Getting Started with Your GPIB-98Turbo and the NI-488.2TM Software for MS-DOS

(和文マニュアル)

December 1993 Edition

Part Number 370912A-01

日本ナショナルインスツルメンツ株式会社
〒142 東京都品川区戸越5-14-23　池田ビル3F
TEL 03(3788)1921　FAX 03(3788)1923

National Instruments Corporate Headquarters
6504 Bridge Point Parkway
Austin, TX 78730-5039
(512) 794-0100
Technical support fax:　(800) 328-2203
(512) 794-5678

Branch Offices:
Australia (03) 879 9422, Austria (0662) 435986, Belgium 02/757.00.20,
Canada (Ontario) (519) 622-9310, Canada (Québec) (514) 694-8521,
Denmark 45 76 26 00, Finland (90) 527 2321, France (1) 48 14 24 24,
Germany 089/741 31 30, Italy 02/48301892, Japan (03) 3788-1921,
Netherlands 03480-33466, Norway 32-848400, Spain (91) 640 0085,
Sweden 08-730 49 70, Switzerland 056/20 51 51, U.K. 0635 523545

製品保証

ナショナルインスツルメンツの全ての製品は、製品の出荷日から2年間の保証を致します。本保証には、ボード、部品、ケーブル、コネクタ、スイッチ、等の不良が含まれます。ただし、誤用によって生じた故障、または不良の場合は、保証期間内であってもその対象ではありません。保証期間を過ぎた場合は有償となります。

版権

商標

はじめに

このマニュアルは、ナショナルインスツルメンツ社製GPIB-98Turboインターフェースボード及びNI-488.2 for MS-DOSハンドラーをインストールして構成するための手順が説明されています。このパッケージは、NEC PC9801シリーズ及びその互換機で使用することを前提とします。このマニュアルは、*NI-488.2 Software Reference Manual for MS-DOS* (P/N320282-01)と一緒にご使用下さい。

マニュアルの編成

このマニュアルは次のように編成されています。

- 「第1章 紹介」では、GPIB-98Turboインターフェースボードについて簡単に説明しています。そして、GPIB-98Turboのキットの内容とオプション製品のリストを示します。
- 「第2章 インストレーションと基本プログラミング例」では、ハードウェアとソフトウェアのインストレーションの手順について説明し、そして基本的なプログラミング例を紹介します。
- 「第3章 NI-488.2ルーチンを使った拡張プログラムの作成」では、NI-488.2ルーチンの紹介とNI-488.2プログラムの作成を順を追って説明しています。
- 「付録A ハードウェアとソフトウェアのコンフィギュレーション設定の変更」では、GPIB-98Turboインターフェースボードのコンフィギュレーション設定の変更を順を追って説明しています。
- 「付録B インタラプトラインとDMAチャンネルのパラメータ」では、システムないのGPIB-98Turboと他のアダプタでぶつかりがあるかどうかを判断するための説明を行っています。

- 「付録C 仕様」では、GPIB-98Turboの仕様がリスト表示されています。
- 「付録D カスタマコミュニケーション」では、製品に関してナショナルインスツルメンツへのご意見をお寄せいただくためのフォーム作成について説明します。

目次

図

表

第1章
紹介

この章では、GPIB-98Turboインターフェースボードを紹介します。GPIB-98Turboキットの内容とオプション製品をリスト表示します。さらに、GPIB-98Turboを開梱するときの手順を説明します。

GPIB-98Turboは、拡張スロットを備えたNEC PC9801シリーズコンピュータ及び互換機で使用し、全機能を持つ高性能なIEEE-488インターフェースボードです。GPIB-98TurboによってPC9801コンピュータは高性能なIEEE-488コントローラになります。

図1-1は、GPIB-98Turboインターフェースボードです。

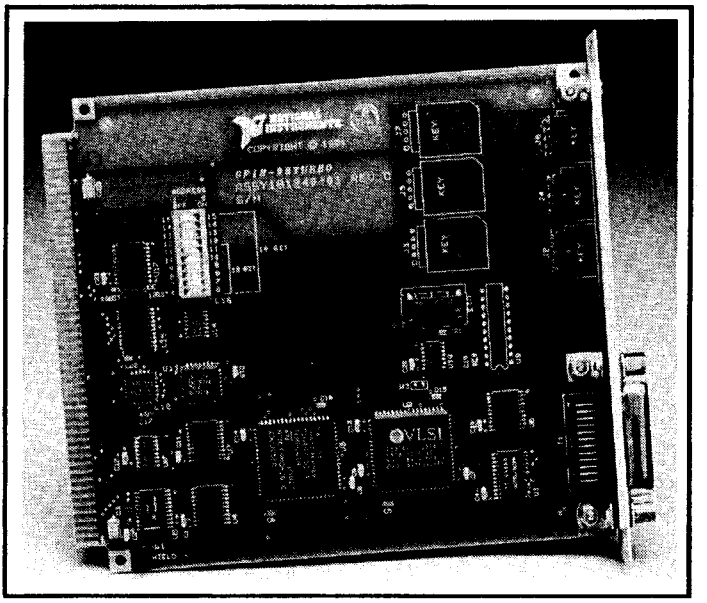


図1-1 GPIB-98Turboインターフェースボード

キットの内容

キットには次の製品が含まれています。

製品	パーツ番号
GPIB-98Turboインターフェースボード	181240-01
3.5インチ NI-488.2 Distribution Disk for GPIB-98Turbo Handler, N88BASIC, QuickBASIC, C & Universal Interfaces または 5.25インチ NI-488.2 Distribution Disk for GPIB-98Turbo Handler, N88BASIC, QuickBASIC, C & Universal Interfaces	422755-71 420778-71 & 420779-71
NI-488.2 Software Reference Manual for MS-DOS	320282-01
Universal Language Interface Using HP-Style Calls	320135-90
Getting Started with Your GPIB-98Turbo and the NI-488.2™Software for MS-DOS	320341-01

キットにこれらのものが含まれていることを確認して下さい。何か不足がありましたらナショナルインスツルメンツまで御連絡して下さい。

オプション製品

製品	パーツ番号
シングルシールドGPIBケーブル	
Type X1 ケーブル － 1 m	763001-01
Type X1 ケーブル － 2 m	763001-02
Type X1 ケーブル － 4 m	763001-03
ダブルシールドGPIBケーブル	
Type X2 ケーブル － 1 m	763061-01
Type X2 ケーブル － 2 m	763061-02
Type X2 ケーブル － 4 m	763061-03
GPIB Connector Extender	760402-01

GPIB-98Turboの開梱

GPIB-98Turboを開梱するときは次の手順で行って下さい。

1．パッケージに含まれている製品が、この章の最初にあるキットのパーツリストと一致しているかどうかを確認して下さい。この時点ではまだプラスチックバッグからボード取り出さないで下さい。

2．GPIB-98Turboボードは、静電気によるダメージから保護するために静電気防止プラスチックバッグに入れて出荷しています。ボード上のいくつかの部品は静電気が放電するとダメージを受けることがあります。ボードを取り扱う上でこのようなダメージを避けるには、バッグからボードを抜き出す前に、コンピュータのシャーシの金属部分にプラスチックバッグを触れさせておきます。

3．バッグからボードを取り出し、ボード上のコンポーネントに緩

みがないか、また、ダメージを受けていないかを検査して下さい。何等かの原因でボードにダメージが発見された場合は、ナショナルインスツルメンツに御連絡下さい。

第２章
インストレーションと基本プログラミング例

この章では、GPIB-98TurboインターフェースボードとNI-488.2ソフトウェアのインストレーション手順について説明します。次に述べる手順どおりに進まなかったり、不明な点がある場合は、付録A「ハードウェアコンフィギュレーション設定の変更」を参照して下さい。

さらに、この章ではNI-488.2ルーチンとNI-488.2ハンドラーのインストレーションを行います。

インストレーション

次の手順に従ってGPIB-98TurboとNI-488.2ハンドラーのインストールを行います。

ステップ１ － ハードウェアのインストレーション

次の手順に従ってGPIB-98Turboインターフェースボードをインストールします。

1．コンピュータの電源を切ります。

2．電源コードを抜いておきます。

3．コンピュータ背面の空きスロットを選び、拡張スロットのカバーを外します。

4．ガイドレールに沿ってGPIB-98Turboの端を入れ、奥まで差し込みます。

5．GPIB-98Turboのブラケットを背面パネルにネジ止めします。

6．電源コードを差し込みます。

7．コンピュータの電源を入れます。

コンピュータは正常に立ち上がるはずですが、もし正常に立ち上がらなければGPIB-98TurboのベースI/Oアドレス変更する必要があるかもしれません。

GPIB-98Turboのデフォルトベースアドレスは、07D0（16進数）となっています。この設定が他のハードウェアとぶつかっているのがわかっていれば、付録A「ハードウェアとソフトウェアのコンフィギュレーション設定の変更」を参照して下さい。そうでなければ、次のステップに進みます。

ステップ2 － ソフトウェアのインストレーション

次の手順に従ってNI-488.2ソフトウェアをインストールします。

1．NI-488.2ディストリビューションディスケットをフロッピードライブ（例えばBドライブ）に入れ、次のコマンドを入力します。

```
b:install/q                    <リターン>
```

