
- Windows 用制御ライブラリ(添付 CD-ROM に収録)
- ユーティリティソフトウエア(添付 CD-ROM に収録)
- PDF 版取り扱い説明書(添付 CD-ROM に収録)
- 保証書
- ユーザー登録用紙

PDF 版取扱説明書の読み方

本品の取扱説明書は P DF とよばれる形式のファイルで供給されています。P DF は AdobeSystems 社により開発された文書フォーマットであり、これを読むためには Adobe 社より供給されている AdobeAcrobatReader をインストールする必要があります。このソフトウエアは本品添付の CD-ROM に AcrobatReader が収録してあります。インストールを行うには \ Acroread \ Reader の下にある Setup.exe を実行してください。
なお印刷された取扱説明書は添付されません。必要に応じて PDF 版から印刷されますようお願いいたします。また現在のところ有償・無償を問わず印刷された取扱説明書の供給予定はありません。御了承ください。

ソフトウエアと取扱説明書のバージョンアップ

UV-11 に添付されているソフトウエアと取扱説明書は不備な点の修正や機能追加に伴いバージョンアップされますが、これらは弊社ホームページより最新版を提供していく予定です。

弊社ホームページの URL は
http://www.keisoku.co.jp/
また UV-11 のページは
http://www.keisoku.co.jp/pro/pw/uv11.html
になります。適時ここを参照するようお願いします。

※ UV-11 のページにつきましては今後アドレスが変更になるかもしれません。その際はルートページから参照されますようお願いします。

保証規定

UV-11 は出荷日から1 年間の うちに通常使用環境下において使用不能にならないことを保証しています。本製品は弊社規定の検査に合格しておりその品質については万全を期していますが通常使用中になんらかの故障等により動作不能となった場合は無償にて修理させて頂きます。但し故障原因が弊社の製造上の理由に拠らない場合は有償にとなりますので御了承ください。
添付のソフトウエアの媒体については通常の使用であれば 90 日間読み出し不能にならないことを保証します。保証期間中に読み出し不能となった場合は無償にて代替品を送付させて頂きます。また保証期間外の場合は有償となります。

保証の限定

(株)計測技術研究所は上記保証規定以外のいかなる保証も行いません。UV-11 を使用して生じたいかなる損害についてもこれを保証するものではありません。また設置上の問題や火事、風水害、停電、電源サージ、その他なんらかの事故による破損等は上記保証の範囲外となります。

警告

UV-11 は医療または診療用には作られていません。この警告を無視し上記目的に使用した場合、そのいかなる損害も保証するものではありません。

著作権

登録商標

イントロダクション

- UV-11 の概要と基本

UV-11 の概要

UV-11 はパーソナルコンピュータからGPIB 機器と通信を行うことを目的とした機器です。UV-11 は USB インターフェースを備え、パーソナルコンピュータから非常に簡単に利用することができます。UV-11 はプラグ&プレイに対応しています。ホストコンピュータへ接続する場合、パーソナルコンピュータの電源を切ったり再起動したりする必要はありません。また新たに割込み番号や DMA チャンネルを必要としません。UV-11 は USB ハブと併用することにより最大 10 台同時に 1 台のパーソナルコンピュータへ接続して利用することができますが、この場合でも新たな割込み番号などが必要になることはありません。またパーソナルコンピュータの電源が入ったまま好きなときに UV-11 の接続・切り離しができます。

UV-11 は USB ケーブルにて給電される電源で動作します。このため AC 電源を必要としません。ただし UV-11 は 1 台に付き 300mA 程度の電流を消費します。このため USB 規格上 1 つの USB ポートに 2 台以上の UV-11 を接続してご利用になることはできません。また UV-11 以外の機器でも電流容量の制限から同時にご利用になれない機器があります。

UV-11 は USB インターフェースをサポートする Microsoft 社の Windows98（セカンドエディションを含む）又は WindowsMe 及び Windows2000（何れも日本語版のみ）で利用することができます。これ以外の OS では動作しません。必ずこれらの動作しているパーソナルコンピュータへ接続して使用します。※1

UV-11 の GPIB インターフェースは IEEE488.1 に準拠しています。※2 UV-11 に接続できる GPIB 機器は UV-11 自身を除いて 14 台までです。この台数は GPIB の仕様に基づくものです。また UV-11 は GPIB インターフェースを電気的に絶縁していません。UV-11 を介してパーソナルコンピュータと GPIB 機器は GND 間が接続されます。オシロスコープなど GND とプローブ GND が絶縁されていない測定器などをご利用になる場合は十分、ご注意ください。

UV-11 を使用する際は専用のデバイスドライバと呼ばれるソフトウエアが必要となります。これは UV-11 に標準で添付されています。デバイスドライバをパーソナルコンピュータに組み込む際にインストール作業が必要となりますがその手順はパーソナルコンピュータ画面上に表示されますので、これに従うだけでできます。具体的には 21 ページから始まるデバイスドライバのインストールをご覧ください。デバイスドライバのインストールは 1 台のパーソナルコンピュータにつき 1 回だけ必要となります。これは複数台の UV-11 を利用する場合も同様です。デバイスドライバのインストールが正しく終われば UV-11 は直ちに利用できるようになります。

UV-11 を制御するには添付のライブラリを使用します。ライブラリは通常の DLL 形式と ActiveX 形式の 2 種類を提供します。DLL は Microsoft VisualC++ で利用する際利用されることを想定しています。ActiveX は VisualBasic を初めとするスクリプトが使用できるソフトウエア全般での利用を想定しています。スクリプトが利用できるソフトウエアとしては VisualBasic の他 MicrosoftExcel、Word、Access といったものなど非常に多くあります。

※1 詳しくは 17 ページの「ホストコンピュータの条件」を参照してください。
※2 IEEE488.2 には対応していません。

USB インターフェースについて

USB(UniversalSerialBus)はパーソナルコンピュータ用に開発された比較的高速なシリアル のインターフェースです。パーソナルコンピュータと外部周辺機器を接続することを目的 としています。USB は標準でプラグ＆プレイに対応しており、簡単に周辺機器が接続できま す。電源の入った状態での抜き差し(活線挿抜)できるようになっており、パーソナルコン ピュータの電源を切ること無く周辺機器を接続できます。USB は 1.5Mbps と 12Mbps の 2 種類の通信速度をサポートしています。ユーザーは機器による通信速度の違いを意識する必要はありません。これらは機器ごとに自動的に判別されます。UV-11 は 12Mbps で通信 を行っています。1 つの USB インターフェースには 127 台までの周辺機器を接続すること ができます。USB インターフェースは既に殆どのパーソナルコンピュータに標準で装備されています。
USB 機器は必ず USB をサポートする OS が動作するパーソナルコンピュータに接続される必要があります。※1USB 機器同士で直接データをやり取りすることはできません。USB 機器を数多く接続する際には USB ハブと呼ばれる機器が必要となります。USB ハブは最大 5 段まで接続できます。
USB 機器を接続するケーブル長は最大 5m です。12Mbps 用のケーブルは 1.5Mbps 用としても使用できますがその反対はできません。ケーブルのコネクタは誤接続防止用に両端の 形状が異なっています。通常 A タイプと呼ばれる幅広の側をパーソナルコンピュータに、A タイプと呼ばれる細い方を機器側に接続します。

USB ケーブルについての注意

USB ケーブルには 1.5Mbps 専用と 1.5Mbps/12Mbps 兼用の 2 種類があります。UV-11 に使 用するケーブルは必ず 12Mbps に対応したものを使用してください。※2

※1 本製品は Microsoft Windows 以外には対応していません
※2 通常単品で販売されているケーブルは 12Mbps に対応しています。

GPIB について

GPIB とはコンピュータと計測器の間で通信を行うことを主な目的としているバスです。元々 HP(ヒューレット・パッカード)社で開発されたため HP-IB と呼ばれることもあります。また IEEE にて規格として定められたため IEEE488 と呼ばれることもあります。GPIB は以 下のような特徴を持っています。

- データ線 8 本、ハンドシェーク線 3 本、管理用線 5 本グランド線 8 本からなる
- 接続はスター、デイジーチェーンなど組み合わせが自由
- ピギーバック式コネクタを使用している
- 計測機器の分野では事実上の標準である(採用メーカーが多い)
- 1つのバスに接続できる機器は 15 台までである
- 最大データ転送速度が 1MB 程度である

また GPIB はデータを転送することが主な目的でそのデータの内容については規定されてい ません。どのようなデータを転送すべきかはユーザーが判断します。

GPIB 機器の接続

GPIB 機器は UV-11 に 14 台まで接続できますがその接続には以下のような注意点があります。

- デバイス間の距離はどこをとっても 4m 以内でなければならない
- バス全体でのデバイス間の平均接続距離は 2m 以下でなければならない
- 1 つのバスでの GPIB ケーブルのトータル距離は 20m 以内または機器の台数×2m 以内でな ければならない

また高速な処理をする場合には以下の点に気を付けます

- ケーブル長は可能な限り短くする
- 1 つのバスでの GPIB ケーブルのトータル距離を 15m 以内にする
- 1 つのバスに接続されている機器の 2/3 は電源を入れる
- 同一バス上に遅い機器を接続しない

これらの点を気を付ける必要がある理由としては

- GPIB は同一バス上にある最も遅い機器に合わせてデータを転送する
- GPIB の信号レベルは TTL であり耐ノイズ性はあまり高くない
- GPIB インターフェースには誤り訂正の機能がない

といったことが挙げられます。

GPIB 機器との基本的な通信

GPIB の基本的な用語と通信の要領を説明します。

コントローラ

UV-11 は GPIB バスコントローラです。UV-11 は GPIB バスを使用する上で必要となるトーカーやリスナーの指定やインターフェースクリアを初めとする特別なメッセージを各 GPIB 機器へ送ったりする機能を持っています。基本的に 1つの GPIB バス中にコントローラは 1つしか存在してはいけません。ただしパスコントロール機能を持っているコントローラであれば複数存在することが許されています。ただこの場合でもバスを制御できるコントローラ（アクティブコントローラと言います）は 1 度に 1 台しか存在してはいけません。複数のコントローラが接続された GPIB バスには通常システムコントローラと呼ばれる特別なコントローラが存在します。このコントローラは他のコントローラがアクティブの場合でも強制的に制御権を獲得できます。UV-11 はデフォルトでシステムコントローラとなっています。

※コントローラからコントローラへ制御権を移す機能をパスコントロールと呼びます。

アドレス

GPIB 機器が通信を行うには対象となる GPIB 機器が特定できなくてはいけません。GPIB ではこれを 0～30 の GPIB アドレスを一意に割り振ることにより行っています。GPIB アドレスは GPIB 機器に予め設定されています。設定の方法は機器により違いますのでそれぞれの取扱説明書を参照してください。GPIB アドレスは同一の GPIB バスに接続される機器間で重複があってはいけません。また UV-11 も GPIB 機器ですのでアドレスを持っています。通常コントローラはアドレス 0 であることに習い UV-11 もデフォルトのアドレスは 0 となっています。

※UV-11 では 2 次アドレスのサポートは行っていません。

基本的な機器の制御

コントローラは各 GPIB 機器を制御するためいくつかのメッセージ（コマンド）を使用します。ここでは基本的な制御メッセージ（コマンド）について説明します。

・インターフェースクリア

GPIB バスに接続される全ての機器の GPIB インターフェースの状態を初期化します。初期化されるのは GPIB インターフェースだけで機器の状態は初期化されません。

・リモートイネーブル

GPIB 機器の制御をリモート状態（GPIB からの制御ができる状態）にします。

・デバイスクリア／セレクトデバイスクリア
　デバイスクリアは GPIB バスにつながる全ての GPIB 機器の状態を初期化します。セレクトデバイスクリアは特定のGPIB 機器を初期化します。

・デバイストリガ
　測定の開始など、GPIB 機器にトリガを掛けます。

・ローカルロックアウト
　ローカルからの操作（例えばパネルからの操作）を禁止します。

・ゴーツーローカル
　ローカルからの操作を可能にします。

トーカーとリスナー

GPIB 機器が通信を行うためには、データを送る側と受け取る側が予め指定されていなければなりません。GPIB ではデータを送る側をトーカー、データを受け取る側をリスナーと呼んでいます。トーカーとリスナーの指定は GPIB バスにトーカーアドレスとリスナーアドレスを送出することにより行われます。UV-11 の制御ライブラリではトーカーとリスナーの指定は以下のように行います。

　GpibSetTalkerAddrees(0～30)
　GpibSetListenerAddress(0～30, 1 or 0)
　 他の機器のリスナー指定を解除する場合は 1 解除しない場合は 0)

トーカーは一度に 1台しか存在してはいけませんが、リスナーは同時に複数存在しても構いません。同じ機器が複数台接続される場合、同時にリスナーに指定して設定などをまとめて行うような使い方ができます。

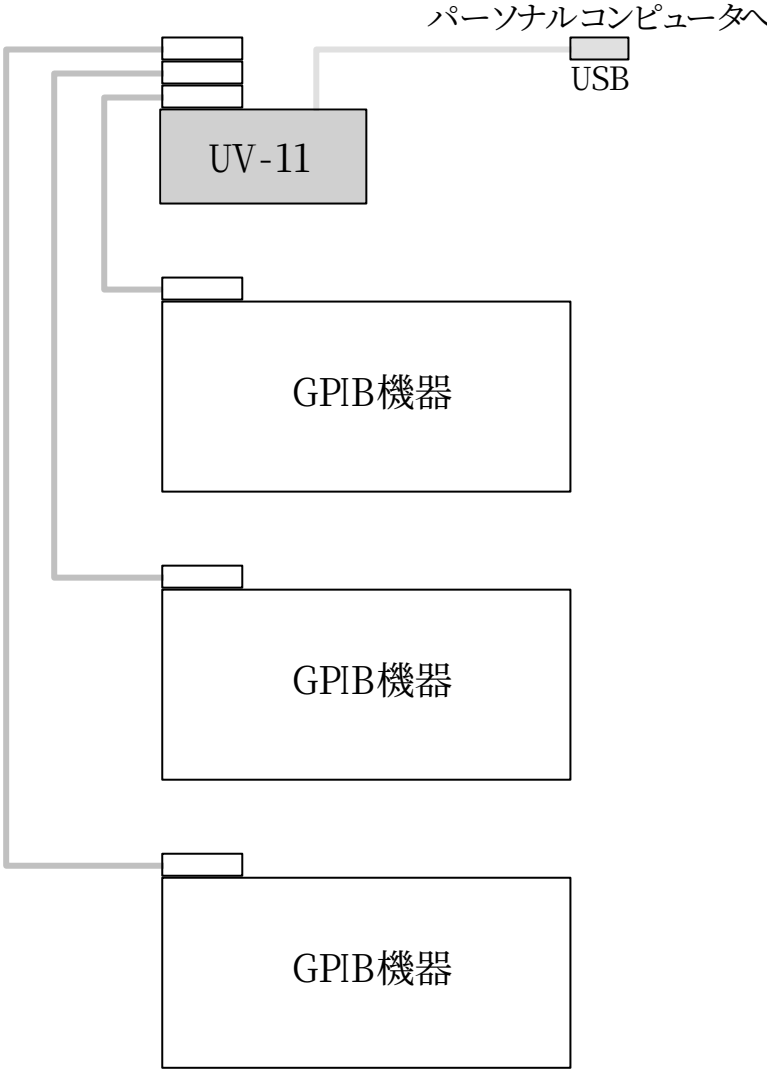
データとデリミター

GPIB バスで扱われるデータは基本的に 8bit の任意データですが、多くの場合 ASCII 文字列が使用されます。またこの場合データの終わりを検出する方法としてある特定の文字パターンが使用されてます。このパターンをデリミター と呼んでいます。デリミターは通常 1 ないし 2 文字で構成されます。UV-11 の場合このデリミターを送信と受信でそれぞれ設定できます。また文字列以外のバイナリーデータの通信を行うためデリミターをなしにすることができます。この場合データの終わりは EOI メッセージで検出します。UV-11 は送受信時に EOI の送出と検出を行うようにしています。殆どの GPIB 機器は EOI を送受信の終了として認識又は送出を行いますが、中には EOI を送出しない物もあります。

