
ADVANTEST®
株式会社アドバンテスト

ネットワーク・アナライザ
プログラミング・ガイド

MANUAL NUMBER GJA03 9511

適用機種

R3752A/B/E R3753A/B/E
R3764A/B/C R3766A/B/C
R3765A/B/C R3767A/B/C

当社の製品が外国為替および外国貿易管理法の規定により、戦略物資あるいは役務等に該当する場合、輸出する際には日本国政府の許可が必要です。

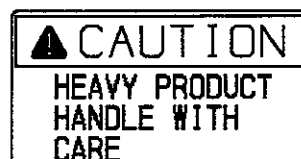
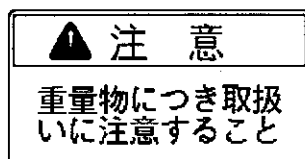
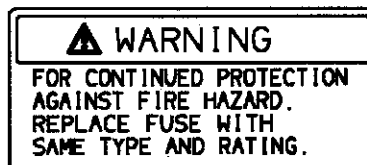
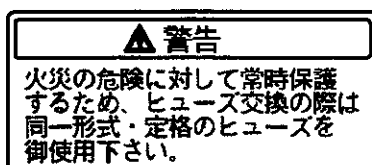
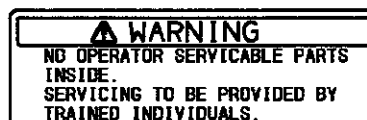
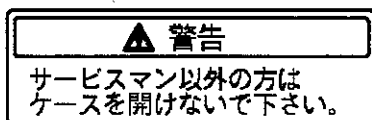
本器を安全に取り扱うための注意事項

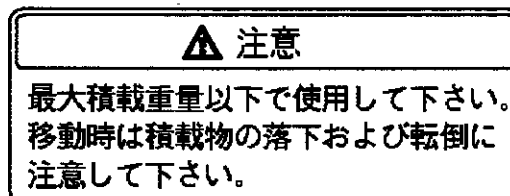
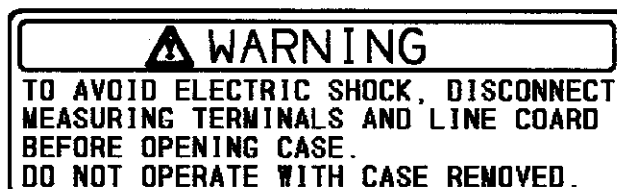
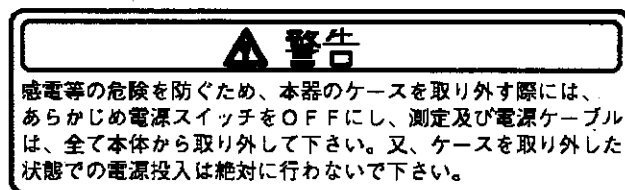
本器の機能を十分にご理解いただき、より効果的にご利用いただくために、必ずご使用前に取扱説明書をお読み下さい。また、本器の誤った使用、不適切な使用等に起因する運用結果につきましては、当社は責任を負いかねますのでご了承下さい。

本器の操作・保守等の作業を行う場合、特に人身に対する安全には細心の注意を払い、正しい方法で本器を使用し、常に安全に心がけて頂くようお願い致します。

■危険警告ラベル

アドバンテストの製品には、特有の危険が存在する場所に以下の危険警告ラベルが貼られています。取り扱いには十分注意して下さい。また、これらのラベルを破いたり、傷ついたりしないで下さい。また、日本国内で製品を購入し海外で使用する場合は、必要に応じて英語版の危険警告ラベルをお貼り下さい。危険警告ラベルについてのお問い合わせは、当社の最寄りの営業所までお願いします。所在地および電話番号は巻末に記載してあります。





■ 基本的注意事項

火災、火傷、感電、怪我などの防止のため、以下の注意事項をお守り下さい。

- 電源電圧に応じた電源ケーブルを使用して下さい。ただし、海外で使用する場合は、それぞれの国の安全規格に適合した電源ケーブルを使用して下さい。また、電源ケーブルの上には重い物をのせないで下さい。
- 電源プラグをコンセントに差し込むときは、電源スイッチをOFFにしてから奥までしっかり差し込んで下さい。
- 電源プラグをコンセントから抜くときは、電源スイッチをOFFにしてから、電源ケーブルを引っぱらずにプラグを持って抜いて下さい。このとき、濡れた手で抜かないで下さい。
- 電源投入前に、本器の電源電圧が供給電源電圧と一致していることを確認して下さい。
- 電源ケーブルは、保護接地端子を備えた電源コンセントに接続して下さい。保護接地端子を備えていない延長コードを使用すると、保護接地が無効になります。

- 3ピン-2ピン変換アダプタを使用する場合は、アダプタから出ている接地ピンをコンセントのアースに接地するか、または背面パネルにアース端子があるものは外部のアースと接続し、大地接地して下さい。また、アダプタの接地ピンの短絡に注意して下さい。
- 電源電圧に適合した規格のヒューズを使用して下さい。
- ケースを開けたままで本器を使用しないで下さい。
- 規定の周囲環境で本器を使用して下さい。
- 本器の上に重いものをのせないで下さい。また、花瓶や薬品などの液体の入った容器を本器の上またはそばに置かないで下さい。
- 本器の通風口などから、金属類や燃えやすいものなどを差し込んだり、落としたりしないで下さい。
- レーザ光を放射する製品については、レーザ光放射時、出力コネクタ端または接続されているファイバの出力端を直視しないで下さい。

■ 取扱説明書中での注意表記

取扱説明書中で使用している注意事項に関するシグナル・ワードとその定義は以下のとおりです。

| | | |
|-----------|---|-----------------------------|
| 危険 | ： | 重度の人身障害（死亡や重傷）の恐れがある注意事項 |
| 警告 | ： | 人身の安全／健康に関する注意事項 |
| 注意 | ： | 製品／設備の損傷に関する注意事項または使用上の制限事項 |

■ 製品上の安全マーク






アドバンテストの製品には、以下の安全マークが付いています。



： 取扱い注意を示しています。人体および製品を保護するため、取扱説明書を参照する必要がある場所に付いています。



： アース記号を示しています。感電防止のため機器を使用する前に、接地が必要なフィールド・ワイヤリング端子を示しています。

-  : 高電圧危険を示しています。1000V以上の電圧が入力または出力される場所に付いています。
-  : フレーム（またはケース）端子を示しています。製品の外部フレーム（またはケース）部と接続されている端子に付いています。
-  : 交流（電流または電圧）を示しています。
-  : 直流（電流または電圧）を示しています。
-  : 交流（電流または電圧）もしくは直流（電流または電圧）を示しています。

■本器の廃棄時の注意

製品を廃棄する場合は有害物質に注意し、適正に処理いただきますようお願いいたします。また、廃棄方法等についてのお問い合わせは、当社の最寄りの営業所までお願いいたします。所在地および電話番号は巻末に記載してあります。

- 有害物質 :
- (1) PCB（ポリ塩化ビフェニール）
 - (2) 水銀
 - (3) Ni-Cd（ニッケル-カドミウム）
 - (4) その他
シアン、有機リン、六価クロムを有する物およびカドミウム、鉛、砒素を溶出する恐れのある物（半田付けの鉛は除く）

■寿命部品

本器には、磨耗などによる寿命を考慮する必要のある部品が使用されている場合があります。所定の性能を維持するため、寿命部品については定期的な交換をお願いいたします。なお、寿命部品についてのお問い合わせは、当社の最寄りの営業所までお願いいたします。所在地および電話番号は巻末に記載してあります。

この取扱説明書の使い方

本書は、おもにR3752/53, R3764/66, R3765/67シリーズ（本体）でエディタを用いたBASICプログラムの作成、実行について説明しています。

【関連マニュアル】

- R3752 ネットワーク・アナライザ取扱説明書
 - R3753 ネットワーク・アナライザ取扱説明書
 - R3764/66 ネットワーク・アナライザ取扱説明書
 - R3765/67 ネットワーク・アナライザ取扱説明書
- …ネットワーク・アナライザの各部名称、機能説明、キー操作などを説明しています。
- R3752/53 プログラミング・マニュアル
 - R3764/66, R3765/67 プログラミング・マニュアル
- …内蔵BASIC、 GPIBについて説明しています。

【本書上でのキーの区別】

- 本体パネル・キー： 実線枠囲みのキーで表します。

(例)

| |
|-----|
| ENT |
|-----|

、

| |
|----|
| BS |
|----|

、

| |
|---|
| 0 |
|---|

 ~

| |
|---|
| 9 |
|---|

- 本体ソフトキー： 点線枠囲みのキーで表します。

(例)

| |
|------|
| EDIT |
|------|

- IBM-PCキーボード： 二重線枠囲みのキーで表します。

(例)

| |
|-------|
| Enter |
|-------|

、

| |
|-----------|
| Backspace |
|-----------|

、

| |
|---|
| 0 |
|---|

 ~

| |
|---|
| 9 |
|---|

【本書上でのコマンドの区別】

- キーボードから入力するコマンド
： 括弧をつけないコマンドで表します。
(例) EDITコマンド、PRINT

- エディタのメニューから選択するコマンド
： 括弧をつけたコマンドで表します。
(例) [EDIT] コマンド、[Print]

| | |
|--------------------------------------|--------|
| 目次 | |
| 1. はじめに | 1 - 1 |
| 1.1 本BASIC の特長 | 1 - 1 |
| 1.2 エディタのパッケージ | 1 - 1 |
| 1.3 必要な機器 | 1 - 1 |
| 1.4 インストール | 1 - 2 |
| 1.4.1 インストール方法 | 1 - 2 |
| 1.4.2 システムの設定 | 1 - 2 |
| 1.5 ディスク変換プログラム | 1 - 3 |
| 2. プログラミングの基礎 | 2 - 1 |
| 2.1 プログラム・モード | 2 - 1 |
| 2.1.1 プログラム・モードの起動 | 2 - 1 |
| 2.1.2 ダイレクト・モード | 2 - 2 |
| 2.2 コマンドを使用したBASIC 命令 | 2 - 4 |
| 2.2.1 PRINT 命令 | 2 - 4 |
| 2.2.2 代入命令 | 2 - 5 |
| 2.3 プログラムの作成と実行 | 2 - 6 |
| 2.3.1 プログラム入力と実行(LIST, RUN) | 2 - 6 |
| 2.3.2 プログラムの消去(SCRATCH) | 2 - 7 |
| 2.3.3 実行中のデータ入力 | 2 - 8 |
| 2.4 プログラムの保存(INITIALIZE, SAVE) | 2 - 9 |
| 2.5 プログラムの読み出し(LOAD) | 2 - 11 |
| 2.5.1 ファイル名が見たいとき(CAT) | 2 - 11 |
| 2.5.2 プリンタへの出力(GLIST, LLIST) | 2 - 11 |
| 2.6 エディット・モード | 2 - 12 |
| 2.6.1 エディタの起動 | 2 - 12 |
| 2.6.2 エディタのプログラミング環境 | 2 - 13 |
| 2.6.3 メニューのオープン | 2 - 13 |
| 2.6.4 編集コマンドの選び方 | 2 - 14 |
| 2.6.5 対話ボックスの使い方 | 2 - 15 |
| 2.6.6 編集コマンドを直接実行する方法 | 2 - 16 |
| 2.6.7 エディタの終了 | 2 - 17 |
| 2.7 プログラムの編集 | 2 - 18 |
| 2.7.1 文字の挿入 | 2 - 18 |
| 2.7.2 行の挿入 | 2 - 19 |
| 2.7.3 文字と行の削除 | 2 - 20 |
| 2.7.4 ブロック編集 | 2 - 22 |
| 3. BASIC エディタの機能 | 3 - 1 |
| 3.1 BASIC エディタの起動 | 3 - 1 |
| 3.1.1 エディタの起動 | 3 - 1 |
| 3.1.2 BASIC エディタ画面 | 3 - 1 |
| 3.2 メニューのオープンとコマンドの実行 | 3 - 3 |

| | | |
|-------|---------------------------------------|--------|
| 3.2.1 | キー操作によりコマンドを実行する | 3 - 3 |
| 3.2.2 | ショートカットキーによりコマンドを実行する | 3 - 4 |
| 3.3 | 対話ボックスの使い方 | 3 - 6 |
| 3.4 | メッセージ・ラインの使い方 | 3 - 8 |
| 3.5 | ウィンドウの使い方 | 3 - 8 |
| 3.5.1 | 各ウィンドウの働き | 3 - 8 |
| 3.5.2 | アクティブ・ウィンドウの変更 | 3 - 8 |
| 3.5.3 | ウィンドウ・サイズの変更 | 3 - 8 |
| 3.5.4 | アクティブ・ウィンドウのスクロール | 3 - 9 |
| 4. | F1:File メニュー (ファイル・メニュー) | 4 - 1 |
| 4.1 | [New] コマンド | 4 - 2 |
| 4.2 | [Open...] コマンド | 4 - 3 |
| 4.2.1 | ファイルの指定 | 4 - 3 |
| 4.2.2 | ディレクトリ内容の一覧表示 | 4 - 4 |
| 4.3 | [Insert...] コマンド | 4 - 6 |
| 4.4 | [Save] コマンド | 4 - 7 |
| 4.5 | [Save as...] コマンド | 4 - 7 |
| 4.6 | [New subfile] コマンド | 4 - 8 |
| 4.7 | [Open subfile] コマンド | 4 - 9 |
| 4.8 | [Close subfile] コマンド | 4 - 10 |
| 4.9 | [Print with SIO] コマンド | 4 - 10 |
| 4.10 | [Print with GPIB] コマンド | 4 - 10 |
| 4.11 | [Save and exit] コマンド | 4 - 10 |
| 4.12 | [Quit] コマンド | 4 - 11 |
| 5. | エディタの基本操作 | 5 - 1 |
| 5.1 | テキストの入力 | 5 - 1 |
| 5.2 | テキストの選択 | 5 - 1 |
| 5.3 | テキストのインデント | 5 - 1 |
| 5.4 | 編集コマンドの概要 | 5 - 2 |
| 6. | F2>Edit メニュー (編集メニュー) | 6 - 1 |
| 6.1 | クリップ・ボードについて | 6 - 2 |
| 6.2 | [Cut] コマンド | 6 - 2 |
| 6.3 | [Copy] コマンド | 6 - 2 |
| 6.4 | [Paste] コマンド | 6 - 3 |
| 6.5 | [Upper case] コマンド | 6 - 3 |
| 6.6 | [Lower case] コマンド | 6 - 3 |
| 7. | F3:View メニュー (ビュー・メニュー) | 7 - 1 |
| 7.1 | [Buffer list...] コマンド | 7 - 1 |
| 7.2 | [Next buffer] コマンド | 7 - 3 |
| 7.3 | [Split window] コマンド | 7 - 3 |
| 7.4 | [Execution display] コマンド | 7 - 4 |

| | | |
|--------|---------------------------|---------|
| 8. | F4:Search メニュー (検索メニュー) | 8 - 1 |
| 8.1 | [Find...] コマンド | 8 - 1 |
| 8.2 | [Find word] コマンド | 8 - 3 |
| 8.3 | [Find again] コマンド | 8 - 3 |
| 8.4 | [Replace...] コマンド | 8 - 4 |
| 8.5 | [Find label...] コマンド | 8 - 6 |
| 9. | F5:Runメニュー (実行メニュー) | 9 - 1 |
| 9.1 | [Start] コマンド | 9 - 2 |
| 9.2 | [Initialize] コマンド | 9 - 2 |
| 9.3 | [Continue] コマンド | 9 - 2 |
| 9.4 | [Upload] コマンド | 9 - 2 |
| 9.5 | [Download] コマンド | 9 - 2 |
| 10. | ネットワーク・アナライザでの自動測定 | 10 - 1 |
| 10.1 | OUTPUTとENTER 命令を使用したプログラム | 10 - 1 |
| 10.1.1 | プログラムの実行 | 10 - 1 |
| 10.1.2 | USING を使用したプログラム | 10 - 5 |
| 10.2 | ビルトイン関数 | 10 - 7 |
| 10.2.1 | ビルトイン関数の使用 | 10 - 7 |
| 10.2.2 | ビルトイン関数を使用したプログラム | 10 - 8 |
| 10.2.3 | 測定値を判定するプログラム | 10 - 9 |
| 10.2.4 | パラレルI/O ポートに出力する | 10 - 13 |
| 11. | 波形解析プログラム例 | 11 - 1 |
| 11.1 | MAX およびMIN レベル自動測定プログラム | 11 - 1 |
| 11.2 | セラミック・フィルタ自動測定プログラム | 11 - 4 |
| 11.3 | リップル解析プログラム | 11 - 7 |
| 11.4 | バンドパス・フィルタの測定例 | 11 - 11 |
| 11.5 | クリスタル共振点の測定例 | 11 - 13 |
| 11.6 | ビルトイン関数の使い方 | 11 - 15 |
| 11.6.1 | 基本関数 | 11 - 15 |
| 11.6.2 | 最大および最小値解析関数の使用例 | 11 - 16 |
| 11.6.3 | 減衰レベル解析関数の使用例 | 11 - 17 |
| 11.6.4 | リップル解析関数の使用例(1) | 11 - 21 |
| 11.6.5 | リップル解析関数の使用例(2) | 11 - 27 |
| 11.6.6 | ダイレクト・サーチ関数の使用例 | 11 - 30 |
| 11.6.7 | データ転送 | 11 - 32 |
| 11.7 | リミット・ラインの設定 | 11 - 33 |
| 11.8 | 全 Sパラメータの 4画面表示 | 11 - 35 |
| 12. | 外部コントローラの使用例 | 12 - 1 |
| 12.1 | プログラミングの前に | 12 - 1 |

| | |
|---|---------|
| 12.1.1 GPIBのモード | 12 - 2 |
| 12.1.2 本体との接続 | 12 - 2 |
| 12.1.3 GPIBアドレスの設定 | 12 - 2 |
| 12.2 プログラムの書き方 | 12 - 4 |
| 12.2.1 N88-BASICでのプログラムの書き方 | 12 - 4 |
| 12.2.2 HP-BASICでのプログラムの書き方 | 12 - 6 |
| 12.2.3 QuickBASICでのプログラムの書き方 | 12 - 8 |
| 12.2.4 Cでのプログラムの書き方 | 12 - 14 |
| 12.3 外部コントローラによるリモート・コントロール | 12 - 18 |
| 12.3.1 通常のGPIBコマンドの転送 | 12 - 18 |
| 12.3.2 "@"付き内蔵BASICコマンドの転送 | 12 - 19 |
| 12.4 掃引終了の検出 | 12 - 21 |
| 12.4.1 N88-BASICで掃引終了を検出する | 12 - 21 |
| 12.4.2 HP-BASICで掃引終了を検出する | 12 - 22 |
| 12.4.3 QuickBASICで掃引終了を検出する | 12 - 22 |
| 12.4.4 Cで掃引終了を検出する | 12 - 24 |
| 12.5 トレース・データの転送 | 12 - 26 |
| 12.5.1 本体からPC-9801へのトレース・データ転送 | 12 - 26 |
| 12.5.2 PC-9801から本体へのトレース・データ出力 | 12 - 30 |
| 12.5.3 本体からHP-BASICへのトレース・データ転送 | 12 - 33 |
| 12.5.4 本体からQuickBASICへのトレース・データ転送 | 12 - 35 |
| 12.5.5 本体からCへのトレース・データ転送 | 12 - 39 |
| 12.6 内蔵BASICと外部コントローラを同時に使うために | 12 - 47 |
| 12.6.1 内蔵BASICでの送信プログラム | 12 - 47 |
| 12.6.2 N88-BASICでの受信プログラム | 12 - 49 |
| 12.6.3 HP-BASICでの受信プログラム | 12 - 51 |
| 12.6.4 QuickBASICでの受信プログラム | 12 - 52 |
| 12.6.5 ANSI-Cでの受信プログラム | 12 - 56 |
| 12.7 BASICプログラムのダウン・ロード | 12 - 61 |
| 12.7.1 N88-BASICでのダウン・ロード・プログラム | 12 - 62 |
| 12.7.2 HP-BASICでのダウン・ロード・プログラム | 12 - 64 |
| 12.7.3 QuickBASICでのダウン・ロード・プログラム | 12 - 65 |
| 12.7.4 Cでのダウン・ロード・プログラム | 12 - 68 |
| 12.8 BASICプログラムのアップ・ロード | 12 - 71 |
| 12.9 補正データの転送 | 12 - 73 |
| 12.9.1 本体とPC-9801との間の補正データ転送(N88-BASIC) | 12 - 73 |
| 12.9.2 本体とPC/ATとの間の補正データ転送(C言語) | 12 - 76 |

| | |
|----------|-------|
| 索引 | I - 1 |
|----------|-------|

図一覽

| 図番号 | 名 称 | ページ |
|---------|-------------------------------|---------|
| 2 - 1 | プログラム (ダイレクト) ・モードの画面 | 2 - 1 |
| 2 - 2 | エディット・モードの画面 | 2 - 13 |
| 2 - 3 | F1:File メニュー | 2 - 14 |
| 2 - 4 | [Open...] コマンドの対話ボックス | 2 - 16 |
| 3 - 1 | エディタ画面の構成要素 | 3 - 2 |
| 3 - 2 | F1:File メニュー | 3 - 3 |
| 3 - 3 | [Open...] コマンドの対話ボックス | 3 - 6 |
| 3 - 4 | [Replace...] コマンドの対話ボックス | 3 - 6 |
| 4 - 1 | [Open...] コマンドの対話ボックス | 4 - 3 |
| 4 - 2 | [Insert...] コマンドの対話ボックス | 4 - 6 |
| 4 - 3 | [Save as...] コマンドの対話ボックス | 4 - 7 |
| 4 - 4 | [New subfile] コマンドの対話ボックス | 4 - 8 |
| 4 - 5 | [Open subfile] コマンドの対話ボックス | 4 - 9 |
| 7 - 1 | [Buffer list...] コマンドの対話ボックス | 7 - 2 |
| 8 - 1 | [Find...] コマンドの対話ボックス | 8 - 2 |
| 8 - 2 | [Replace...] コマンドの対話ボックス | 8 - 4 |
| 8 - 3 | [Find label...] コマンドの対話ボックス | 8 - 6 |
| 10 - 1 | 実行結果の画面表示 | 10 - 4 |
| 10 - 2 | 判定プログラムの実行結果 | 10 - 13 |
| 11 - 1 | 実行結果の画面表示 (バンドパス・フィルタの測定) | 11 - 12 |
| 11 - 2 | 実行結果の画面表示 (クリスタル共振点の測定) | 11 - 14 |
| 11 - 3 | 最大および最小値解析関数 | 11 - 16 |
| 11 - 4 | 減衰レベルの解析 | 11 - 18 |
| 11 - 5 | MBNDI | 11 - 19 |
| 11 - 6 | MBNDO | 11 - 20 |
| 11 - 7 | RPL1 | 11 - 22 |
| 11 - 8 | RPL2 | 11 - 23 |
| 11 - 9 | RPL4 | 11 - 23 |
| 11 - 10 | RPL3 | 11 - 24 |
| 11 - 11 | 極大値の最大値と最小値 | 11 - 25 |
| 11 - 12 | リップルのレスポンスおよび周波数 | 11 - 26 |
| 11 - 13 | 特定リップルの解析 | 11 - 28 |
| 11 - 14 | 全リップルの解析 | 11 - 29 |
| 11 - 15 | ダイレクト・サーチ | 11 - 30 |
| 11 - 16 | レスポンスに対応する帯域幅 | 11 - 31 |
| 11 - 17 | ゼロ位相のサーチ | 11 - 32 |

表一覽

| 表番号 | 名 称 | ページ |
|--------|---------------------|--------|
| 3 - 1 | ショートカットキー操作 | 3 - 4 |
| 5 - 1 | 編集コマンド一覽 | 5 - 2 |
| 12 - 1 | GPIBケーブル (別売) | 12 - 2 |
| 12 - 2 | 主な外部コントローラの特長 | 12 - 4 |

例一覧

| 例番号 | 名 称 | ページ |
|---------|--|---------|
| 10 - 1 | OUTPUTとENTER 命令を使用したプログラム | 10 - 1 |
| 10 - 2 | OUTPUTとENTER 命令を使用したプログラム (ビルトイン関数使用) | 10 - 2 |
| 10 - 3 | USING を使用した表示プログラム | 10 - 5 |
| 10 - 4 | CVALUB関数を使用したプログラム | 10 - 7 |
| 10 - 5 | ビルトイン関数を使用したプログラム | 10 - 8 |
| 10 - 6 | 測定値を判定するプログラム | 10 - 11 |
| 10 - 7 | 測定値を判定するプログラム (パラレルI/O ポート使用) | 10 - 15 |
| 11 - 1 | MAX およびMIN レベルの自動測定プログラム | 11 - 1 |
| 11 - 2 | セラミック・フィルタ自動測定プログラム | 11 - 4 |
| 11 - 3 | リップル解析プログラム | 11 - 7 |
| 11 - 4 | バンドパス・フィルタの測定 | 11 - 11 |
| 11 - 5 | クリスタル共振点の測定 | 11 - 13 |
| 11 - 6 | リミット・ラインの設定 | 11 - 33 |
| 11 - 7 | 全 Sパラメータ 4画面表示 | 11 - 35 |
| 12 - 1 | PC-9801 上での GPIBコントロール・プログラム (N88-BASIC) | 12 - 4 |
| 12 - 2 | HP9000上での GPIBコントロール・プログラム (HP-BASIC) | 12 - 6 |
| 12 - 3 | PC/AT 上での GPIBコントロール・プログラム (QuickBASIC) | 12 - 8 |
| 12 - 4 | PC/AT 上での GPIBコントロール・プログラム (ANSI-C) | 12 - 14 |
| 12 - 5 | N88-BASIC での GPIBコマンド転送 | 12 - 18 |
| 12 - 6 | HP-BASICでの GPIBコマンド転送 | 12 - 18 |
| 12 - 7 | QuickBASICでの GPIBコマンド転送 | 12 - 18 |
| 12 - 8 | C での GPIBコマンド転送 | 12 - 19 |
| 12 - 9 | N88-BASIC での BASICコマンド転送 | 12 - 19 |
| 12 - 10 | HP-BASICでの BASICコマンド転送 | 12 - 19 |
| 12 - 11 | QuickBASICでの BASICコマンド転送 | 12 - 20 |
| 12 - 12 | C での BASICコマンド転送 | 12 - 20 |
| 12 - 13 | N88-BASIC での 掃引終了の検出 | 12 - 21 |
| 12 - 14 | HP-BASICでの 掃引終了の検出 | 12 - 22 |
| 12 - 15 | QuickBASICでの 掃引終了の検出 | 12 - 23 |
| 12 - 16 | MicrosoftCでの 掃引終了の検出 | 12 - 24 |
| 12 - 17 | PC-9801 での データ入力 (ASCIIフォーマット) | 12 - 26 |
| 12 - 18 | PC-9801 での データ入力 (バイナリ・フォーマット) | 12 - 28 |
| 12 - 19 | PC-9801 での データ出力 (ASCIIフォーマット) | 12 - 30 |
| 12 - 20 | PC-9801 での データ出力 (バイナリ・フォーマット) | 12 - 31 |
| 12 - 21 | HP-BASICでの データ入力 (ASCIIフォーマット) | 12 - 33 |
| 12 - 22 | HP-BASICでの データ入力 (バイナリ・フォーマット) | 12 - 34 |
| 12 - 23 | QuickBASICでの データ入力 (ASCIIフォーマット) | 12 - 35 |
| 12 - 24 | C での データ入力 (ASCIIフォーマット) | 12 - 39 |
| 12 - 25 | C での データ入力 (バイナリ・フォーマット) | 12 - 43 |
| 12 - 26 | 内蔵 BASICの送信プログラム | 12 - 47 |
| 12 - 27 | N88-BASIC での受信プログラム | 12 - 49 |
| 12 - 28 | HP-BASICでの受信プログラム | 12 - 51 |
| 12 - 29 | QuickBASICでの受信プログラム | 12 - 52 |
| 12 - 30 | C での受信プログラム | 12 - 56 |