ここで、*b:*はフロッピードライブ名です。

INSTALLプログラムは、NI-488.2ソフトウェアを直ちにインストールし、GPIB-98Turboのテストプログラムを走らせます。INSTALLはドライブAのGPIB-98Tディレクトリにソフトウェアをインストールします。ここで、立ち上げドライブはAとします。

これらの前提がシステムで確定していなければ、/qオプションを使わずにINSTALLプログラムを走らせます。INSTALLプログラムのさらに詳細な説明については、*NI-488.2 Software Reference Manual for MS-DOS* の第2章を参照して下さい。

2．モニタに表示される指示に従って行います。

3．コンピュータを立ち上げ直します。

ソフトウェアのインストレーションが完了したら、次のセクションである「プログラミング例」に進みます。

ソフトウェアインストレーションのいずれかでもうまくいかない場合、ハードウェア設定かあるいはDMAチャンネルまたはインタラプトラインを変更する必要があります。

デフォルト設定は次のとおりです。

- インタラプトライン　3
- DMAチャンネル　3*

これらの設定が他のハードウェアでぶつかっているのがわかっている場合、付録A「ハードウェアとソフトウェアのコンフィギュレーション設定の変更」と付録B「インタラプトラインとDMAチャンネルのパラメータ」を参照して下さい。インストレーションが完了したら、ステップ3の「IBCONFの実行」に進みます。

ステップ3 － IBCONFによるコンピュータの機種とスロット番号の選択

ハードウェアとソフトウェアのインストールが完了したら、NI-488.2ソフトウェアに次の情報を設定しなければなりません。

- コンピュータの機種
- GPIB-98Turboがインストールされているスロット番号

---

*ほとんどのコンピュータはDMAチヤンネル3が使えます。詳しい説明は付録Bで行います。

ユーティリティプログラムIBCONF.EXEを使ってこれらの設定を行います。IBCONF.EXEは、インターフェースボードとボードに接続されているGPIBデバイスの構成要素を変更するために使用し、画面操作による対話式のプログラムです。IBCONFの詳細については、*NI-488.2 Software Reference Manual for MS-DOS* に述べられています。しかし、Getting Startedの目的は、コンピュータの機種とスロット番号をどのように設定するかを知るためにあります。

GPIB-98Turboがインストールされているコンピュータの機種とスロット番号を選択するには、次の手順で行います。

1．インストールされたGPIBディストリビューションファイルが含まれているディレクトリに変更し、次のコマンドを入力します。

```
ibconf                          <リターン>
```

2．最初の画面が現れた後、何かキーを押します。ハンドラーに対してコンピュータの機種を選択するのが初めての場合、NEC Machine Typesという画面が表示されます。ここでは、ステップcの手順を飛ばしてステップdに進むこともできます。

3．画面の下部にファンクションキーが表示されます。これによって、NEC Machine Typesという画面が現れます。

4．NECパーソナルコンピュータの機種リストが画面に表示されます。これらはアルファベット順に表示されます。矢印キーを使って使用している機種にカーソルを合わせます。

注意）表示リストの中に、使用コンピュータの機種名がない場合は、矢印キーを使ってOtherに合わせます。

5．F9キーを押して前画面に戻ります。

6．F8キーを押して Board Characteristics画面を選択します。

7．上下の矢印キーを使ってSelect Slotのパラメータに合うようにリストをスクロールさせます。

8．左右の矢印キーを使って、GPIB-98Turboインターフェースボードが入っているスロットのスロット番号を設定します。

9．画面左下部にSelect Changes? という質問が表示されますのでyesを意味するyを入力してリターンキーを押します。

10．そして、ロードされているハンドラーとディスク上のハンドラーファイルを変更するかどうかを問われますので、yesのyを入力してリターンキーを押します。

これで変更内容がセーブされ、ソフトウェアとハードウェアが完全にインストールされました。

プログラミング例

次のプログラムは、新しいNI-488.2ルーチンを使用した簡単なQuickBASICのプログラムです。このプログラムは、コメントを含んでいますが、僅か8行のコードですので、プログラムすべてをQuickBASIC環境へ容易に入力して実行することができます。または、ディストリビューションディスケットに収められているサンプルプログラムのひとつを調べて実行することもできます。

このプログラムは、GPIBアドレスが6で、GPIB-98Turboのボード番号0(gpib0)を通してコンピュータに接続されているフルーク45マルチメータからデータを取り込みます。各ルーチンについての情報は、*NI-488.2 Software Reference Manual for MS-DOS* を参照して下さい。

```
REM $INCLUDE 'A:\GPIB-98T\QBDECL.BAS'
DIM Reading as string * 30
'
' Assert Interface Clear (IFC) for more than 100usec on GPIB
' interface board number zero (0).
'
CALL SendIFC(0)                         'initialize bus.
'
' Selectively clear the multimeter: address it to listen
' (GPIB address 6 on board number 0) and send it a SDC
' multiline command.
'
CALL DevClear (0, 6)                    'clear the multimeter
'
```

```
' Instruct the multimeter (board number 0, GPIB address 6) to
' reset itself (*RST); measure volts direct current (VDC);
' select range setting number 2 (TRIGGER 2); trigger and
' start a measurement (*TRG); send the measurement results
' back (VAL?); and append a linefeed character with EOI
' asserted as the last byte sent to the meter.
'
CALL Send(0, 6, "*RST;VDC;RANGE 2;*TRIGGER 2;TRG;VAL?",NLend)
'
' Read the last measured value from the multimeter (board 0,
' GPIB address 6) into a string .  Stop reading when the
' meter displays the END message.
'
CALL Receive(0, 6, Reading$, STOPend)
PRINT Reading$
END
```

同じ結果は標準のNI-488ファンクションを使っても得られます。次にそのQuickBASICのプログラムを示します。

```
REM $INCLUDE: 'A:\GPIB-98T\QBDECL.BAS'
DIM Reading AS string * 30
CALL ibdev(0, 6, 0, 12, 1, 0, Fluke45%)  'open device
CALL ibclr(Fluke45%)                                'clear device
CALL ibwrt(Fluke45%, "*RST; VDC; RANGE 2; TRIGGER 2; *TRG; VAL?")
CALL ibrd(Fluke45%, Reading$)
PRINT Reading$
END
```

要約

この章では、GPIB-98Turboインターフェースボードをどのようにインストールするか、また、NI-488.2ソフトウェアをどのように自動的にインストールするかについて学びました。さらに、短いプログラムをどのように書くかを学びました。これで実際のアプリケーションプログラムを書けるようになります。

さらに詳細な例題プログラムは第3章に述べられています。そしていくつかの例が*NI-488.2 Software Reference Manual for MS-DOS* に紹介されています。

第３章
ＮＩ－４８８．２ルーチンによる拡張プログラムの作成

この章では、NI-488.2 for MS-DOSハンドラーとそのルーチン及びそれらの機能について紹介します。また、NI-488.2ルーチンがいかに容易に使えるかを見るためにQuickBASICで例題プログラムを作成します。

NI-488.2ルーチンを使用することにより、IEEE-488.2 - 1987規格のすべてを利用することができます。これらのルーチンは、IEEE-488.2で定義されているコントローラのコマンドとプロトコルと完全に互換性があります。NI-488.2ルーチンは、*NI-488.2 Software Reference Manual for MS-DOS* の中でさらに詳しく述べられています。

NI-488.2ルーチンは容易に習得でき使うことができます。つまり、数種類のルーチンだけでほとんどのアプリケーションプログラムが書けます。

インターフェースボード

NI-488.2ハンドラーは、2枚までのGPIB-98Turboインターフェースボードをサポートしています。これらのボードは、アプリケーションプログラムからは番号によって参照されます。

- 1枚だけGPIB-98Turboがコンピュータにインストールされている場合は、そのリファレン ス番号は0です。

- 2枚のボードがコンピュータにインストールされている場合は、そのリファレンス番号は 1枚目のGPIB-98Turboが0で、2枚目のボードが1となります。

  どちらのボードが0で、どちらかのボードが1かわからないときは、コンフィギュレーションユーティリティIBCONFを走らせます。IBCONFによってボード番号とボードのベースアドレスの関

係がわかります。つまり、ベースアドレスによってボードを確認します。IBCONFの実行と使用についての追加情報は、*NI-488.2 Software Reference Manual for MS-DOS* の第2章を参照して下さい。

コーリングシンタックス

NI-488.2ルーチンのコーリングシンタックスは、使用する言語によって僅かに違います。簡潔にするために、このマニュアルの中で使われるシンタックスはマイクロソフトQuickBASICとCとなっています。他の言語のシンタックスは、*NI-488.2 Software Reference Manual for MS-DOS* 及び対応する言語インターフェースマニュアルに述べられています。

ＮＩ－４８８．２プログラム作成手順

このセクションでは、NI-488.2ルーチンの使い方を紹介します。例題プログラムを順を追ってQuickBASICで作成していきます。第２章で紹介した以上のいくつかのNI-488.2ルーチンを使用します。このプログラムは、ディレクトリA:\GPIB-98Tの中にQSAMP488.BASファイルに入っています。

この例題プログラムは、フルーク社のモデル45デジタルマルチメータを設定し、10個の電圧測定値を読み込み、そしてその平均値を算出します。エラーの検出と報告はプログラムの中で行います。このプログラムは、フルーク45の計測器を制御する前にバス上のすべてのリスナを検出して認識します。