GPIB の接続形態

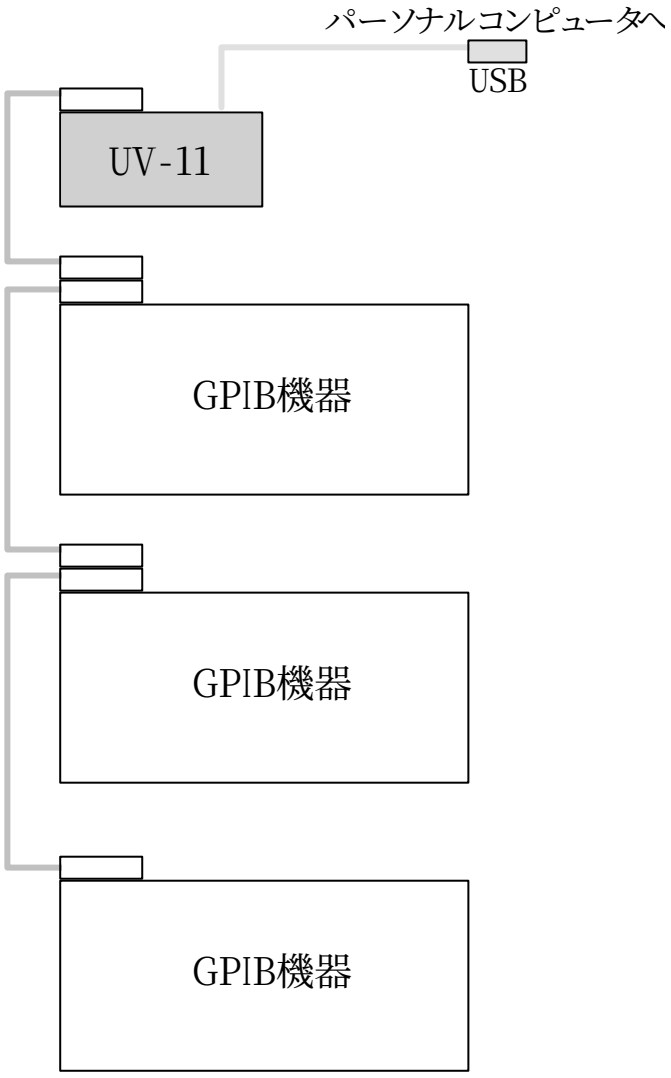
スター接続

全ての GPIB 機器を UV-11 に直接接続する形態です。



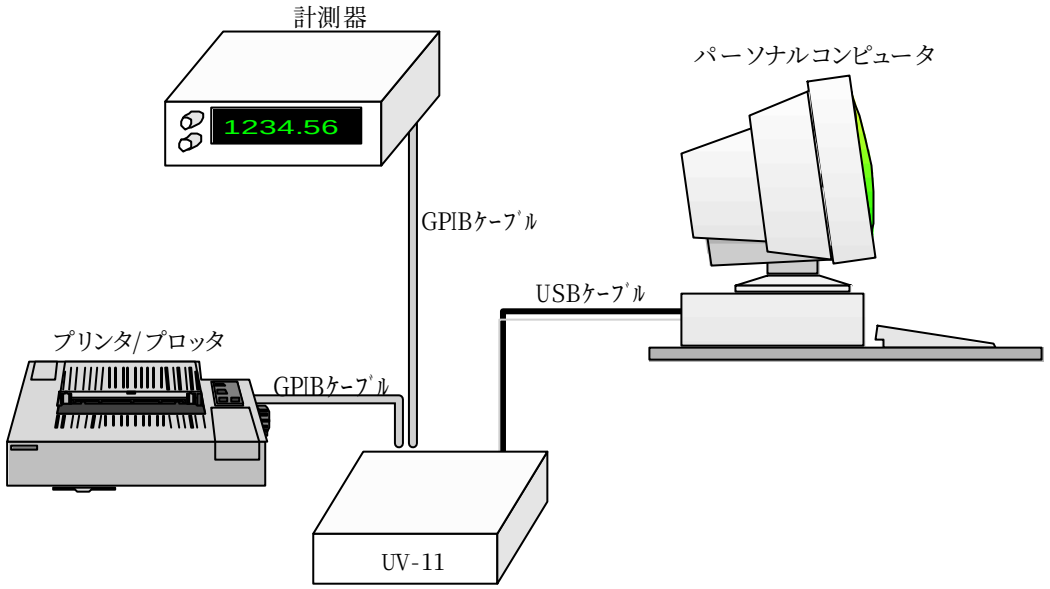
デイジーチェーン接続

UV-11 を基点として機器間を順番に接続していきます。



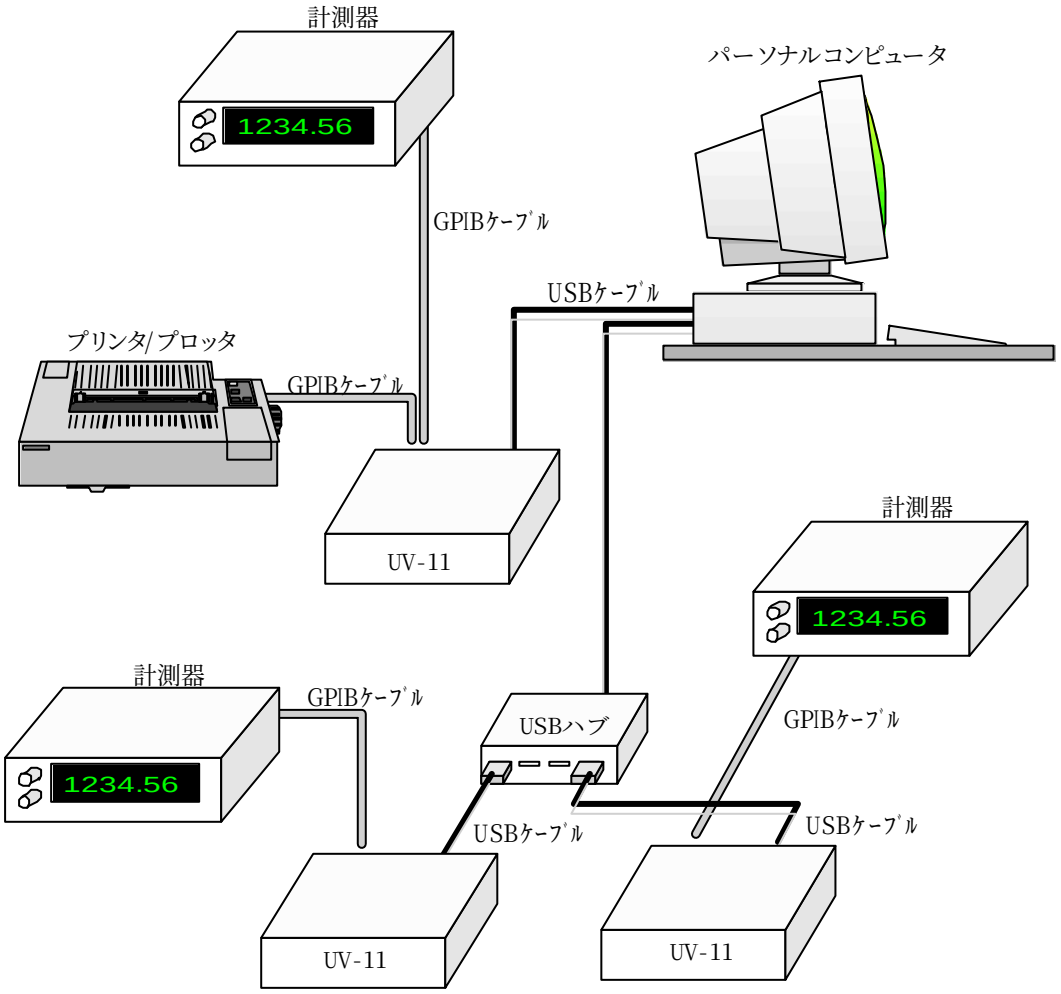
基本的な UV-11 の接続

UV-11 を使用してシステムを構築する場合必ず 1台以上の USB インターフェースを備えたパーソナルコンピュータが必要となります。必要な要件に関しては「ホストコンピュータの条件」項を参照してください。下図は最も典型的な構成を表しています。
UV-11 は USB ケーブルでパーソナルコンピュータへ接続するだけで使用できます。UV-11 の GPIB インターフェースへ利用したい GPIB 機器を接続します。GPIB ケーブルの接続は UV-11 を基点にデイジーチェーン方式で接続する方法と UV-11 へ全ての GPIB ケーブルを接続するスター形式の接続方法があります。何れの接続方法でもお客様のご都合に合わせ選択できます。



複数台の接続

下記の図は 3 台の UV-11 を1 台のパーソナルコンピュータへ接続した場合です。
パーソナルコンピュータに USB コネクタが 2 つついていることを想定しています。
もし1 つしかない場合は全ての UV-11 をUSB ハブ経由で接続することもできます。
下図の場合各 UV-11 は独立して GPIB バスに接続されていますので、最大 42 台の GPIB 機器を制御することが可能です。また下図では USB インターフェースに UV-11 しか接続されていませんが、他の USB 機器を同時に接続することも可能です。
UV-11 は 1 台の PC に最大 10 台まで接続できます。UV-11 は接続が成立した順に自動的にデバイス名が割り振られます。デバイス名は UV11-X(X は 0～9)となります。
このデバイス名を直接ユーザーが使うことはありません。制御ライブラリの関数を使用することにより、操作対象となるデバイスを選択する方法によりユーザーは複数の UV-11 を制御することができます。



ホストコンピュータの条件

ハードウエア要件

IBM-PC-AT とその互換機で USB インターフェースを有するもの。
※NEC 社 PC98 シリーズはサポートしておりません。

ソフトウエア要件

Microsoft Windows98/SE(日本語版)
Microsoft WindowsMe(日本語版)
Microsoft Windows2000Professional(日本語版)

※Windows2000Server 及び AdvancedServer での動作は保証致しかねます。
※日本語版以外での動作は保証致しかねます。
※WindowsNT(4.0 も含む)についてはサポートしておりません。
※Apple 社 Macintosh についてはサポートしておりません。
※その他上記に明記します OS 以外はサポートしておりません。

電源

UV-11 は USB バス電源を必要とします。USB バス電源とは USB ケーブルによりホストコンピュータ又は USB ハブより供給される電源の事です。規格上 USB1 ポートで供給できる電源の最大電流は 500 ミリアンペア/1 ポートです。UV-11 は 1 台あたり約 300 ミリアンペアの電流を消費します。このため 1 ポートに 1 台の UV-11 しか接続できません。同一のポートに他の機器を接続した場合供給される電流の不足により動作しない、または不安定になる場合があります。他の機器とのポートの共有状態での動作は保証致しかねます。
またハブの場合自己電源ではなくUSB バス電源で動作する物が有りますがこのようなハブに接続して使用する場合、UV-11 が動作するのに必要な電源を供給できる物である事を確認してください。

※個別の USB ハブとの動作確認は行っておりません。上記の動作条件はあくまで目安でありいかなる動作保証をする物ではありません。
※USB ハブによってはバス電源で動作時のバスへの供給電圧が低くUV-11 が正常に動作しない物もあります。このような場合にはセルフパワードタイプのハブをご利用頂けますようお願い致します。

設置に関して

UV-11 には設置方向等はありません。弊社としては水平に設置した状態を標準と考えていますが垂直に設置されても、逆さまに置かれても機能的には問題ないと考えています。ただし熱を発しますので放熱には十分気を付けてください。高温になる場所での使用はおやめください。

使用環境

電源	AC100V 50/60Hz
温度条件	5℃～40℃
湿度条件	20%～80% 結露しないこと)

外観とその説明

UV-11 には電源スイッチが有りません。USB ケーブルに接続され必要な電源が供給されると動作します。また完全に停止させたい場合は USB ケーブルを外してください。背面には USB コネクタ、GPIB コネクタ及び動作状態を確認するための LED ランプがあります。UV-11 に電源が供給されると最初 LED ランプが赤く点滅します。この状態は電源が投入され UV-11 自身の動作準備が整った状態を示します。その後 LED ランプが緑色に点滅します。これは UV-11 がパーソナルコンピュータから認識されソフトウエアによる制御が可能になった事を示します。USB ポートを持ったパーソナルコンピュータでも USB をサポートしていない OS(例えば WindowsNT4.0 など)が動作している状態で接続した場合には緑色にはなりません。パーソナルコンピュータを再起動した場合はパーソナルコンピュータによりますが一端給電が停止されますので、ケーブルを最初に接続した状態に戻ります。

※再起同時の給電状態についてはパーソナルコンピュータにより異なる動作をする物もあります

UV-11 についている USB コネクタは B タイプ用です。USB ケーブルの細い方を接続してください。

USB コネクタの抜き差しについて

USB コネクタは電源が入った状態のまま抜き差しする活線挿抜に対応しています。基本的に任意のタイミングで抜き差しを行うことができますが、UV-11 を利用したソフトウエアが動作している際に、ケーブルを抜き差しした場合予期せぬ不具合が発生する可能性があります。抜き差しされる際は UV-11 を利用するソフトウエアが動作状態にないことを確認して行うことを強くお勧めします。