ネットワーク・アナライザ
プログラミング・ガイド

例一覽

| 例番号 | 名 称 | ページ |
|---------|--|---------|
| 12 - 31 | N88-BASIC 用ダウン・ロード・プログラム | 12 - 62 |
| 12 - 32 | HP-BASIC用ダウン・ロード・プログラム | 12 - 64 |
| 12 - 33 | QuickBASIC用ダウン・ロード・プログラム | 12 - 65 |
| 12 - 34 | C 用ダウン・ロード・プログラム | 12 - 68 |
| 12 - 35 | N88-BASIC 用アップ・ロード・プログラム | 12 - 72 |
| 12 - 36 | 本体とPC-9801 との間の補正データ転送 (バイナリ・フォーマット) | 12 - 73 |
| 12 - 37 | 本体とPC/AT との間の補正データ転送 (バイナリ・フォーマット) | 12 - 76 |

1. はじめに

R3752/53シリーズおよびR3764/66, R3765/67 シリーズ (本体) には BASICプログラムの実行環境が添え付けられていて、外部のパソコン等で作成した BASICプログラムを実行できます。また、本書で説明するエディタを使用してプログラム開発をすることができます。

(注) 工場出荷時には、エディタはインストールされていませんので、最初にインストールしてから使用して下さい。

1.1 本BASIC の特長

本BASIC は、行番号なしの BASICプログラムをロードして実行することができます。行番号なしのプログラム・ファイルをロードすると、自動的に最初の行から番号をつけます。行番号が必要ないので、プログラムの編集が容易にできます。

本体で BASICプログラムを作成する場合、エディタをインストールします。このエディタを使うと、エディタの中でプログラム・ファイルをロードしたり、実行したりすることができます。エディタの中でプログラムを実行した場合、実行時に起こるエラー (ランタイム・エラー) をトレースします。エラーが発生すると、エラーが起きた行にカーソルを自動的に移動します。

エディタが備えているコマンドのほとんどは、メニューから実行できるので、マニュアルを見ずに、コマンドの実行が簡単にできます。

1.2 エディタのパッケージ

エディタのパッケージには、以下のものが含まれます。

- エディタのインストール・ディスク : 1枚
(PR37670001-FK, HD 1.44Mバイト)
- サンプル・プログラム・ディスク : 1枚
(PR37670003-FJ, DD 720k バイト)
- プログラミング・ガイド : 本書

1.3 必要な機器

BASIC とエディタを使用するには、以下に示すものを用意して下さい。

- ネットワーク・アナライザ R3764/66シリーズまたはR3765/67シリーズ
もしくは、システム・ソフトウェアのバージョンが B00以降のネットワーク・アナライザ R3752/53シリーズ
- IBM-PC互換キーボード 101 または106
- 外部CRT ディスプレイ (R3752, R3764/66)

1.4 インストール

エディタを使用する前には、必ずエディタをインストールして下さい。

1.4.1 インストール方法

エディタのインストールは、付属のインストール・ディスクを本体に挿入し再起動する、という簡単な操作です。

●インストールする

- ① 本体の電源を切ります。
- ② エディタのインストール・ディスクを本体のドライブへ挿入します。
- ③ 本体の電源を入れます。

インストールが開始されると、画面にメッセージ "INSTALLING EDITOR..." が表示されます。

- ④ 画面にメッセージ "COMPLETE" が表示されるまで待ちます。
- ⑤ エディタのインストール・ディスクを本体のドライブから抜き取ります。
- ⑥ 一度本体の電源を切り、再度起動します。

これで、本体のメモリ・ディスク (C:BIN) にエディタがインストールできました。

(注) インストールしたエディタのファイル (C:BIN/EDIT, X) のコピーまたは修正を行った場合、エディタの動作は保障できません。
レジスタを多く使用した状態でインストールすると、インストールできない場合があります。そのときは、レジスタを削除してからインストールして下さい。

1.4.2 システムの設定

●101 キーボードを使う

システムの設定が必要です。
本体のサービス・メニューで、キーボード・タイプを 101 にセットして下さい。
そのまま使うと、キー・コードが化ける場合があります。

●106 キーボードを使う

システムの設定は必要ありません。

1.5 ディスク変換プログラム

当社製の旧ネットワーク・アナライザ(R4611/R3751シリーズ/R3762シリーズ)で作成したプログラム(フロッピー・ディスク)は、そのまま使用できません。別売のディスク変換プログラム(PR37670002-FK)を用いて、MS-DOSディスク・フォーマットに変換して下さい。

このプログラムは、すべて正面パネルからの操作となります。

注意

変換元のディスクは、書き込み禁止モードにしておいて下さい。
間違えて、MS-DOSディスク・フォーマットに初期化しないためです。

● ディスク・フォーマットを変換する

- ① 本体(R3764/66, R3765/67シリーズ)の電源を切ります。
- ② ディスク変換プログラム・ディスクを本体のドライブへ挿入します。
- ③ 本体の電源を入れます。
- ④ 画面に下の表示が現れたら、フロッピー・ディスクを本体のドライブから抜き取ります。

```
DISK CONVERT (to MS-DOS)
-----
INSERT OLD-TYPE DISK
  [1]: OK
  [0]: CANCEL
 [BS]: EXIT

Ready?
```

- ⑤ フォーマット変換するフロッピー・ディスクを本体のドライブに挿入します。
- ⑥ 正面パネルのテンキー を押します。画面が下のようになります。

```
DISK CONVERT (to MS-DOS)
-----
reading.....
```

フロッピー・ディスクを正常に読み終わると、画面が下のようになります。

```
DISK CONVERT (to MS-DOS)
-----
INSERT NEW DISK (2DD)
  [1]: OK
  [0]: CANCEL
  [BS]: EXIT

Ready?
```

(注) 不正なフロッピー・ディスクを入れると、エラー・メッセージが表示されます。この場合、テンキー（どれでも構いません）を押して下さい。画面は最初の表示に戻るため、別のフロッピー・ディスクを挿入して再度⑥を実行して下さい。

⑦ フロッピー・ディスクを本体のドライブから抜き取ります。

⑧ 新しい 2DDのフロッピー・ディスクを本体に挿入します。

⑨ ● フォーマット済みのフロッピー・ディスクを挿入した場合

正面パネルのテンキー を押すと、旧ネットワーク・アナライザのディスク内容を、新しいディスクに変換してコピーします。ディスクのコピーが終了すると、最初の表示になります。

● フォーマットされていないフロッピー・ディスクを挿入した場合

正面パネルのテンキー を押すと、画面が下のようになります。

```
DISK CONVERT (to MS-DOS)
-----
FORMAT DISK
  [1]: OK
  [0]: CANCEL
  [BS]: EXIT

Ready?
```

正面パネルのテンキー を押すと、フロッピー・ディスクをフォーマット後、旧ネットワーク・アナライザのディスク内容を新しいディスクに変換してコピーします。ディスクのコピーが終了すると、最初の表示になります。

⑩ 続けて別のディスクを変換する場合は、同様に繰り返して下さい。

ディスク変換を終える場合は、正面パネルの を押します。

2. プログラミングの基礎

2.1 プログラム・モード

2.1.1 プログラム・モードの起動

(1) プログラム・モードを起動する

ネットワーク・アナライザに接続したキーボードを使用して BASICのプログラムを作るためには、測定モードからプログラム・モードに入らなければなりません。

● R3764/66シリーズの場合

R3764/66では、電源投入時に自動的にプログラム・モードが起動するため、特別な操作は必要ありません。

● R3765/67シリーズの場合

R3765/67では、電源投入後の状態は、フィルタなどの周波数を計る測定モードになっているので、以下の操作が必要です。

- ① プログラム・モードにするには、正面パネル・キーの **RUN** を押します。

プログラム・メニューが表示され、画面上にカーソルが現れます。
([図2-1]参照)

- ② プログラム・モードを解除するには、**CH1** および **CH2** 以外のいずれかの正面パネル・キーを押します。
プログラム・モードが解除されると、画面からカーソルが消えます。

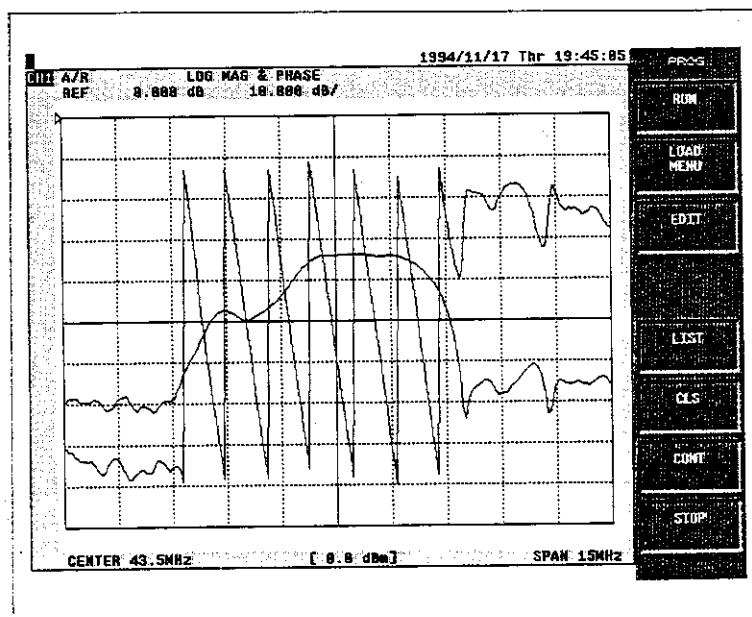


図 2 - 1 プログラム (ダイレクト) ・モードの画面

(2) プログラム・モード状態

プログラム・モードに入ると、外部キーボードから BASICコマンドやプログラムが入力できるようになります。

プログラム・モードには、「ダイレクト・モード」と「エディット・モード」があり、最初はダイレクト・モードに設定されています。

ダイレクト・モードでは、入力した文字列が直接 BASICへ送られて実行されます。

([2.1.2項] 参照)

エディット・モードでは、入力した文字列は実行されません。([2.6節] 参照)

プログラム・モード状態でも測定は実行され、ダイレクト・モードでは測定波形も表示されたままとなります。入力したテキストや BASICの PRINT構文などを使って出力した文字列は測定波形に重なって表示されます。

これらの文字列が表示される部分を「テキスト画面」といい、測定波形などが表示される部分を「測定画面」といいます。この2つの画面は合成されて表示されます。このため、表示する位置や測定波形によっては、テキスト画面上の文字が見づらくなる場合があります。この場合、テキスト画面を見易くすることができます (3)参照)。

(3) テキスト画面を見易くする方法

以下の2通りあります。

- 測定画面をDISP文で禁止し、テキスト画面だけにします(波形データは見えなくなる)。
- 画面全体を上下2つに分けて表示します(上に測定画面、下にテキスト画面)。

① 正面パネルの DISP を押します。

② ディスプレイ・メニューをDUAL CH OFF および SPLIT CH ONに設定します。

③ プログラム・モードで CONSOLE文を使ってスクロール領域を画面の下半分に設定します。

(注) この方法では、アクティブ・チャンネルのみの測定となります。両方のチャンネルをON(DUAL CH ON)の場合には使用できません。

2.1.2 ダイレクト・モード

ダイレクト・モードのときは、カーソル移動キーを使ってカーソルを自由に移動することはできません。キーボードから入力した文字は、カーソル位置に表示されます。1文字入力するごとに、カーソルは1つ右へ進みます。

入力した文字を訂正する場合は、Backspace を押して1つ左の文字を消します。

Enter を押すと、入力した文字列をコマンドとして実行します。

キーボードからの入力方法は、以下の 2通りあります。

- 1つの命令を直接入力して即座に実行し結果を出す「ダイレクト方式」
- 複数の命令を使って 1つの処理手順を作る「プログラム方式」

ダイレクト・モードを使うと、個々の BASICのステートメントを直接実行させ、その働きを確かめることができます。ステートメントをダイレクト・モードで実行すると、プログラム全体を実行させなくても、ステートメントの動作を確認できます。

- ダイレクト・モードでコマンドを実行する

① CLS と入力して を押すと、テキスト画面全体を消去することができます。

② 次のステートメントを正確に入力して下さい。

```
PRINT 2+5
```

③ を押し、入力したコードを実行して下さい。

PRINTステートメントを実行すると、テキスト画面に 7が表示されます。

2.2 コマンドを使用した BASIC 命令


ダイレクト・モードでは、1つの命令が入力されるとすぐに実行して結果を出します。その場かぎりの計算や、今まで入力したプログラムの一覧を表示させる命令など記憶させる必要のないものに使います。

2.2.1 PRINT 命令

PRINT コマンドは、そのあとに続く式を評価して、その結果を画面に表示する命令です。また、式のかわりに文字列を画面に表示することもできます。

- PRINT の後に続く式を計算して、その結果を画面に表示する

BASIC での四則演算（加算、減算、乗算、除算）は、それぞれプラス記号(+)、マイナス記号(-)、アスタリスク(*)、およびスラッシュ(/)で表します。べき乗は、キャレット(^)で表します。

次のステートメントをダイレクト・モードで入力して、 を押し、演算の結果を確認して下さい。


| 演算 | 結果 |
|-----------------|----------------|
| PRINT 2.0 + 3.0 | 5.0 |
| PRINT 2.0 - 3.0 | -1.0 |
| PRINT 2.0 * 3.0 | 6.0 |
| PRINT 2.0 / 3.0 | 0.666666666666 |
| PRINT 2.0 ^ 3.0 | 8.0 |

- 文字列を画面に表示する

文字列とは、ダブルクォーテーション (")で囲まれた一連の文字のことです。

- ① 次のステートメントを正確に入力して下さい。

```
PRINT "Hello, world"
```

- ②  を押して下さい。

【実行結果】 Hello, world

計算結果を一度他の変数に代入し、その変数を使用して結果を出力することもできます。

2.2.2 代入命令


代入命令は、(=) をはさんで左辺の<変数>に右辺の<式>を代入させる命令です。

<式>の内容は、数値、文字列など、どのような型でも構いません。但し、<式>の型は<変数>の型と一致しなければなりません。一致していない場合は、左辺の<変数>に合わせて代入されます。

● 計算結果の代入

- ① 次のステートメントを正確に入力して下さい。

```
A=(5+3)*2  
B=2*5+6/2
```

- ②  を押して下さい。

- ③ 次のステートメントを入力して、A の値を表示します。

```
PRINT A
```

【実行結果】 64.0

- ④ 次のステートメントを入力して、B の値を表示します。

```
PRINT B
```

【実行結果】 13.0

この例では、 $(5+3)^2$ の式と $2 \times 5 + 6 \div 2$ の式を計算して、その答えをそれぞれ A と B に代入しています。A と B の内容は、PRINT 命令を使用して表示しています。

2.3 プログラムの作成と実行

プログラム方式はすぐには命令を実行せずに、一度本体の記憶装置（メモリ）に記憶します。このため、コマンドは一度実行させると命令は消えますが、プログラムは実行終了後もメモリに残っているため何回でも実行することができます。

2.3.1 プログラム入力と実行（LIST, RUN）

プログラムの入力方法は、前述のコマンド入力のステートメントの前に行番号をつければよいだけです。

ステートメントの前に行番号のある行を「プログラム行」、行番号の指定されていない行を「コマンド行」といいます。

例) 10 PRINT 2+5 : 行番号10のプログラム行
PRINT 2+5 : コマンド行

BASIC の各プログラム行は、行番号で始まります。行番号はプログラム行をメモリに記憶する順序を示し、実行も行番号の小さい行から行われます。

● プログラム例

```
10 A=(5+3)^2
20 B=2*5+6/2
30 PRINT A
40 PRINT B
50 STOP
```

(1) プログラム・リストを表示する（LIST）

LISTコマンドは、画面のスクロール・エリアにプログラム一覧を表示する命令です。

- ① 次のコマンドを正確に入力して下さい。

LIST

- ② を押して下さい。

(2) 一部のプログラム行を表示する (LIST 開始行, 終了行)

LIST 開始行, 終了行は、指定された範囲のプログラム・リストを画面に表示する命令です。

(注) 開始行を省略すると、プログラムの最初の行から終了行まで表示します。
終了行を省略すると、開始行から最後の行まで表示します。
両方とも省略すると、最初の行から最後も行まで表示します。

- ① 次のステートメントを正確に入力して下さい。

```
LIST 10,40
```

- ② を押して下さい。

【実行結果】 行番号10から40までのプログラム行を表示します。

(3) 入力終了後、プログラムを実行する (RUN)

RUN コマンドは、入力したプログラムを実行する命令です。

- ① 次のコマンドを入力して下さい。

```
RUN
```

- ② を押して下さい。

【実行結果】 入力ミスがなければ画面に"64.0"と"13.0"が表示されます。

2.3.2 プログラムの消去 (SCRATCH)

SCRATCH コマンドは、すでに入力してあるプログラムや、変数を消去する命令です。
新しくプログラムを入力するときは、必ずこれまでのプログラムを消去して下さい。

● プログラムを消去する

- ① 次のコマンドを入力して下さい。

```
SCRATCH
```

- ② を押して下さい。

【実行結果】 テキスト画面に次のメッセージが表示されます。

```
BASIC Ready.
```

2.3.3 実行中のデータ入力

これまでのプログラムは、計算のデータをあらかじめプログラムの中に組み込んでいました。ここでは実行してから計算データを入力して結果を出力するプログラムを作ります。

● 三角形の面積を求めるプログラム例

- ① CLS 命令を入力して画面をクリアします。
- ② 次のプログラムを正しく入力して下さい。

```
10 INPUT "TEIHEN ?=", A
20 INPUT "TAKASA ?=", B
30 C=A*B/2
40 PRINT "KOTAE", C
50 STOP
```

- ③ RUN 命令を入力してプログラムを実行します。
- ④ キーボードからの入力を待つ状態になります。
"TEIHEN ?=" と表示して、底辺の値を聞いてくるので、キーボードから数値を入力します。

"TAKASA ?=" と表示して、高さの値を聞いてくるので、キーボードから数値を入力します。

面積を計算し "KOTAE" と表示して、答えの数値が出力されます。

2.4 プログラムの保存 (INITIALIZE, SAVE)

保存するプログラムは、フロッピー・ディスクに記憶して下さい。

行番号を付けてプログラム行を入力し、Enter を入力すると、メモリに記憶されますが、電源を切ると消えてしまうためです。また、SCRATCHコマンドを使ったときも同じです。

フロッピー・ディスクを用意し、INITIALIZEコマンドで初期化 (インテライズ) します。初期化とは、ネットワーク・アナライザで使えるように、必要な情報を書き込むことをいいます。INITIALIZEは、INITと省略できます。

(注) フロッピー・ディスクの2DD を 2HDの書式で初期化することはできません。逆に、2HD を 2DDの書式で初期化することもできません。

R3764/66およびR3765/67のフロッピー・ディスク構造は、MS-DOSのディスク構造と同じです。よって、MS-DOSによって初期化されたフロッピー・ディスクをそのまま使用することができます。

パソコンなどで初期化されたフロッピー・ディスクを使用する場合は、INITIALIZEコマンドを使う必要はありません。但し、720Kバイトで初期化された 2HDのフロッピー・ディスクや1.2Mバイト、1.4Mバイトで初期化された 2DDのフロッピー・ディスクは使用できません。

● フロッピー・ディスクを初期化する (INITIALIZE)

・ 2DD のフロッピー・ディスクの場合

- ① CLOSE * コマンドにより、オープンしているすべてのファイルをクローズします。
- ② フロッピー・ディスクを本体のドライブに挿入します。
- ③ 次のコマンドを入力します。

```
INITIALIZE "MYDISK" 0
```

これを実行すると、"MYDISK"というボリューム名を付けてフロッピー・ディスクが初期化されます。ボリューム名は12文字までが有効で、自由に名前を指定することができます (省略も可能)。初期化後のフロッピー・ディスク・サイズは、720Kバイトになります。

・ 2HD のフロッピー・ディスクの場合

以下のように指定します。

| 書式 | セクタ・サイズ | トラック当たりのセクタ数 | 全容量 |
|-----------------------|---------|--------------|---------|
| INITIALIZE "MYDISK" 1 | 1024バイト | 8セクタ | 1.2Mバイト |
| INITIALIZE "MYDISK" 2 | 512バイト | 18セクタ | 1.4Mバイト |
| INITIALIZE "MYDISK" 3 | 512バイト | 15セクタ | 1.2Mバイト |

● プログラムを保存する (SAVE)

プログラムをフロッピー・ディスクに保存するには、SAVEコマンドを使います。

- ① 初期化済みのフロッピー・ディスクを本体のドライブに挿入します。
- ② SAVEコマンドを入力します。

```
SAVE "A:MYFILE.BAS"
```

(注) ファイル名は、8文字までの名前と、3文字までの拡張子で指定できます。すでにセーブされているファイル名と同じファイル名でセーブすると、上書きされて古いプログラムは失われます。

セーブしても電源を切らなければ、メモリにプログラムが残っています。新しくプログラムを作成するときは、SCRATCH コマンドで消してから入力して下さい。

パソコンなどでフロッピー・ディスクをフォーマットし、サブディレクトリを作成した場合、そのサブディレクトリにセーブすることができます。サブディレクトリにセーブする場合は、ファイル名の前にディレクトリ名を指定します。

```
SAVE "A:MYSUB1/MYFILE.BAS"
```

この例では、"MYSUB1"というディレクトリの下にプログラムを保存します。なお、BASIC ではサブディレクトリを作成する機能を持っていません。

2.5 プログラムの読み出し (LOAD)

保存のプログラムをメモリに読み出す (ロード) には、LOAD コマンドを使います。

- ① フロッピー・ディスクを本体のドライブに挿入します。
- ② LOAD コマンドを入力します。

```
LOAD "A:MYFILE.BAS"
```

(注) ・ファイル名は、SAVE で付けた名前を入力して下さい。
・プログラムをロードすると、前に入力してあったプログラムは消されます。
(LOAD コマンドは、SCRATCH も一緒に実行します。)

2.5.1 ファイル名が見たいとき (CAT)

ロードするときファイル名を忘れてしまったときや、フロッピー・ディスクにどんなファイルがセーブされているか知りたいときは、CAT コマンドを使います。

- ① フロッピー・ディスクを本体のドライブに挿入します。
- ② CAT コマンドを入力します。

```
CAT
```

【実行結果】 テキスト画面のスクロール・エリアに次のように表示されます。

| NO | :File Name | Bytes | At |
|----|-------------|-------|----|
| 1 | :MYFILE.BAS | 418 | 0 |

この表示されている内容は、左から順に、登録番号、ファイル名、ファイルの文字数、ファイルの属性です。

2.5.2 プリンタへの出力 (GLIST, LLIST)

もう 1 つのプログラムの保存方法として、プリンタ装置を使用してプログラムをプリンタ用紙に印字させておくこともできます。

プリンタ出力方法は、以下の 2 通りあります。

- GLIST コマンドで、 GPIB 経由でプリンタに出力する
- LLIST コマンドで、シリアル・ポートに接続されたプリンタに出力する

2 つの命令の書き方は同じですが、GLIST を使う場合、あらかじめ GPIB のモードを SYSTEM CONTROLLER にし、プリンタの GPIB アドレスを設定しておかなくてはなりません。

●アドレス18の GPIB プリントに印字する

- ① "CONTROL 7:1"を入力し、SYSTEM CONTROLLER モードにします。
- ② "PRINTER 18"を入力し、プリンタの GPIB アドレスを設定します。
- ③ GLIST コマンドを入力します。

```
GLIST 10,100
```

【実行結果】 行番号10から100 までを印字します。

2.6 エディット・モード

プログラムの編集を行う場合は、エディット・モードを使用します。

ダイレクト・モードでプログラムの編集を行う場合、カーソルを自由に移動できません。そのため、一度入力してしまったプログラム行を修正するには、その行すべてを入力する必要があります、たいへん不便です。

エディット・モードでは、プルダウン形式のメニューを操作することにより、プログラムの編集からデバッグ、プログラムの実行まで、プログラミングに必要なすべての操作ができるようになっています。このプログラム環境を「BASIC エディタ」と呼びます。

ここでは、このエディタを使用して、簡単なプログラムの編集を行います。エディタについての詳細は、[3. BASIC エディタの機能]を参照して下さい。

2.6.1 エディタの起動

エディタを起動する前に、付属のインストール・ディスクを用いて BASIC エディタを本体にインストールして下さい（[1.4.1項参照]）。インストールしないで、エディタを起動しようとするすると、以下のメッセージが表示され実行できません。

```
Editor not installed
```

●エディット・モードにする

外部キーボードの F12 キーを押して、エディット・モードにします。

エディット・モードに入ると、画面は [図2-2] のようになります。
エディタ画面では、測定波形やテキスト画面は表示されません。

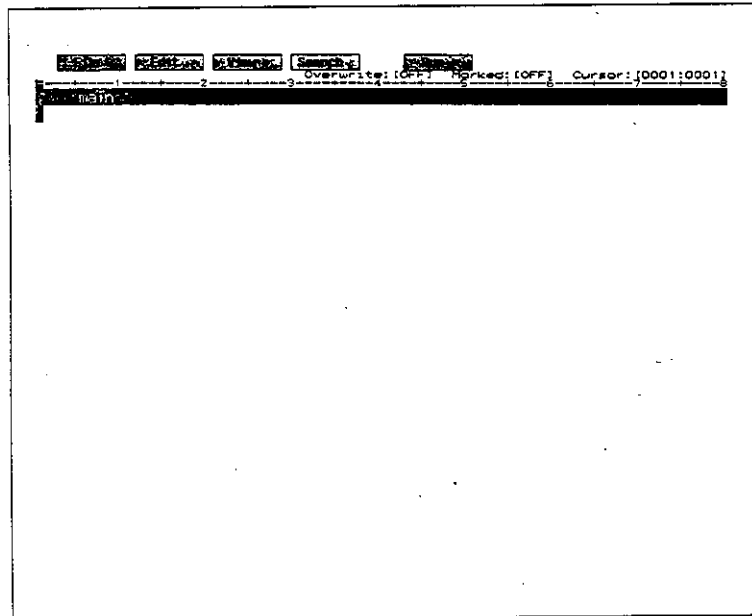


図 2 - 2 エディット・モードの画面





2.6.2 エディタのプログラミング環境

エディタには、数々のプログラミング・ツールがあります。これらのツールでは、プログラムの編集、ファイル管理、印刷などを行うことができます。

エディタ画面の一番上にあるメニュー・バーには、複数のメニューがあります。このメニューの中には、エディタのプログラミング環境を制御するコマンドがあります。これらのコマンドを使って、ビュー・ウィンドウの中でプログラムの作成や修正ができます。

2.6.3 メニューのオープン

エディタのほとんどのコマンドはメニューから選んで実行できます。メニュー・バーは左から順にキーボードのファンクション・キー (F1, F2, ...) に対応して割り付けられています。メニューをオープンするには、対応するファンクション・キーを押します。