この例題では、GPIB-98Turboインターフェースボードのボード番号を0としています。

ステップ１　準備

プログラム作成の第1ステップは、ディストリビューションディスケットにあるファイルからNI-488.2ルーチンの定義をロードすることです。

NI-488.2 QuickBASIC言語インターフェース用の宣言ファイルを次のようにインクルードします。

```
REM $INCLUDE: 'A:\GPIB-98T\QBDECL.BAS'
```

エラーをチェックするために、すべてのGPIBエラーを処理するサブルーチンを宣言します。

```
DECLARE SUB gpiberr (msg$)
```

次に示す配列は、IEEE-488アドレスのリストを格納するために使い、バス上のすべてのリスナを検出するために使います。

```
DIM instruments% (31)
DIM result% (31)
DIM Reading AS string * 30
```

ステップ２　初期化

IEEE-488バスとGPIB-98Turboコントローラの初期化では次のことを行います。

- IEEE-488インターフェースボードを各デバイスに対して静的な状態にする
- GPIB-98Turboをコントローラインチャージにする
- GPIB-98Turboをアクティブコントローラ状態(CACS)にする

初期化は、SendIFCで行います。SendIFCの1番目であり、1つだけの引数はGPIB-98Turboインターフェースボード番号です。

```
CALL SendIFC(0) AND EERR
IF IBSTA% AND EERR THEN
    CALL gpiberr("SendIFC Error")
    STOP
END IF
```

ステップ３　全リスナの検出

IEEE-488バスに接続されるすべてのIEEE-488 1次アドレスを格納する配列を作ります。バス上には1台の接続デバイス（1次アドレス:0～30）があると仮定します。アドレスリストの最後には定数NOADDRが付いていなければなりません。このNOADDRは、このプログラムの先頭にインクルードしたQBDECL.BASファイルの中で定義されています。

```
FOR K% = 0 TO 30
    instruments%(K%) = K%
NEXT K%
instruments%(31) = NOADDR
```

次に、IEEE-488バスに接続されているデバイスを検出します。これはFindLstnを使えば容易に行えます。第1引数はGPIB-98Turboのボード番号です。第2引数は、上記で作成された計測器のリストです。第3引数は実際に検出した計測器のアドレスリストです。最後の引数は検出する最大デバイス数です。（これは限界に達すると停止します。）

```
PRINT "Finding all listeners on the bus..."

CALL FindLstn(0, instruments%(), result%(), 31)
IF IBSTA% AND EERR THEN
    CALL gpiberr("FindLstn error")
    STOP
END IF
num.listeners% = ibcnt% -1

PRINT "No. of instruments found = ", num.listeners%
```

ステップ４　計測器の認識

デバイスのアドレスが決定されると、確認のため各デバイスに問い合わせを送ります。この例では、すべての計測器がIEEE-488.2互換であり、488.2の確認問い合わせの*IDN?を認識するものとします。

さらに、FindLstnがGPIB-98Turboインターフェースボードが1次アドレス0であることを検出したと仮定します。（これはデフォルトです）したがって、配列resultの0は省略できます。

```
FOR K% = 1 TO num.listeners%
    CALL Send(0, result%(K%), "*IDN?", NLend)
    IF IBSTA% AND EERR THEN
        CALL gpiberr("Send error")
        STOP
    END IF

    CALL Receive(0, result%(K%), Reading$, STOPend)
    IF IBSTA% AND EERR THEN
        CALL gpiberr("Receive error")
        STOP
    END IF

    pad% = result%(K%) AND &HFF
    PRINT "The instrument at address ";pad%; " is: ",
        LEFT$(Reading$, IBCNT%)
    IF LEFT$(Reading$, 9) = "FLUKE, 45" THEN
        fluke% = pad%
        PRINT "***** We found the fluke 45 *****"
        CALL found(fluke%)
        GOTO progend
    END IF
NEXT K%
PRINT "Did not find the Fluke!"
progend:
CALL IBONL (0,0)
END
```

ステップ５　計測器の初期化

マルチメータが検出されましたから次はクリアを行います。IEEE-488バスに接続されたデバイスの初期化は、DevClearで行います。第1引数はGPIB-98Turboインターフェースのボード番号です。第2引数はマルチメータのIEEE-488アドレスです。

```
CALL DevClear(0, fluke%)
IF IBSTA% AND EERR THEN
    CALL gpiberr("DevClear error")
    STOP
END IF

CALL Send(0, fluke%, "*RST", NLend)
IF IBSTA% AND EERR THEN
    CALL gpiberr("Send *RST error")
    STOP
END IF
```

ステップ６　計測器の設定

初期化した後、計測器は命令受信状態となります。マルチメータを設定するには、Sendルーチンを使ってデバイス固有のコマンドを送ります。Sendルーチンの第1引数はGPIBインターフェースのボード番号です。第2引数はマルチメータに送るストリングバイトです。

この例題のデバイスコマンドは、オートレンジ(AUTO)を使って交流電圧(VAC)を測定し、計測を開始する前にコントローラからのトリガ(TRIGGER 2)を待つように命令します。最後の命令(*SRE 16)は、計測が完了し、その結果を送信する準備ができたときにIEEE-488サービスリクエストの信号線(SRQ)を送出します。最後の引数NLendは、QBDECL.BASの中に定義されている定数です。Sendを使ってマルチメータに送られるメッセージの最後にEOI送出と同時にラインフィードキャラクタを付加します。

実際に計測器にトリガをかけて、測定値を読み込むためのコールは、10回繰り返すループの中に置きます。

```
CALL Send(0, fluke%, "VAC; AUTO; TRIGGER 2; *SRE 16", NLend)
IF IBSTA% AND EERR THEN
    CALL gpiberr("Send setup error")
    STOP
END IF
SUM = 0.0
FOR M% = 1 TO 10
```

ステップ７　計測器のトリガ

前のステップでマルチメータは計測を開始する前にトリガを待つように設定されました。ここで、マルチメータにトリガコマンドを送りたいとします。これを行うには、Triggerルーチンを使います。しかし、フルーク45はIEEE-488.2互換ですので、トリガコマンド*TRGを送ることができます。他のコマンドのVAL?は次にトリガされた読み込み値をIEEE-488.2の出力バッファに送るようにメータを設定します。

```
CALL Send(0, fluke%, "*TRG; VAL?", NLend)
IF IBSTA% AND EERR THEN
    CALL gpiberr("Send trigger error")
    STOP
END IF
```


ステップ８　測定完了を待つ

メータがトリガされた後、計測を行い、フロントパネルにその値を表示します。これが終了すると、メータはSRQを送出します。

TestSRQまたはWaitSRQルーチンのどちらかを使ってSRQの送出を検出できます。計測完了を待っている間に何か実行したいプロセスがある場合、TestSRQを使うことができます。例えば、現時点では特に行うことがありませんので、WaitSRQルーチンを使用します。

WaitSRQの第1引数はGPIB-98Turboのボード番号です。第2引数はWaitSRQによって返されるフラグで、SRQが送出されているかどうかを示します。

```
CALL WaitSRQ(0, SRQasserted%)
IF SRQasserted% = 0 THEN
    CALL gpiberr("WaitSRQ Error")
    STOP
END IF
```

SRQの検出後、状態を確認するためにメータをポーリングします。これは、ReadStatusByteを使って行われます。このプロシージャの第1引数はGPIB-98Turboのボード番号です。最後の引数は、ReadStatusByteが計測器のステータスバイトをストアするために使う変数です。

```
CALL ReadStatusByte(0, fluke%, status%)
IF IBSTA% AND EERR THEN
    CALL gpiberr("ReadStatusByte error")
    STOP
END IF
```

ステータスバイトが得られたら、メータが送るメッセージがあるかどうかを確認する必要があります。ステータスバイトのビット4をチェックすることによって調べられます。

```
IF (status% AND &H010) <> &H010 THEN
    CALL gpiberr("Improper status byte")
    PRINT "Status byte: "; HEX$(status%)
    STOP
END IF
```

ステップ９　計測値の読み込み

Receiveファンクションを使ってIEEE-488バスを通して計測値を読み込みます。第1引数はGPIBインターフェースのボード番号です。第2引数はマルチメータのIEEE-488アドレスです。第3引数はReceiveファンクションがマルチメータからデータバイトを格納するストリングです。最後の引数はReceiveファンクションがENDメッセージと一緒に送られたバイトを受け取ると終了することを示します。そしてループが繰り返されます。平均値が10回の測定後に表示されます。

```
CALL Receive(0, fluke%, Reading$, STOPend)
IF IBSTA% AND EERR THEN
    CALL gpiberr("Receive error")
    STOP
END IF

rd$=LEFT$(Reading$,IBCNT%)
PRINT "Reading: "; rd$
SUM = SUM + VAL(Reading$)
NEXT M%
PRINT "The average of the 10 reading is", SUM/10
END SUB
```