GPIB コネクタの抜き差しについて

GPIB コネクタは活線挿抜には対応していません。電源が入ったまま抜き差しした場合 GPIB インターフェースが破損する可能性があります。GPIB ケーブルを抜き差しするときは一旦 UV-11 を USB ケーブルから外しの電源を OFF するようにしてください。
また物理的に大きな力が加わらないよう気を付けてください。取り付けネジをドライバで強く締めるとコネクタが破損する場合もありますので注意してください。

初めて使う

- 箱から出して使えるようになるまで

梱包内容の確認

箱を開けましたらまず、梱包内容をご確認ください。内容の詳細は本取扱説明書の3 ページに記載されています。
品質には万全を期しておりますが万が一不足品がありましたらお手数ですが弊社サポートへご一報頂けますようお願い致します。

GPIB 機器との接続

GPIB 機器とは必ず UV-11 に電源が入っていない状態で行ってください。電源の入ったパーソナルコンピュータと USB ケーブルで接続すると自動的に UV-11 に電源が入ります。このため GPIB 機器との接続は UV-11 をパーソナルコンピュータに接続する前に行うようにします。

パーソナルコンピュータへの接続

UV-11 とGPIB 機器との接続が終了したらUV-11 をパーソナルコンピュータと接続します。まず添付の USB ケーブルの幅広のコネクタの方をパーソナルコンピュータに接続してください。この際パーソナルコンピュータの電源は入っていてもいなくてもどちらでも構いません。次にUSB ケーブルの細い方のコネクタを UV-11 に接続します。このときパーソナルコンピュータの電源が入っていなければ入れてください。入っていればそのままデバイスドライバのインストールに進みます。

デバイスドライバのインストール

UV-11 をパーソナルコンピュータに接続するにはデバイスドライバのインストールが必要となります。UV-11 をパーソナルコンピュータに最初に接続すると自動的にそのインストールを求められます。インストールの手順は以下のようになります。

※以下の手順は Windows2000 でのものですが、Windows98/Me でもほぼ同様な手順になります。

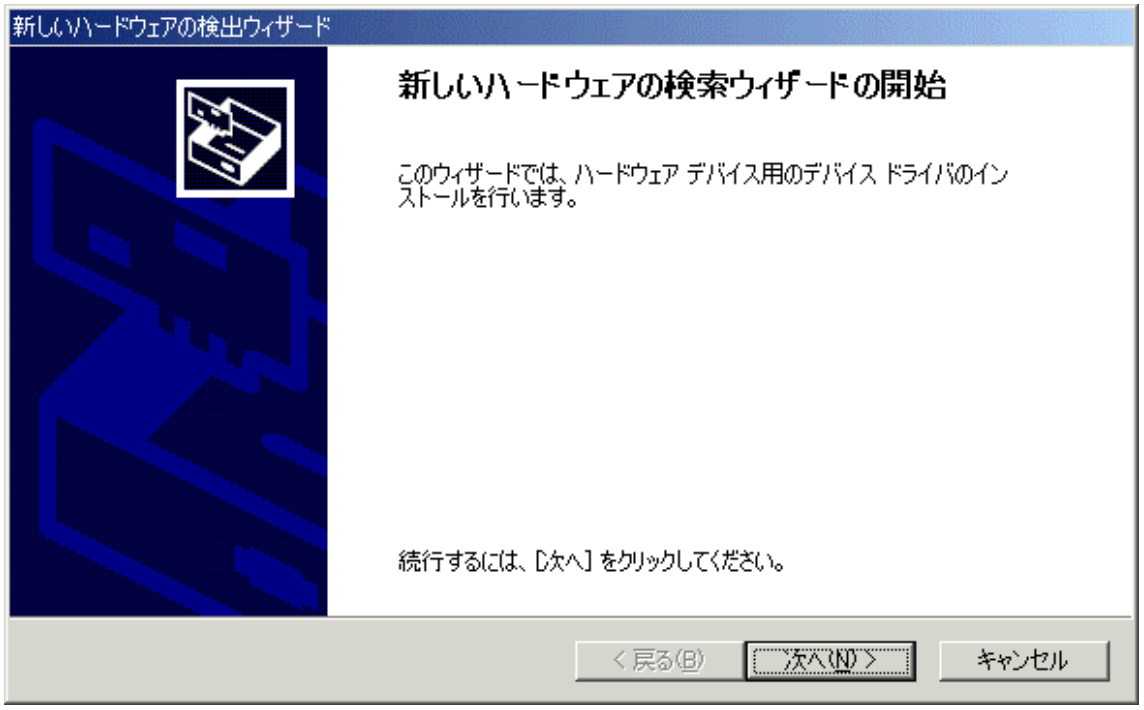
新しいデバイスの検出

初めて UV-11 をパーソナルコンピュータに接続すると下記の様な画面が表示されます。

続けて検出された新しいハードウエア用のデバイスドライバをインストールするための検索ウイザードが開始されます。

検索ウイザード

新しいハードウエア用のデバイスドライバを検索するためのウイザードが自動的に起動します。ここでは 次へ」ボタンを押します。



検索方法の指定

次に検索方法を聞いてきますので「デバイスに最適なドライバを検索する」を選択して 次へ」ボタンを押します。

ドライバファイルの特定

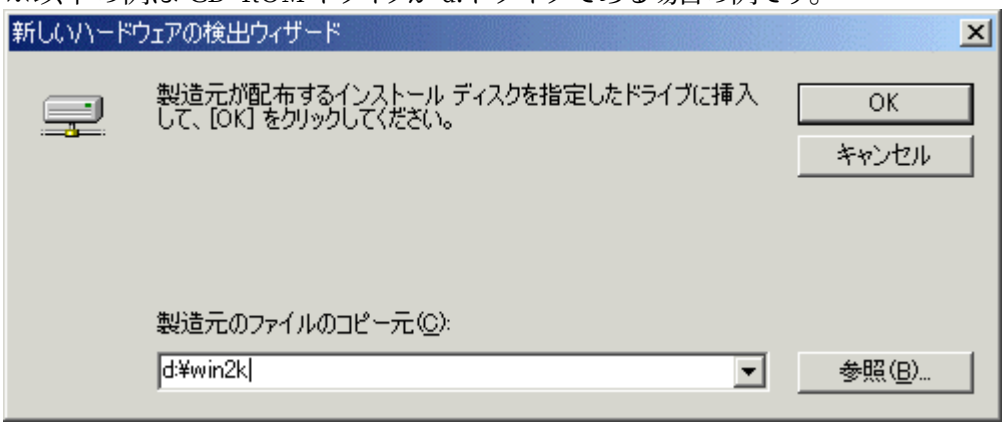
次にドライバファイルを検索する場所を聞いてきます。この時本製品添付の CD-ROM からドライバを読み込ませる必要があります。以下ののように 場所を指定」を選択して 次へ」ボタンを押します。

添付 CD-ROM のセット

ここで本製品添付の CD-ROM をパーソナルコンピュータの CD-ROM ドライブにセットして UV-11 のデバイスドライバの読み込みができるようにします。

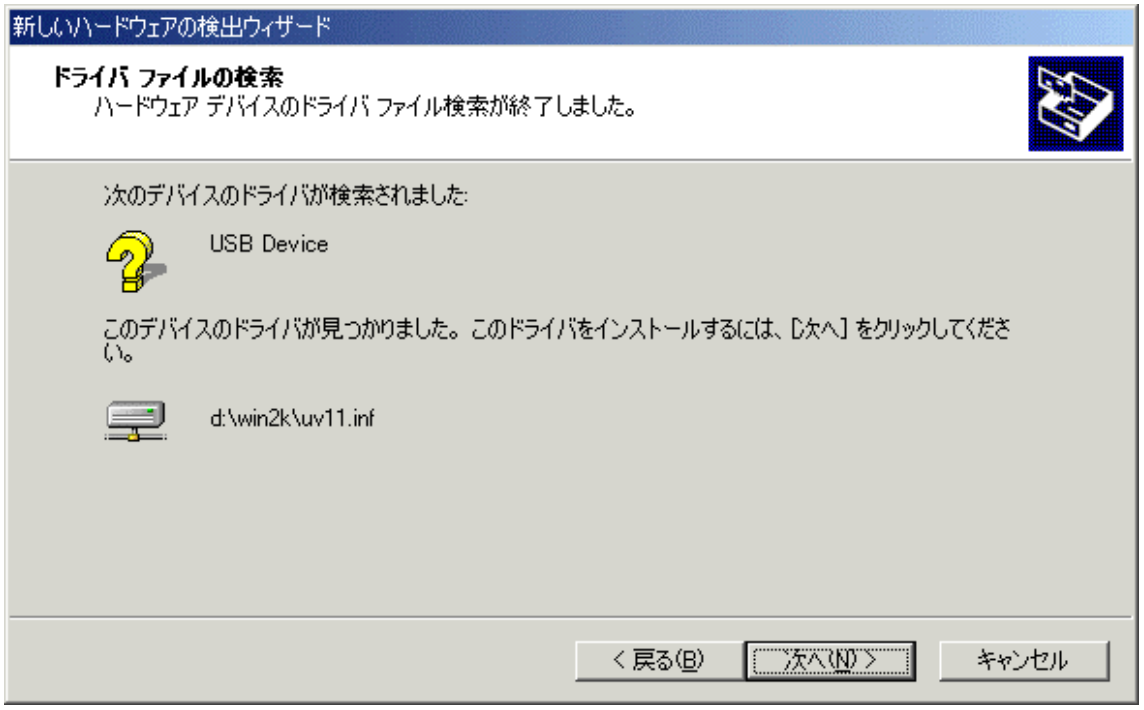
コピー元の指定

CD-ROM をセットしたら「コピー元」を以下のように指定し、「OK」ボタンを押します。
※以下の例は CD-ROM ドライブが d:ドライブである場合の例です。



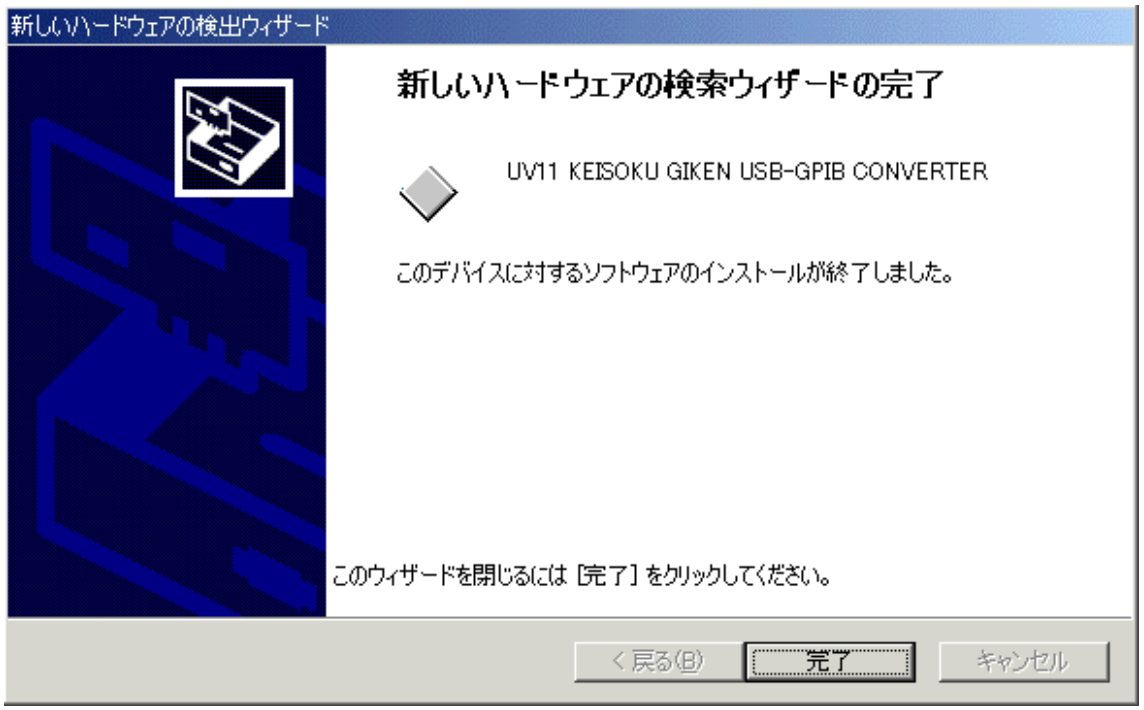
インストールの実行

コピー元を指定すると以下のような画面になりますので「次へ」ボタンを押してインストールを実行します。



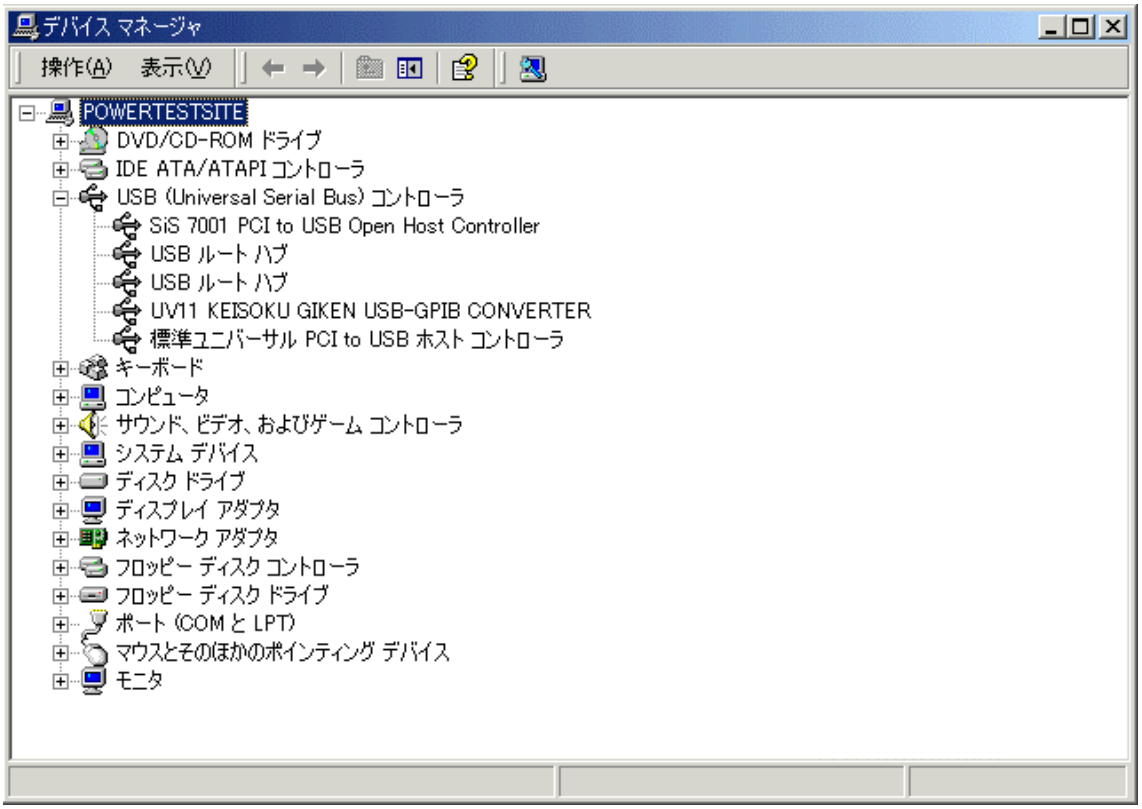
インストール完了

正常にインストールが終了すると以下の画面が表示されます。



インストール結果の確認

正しくインストールされた場合 Windows のデバイスマネジャにより下図のように UV-11 用のデバイスドライバが正しくインストールされたことが確認できます。