メニューをオープンしたら、方向キー ( ,  ,  , ) を使ってメニューを選ぶことができます。

●メニューをオープンする手順

① を押して、F1:Fileメニューを選んで下さい。

, で、オープンしたメニューにあるコマンドを選ぶことができます。

, で、オープンしたメニューの左隣か右隣にあるメニューをオープンできます。

② コマンドを実行せずにメニューを閉じるには、再度 を押すか、 を押します。

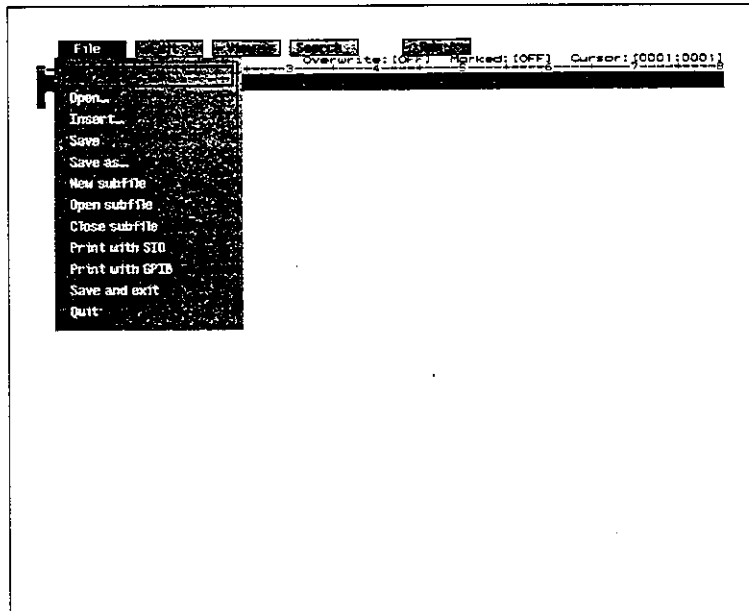


図 2 - 3 F1:Fileメニュー

2.6.4 編集コマンドの選び方

編集コマンドを選んで実行するには、メニューの中で か を使い、目的のコマンドを選び、 を押します。

コマンド名の後に省略記号(...)が続く場合、そのコマンドを実行すると [図2-4]のような対話ボックスが現れます。コマンド名の後に省略記号が付いていない場合、すぐにコマンドが実行されます。

2.6.5 対話ボックスの使い方

コマンド名の後に省略記号(...)が続くコマンドを実行すると、対話ボックスが現れます。エディタは、この対話ボックスを使ってコマンドを実行する前に必要な情報を表示します。




(1) 対話ボックスの構成要素

対話ボックスは、エディタの情報を与えるために使います。






| 構成要素 | 解説 |
|------------|--------------------------|
| テキスト・フィールド | ファイル名などのテキストを入力するために使う。 |
| リスト・ボックス | 複数の項目の中から 1つの項目を選ぶために使う。 |
| コマンド・ボタン | 使用できるキーを表示する。 |


(2) 対話ボックスの操作

はじめて対話ボックスをオープンすると、デフォルトの設定が表示されます。この後、対話ボックスをオープンすると、前に選んだ設定がそのまま表示されます。

対話ボックスにある項目間を移動するには、、 または  を使います。

● F1:File メニューの〔Open...〕コマンドの対話ボックスをオープンする

- ①  を押して、F1:File メニューをオープンします。
- ② 〔Open...〕コマンドがハイライト表示されるまで  を押して下さい。
- ③  を押して、対話ボックスをオープンします。
- ④  または  を続けて押して、“Catalog:” リスト・ボックスヘカーソルを移して下さい。

リスト・ボックスに表示されている項目が多いときは、 を押すと、リスト・ボックス内の選択項目を変えることができます。

- ⑤ **Esc** を押して、ファイルを読み込まずに、対話ボックスを閉じて下さい。

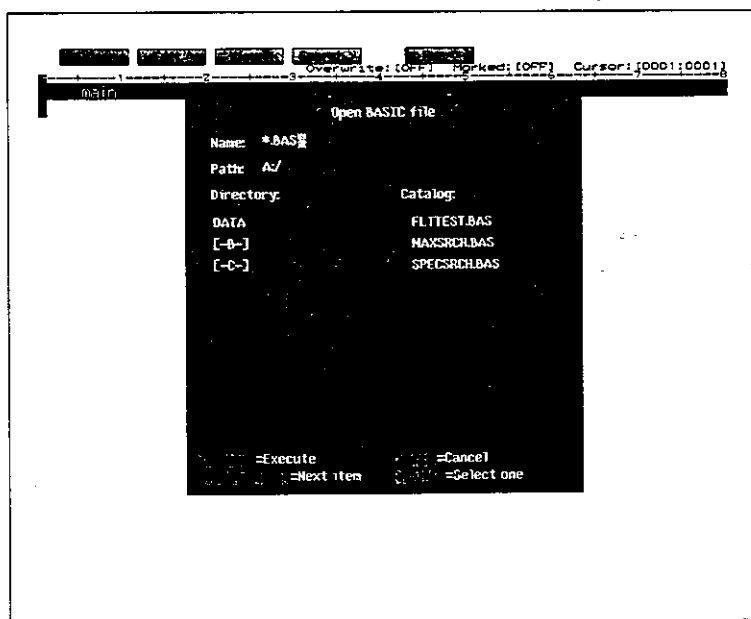


図 2 - 4 (Open...) コマンドの対話ボックス

2.6.6 編集コマンドを直接実行する方法

ショートカットキーと組み合わせて使うと、コマンドを直接実行することができます。コマンドを直接実行すると、メニューをオープンしたり、メニュー・コマンドを選ぶ必要がなくなります。

ショートカットキーは、主要なエディット・コマンドで使うことができます。

(1) コマンドを選ぶためのショートカットキー

ショートカットキーが使えるコマンドは、コマンド名の右側にショートカットキーの表示があります。

例) カット & ペーストするとき

F2:Edit メニューの [Cut] コマンドは、**Shift** を押しながら **Delete** を押すと実行できます。

[Paste] コマンドは、**Shift** を押しながら **Insert** を押すと実行できます。

(2) その他の組み合わせキー

ほとんどのキーは、、、 とファンクション・キー、
、、、、、などを組み合わせて使
います。

例) 、を押すと、画面上のカーソルを右または左に移動できますが、
カーソルが置かれている単語の左隣または右隣の単語にカーソルを移動させる
には、を押しながら または を押して下さい。

2.6.7 エディタの終了

●エディタを終了する

- ① F1:File メニューの [Quit] コマンドを実行します。
- ② 保存していないファイルがあるときに、エディタを終了しようとするすると、終了していいのか確認のメッセージが現れます。

を押すと、ファイルをセーブせずに終了し、ダイレクト・モードになります。

を押すと、元の編集画面に戻ります。ファイルをセーブしてから、再度コマンドを実行します。

2.7 プログラムの編集

ここでは、エディタを使ってプログラムを編集する方法を説明します。
プログラムの編集とは、一度書いたプログラムの修正、追加、訂正などを行うことです。

ダイレクト・モードでプログラム行を入力する場合、必ず行番号を付ける必要がありますが、エディット・モードでは行番号は必要ありません。
行番号のついていないプログラム・ファイルを指定すると、BASIC が自動的に行番号を割り付けます。

注意

これから説明するプログラム例では、行番号を除いて記述します。


2.7.1 文字の挿入

すでにプログラムした行に文字を挿入したい場合は、挿入したい場所の後の文字にカーソルを移動して、挿入したい文字をそのまま入力します。
入力を行うと、現在のカーソル位置に文字が挿入され、カーソル位置から終わりまでの文字が1文字右に移動します。

●文字を挿入する

- ① 次のプログラム行を入力して下さい。

```
PRNT "NUMBER"█
```

- ② PRNTのNへ  を押してカーソルを移動します。

```
PRNT █ "NUMBER"
```

- ③ Iを入力します。Nの位置にIが挿入されて、N以降の文字が右へずれます。


```
PRINT █ "NUMBER"
```


 Insert

を押すと、カーソル位置はそのまま空白文字が挿入されます。



カーソルの右隣の文字から行末までの文字が1文字右へ移動し、カーソル位置に空白文字が表示されます。

プログラムは、1行につき最高511文字まで有効です。
但し、80文字を越える場合は、画面には80文字までしか表示されません。

 を押して、カーソルを表示されている行の最後の文字へ移動すると、残りの文字が画面に表示されます。

行の先頭の文字を表示するには、 を押してカーソルを表示されている行の最初の文字へ移動します。

2.7.2 行の挿入



プログラムの行と行の間に新しく行を挿入したいときには、 を押しながら、 を押します。

● 行を挿入する


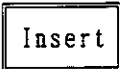
- ① 次のプログラム行を入力して下さい。

```
INPUT A  
C=A*B
```


- ② このプログラムの"INPUT A"と"C=A*B"の間に"INPUT B"という命令行を追加します。

 を押しながら  を押すと、カーソルが行頭へ移動します。

```
INPUT A  
=A*B
```

- ③ カーソルを行頭に移動したら、 を押しながら  を押します。

```
INPUT A  
  
C=A*B
```

 を押しても空白行を挿入できませんが、この場合、カーソルは次行の先頭に移動します。

行を 2 つに分ける場合、行の途中で **Ctrl** を押しながら **Insert** を押す、または **Enter** を実行すると、カーソル位置の文字から行末までの文字が次行に移動します。

2.7.3 文字と行の削除

文字を削除する場合は、**Backspace** または **Delete** を押します。

Backspace を使用すると、カーソルの 1 つ左隣の文字が削除されて、カーソルとカーソル位置から行末までの文字が左へ 1 つ移動します。

Delete を使用すると、カーソル位置の文字が削除されて、カーソルの右隣の文字から行末までの文字が左へ 1 つ移動します。

(1) Backspace を使った文字の削除

- ① 次の行を入力して下さい。

```
PRINT "A" |
```

- ② カーソル移動キーを使って、カーソルを 2 つ目の I の文字へ移動して下さい。

```
PR|INT "A"
```

- ③ **Backspace** を押して下さい。

1 つ目の I が削除されて INT "A" が左へ移動します。

```
PR|INT "A"
```

Backspace を行頭で実行すると、行を連結することができます。

カーソルとカーソル位置の行が上の行の行末へ移動します。

(2) Deleteを使つての文字の削除

- ① 次の行を入力して下さい。

```
PRIINT "A" |
```

- ② カーソル移動キーを使って、カーソルを 2つ目の Iの文字へ移動して下さい。

```
PR|IINT "A"
```

- ③ **Delete** を押して下さい。

カーソル位置のI が削除されて NT "A" が左へ移動します。

```
PR|NT "A"
```

Delete を文末で実行すると、行を連結することができます。

下の行全体がカーソル位置へ移動します。

(3) カーソル位置から行末までの削除

カーソル位置から行末までの行を削除するには、F2:Edit メニューの
{Delete line} コマンドを使います。

- ① 次のプログラム行を入力して下さい。

```
A=1+3  
B=5*4  
PRINT A |
```



- ② カーソル移動キーを使って、カーソルを B=5*4の行頭へ移動して下さい。

```
A=1+3  
|B=5*4  
PRINT A
```

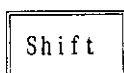
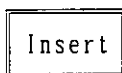
- ③ **F2** を押して、F2:Edit メニューをオープンし、{Delete line} コマンド
を実行して下さい。B=5*4 の行が削除され、下の行が上へ移動します。

```
A=1+3  
|PRINT A
```

行の途中で〔Delete line〕コマンドを実行すると、カーソル位置の文字から行末までの文字が削除されます（下の行は移動しません）。

 を押しながらか  を押すと、一回のキー操作でこのコマンドを実行できます。

もし、あやまって〔Delete line〕コマンドを実行してしまった場合、すぐに〔Paste〕コマンドを実行すると、削除した内容に戻すことができます。

 を押しながらか  を押すと、一回のキー操作でこのコマンドを実行できます。

2.7.4 ブロック編集

ブロック編集とは、編集する行をまとめて指定することをいいます。エディタでは、文字単位の編集と行単位の編集ができます。行をまとめて指定すると、広い範囲をまとめて削除したり、移動することができます。ここでは、行のまとまりを「ブロック」といいます。ブロックを選ぶには、カーソルをブロックの先頭に移動し、マーカをセットし、編集コマンドを実行します。

(1) マーカのセット

ブロック編集する行頭にマーカをセットします。

① 次のプログラム行を入力して下さい。

```
INPUT A
INPUT B
C=A+B
PRINT A
```



② カーソル移動キーを使って、カーソルを C=A+B の行頭へ移動して下さい。

```
INPUT A
INPUT B
█C=A+B
PRINT A
```

③ を押しながらか を押して下さい。

マーカがセットされると、メッセージ・ラインに Mark set と表示されます。

マーカはブロック編集以外にも、カーソル位置の記憶に使うことができます。

 を押しながらか  を押すと、マーカとカーソルを交換します。

マーク位置にカーソルを移動し、現在のカーソル位置にマーカをセットします。

(2) カットとコピー

セットしたマーカからブロックを選択して、カットやコピーができます。

(注) マーカがセットされていないと、カットもコピーもできません。
このような場合、メッセージ・ラインに No mark in this window と表示されます。

ブロック編集するために選んだ行のブロックをカットするとは、選んだ行をまとめて削除するという事です。
ブロックのコピーとは、選んだブロックを複写するという事です。
削除されたブロックやコピーされたブロックは、メモリ上のある場所に保管されます。この場所を「クリップ・ボード」といいます。

クリップ・ボードの内容は、直接見ることはできません。クリップ・ボードの内容は、ペーストすると表示されます。
クリップ・ボードの内容は、カットやコピー、行の削除をするたびに更新されます。最後にカットやコピー、行の削除した内容が保管されます。
クリップ・ボードの内容は、後述するペーストによってカーソルの位置に挿入することができます。

● [Cut] コマンドを実行する

① 次のプログラム行を入力して下さい。

```
INPUT A  
INPUT B  
C=A+B  
PRINT A
```

② C=A+B の行頭にマーカをセットした後、カーソルを行末へ移動して下さい。

③ **F2** を押して F2:Editメニューをオープンして下さい。

④ [Cut] コマンドを実行して下さい。

```
INPUT A  
INPUT B  
PRINT A
```

選んだテキストはクリップ・ボードに送られます。
一回に送ることができるテキスト・ブロックは 1ブロックだけです。

Shift を押しながら **Backspace** を押すと、一回のキー操作でこのコマンドを実行できます。

Backspace だけを押すと、カーソル位置の左隣の文字だけが削除されます。

(3) テキスト・ブロックのペースト

クリップ・ボードに送ったテキストは、新しいテキストをクリップ・ボードに送るか、エディタを終了するまで残ります。

● プログラム行をペーストする

① 次のプログラム行を入力して下さい。

```
INPUT A
INPUT B
C=A+B
PRINT A
█
```

② **Ctrl** を押しながら **Space** を押して、カーソル位置にマーカをセット

します。

③ 矢印キーを使って、C=A+Bの行頭にカーソルを移動します。

④ **Alt** を押しながら **Space** を押して、カーソルとマーカを交換します。

⑤ **F2** を押して F2:Editメニューをオープンして下さい。

⑥ [Copy] コマンドを実行して下さい。

⑦ **F2** を押して F2:Editメニューをオープンして下さい。

⑧ [Paste] コマンドを実行して下さい。

```
INPUT A
INPUT B
C=A+B
PRINT A
C=A+B
PRINT A
█
```

Shift を押しながら **Insert** を押すと、一回のキー操作でこのコマンドを実行できます。

Insert だけを押しすると、カーソル位置に空白文字だけを挿入します。

3. BASIC エディタの機能

この章では、[2.6 エディタ・モード]で説明したBASICエディタの各機能について詳しく説明します。先に述べた通り、エディタが持つプルダウン形式のメニューにより、プログラムの編集からデバッグ、プログラムの実行まで、プログラミングに必要なすべての操作ができるようになっていきます。

以下の項目を説明します。

- BASIC エディタを起動する方法
- メニューのコマンドを実行する方法
- 対話ボックスのオプションを選ぶ方法
- リスト・ボックスのスクロール
- 対話ボックスやウィンドウでのテキストの選び方
- ウィンドウ・サイズの変え方
 - ダイレクト画面の使い方
 - ウォッチング・ウィンドウの使い方

3.1 BASIC エディタの起動

エディタは、外部キーボードから起動します。（[3.1.1項]参照）

エディタを起動する前に、付属のインストール・ディスクを用いて BASICエディタを本体にインストールしなければなりません（[1.4.1項]参照）。もし、インストールしないで、エディタを起動しようとするすると、以下のメッセージが現れ、起動しません。

```
Editor not installed
```

3.1.1 エディタの起動

- エディット・モードにする

外部キーボードの

| |
|-----|
| F12 |
|-----|

 を押して下さい。

3.1.2 BASIC エディタ画面

エディタを起動すると、エディタの画面が現れます。
エディタ画面では、測定波形や BASICテキスト画面は表示されません。
[図3-1]にエディタ画面の構成要素を示します。

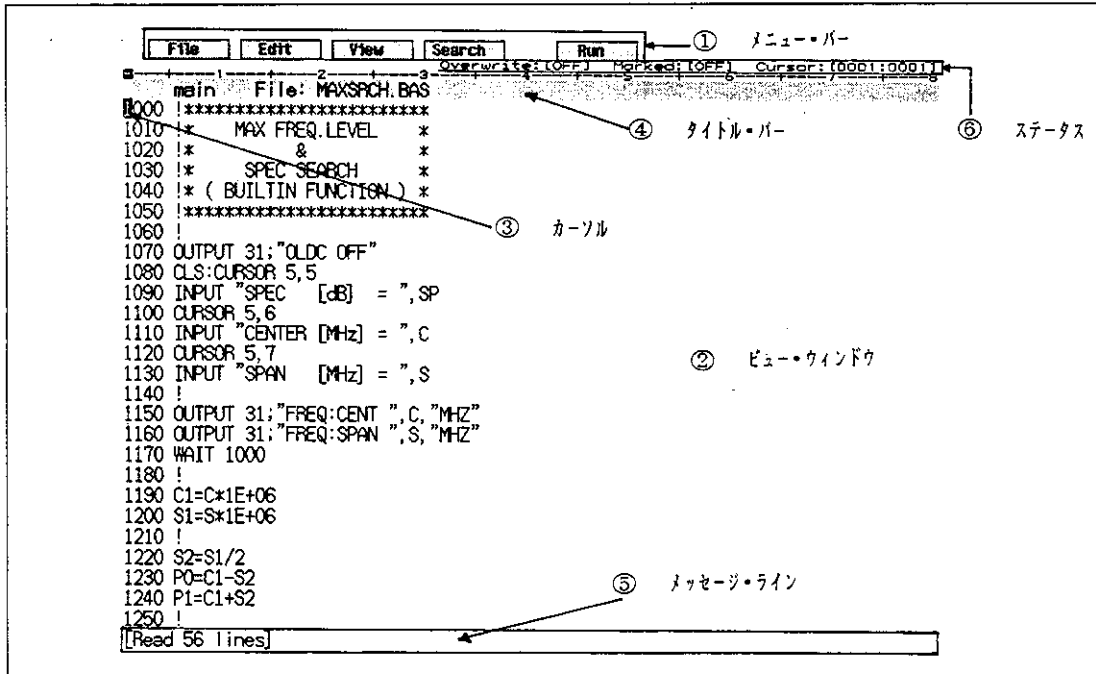


図 3 - 1 エディタ画面の構成要素

● 画面の構成要素

| 構成要素 | 解説 |
|-------------|--|
| ① メニュー・バー | 各メニューの名前を表示する。 |
| ② ビュー・ウィンドウ | プログラムのテキストを表示する。 |
| ③ カーソル | テキストを入力したり編集する位置を示す。 カーソルはアクティブなウィンドウの中に現れる。 |
| ④ タイトル・バー | プログラムやサブファイルのバッファ名・ファイル名を示す。 BASIC プログラムは、バッファ名が main で表される。 それ以外のサブ・ファイルは、ファイル名と同じ。 編集を加えたファイルは、タイトル・バーの先頭にアスタリスク(*)が表示される。 |
| ⑤ メッセージ・ライン | 編集中に起きたエラーや追加情報を表示する。 |
| ⑥ ステータス | エディタのステータス(Overwrite/Marked/Cursor)を表示する。 ● Overwrite : 挿入/上書きモードを示す。 ● Marked : テキスト・ブロック (テキストの範囲指定)の状態を示す。 ● Cursor : カーソルの画面上の位置をX/Y で示す。 |

3.2 メニューのオープンとコマンドの実行

エディタの各コマンドは、メニューバーのプルダウン・メニューの中にあります。
[図3-2]は、プルダウン・メニューの1つである F1:Fileメニューを示しています。

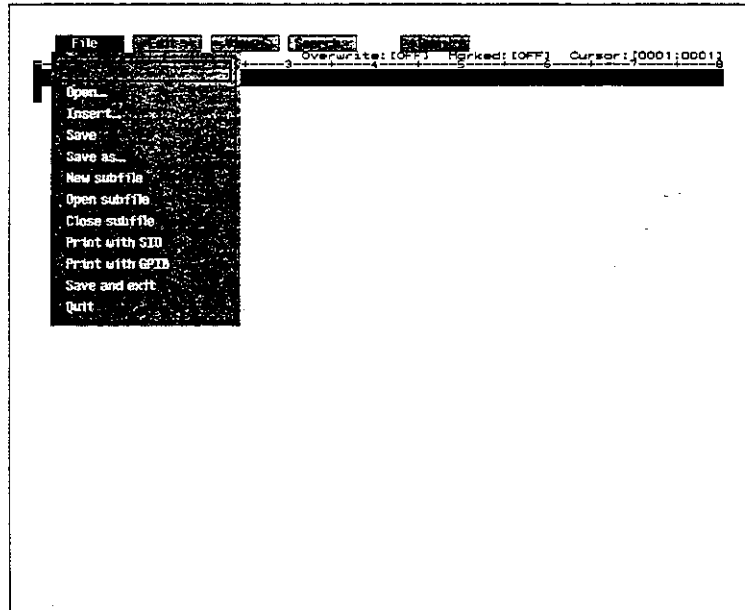


図 3 - 2 F1:File メニュー

プログラミング環境は、できるだけ簡単に操作できるよう設計されています。
ファンクション・キーを押すことによって、メニューをオープンし、コマンドの実行ができます（[3.2.1項] 参照）。

コマンドによっては、メニューを開けずに、ショートカットキーを使って、コマンドを直接実行することができます（[3.2.2項] 参照）。

名前の後に、省略記号(...) が付いてるコマンドは、コマンドの実行前に、より詳しい情報を指定する必要があることを示しています。

この種のコマンドを実行すると、ビュー・ウィンドウに対話ボックスが現れ、コマンドの実行に必要な項目を指定します。



3.2.1 キー操作によりコマンドを実行する


外部キーボードを使い、メニューをオープンして、エディタのコマンドを実行する方法を説明します。


メニューをオープンするには、メニュー名に対応するファンクション・キー（ F1 ~ F5 ）を押します。

あるメニューがオープンされているときに、←、→ を押すと、隣合わせの左右のメニューが変わってオープンされます。


メニューをオープンすると、そのメニューのコマンドが表示されます。実行可能なコマンドは、コマンド名がハイライト表示されます。

 ,  を押すと、コマンド名のハイライト表示が上下に移動します。

 を押すと、そのハイライト表示しているコマンドを実行できます。

コマンドの実行をキャンセルするには、 を押します。オープンしているメニューを閉じて、元の状態に戻ります。

名前の後に、省略記号(...) が付いているコマンドを実行すると、画面上に対話ボックスが現れます。この対話ボックスの中で、コマンドの実行に必要な項目を入力して下さい。

対話ボックスを取り消すには、 を押して下さい。コマンドの実行そのものが取り消されます。

3.2.2 ショートカットキーによりコマンドを実行する









ショートカットキーを使い、コマンドを実行する方法を説明します。

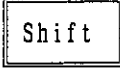

ショートカットキーを使うと、一回のキー操作でメニュー・コマンドを実行できます。メニュー・コマンドのショートカットキーは、メニュー上のコマンド名の右側に表示されます。

[表3-1]に、BASICエディタのショートカットキーの働きと、それに対応するコマンドの一覧を示します。

表 3 - 1 ショートカットキー操作

(1/2)

| キー操作 | 解説 |
|---|---------------------|
|  +  | 選択されている領域をカットする。 |
|  +  | クリップ・ボードの内容をペーストする。 |
|  +  | カーソル位置から行末までをカットする。 |
|  +  | クリップ・ボードの内容をペーストする。 |

(注) キー操作  +  とは、Shift キーを押しながらBackspace キーを押すという意味です。(以下も同様)

(2/2)

| キー操作 | 解説 |
|-------------|------------------------------|
| Ctrl + F3 | 次のバッファを読み込む。 |
| Alt + F3 | 〔Buffer list...〕対話ボックスを表示する。 |
| Ctrl + F4 | 次の検索対象を検索する。 |
| Alt + F4 | 次の検索対象を逆方向に検索する。 |
| Ctrl + F5 | プログラムを中断位置から再開する。 |
| Alt + F5 | プログラムを先頭から開始する。 |
| F11 | 次のウィンドウをアクティブにする。 |
| Shift + F11 | 前のウィンドウをアクティブにする。 |
| F12 | 編集画面と出力画面を切り替える。 |
| Shift + F12 | ウィンドウ分割を交互に切り替える。 |

3.3 対話ボックスの使い方

(1) 対話ボックスの働き

コマンドの実行前に、オプション指定などが必要なとき、エディタは対話ボックスを表示します。[図3-3]、[図3-4]に、対話ボックスの構成要素を示します。

対話ボックスは、以下のような働きをします。

- ファイル名の入力を求める
- オプションの設定を求める

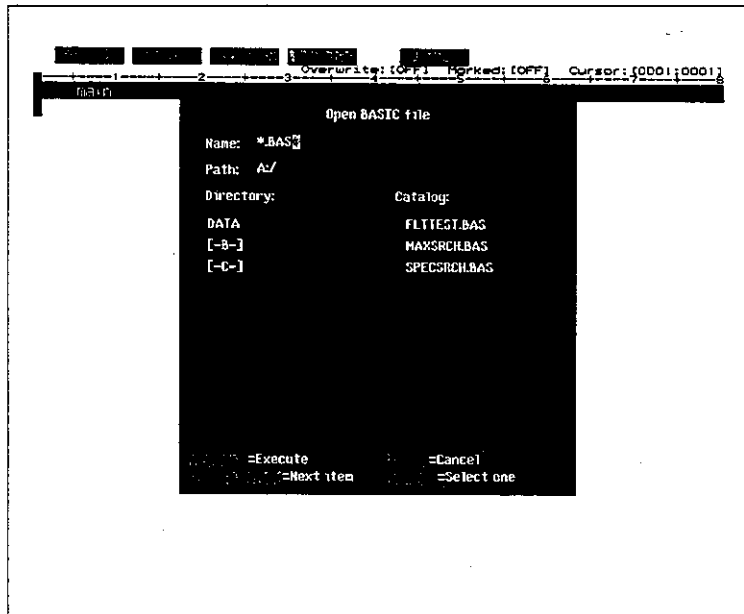


図 3 - 3 (Open...) コマンドの対話ボックス

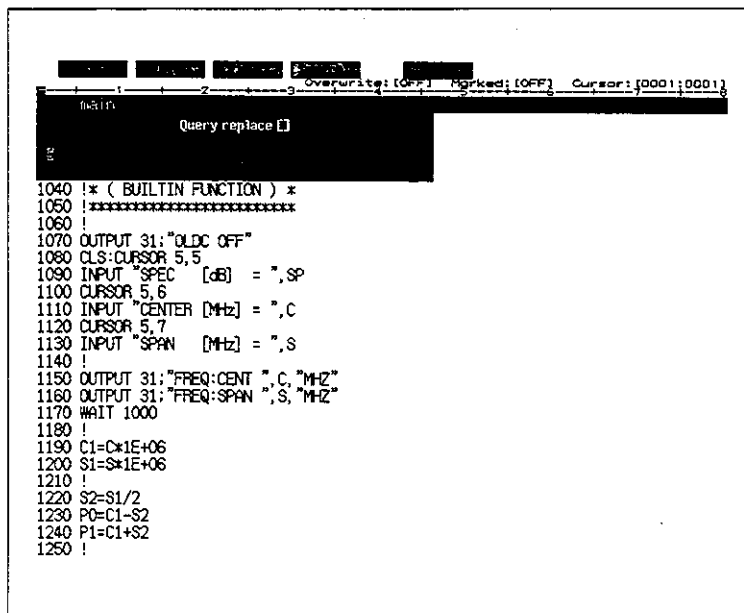


図 3 - 4 (Replace...) コマンドの対話ボックス

(2) カーソルの移動

対話ボックスの項目から項目へカーソルを移動するには、以下の方法があります。

- を押して、次の項目へ移動します。
- を押しながら を押して、前の項目へ移動します。

(3) 対話ボックスの各項目の働き

パス名指定フィールド :