QuickBASICの完全なアプリケーションプログラム

次のプログラムは前のステップを１つのプログラムにまとめたものです。

```
REM $INCLUDE: 'A:\GPIB-98T\QBDECL.BAS'

DECLARE SUB gpiberr (msg$)
DECLARE SUB found(fluke%)

DIM instruments% (31)
DIM result% (30)
DIM Reading AS STRING*10
    SHARED

' The board must be the Controller-In-Charge to perform
' the Find All Listeners protocol.
```

```
CLS
CALL SendIFC(0) AND EERR
IF IBSTA% AND EERR THEN
    CALL gpiberr("SendIFC Error")
    STOP
END IF

' Create an array with all of the valid GPIB primary
' addresses.
' This array will be given to the Find All Listeners
' protocol.

FOR K% = 0 TO 30
    instruments%(K%) = K%
NEXT K%
instruments%(31) = NOADDR

' Find all of the listeners on the bus.

PRINT "Finding all listeners on the bus..."

CALL FindLstn(0, instruments%(), result%(), 31)
IF IBSTA% AND EERR THEN
    CALL gpiberr("FindLstn error")
    STOP
END IF
num.listeners% = ibcnt% -1

PRINT "No. of instruments found = ", num.listeners%

' Now send the *IDN? command to each of the devices that we
' found.
'
' The GPIB-98Turbo interface board is at address 0 by default.
' The board does not respond to *IDN?, so skip it.

FOR K% = 1 TO num.listeners%
    CALL Send(0, result%(K%), "*IDN?", NLend)
    IF IBSTA% AND EERR THEN
        CALL gpiberr("Send error")
        STOP
    END IF

    CALL Receive(0, result%(K%), Reading$, STOPend)
    IF IBSTA% AND EERR THEN
        CALL gpiberr("Receive error")
        STOP
    END IF

    pad% = result%(K%) AND &HFF
    PRINT "The instrument at address ";pad%; " is: ",
        LEFT$(Reading$, IBCNT%)
```

```
    IF LEFT$(Reading$, 9) = "FLUKE, 45" THEN
        fluke% = pad%
        PRINT "***** We found the fluke 45 *****"
        GOTO progend
    END IF
NEXT K%
PRINT "Did not find the Fluke!"
progend:
CALL IBONL (0,0)
END

sub found(fluke%) STATIC

' Reset the Fluke.

CALL DevClear(0, fluke%)
IF IBSTA% AND EERR THEN
    CALL gpiberr("DevClear error")
    STOP
END IF

CALL Send(0, fluke%, "*RST", NLend)
IF IBSTA% AND EERR THEN
    CALL gpiberr("Send *RST error")
    STOP
END IF

' Setup for a test. Allow the Fluke to assert
' SRQ when it has a message to send.

CALL Send(0, fluke%, "VAC; AUTO; TRIGGER 2; *SRE 16", NLend)
IF IBSTA% AND EERR THEN
    CALL gpiberr("Send setup error")
    STOP
END IF
SUM = 0
FOR M% = 1 TO 10

    ' Trigger the Fluke.

    CALL Send(0, fluke%, "*TRG; VAL?", NLend)
    IF IBSTA% AND EERR THEN
        CALL gpiberr("Send trigger error")
        STOP
    END IF

    ' Wait for the Fluke to assert SRQ, meaning it is
    ' ready with the measurement.

    CALL WaitSRQ(0, SRQasserted%)
    IF SRQasserted% = 0 THEN
        CALL gpiberr("WaitSRQ Error")
```



```
        STOP
    END IF

    ' Read its status byte.  Make sure that the MAV (Message
    ' Available) bit is set.

    CALL ReadStatusByte(0, fluke%, status%)
    IF IBSTA% AND EERR THEN
        CALL gpiberr("ReadStatusByte error")
        STOP
    END IF

    IF (status% AND &H010) <> &H010 THEN
        CALL gpiberr("Improper status byte")
        PRINT "Status byte: "; HEX$(status%)
        STOP
    END IF

    ' Read the measurement.

    CALL Receive(0, fluke%, Reading$, STOPend)
    IF IBSTA% AND EERR THEN
        CALL gpiberr("Receive error")
        STOP
    END IF

    rd$=LEFT$(Reading$,IBCNT%)
    PRINT "Reading: "; Reading$
    SUM = SUM + VAL(Reading$)
NEXT M%

PRINT "The average of the 10 readings is", SUM/10
END SUB
```

エラー処理サブルーチン

メインプログラムは、エラーが発生するとこのルーチンをコールします。このサブルーチンはすべてのエラーに対してフルメッセージで与えられます。単純にしたければ、最初の3行と最終行だけを使います。

```
SUB gpiberr(msg$) STATIC

LOCATE 15,1
PRINT msg$
```

```
PRINT "IBSTA=&H"; HEX$(IBSTA%); " < ";
IF IBSTA% AND EERR THEN PRINT " ERR";
IF IBSTA% AND TIMO THEN PRINT " TIMO";
IF IBSTA% AND EEND THEN PRINT " END";
IF IBSTA% AND SRQI THEN PRINT " SRQI";
IF IBSTA% AND RQS THEN PRINT " RQS";
IF IBSTA% AND CMPL THEN PRINT " CMPL";
IF IBSTA% AND LOK THEN PRINT " LOK";
IF IBSTA% AND PREM THEN PRINT " REM";
IF IBSTA% AND CIC THEN PRINT " CIC";
IF IBSTA% AND AATN THEN PRINT " ATN";
IF IBSTA% AND TACS THEN PRINT " TACS";
IF IBSTA% AND LACS THEN PRINT " LACS";
IF IBSTA% AND DTAS THEN PRINT " DTAS";
IF IBSTA% AND DCAS THEN PRINT " DCAS";
PRINT ">                                    "

PRINT "IBERR"; IBERR%;
IF IBERR% = EDVR THEN PRINT " EDVR <DOS Error>        "
IF IBERR% = ECIC THEN PRINT " ECIC <Not CIC>          "
IF IBERR% = ENOL THEN PRINT " ENOL <No Listener>      "
IF IBERR% = EADR THEN PRINT " EADR <Address error>    "
IF IBERR% = EARG THEN PRINT " EARG <Invalid argument>"
IF IBERR% = ESAC THEN PRINT " ESAC <Not Sys Ctrlr>    "
IF IBERR% = EABO THEN PRINT " EABO <Op. aborted>      "
IF IBERR% = ENEB THEN PRINT " ENEB <No GPIB board>    "
IF IBERR% = EOIP THEN PRINT " EOIP <Async I/O in prg>"
IF IBERR% = ECAP THEN PRINT " ECAP <No capability>    "
IF IBERR% = EFSO THEN PRINT " EFSO <File sys. error>  "
IF IBERR% = EBUS THEN PRINT " EBUS <Command error>    "
IF IBERR% = ESTB THEN PRINT " ESTB <Status byte lost>"
IF IBERR% = ESRQ THEN PRINT " ESRQ <SRQ stuck on>     "
IF IBERR% = ETAB THEN PRINT " ETAB <Table Overflow>   "

PRINT "IBCNT="; IBCNT%; "                             "
' CALL the IBONL function to disable the handler and software.
CALL IBCONL(0,0)
END SUB
```

コンパイルとリンク

実行プログラムを作成するには、次の手順に従って行います。

1．インストレーション時に作成されたA:\GPIB-98Tディレクトリ
　からQuickBASIC4.5言語インターフェース(QBIB.OBJ)を
　QuickBASICディレクトリ(A:\QB45)にコピーします。

2．アプリケーションプログラムをコンパイルします。（作成した例題プログラムはACVOLTS.BASのファイルにあるものとします。）

```
bc /o acvolts.bas;                <リターン>
```

3．次のようにコマンドを入力して言語インターフェースとリンクします。

```
link acvolts.obj+gpib.obj;        <リターン>
```

リンクコマンドは実行可能ファイルACVOLTS.EXEを作成します。

4．フルーク45DVMがあれば、次のコマンドを入力してプログラムを走らせます。

```
acvolts                           <リターン>
```

Ｃの完全なアプリケーションプログラム

マイクロソフトＣで書かれたプログラムを次に示します。

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

/* decl.h contains constants, declarations, and function
 * prototypes.
 */

#include <a:\gpib-98t\decl.h>

/*
 * found is a function called when the Fluke 45 is identified
 * as a Listener on the GPIB.  gpiberr is an error function
 * that is called when a 488.2 function fails.
 */

void found (unsigned int fluke);
void gpiberr(char *msg);

#define MAVbit 0x10 /* Position of the Message Available bit. */
```

```
char     buffer[101];   /* Data received from the Fluke 45.          */
int      loop,          /* FOR loop counter and array index.         */
         m,             /* FOR loop counter.                         */
         num_listeners, /* Number of Listeners on GPIB.              */
         SRQasserted;   /* Set to indicate if SRQ is asserted.       */
double   sum;           /* Accumulator of measurements.              */
unsigned int   instruments[32],/* Array of primary addresses.        */
         result[31],    /* Array of listen addesses.                 */
         fluke,         /* Primary address of the Fluke 45.          */
         pad,           /* Primary address of Listener on GPIB.      */
         statusByte;    /* Serial Poll Response Byte.                */


void main() {

    system("cls");

/*
 * Your board must be the Controller-In-Charge to find all
 * Listeners on the GPIB.  To accomplish this, the function
 * SendIFC is called.  If the error bit ERR is set in ibsta,
 * call gpiberr with an error message.
 */

    SendIFC(0);
    if (ibsta & ERR) {
        gpiberr ("SendIFC error");
        exit(1);
    }