※上記の表示は UV-11 を接続していない場合には表示されなくなります。

動作の確認

上記まで確認できた場合は UV-11 用ソフトウエアのセットアップを実行し、簡易 GPIB コントールソフトを使用して製品名やシリアル番号などの製品情報を確認します。これらの情報が正しく読みとれれば UV-11 はパーソナルコンピュータから正しく認識されています。

ソフトウエア

- 構成と使い方

添付されるソフトウエア

UV-11 には標準でユーティリティと制御ライブラリのソフトウエアが添付されています。

UV11.DLL

通常の DLL (ダイナミックリンクライブラリ) です。VisualC++等で使用できます。このライブラリは 32bit システム専用です。またこの DLL は GPIBC.EXE でも使用します。

UV11FUNC.H

C/C++用の関数定義ファイルです。UV11.DLL に含まれる関数の定義が含まれています。C/C++ソースコードファイルの先頭でインクルードするようにしてください。

UV11.LIB

UV11.DLL のリンク用ファイルです。VisualC++等で UV11.DLL を利用する際このファイルをリンクの対象とします。

UV11.OCX

UV-11 を制御するための ActiveX 形式のモジュールです。このモジュールは VuslaBasic を初めとする ActiveX を利用できる全てのソフトウエアから利用できます。このライブラリは予めシステムに登録して使用します。

UV11.TLB

UV11.OCX を VisualC++などから使用するためのタイプライブラリです。

GPIBC.EXE

UV-11 を使用して GPIB 機器を直接制御する為のプログラムです。UV-11 が正常に動作しているか確認することができます。また実際に制御プログラムを書く前に制御対象となる機器の動作をインタラクティブに見ることができます。

SWGPIB.DLL

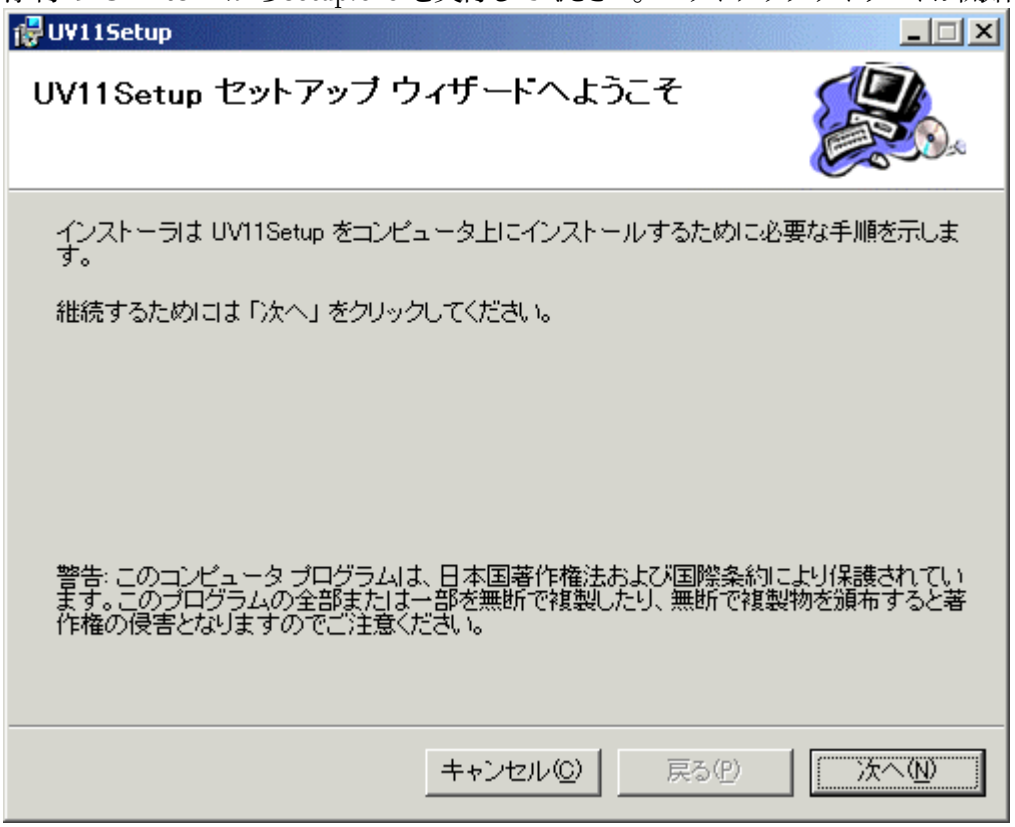
弊社販売の PW-6000 用のデバイスドライバです。UV-11 単体でお買いあげの場合は不要になります。

サンプルプログラム

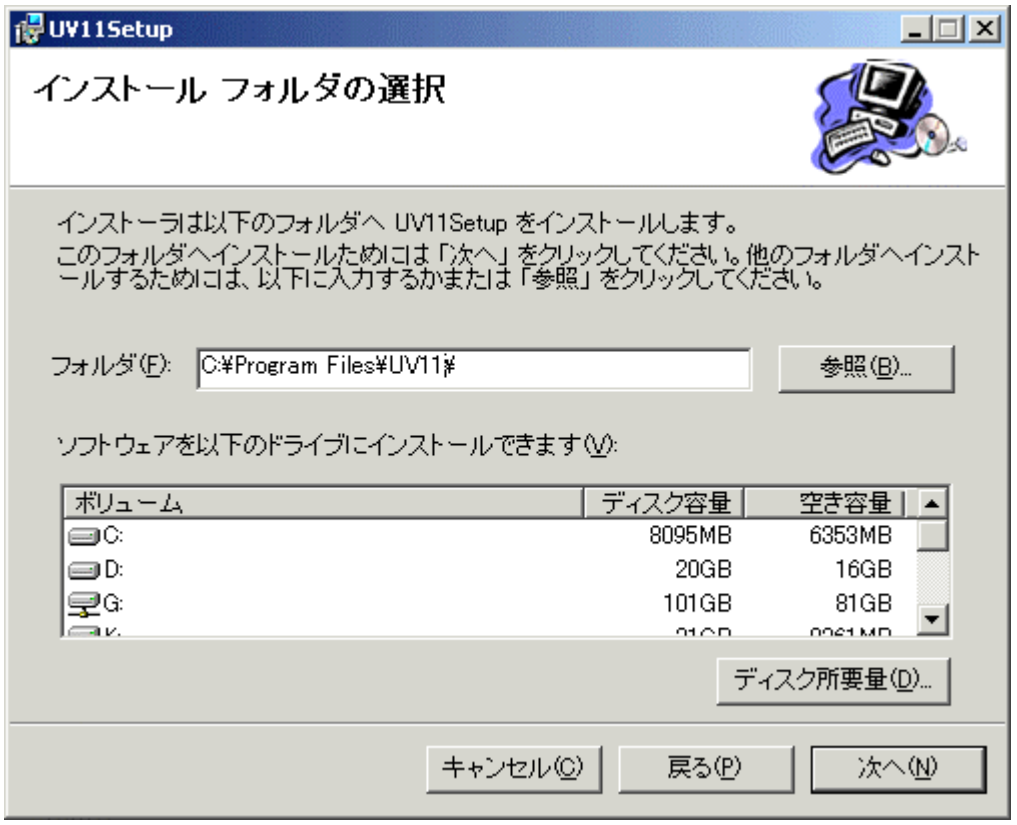
Excel から UV11.OCX を利用するプログラムを Excel ファイルの形式で添付しています。
また弊社電子負荷の EL-302 を制御する VisualBasic で作成した制御パネルも添付しました。

ソフトウエアのインストール

添付の CD-ROM からsetup.exe を実行してください。セットアップウイザードが開始されます。



下記の様にインストール先を聞いてきますので任意の場所を指定して 次へ」ボタンを押します。



以上でインストールは完了です。

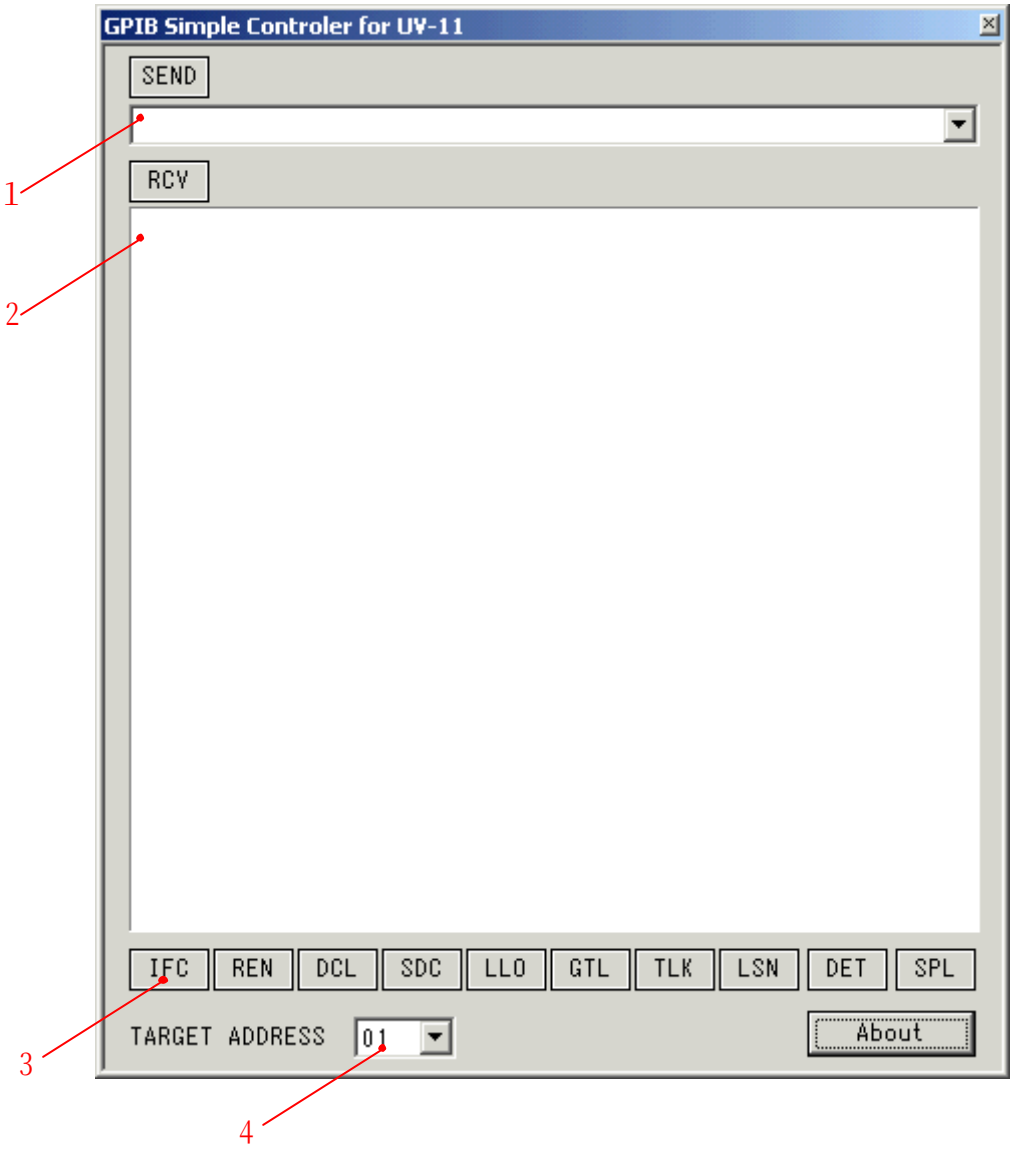
GPIBC.EXE を使って GPIB 機器を直接制御する

インストール結果の確認

インストールが正常に終了していれば直ちに UV-11 を利用できるようになります。
指定したインストール場所に GPIBC.EXE という名称の確認用のソフトウエアがインストールされていますのでこれを使用して動作を確認することができます。

GPIBC.EXE の起動

GPIBC.EXE をダブルクリックして起動してください。GPIBC.EXE は UV10.DLL を使用します。もしコピーして別の場所で利用される場合は UV10.DLL もペアでコピーしてください。



各部の説明

1.送信テキストコンボボックス
GIIB 機器に送信する文字列をキーボードから入力します。過去に送信した履歴を選択することができます。
2.受信テキストリスト
受信した GPIB 機器からの文字列や受信送信に関わるステータスを表示します。
結果は後から追加されてゆき、画面一杯になるとスクロールします。
3.制御機能ボタン

IFC	インターフェースクリア
REN	リモートイネーブル
DCL	デバイスクリア
SDC	セレクトデバイスクリア
LLO	ローカルロックアウト
TLK	トーカーに指定
LSN	リスナーに指定
DET	デバイストリガ
SPL	シリアルポーリング

4.操作対象となる機器のアドレスの設定
制御機能ボタンを含め、送受信の対象となる機器のアドレスを選択します。

UV-11 の製品情報を見る

GPIBC.EXE の About 画面に接続されて UV-11 の製品情報が表示されます。UV-11 を実際に接続して About ボタンを押してみてください。下記のような画面が表示されれば正常に動作しています。

GPIBC.EXE の使い方

例として UV-11 にアジレントテクノロジ社の34401A デジタルマルチメータを制御します。

接続

最初に UV-11 をパーソナルコンピュータに USB ケーブルで接続する前に34401A とUV-11 を GPIB ケーブルで接続します。この際 GPIB 機器の電源は入れない状態で作業することをお勧めします。接続が終わったらUSB ケーブルをUV-11 に接続して利用出来る状態にします。
USB ケーブルが接続された際 UV-11 の LED が緑色に点滅していることを確認してください。

初期化

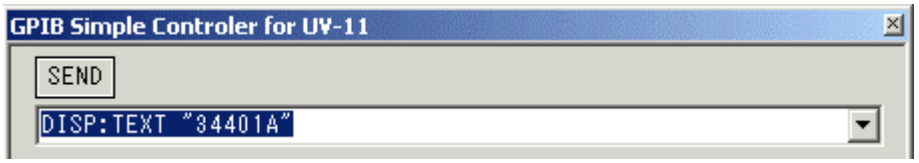
GPIBC.EXE を起動します。正常に UV-11 が動作していれば直ちに GPIBC.EXE は起動します。
まずインターフェースクリアを実施して GPIB インターフェースを初期化します。インターフェースクリアが無くても電源投入後、動作する機器もありますがインターフェースの状態を初期化するため GPIBC.EXE の IFC ボタンを押してインターフェースクリアを実施します。その他必要に応じて REN LLO などを実施してください。