カレント・ディレクトリのパスを表示します。

Nameフィールドで新しいパス名を入力するか、Directory ボックスで適切なディレクトリを選ぶと、パスの表示が変わります。

テキスト入力フィールド :

入力されたテキストを表示します。

リスト・ボックス :

ディレクトリやファイルなどの一覧を表示します。

コマンド・ボタン :

コマンドの実行に必要なキーの一覧を表示します。

対応するキーを押して実行します。

コマンドを実行するには、 を押します。

3.4 メッセージ・ラインの使い方

メッセージ・ラインは、画面の一番下に表示されます。
コマンドの実行の確認や取り消しを求めたり、エラーを通知したりします。
メッセージ・ラインは、次に行う操作で消去されます。

3.5 ウィンドウの使い方

読み込んだプログラムを表示する部分を「ビュー・ウィンドウ」といいます。
ビュー・ウィンドウは、分割して編集することができます。

編集するファイルやプログラムには、ファイル名の他にバッファ名が付けられます。
編集中のファイルで、ビュー・ウィンドウに表示されないものは、バッファ名によって
参照できます。

BASIC プログラムには、main というバッファ名が付けられます。
編集中のファイルで実際に実行できるのは、バッファ名 main のものだけです。


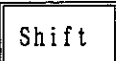

3.5.1 各ウィンドウの働き

ビュー・ウィンドウは、上下 2つのウィンドウに分けることができます。
ウィンドウを 2つに分けると、プログラムの 2つの部分を同時に見ながら編集できます。

ビュー・ウィンドウを分割するには、F3:View メニューの [Split window] コマンド
を実行して下さい。
再度このコマンドを実行すると、画面は元に戻ります。

3.5.2 アクティブ・ウィンドウの変更

カーソルを含むウィンドウを「アクティブ・ウィンドウ」と呼びます。
別のウィンドウをアクティブにするには、以下のいずれかの操作を行って下さい。

-  を押して、画面上で下にあるウィンドウを順次アクティブにします。
-  を押しながら  を押して、画面上で上にあるウィンドウを順次アクティブにします。

3.5.3 ウィンドウ・サイズの変更

ウィンドウ・サイズを 1行単位で、大きくしたり、小さくしたりできます。
また、1つのウィンドウを画面いっぱいに表示することもできます。

ウィンドウ・サイズを変えるには、サイズを変えるウィンドウをアクティブにし、次
に示すキー操作を行って下さい。

| キー操作 | 解説 |
|----------------|------------------------|
| Alt + PageUp | アクティブ・ウィンドウを 1行分大きくする。 |
| Alt + PageDown | アクティブ・ウィンドウを 1行分小さくする。 |
| Shift + f12 | アクティブ・ウィンドウを画面いっぱいにする。 |

3.5.4 アクティブ・ウィンドウのスクロール

ビュー・ウィンドウに表示されていない、ファイルの上下の部分を見たいときは、ビュー・ウィンドウをスクロールして下さい。
カーソルを画面の端まで移動させて方向キーを押すと、スクロールが始まります。但し、左右へのスクロールは、カーソル行だけがスクロールします。

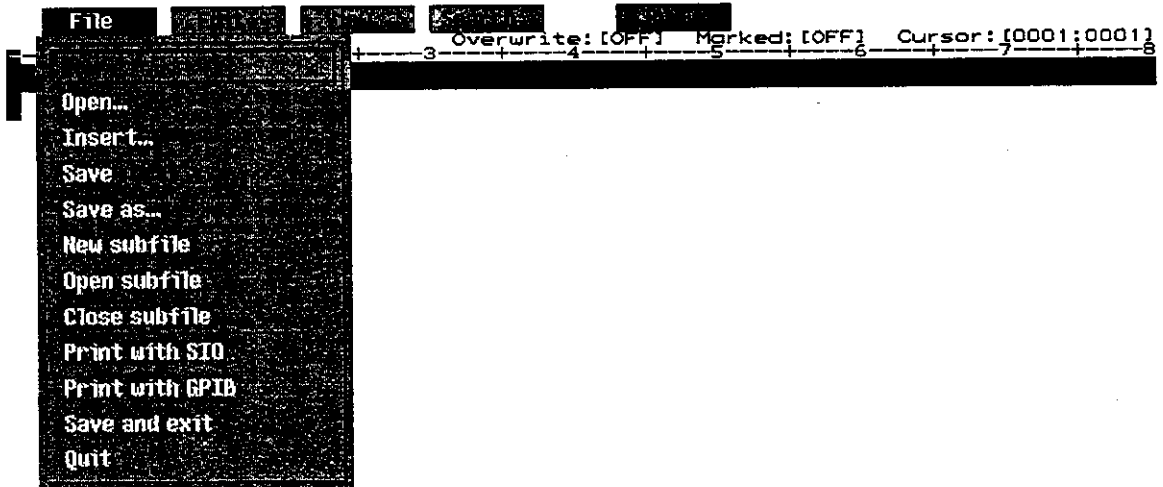
| キー操作 | 解説 |
|-------------|------------------|
| Home | ファイルの先頭に移動する。 |
| End | ファイルの終りに移動する。 |
| PageUp | 1 ページ分上にスクロールする。 |
| PageDown | 1 ページ分下にスクロールする。 |
| Ctrl + Home | 行の先頭に移動する。 |
| Ctrl + End | 行の終りに移動する。 |
| Ctrl + ↑ | 1 行分上にスクロールする。 |
| Ctrl + ↓ | 1 行分下にスクロールする。 |

MEMO 

4. F1:File メニュー (ファイル・メニュー)

F1:File メニューは、BASICで使用するファイルを編集するために用意されています。

F1:File メニューのコマンドを実行すると、新規ファイルを作ったり、フロッピー・ディスクからファイルをロードしたり、ファイルを修正したりすることができます。また、F1:File メニューのコマンドを使って、ファイルをライン・プリンタに印刷したり、エディタの作業を終わらせることができます。



- [New] コマンド
すでにロードされている BASICプログラムを消去します。
新しいプログラムを書き始めるときに使用します。
- [Open...] コマンド
フロッピー・ディスクに保存されているプログラムをロードします。
対話ボックスのファイルとディレクトリの一覧からファイルを選んで下さい。
- [Insert...] コマンド
2つのファイルの内容を結合し、1つのファイルにします。
- [Save] コマンド
アクティブなウィンドウに表示されているファイルの内容をフロッピー・ディスク・ファイルに書き込みます。
- [Save as...] コマンド
作業中のファイルを指定した名前でもフロッピー・ディスク・ファイルに書き込みます。
- [New subfile] コマンド
通常のテキストファイルを作ります。subfile は実行できません。
- [Open subfile] コマンド
フロッピー・ディスク上に保存されているプログラムや一般ファイルをsubfileとして読み込みます。
- [Close subfile] コマンド
編集中のテキスト・ファイルをメモリから解放します。

- [Print with SIO] コマンド
アクティブなウィンドウに表示されているファイルの内容をSIOポートより出力します。
- [Print with GPIB] コマンド
アクティブなウィンドウに表示されているファイルの内容をGPIBポートより出力します。
- [Save and exit] コマンド
編集中のファイルをすべてフロッピー・ディスク・ファイルに書き込み、エディタを終了します。
- [Quit] コマンド
エディタを終了します。

4.1 [New] コマンド

F1:File メニューの [New] コマンドは、それまでにロードされているBASIC プログラムを消去し、まったく新規にプログラムの入力を始められるようにします。

現在、メモリにロードされているプログラムが保存されていない場合は、メッセージ・ラインに以下のメッセージが表示されます。

Discard changes [y/n]?

Y

を押すと、フロッピー・ディスクに保存せずにメモリから解放し、アクティブ・ウィンドウには、別の編集中のファイルが変わって表示されます。

N

を押すと、別のバッファの編集に移ります。

[Save] コマンドまたは [Save as...] コマンドでフロッピー・ディスクに保存してから、[New] コマンドを実行して下さい。

4.2 [Open...] コマンド

F1:File メニューの [Open...] コマンドは、すでにフロッピー・ディスクに保存されているプログラムをロードします。
このコマンドを実行すると、対話ボックスが現れ、カレント・ディレクトリにある拡張子が .BAS のファイルの一覧を表示します。
この対話ボックスで、別のディレクトリや別のフロッピー・ディスクにあるファイルの一覧も表示することができます。

[Open...] コマンドを実行すると、[図4-1]のような対話ボックスが現れます。

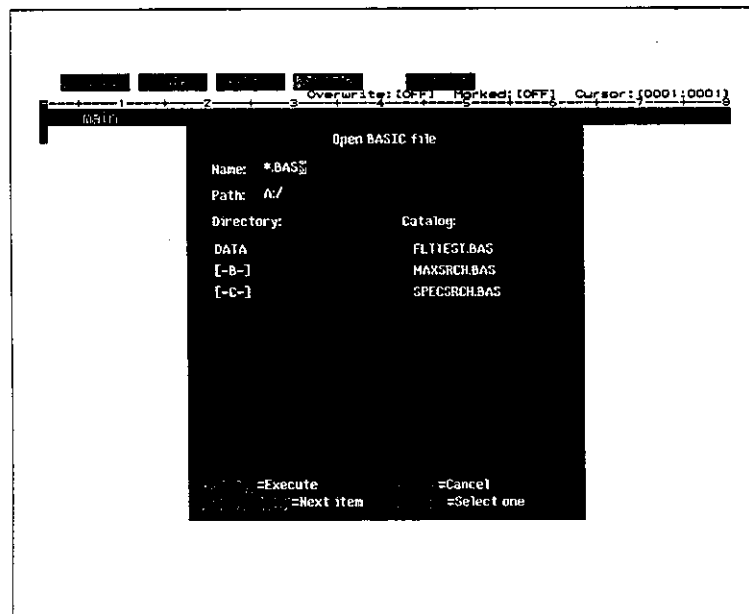


図 4 - 1 [Open...] コマンドの対話ボックス

4.2.1 ファイルの指定

ディレクトリとアクセスできるドライブの一覧は、Directory ボックスに表示されません。ファイルの一覧は、対話ボックスの中の Catalog ボックスに表示されます。

それぞれ、一度に表示できない場合は、、 で上下にスクロールします。

ロードしたいファイルを指定するには、以下に示す 2つの方法があります。

●直接ファイル名を入力する方法

Nameフィールドにプログラムのファイル名を入力して、 を押して下さい。

Nameフィールドに表示されている内容を消去する場合、 を押します。

そうすると、カーソルの左位置の文字を消去し、カーソルが 1文字分左へ移動します。

● Catalog ボックスから選択する

を続けて押して、Catalog ボックスに移ります。 , を使って、
ハイライト表示をロードしたいファイルの上に移動させ、 を押して下さい。

4.2.2 ディレクトリ内容の一覧表示

を続けて押して、Directory ボックスに移ります。 , を使って、
ハイライト表示をアクセスするディレクトリの上に移動させ、 を押します。

選択されたディレクトリにある、すべてのサブディレクトリと、そのディレクトリにある .BAS という拡張子を持ったファイルの一覧を表示します。

Catalog ボックスからファイル名を選ぶか、Nameフィールドでファイル名を入力して、
 を押すと、そのファイルがメモリにロードされます。

ディレクトリ内容を一覧表示する方法を、以下に示します。

- カレント・ディレクトリにある、すべてのファイルを表示する

Nameフィールドで、* と入力して を押します。

- フロッピー・ディスクのルート・ディレクトリのファイルを表示する

Pathフィールドで、A:/ と入力して を押します。

- SUB という名前のサブ・ディレクトリにあるすべてのファイルを表示する

Directory ボックスで、SUB という名前をハイライト表示にして、 を押します。次に、Nameフィールドで * と入力して、 を押します。

- カレント・ディレクトリの 1つ上のディレクトリのファイルを表示する

Directory ボックスで .. をハイライト表示にして、 を押します。

- 名前が 6文字のファイルを表示する

Nameフィールドで ?????? と入力して、 を押します。

- 名前が Bで始まるファイルを表示する

Nameフィールドで B* と入力して、 を押します。

- 拡張子が DATのファイルを表示する

Nameフィールドで *.DAT と入力して、 を押します。

- 拡張子が2文字のファイルを表示する

Nameフィールドで *.?? と入力して、 を押します。

- 拡張子の最後の文字が Bから Fのファイルを表示する

Nameフィールドで *.*[B-F] と入力して、 を押します。

4.3 [Insert...] コマンド

[Insert...] コマンドは、別のファイルの内容を作業中のファイルのカーソル位置に挿入します。

[Insert...] コマンドを実行すると、[図4-2]のような対話ボックスが現れます。対話ボックスの使い方は、[Open...] コマンドの対話ボックスとまったく同じです。

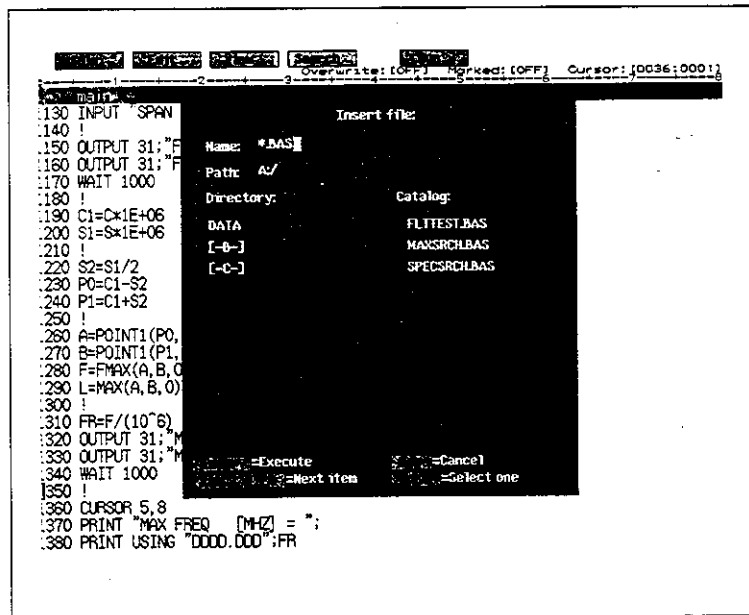


図 4 - 2 [Insert...] コマンドの対話ボックス

4.4 [Save] コマンド

[Save] コマンドは、作業中のファイルの内容をフロッピー・ディスク・ファイルに保存します。

保存しようとするファイルにすでに名前が付いていると、[Save] コマンドは、フロッピー・ディスク上の同名のファイルに上書きします。ファイル名が付いていない場合には、メッセージ・ラインに「No file name」が表示され、ファイルは保存されません。[Save as...] コマンドにてファイル名を指定して保存して下さい。

4.5 [Save as...] コマンド

[Save as...] コマンドは、作業中のファイルを指定した名前で保存します。このコマンドは、ファイルに新しい名前を付けたり、修正前のファイルを変更しないときに使います。ファイルに新しい名前を付けると、古いファイルは元の名前のままフロッピー・ディスク上に残ります。

[Save as...] コマンドを実行すると、[図4-3]のような対話ボックスが現れます。新しい名前を入力して保存すると、ウィンドウのタイトル・バーのファイル名が変わります。

```

main
1130 INPUT "SPAN [MHz] = ",S
1140 !
1150 OUTPUT 31:"FREQ:CENT ",C,"MHz"
1160 OUTPUT 31:"FREQ:SPAN ",S,"MHz"
Write file:
Z
1210 !
1220 S2=S1/2
1230 P0=C1-S2
1240 P1=C1+S2
1250 !
1260 A=POINT1(P0,0)
1270 B=POINT1(P1,0)
1280 F=FMAX(A,B,0)
1290 L=MAX(A,B,0)
1300 !
1310 FR=F/(10^6)
1320 OUTPUT 31:"MARK:ACT 1.",F
1330 OUTPUT 31:"MARK:LET CENT"
1340 WAIT 1000
1350 !
1360 CURSOR 5,8
1370 PRINT "MAX FREQ [MHz] = ";
1380 PRINT USING "0000.000";FR
    
```

図 4 - 3 [Save as...] コマンドの対話ボックス

4.6 [New subfile] コマンド

[New subfile] コマンドは、BASICプログラム以外のテキスト・ファイルを作成します。subfile として作られたファイルは、F5:Runメニューを使って実行することはできません。

[New subfile] コマンドを実行すると、[図4-4]のような対話ボックスが現れます。

```

main
1130 INPUT "SPAN [MHz] = ",S
1140 !
1150 OUTPUT 31:"FREQ:CENT " C,"MHz"
1160 OUTPUT 31:"FREQ:SPAN " S,"MHz"
Use buffer:
a
1210 !
1220 S2=S1/2
1230 P0=C1-S2
1240 P1=C1+S2
1250 !
1260 A=POINT1(P0,0)
1270 B=POINT1(P1,0)
1280 F=PMAX(A,B,0)
1290 L=MAX(A,B,0)
1300 !
1310 FR=F/(10^6)
1320 OUTPUT 31:"MARK:ACT 1," F
1330 OUTPUT 31:"MARK:LET CENT"
1340 WAIT 1000
1350 !
1360 CURSOR 5,8
1370 PRINT "MAX FREQ [MHz] = ";
1380 PRINT USING "0000.000",FR
    
```

図 4 - 4 [New subfile] コマンドの対話ボックス

この対話ボックスで、[バッファ名]を入力します。バッファ名とは、エディタが内部で編集ファイルを管理するために使う名前です。ここで編集したファイルを保存する場合には、[Save] または [Save as...] コマンドでファイル名を指定して保存して下さい。

プログラム・ファイルは 1ファイルしか読み込むことができませんが、サブ・ファイルは複数ファイル読み込むことができます。サブ・ファイルはF3:Viewメニューの[Buffer list...]で、バッファ名を通して現在編集にあるファイルをビュー・ウィンドウに表示したり、ファイル情報を表示することができます。

BASIC プログラムとして実行できるバッファ名は、mainとなっています。[New]、[Open...] コマンドで編集するファイルには、必ずmainが割り付けられます。

4.7 [Open subfile] コマンド

[Open subfile] コマンドは、フロッピー・ディスクからテキスト・ファイルを読み込みます。このコマンドで読み込んだファイルは、F5:Runメニューを使って実行することはできません。

[Open subfile] コマンドを実行すると、[図4-5]のような対話ボックスが現れます。

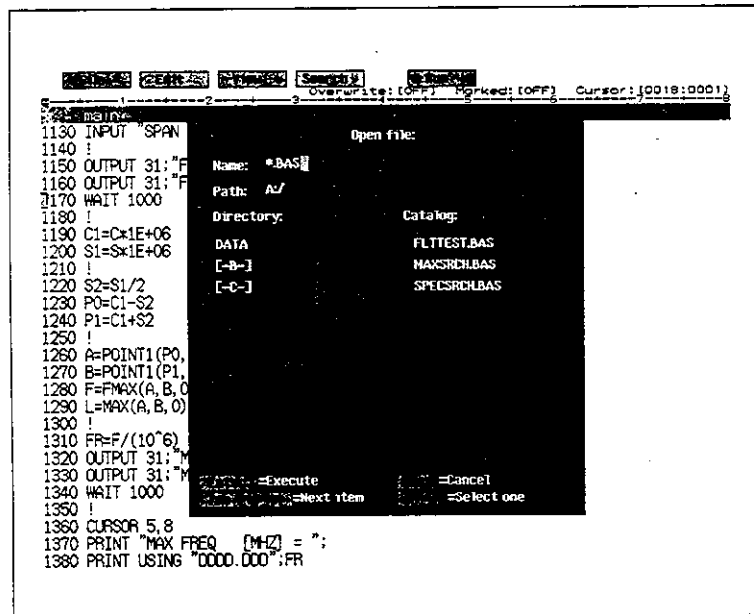


図 4 - 5 [Open subfile] コマンドの対話ボックス

対話ボックスの使い方は、[Open...] コマンドの対話ボックスとまったく同じです。

プログラム・ファイルは 1ファイルしか読み込むことができませんが、サブ・ファイルは複数ファイル読み込むことができます。サブ・ファイルはF3:Viewメニューの[Buffer list...]で、バッファ名を通して現在編集にあるファイルをビュー・ウィンドウに表示したり、ファイル情報を表示することができます。

BASIC プログラムとして実行できるバッファ名は、mainとなっています。[New]、[Open...] コマンドで編集するファイルには、必ずmainが割り付けられます。

4.8 [Close subfile] コマンド

[Close subfile] コマンドは、アクティブ・ウィンドウで編集中のサブファイルをメモリから消去します。

作業中、変更されたサブファイルがフロッピー・ディスクに保存されていない場合は、メッセージ・ラインに以下のメッセージが表示されます。

Discard changes [y/n]?

Y

を押すと、フロッピー・ディスクに保存せずにメモリから解放し、アクティブ・ウィンドウには、別の編集中のファイルが変わって表示されます。

N

を押すと、別のバッファの編集に移ります。

[Save] コマンドまたは [Save as...] コマンドでフロッピー・ディスクに保存してから、[Close subfile] コマンドを実行して下さい。

4.9 [Print with SIO] コマンド

[Print with SIO] コマンドは、アクティブ・ウィンドウに表示されているファイルの内容を SIOポート (RS-232Cポート) より出力します。

[Print with SIO] コマンドを使用する場合には、プリンタを本体のRS-232Cポートに接続して下さい。

4.10 [Print with GPIB] コマンド

[Print with GPIB] コマンドは、アクティブ・ウィンドウに表示されているファイルの内容をGPIBポートより出力します。

[Print with GPIB] コマンドを使用する場合には、プリンタを本体のGPIBポートに接続して下さい。

4.11 [Save and exit] コマンド

[Save and exit] コマンドは、現在ロードされているファイルのうち、作業中に編集を加えられたすべてのファイルを保存し、エディタを終了します。現在ロードされているファイルの名前は、F3:Viewメニューの[Buffer list...] コマンドの対話ボックスに表示されます。

ファイル名が指定されていないファイルがある場合は、No file nameと表示され、元の画面に戻ります。[Save as...] コマンドでファイルに名前をつけて下さい。

4.12 [Quit] コマンド

[Quit] コマンドは、エディタをメモリから解放し、終了します。

エディタを終了するとき、まだ保存していない新規のファイルや、編集を加えたプログラムがあると、メッセージ行に以下の表示を行います。

Modified buffers exist. Save all [y/n]?