/*
 * Create an array containing all valid GPIB primary addresses.
 * This array (instruments) will be given to the function
 * FindLstn to find all Listeners.  The constant NOADDR,
 * defined in decl.h, signifies the end of the array.
 */

    for (loop = 0; loop <= 30; loop++) {
    instruments[loop] = loop;
    }
    instruments[31] = NOADDR;

/*
 * Print message to tell user that the program is searching
 * for all active Listeners.  Find all of the Listeners on
 * the bus.  Store the listen addresses in the array RESULT.
 * If the error bit ERR is set in ibsta, call gpiberr with
 * an error message.
 */

    printf ("Finding all listeners on the bus...\n");
    printf("\n");
```



```
    FindLstn (0, instruments, result, 31);
    if (ibsta & ERR) {
        gpiberr ("FindLstn error");
        exit(1);
    }

/*
 * Assign the value of ibcnt to the variable num_listeners.
 * The GPIB interface board is detected as a Listener on the
 * bus; however, it is not included in the final count of the
 * number of Listeners.   Print the number of Listeners found.
 */

    num_listeners = ibcnt - 1;

    printf ("Number of instruments found = %d\n", num_listeners);

/*
 * Send the *IDN? command to each device that was found.  Your
 * GPIB interface board is at address 0 by default.  The board
 * does not respond to *IDN?, so skip it.
 *
 * Establish a FOR loop to determine if the Fluke 45 is a
 * Listener on the GPIB.  The variable LOOP will serve as a
 * counter for the FOR loop and as the index to the array
 * Result.
 */

    for (loop = 1; loop <= num_listeners; loop++) {

   /*
    * Send the identification query to each listen address in
    * the array result.  The constant NLend, defined in decl.h,
    * instructs the function Send to append a linefeed
    * character with EOI asserted to the end of the message.
    * If the error bit ERR is set in ibsta, call gpiberr
    * with an error message.
    */

        Send (0, result[loop], "*IDN?", 5L, NLend);
        if (ibsta & ERR) {
            gpiberr ("Send error");
            exit(1);
        }

   /*
    * Read the name identification response returned from each
    * device.  Store the response in the array buffer.  The
    * constant STOPend, defined in decl.h, instructs the
    * function Receive to terminate the read when END is
    * detected.  If the error bit ERR is set in ibsta,
    * call gpiberr with an error message.
    */
```

```
        Receive (0, result[loop], buffer, 100L, STOPend);
        if (ibsta & ERR) {
            gpiberr ("Receive error");
            exit(1);
        }

    /*
     * The low byte of the listen address is the primary
     * address.  Assign the variable PAD the primary address
     * of the device.  The macro GetPAD, defined in decl.h,
     * returns the low byte of the listen address.
     */

        pad = GetPAD(result[loop]);

    /*
     * Use the null character to mark the end of the data
     * received in the array buffer.  Print the primary
     * address and the name identification of the device.
     */

        buffer[ibcnt] = '\0';
        printf("The instrument at address %d is a %s\n", pad,
            buffer);

    /*
     * Determine if the name identification is the FLUKE 45.  If
     * it is the FLUKE 45, assign pad to fluke,  print message
     * that the FLUKE 45 has been found, call the function
     * found, and terminate the FOR loop.
     */

        if (strcmp(buffer, "FLUKE, 45", 9) == 0) {
            fluke = pad;
            printf ("***** We found the fluke *****\n");
            found(fluke);
            break;
        }

    }   /*  End of FOR loop */

    if (loop > num_listeners)
        printf ("Did not find the Fluke!\n");

    }

/* Call the ibonl function to disable the hardware and
 * software.
 */

    ibonl (0,0);

}
```



```
/*=============================================================
 *                          Function found
 * This function is called if the Fluke 45 has been identified
 * as a Listener in the array Result.  The variable fluke is
 * the primary address of the FLUKE 45.  Ten measurements are
 * read from the Fluke 45 and the average of the sum is
 * calculated.
 *=============================================================
 */

void found(unsigned int fluke) {

/* Reset the Fluke 45 using the functions DevClear and Send. */
/*
 * DevClear will send the GPIB Selected Device Clear (SDC)
 * command message to the Fluke 45.  If the error bit ERR is
 * set in ibsta, call gpiberr with an error message.
 */

    DevClear (0, fluke);
    if (ibsta & ERR) {
        gpiberr ("DevClear error");
        exit(1);
    }

/*
 * Use the function Send to send the IEEE-488.2 reset command
 * (*RST) to the Fluke 45.  The constant NLend, defined in
 * decl.h, instructs the function Send to append a linefeed
 * character with EOI asserted to the end of the message.
 * If the error bit ERR is set in ibsta, call gpiberr with
 * an error message.
 */

    Send (0, fluke, "*RST", 4L, NLend);
    if (ibsta & ERR) {
        gpiberr ("Send *RST error");
        exit(1);
    }

/*
 * Use the function Send to send device configuration commands
 * to the Fluke 45.  Instruct the Fluke 45 to measure volts
 * alternating current (VAC) using auto-ranging (AUTO), to wait
 * for a trigger from the GPIB interface board (TRIGGER 2),
 * and to assert the IEEE-488 Service Request line, SRQ, when
 * the measurement has been completed and the Fluke 45 is
 * ready to send the result (*SRE 16).  If the error bit ERR
 * is set in ibsta, call gpiberr with an error message.
 */
```

```
    Send (0, fluke, "VAC; AUTO; TRIGGER 2; *SRE 16, 29L, NLend);
    if (ibsta & ERR) {
        gpiberr ("Send setup error");
        exit(1);
    }

/* Initialized the accumulator of the ten measurements to
 * zero.
 */

    sum = 0.0;

/*
 * Establish FOR loop to read the ten measuements.  The
 * variable m will serve as the counter of the FOR loop.
 */
    for (m=0; m < 10; m++) {

        /*
         * Trigger the Fluke 45 by sending the trigger command
         * (*TRG) and request a measurement by sending the
         * command "VAL1?".  If the error bit ERR is set in ibsta,
         * call gpiberr with an error message.
         */

            Send (0, fluke, "*TRG; VAL?", 10L, NLend);
            if (ibsta & ERR) {
                gpiberr ("Send trigger error");
                exit(1);
           }

        /*
         * Wait for the Fluke 45 to assert SRQ, meaning it is
         * ready to send a measurement.  If SRQ is not
         * asserted within the timeout period, call gpiberr
         * with an error message.  The timeout period by default
         * is 10 seconds.
         */

            WaitSRQ (0, &SRQasserted);
            if (ibsta & ERR) {
                gpiberr ("WaitSRQ error");
                exit(1);
            }
        /*
         * Read the serial poll status byte of the FLuke 45.  If
         * the error bit ERR is set in ibsta, call gpiberr with
         * an error message.
         */
```



```
        ReadStatusByte (0, fluke, &statusByte);
        if (ibsta & ERR) {
            gpiberr ("ReadStatusByte error");
            exit(1);
        }

    /*
     * Check if the Message Available Bit (bit 4) of the
     * return status byte is set.  If this bit is not set,
     * print the status byte and call gpiberr with an
     * error message.
     */

        if (!(statusByte & MAVbit)) {
            gpiberr ("Improper status byte");
            printf (" Status byte = 0x%x\n", statusByte);
            exit(1);
        }

    /*
     * Read the Fluke 45 measurement.  Store the measurement
     * in the variable buffer.  The constant STOPend,
     * defined in decl.h, instructs the function receive to
     * terminate the read when END is detected.  If the error
     * bit ERR is set in ibsta, call gpiberr with an error
     * message.
     */

        Receive (0, fluke, buffer, 100L, STOPend);
        if (ibsta & ERR) {
            gpiberr ("Receive error");
            exit(1);
        }

    /*
     * Use the null character to mark the end of the data
     * received in the array buffer.  Print the measurement
     * received from the Fluke 45.
     */

        buffer[ibcnt] = '\0';
        printf("Reading :  %s\n", buffer);

    /* Convert the variable buffer to its numeric value and
     * add to the accumulator.
     */
        sum = sum + atof(buffer);

} /* Continue FOR loop until 10 measurements are read. */

/* Print the average of the ten readings. */
```

```
    printf ("The average of the 10 readings is : %f\n", sum/10);

}

/*============================================================
 *                        Function gpiberr
 * This function will notify you that a NI-488.2 routine
 * failed by printing an error message.  The status variable
 * ibsta will also be printed in hexadecimal along with the
 * mnemonic meaning of the bit position.  The status variable
 * iberr will be printed in decimal along with the mnemonic
 * meaning of the decimal value.  The status variable ibcntl
 * will be printed in decimal.
 *
 * The NI-488 function ibonl is called to disable the hardware
 * and software.
 *============================================================
 */

void gpiberr(char *msg)
{
    printf ("%s\n", msg);

    printf ("ibsta = &H%x  <", ibsta);
    if (ibsta & ERR ) printf (" ERR");
    if (ibsta & TIMO) printf (" TIMO");
    if (ibsta & END ) printf (" END");
    if (ibsta & SRQI) printf (" SRQI");
    if (ibsta & RQS ) printf (" RQS");
    if (ibsta & CMPL) printf (" CMPL");
    if (ibsta & LOK ) printf (" LOK");
    if (ibsta & REM ) printf (" REM");
    if (ibsta & CIC ) printf (" CIC");
    if (ibsta & ATN ) printf (" ATN");
    if (ibsta & TACS) printf (" TACS");
    if (ibsta & LACS) printf (" LACS");
    if (ibsta & DTAS) printf (" DATS");
    if (ibsta & DCAS) printf (" DCAS");
    printf (">\n");

    printf ("iberr= %d", iberr);
    if (iberr == EDVR) printf (" EDVR <DOS Error>\n");
    if (iberr == ECIC) printf (" ECIC <Not CIC>\n");
    if (iberr == ENOL) printf (" ENOL <No Listener>\n");
    if (iberr == EADR) printf (" EADR <Address error>\n");
    if (iberr == EARG) printf (" EARG <Invalid argument>\n");
    if (iberr == ESAC) printf (" ESAC <Not Sys Ctrlr>\n");
    if (iberr == EABO) printf (" EABO <Op. aborted>\n");
    if (iberr == ENEB) printf (" ENEB <No GPIB board>\n");
    if (iberr == EOIP) printf (" EOIP <Async I/O in prg>\n");
    if (iberr == ECAP) printf (" ECAP <No capability>\n");
    if (iberr == EFSO) printf (" EFSO <File sys. error>\n");
```