アドレスの設定

GPIB 機器を制御するためにはその GPIB アドレスが必要です。GPIB アドレスを知るには直に GPIB 機器を操作して GPIB 機器に設定されているアドレスを調べる必要があります。設定方法は機器毎にことなるため各機器の取り扱い説明書を参照して調べてください。ここではアドレスが 12 番であるとして例を示します。12 番のアドレスの機器を操作する時は GPIBC.EXE のアドレス設定を以下のようにします。

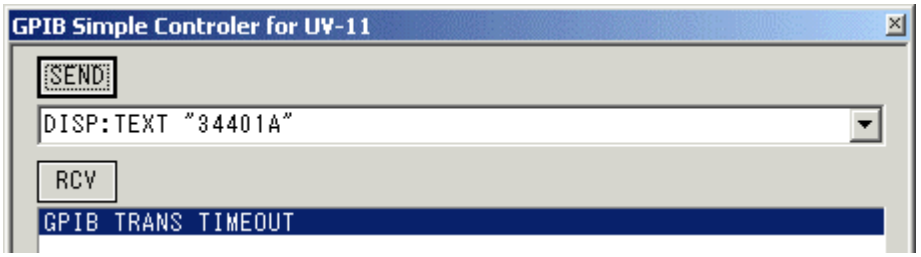
コマンド送信

試しに送信が正しくできるか、34401A の表示パネルに任意の文字列を表示してみます。

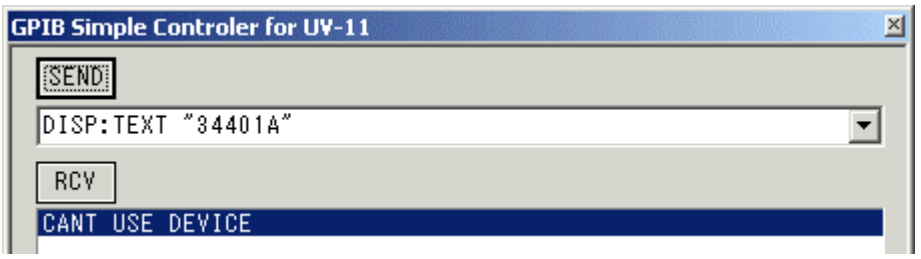


SEND ボタンを押すと、正しく送信できれば34401A の表示パネルに 34401A の文字が表示されます。

下記のように出る場合はアドレスが正しいか再度確認してください。

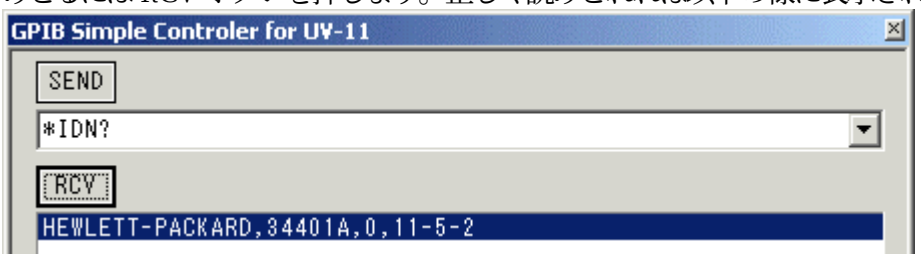


下記のように表示される場合はインターフェースが初期化されていない場合がありますのでIFCボタンを押して初期化します。



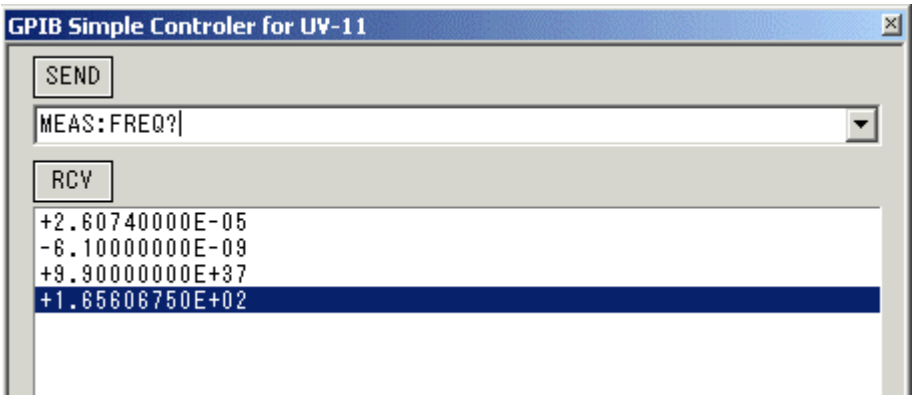
データの受信

ここでは34401Aのid文字列を受信してみます。先ほどと同じようにして*IDN?コマンドを送ります。34401Aに正しく送信されていればID情報の送信状態になっているのでデータを読みとります。読みとるにはRCVボタンを押します。正しく読みとれれば以下の様に表示されます。



測定

ここまで正しく動作すれば34401Aの制御は正常に行われていますので、実際に測定してデータを取ってみます。



上記の例では

MEAS:VOLT:DC? -> RCV
MEAS:CURR:DC? ->RCV
MEAS:RES? ->RCV
MEAS:FREQ? ->RCV

と連続して送信受信を行った結果を示しています。
上記の様にコマンド送信->データ受信というのが一つの制御パターンですが、中には連続して測定値を返すモードなどもあります。この場合はデータ受信を連続して実施することにより測定値を得ます。

MicrosoftExcel2000 からUV-11 を使って計測値をシートに取り込む

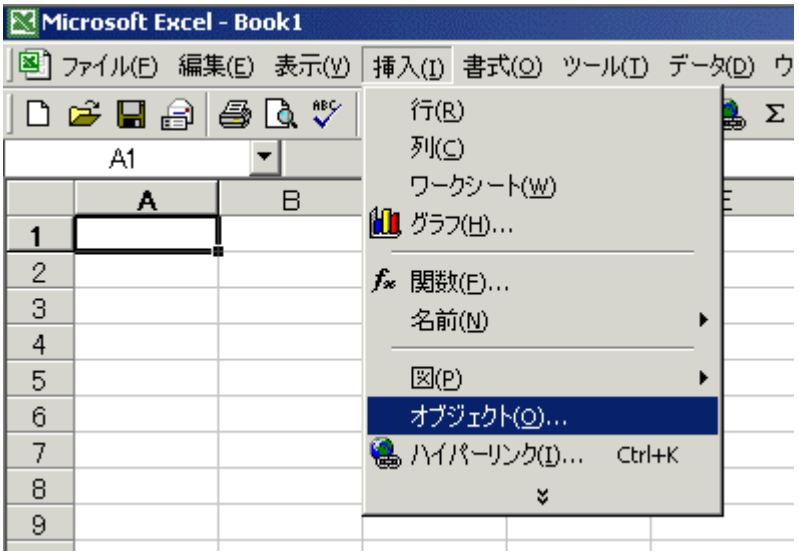
次に実際に UV-11 を利用する方法について説明します。
具体的な例として Microsoft 社の Excel2000 を利用して GPIB 接続されたデジタル電圧計から電圧値を取り込みワークシート上に表示させます。

UV-11 を制御するモジュール

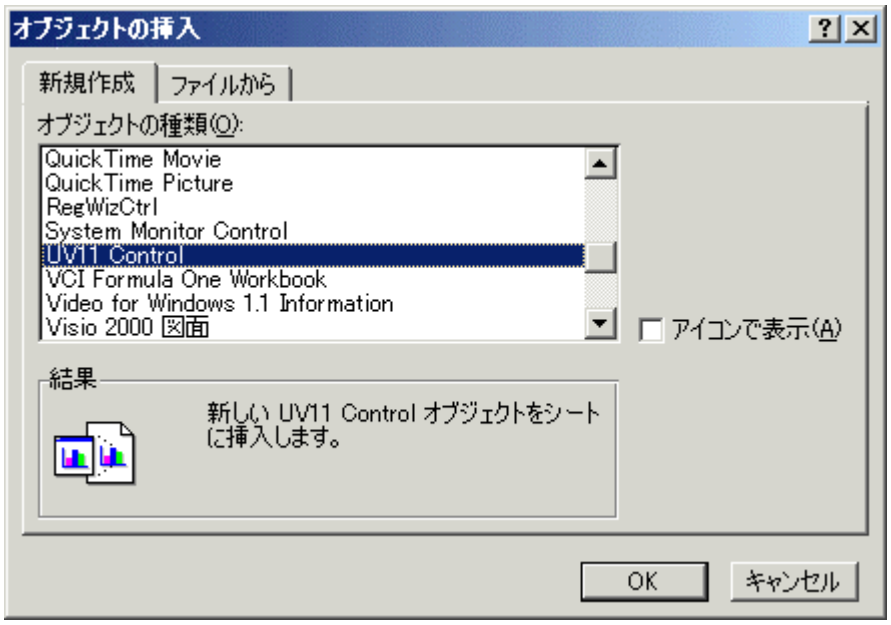
ExcelからUV-11を制御するにはUV11.OCXを使用します。このファイルは先ほどのインストールにより使用可能な状態になります。インストールが済んでいない場合はまずインストールを実施してください。

UV-11 制御コントロールの挿入

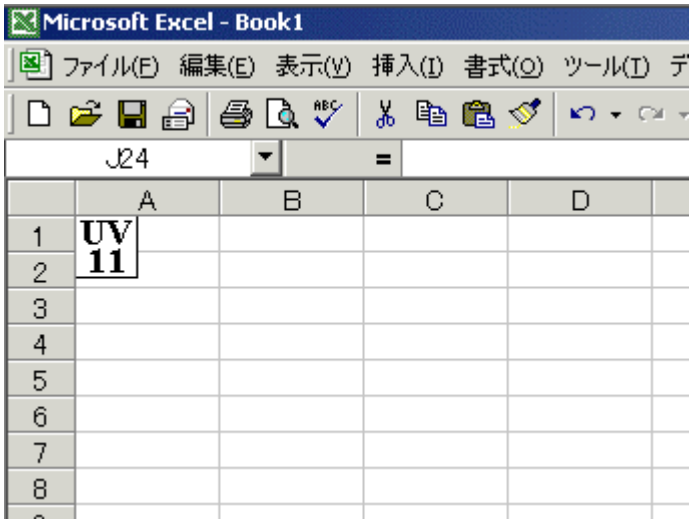
ワークシートに UV11 制御コントロールを挿入します。下記のようにメニューの挿入からオブジェクトを選択します。



オブジェクトの選択メニュー画面からUV11 Controlを選択し、「OK」ボタンを押します。

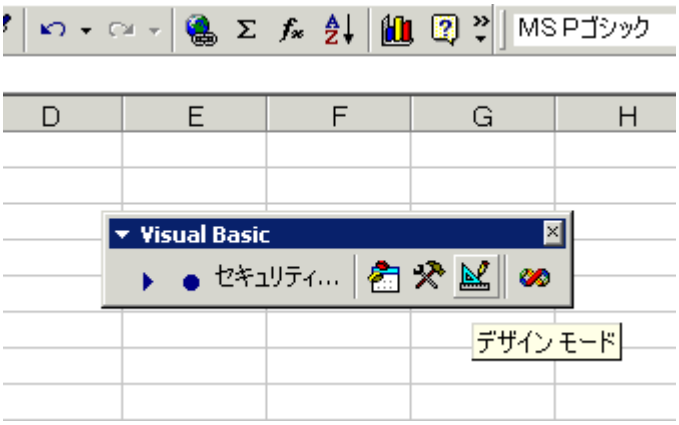


挿入されたオブジェクトは下記の様にアイコンで表示されます。



挿入されたオブジェクトには自動的にUV111という名称が付けられます。以降この名称を使用してプログラムからUV11オブジェクトを使用することになります。なお2個目以降挿入される順に名称はUV112、UV113と付けられます。

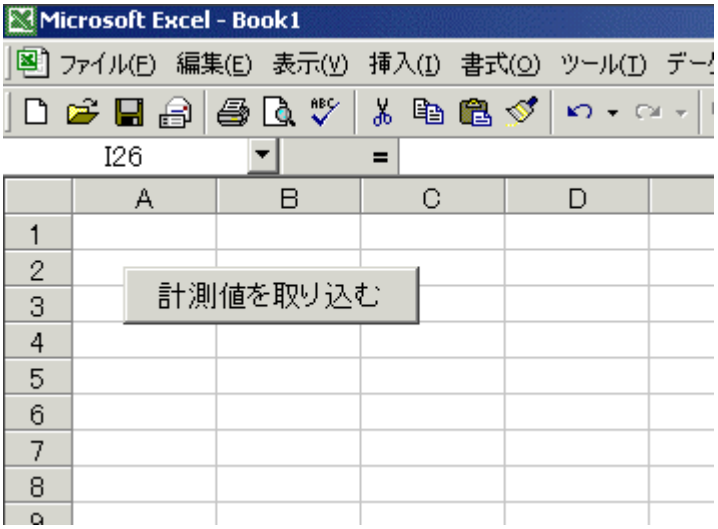
このアイコンはワークシート上のどこに配置されても構いません。表示の邪魔にならない場所に移動してください。アイコンを移動する際はViaulBasicツールバーでデザインモードにします。



デザインモードを終了するとアイコンは表示されなくなりますが、これは正常な動作です。

取り込み動作を開始するためのボタンを作成する

今回は計測器からの取り込み動作を行う開始の指示を与えるきっかけを与えるためにボタンを作成します。このボタンはあくまでサンプルでありボタンでなければ取り込みができないと言うことではありません。



取り込みプログラムを書く

ボタンが押されたら計測値を取り込むためのプログラムを記述します。この例ではVisualBasicで記述します。
UV-11の機能を利用するには先ほど挿入したUV11オブジェクトのUV111を使用して
UV111.XXXという形式で機能を呼び出すようにします。XXXの場所には使用したい機能の名称が入ります。

[Sample]

以下のサンプルではアジレントテクノロジ社のデジタルマルチメータHP34401Aを使用して直流電圧値を測定しこの値をsheet1の先頭のセルに代入する例です。

```
Private Sub CommandButton1_Click()
    UV111.GpibInterfaceClear                    ' インターフェースクリアを送出
    UV111.GpibRemoteEnable                      ' リモート状態に移行

    UV111.GpibSendString 1, "MEAS:VOLT:DC?"     ' 電圧測定コマンドを送信

    Dim v As Variant                            ' 受信データを受け取るための変数
    UV111.GpibReceiveString 1, v, 30            ' 電圧値の受信
    Sheet1.Cells(1, 1) = CSng(v)                ' 受信データをセルに代入
End Sub
```