Y を押すと、変更が加えられたファイルを保存せずに、エディタを終了します。

N を押すと、元の画面に戻ります。

すべてを保存して終了する場合は、[Save and exit] コマンドを実行して下さい。

MEMO 

5. エディタの基本操作

この章では、エディタの基本的な使い方やプログラム・テキストを入力する方法について述べます。

以下の項目を説明します。

- テキストの入力とカーソル移動
- テキストの削除と挿入
- テキストのブロック移動とコピー
- 文字、単語、または文の検索と置き換え
- 他のファイルからテキストをコピーする方法

5.1 テキストの入力

テキストに文字を書き加えるには、「挿入」と「上書き」の2つの方法があります。挿入モードのとき、エディタは入力した文字をカーソルの左側に挿入します。上書モードのとき、カーソル位置の文字が入力した文字と置き換えられます。

挿入モードと上書モードを切り替えるには、 を押しながらか を押し

ます。

モードの設定状態は、ステータス・ライン上で確認できます。

挿入モード時には Overwrite:[OFF]、上書きモード時には Overwrite:[ON] と表示されます。

5.2 テキストの選択

編集機能を使ってテキスト・ブロックを操作する場合、まず、テキストの範囲を選んで編集を加える部分を指定します。

- ① 目的のテキストの先頭にカーソルを移動し、 を押しながらか を押し、
て、マークをセットします。
マークがセットされると、ステータス・ライン上の Marked:[] が ON になります。

- ② テキストの終りへ移動し、編集コマンドを実行します。

一度、マークがセットされると、マークがセットされたテキストから現在のカーソル位置のテキストまでが、いつも選択状態になります。

5.3 テキストのインデント

テキストをインデントすると、プログラムの構造が読みやすくなります。

テキストをインデントするには、行頭で または を入力します。

行頭から空白文字をスキップしたカラム位置に、インデントがセットされます。







以降、 を押すと、次の行の同じカラム位置にカーソルが移動します。

5.4 編集コマンドの概要

カーソルを移動したり、アクティブ・ウィンドウを移動するのに、簡単なキーの組み合わせで行うことができます。以下に編集コマンドの一覧を示します。

表 5 - 1 編集コマンド一覧

(1/3)

| | キー操作 | 解説 |
|----------------|---|--------------------|
| (1) カーソルの移動 |  | 1 文字左に移動する。 |
| |  | 1 文字右に移動する。 |
| | Ctrl +  | 1 単語左に移動する。 |
| | Ctrl +  | 1 単語右に移動する。 |
| |  | 1 行上に移動する。 |
| |  | 1 行下に移動する。 |
| | Home | ファイルの先頭に移動する。 |
| | End | ファイルの終りに移動する。 |
| | Ctrl + Home | カーソル行の先頭に移動する。 |
| | Ctrl + End | カーソル行の終りに移動する。 |
| | Alt + Home | 他のウィンドウへ移動する。 |
| | Alt + End | 他のウィンドウへ移動する。 |
| | Ctrl + Space | カーソル位置にマーカをセットする。 |
| | Alt + Space | マーカ位置とカーソル位置を交換する。 |

(2/3)

| | キー操作 | 解説 |
|-----------|--|--|
| (2) 挿入 | <p>Enter</p> <p>Insert</p> <p>Ctrl + Insert</p> <p>Shift + Insert</p> <p>Alt + Insert</p> | <p>改行を挿入する。</p> <p>1文字分の空白を挿入する。</p> <p>空白行を挿入する。</p> <p>クリップ・ボードの内容を挿入する。</p> <p>挿入／上書きモードを切り替える。</p> |
| (3) 削除 | <p>Backspace</p> <p>Delete</p> <p>Ctrl + Backspace</p> <p>Ctrl + Delete</p> <p>Shift + Backspace</p> <p>Shift + Delete</p> | <p>カーソルの左側の文字を削除する。</p> <p>カーソル位置の文字を削除する。</p> <p>カーソル位置の単語の左側を削除する。</p> <p>カーソル位置の単語の右側を削除する。</p> <p>選択したテキストを削除する。</p> <p>行末までを削除する。</p> |

(3/3)

| | キー操作 | 解説 |
|--------------------------|-----------------|------------------|
| (4) スク ロ ー ル | Ctrl + ↑ | 1 行上にスクロールする。 |
| | Ctrl + ↓ | 1 行下にスクロールする。 |
| | PageUp | 1 ページ分上にスクロールする。 |
| | PageDown | 1 ページ分下にスクロールする。 |
| | Ctrl + PageUp | ウィンドウを 1行分大きくする。 |
| | Ctrl + PageDown | ウィンドウを 1行分小さくする。 |

6. F2:Edit メニュー (編集メニュー)

F2:Edit メニューには、プログラムやテキストを書いたり、入れ換えたりするために使うコマンドが用意されています。

F2:Edit メニューのコマンドを使うと、テキストのカット、コピー、ペーストを行ったりすることができます。

編集するテキストは、あらかじめ選択しておく必要があります。選択方法は、[5.2 テキストの選択] を参照して下さい。



- [Cut] コマンド
選んだテキストを削除してクリップ・ボードへ送ります。
- [Copy] コマンド
選んだテキストのコピーをクリップ・ボードへ送ります。
- [Paste] コマンド
クリップ・ボードの内容をカーソル位置に挿入します。
- [Upper case] コマンド
選んだテキストを大文字に変換します。
- [Lower case] コマンド
選んだテキストを小文字に変換します。

6.1 クリップ・ボードについて

クリップ・ボードとは、カットまたはコピーしたテキストを一時的に保管する場所のことです。

F2:Edit メニューの [Cut] および [Copy] でテキストの削除や複写を行うと、それらのテキストはクリップ・ボードに保管されます。

クリップボードの内容は、[Paste] コマンドを使って、カーソル位置に挿入することができます。クリップ・ボードには同時に 1つのブロックしか保管できません。

クリップ・ボードのテキストは、何度でもペーストできます。

6.2 [Cut] コマンド

[Cut] コマンドは、選んだテキストを画面上から削除し、クリップ・ボードへ送ります。[Cut] コマンドを実行する場合、必ずテキストを選んだ後に行ってください。

[Cut]、[Paste] コマンドを使うと、行やテキスト・ブロックの移動ができます。

● 行やテキスト・ブロックの移動

- ① 移動したいテキストを選びます。([5.2 節] 参照)
- ② F2:Edit メニューの [Cut] コマンドを実行します。
- ③ カットしたテキストを挿入したい位置にカーソルを移動します。
- ④ F2:Edit メニューの [Paste] コマンドを実行します。

● ショートカットキー :

Shift + Backspace

6.3 [Copy] コマンド

[Copy] コマンドは、選んだテキストをそのままにして、テキストのコピーをクリップ・ボードへ送ります。[Copy] コマンドを実行する場合は、必ずテキストを選択した後に行ってください。

[Copy]、[Paste] コマンドを使うと、プログラムの一部または全部を複製することができます。

● プログラムの一部または全部を複製する

- ① 複製したいテキストを選びます。([5.2 節] 参照)
- ② F2:Edit メニューの [Copy] コマンドを実行します。
- ③ コピーしたテキストは、挿入したい位置にカーソルを移動します。
- ④ F2:Edit メニューの [Paste] コマンドを実行します。
- ⑤ 必要に応じて、コピーしたテキストを修正して下さい。

6.4 [Paste] コマンド

[Paste] コマンドは、クリップ・ボードの内容のコピーをカーソル位置に挿入します。このコマンドは、クリップ・ボードにテキストが保管されているときだけ使うことができます。

● ショートカットキー : Shift + Insert

6.5 [Upper case] コマンド

[Upper case] コマンドは、選んだテキスト全体の英文字を大文字に変換します。
[Upper case] コマンドを実行する場合は、必ずテキストを選択した後に行ってください。

6.6 [Lower case] コマンド

[Lower case] コマンドは、選んだテキスト全体の英文字を小文字に変換します。
[Lower case] コマンドを実行する場合は、必ずテキストを選択した後に行ってください。

MEMO 

7. F3:View メニュー (ビュー・メニュー)

F3:View メニューには、ビュー・ウィンドウを分割したり、すでに読み込んでいるファイルの中を、ビュー・ウィンドウに表示して編集することができます。



- [Buffer list...] コマンド
すでにフロッピー・ディスクから読み込んでいるプログラムや、サブ・ファイルのバッファ一覧を見ることができます。
また、バッファ一覧の中からファイルを選び、ビュー・ウィンドウに表示させることができます。
- [Next buffer] コマンド
アクティブ・ウィンドウに次のバッファのファイルを表示します。
- [Split window] コマンド
ビュー・ウィンドウを上下 2つに分割します。
- [Execution display] コマンド
エディタ画面と測定画面とを切り替えます。

7.1 [Buffer list...] コマンド

[Buffer list...] コマンドを使うと、現在読み込まれているファイルの一覧を見て、目的のファイルを選ぶことができます。選択されたファイルは、ビュー・ウィンドウに表示されます。

[Buffer list...] コマンドを実行すると、[図7-1]のような対話ボックスが現れます。編集したいファイルを選択して下さい。

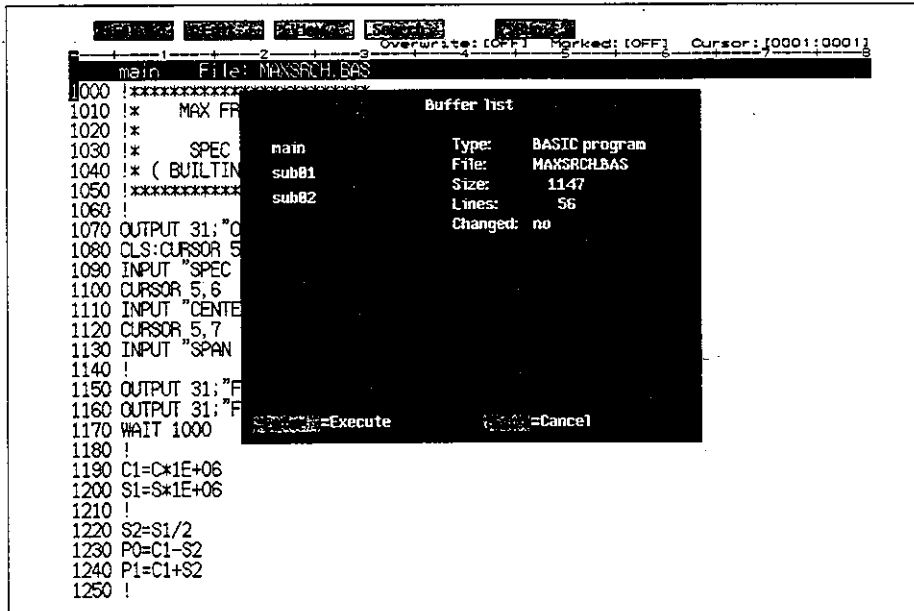


図 7 - 1 (Buffer list...) コマンドの対話ボックス

● ファイルを表示する

① F3:View メニューの (Buffer list...) コマンドを実行します。
[図7-1]のような対話ボックスが現れます。

② , , , を使って、目的のバッファ名をハイライト表示にし、以下の操作を 1つ実行します。

・ハイライト表示にした項目をアクティブ・ウィンドウに表示するには、

を押します。

・処理を取り消す場合は、 を押します。

● ショートカットキー : +

7.2 [Next buffer] コマンド

編集メモリに複数のファイルを読み込んでいるとき、[Next buffer] コマンドを実行すると、アクティブ・ウィンドウにアルファベット順で次のバッファを表示します。このコマンドは、読み込んでいるファイルが 1つしかない場合、何も機能しません。

- ショートカットキー : Ctrl + F3





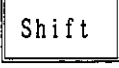

7.3 [Split window] コマンド

[Split window] コマンドは、ビュー・ウィンドウを上下 2つに分割します。ビュー・ウィンドウを分割すると、ファイルの 2つの部分を同時に見ながら作業することができます。

- 複数のファイルを同時に見ながら作業する
 - ① [Split window] コマンドで、ビュー・ウィンドウを分割します。
 - ② F1:File メニューで、[Open subfile] コマンドを実行するか、または F3:View メニューで [Buffer list...] コマンドを実行し、アクティブ・ウィンドウにファイルを表示させます。(アクティブ・ウィンドウは、カーソルのあるウィンドウです。)
- [Split window] コマンドでビュー・ウィンドウを分割して作業する
 - ① F3:View メニューの [Split window] コマンドを実行します。
 - ② Alt を押しながら Home (または Alt を押しながら End) を押すと、カーソルがもう一方のウィンドウに移動します。これにより、カーソルのあるウィンドウがアクティブ・ウィンドウとなります。
 - ③ 再度 F3:View メニューの [Split window] コマンドを実行すると、そのときアクティブであるウィンドウがビュー・ウィンドウいっぱいになり、アクティブでないウィンドウは閉じます。

● ウィンドウ・サイズの変更

ウィンドウ・サイズを変えるには、サイズを変えるウィンドウをアクティブにて、以下の操作を行って下さい。

| キー操作 | 解説 |
|---|-------------------------|
|  +  | アクティブ・ウィンドウが 1行分大きくなる。 |
|  +  | アクティブ・ウィンドウが 1行分小さくなる。 |
|  +  | アクティブ・ウィンドウが画面いっぱいに広がる。 |

7.4 [Execution display] コマンド

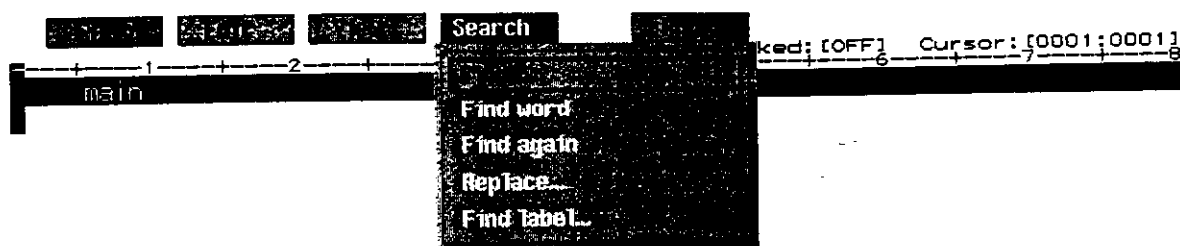
[Execution display] コマンドは、BASIC の編集画面と測定画面を切り替えて表示します。このコマンドは、プログラムの編集中なら、いつでも使えます。特にプログラムが測定条件の切り替えを行った後、その結果を確認するために使うと便利です。

● ショートカットキー :



8. F4:Search メニュー (検索メニュー)

F4:Search メニューのコマンドを使うと、ある文字列の検索や検索した文字列の置き換えができます。大きなプログラムを編集している場合、スクロールに時間がかかります。このとき、目的の文字列を検索してカーソルをジャンプさせることができます。また、変数名を変更したい場合は、置換機能を使うことによって、簡単に変更することができます。



- [Find...] コマンド
指定した文字列を検索し、カーソルを検索文字列の後に移動します。
- [Find word] コマンド
カーソル位置の単語と同じ文字列を検索し、カーソルを検索文字列の後に移動します。
- [Find again] コマンド
直前に検索した文字列を現在のカーソル位置から検索します。
- [Replace...] コマンド
指定した文字列を検索し、新しいテキストと置き換えます。
- [Find label...] コマンド
指定した行ラベルを検索します。

8.1 [Find...] コマンド

[Find...] コマンドは、指定した文字列をカーソル位置より検索し、文字列が見つかったら、その文字列の後へカーソルを移動します。
検索文字列は、文字 (スペース等を含む)、文字列、単語のいずれでも構いません。

[Find...] コマンドを実行すると、[図8-1]のような対話ボックスが現れます。
テキスト・ボックスに検索したい文字列を入力して下さい。

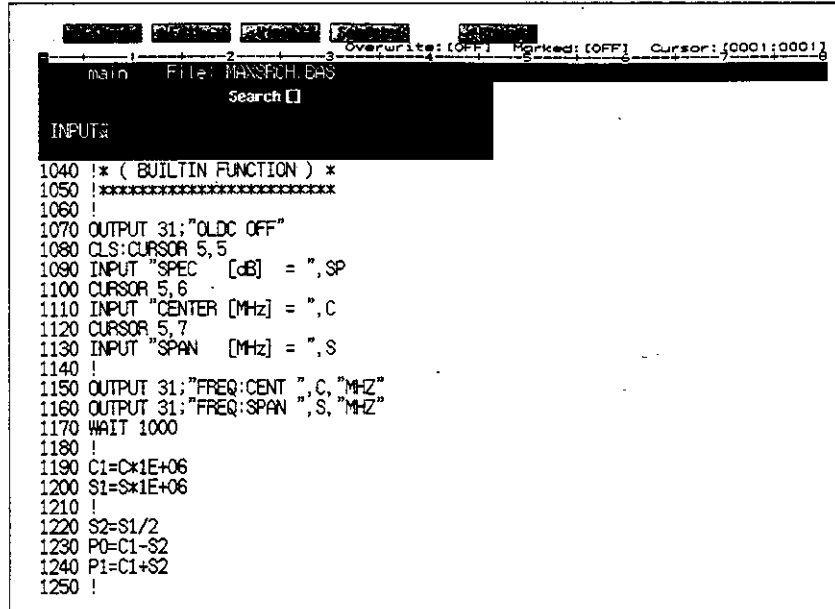


図 8 - 1 [Find...] コマンドの対話ボックス

検索は、カーソル位置よりアクティブ・ウィンドウに表示されているファイル全体に渡って行われます。

●テキストを検索する

- ① F4:Search メニューの [Find...] コマンドを実行します。
- ② Find text テキスト入力フィールドで、検索するテキストを入力して下さい。

指定した文字列が見つかった場合は、カーソルが検索文字列の後へ移動します。

指定した文字列が見つからないと、Not found というメッセージが、画面の下に表示され、カーソルは移動しません。次に行う操作でメッセージは消去できます。

8.2 [Find word] コマンド

[Find word] コマンドは、アクティブ・ウィンドウのカーソル位置にある単語（連続した英数字）を検索します。

- [Find word] コマンドの使い方

- ① 検索したい単語上にカーソルを移動させます。選べる単語の長さは 1行以内です。
- ② F4:Search メニューの [Find word] コマンドを実行します。

8.3 [Find again] コマンド

[Find again] コマンドは、直前に検索したテキストを再度検索します。つまり、[Find...], [Find word], [Replace...] コマンドで指定した検索文字列が対称となります。検索文字列が入力されていない状態では、[Find again] コマンドは何も処理しません。

- ショートカットキー :

| | | |
|------|---|----|
| Ctrl | + | F4 |
|------|---|----|

 (カーソル位置から後方向に)

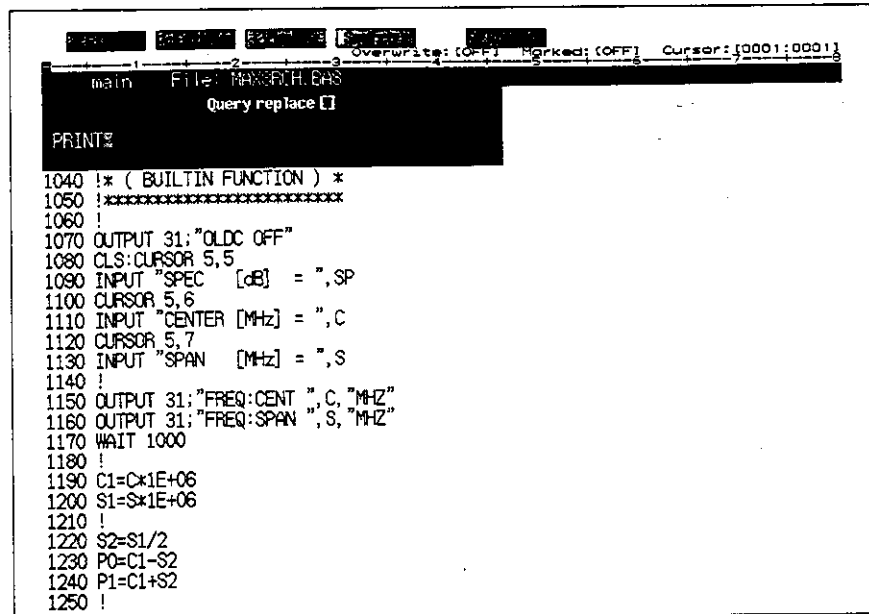
| | | |
|-----|---|----|
| Alt | + | F4 |
|-----|---|----|

 (カーソル位置から前方向に)

8.4 [Replace...] コマンド

[Replace...] コマンドは、指定した文字列を検索して、それを別の文字列に置き換えます。検索する文字列は、文字や単語、文字と単語の組み合わせが可能です。

[Replace...] コマンドを実行すると、[図8-2]のような対話ボックスが現れます。この対話ボックスで、検索文字列、置換文字列を入力して下さい。



```
main File: MAXSEARCH.BAS
Query replace [ ]

PRINTZ
1040 !* ( BUILTIN FUNCTION ) *
1050 !*****
1060 !
1070 OUTPUT 31: "OLDC OFF"
1080 CLS:CURSOR 5,5
1090 INPUT "SPEC [dB] = ",SP
1100 CURSOR 5,6
1110 INPUT "CENTER [MHz] = ",C
1120 CURSOR 5,7
1130 INPUT "SPAN [MHz] = ",S
1140 !
1150 OUTPUT 31: "FREQ:CENT ",C,"MHz"
1160 OUTPUT 31: "FREQ:SPAN ",S,"MHz"
1170 WAIT 1000
1180 !
1190 C1=C*1E+06
1200 S1=S*1E+06
1210 !
1220 S2=S1/2
1230 P0=C1-S2
1240 P1=C1+S2
1250 !
```

図 8 - 2 [Replace...] コマンドの対話ボックス

● 指定テキストを置換する

- ① F4:Serach メニューの [Replace...] コマンドを実行します。
- ② Find text テキスト入力フィールドで検索文字列を入力します。
- ③ Replace text テキスト入力フィールドで置換文字列を入力します。
- ④ を押すと、テキストの置換が始まります。

- ⑤・文字列が見つかったとき、メッセージ行に以下のメッセージが表示されます。

Replace 'find-text' with 'replace-text' ?

Y を押すと、文字列を置き換えてから次の検索を行います。

N を押すと、置換を行わずに次の検索を行います。

この状態で選択できる機能を、以下に示します。

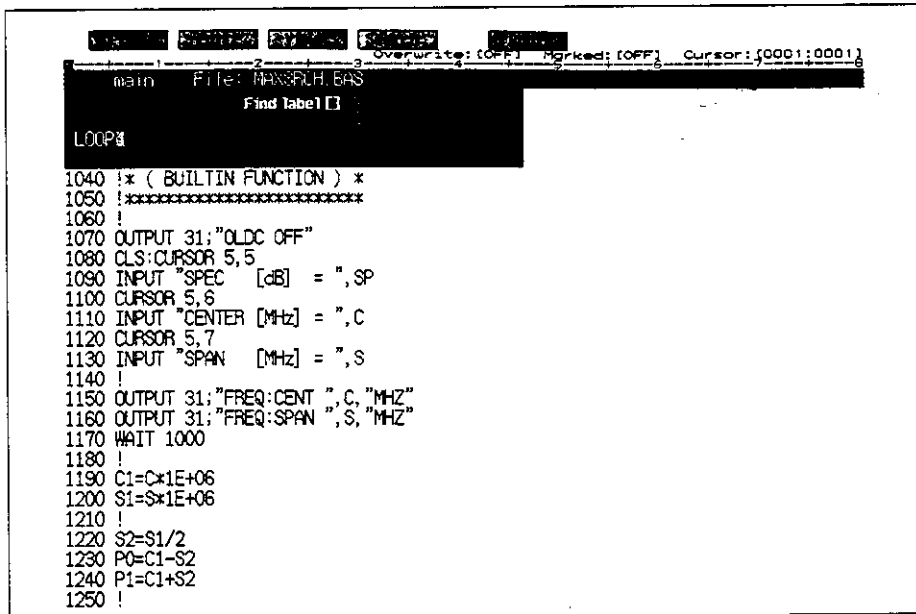
| キー操作 | 解説 |
|------------------------------|---------------------------|
| <input type="checkbox"/> Esc | カーソル検索文字列に移動したまま、検索を中止する。 |
| <input type="checkbox"/> ! | 残りを一括して置換する。 |
| <input type="checkbox"/> ? | キー操作リストを表示する。 |
| <input type="checkbox"/> . | 検索を中止して、検索を始めた位置にカーソルを戻す。 |
| <input type="checkbox"/> Y | テキストを置き換えて次を検索する。 |
| <input type="checkbox"/> N | 置換を行わずに次を検索する。 |

- ・検索文字列が見つからなかったときは、Not found というメッセージがメッセージ行に表示されます。メッセージは、次に行う操作で消去されます。

8.5 [Find label...] コマンド

[Find label...] コマンドは、BASIC プログラムの行ラベルを検索します。
行ラベルは、必ずアスタリスク (*)から始まるので、Find text テキスト入力フィールドで入力したテキストに、アスタリスクが付加されたものだけが検索対象となります。

[Find label...] コマンドを実行すると、[図8-3]のような対話ボックスが現れます。

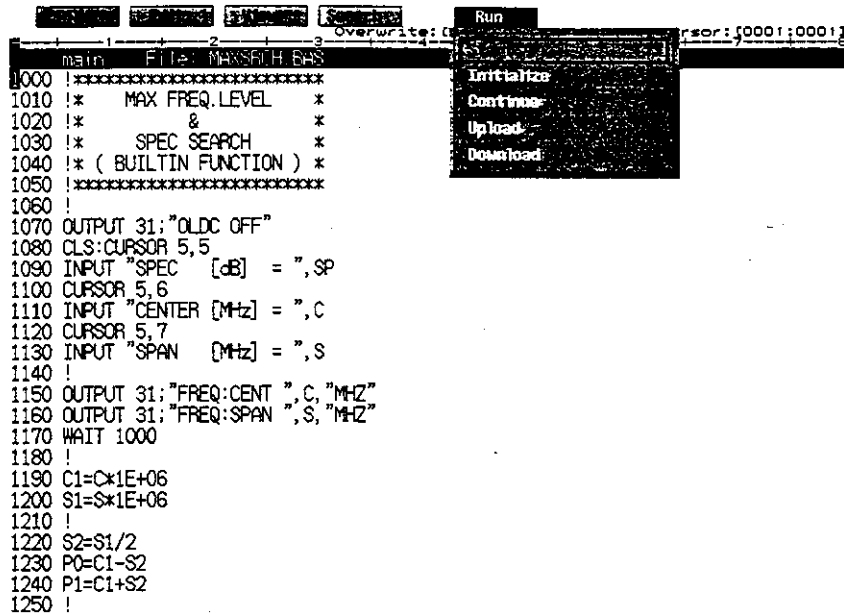


```
main File: HANDBOOK.BAS Overwrite: [OFF] Marked: [OFF] Cursor: {0001:0001}
Find label [ ]
LOOP
1040 !* ( BUILTIN FUNCTION ) *
1050 !*****
1060 !
1070 OUTPUT 31;"OLDC OFF"
1080 CLS:CURSOR 5,5
1090 INPUT "SPEC [dB] = ",SP
1100 CURSOR 5,6
1110 INPUT "CENTER [Mhz] = ",C
1120 CURSOR 5,7
1130 INPUT "SPAN [MHz] = ",S
1140 !
1150 OUTPUT 31;"FREQ:CENT ",C,"MHZ"
1160 OUTPUT 31;"FREQ:SPAN ",S,"MHZ"
1170 WAIT 1000
1180 !
1190 C1=C*1E+06
1200 S1=S*1E+06
1210 !
1220 S2=S1/2
1230 P0=C1-S2
1240 P1=C1+S2
1250 !
```

図 8 - 3 [Find label...] コマンドの対話ボックス

9. F5:Runメニュー (実行メニュー)

F5:Runメニューは、BASICのプログラミング環境でのプログラム実行や、BASICメモリからの Upload、BASICメモリへの Download などを行うコマンドがあります。



The screenshot shows a BASIC editor window with a file named 'MAXSEARCH.BAS'. The program code is as follows:

```
1000 !*****  
1010 !* MAX FREQ LEVEL *  
1020 !* & *  
1030 !* SPEC SEARCH *  
1040 !* ( BUILTIN FUNCTION ) *  
1050 !*****  
1060 !  
1070 OUTPUT 31;"OLDC OFF"  
1080 CLS:CURSOR 5,5  
1090 INPUT "SPEC [dB] = ",SP  
1100 CURSOR 5,6  
1110 INPUT "CENTER [MHz] = ",C  
1120 CURSOR 5,7  
1130 INPUT "SPAN [MHz] = ",S  
1140 !  
1150 OUTPUT 31;"FREQ:CENT ",C,"MHZ"  
1160 OUTPUT 31;"FREQ:SPAN ",S,"MHZ"  
1170 WAIT 1000  
1180 !  
1190 C1=C*1E+06  
1200 S1=S*1E+06  
1210 !  
1220 S2=S1/2  
1230 P0=C1-S2  
1240 P1=C1+S2  
1250 !
```

Over the 'Run' menu, the following options are visible:

- Initialize
- Continue
- Upload
- Download

- [Start] コマンド
プログラムを実行します。
- [Initialize] コマンド
すべての数値変数を 0 にリセットし、BASIC を初期状態にします。
- [Continue] コマンド
実行が中断されたステートメントから、プログラムの実行を再開します。
このとき、変数の値はリセットされません。
- [Upload] コマンド
BASIC メモリにロードされているプログラムを、エディタに読み込みます。
- [Download] コマンド
現在編集中のプログラムを、BASIC メモリへロードします。

9.1 [Start] コマンド

F5:Runメニューの [Start] コマンドは、プログラムを実行します。
編集中のプログラムが、まだ BASICメモリにロードされていない場合や、編集を加えたときには、最初にプログラムを BASICメモリへロードします。
プログラムの開始行に行番号がなければ、自動的に行番号が付加されます。

プログラムの実行を一時中断するには、Pause を押して下さい。

プログラムを再開するには、F5:Runメニューの [Continue] コマンドを実行して下さい。

プログラムを最初から実行するには、[Start] コマンドを実行して下さい。

プログラムを実行を開始すると測定画面に変わり、プログラムが中断するとエディタ画面に戻ります。

● ショートカットキー : Alt + F5

9.2 [Initialize] コマンド

[Initialize] コマンドは、プログラムをデバッグするときに使います。
このコマンドは、プログラム中のすべての変数の値を 0 にリセットします。
このコマンドは、プログラムを実行する準備をしますが、プログラムの実行はしません。

9.3 [Continue] コマンド

[Continue] コマンドは、デバッグのために使います。
このコマンドを使うと、ブレーク・ポイントやウォッチ・ポイントから、次のブレーク・ポイントやウォッチ・ポイントまで、プログラムを実行することができます。
このコマンドは、プログラムの実行を一時中断したときに、最後に実行されたステートメントから実行を再開するか、またはプログラムを最初から実行します。

● ショートカットキー : Ctrl + F5

9.4 [Upload] コマンド

[Upload] コマンドは、BASICメモリにロードされているプログラムをエディタへ読み込みます。

9.5 [Download] コマンド

[Download] コマンドは、BASIC プログラムを実行せずに BASICメモリに編集中のプログラムをロードします。
行番号を付けずに編集されたプログラムに対しては、自動的に行番号を与え、BASICメモリにロードします。(編集中のプログラムに直接、行番号は付加されません。)