```
    if (iberr == EBUS) printf (" EBUS <Command error>\n");
    if (iberr == ESTB) printf (" ESTB <Status byte lost>\n");
    if (iberr == ESRQ) printf (" ESRQ <SRQ stuck on>\n");
    if (iberr == ETAB) printf (" ETAB <Table Overflow>\n");
    if (iberr == EDVR) printf (" EDVR <DOS Error>\n");

    printf ("ibcnt = %d\n", ibcnt);
    printf ("\n");

/* Call the ibonl function to disable the hardware and
 * software.
 */

    ibonl (0,0);

}
```

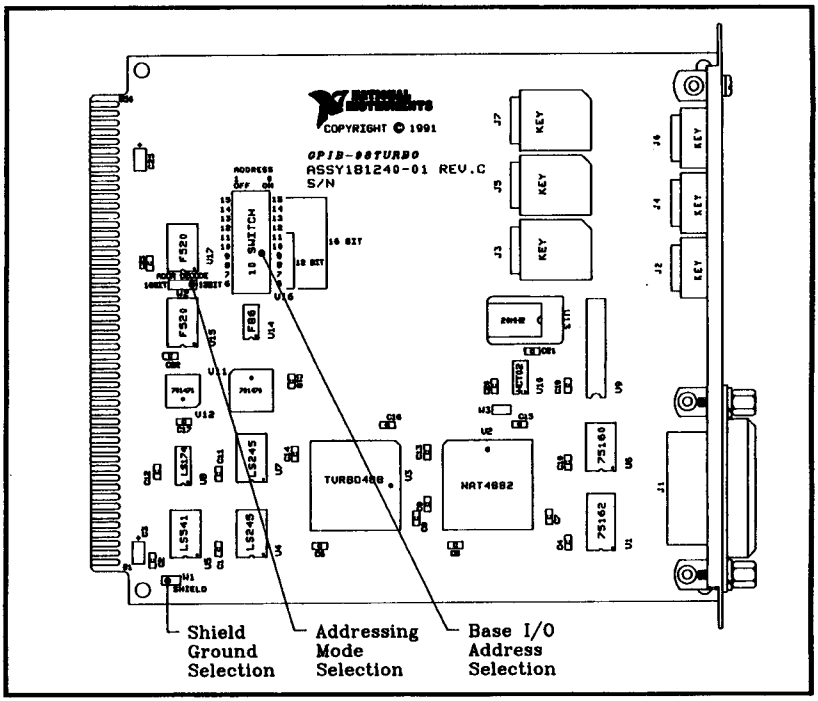
ヒント

ソフトウェアとハードウェアがインストールされ、いくつかのNI-488.2ルーチンを使用したら、GPIB-98Turboに用意されているすべてのNI-488.2ルーチンの詳細について*NI-488.2 Software Reference Manual for MS-DOS*をお読み下さい。これらのファンクションと機能について読まれた後は、*NI-488.2 Software Reference Manual for MS-DOS*の第６章に述べられているIBICプログラムを使ってインタラクティブな環境でプログラマブル計測器やデバイスでこれらのファンクションを使って実行して下さい。

付録A
ハードウェアとソフトウェアのコンフィギュレーション設定の変更

この付録では、GPIB-98Turboインターフェースボードを構成してインストールする手順が説明されています。ハードウェア設定を変更する前にコンピュータの電源を切り、ボードを抜きます。

図A-1にGPIB-98Turboのコンフィギュレーションジャンパーとスイッチの配置を示します。



図A-1　GPIB-98Turbo部品配置図

スイッチとジャンパー位置

工場出荷時のコンフィギュレーションジャンパーとスイッチの設定は、PC9801シリーズのほとんどの機種に合うように設定されています。これらの設定はNI-488.2 for MS-DOSハンドラーの中のソフトウェアコンフィギレーションと完全に一致していなくてはなりません。GPIB-98Turbo上のスイッチ及びジャンパーの工場出荷時の設定と設定可能なコンフィギュレーションを表A-1に示します。

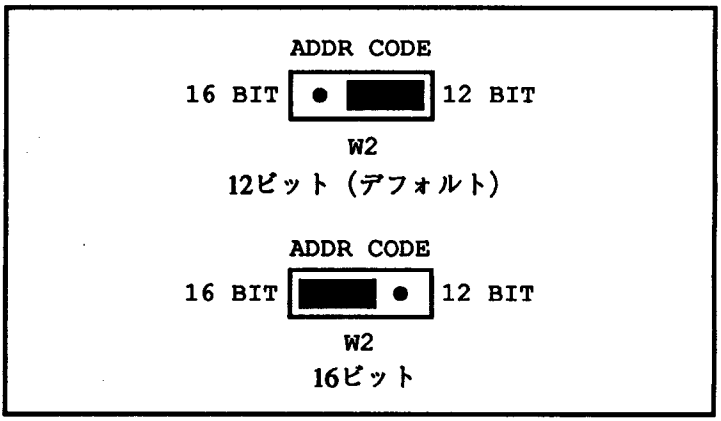
表A-1 工場出荷時設定と設定可能なコンフィギュレーション

スイッチとジャンパー	デフォルト	変更
アドレシングモード	12ビット	16ビット
ベースI/Oアドレス	07D0(hex)	00D0, 01D0, 02D0 等
シールド コンフィギュ レーション	ロジックグラウンドと シールドグラウ ンドを接続	非接続

12ビットと16ビットのアドレシングモードの選択

GPIB-98Turboは、W2のジャンパーが12ビットアドレシングモードに工場出荷時に設定されています。これはほとんどのPC9801シリーズコンピュータで使用することができます。16ビットアドレシングモードを必要とする場合は次の手順で行います。

1．図A-2に示すようにGPIB-98Turbo上のW2と印刷されている3ピンが配置されています。さらにADDRDECODEとも印刷されています。



図A-2　アドレシングモードの選択

2．中央ピンと左ピンに小さなジャンパーを差し込みます。これで16ビット側になります。

ベースI/Oアドレス

ベースI/Oアドレスは、次に示す手順にしたがって行います。ここでは、ベースI/Oアドレスとして07D0(hex)を選択することとします。

1．ベースI/Oアドレスを16進数から2進数に変換します。

2．下6ビットは無視して、残りのビットは0000 0111 11となります。

	0	7	D	0
07D0 =	0000	0111	1101	0000
使用ビット =	0000	0111	11	

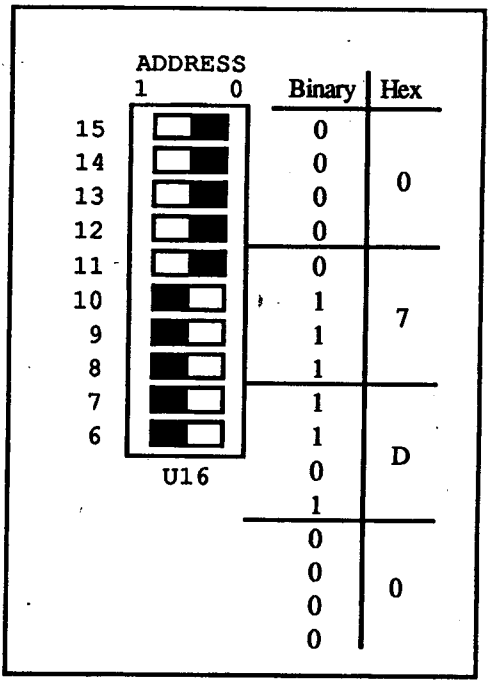
3．インターフェースボード上のディップスイッチU16をこの値に合わせます。6と印刷されているスイッチが最下位ビットです。

アドレス =	0	0	0	0	0	1	1	1	1	1						
スイッチU16 =	15	14	13	12	11	10	9	8	7	6						

4．ディップスイッチをアドレスビットに合うように設定します。

アドレスビットが0なら、0とマークされた側を押し下げます。
アドレスビットが1なら、1とマークされた側を押し下げます。

図A-3に示すように、スイッチの黒く塗られた側が押し下げる側です。



図A-3　ベースI/Oアドレス設定

注意）12ビットアドレスモードでは、上位のビット（スイッチ12～15）は使用しません。したがって、スイッチ6～11だけを設定します。

5．ボードを再インストールした後、IBCONFを使ってベースI/Oアドレスを変更します。

これでベースI/Oアドレスが設定されます。ベースI/Oアドレスに関する詳しい情報は、「ベースI/Oアドレス選択時の注意」をお読み下さい。必要でなければ、シールドコンフィギュレーションの設定を行うために、ステップ3「シールドグラウンドの選択」へ進みます。

ベースI/Oアドレス設定時の注意

ベースI/Oアドレスは、U16のスイッチによって決められます。工場出荷時、このベースI/Oアドレスは07D0(hex)に設定されています。GPIB-98Turboは、連続した32バイトのI/O空間を使用します。例えば、ベースI/Oアドレス07D0(hex)は、07D0から07EFのすべてのアドレスを使用します。

GPIB-98Turboは、12ビットまたは16ビットのアドレス（xxD0）を構成できます。アドレスxxD0のxxは、00～07です。例えば、次のアドレスがベースI/Oアドレスとして設定できます。

00D0, 01D0, 02D0, 03D0, 04D0, 05D0, 06D0, 07D0

アドレスの下位6ビット(0～5)は、GPIB-98TurboによってGPIBアダプタのレジスタを選択するために使われ、変更することはできません。したがって、ベースI/Oアドレスが決まると、A5からA0は常にバイナリで010000となります。

コンピュータに2枚のGPIB-98Turboインターフェースボードをインストールする場合、それぞれのボードには異なったベースI/Oアドレスを設定しなければなりません。NI-488.2 for MS-DOSソフトウェアハンドラーは、GPIB-98Turboのベース I/Oアドレスとして次のデフォルト値に設定されています。