上記の例ではまずボタンが押されるとUV-11からGPIBバス制御命令のインターフェースクリアとリモートイネーブルが送出されます。これによりGPIB機器を利用するための準備を行います。
次に電圧測定を行うためのコマンドを送信します。最初の引数の1はGPIBアドレスですのでご利用になる機器に合わせてアドレス値を変更してください。2番目の引数がコマンドです。このコマンドは利用する機器によりそれぞれ異なります。上記の例はHP34401Aのコマンドですが、使用する機器、機能がことなれば送信するコマンドも変更する必要があります。
次に測定値を受け取ります。受け取るデータは一端Variant型の変数で受け取る必要があります。
Dim v As Variantの行は受け取り用の変数の定義を行っています。
次に実際にデータを受け取ります。最後の引数に30を指定しているのは受け取りに必要と予想される最大文字数です。この値は多めに設定する分には問題ありません。
最後に受け取ったデータをセルに代入しています。Sheet1というのはワークシートの最初のシートに付いているデフォルトの名称です。この例では最後に単精度の実数に変換して代入しています。

また実行中、エラーが発生した場合にはエラーイベントが発生していますエラー内容を表示するには以下のようなプログラムを追加します。

```
Private Sub UV111_UV11Error(ByVal ErrorCode As Long, ByVal ErrorMessage As String)
    MsgBox ErrorMessage
End Sub
```

UV-11 で作成したプログラムを移植するには

UV-11の前のモデルであるUV-11を使用して作成したプログラムの移行は制御するオブジェクトを入れ替えることにより行います。
UV-11の制御プログラムはUV10Xというオブジェクトを作成し、このオブジェクトの機能を利用することにより作成されていると思います。このオブジェクトとUV11Xオブジェクトはプロパティ、メソッド共に完全な互換性があります。このためオブジェクトをそっくり入れ替えることでUV-11用に作成したプログラムをUV-11に対応させることができます。

UV-11 用のオブジェクトを作成する

移植したいプログラム上にUV11制御オブジェクトを作成します。

UV-11 用のオブジェクトを削除する

元々あるUV-11用のオブジェクトを削除します。このとき削除する前にこのオブジェクトの名称を確認、記録しておいてください。

UV-11用オブジェクトの名称を削除したUV10用オブジェクトの名称に変更する

先に作成しておいたUV-11のオブジェクト名を、削除したUV-11用のオブジェクト名に変更します。
たとえばUV-11用のオブジェクトの名がUV101ならばUV-11用のオブジェクト名（デフォルトでUV111、UV112…となっています）をUV101に変更します。

以上でUV-11用に作成したプログラムをUV-11で動作させることができます。

ActiveX メソッドリファレンス

UV-11 制御ActiveX の提供するメソッドのリファレンスです。
メソッドの引数及び戻り値はVisualBasic のデータ型に合わせて記述しています。

GetInfomation(*InfomationNo* As Integer,*rVal* As Variant)As long

説明

シリアル番号などの UV-11 の情報を得ます。
情報は全て 8 文字固定の文字列で返されます。

引数

InfomationNo

情報の種類を指定します。

0

機器の名前を返します。UV-11 の場合"UV-11"という文字列(固定)が返されます。

1

USB ベンダーID とUSB プロダクトID を返します。UV-11 の場合"06711002"という文字列(固定)が返されます。0671 が弊社に割り当てられているUSB ベンダーID です。
1002 は弊社で UV-11 に割り当てたプロダクトID です。いずれも16 進数で表記されています。

2

プロダクトリビジョンとファームウエアのバージョン番号を返しますプロダクトリビジョンとは UV-11 のハードウエア部の世代を示す番号です。

3

シリアル番号を返します。例えば"23110001"のような 8 桁の番号となります。
上記以外の値は無効(エラー)となります。

rVal

戻される文字列の格納先の変数を指定します。バリアント型の変数を渡すと文字列形式で返します。戻される文字列は常に 8 文字固定です。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCode **プロパティ**で内容を知ることができます。

サンプル

```
Dim a As Sting *8
UV111.GetInfomation(0,a)'デバイス名を得ます
Text1.Text =a
```

GpibBusTimeOut(*time* As long)As long

説明

GPIB バスタイムアウト時間を設定します。データ出力の遅いデバイスからデータを読み込む場合などはこの時間を大きく設定します。

引数

time
タイムアウトまでの時間をmsec 単位で指定します。
10msec ～60000msec の間で 10msec 刻みで指定できます。
10msec 未満の端数は切り捨てになります。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCode プロパティで内容を知ることができます。

備考

UV-11 の電源投入時の値は、5000msec です。

サンプル

UV111.GpibBusTimeOut(1000)' タイムアウト時間を１秒に設定します

GpibDeviceClear As long

説明

全てのGPIB 機器に有効なにデバイスクリアメッセージを送出します。
特定のGPIB 機器にデバイスクリアメッセージを送るときはGpibSelectDeviceClear メソッドを使用します。

引数

ありません。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCode プロパティで内容を知ることができます。

GpibDeviceSelect(*DeviceNo* As Integer)As long

説明

UV-11ActiveX の制御の対象となるUV-11 を選択します。UV-11のメソッドは現在選択されているUV-11にのみ作用します。UV-11 は同時に10 台までの接続をサポートしています。複数のUV-11 を個別に制御するためには、制御の対象となるUV-11 を選択する必要があります。
選択された状態は次に選択を変更するまで有効です。 1 度選択すれば同じUV-11 を制御する限り、再びこの関数を実行する必要はありません。またデフォルトで最初に接続されたUV-11 が選択されていますので、 1 台しか接続しない場合はこの関数を実行する必要はありません。

引数

DeviceNo
デバイス番号を指定します。UV-11に割り振られるデバイス番号は接続した順に0 ～9 となります。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCode プロパティで内容を知ることができます。

備考

デバイス番号の割り振りはUV-11を接続した順になりますが、ここでいう接続とは電源が入り、パーソナルコンピュータから認識されることを意味しています。
接続されていても電源が入っていない場合など、パーソナルコンピュータから認識されていない場合はデバイス番号は割り振られません。複数台接続されたUV-11 を識別するにはGetInfomation 関数で各UV-11 のシリアル番号を読取ることで行います。

サンプル

UV111.GpibDeviceSelect(1)' ２番目に接続された UV-11 を選択する

GpibDeviceTrigger(*address* As Integer)As long

説明

指定されたGPIB アドレスの機器にデバイストリガーメッセージを送出します。

引数

addr
対象となる機器のGPIB アドレスを指定します。有効なアドレスは0 ～30 です。
UV-11のGPIB アドレスを指定した場合はなにもせず終了します。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCode プロパティで内容を知ることができます。

サンプル

UV111.GpibDeviceTrigger(1)

GpibGetAddressMode(*mode* As Variant)As long

説明

UV-11 のGPIB インターフェースの状態を得ます。
得られる情報は
・UV-11 がアクティブなコントローラであるか否か
・UV-11 がトーカーに指定されているか否か
・UV-11 がリスナーに指定されているか否か
・GPIB バスがリモート状態になっているか否か
の4 つです。

引数

mode
状態入れる変数です。
戻り値の各Bit の意味は以下の通りです。
bit7 ～bit4(未使用)
bit3 1 でアクティブなコントローラ
bit2 1 でトーカー指定
bit1 1 でリスナー指定
bit0 1 でリモート状態

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、ErrorCode プロパティで内容を知ることができます。

GpibGotoLocal(*address* As Integer)As long

説明

指定のGPIB 機器にGotoLocal メッセージを送出します。

引数

address
対象となる機器のGPIB アドレスを指定します。有効なアドレスは0 ～30 です。UV-11の
GPIB アドレスを指定した場合はなにもせず終了します。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCode プロパティで内容を知ることができます。

サンプル

UV111.GpibGotoLocal(1)

GpibInterfaceClear As long

説明

GpibInterfaceClear 関数はインターフェースクリアメッセージをGPIB バスに送出します。

引数

ありません。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCode プロパティで内容を知ることができます。

GpibNoListener As long

説明

全ての機器のリスナー指定を解除します。

引数

ありません。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCoce プロパティで内容を知ることができます。

GpibNoTalker As long

説明

全ての機器のトーカー指定を解除します。

引数

ありません。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCode プロパティで内容を知ることができます。

GpibParallelPoll(*StatusByte* As Variant)As long

説明

パラレルポーリングを実行しその結果を返します。

引数

StatusByte
ステータスバイトを入れる変数を指定します。
引数のデータ型はInteger ですが、返される値は8Bit の符号なしデータです。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCoce プロパティで内容を知ることができます。

サンプル

```
Dim a As Variant
UV111.GpibParallelPoll(a)
```

GpibPassControl(*address* As Integer)As long

説明

GPIB バス上のアクティブコントローラを指定します。この関数は自分自身が現在アクティブコントローラでなければ機能しません。当然ながらこの関数を実行した場合、そのUV-11 自身はアクティブコントローラではなくなります。

引数

address
0 ～30 のアクティブコントローラにしたい機器のGPIB アドレスを指定します。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCode プロパティで内容を知ることができます。

備考

GPIB バス中にあるシステムコントローラはインターフェースクリアメッセージをバスに送出することにより、強制的にアクティブコントローラになれます。システムコントローラはGPIB バス中に１台しか存在してはいけません。UV-11 は電源投入時はシステムコントローラとして機能します。UV-11 をただのコントローラとして機能させるにはGpibSystemControler 関数を使用します。

サンプル

UV111.GpibPassControl(1)

GpibReceiveData(*address* As Integer,data As Variant,length As Integer)As long

説明

GpibReceiveData メソッドはGPIB 機器からデータを受信します。
このメソッドは受信時にトーカーを指定するタイプと指定しないタイプの2種類が利用できます。

引数

address
0 ～30 のトーカーアドレスを指定します。データ受信前に、指定のアドレスの機器をトーカーに指定します。
data
受信データの格納先の変数です。この引数はバリアント型として渡しますが戻ってきたときには、Byte 型の配列となっています。データの中身を参照する際は配列として扱うようにしてください。
len
最大受信データ数を指定します。データの受信はこの数以下で行われます。
指定の値に達しない場合でもGPIB の受信処理が終了した場合には終了します。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCode プロパティで内容を知ることができます。

備考

データ受信の終わりは、受信用デリミタ またはEOI により識別します。受信用デリミタ の設定については
GpibSetReceiveDelimiter 関数を参照してください。

サンプル

```
Dim data As Variant
UV111.GpibReceiveData(1,data,1024)
data(0) ‘データを参照する際は配列形式で
```

GpibReceiveString(*address* As Integer,String As Variant,length As Integer) As long

説明

GpibReceiveString メソッドはGPIB 機器から文字列データを受信します。指定された長さのデータを受信しその最後にNULL 文字を付加して返します。

引数

address
0 ～30 のトーカーアドレスを指定します。データ受信前に、指定のアドレスの機器をトーカーに指定します。
String
受信データの格納先の変数です。この引数はバリアント型として渡しますが戻ってきたときには、String 型となっています。
len
最大受信データ数を指定します。データの受信はこの数以下で行われます。
指定の値に達しない場合でもGPIB の受信処理が終了した場合には終了します。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCode プロパティで内容を知ることができます。

備考

データ受信の終わりは、受信デリミタ またはEOI により識別します。受信デリミタ の設定についてはGpibSetReceiveDelimiter 関数を参照してください。

サンプル

```
Dim String As Variant
UV111.GpibReceiveData(1,String,1024)
Text1.Text =String
```

GpibRemoteEnable As long

説明

GPIB バスにリモートイネーブルメッセージを送出します。

引数

ありません。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCode プロパティで内容を知ることができます。

サンプル

UV111.GpibRemoteEnable

GpibSelectDeviceClear(*address* As Integer)As long

説明

指定のGPIB 機器にデバイスクリアメッセージを送出します。
GpibDeviceClear メソッドは全てのGPIB 機器に有効なデバイスクリアメッセージを送出しますが、このメソッドが送出する関数は指定したGPIB 機器のみに有効となります。

引数

address
対象となる機器のGPIB アドレスを指定します。有効なアドレスは0 ～30 です。
UV-11 のGPIB アドレスを指定した場合はなにもせず終了します。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCode プロパティで内容を知ることができます。

サンプル

UV111.GpibSelectDeviceClear(1)

GpibSendString(*address* As Integer,*String* As String,*wait* As Integer)As long

説明

GpibSendString メソッドはGPIB 機器に文字列データを送信します。
出力するデータ数は文字列の長さに一致するため、出力データ数を指定する必要はありません。

引数

address
0 ～30 のリスナーアドレスを指定します。このメソッドはデータ送信前に、指定のアドレスの機器をリスナーに指定します。また指定以外のリスナー指定は解除されます。
String
送信文字列データです。
wait
UV-11 からのデータ送信が終了するまで待つか否かを設定します。
0 を指定すると待ちます。それ以外では待たずにPC からUV-11 へのデータ送信が終了した段階で戻ってきます。この場合送信が成功したか否かの確認はできません。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCode プロパティで内容を知ることができます。

備考

送信データには自動的に送信デリミター文字列が付加されます。
送信デリミター文字列の指定についてはGpibSetSendDelimiter 関数の説明を参照してください。

サンプル

```
UV111.GpibSendString 1,"コマンド", 1
```

GpibSendStringN(*String* As String,*wait* As Integer)As long

説明

GpibSendStringN メソッドはGPIB 機器に文字列データを送信します。
GpibSendStringメソッドと異なり、リスナー機器の指定を行いません。
同一のバスに繋がる同一の機器に同じコマンドを一斉に送る時などに使用すると便利です。
事前にリスナー機器を指定するにはGpibSetLisner メソッドを使用します。

引数

address
0 ～30 のリスナーアドレスを指定します。このメソッドはデータ送信前に、指定のアドレスの機器をリスナーに指定します。また指定以外のリスナー指定は解除されます。
String
送信文字列データです。
wait
無条件に1を指定してください。この項は互換性のためにあります。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCode プロパティで内容を知ることができます。

備考

送信データには自動的にデリミター文字列が付加されます。
デリミター文字列の指定についてはGpibSetSendDelimiter 関数の説明を参照してください。

サンプル

```
UV111. GpibSetListenter 1, False        ‘ 最後の引数をFalseにすると複数の機器
                                        ‘ を同時に リスナーに指定できます
UV111. GpibSetListenter  2, False       ‘2もリスナーに指定
UV111. GpibSetListenter 3, False        ‘3もリスナーに指定

UV111.GpibSendStringN "コマンド", 1      ‘1,2,3のアドレスの機器に同じコマンドを
                                        ‘送ります
```