10. ネットワーク・アナライザでの自動測定

この章では、実際にネットワーク・アナライザ（本体）で測定するプログラムの作り方を説明します。

(注) この章で示すプログラムは、R3752/53用の例です。R3765/67, R3764/66 上で使用するには、初期設定状態や周波数範囲などの違いに応じて、変更を要する場合があります。

10.1 OUTPUTとENTER 命令を使用したプログラム

10.1.1 プログラムの実行

周波数を指定して、その位置にマーカを出現させ、データを読み込んで、その周波数とレベルと位相を表示するプログラムです（このプログラムは、R3752 では実行できません）。

例 10 - 1 OUTPUT とENTER 命令を使用したプログラム

```
100 !*****
110 !*   OUTPUT / ENTER   *
120 !*****
130 !
140 OUTPUT 31;"OLDC OFF"
150 OUTPUT 31;"MARK:ACT 1,380E+6"
160 OUTPUT 31;"FETC?"
170 ENTER 31;F,L,P
180 PRINT "FREQ [Hz] = ",F
190 PRINT "LEVEL [dB] = ",L
200 PRINT "PHASE [deg]= ",P
210 STOP
```

[例10-1] のプログラムを RUN(BASICコマンド) で実行させると、マーカが出現して、その示す位置の周波数とレベルが表示します。

```
FREQ [Hz] = 3.8e+08
LEVEL [dB] = 0.7818921033
PHASE [deg] = 109.241912841
```

このプログラムの実行結果で表示しているレベルの値は、DUT(Device Under Test: 被測定物) に380MHzのセラミックのBPF(バンド・パス・フィルタ) を使用しています。

R3752 では、マーカを扱う GPIB コマンドを持っていません。このため、R3752 で波形解析を行う場合はビルトイン関数を利用します。

[例10-1] のプログラムは以下のようになります (このプログラムは、R3752 でも使用できます)。

ビルトイン関数についての詳細は、[10.2 節] を参照して下さい。

例 10 - 2 OUTPUT と ENTER 命令を使用したプログラム (ビルトイン関数使用)

```
100 PRINT "FREQ [Hz] = ", 3.8e+8
110 PRINT "LEVEL [dB] = ", CVALUE(3.8e+8, 0)
120 PRINT "PHASE [deg]= ", CVALUE(3.8e+8, 8)
130 STOP
```

【プログラムの解説】

[例10-1] のプログラムの流れを解説します。

| 例10-1の解説 | |
|----------|--|
| 100 | |
| 110 | 注釈行 |
| 130 | |
| 140 | 本体の GPIB コマンド・モードを設定する |
| 150 | 1 番のマーカを 380MHz に設定する |
| 160 | 150 行で表示した 1 番のマーカの位置を送出させる |
| 170 | 160 行で送出处のデータを受け取り、必要なデータを変数に代入する (ここでは、周波数とレベルが必要なので、F に周波数を、L にレベルを、 P に位相を代入する) |
| 180 | 変数 F に入っているデータを PRINT 文で画面に表示する |
| 190 | 変数 L に入っているデータを PRINT 文で画面に表示する |
| 200 | 変数 P に入っているデータを PRINT 文で画面に表示する |
| 210 | プログラム終了 |

このプログラムは電源投入後の設定のまま、DUT を付けてプログラムを実行します。

【プログラム命令の解説】

[例10-1] のプログラムで使用したOUTPUT, ENTER, REN (または!)命令について説明します。

(1) OUTPUT命令

OUTPUT 装置アドレス; { 数値表現式
 { 文字列表現式 }

OUTPUT命令は、装置アドレス番号で指定された装置へ、数値または文字列で書かれたデータや命令を送り、指定された装置に実行するように指示します。

140 行目に、OUTPUT 31 とありますが、この"31"が装置アドレス番号です。これは、本体の測定部への送示を示しています。本体に、OLDC OFF (IEEE488.2-1987コマンド・モードの設定)を実行させます。このようにOUTPUT命令とGPIBコマンドを使用することで本体を制御することができます。

さらに装置アドレスを変えることで、外部機器も制御できます。

(2) ENTER 命令

ENTER 装置アドレス; { 数値表現式
 { 文字列表現式 }

ENTER 命令は、装置アドレスで指定された装置からGPIBを通してデータを受け取り、数値または文字列としてBASICの変数に入れます。

[例10-1] ではENTER は、応答形式として使用しています。これはOUTPUTと組み合わせて使用します。

```
150 OUTPUT 31;"MARK:ACT 1.380E+6"  
160 OUTPUT 31;"FETC?"  
170 ENTER 31;F,L,P
```

- 150 行目のOUTPUT命令で、装置アドレス31 (本体) に、周波数380MHzの位置に 1番のマーカを出すように設定します。
- 160 行のOUTPUT命令で、同じく本体に"FETC?" と指定しています。この ?は、GPIBコマンドのすぐ後に付けるもので、設定や測定値を知りたい場合 (データのクエリ) に使用します。この場合は、マーカのデータ (周波数、レベル) を聞いています。
- 170 行のENTER 命令でマーカのデータを受け取ります。そして必要なデータを変数 F, L, P に入れています。このFには周波数、Lにはレベル、Pには位相が入ります。

使用するGPIBコマンドによって、送出されるデータの内容は異なります。詳しくは別冊の「プログラミング・マニュアル」を参照して下さい。

(3) REM 命令

REM 命令は、プログラムに注釈行を付けるときに使用します。REM の後にくる文字はすべて注釈文として見なされます。REM 命令は、!(感嘆符) で代用できます。

```
10 REM PROGRAM1  
10 !PROGRAM1 } 同じ意味
```

このプログラムをR3753 で実行すると、[図10-1] のように表示されます。

【実行結果】

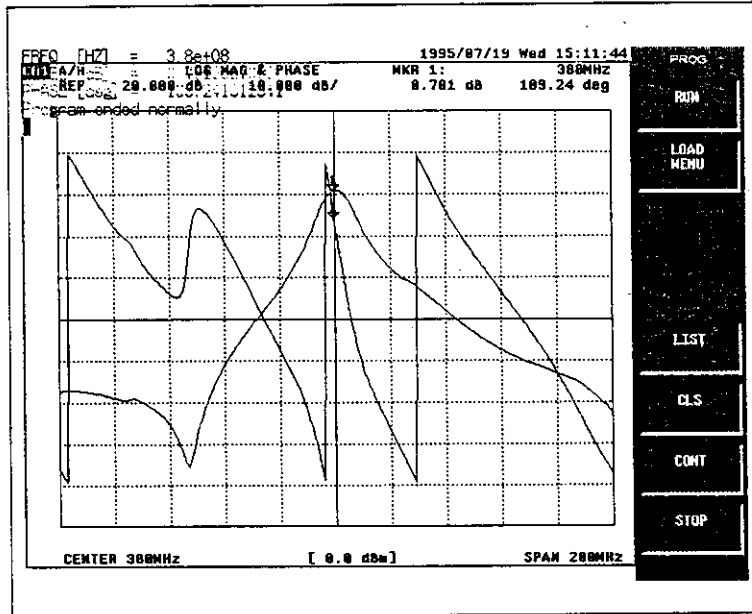


図 10 - 1 実行結果の画面表示

これを見ると、2つの波形が表示されています。これは電源投入後の測定FORMATが、R3753 ではLOG MAG & PHASE の 2トレース表示になっているためです。

中心より少し高いところに▼マークが見えますが、これがマーカで、この位置の周波数が380MHzであることを示しています。表示座標は、グラフの一番上のライン（リファレンス・ライン）を基準にしています。

画面の上にRBF 20.000dBと書かれているので、現在のリファレンス値は20dBです。

そして、グラフ中の 1マスごとの間隔は、図のリファレンス値 REFの右に10.000dBと表示されている通り10dBです。

画面最上部にある

| | |
|----------|------------|
| MKR 1: | 380MHz |
| 0.781 dB | 109.24 deg |

の表示は、マーカが示している380MHzの位置のレベルが0.781dB であることを示しています。

また、レベルの値は掃引ごとに変わり、毎回同じ値とは限りません。このプログラムの実行結果は、図中にも表示していますが、以下ようになります。

| | |
|---------------|---------------|
| FREQ [Hz]= | 3.8e+8 |
| LEVEL [dB]= | 0.7818921033 |
| PHASE [deg] = | 109.241912841 |

FREQ (周波数) の3.8e+8という値は、380,000,000Hz(ヘルツ) であり、LEVEL(レベル) は 0.781dB、PHASE(位相) は109.24deg です。

10.1.2 USING を使用したプログラム

【例10-1】のプログラムをFREQは分かりやすいようにMHz(メガヘルツ) 単位で表示し、LEVEL とPHASE は見やすいように小数点第三位までの表示に直してみます。

例 10 - 3 USINGを使用した表示プログラム

```
100 !*****
110 !*   OUTPUT / ENTER   *
120 !*   (GPRINT & USING) *
130 !*****
140 !
150 OUTPUT 31;"OLDC OFF"
160 OUTPUT 31;"MARK:ACT 1.380E+6"
170 OUTPUT 31;"FETC?"
180 ENTER 31;F,L,P
190 !
200 FR=F/(10^6)
210 !
220 GPRINT "FREQ  [MHz] = ";
230 GPRINT USING "DDDD.DDD";FR
240 GPRINT "LOGMAG [dB] = ";
250 GPRINT USING "MDDD.DDD";L
260 GPRINT "PHASE [deg] = ";
270 GPRINT USING "MDDD.DDD";P
280 STOP
```

このプログラムを実行すると、結果は以下の表示になります。

【実行結果】

```
FREQ [MHz]= 380.000
LEVEL [dB]=  0.781
PHASE [deg]= 109.241
```

【プログラムの解説(GPRINT USING)】

[例10-3] のプログラムの流れを解説します。

| 例10-3の解説 | |
|----------|--|
| 100 | { |
| 110 | コメント行 |
| 140 | |
| 150 | 本体に新コマンド・モードを指示する |
| 160 | 本体の 1番目のマーカを380MHzに置く |
| 170 | 本体の 1番目のマーカ・データを送るように指示する |
| 180 | 170 行で送られてきたデータを変数F, L, P に代入する |
| 190 | コメント行 |
| 200 | F(周波数)をMHz 単位にするため 10^{-6} で除算し、商をFRに代入する |
| 210 | コメント行 |
| 220 | 「FREQ [MHz] =」をプリンタに出力する (改行しない) |
| 230 | FRに代入された値を220 行で出力した直後から、少数部 3桁までをプリンタに出力する (改行する) |
| 240 | 「LOGMAG [dB] =」をプリンタに出力する (改行しない) |
| 250 | L に代入された値を240 行で出力した直後から、少数部 3桁までをプリンタに出力する (改行する) |
| 260 | 「PHASE [deg] =」をプリンタに出力する (改行しない) |
| 270 | P に代入された値を260 行で出力した直後から、少数部 3桁までをプリンタに出力する (改行する) |
| 280 | プログラムを止める |

[例10-3] のプログラムは、[例10-1] のプログラムで使用していない新しい命令があります。220 ~270 行目にあるGPRINT USINGです。

GPRINT USINGとして、プログラム中では使用していますが、GPRINT単独でも使うことができます。

GPRINTは PRINT命令とほぼ同じですが、画面に表示するのではなく GPIBポートにデータを出力します。つまり、GPRINTの後に続く変数や”(ダブルクォーテーション)で囲まれた文字列などを GPIBポートから出力できます。この GPIBポートにプリンタを接続すると、プリンタへデータを出力することができます。

行末に付いている;(セミコロン)は改行なしを示しています。次の出力は改行されず、前の出力に続きます。

書式指定式(PRINT USING/GPRINT USING)

PRINT USING 命令は、書式設定によって決定されるイメージ仕様に従って、文字や数値を出力します。

[例10-3] のプログラムの230 行を見て下さい。この行中にある”DDDD.DDD”;FR は、FRの中に代入された数値を少数部 3桁まで表示し、整数部は 3桁までならば余った桁に空白(スペース)を表示するように指定しています。

また、250 行と270 行の”MDDD.DDD”は、230 行と同じように表示しますが、変数L または Pの値がマイナスならば数値の前に -を付け、プラスならば空白(スペース)を表示するように指定しています。

ここではGPRINT USINGを使っているので、これらの結果はプリンタに出力されます。なお、USING を使うと、自動的に改行コードが加えられます。

10.2 ビルトイン関数

ビルトイン関数は、取り込んだ測定データを本体内蔵のCPUで高速に演算、解析する組み込み関数です。

この関数は、従来のようにGPIBプログラム・コードのOUTPUTとENTERを使ったデータの転送を必要とせず、本体内蔵部のCPUで直接、高速演算をしているので処理にかかる時間が格段に向上します。

R3752では、本体にマーカ解析機能が入っていません。このためR3752で波形データの解析を行う場合は、ビルトイン関数を使ったプログラムを作成しなければなりません。ビルトイン関数は、R3752/53およびR3764/66, R3765/67の全機種で使用できます。

10.2.1 ビルトイン関数の使用

ビルトイン関数は、今まで使ってきた変数と同様に、必要な値を変数に代入する方法を使います。

例えば、ビルトイン関数のCVALUE関数（周波数の値を指定して、その周波数の測定レスポンス値（レベル）を求める関数）の形式は以下の通りです。

CVALUE（指定する周波数、測定しているCHの指定）

例として、CH1に接続したDUT（被測定物）の周波数380MHzのレベルを求めるプログラムを以下に示します。

例 10 - 4 CVALUE 関数を使用したプログラム

```
100 A=3.8e+8  
110 L=CVALUE(A,0)  
120 PRINT L
```

このプログラムでは、まず周波数380MHzを変数Aの中に入れておきます。

次に110行でレベルを求めるのですが、指定する周波数に先ほどのAを入れ、CHの指定はCH1にDUTを接続しているので0とします。そして、レベル値が出たらその値を変数Lに入れておきます。

その後、120行でPRINT命令を使って画面に表示しています。

このようにビルトイン関数は、演算式の中に組み込むことができます。そのため、普通の変数計算と同じように使うことができます。

ビルトイン関数の詳細については、「プログラミング・マニュアル」の[4.4 ビルトイン関数]を参照して下さい。

10.2.2 ビルトイン関数を使用したプログラム

ここでは、複数のビルトイン関数を使ってプログラムを作ります。
[例10-3]のプログラムをビルトイン関数を使って書き換えると、以下のようになります。変更した行は150行～190行で他は同じです。実行結果も同様になります。

例 10 - 5 ビルトイン関数を使用したプログラム

```

100 !*****
110 !*   OUTPUT / ENTER   *
120 !*   (BUILTIN)      *
130 !*****
140 !
150 OUTPUT 31;"OLDC OFF"
160 AP=POINT1(3.8e+8,0)
170 F=FREQ(AP,0)
180 L=VALUE(AP,0)           ! 1st data (CH1)
190 P=VALUE(AP,8)          ! 2nd data (CH1)
200 FR=F/(10^6)
210 !
220 PRINT "FREQ [MHz] = ";
230 PRINT USING "DDDD.DDD";FR
240 PRINT "LOGMAG [dB] = ";
250 PRINT USING "MDDD.DDD";L
260 PRINT "PHASE [deg] = ";
270 PRINT USING "MDDD.DDD";P
280 STOP

```

このプログラムは、[例10-3]のプログラムと同じ380MHzのセラミックBPFをDUTとして使っています。

160行でPOINT1関数を使っています。これは周波数を指定することで、その周波数に最も近い測定周波数がアドレス・ポイントの何ポイント目にあたるかを計算する関数です。(アドレス・ポイントは、測定データの解析範囲の指定や測定データ中の位置指定に使います。0～1200の値で指定します。)

ここでは、最初にPOINT1関数を使用して周波数をアドレス・ポイントに変換しています。POINT1関数の書式は次の通りです。

POINT1 (指定する周波数、解析チャンネル)

ビルトイン関数のほとんどは、CVALUEやPOINT1のような書き方で使います。

170行では求めたアドレス・ポイントを使い、FREQ関数で周波数を求め、変数Fに代入しています。周波数は、380MHzと分かっているので、本来ならば使う必要はありません。ここでは、アドレス・ポイントから周波数値を求める説明をするために使用しています。FREQ関数の書式は次の通りです。

FREQ (アドレス・ポイント、解析チャンネル)

180 行でアドレス・ポイントの入った変数APを使って、VALUE 関数で振幅を求めて変数 Lに代入します。VALUE 関数の形式は次の通りです。

VALUE(アドレス・ポイント、解析チャンネル)

190 行では位相を求めて変数 Pに代入します。

残りの行は、[例10-3] のプログラムと同様に周波数、振幅、位相を画面上に表示 (GPRINT の代わりにPRINT を使用) しています。

このような短いプログラムでは、処理速度の違いは分かりませんが、もっと複雑で長いプログラムをビルトイン関数を使用して作ると、高速に処理できます。

ビルトイン関数、解析チャンネルなどの詳細は、「R3764/66, R3765/67 プログラミング・マニュアル」の[4.4 ビルトイン関数]を参照して下さい。

10.2.3 測定値を判定するプログラム

今までのプログラムは、決まった周波数の値を直接プログラムの中に入れて振幅と位相を求めていました。

ここでは、INPUT 命令を使って、中心周波数(CENTER)とスパン(SPAN)の値を入力し、その周波数範囲における振幅最大点での周波数と振幅を求めるプログラムを作ります。また、このプログラムには振幅最大点での振幅値が一定の基準値に達しているか判定する処理も組み込みます。

【判定プログラムの説明】

初期設定として、必要なCENTER, SPANそして振幅値が何レベル以上ならば合格とするかの基準値をINPUT 命令を使って変数に代入します。

```
INPUT "SPEC [dB] = ", SP
INPUT "CENTER [MHz] = ", C
INPUT "SPAN [MHz] = ", S
```

これを実行したときSPEC値はそのままの値を入力しますが、CENTERとSPANは MHzの単位なので、150MHzの値にしたい場合は 150だけを入力します。

必要な値を入力したら、その値の計測モードのCENTERとSPANを設定します。

```
OUTPUT 31;"OLDC OFF"
OUTPUT 31;"FREQ:CENT ";C;"MAHZ"
OUTPUT 31;"FREQ:SPAN ";S;"MAHZ"
```

OLDC OFFは GPIBの コマンド・モードを新コマンド・モードに設定します。CENTERとSPANの周波数設定は、GPIBプログラム・コードのFREQ:CENT とFREQ:SPAN を使用します。

OLDC ON にすると、R3751 およびR3762/63で使っていたコマンド名を使用できますが、新コマンド・モードの方がプログラムが読みやすくなります。

次にCENTERとSPANの値が入った変数 Cと SをMHz 単位からHz単位に変換しておきます。

```
C1=C*1.0e+6  
S1=S*1.0e+6
```

Hz単位のCENTERの値を変数C1へ、SPANの値を変数S1へ代入します。
ビルトイン関数を使う前に計測モードのSTART およびSTOPの値を求めます。これは、SPAN値を 2で割って、その値とCENTER値を足すとSTOP値、CENTER値からその値を引くとSTART 値になります。

```
S2=S1/2  
P0=C1-S2  
P1=C1+S2
```

START とSTOPを求めたら、ビルトイン関数を使って START～STOP間の最大周波数とその振幅を求めます。

```
A=POINT1(P0,0)  
B=POINT1(P1,0)  
F=FMAX(A,B,0)  
L=MAX(A,B,0)
```

まず、P0とP1をビルトイン関数で使うアドレス・ポイントに変換します。開始アドレス・ポイントを Aに、終了アドレス・ポイントを Bに代入します。

そして、FMAX関数を使用して振幅が最大となるポイントの周波数を求め、MAX 関数で最大振幅値をサーチします。

FMAX (開始アドレス・ポイント、終了アドレス・ポイント、解析チャンネル)
MAX (開始アドレス・ポイント、終了アドレス・ポイント、解析チャンネル)

求めた最大振幅ポイントの周波数値は変数 F、その振幅値は Lに代入します。ビルトイン関数の処理が終了した後、求めた値を PRINT命令で画面上に表示します。

```
FR=F/(10*6)  
!  
PRINT "MAX  FREQ [MHz] = ";  
PRINT USING "DDDD.DDD";FR  
PRINT "MAX  LEVEL [dB] = ";  
PRINT USING "MDDD.DDD";L  
PRINT "SPEC LEVEL      = ";  
PRINT USING "MDDD.DDD";SP
```


最大周波数の値が入った変数 FをHz単位か MHz単位に変更します。
そして、最大周波数、レベル、スペック（基準値）の順に表示します。変数の値を表示するときは小数点以下 3桁までの表示と、桁をそろえるために PRINT USING命令を使います。

最後に最大周波数のレベルと、最初に入力した基準値（スペック）とを比較して、最大周波数がスペックを満たしていれば PRINT命令で SPEC OK!!と表示し、スペック値に満たなければ SPEC NG!!と表示します。

```
IF L<SP THEN GOTO *NG
!
PRINT "*** SPEC OK!! ***"
STOP
!
! 'NG' DISPLAY
!
*NG
PRINT "*** SPEC NG!! ***"
STOP
```

判定プログラム全体を以下に示します。

例 10 - 6 測定値を判定するプログラム

(1/2)

```
100 !*****
110 !* MAX FREQ. AND LEVEL SEARCH *
120 !*      & *
130 !*      JUDGE SPEC *
140 !*      (BY BUILTIN) *
150 !*****
160 !
170 OUTPUT 31;"OLDC OFF"
180 CLS
190 INPUT "SPEC [dB] = ",SP
200 INPUT "CENTER [MHz] = ",C
210 INPUT "SPAN [MHz] = ",S
220 !
230 OUTPUT 31;"FREQ:CENT ";C;"MAHZ"
240 OUTPUT 31;"FREQ:SPAN ";C;"MAHZ"
250 !
260 !
270 C1=C*1E+6
280 S1=S*1E+6
290 !
300 S2=S1/2.0
310 P0=C1-S2
320 P1=C1+S2
330 !
```

(2/2)

```
340 A=POINT1(P0,0)
350 B=POINT1(P1,0)
360 F=FMAX(A,B,0)
370 L=MAX(A,B,0)
380 !
390 FR=F/(10.0^6)
400 OUTPUT 31;"MARK:ACT 1. ";FR
410 OUTPUT 31;"MARK:LET CENT"
420 !
430 !
440 PRINT "MAX FREQ [MHz] = ";
450 PRINT USING "DDDD.DDD";FR
460 PRINT "MAX LEVEL [dB] = ";
470 PRINT USING "MDDD.DDD";L
480 PRINT "SPEC LEVEL [dB] = ";
490 PRINT USING "MDDD.DDD";SP
500 !
510 IF L<SP THEN GOTO *NG
520 !
530 PRINT "*** SPEC OK !! ***"
540 STOP
550 !
560 ! 'NG' DISPLAY
570 !
580 *NG
590 PRINT "*** SPEC NG !! ***"
600 STOP
```

[例10-6] を実行すると、最初にSPECの値を問い合わせてきます。
ここでは、DUT に380MHzのセラミック・フィルタを使用して、スペック・レベル(SPEC)
-10dB, CENTER 380MHz, SPAN 200MHz で測定します。

[例10-6] を実行すると、結果は以下のようになります。

【実行結果】

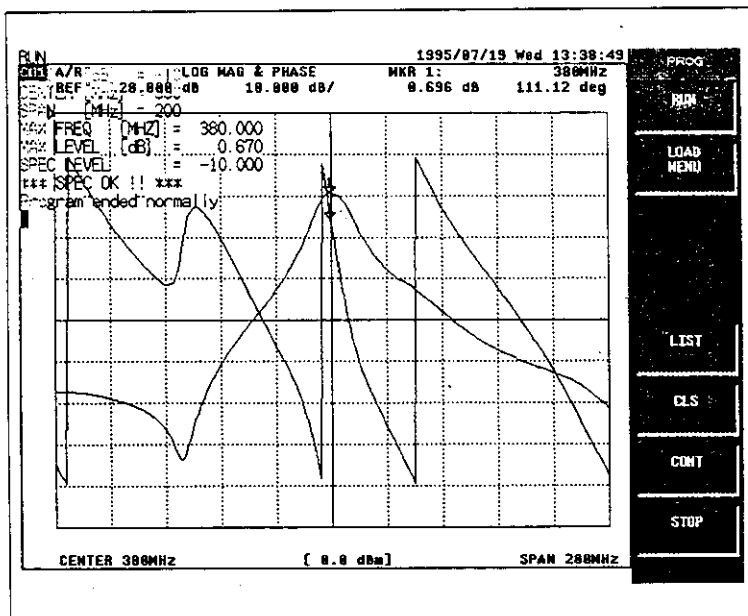


図 10 - 2 判定プログラムの実行結果

10.2.4 パラレルI/O ポートに出力する

[例10-6] のプログラムでは、判定した結果を PRINT命令で「OK!!」または「NG!!」と表示しましたが、ここではパラレルI/O ポートを使用して出力します。パラレルI/O ポートのINPUT1（外部入力）を使用して、トリガ・スイッチによって測定を開始させるプログラムを作成します。

回路例として、本体取扱説明書「パラレルI/O のトリガ・スイッチによってプログラムを動作させる場合」の回路図を使用します。

（各機種の取扱説明書の「機能説明・周辺機器との通信」を参照して下さい。）

【プログラムの作成】

① [例10-6] のプログラムの170行と180行の間にポートのモード設定を追加します。

```
171 OUTPUT 36;16
172 OUTPUT 35;80
173 OUTPUT 35;112
```

- 171行では、パラレルI/Oポートを使用するため、ポートのモード設定を行います。ここでは、A、B、C、Dポートすべてを出力ポートに設定しています。
- 172行と173行では、OUTPUT1 および OUTPUT2をリセットします。つまり、測定中のLED(OUTPUT1)と測定待ちのLED(OUTPUT2)を消灯します。

- ② 測定開始待ちとして、スイッチが押されるまでプログラムを待機させるプログラムを260行と270行の間に追加します。

```
261 OUTPUT 35:48
262 ENTER 34:A
263 WAIT 500
264 IF A<>1 THEN GOTO 262
265 OUTPUT 35:112
```

- 261行では OUTPUT2をセット状態にして LEDを点灯させます。これは測定待ち (READY) を示し、スイッチの入力を待っていることを LEDで表示します。
 - 262行と264行は、スイッチの入力があるまでループし続けてプログラムのシーケンスを止めています。スイッチを押さずにいると、262行の数値変数 Aには何も入力されず、0のまま273行へ進みます。
 - 263行のWAITは、スイッチの入力が正確に行われるように少し間をとっています。(WAIT 時間: msec; 0~65535)
 - 264行は、スイッチが押されていない時変数 Aは 0なので、条件式は成立し、スイッチが押されるまで繰り返されます。
スイッチが押されると262行のENTER 34:A (外部入力) の指定によって数値変数 Aに 1が代入されます。これによって条件式は成立しなくなり、265行へ進みます。
 - 265行では OUTPUT2の LEDを消灯します。これはスイッチが入力されて、測定開始待ち (READY) の状態でなくなったことを示しています。
また、スイッチを押すと OUTPUT1および OUTPUT2の LEDを点灯します。OUTPUT2の LEDは265行で消灯しますが、OUTPUT1は指定がないため点灯したままとなります。これは測定を行っているという合図で、測定中 (MEAS)を示しています。
- ③ 測定が終了した時にLEDを消灯するプログラムを、430行と440行の間に追加します。

```
431 OUTPUT 35:80
```