ボード	ベースアドレス
gpib0	07D0 (hex)
gpib1	06D0 (hex)

コンピュータにインストールされている他のボードによって、すでにこの空間が使われていないことを確認して下さい。このI/Oアドレス空間が他で使われている場合は、GPIB-98Turboかまたは他のデバイスのどちらかのI/Oアドレスを変更して下さい。

もしGPIB-98Turboのベース I/Oアドレスをデフォルト設定から変更する場合は、付録DのGPIB-98Turboハードウェア／ソフトウェアコンフィギュレーションフォームに新しい設定を記録しておいて下さい。また、ベースI/Oアドレスを変更した場合は、NI-488.2 MS-DOSハ

ンドラーの変更も同じ設定にしなければなりません。（後述のソフトウェアコンフィギュレーションを参照して下さい）

アドレスのぶつかり

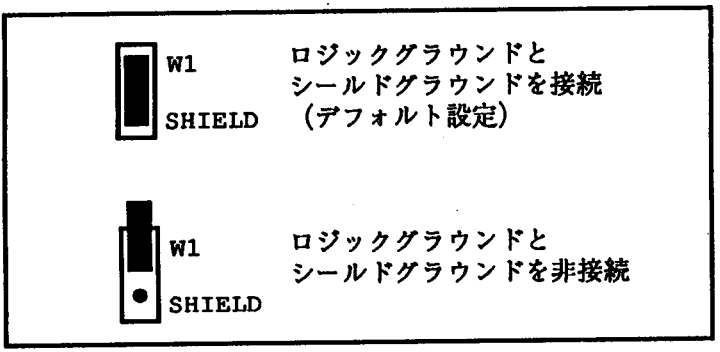
I/Oアドレスは、GPIB-98Turboインターフェースボードと他のPC9801用のボードやアダプタとぶつかる場合があります。ベースI/Oアドレスがぶつかるときの現象は様々で、極端な例ではコンピュータが立ち上がらないこともあります。それらが現れると、おかしな動作をするような現象として表面化します。

ナショナルインスツルメンツは、動作するようなデフォルトベースI/Oアドレスを選択するようにしています。しかし、PC9801コンピュータで使える多くの異なるインターフェースボードがあるため、すべてのシステムで動作を保証するベースI/Oアドレスを選択することは不可能です。したがって、インストレーションを進める前にシステム内のベースI/Oアドレスの割り付けを確かめておいて下さい。

シールドグラウンドの選択

GPIB-98Turboは工場出荷時にロジックグラウンドとシールドグラウンドが接続された位置にジャンパーが設定されています。このコンフィギュレーションは、GPIB-98Turboに備えられている、PC9801シリーズコンピュータまたは互換機から発するEMIを最小限にするもので、推奨する設定です。しかし、アプリケーションでシールドグラウンドとロジックグラウンドを切り離す必要がある場合は、W1のジャンパーを取り外し、ジャンパーピンの片方だけを差し込むようにします。

ロジックグラウンドとシールドグラウンドの接続及び非接続のジャンパー設定を図A-4に示します。



図A-4　グラウンドジャンパーの設定

新規設定

次の空白部分に新しいベースI/Oアドレスとグラウンド設定を記録しておきます。そうすれば、コンフィギュレーション時やソフトウェアのインストール時に便利です。

GPIB-98Turbo	新しい設定
ベースI/Oアドレス	_____
グラウンド	_____

ソフトウェアコンフィギュレーション

ベースI/Oアドレスを変更した場合、NI-488.2 for MS-DOSハンドラーも適切な変更を行って一致させなければなりません。これを行うには、コンフィギュレーションユーティリティIBCONFを走らせ、ベースI/Oアドレスのパラメータをエディットしなければなりません。IBCONFの実行手順については、*NI-488.2 Software Reference Manual for MS-DOS*の第2章を参照して下さい。

この付録の残りの部分では、*NI-488.2 Software Reference Manual for MS-DOS* で説明されていないGPIB-98TurboのIBCONFのいくつかの特徴を述べます。この付録の残りの部分は、*NI-488.2 Software Reference Manual for MS-DOS*の第2章のIBCONFのセクションと共に必ずお読み下さい。

ＩＢＣＯＮＦでの機種選択

コンフィギュレーションユーティリティIBCONFは、NEC PC9801シリーズコンピュータで使用し、*NI-488.2 Software Reference Manual for MS-DOS*で述べられていない特徴を説明します。ここでは、機種選択画面について説明します。

次のステップでは、PC9801上でIBCONFを走らせる一般的なステップです。さらに完全な手順はこのマニュアルの第2章の中の「ステップ3 IBCONFによる機種とスロット番号の選択」を参照して下さい。

1．画面上の機種リストから使用する機種を選択します。リストに機種名がない場合はOtherを選択します。

2．Board Characteristics画面に移り、現在GPIB-98Turboがインストールされているスロット番号を選択します。同時に必要であればDMAチャンネルを選択することもできます。

3．リターンキー、ESCキー、またはF9キーを押してこの画面を抜けます。

他のコンピュータにGPIB-98Turboを移した場合、どちらかが変更されていればIBCONFを走らせ、機種とスロット番号を選択しなければなりません。

ある機種を選択すると、その機種で選択可能なスロット番号とDMAチャンネルが表示されます。したがって、存在しないスロットや存在しないDMAチャンネルを選択することはできません。しかし、画面上のリストに載っていない機種の場合は、Otherを選択します。この場合、IBCONFは次の選択を表示します。

- 1から15のスロット番号
- 0から7またはNONEのDMAチャンネル

このときだけは選択可能なDMAチャンネルと使用するスロットを知っていなければなりません。ほとんどの場合、DMAチャンネルは3となります。詳細についてはコンピュータ付属のマニュアルを参照して下さい。

NEC Model Types画面の右側には選択可能なスロット番号とDMAチャンネルが表示されます。上にはすべての選択可能なスロットをリスト表示し、下にはそのスロットに対応するデフォルトDMAチャンネルが表示されます。違う色で表示されるDMAチャンネル番号は標準でないDMAチャンネルを示します。

NEC Model Types画面に入り、機種を変更してもF6キーを押せば元の選択にすぐに戻すことができます。したがって、間違っても容易にやり直すことができます。

バッチモードで機種を選択するには、コンフィギュレーションファイルのどこかに次の行を入力しておきます。

```
NEC model type
```

ここで、model typeはコンピュータの機種名です。機種が確定していれば、この行の後すぐにmodel typeの機種に変更されます。確定機種名のリストは、README.DOSファイルの中に書かれています。機種名はREADME.DOSファイルに書かれている名前と完全に一致していなければなりません。例えば、機種RA21の場合、次のようになります。

```
NEC 9801-RA21
```

機種選択がファイルの中のどこかにあれば、スロット番号の選択はfind name#と対でボードを見つけた後行わなければなりません。