GpibSendData(*address* As Integer,*data* As Variant,*length* As Integer,*wait* As Integer) As long

説明

GpibSendData メソッドはGPIB 機器にデータを送信します。
GpibSendStringメソッドとの違いは送出するデータが文字以外でも送信できると言う点にあります。GpibSendStringでは送信するデータを全て文字として、与えられたデータの中に文字列の終端データを見つけた時点で送信を終了しますが、GpibSendDataでは与えられたデータの内容に関わらず指定されたデータ数だけ送信します。データの内容についてはユーザーが管理します。

引数

address
0 ～30 のリスナーアドレスを指定します。このメソッドはデータ送信前に、指定のアドレスの機器をリスナーに指定します。また指定以外のリスナー指定は解除されます。
data
送信文字列データです。このデータはByte 型の配列でなければなりません。ただし値の引き渡しはバリアント型として行います具体的にはサンプルをご覧ください。
length
送信データ数です。
wait
無条件に1を指定してください。この項は互換性のために存在します。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCode プロパティで内容を知ることができます。

備考

送信データには自動的に送信デリミター文字列が付加されます。
送信デリミター文字列の指定についてはGpibSetSendDelimiter 関数の説明を参照してください。
※バイナリデータを送信する際は受信側の受信デリミタを無しに指定しないとうまく送信できない場合があります。

サンプル

```
Dim val(0 To 3)As Byte                  'BYTEサイズのデータ配列(4byte分)を作成します
Dim data As Variant
val(0)=1                                '送信データを配列に設定します
val(1)=2
val(2)=3
val(3)=4
data =val                               '最後に作成した送信データ配列をVariant変数
UV111.GpibSendData 1,data, 4, 1         '経由でGpibSendDataメソッドに渡すため、配列を
                                        '代入します
```

GpibSendDataN(*data* As Variant,*length As Integer*,*wait* As Integer)As long

説明

GpibSendDataN メソッドはGPIB 機器にデータを送信します。
このメソッドには送信前にリスナーを指定しない以外はGpibSendDataメソッドと同じ動作をします。複数のリスナーを指定して同時に同じデータを送信したい場合などに使用します。

引数

address
0 ～30 のリスナーアドレスを指定します。このメソッドはデータ送信前に、指定のアドレスの機器をリスナーに指定します。また指定以外のリスナー指定は解除されます。
data
送信文字列データです。このデータはByte 型の配列でなければなりません。ただし値の引き渡しはバリアント型として行います具体的にはサンプルをご覧ください。
length
送信データ数です。
wait
無条件に1を指定してください。この項は互換性のために存在します。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCode プロパティで内容を知ることができます。

備考

送信データには自動的に送信デリミター文字列が付加されます。
送信デリミター文字列の指定についてはGpibSetSendDelimiter 関数の説明を参照してください。

GpibSerialPoll(*address* As Integer,*StatusByte* As Variant)As long

説明

指定したGPIB 機器に対してシリアルポーリングを実行しその結果を返します。

引数

address
対象となる機器のGPIB アドレスを指定します。有効なアドレスは0 ～30 です。
StatusByte
ステータスバイトを入れる変数を指定します。
引数のデータ型はInteger ですが、返される値は8Bit の符号なしデータです。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCode プロパティで内容を知ることができます。

サンプル

```
Dim a As Variant
UV111.GpibSerialPoll 1,a
```

GpibSetListenter(*address* As Integer,ul As Boolean)As long

説明

指定されたGPIB アドレスの機器をリスナーに指定します。

引数

address

リスナーに指定する機器のGPIB アドレスを指定します。有効なアドレスは0 ～30 です。

ul

True を指定した場合*addr* で指定した以外の機器のリスナー指定を解除します。

False を指定した場合解除しません。複数の機器に同じメッセージを送信する際などは便利です。

戻り値

SUCCESS 実行に成功しました。

FAIL 実行に失敗しました。

失敗した場合その状態は、ErroeCode プロパティで内容を知ることができます。

サンプル

UV111.GpibSetListenter 1

GpibSetReceiveDelimiter(*delimiter* As Integer)As long

説明

受信用のデリミター(区切り文字)を設定します。

引数

delimiter

区切り文字を指定します。

区切り文字には(下記の値は何れも16進数であり、表記方法はVisualBasic形式です)

&H0d(CR)

&H0a(LF)

0(デリミタなしEOIのみ、バイナリデータの受信などに使用します)

を指定します。

戻り値

SUCCESS 実行に成功しました。

FAIL 実行に失敗しました。

失敗した場合その状態は、ErrorCode プロパティで内容を知ることができます。

備考

UV-11 は受信終了を検出する手段として、区切り文字だけでなくEOI の検出も行っています。これを利用すればバイナリデータの受信もできます。EOI の検出を無効にすることはできません。区切り文字を受け取ってからも一定時間EOI の検出を試みます。このためEOIを送出しない機器からデータを受信する際は余分に時間が掛かります。EOI がそのまま検出されない場合もその時点で受信終了とみなします。エラーにはなりません。

サンプル

```
UV111.GpibSetReceiveDelimiter &H0a          '16進数で指定した場合です
```

GpibSetSendDelimiter(*delimiter* As String)As long

説明

送信用のデリミター(区切り文字列)を設定します。

引数

delimiter

区切り文字列を指定します。区切り文字列は1 または2 文字から構成されます。
また引数を省略することでデリミターなしの設定もできます。
使用できる文字は任意ですが、通常&H0d(CR)と0x0a(LF)が使用されます。
2文字以上データを設定しても3文字目以降は無視されます。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCode プロパティで内容を知ることができます。

備考

デリミター文字列以外にUV-11 は必ず送信の終わりにEOI メッセージを送信します。
デリミターなしの場合でもリスナーがEOI に対応していれば送信の終わりを検出することができます。

サンプル

```
Dim a As String
a =Chr(&H0d)&Chr(&H0a)              '16進数で文字列を作成します
UV111.GpibDeviceClear a             'デリミターは0x0d,0x0a に設定されます
```

GpibSetSRQStatus(*StatusByte* As Integer,*Pending* As Integer,*time* As Long) As long

説明

ステータスバイトの設定を行うと同時にサービス要求を発生します。

引数

StatusByte
ステータスバイトです。8Bit の任意のデータですがBit6 は使用できません。

Pending
この関数の戻りのタイミングを指定します。
0
設定と同時にこの関数は戻ります。結果については考慮しません。
1
サービスリクエストが受け付けられ、ステータスバイトの送出が行われた時点で戻ります。上記以外の値は無効です。

time
Pending 引数に1 を指定した場合のタイムアウト時間を指定します。
指定する値はmsec 単位で10msec から60000msec まで10msec刻みで指定します。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCode プロパティで内容を知ることができます。

備考

ステータスバイトの内容についてはこのライブラリでは特に定義しません。ユーザーがその意味付けを行う必要があります。

サンプル

```
UV111.GpibSetSRQStatus &H10,1,2000          'bit4を1にして2秒間待ちます
```

GpibSetTalker(*address As Integer*) As long

説明

指定されたGPIB アドレスの機器をトーカーに指定します。

引数

addr

トーカーに指定する機器のGPIB アドレスを指定します。有効なアドレスは0 ～30 です。

※当然ですがGPIBバスライン上には1つしかトーカーは存在できません。

戻り値

SUCCESS 実行に成功しました。

FAIL 実行に失敗しました。

失敗した場合その状態は、ErrorCode プロパティで内容を知ることができます。

GpibSystemControler(*sel* As Integer) As long

説明

GpibSystemControler 関数はUV-11 をシステムコントローラとするかそうでない普通のコントローラとするかを設定します。

引数

sel

1 でシステムコントローラ、1 以外でそうでない普通のコントローラとなります。

戻り値

SUCCESS 実行に成功しました。
FAIL 実行に失敗しました。
失敗した場合その状態は、ErrorCodeプロパティで内容を知ることができます。

備考

同一GPIB バス中に複数のコントローラが存在することは仕様上許されていますが、
その中でも特別の権限を持ったコントローラをシステムコントローラといいます。
システムコントローラは他のコントローラの状態によらずバスのコントロール権を
得ることができるなど特別の存在であるため同一のバス中には１台しか存在が許されません。
UV-11 を複数同一のGPIB バスに接続する場合はこの関数を使用してシステムコントローラ
とそうでないコントローラに個別に設定する必要があります。
ヒントシステムコントローラはGPIB バスにインターフェースクリアメッセージを送出する
ことのより強制的にアクティブコントローラになります。

サンプル

UV111.GpibSystemControler 1

UV11.DLL 関数リファレンス

UV-11 制御用ライブラリUV11.DLL に含まれる関数のリファレンスです。
データ型については VisualC++の仕様を基に記述してあります。

int GetErrorCode(void)

説明

ライブラリ関数の実行エラーコードを返します。
返されるエラーコードはこの関数を実行する前にエラーとなった関数のものです。
関数を実行してもエラーとならなかった場合は、その値を保持します。

引数

ありません。

戻り値

```
#define UNEXPECTED_ERROR 1              //予期しないエラー
#define ARGMENT_INVALID 2               //引数が不正
#define OUT_OF_MEMORY 3                 //メモリ不足
#define NO_DEVICE 4                     //指定のデバイスは存在しません
#define CANT_USE_DEVICE 5               //指定されているデバイスは使用できません
#define SRQ_CONTROLER_INACTIVE 6        //コントローラとしてアクティブな状態です
#define SRQ_NOT_REMOTE 7                //リモート状態にありません
#define SRQ_SET_PENDING 8               //SRQ の送出待ちです
#define SRQ_SET_TIMEOUT 9               //SRQ の送出タイムアウト
#define GPIB_TRANS_TIMEOUT 10           //GPIB バスデータ送信タイムアウト
#define GPIB_RCV_TIMEOUT 11             //GPIB バスデータ受信タイムアウト
#define USB_TRANS_TOOBIG -1             // USB通信、送信バッファーサイズを超えるデータ
#define USB_TRANS_NODEVICE -2           // USB通信、送信デバイスがない
#define USB_TRANS_TIMEOUT -3            // USB通信、送信タイムアウト
#define USB_RCV_TOOBIG -4               // USB通信、受信バッファーサイズを超えるデータ
#define USB_RCV_NODEVICE -5             // USB通信、受信デバイスがない
#define USB_RCV_TIMEOUT -6              //USB通信、受信タイムアウト
```

以上の定義は UV11FUNC.H にあります。

int GetInfomation(BYTE *InfoNo*,LPSTR *ret*)

説明

シリアル番号などのUV-11 の情報を得ます。
情報は全て8 文字固定の文字列で返されます。

引数

InfoNo
情報の種類を指定します。
0
機器の名前を返します。UV-11 の場合"UV-11"という文字列(固定)が返されます。
1
USB ベンダーID とUSB プロダクトID を返します。UV-11 の場合"06711002"という文字列(固定)が返されます。0671 が弊社に割り当てられているUSB ベンダーID です。1002 は弊社でUV-11 に割り当てたプロダクトID です。いずれも16 進数で表記されています。
2
プロダクトリビジョンとファームウエアのバージョン番号を返します。プロダクトリビジョンとはUV-11 のハードウエア部の世代を示す番号です。
3
シリアル番号を返します。例えば"23110001"のような8 桁の番号となります。
上記以外の値は無効(エラー)となります。
ret
戻される文字列の格納先の先頭ポインタを指定します。戻される文字列は常に8 文字固定です。

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

int GpibDeviceClear(void)

説明

全てのGPIB 機器に有効なにデバイスクリアメッセージを送出します。
特定のGPIB 機器にデバイスクリアメッセージを送るときはGpibSelectDeviceClear
関数を使用します。

引数

ありません。

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

int GpibDeviceSelect(BYTE *d*n)

説明

UV-11 の制御関数の対象となるUV-11 を選択します。UV-11 の制御関数は現在選択されているUV-11 にのみ作用します。UV-11 は同時に10 台までの接続をサポートしています。複数のUV-11 を個別に制御するためには、制御関数の対象となるUV-11 を選択する必要があります。選択された状態は次に選択を変更するまで有効です。１度選択すれば同じUV-11 を制御する限り、再びこの関数を実行する必要はありません。
またデフォルトで最初に接続されたUV-11 が選択されていますので、１台しか接続しない場合はこの関数を実行する必要はありません。

引数

dn
デバイス番号を指定します。
UV-11 に割り振られるデバイス番号は接続した順に0 ～9 となります。

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

備考

デバイス番号の割り振りはUV-11 を接続した順になりますが、ここでいう接続とは電源が入り、パーソナルコンピュータから認識されることを意味しています。
複数台接続されたUV-11 を識別するにはGetInfomation 関数で各UV-11 のシリアル番号を読取ることで行います。

int GpibDeviceTrigger(BYTE *address*)

説明

指定されたGPIB アドレスの機器にデバイストリガーメッセージを送出します。

引数

address
対象となる機器のGPIB アドレスを指定します。有効なアドレスは0 ～30 です。
UV-11 のGPIB アドレスを指定した場合はなにもせず終了します。

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

int GpibGetAddressMode(BYTE *_mode_)

説明

UV-11 のGPIB インターフェースの状態を得ます。
得られる情報は
・UV-11 がアクティブなコントローラであるか否か
・UV-11 がトーカーに指定されているか否か
・UV-11 がリスナーに指定されているか否か
・GPIB バスがリモート状態になっているか否か
の４つです。

引数

mode
状態代入する変数へのポインタです。
戻り値の各Bit の意味は以下の通りです。
bit7 ～bit4(未使用)
bit3 1 でアクティブなコントローラ
bit2 1 でトーカー指定
bit1 1 でリスナー指定
bit0 1 でリモート状態

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

int GpibGotoLocal(BYTE *addr*)

説明

指定のGPIB 機器にGotoLocal メッセージを送出します。

引数

addr

対象となる機器のGPIB アドレスを指定します。有効なアドレスは0 ～30 です。

UV-11 のGPIB アドレスを指定した場合はなにもせず終了します。

戻り値

SUCCESS 関数の実行に成功しました。

FAIL 関数の実行に失敗しました。

失敗した場合その状態は、GetErrorCode 関数で知ることができます。

int GpibInterfaceClear(void)

説明

GpibInterfaceClear 関数はインターフェースクリアメッセージをGPIB バスに送出します。

引数

ありません。

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

int GpibLocalLockOut(void)

説明

GPIB バスにローカルロックアウトメッセージを送出します。

引数

ありません。

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

int GpibParallelPoll(BYTE *stb)

説明

パラレルポーリングを実行しその結果を返します。

引数

stb ステータスバイトの格納先アドレスを指定します。

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

int GpibPassControl(BYTE *addr*)