- 431行は OUTPUT1の LEDをリセット状態にし、LEDを消灯します。
- ④ スペックの値で判定したときのI/Oポートの出力ですが、ここでは簡単に、OKならば Aポートに出力し、NGなら Bポートに出力することにします。
OK出力を530行と540行の間に、NG出力を590行と600行の間に追加します。

```
531 OUTPUT 33:1
```

```
591 OUTPUT 34:1
```

- 判定がOKの場合は Aポートに 1を、NGの場合は Bポートに 1を出力します。
このOKおよびNG出力は、パラレルI/O コネクタ（各機種取扱説明書の「機能説明・周辺機器との通信」を参照）のNo.5(Aポートの1)とNO.13 (Bポートの1)に LEDを付けると、どちらかが点灯するかで結果が見えます。

[例10-7] にプログラム・リストを示します。

例 10 - 7 測定値を判定するプログラム (パラレルI/O ポート使用) (1/2)

```

100 !*****
110 !* MAX FREQ. AND LEVEL SEARCH *
120 !*           &                *
130 !*           JUDGE SPEC        *
140 !*           (BY BUILTIN)      *
150 !*****
160 !
170 OUTPUT 31;"OLDC OFF"
171 OUTPUT 36;16                ! A, B, C, D -> OUTPUT
172 OUTPUT 35;80                ! RESET OUTPUT1
173 OUTPUT 35;112              ! RESET OUTPUT2
180 CLS
190 INPUT "SPEC [dB] = ", SP
200 INPUT "CENTER [MHz] = ", C
210 INPUT "SPAN [MHz] = ", S
220 !
230 OUTPUT 31;"FREQ:CENT ";C;"MAHZ"
240 OUTPUT 31;"FREQ:SPAN ";C;"MAHZ"
250 !
260 !
261 OUTPUT 35;48                ! SET OUTPUT1 AND OUTPUT2
262 ENTER 34;A                 ! READ OUTPUT1 (INPUT1)
263 WAIT 500                   ! WAIT 500MSEC
264 IF A<>1 THEN GOTO 262      ! CHECK TRIGGER SWITCH INPUT
265 OUTPUT 35;112              ! RESET OUTPUT2
270 C1=C*1E+6
280 S1=S*1E+6
290 !
300 S2=S1/2.0
310 P0=C1-S2
320 P1=C1+S2
330 !
340 A=POINT1(P0,0)
350 B=POINT1(P1,0)
360 F=FMAX(A,B,0)
370 L=MAX(A,B,0)
380 !
390 FR=F/(10.0^6)
400 OUTPUT 31;"MARK:ACT 1. ";FR
410 OUTPUT 31;"MARK:LET CENT"

```

(2/2)

```
420 !
430 !
431 OUTPUT 35;80                ! RESET OUTPUT1
440 PRINT "MAX FREQ [MHz] = ";
450 PRINT USING "DDDD.DDD";FR
460 PRINT "MAX LEVEL [dB] = ";
470 PRINT USING "MDDD.DDD";L
480 PRINT "SPEC LEVEL [dB] = ";
490 PRINT USING "MDDD.DDD";SP
500 !
510 IF L<SP THEN GOTO *NG
520 !
530 PRINT "*** SPEC OK !! ***"
531 OUTPUT 33;1                ! WRITE 1 TO PORT-A
540 STOP
550 !
560 ! 'NG' DISPLAY
570 !
580 *NG
590 PRINT "*** SPEC NG !! ***"
591 OUTPUT 34;1                ! WRITE 1 TO PORT-B
600 STOP
```

11. 波形解析プログラム例

この章では、波形解析プログラム例を、ビルトイン関数の使い方を交えながら説明します。

(注) この章で示すプログラムは、R3752/53用の例です。R3765/67, R3764/66 上で使用するには、初期設定状態や周波数範囲などの違いに応じて、変更を要する場合があります。

11.1 MAX およびMIN レベル自動測定プログラム

ビルトイン関数MAX, MINを使って、最大および最小レベル値を自動測定するプログラム例を [例11-1] に示します。

例 11 - 1 MAXおよびMIN レベルの自動測定プログラム

(1/2)

```

1000 !*****
1010 !
1020 !      MAX-MIN LEVEL MEASUREMENT
1030 !
1040 !*****
1050 *MAIN
1060     GOSUB *SETUP
1070     GOSUB *CAL
1080     CLS
1090     *MEAS_LOOP
1100         GOSUB *MEAS
1110         GOSUB *RESULTS
1120         GOTO *MEAS_LOOP
1130 !
1140 !-----
1150 *SETUP
1160     OUTPUT 31;"OLDC OFF"
1170     OUTPUT 31;"DISP:ACT 1;:FUNC1:POW AR;:CALC:FORM MLOP"
1180     OUTPUT 31;"DISP:Y:PDIV 10"
1190     OUTPUT 31;"DISP:Y:RPOS 10"
1200     OUTPUT 31;"DISP:Y:RLEV 0"
1210     OUTPUT 31;"POW ODBM"
1220     !
1230     OUTPUT 31;"SWE:POIN 201"
1240     OUTPUT 31;"FREQ:STAR 100MAHZ"
1250     OUTPUT 31;"FREQ:STOP 200MAHZ"
1260     RETURN
1270 !
1280 !-----
1290 *CAL
1300     CURSOR 6,9
1310     PRINT "CONNECT [THROUGH]"
1320     CURSOR 6,10
1330     INPUT "IF OK THEN PRESS 'ENT' or 'X1'",D$
1340     OUTPUT 31;"CORR:COLL NORM;*OPC?"
1350     ENTER 31;A

```

(2/2)

```

1360     RETURN
1370 !
1380 !-----
1390 *MEAS
1400     CURSOR 5,10
1410     PRINT "CONNECT DUT"
1420     CURSOR 5,11
1430     INPUT "IF OK THEN PRESS 'ENT' or 'X1'",D$
1440     !
1450     MAX_DT=MAX(0,1200,0)
1460     MIN_DT=MIN(0,1200,0)
1470     RETURN
1480 !
1490 !-----
1500 *RESULTS
1510     CLS
1520     CURSOR 5,15
1530     PRINT "MAX VALUE [dB] = ";
1540     PRINT USING "3D.3D";MAX_DT
1550     CURSOR 5,16
1560     PRINT "MIN VALUE [dB] = ";
1570     PRINT USING "3D.3D";MIN_DT
1580     RETURN

```

このプログラムで使用する MAX関数は最大レスポンス値をサーチし、MIN 関数は最小レスポンス値をサーチします。

これらの関数は、共振点や反共振点のレスポンス値を求める場合などにも使用します。以下に、[例11-1] のプログラムの解説をします。

(1/2)

例11-1のプログラムの解説

| | | |
|------|---|-----------------------------|
| 1000 | { | コメント行 |
| 1040 | | |
| 1050 | | メイン・ルーチンのラベルMAIN |
| 1060 | | 初期設定ルーチンSETUP を呼び出す |
| 1070 | | キャリブレーション・ルーチンCAL を呼び出す |
| 1080 | | 画面をクリアする |
| 1090 | | 測定繰り返しループのラベルMEAS_LOOP |
| 1100 | | 測定ルーチンMEASを呼び出す |
| 1110 | | 表示ルーチンRESULTを呼び出す |
| 1120 | | 測定のループ |
| 1130 | | |
| 1130 | } | コメント行 |
| 1140 | | |
| 1150 | | 初期設定ルーチンのラベルSETUP |
| 1160 | | IEEE488.1-1987コマンド・モードを解除する |

例11-1のプログラムの解説

| | |
|------|--|
| 1170 | アクティブ・チャンネルをCH1 に、入力ポートをA/R に、フォーマットをLOGMAG & PHASEに設定する |
| 1180 | スケール分解能を10dBに設定する |
| 1190 | リファレンス・ポジションを10% に設定する |
| 1200 | リファレンス・レベルを0dB に設定する |
| 1210 | 出力レベルを0dBmに設定する |
| 1220 | コメント行 |
| 1230 | 測定ポイント数を 201ポイントに設定する |
| 1240 | 掃引開始周波数を100MHzに設定する |
| 1250 | 掃引終了周波数を200MHzに設定する |
| 1260 | 初期設定ルーチンを抜ける |
| 1270 | ＼ |
| | コメント行 |
| 1280 | |
| 1290 | キャリブレーション・ルーチンのラベルCAL |
| 1300 | カーソルを移動する |
| 1310 | メッセージ"CONNECT[THROUGH]"を表示する |
| 1320 | カーソルを移動する |
| 1330 | メッセージ"IF OK THEN PRESS 'ENT' or 'X1'"を表示して、入力を待つ |
| 1340 | キャリブレーションを実行する |
| 1350 | キャリブレーションが終了するまで待つ |
| 1360 | キャリブレーション・ルーチンを抜ける |
| 1370 | ＼ |
| | コメント行 |
| 1380 | |
| 1390 | 測定ルーチンのラベルMEAS |
| 1400 | カーソルを移動する |
| 1410 | メッセージ"CONNECT DUT" を表示する |
| 1420 | カーソルを移動する |
| 1430 | メッセージ"IF OK THEN PRESS 'ENT' or 'X1'"を表示して、入力を待つ |
| 1440 | コメント行 |
| 1450 | 最大値を取得する |
| 1460 | 最小値を取得する |
| 1470 | 測定ルーチンを抜ける |
| 1480 | ＼ |
| | コメント行 |
| 1490 | |
| 1500 | 表示ルーチンのラベルRESULTS |
| 1510 | 画面をクリアする |
| 1520 | ＼ |
| | カーソルを移動して最大値を表示する |
| 1540 | |
| 1550 | ＼ |
| | カーソルを移動して最小値を表示する |
| 1570 | |
| 1580 | 表示ルーチンを抜ける |

11.2 セラミック・フィルタ自動測定プログラム

セラミック・フィルタのインサージョン・ロスおよび 3dB幅の周波数値を求めるプログラム例を [例11-2] に示します。

例 11 - 2 セラミック・フィルタ自動測定プログラム

(1/2)

```
1000 !*****
1010 !
1020 !     CERAMIC FILTER MEASUREMENT
1030 !
1040 !*****
1050 *MAIN
1060     GOSUB *SETUP
1070     GOSUB *CAL
1080     CLS
1090     *MEAS_LOOP
1100         GOSUB *MEAS
1110         GOSUB *RESULTS
1120         GOTO *MEAS_LOOP
1130 !
1140 *SETUP
1150     NA=31
1160     OUTPUT NA;"OLDC OFF"
1170     OUTPUT NA;"SYST:PRES;:INIT:CONT OFF;:STAT:OPER:ENAB 8;*ESB 128;*OPC?"
1180     ENTER NA;A
1190     OUTPUT NA;"FREQ:SPAN 10KHZ"
1200     OUTPUT NA;"FREQ:CENT 10.7MAHZ"
1210     SPOLL(NA)
1220     RETURN
1230 !
1240 *CAL
1250     CURSOR 6.9
1260     PRINT "CONNECT [THROUGH]"
1270     CURSOR 6.10
1280     INPUT "IF OK THEN PRESS 'ENT' or 'X1'",D$
1290     OUTPUT NA;"CORR:COLL NORM;*OPC?"
1300     ENTER NA;A
1310     RETURN
1320 !
1330 *MEAS
1340     CURSOR 6.25
1350     PRINT "CONNECT DUT"
1360     CURSOR 6.26
1370     INPUT "IF OK THEN PRESS 'ENT' or 'X1'",D$
1380 !
1390     ON ISRQ GOTO *LPOUT
1400     ENABLE INTR
1410     OUTPUT NA;"INIT"
1420     *LP
1430     GOTO *LP
```

(2/2)

```

1440 !
1450 *LPOUT
1460     SPOLL(NA)
1470     DISABLE INTR
1480     I_LOSS=MAX(0, 1200, 0)
1490     MAX_F=FMAX(0, 1200, 0)
1500     BW3DB=CBND(MAX_F, 3, 0)
1510     RETURN
1520 !
1530 *RESULTS
1540     CURSOR 5, 4
1550     PRINT "I LOSS [dB]           = ";
1560     PRINT USING "3D. 3D"; I_LOSS
1570     CURSOR 5, 5
1580     PRINT "3 DB BAND WIDTH [MHz] = ";
1590     PRINT USING "3D. 3D"; BW3DB/1E+6
1600     RETURN
    
```

【例11-2】のプログラムで使用するCBND関数は、帯域幅を求める関数です。指定周波数から、指定された減衰レベルだけ減衰したポイントをサーチし、帯域幅を求めます。サーチは指定されたアドレス・ポイントから外側に行います。

このプログラムでは、FMAX関数で最大レスポンスの周波数を求め、CBND関数でその周波数から 3dBダウンの帯域幅を求めています。

以下に、【例11-2】のプログラムを解説します。

(1/2)

例11-2のプログラムの解説

| | |
|------|---|
| 1000 | コメント行 |
| 1040 | メイン・ルーチンのラベルMAIN |
| 1050 | 初期設定ルーチンSETUP を呼び出す |
| 1060 | キャリブレーション・ルーチンCAL を呼び出す |
| 1070 | 画面をクリアする |
| 1080 | 測定繰り返しループのラベルMEAS_LOOP |
| 1090 | 測定ルーチンMEASを呼び出す |
| 1100 | 表示ルーチンRESULTを呼び出す |
| 1110 | 測定のループ |
| 1120 | コメント行 |
| 1130 | 初期設定ルーチンのラベルSETUP |
| 1140 | IEEE488.1-1987コマンド・モードを解除する |
| 1150 | 本体をプリセットした後、シングル掃引モードにし、掃引終了でSRQ 要求を発生するように設定する |
| 1160 | |
| 1170 | |

例11-2のプログラムの解説

```

1180 設定が完了するまで待つ
1190 掃引中心周波数を10kHz に設定する
1200 掃引周波数スパンを10.7MHz に設定する
1210 シリアル・ポールを行い、RSV ビットを落とす
1220 初期設定ルーチンを抜ける
1230 コメント行
1240 キャリブレーション・ルーチンのラベルCAL
1250 カーソルを移動する
1260 メッセージ"CONNECT[THROUGH]"を表示する
1270 カーソルを移動する
1280 メッセージ"IF OK THEN PRESS 'ENT' or 'X1'"を表示して、入力を待つ
1290 キャリブレーションを実行する
1300 キャリブレーションが終了するまで待つ
1310 キャリブレーション・ルーチンを抜ける
1320 コメント行
1330 測定ルーチンのラベルMEAS
1340 カーソルを移動する
1350 メッセージ"CONNECT DUT" を表示する
1360 カーソルを移動する
1370 メッセージ"IF OK THEN PRESS 'ENT' or 'X1'"を表示して、入力を待つ
1380 コメント行
1390 サービス要求割り込みの分岐先を指定する
1400 割り込みを許可する
1410 掃引を 1回実行する
1420
1430 { 割り込み待ちループを繰り返す
1440
1450 コメント行
1460 サービス要求割り込みの分岐先ラベルLPOUT
1470 シリアル・ポールを行い、RSV ビットを落とす
1480 割り込みを禁止する
1490 MAX 関数でレベルの最大値を求めて、変数 I_LOSSに代入する
1500 FMAX関数でレベルが最大となる測定周波数を求めて、変数 MAX_F に代入する
1510 CBAND 関数で3dB の帯域幅を求め、変数 BW3DBに代入する
1520 割り込み処理ルーチンを抜ける
1530 コメント行
1540 表示ルーチンのラベルRESULTS
1550
1560 { カーソルを移動してインサクション・ロス値を表示する
1570
1580 { カーソルを移動して3dB 帯域幅の周波数を表示する
1590
1600 表示ルーチンを抜ける

```

11.3 リップル解析プログラム

[例11-3] にリップル関数を使ったプログラム例を示します。

例 11 - 3 リップル解析プログラム

(1/2)

```

1000 !*****
1010 !
1020 !     RIPPLE MEASUREMENT
1030 !     (NO USED SRQ)
1040 !
1050 !*****
1060 DIM PR1$(25), PR2$(25), PR3$(25)
1070 !
1080 *MAIN
1090     GOSUB *SETUP
1100     CLS
1110     *MEAS_LOOP
1120         GOSUB *MEAS
1130         GOSUB *RESULTS
1140         GOTO *MEAS_LOOP
1150 !
1160 *SETUP
1170     NA=31
1180     OUTPUT NA;"OLDC OFF"
1190     OUTPUT NA;"SYST:PRES;:INIT:CONT OFF"
1200     OUTPUT NA;"DISP:FORM ULOW"
1210     OUTPUT NA;"CALC:FORM MLOD"
1220     OUTPUT NA;"FREQ:CENT 17.9MAHZ;SPAN 30KHZ"
1230     OUTPUT NA;"BAND 1KHZ"
1240     OUTPUT NA;"SWE:TIME 1SEC"
1250     RETURN
1260 !
1270 *MEAS
1280     CURSOR 6.25
1290     PRINT "CONNECT DUT"
1300     CURSOR 6.26
1310     INPUT "IF OK THEN PRESS 'ENT' or 'X1'", D$
1320 !
1330     OUTPUT NA;"INIT;*OPC?"
1340     ENTER NA;DUMMYS
1350     OUTPUT NA;"DISP:Y8 AUTO"
1360 !
1370     A1=PMAX(0,1200,0)
1380     A2=BNDL(A1,3,0)
1390     A3=BNDH(A1,3,0)
1400     A4=POINT2(A2,0)
1410     A5=POINT2(A3,0)

```

(2/2)

```
1420 B1=RPL2(A4, A5, 1, 0.001, 0) ! LOGMAG RIPPLE
1430 B2=RPL4(A4, A5, 1, 0.001, 0)
1440 IF B1<B2 THEN
1450 B3=B2
1460 ELSE
1470 B3=B1
1480 END IF
1490 C1=RPL2(A4, A5, 1, 1e-08, 8) ! DELAY RIPPLE
1500 C2=RPL4(A4, A5, 1, 1e-08, 8)
1510 IF C1<C2 THEN
1520 C3=C2
1530 ELSE
1540 C3=C1
1550 END IF
1560 RETURN
1570 !
1580 *RESULTS
1590 PR1$="LOGMAG RIPPLE [dB] ="
1600 CURSOR 0, 16:PRINT USING "k, M2D.5D";PR1$, B3
1610 PR2$="DELAY RIPPLE [us] ="
1620 CURSOR 0, 17:PRINT USING "k, M2D.5D";PR2$, C3*10^6
1630 RETURN
```

このプログラムでは、最初にP_{MAX}、B_{NDL}、B_{NDH}で解析する周波数範囲を求めます。P_{MAX}関数で最大レスポンスの測定ポイントを求めて、その測定ポイントから帯域幅を計算します。B_{NDL}関数で帯域幅の低周波数側の周波数を、B_{NDH}関数で帯域幅の高周波数側の周波数を求め、この周波数範囲でリップル解析を指定します。

次に、POINT2関数でアドレス・ポイントに変換した後、リップル解析を行います。リップル解析関数はいろいろありますが、このプログラムではRPL2とRPL4を使います。両方とも隣接する極大値と極小値の最大値を求める関数ですが、極大、極小のペアの取り方が異なります。RPL2関数では極大値の方が低周波数側になり、RPL4関数では極大値の方が高周波数側になります。

以下に、[例11-3]のプログラムを解説します。

(1/2)

例11-3のプログラムの解説

| | |
|------|--|
| 1000 | コメント行 |
| 1050 | |
| 1060 | 文字列配列を定義する |
| 1070 | コメント行 |
| 1080 | メイン・ルーチンのラベルMAIN |
| 1090 | 初期設定ルーチンSETUPを呼び出す |
| 1100 | 画面をクリアする |
| 1110 | 測定繰り返しループのラベルMEAS_LOOP |
| 1120 | 測定ルーチンMEASを呼び出す |
| 1130 | 表示ルーチンRESULTを呼び出す |
| 1140 | 測定のループ |
| 1150 | コメント行 |
| 1160 | 初期設定ルーチンのラベルSETUP |
| 1170 | 変数NAにアドレス31を代入する |
| 1180 | IBEE488.1-1987コマンド・モードを解除する |
| 1190 | 本体をプリセットし、シングル掃引モードにする |
| 1200 | 分割画面モードを設定する |
| 1210 | 計算フォーマットをLOGMAG & DELAYに設定する |
| 1220 | 掃引中心周波数を17.9MHz、スパンを30kHzに設定する |
| 1230 | リゾリューション・バンド幅を1kHzに設定する |
| 1240 | 掃引時間を1秒に設定する |
| 1250 | 初期設定ルーチンを抜ける |
| 1260 | コメント行 |
| 1270 | 測定ルーチンのラベルMEAS |
| 1280 | カーソルを移動する |
| 1290 | メッセージ"CONNECT DUT"を表示する |
| 1300 | カーソルを移動する |
| 1310 | メッセージ"IF OK THEN PRESS 'ENT' or 'X1'"を表示して、入力を待つ |
| 1320 | コメント行 |
| 1330 | 掃引を1回実行し、OPCクエリを送る |
| 1340 | 掃引が終了するまで待つ |
| 1350 | Y軸の自動設定を行う |
| 1360 | コメント行 |
| 1370 | レベルが最大となる測定ポイントを求め、変数A1へ代入する |
| 1380 | 3dB帯域幅の低周波数側の周波数を求め、変数A2へ代入する |
| 1390 | 3dB帯域幅の高周波数側の周波数を求め、変数A3へ代入する |
| 1400 | 周波数A2をアドレス・ポイントに変換し、A4へ代入する |
| 1410 | 周波数A3をアドレス・ポイントに変換し、A5へ代入する |
| 1420 | RPL2関数で、振幅データ中から隣接する極大値と極小値の最大値を求める |
| 1430 | RPL4関数で、振幅データ中から隣接する極大値と極小値の最大値を求める |
| 1440 | |
| 1480 | この中で大きい値を変数B3へ代入する |

(2/2)

| 例11-3のプログラムの解説 | |
|----------------|-------------------------------------|
| 1490 | RPL2関数で、遅延データ中から隣接する極大値と極小値の最大値を求める |
| 1500 | RPL4関数で、遅延データ中から隣接する極大値と極小値の最大値を求める |
| 1510 | { |
| | この中で大きい値を変数C3へ代入する |
| 1550 | |
| 1560 | 測定ルーチンを抜ける |
| 1570 | コメント行 |
| 1580 | 表示ルーチンのラベルRESULTS |
| 1590 | { |
| | 振幅データのリップル解析値を表示する |
| 1600 | |
| 1610 | { |
| | 遅延データのリップル解析値を表示する |
| 1620 | |
| 1630 | 表示ルーチンを抜ける |

11.4 バンドパス・フィルタの測定例

ここでは、中心周波数 10.7MHzのバンドパス・フィルタの測定を例に、フィルタ解析プログラムを説明します。[例11-4]にプログラム例を示します。

例 11 - 4 バンドパス・フィルタの測定

(1/2)

```

1000 !*****
1010 !
1020 !      BAND PASS FILTER ANALYSIS
1030 !      f=10.7MHz
1040 !
1050 ! FILE:BPF.BAS
1060 !*****
1070 *MAIN
1080     GOSUB *SETUP
1090     GOSUB *CAL
1100     CLS
1110     *MEAS_LOOP
1120         GOSUB *MEAS
1130         GOSUB *RESULTS
1140         GOTO *MEAS_LOOP
1150 !
1160 *SETUP
1170     INTEGER EV
1180     DIM L(2),F(2,4)
1190     NA=31 :EV=1 :L(1)=3.0 :L(2)=60.0
1200     OUTPUT NA;"OLDC OFF"
1210     OUTPUT NA;"SYST:PRES;:INIT:CONT.OFF;:STAT:OPER:ENAB 8;*SRE 128;*OPC?"
1220     ENTER NA;A
1230     OUTPUT NA;"CALC:FORM MLOG"
1240     OUTPUT NA;"FREQ:SPAN 2MHZ;CENT 10.7MAHZ"
1250     RETURN
1260 !
1270 *CAL
1280     CURSOR 6,9 :PRINT "CONNECT [THROUGH]"
1290     CURSOR 6,10 :INPUT "IF OK THEN PRESS 'ENT' or 'X1'",D$
1300     OUTPUT NA;"CORR:COLL NORM;*OPC?" :ENTER NA;A
1310     RETURN
1320 !
1330 *MEAS
1340     CURSOR 6,25 :PRINT "CONNECT DUT"
1350     CURSOR 6,26 :INPUT "IF OK THEN PRESS 'ENT' or 'X1'",D$
1360     OUTPUT NA;"INIT" :WAIT EVENT EV
1370     AP=PMAX(0,1200,0)
1380     NP=MBNDI(0,1200,AP,2,L(1),F(1,1),0)
1390     QF=F(1,3)/F(1,4)                ! QF = CF(3dB) / BW(3dB)
1400     SF=F(2,4)/F(1,4)                ! SF = BW'(60dB) / BW(3dB)
1410     RETURN

```

(2/2)

```

1420 !
1430 *RESULTS
1440 CURSOR 5.4 :PRINT "C. F [MHz] = "; :PRINT USING "3D.7D";F(1,3)/1.0E+6
1450 CURSOR 5.5 :PRINT "L. F [MHz] = "; :PRINT USING "3D.7D";F(1,1)/1.0E+6
1460 CURSOR 5.6 :PRINT "R. F [MHz] = "; :PRINT USING "3D.7D";F(1,2)/1.0E+6
1470 CURSOR 5.7 :PRINT " BW [ Hz] = "; :PRINT USING "5D.1D";F(1,4)
1480 CURSOR 5.8 :PRINT " Q      = "; :PRINT USING ".5D";QF
1490 CURSOR 5.9 :PRINT " SF      = "; :PRINT USING ".5D";SF
1500 RETURN
    
```

このプログラムを実行すると、本体の画面表示が [図11-1] のように変わります。

【実行結果】

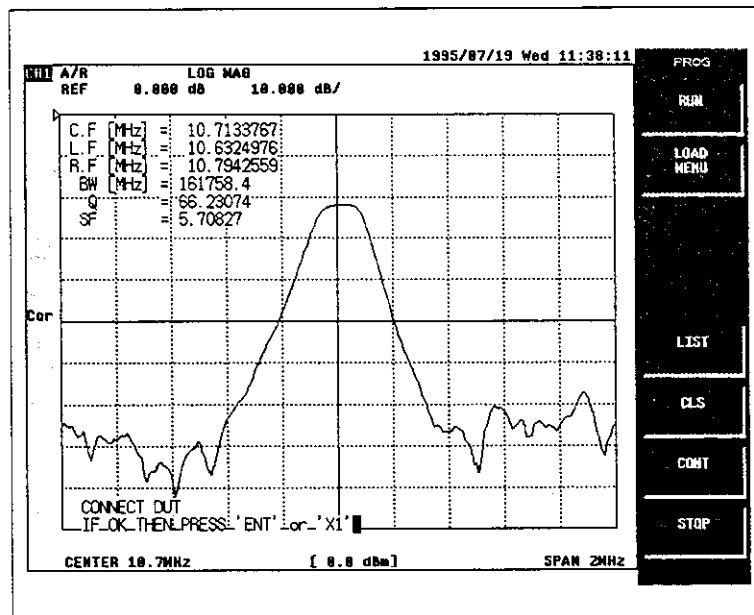


図 11 - 1 実行結果の画面表示 (バンドパス・フィルタの測定)

PMAX関数で最大レスポンス値の測定ポイントを見つけて、MBND関数により減衰レベル 3dB と 60dB の 2つの測定ポイントの帯域幅や周波数を求めます。

MBND関数を利用すると、複数の減衰レベルの解析を一度に行うことができ、それぞれについて、帯域幅の低周波数側の周波数、高周波数側の周波数、中心周波数、帯域を得ることができます。