```
slot #
```

#はGPIB-98Turboがインストールされているスロット番号です。例えば、2番スロットを使うと次のようになります。

```
find board0 slot 2
```

付録B
インタラプトラインとDMAチャンネルのパラメータ

インタラプトラインとDMAチャンネルの設定は、コンフィギュレーションユーティリティプログラムIBCONFで変更することができます。IBCONFを使ってインタラプトラインとDMAチャンネルを設定する前に、PC9801コンピュータがもっているオプションを知っておかなければなりません。DMAまたはインタラプトのどちらかを変更するためにIBCONFユーティリティを走らせる前に、この章を十分にお読み下さい。

インタラプト及びDMAを使って

ほとんどすべての場合、ボードはインタラプトとDMAをディスエーブルにしても動作します。IBCONFユーティリティの中でNONEに設定すればインタラプトとDMAチャンネルの両方をディスエーブルにできます。

インタラプトまたはDMAを使用しなければならない場合があります。

- DMAは高速アプリケーションで必要となります。
- 非同期I/Oではインタラプトが必要になります。

（詳細については*NI-488.2 Software Reference Manual for MS-DOS* を参照して下さい。）

インタラプトの選択

GPIB-98Turboインターフェースボードは、コンピュータのI/Oチャンネル上に5本のインタラプト要求線のうち1本を使用します。インタラプト要求線は、IBCONFユーティリティの中で選択します。GPIB-98Turboハードウェアが使用できるインタラプト要求線は、IR3, 5, 6,

12, 13 です。コンピュータに2枚のGPIB-98Turboインターフェースボードをインストールした場合、それぞれのボードに対して異なったインタラプト要求線を使用するように設定しなければなりません。

NI-488.2 for MS-DOSハンドラーには、GPIB-98Turboインタラプト要求レベルが次に示すデフォルト値になっています。

ボード	ＩＲＱライン
gpib0	3
gpib1	5

インタラプトを使用したくなければ、GPIB-98TurboをIRQラインから論理的に切り離します。これは、IBCONFを走らせ、GPIB-98Turboのインタラプトレベルを NONEに設定し、コンピュータをリセットすることによって行うことができます。

GPIB-98Turboインターフェースボードのインタラプトは、コンピュータのプラグイン型インターフェースボードやアダプタが使用しているインタラプトレベルとぶつかる場合があります。ベースI/Oアドレスのぶつかりのように、その現象はコンピュータが立ち上がらなかったり、想定される時間が経過しないと現れなかったりと様々です。それらが現れると、おかしな動作をするような現象として表面化します。

ナショナルインスツルメンツは、動作するようなデフォルトインタラプトレベルを選択するようにしています。しかし、PC9801コンピュータで使える多くの異なるインターフェースボードがあるため、すべてのシステムで動作を保証するインタラプトを選択することは不可能です。したがって、インストレーションを進める前にシステム内のインタラプトの割り付けを確かめておいて下さい。

DMAチャンネルの選択

GPIB-98TurboのデフォルトDMAチャンネルの設定は、次に示す要素によって異なります。

- PC9801シリーズコンピュータの機種
- GPIB-98Turboがインストールされているスロット

ほとんどの場合、デフォルトDMAチャンネルは3です。しかし、3でない場合もあります。これらの例外を除いては、表B-1に示す通りです。使用するコンピュータが表B-1のリストの中にない場合、または次表のリストのスロットにボードがない場合はデフォルトDMAチャンネルは3です。

表B-1　デフォルトDMAチャンネル以外の設定

機　　種	スロット	ﾃﾞﾌｫﾙﾄDMAﾁｬﾝﾈﾙ
9801 E	6	2
9801 F1/F2/VF/VM	4	2
9801 F3/U/UV/CV	2	2
98 XA	1 2	1 2

他のどのデバイスもデフォルトDMAチヤンネルを使用していないのが確かであれば、現在の設定のままにして抜けます。

しかし、他のデバイスがデフォルトDMAチャンネルを使用していれば、GPIB-98TurboのDMA チャンネルを他の設定に変更しなければなりません。1つの例外を除いて、2枚目ののボードのDMAチャンネルはチヤンネル0です。1つの例外とは、98XAコンピュータ上で、スロット4は第2デフォルトDMAチャンネルをもっていません。つまり、チャンネル3しかアクセスすることができません。

DMAチャンネル0は、しばしば他のデバイスで使われています。デフォルトDMAチャンネルとチャンネル0の両方が他のデバイスで使用されていれば、GPIB-98TurboはDMAを使用することができません。この場合、IBCONFユーティリティの中でDMAチャンネルのパラメータをNONEに設定します。

コンピュータをDMAコントローラとして使用したくなければ、GPIB-98TurboをDMAラインから論理的に切り離すことができます。これは、コンフィギュレーションユーティリティIBCONFを走らせ、GPIB-98TurboのDMAチャンネルをNONEに設定します。

DMAチャンネルがぶつかっているときの現象は様々です。極端な例では、ぶつかりによってコンピュータが立ち上がらなくなります。DMAチャンネルのぶつかり合いは、時々まったく簡単に見つけることができます。それはデータ転送動作(ibrd,ibwrt)がタイムアウトで終了し、1バイトも転送されません。

ナショナルインスツルメンツは、動作するようなデフォルトDMAチャンネルを選択するようにしています。しかし、PC9801コンピュータで使える多くの異なるインターフェースボードがあるため、すべてのシステムで動作を保証するDMAチャンネルを選択することは不可能です。したがって、インストレーションを進める前にシステム内のDMAチャンネルの割り付けを確かめておいて下さい。

付録C
仕様

ＩＥＥＥ－４８８バス

Turbo488クロック	20.0 MHz
GPIBインターフェース コントローラクロック	20.0 MHz
転送レート	
GPIBデバイスからのリード	最大１Mバイト／秒
GPIBデバイスへのライト	最大１Mバイト／秒
GPIBコマンド	350 Kバイト／秒

（実際のスピードは計測器の能力によるため、上記のスピードとはかなり異なる場合があります。）

消費電力

+5 VDC	0.33 A (typical) 1.0 A (max)

物理特性

寸法	17 cm x 14.8 cm
I/Oコネクタ	IEEE-488標準24ピン

動作環境範囲

温度範囲	0℃～70℃
湿度	5％～90％（結露なし）

保存環境範囲

温度範囲	-55℃～150℃
湿度	5％～90％（結露なし）

付録D
カスタマコミュニケーション

本章には、ユーザの皆さんの便宜のために、技術的サポートに必要な情報を記入する書式だけでなく、製品やマニュアル類についてのご意見をお寄せいただくための書式も取り入れました。まずテクニカルサポートフォーム(サポート用紙)にご記入になってからナショナルインスツルメンツにご連絡ください。素早く、確実に問題を解決できるようになります。

ナショナルインスツルメンツは、広範な技術支援を世界中のユーザの皆さんに提供しています。米国およびカナダでは、アプリケーションエンジニアが月曜から金曜の9:00～17:00まで待機しております。これ以外の国では、お近くの支社にご連絡ください。ご相談のファックスは24時間いつご送信くださっても結構です。

日本ナショナルインスツルメンツ(株)
TEL: (03) 3788-1921
FAX: (03) 3788-1923

テクニカルサポートFAXフォーム

問題点またはご質問はできるだけ詳しくお書き下さい。

フリガナ
氏名 _____

会社名 _____ 所属 _____

御住所 _____

_____

電話番号 ___ (_____) _____ FAX番号 ___ (_____) _____

コンピュータの機種 _____

オペレーティングシステム _____

CPUクロック _____ MHz ___ RAM _____ Mバイト ___

マウス _____ 有 _____ 無 _____

他に使用しているボード _____

使用している計測器 _____

ナショナルインスツルメンツ ハードウェア製品 _____

Rev No. _____

ナショナルインスツルメンツ ソフトウェア製品 _____

Rev No. _____

ご質問、問題点、ご意見など _____

_____

_____

_____

_____

GPIB98Turboハードウェア／ソフトウェアコンフィギュレーションフォーム

これは、GPIB-98Turboのハードウェアとソフトウェアをインストールして設定するための必要な手順を完了しても、なおうまく動作しない場合に使用するフォームです。下記に必要事項をご記入の上ナショナルインスツルメンツにご連絡下さい。

ご連絡いただく前に、このフォームにご記入いただくことにより迅速な対応が行えます。このフォームの各情報は、問題点を解決する場合に必要となります。

フォームにご記入いただく場合、各項目の右側に必要な情報を書き留めておいて下さい。正確にご記入いただくことにより、ご質問に対しより正確なご回答が行えます。

ナショナルインスツルメンツ製品

- NI-488.2ソフトウェアのレビジョン(Rev)番号：_____

  ディスクラベル：NI-488.2 Distribution Disk for GPIB-98Turbo
  MS- DOS Handler, N88BASIC, QuickBASIC, C
  &Universal Interfaces

- 言語インターフェースのレビジョン番号：_____
  （これはナショナルインスツルメンツ製品の番号で、
  N88BASIC, QuickBASIC, C以外は、ディスク上のレビジョン番号をご覧下さい。）

- GPIB-98Turboのインタラプトレベル：_____
  （コンフィギュレーションユーティリティIBCONFで確認できます）

- GPIB-98TurboのDMAチャンネル：_____
  (コンフィギュレーションユーティリティIBCONFで確認できます)

- GPIB-98TurboのベースI/Oアドレス：_____
  (これはボード上のディップスイッチによって決定され、コンフィギュレーションユー　ティリティIBCONFで確認できます。ディップスイッチの設定とIBCONFプログラムでの　アドレス設定とは一致していなければなりません。)

ドキュメンテーションコメントフォーム

製品に付属のドキュメントに関するご意見をお寄せ下さい。ご意見をもとに今後の製品向上に努めて参ります。

タイトル： Getting Started with Your GPIB-98Turbo and the NI-488.2™ Software for MS-DOS

December 1993, P/N 370912A-01

マニュアルの編成などについて .....

マニュアルに誤りがありましたら、ページ数と内容をお書き下さい。

フリガナ
氏名 _____

会社名 _____ 所属 _____

御住所 _____

_____

電話番号 __ (____) ____ FAX番号 __ (____) ____