説明

GpibPassControl 関数はGPIB バス上のアクティブコントローラを指定します。
この関数は自分自身が現在アクティブコントローラでなければ機能しません。
当然ながらこの関数を実行した場合、そのUV-11 自身はアクティブコントローラでは
なくなります。

引数

addr
0 ～30 のアクティブコントローラにしたい機器のGPIB アドレスを指定します。

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

備考

GPIB バス中にあるシステムコントローラはインターフェースクリアメッセージをバス
に送出することにより、強制的にアクティブコントローラになれます。システムコン
トローラはGPIB バス中に１台しか存在してはいけません。UV-11 は電源投入時はシステム
コントローラとして機能します。UV-11 をただのコントローラとして機能させる
にはGpibSystemControler 関数を使用します。

int GpibReceiveData(BYTE *data,DWORD len)

int GpibReceiveData(BYTE addr,BYTE *data,DWORD len)

説明

GpibReceiveData 関数はGPIB 機器からデータを受信する関数です。
この関数は多重定義されており、先頭の引数に*addr* がある場合とない場合があります。

引数

addr
0 ～30 のトーカーアドレスを指定します。この引数を設定するとデータ受信前に、指定のアドレスの機器をトーカーに指定します。またこの引数は省略することもできます。この場合トーカーの指定は行いません。
**data*
受信データの格納先の先頭アドレスを指定します。
受信データは BYTE 型のデータです。
len
最大受信データ数を指定します。データの受信はこの数以下で行われます。
指定の値に達しない場合でもGPIB の受信処理が終了した場合には終了します。

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

備考

データ受信の終わりは、受信デリミタ またはEOI により識別します。受信デリミタ の設定についてはGpibSetReceiveDelimiter 関数を参照してください。

int GpibReceiveString(LPSTR *buf*,DWORD *len*)

int GpibReceiveString(BYTE *addr*,LPSTR *buf*,DWORD *len*)

説明

GpibReceiveString 関数はGPIB 機器から文字列データを受信する関数です。指定された長さのデータを受信しその最後にNULL 文字を付加して返します。この関数は多重定義されており、先頭の引数に*addr* がある場合とない場合があります。

引数

addr
0 ～30 のトーカーアドレスを指定します。この引数を設定するとデータ受信前に、指定のアドレスの機器をトーカーに指定します。またこの引数は省略することもできます。この場合トーカーの指定は行いません。
buf
受信文字列データの格納先の先頭アドレスを指定します。バッファーのサイズは指定した受信データ長＋１バイト以上でなければなりません。
len
最大受信データ数を指定します。データの受信はこの数以下で行われます。
指定の値に達しない場合でもGPIB の受信処理が終了した場合には終了します。

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

備考

データ受信の終わりは、受信デリミタ またはEOI により識別します。受信デリミタの設定についてはGpibSetReceiveDelimiter 関数を参照してください。

int GpibRemoteEnable(void)

説明

GPIB バスにリモートイネーブルメッセージを送出します。

引数

ありません。

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

int GpibSelectDeviceClear(BYTE *addr*)

説明

指定のGPIB 機器にデバイスクリアメッセージを送出します。
GpibDeviceClear 関数は全てのGPIB 機器に有効なデバイスクリアメッセージを送出しますが、この関数が送出する関数は指定したGPIB 機器のみに有効となります。

引数

addr
対象となる機器のGPIB アドレスを指定します。有効なアドレスは0 ～30 です。
UV-11 のGPIB アドレスを指定した場合はなにもせず終了します。

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

int GpibSendData(BYTE *data,DWORD len,BYTE wait =1)

int GpibSendData(BYTE addr,BYTE *data,DWORD len,BYTE wait =1)

説明

GpibSendData 関数はGPIB 機器にデータを送信する関数です。
この関数は多重定義されており、先頭の引数に*addr* がある場合とない場合があります。

引数

addr
0 ～30 のリスナーアドレスを指定します。
この引数を設定するとデータ送信前に、指定のアドレスの機器をリスナーに指定します。またこの引数は省略することもできます。この場合リスナーの指定は行いません。
**data*
送信データの先頭アドレスを指定します。
送信データは BYTE 型のデータです。この引数は送信データの格納されているエリアの先頭アドレスを指定します。
len
送信データ数を指定します
wait
無条件に1を指定してください。この引数は互換性のために有ります。

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

備考

送信データには自動的に送信デリミター文字列が付加されます。
この文字列の指定についてはGpibSetSendDelimiter 関数の説明を参照してください。

int GpibSendString(LPCSTR *str*,BYTE *wait =1*)

int GpibSendString(BYT*Eaddr*,LPCSTR *str*,BYTE *wait =1*)

説明

GpibSendString 関数はGPIB 機器に文字列データを送信する関数です。
出力するデータ数は文字列の長さに一致するため、出力データ数を指定する必要はありません。この関数は多重定義されており、先頭の引数に*addr* がある場合とない場合があります。

引数

addr
0 ～30 のリスナーアドレスを指定します。
この引数を設定するとデータ送信前に、指定のアドレスの機器をリスナーに指定します。またこの引数は省略することもできます。この場合リスナーの指定は行いません。
str
送信文字列データの先頭アドレスを指定します。送信文字列データは char 型のデータです。
この引数は送信データの格納されているエリアの先頭アドレスを指定します。
wait
無条件で1を指定してください。この引数は互換性のためにあります。

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

備考

送信データには自動的にデリミター文字列が付加されます。
この文字列の指定についてはGpibSetSendDelimiter 関数の説明を参照してください。

int GpibSerialPoll(BYTE *addr*,BYTE **st*b)

説明

指定したGPIB 機器に対してシリアルポーリングを実行しその結果を返します。

引数

addr

対象となる機器のGPIB アドレスを指定します。有効なアドレスは0 ～30 です。

**stb*

ステータスバイトの格納先アドレスを指定します。

戻り値

SUCCESS 関数の実行に成功しました。

FAIL 関数の実行に失敗しました。

失敗した場合その状態は、GetErrorCode 関数で知ることができます。

int GpibSetListenter(BYTE *addr*,BOOL *ul*)

説明
指定されたGPIB アドレスの機器をリスナーに指定します。

引数
addr
リスナーに指定する機器のGPIB アドレスを指定します。有効なアドレスは0 ～30 です。
ul
1 を指定した場合*addr* で指定した以外の機器のリスナー指定を解除します。
0 を指定した場合解除しません。複数の機器に同じメッセージを送信する際などは便利です。

戻り値
SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

int GpibSetReceiveDelimiter(BYTE *delimiter* =0)

説明

受信用のデリミター(区切り文字)を設定します。

引数

delimiter
区切り文字を指定します。
区切り文字には0x0d(CR)か0x0a(LF)を指定します。これ以外の値の場合は区切り文字なしとなります。この引数はデフォルト値として0 を取ります。引数を省略した場合区切り文字はなしとなります。

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

備考

UV-11 は受信終了を検出する手段として、区切り文字だけでなくEOI の検出も行っています。これを利用すればバイナリデータの受信もできます。EOI の検出はデフォルトで無効にすることはできません。区切り文字を受け取ってからも一定時間EOI の検出を試みます。EOI がそのまま検出されない場合もその時点で受信終了とみなします。

int GpibSetSendDelimiter(LPSTR *delimiter* ="")

説明

送信用のデリミター(区切り文字列)を設定します。

引数

delimiter
区切り文字列の先頭アドレスを指定します。区切り文字列は1 または2 文字から構成されます。また引数を省略することでデリミターなしの設定もできます。
使用できる文字は任意ですが、通常0x0d(CR)と0x0a(LF)が使用されます。

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

備考

デリミター文字列以外にUV-11 は必ず送信の終わりにEOI メッセージを送信します。
デリミターなしの場合でもリスナーがEOI に対応していれば送信の終わりを検出することができます。

int GpibSetSRQStatus(BYTE *st*b,BYTE *pen*d,DWORD *time =200*0)

説明

GpibSetSRQStatus 関数はステータスバイトの設定を行うと同時にサービス要求を発生します。

引数

stb
ステータスバイトです。8Bit の任意のデータですがBit6 は使用できません。

pend
この関数の戻りのタイミングを指定します。
0
設定と同時にこの関数は戻ります。結果については考慮しません。
1
サービスリクエストが受け付けられ、ステータスバイトの送出が行われた時点で戻ります。
上記以外の値は無効です。

time
pend 引数に1 を指定した場合のタイムアウト時間を指定します。
指定する値は10msec 単位で10msec から60000msec まで10msec 刻みで指定できます。
この引数には2000msec のデフォルト値が設定されています。

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

備考

ステータスバイトの内容についてはこのライブラリでは特に定義しません。ユーザーがその意味付けを行う必要があります。

int GpibSetTalker(BYTE *addr*)

説明

指定されたGPIB アドレスの機器をトーカーに指定します。

引数

addr

トーカーに指定する機器のGPIB アドレスを指定します。有効なアドレスは0 ～30 です。

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

int GpibSystemControler(BYTE *sel*)

説明

GpibSystemControler 関数はUV-11 をシステムコントローラとするかそうでない普通のコントローラとするかを設定します。

引数

sel
1 でシステムコントローラ、1 以外でそうでない普通のコントローラとなります。

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

備考

同一GPIB バス中に複数のコントローラが存在することは仕様上許されていますが、その中でも特別の権限を持ったコントローラをシステムコントローラといいます。システムコントローラは他のコントローラの状態によらずバスのコントロール権を得ることができるなど特別の存在であるため同一のバス中には1台しか存在が許されません。
UV-11 を複数同一のGPIB バスに接続する場合はこの関数を使用してシステムコントローラとそうでないコントローラに個別に設定する必要があります。
ヒントシステムコントローラはGPIB バスにインターフェースクリアメッセージを送出することのより強制的にアクティブコントローラになります。

int GpibTimeOut(DWORD *time*)

説明

GPIB バスタイムアウト時間を設定します。

引数

time
タイムアウトまでの時間をmsec 単位で指定します。
10msec ～60000msec の間で10msec 刻みで指定できます。

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

備考

UV-11 の電源投入時の値は、5000msec です。

int GpibUnListen(void)

説明

全ての機器のリスナー指定を解除します。

引数

ありません。

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

int GpibUnTalk(void)

説明

全ての機器の トーカー指定を解除します。

引数

ありません。

戻り値

SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

int SetGpibMyAddress(BYTE *addr*)

説明

UV-11 自身のGPIB アドレスを設定します。

引数

addr

UV-11 自身のGPIB アドレスを指定します。有効なアドレスは0 ～31 です。

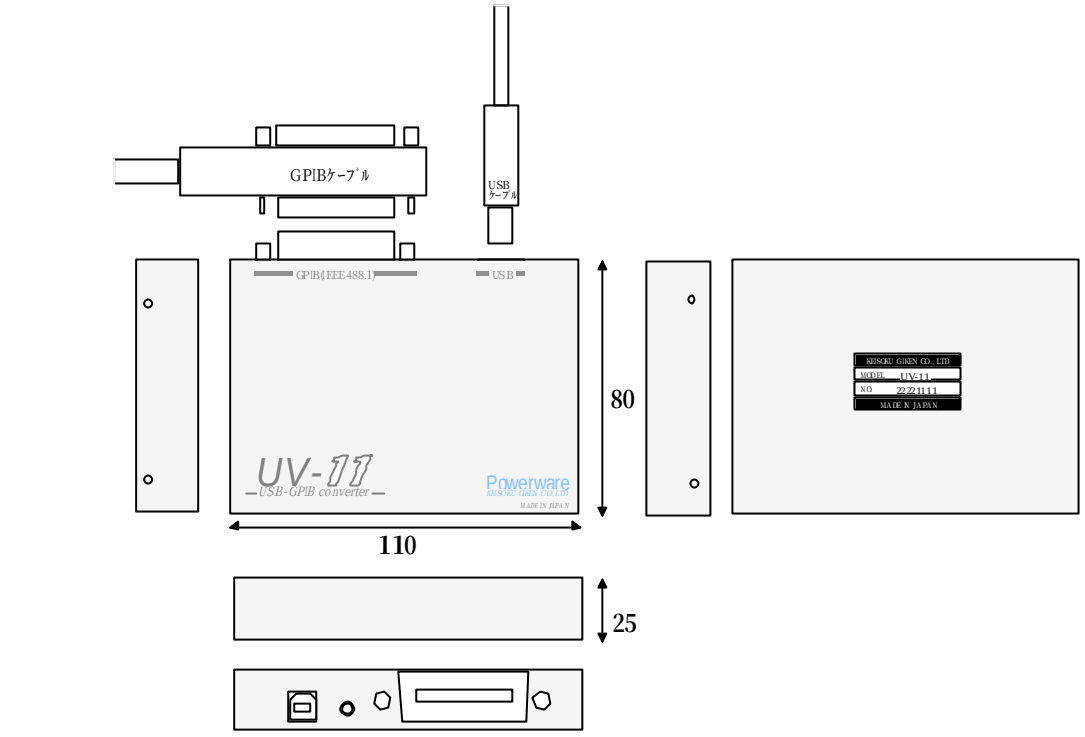
戻り値

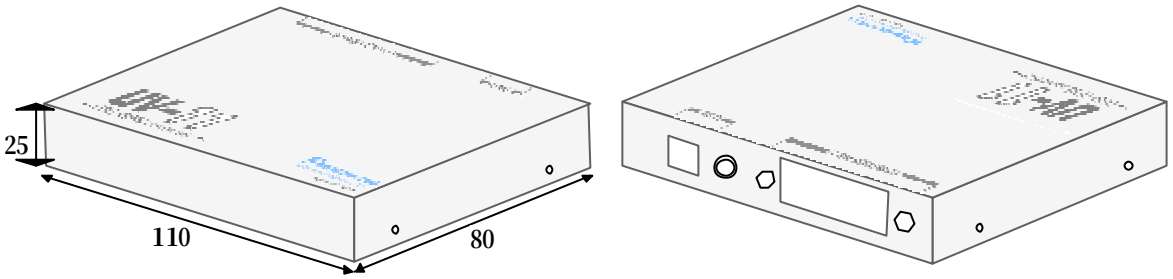
SUCCESS 関数の実行に成功しました。
FAIL 関数の実行に失敗しました。
失敗した場合その状態は、GetErrorCode 関数で知ることができます。

UV10 ハードウエアリファレンス

電気物理仕様

USB （Univasal Serial Bus Rev1.1 準拠）１ポート
GPIB １ポート
電源電圧、消費電流 DC5V ～4.5V 、300mA 以下
寸法 110(W)×25(H)×80(D)
重量 0.28Kg


GPIBケーブル
USB
ケーブル
GPIB(IEEE-488.1)
USB
UV-11
USB-GPIB converter
Powerware
MADE IN JAPAN
80
110
KEISOKU GIKEN CO., LTD
MODEL UV-11
NO 22211111
MADE IN JAPAN
25


25
110
80