11.5 クリスタル共振点の測定例

ここでは、セラミック発振子($f = 45.1\text{MHz}$)の共振点および反共振点を伝送測定により求めるプログラムについて説明します。[例11-5]にプログラム例を示します。

例 11 - 5 クリスタル共振点の測定

(1/2)

```

1000 !*****
1010 !
1020 !     SEARCH RESONANCE POINT
1030 !     f=45.1MHz
1040 !
1050 ! FILE:RESONA. BAS
1060 !*****
1070 *MAIN
1080     GOSUB *SETUP
1090     GOSUB *CAL
1100     CLS
1110     *MEAS_LOOP
1120         GOSUB *MEAS
1130         GOSUB *RESULTS
1140         GOTO *MEAS_LOOP
1150 !
1160 *SETUP
1170     INTEGER EV
1180     NA=31 :EV=1
1190     OUTPUT NA;"OLDC OFF"
1200     OUTPUT NA;"SYST:PRES;:INIT:CONT OFF;:STAT:OPER:ENAB 8;*SRE 128;*OPC?"
1210     ENTER NA;A
1220     OUTPUT NA;"FREQ:SPAN 1MAHZ;CENT 45.1MAHZ"
1230     OUTPUT NA;"BAND 1KHZ"
1240     OUTPUT NA;"CALC:TRAN:IMP:CIMP 12.5;TYPE ZTR"
1250     RETURN
1260 !
1270 *CAL
1280     CURSOR 6,9 :PRINT "CONNECT [THROUGH]"
1290     CURSOR 6,10 :INPUT "IF OK THEN PRESS 'ENT' or 'X1'",D$
1300     OUTPUT NA;"CORR:COLL NORM;*OPC?" :ENTER NA;A
1310     RETURN
1320 !
1330 *MEAS
1340     CURSOR 6,25 :PRINT "CONNECT DUT"
1350     CURSOR 6,26 :INPUT "IF OK THEN PRESS 'ENT' or 'X1'",D$
1360     OUTPUT NA;"INIT" :WAIT EVENT EV
1370     FR1=FMAX(0,1200,0) :AP1=POINT1(FR1,0)
1380     FR2=FMIN(0,1200,0) :AP2=POINT1(FR2,0)
1390     FS1=ZEROPHS(AP1-60,AP1+60,8) :LV1=VALUE(AP1,0) :PH1=VALUE(AP1,8)
1400     FS2=ZEROPHS(AP2-60,AP2+60,8) :LV2=VALUE(AP2,0) :PH2=VALUE(AP2,8)
1410     RETURN

```

(2/2)

```

1420 !
1430 *RESULTS
1440 CURSOR 5.4 :PRINT "RESONANCE FR1 [MHz] = "; :PRINT USING "3D.7D";FR1/1.0E+6
1450 CURSOR 5.5 :PRINT "          FS1 [MHz] = "; :PRINT USING "3D.7D";FS1/1.0E+6
1460 CURSOR 5.6 :PRINT "          LEVEL [dB] = "; :PRINT USING "3D.3D";LV1
1470 CURSOR 5.7 :PRINT "          PHASE [deg] = "; :PRINT USING "3D.7D";PH1
1480 CURSOR 5.8 :PRINT "ANTI-RES FR2 [MHz] = "; :PRINT USING "3D.7D";FR2/1.0E+6
1490 CURSOR 5.9 :PRINT "          FS2 [MHz] = "; :PRINT USING "3D.7D";FS2/1.0E+6
1500 CURSOR 5.10:PRINT "          LEVEL [dB] = "; :PRINT USING "3D.3D";LV2
1510 CURSOR 5.11:PRINT "          PHASE [deg] = "; :PRINT USING "3D.7D";PH2
1520 RETURN
    
```

このプログラムを実行すると、本体の画面表示が [図11-2] のように変わります。
なお、ここではセットアップに π 回路治具 (PIC-001) を使用します。

【実行結果】

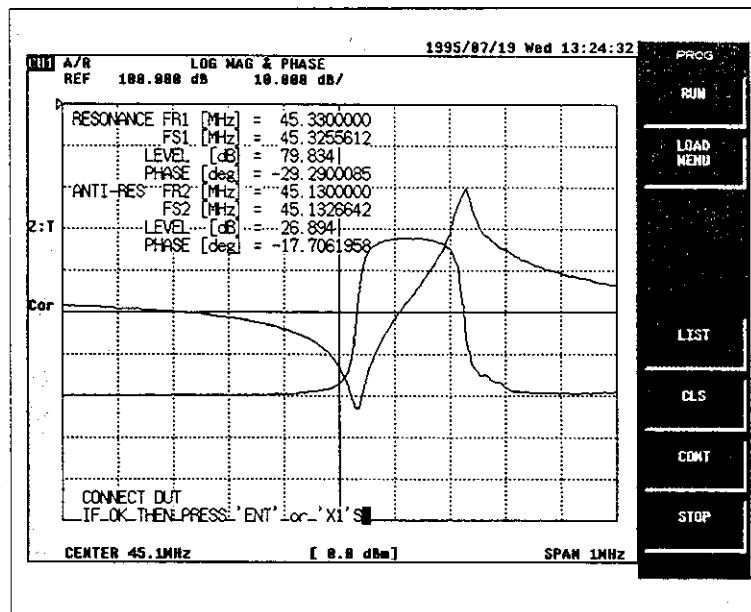


図 11 - 2 実行結果の画面表示(クリスタル共振点の測定)

共振点および反共振点を求める方法には、以下の2通りがあります。

- 最大レベル値および最小レベル値でサーチする方法
- ゼロ位相でサーチする方法

ここでは、最初に FMAX, FMIN 関数で最大レベル値と最小レベル値の測定ポイントをサーチします。

次に、ZEROPHS 関数を使ってその測定ポイント付近でゼロ位相点をサーチします。

11.6 ビルトイン関数の使い方

本器は、帯域幅解析やリップル解析などの波形解析に関するビルトイン関数を装備しています。

R3753 およびR3764/66, R3765/67 ではマーカ機能を使うことにより波形解析を行うことができますが、ビルトイン関数では 1つの関数をコールすることですべての操作を行うことができます。

ビルトイン関数の機能説明については、「R3752/53, R3764/66, R3765/67プログラミング・マニュアル」の[4.4 ビルトイン関数]を参照して下さい。
ここでは、ビルトイン関数の使用例を中心に説明します。

11.6.1 基本関数

以下に示すビルトイン関数は、実際の解析関数で必要なパラメータの算出などの基本変換を行います。

| | |
|---------|----------------------------|
| POINT1 | 指定周波数に最も近い測定ポイントを取得する |
| POINT1L | 指定周波数以下で、最大の測定ポイントを取得する |
| POINT1H | 指定周波数以上で、最小の測定ポイントを取得する |
| POINT2 | 指定周波数に最も近いアドレス・ポイントを取得する |
| POINT2L | 指定周波数以下で、最大のアドレス・ポイントを取得する |
| POINT2H | 指定周波数以上で、最小のアドレス・ポイントを取得する |
| DPOINT | 指定周波数幅に対応するアドレス・ポイント幅を取得する |
| SWPOINT | 最新の測定ポイントを取得する |
| FREQ | 指定アドレス・ポイントに対応する周波数を取得する |
| DFREQ | 指定アドレス間に対応する周波数幅を取得する |
| SWFREQ | 最新の掃引周波数を取得する |
| VALUE | 指定アドレス・ポイントのレスポンス値を取得する |
| DVALUE | 指定アドレス間のレスポンス値差を取得する |
| CVALUE | 指定周波数のレスポンス値を取得する |
| DCVALUE | 指定周波数間のレスポンス値差を取得する |
| SWVALUE | 最新の測定レスポンス値を取得する |

多くのビルトイン関数は、アドレス・ポイントを引数として扱います。他のビルトイン関数を使用するために、これらの関数を使って周波数を測定ポイントに変換します。アドレス・ポイントの絶対範囲は 0~1200です。

この範囲の中で実際の測定値が存在するのが測定ポイントです。測定ポイントは、測定条件で設定される測定ポイント数によって変化します。

測定ポイント以外のアドレス・ポイントのデータは、測定ポイントから補間された値が使用されます。

以下にプログラム例を示します。

```
100 P = POINT1(250.0E6, 0)    ! 250MHzに最も近い測定ポイント
110 V = VALUE(P, 0)          ! 測定値の取得
120 P0 = POINT1L(100.0E6, 0) ! 100MHz以下でアドレス・ポイントの最大値
130 P1 = POINT1H(200.0E6, 0) ! 200MHz以上でアドレス・ポイントの最小値
140 Va = MAX(P0, P1, 0)      ! 最大値を取得する
```

11.6.2 最大および最小値解析関数の使用例

以下に示すビルトイン関数は、指定された範囲内で最大および最小値を解析します。

| | |
|------|---------------------|
| MAX | 最大レスポンス値を取得する |
| MIN | 最小最大レスポンス値を取得する |
| FMAX | 最大レスポンスの周波数を取得する |
| FMIN | 最小レスポンスの周波数を取得する |
| PMAX | 最大レスポンスの測定ポイントを取得する |
| PMIN | 最小レスポンスの測定ポイントを取得する |

これらの関数は、指定されたチャンネルのアドレス・ポイント間で、レスポンス値が最大または最小となる測定ポイントをサーチします。そして、その測定ポイントの解析値が関数値として渡ります。

MAX、MIN関数はレスポンス値を、FMAX、FMIN関数はステイミュラス値（周波数値）を、PMAX、PMINは測定ポイント値を返します。

これらの関数を組み合わせて使用すると、共振点や反共振点を解析することができます。

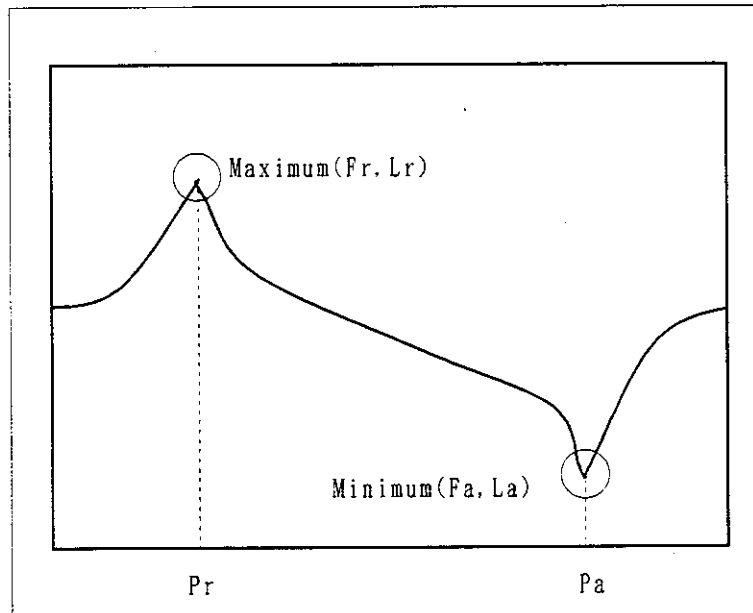


図 11 - 3 最大および最小値解析関数

以下に最大および最小値を解析するプログラム例を示します。

| | |
|---------------------------|-------------------------|
| 100 Vr = MAX(0, 1200, 0) | ! 最大レスポンス値を取得する |
| 110 Fr = FMAX(0, 1200, 0) | ! 最大レスポンスのステイミュラス値を取得する |
| 120 Pr = PMAX(0, 1200, 0) | ! 最大レスポンスの測定ポイントを取得する |
| 130 Va = MIN(0, 1200, 0) | ! 最小レスポンス値を取得する |
| 140 Fa = FMIN(0, 1200, 0) | ! 最小レスポンスのステイミュラス値を取得する |
| 150 Pa = PMIN(0, 1200, 0) | ! 最小レスポンスの測定ポイントを取得する |

最大または最小レスポンスの全解析値を取得する場合は、これらの関数の全部を呼び出す必要はありません。

最初に PMAX または PMIN 関数で測定ポイント値を取得し、この測定ポイント値をパラメータとして、FREQ および VALUE 関数を呼び出します。

すると、MAX, FMAX または MIN, FMIN よりも高速に解析値を取得することができます。

| | |
|---------------------------|-------------------------|
| 100 Pr = PMAX(0, 1200, 0) | ! 最大レスポンスの測定ポイントを取得する |
| 110 Vr = VALUE(Pr, 0) | ! 最大レスポンス値を取得する |
| 120 Fr = FREQ(Pr, 0) | ! 最大レスポンスのステイミュラス値を取得する |
| 140 Pa = PMIN(0, 1200, 0) | ! 最小レスポンスの測定ポイントを取得する |
| 150 Va = VALUE(Pa, 0) | ! 最小レスポンス値を取得する |
| 160 Fa = FREQ(Pa, 0) | ! 最小レスポンスのステイミュラス値を取得する |

11.6.3 減衰レベル解析関数の使用例

以下に示すビルトイン関数は、フィルタなどに特有なパラメータを解析する関数です。

| | |
|-------|----------------------------|
| BND | 指定アドレス・ポイントから帯域幅を取得する |
| BNDL | 指定アドレス・ポイントから帯域幅の低周波数を取得する |
| BNDH | 指定アドレス・ポイントから帯域幅の高周波数を取得する |
| CBND | 指定周波数から帯域幅を取得する |
| CBNDL | 指定周波数から帯域幅の低周波数を取得する |
| CBNDH | 指定周波数から帯域幅の高周波数を取得する |
| MBNDI | 複数の帯域幅解析を外側方向へ行う |
| MBNDO | 複数の帯域幅解析を内側方向へ行う |

(1) BND, BNDL, BNDH, CBND, CBNDL, CBNDH

これらの関数は、指定された減衰レベルから減衰ポイントやバンド幅を解析する関数です。関数名の最初が C で始まるものは基準点をアドレス・ポイントで指定し、C が無いものは周波数で指定します。

フィルタ特有のパラメータを計算する場合、これらの関数を組み合わせて求めることができます。

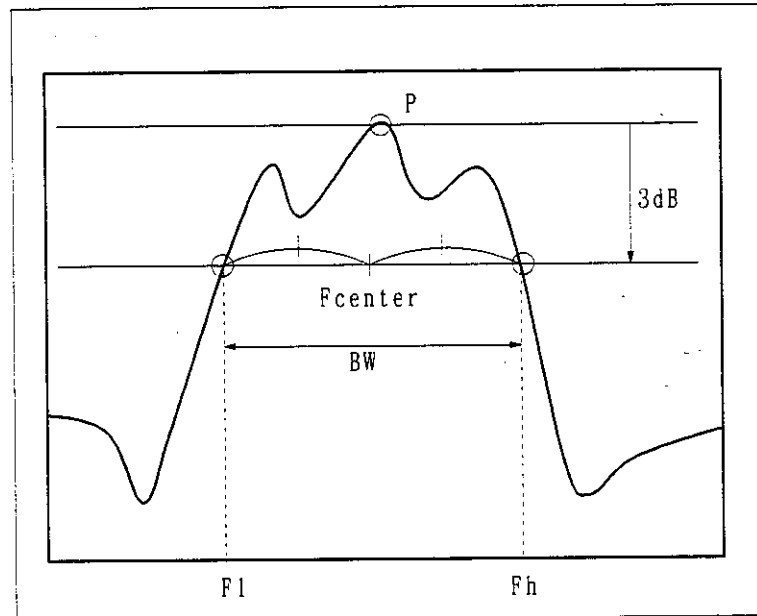


図 11 - 4 減衰レベルの解析

以下に減衰レベルの解析プログラム例を示します。

```

100 P = PMAX(0, 1200, 0) ! 最大レスポンスの測定ポイントを取得する
110 BW = BND(P, 3, 0) ! 減衰レベル 3dBの帯域幅を取得する
120 Fl = BNDL(P, 3, 0) ! 減衰レベル 3dBの帯域幅の低周波数を取得する
130 Fh = BNDH(P, 3, 0) ! 減衰レベル 3dBの帯域幅の高周波数を取得する
140 Fc = (Fl+Fh)*0.5 ! 中心周波数を計算する
150 Q = SQR(Fl*Fh) / BW ! Q を計算する
    
```

(2) MBNDI, MBNDO

複数の減衰レベルの解析を行う場合は、MBNDI またはMBNDO 関数を使用します。これらの関数では、一度に複数の減衰ポイントを解析することができ、1つの減衰レベルに対して、帯域幅の低周波数、高周波数、中心周波数、帯域幅を取得することができます。

減衰レベルの解析を基準ポイントから外側へ向かって解析する場合は、MBNDI 関数を使用します。

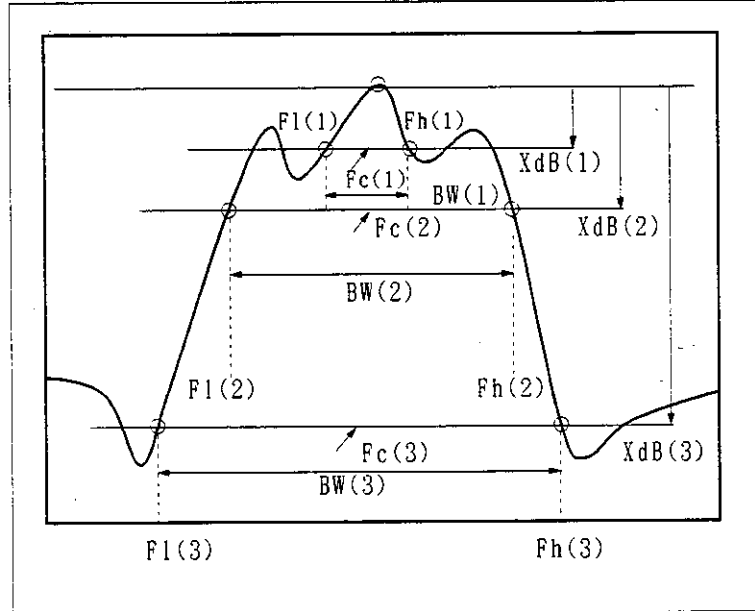


図 11 - 5 MBNDI

以下にMBNDI 関数を使用したプログラム例を示します。

```

100 DIM Levels(3)           ! 減衰レベル指定用の配列定義
110 DIM DataBuffer(3,4)    ! 解析値取得用の配列定義
120 Levels(1) = 1.0        ! 1 番目に解析する減衰レベルを 1.0dBに指定する
130 Levels(2) = 3.0        ! 2 番目に解析する減衰レベルを 3.0dBに指定する
140 Levels(3) = 10.0       ! 3 番目に解析する減衰レベルを10.0dBに指定する
150 P = PMAX(0, 1200, 0)   ! 最大レスポンスの測定ポイントを取得する
160 N = MBNDI(0, 1200, P, 3, Levels(1), DataBuffer(1, 1), 0) ! 複数レベルの解析
170 PRINT DataBuffer(1,1)  ! F1(1)-減衰レベル 1.0dBの帯域幅の低周波数
180 PRINT DataBuffer(1,2)  ! Fh(1)-減衰レベル 1.0dBの帯域幅の高周波数
190 PRINT DataBuffer(1,3)  ! Fc(1)-減衰レベル 1.0dBの帯域幅の中心周波数
200 PRINT DataBuffer(1,4)  ! BW(1)-減衰レベル 1.0dBの帯域幅
210 PRINT DataBuffer(2,1)  ! F1(2)-減衰レベル 1.0dBの帯域幅の低周波数
220 PRINT DataBuffer(2,2)  ! Fh(2)-減衰レベル 1.0dBの帯域幅の高周波数
230 PRINT DataBuffer(2,3)  ! Fc(2)-減衰レベル 1.0dBの帯域幅の中心周波数
240 PRINT DataBuffer(2,4)  ! BW(2)-減衰レベル 1.0dBの帯域幅
250 PRINT DataBuffer(3,1)  ! F1(3)-減衰レベル 1.0dBの帯域幅の低周波数
260 PRINT DataBuffer(3,2)  ! Fh(3)-減衰レベル 1.0dBの帯域幅の高周波数
270 PRINT DataBuffer(3,3)  ! Fc(3)-減衰レベル 1.0dBの帯域幅の中心周波数
280 PRINT DataBuffer(3,4)  ! BW(3)-減衰レベル 1.0dBの帯域幅
    
```

減衰レベルの解析を外側から基準ポイントへ向かって解析する場合は、MBNDO 関数を使用します。

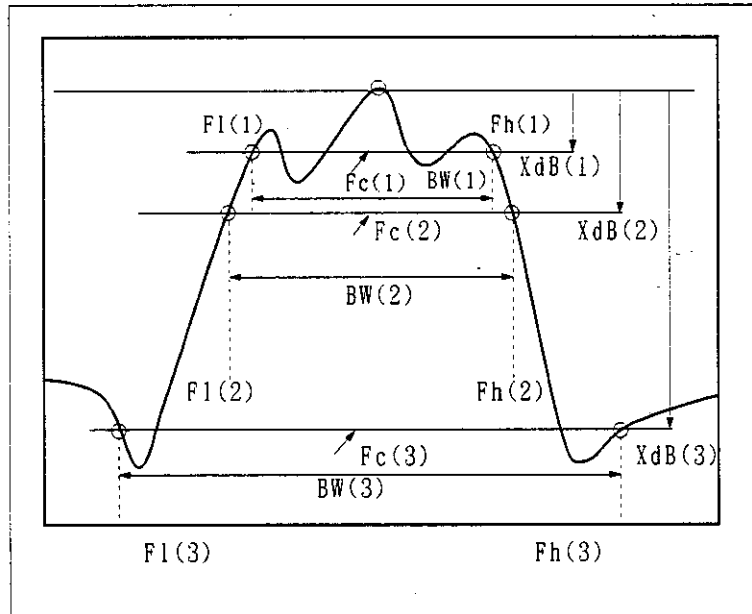


図 11 - 6 MBNDO

以下にMBNDO 関数を使用したプログラム例を示します。

```

100 DIM Levels(3)           ! 減衰レベル指定用の配列定義
110 DIM DataBuffer(3,4)    ! 解析値取得用の配列定義
120 Levels(1) = 1.0        ! 1 番目に解析する減衰レベルを 1.0dBに指定する
130 Levels(2) = 3.0        ! 2 番目に解析する減衰レベルを 3.0dBに指定する
140 Levels(3) = 10.0       ! 3 番目に解析する減衰レベルを10.0dBに指定する
150 P = PMAX(0.1200, 0)    ! 最大レスポンスの測定ポイントを取得する
160 N = MBNDO(0.1200, P, 3, Levels(1), DataBuffer(1,1), 0) ! 複数レベルの解析
170 PRINT DataBuffer(1,1)  ! F1(1)-減衰レベル 1.0dBの帯域幅の低周波数
180 PRINT DataBuffer(1,2)  ! Fh(1)-減衰レベル 1.0dBの帯域幅の高周波数
190 PRINT DataBuffer(1,3)  ! Fc(1)-減衰レベル 1.0dBの帯域幅の中心周波数
200 PRINT DataBuffer(1,4)  ! BW(1)-減衰レベル 1.0dBの帯域幅
210 PRINT DataBuffer(2,1)  ! F1(2)-減衰レベル 1.0dBの帯域幅の低周波数
220 PRINT DataBuffer(2,2)  ! Fh(2)-減衰レベル 1.0dBの帯域幅の高周波数
230 PRINT DataBuffer(2,3)  ! Fc(2)-減衰レベル 1.0dBの帯域幅の中心周波数
240 PRINT DataBuffer(2,4)  ! BW(2)-減衰レベル 1.0dBの帯域幅
250 PRINT DataBuffer(3,1)  ! F1(3)-減衰レベル 1.0dBの帯域幅の低周波数
260 PRINT DataBuffer(3,2)  ! Fh(3)-減衰レベル 1.0dBの帯域幅の高周波数
270 PRINT DataBuffer(3,3)  ! Fc(3)-減衰レベル 1.0dBの帯域幅の中心周波数
280 PRINT DataBuffer(3,4)  ! BW(3)-減衰レベル 1.0dBの帯域幅

```

11.6.4 リップル解析関数の使用例 (1)

以下に示すビルトイン関数は、リップルを解析して結果を取得します。

| | |
|-------|--------------------------------|
| RPL1 | 極大値と極小値の差の最大値を取得する |
| RPL2 | 隣接する極大値と極小値の差の最大値を取得する |
| RPL3 | 隣接する極大値と極小値の差と極小値と極大値の差の合計の最大値 |
| RPL4 | 隣接する極小値と極大値の差の最大値を取得する |
| RPL5 | 極大値の最大値を取得する |
| RPL6 | 極大値の最小値を取得する |
| RPLF | 最初の極大点と極小点との周波数差を取得する |
| RPLR | 最初の極大点と極小点とのレスポンス差を取得する |
| RPLH | 最初の極大点のレスポンス値を取得する |
| FRPLH | 最初の極大点の周波数値を取得する |
| PRPLH | 最初の極大点の測定ポイントを取得する |
| RPLL | 最初の極小点のレスポンス値を取得する |
| FRPLL | 最初の極小点の周波数値を取得する |
| PRPLL | 最初の極小点の測定ポイントを取得する |

サーチの対象とするリップルは、横軸の傾き係数と縦軸の傾き係数で指示します。横軸の傾き係数はアドレス・ポイントで指示し、縦軸の傾き係数はレスポンス値で指定します。

例えば、RPL1関数で 1ポイントあたり 0.5dB上下するリップルを解析する場合は、次のように記述します。

```
100 MaxDiff = RPL1(0.1200, 1, 0.5, 0)
```

(1) RPL1

RPL1関数は、指定された範囲内で、極大値と極小値の差の最大値を取得します。

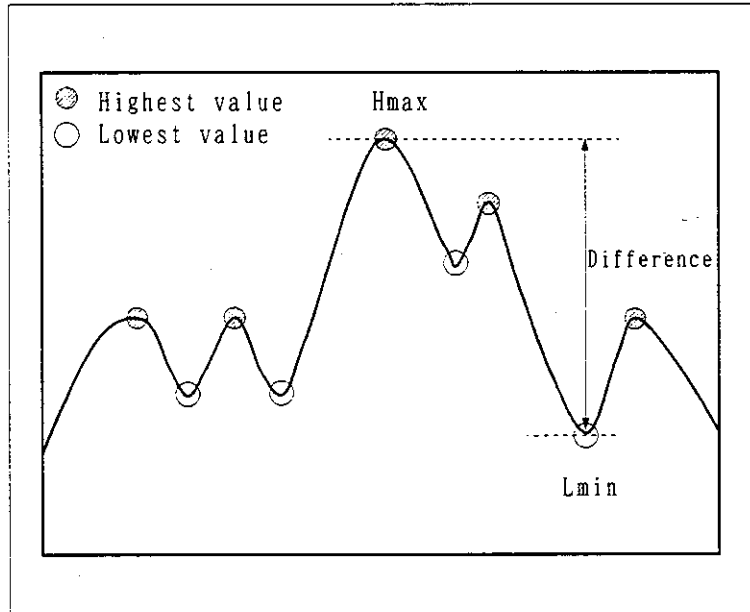


図 11 - 7 RPL1

以下に極大値と極小値の差の最大値を表示するプログラム例を示します。

```
100 MaxDiff = MAX(0, 1200, 0)    ! 極大値と極小値の差の最大値を取得する  
110 PRINT MaxDiff
```

(2) RPL2, RPL4

これらの関数は、隣接する極大値と極小値の差の最大値を取得します。
但し、RPL2は極大値の右側の極小値との差を、RPL4は極大値の左側の極小値との差を検出します。

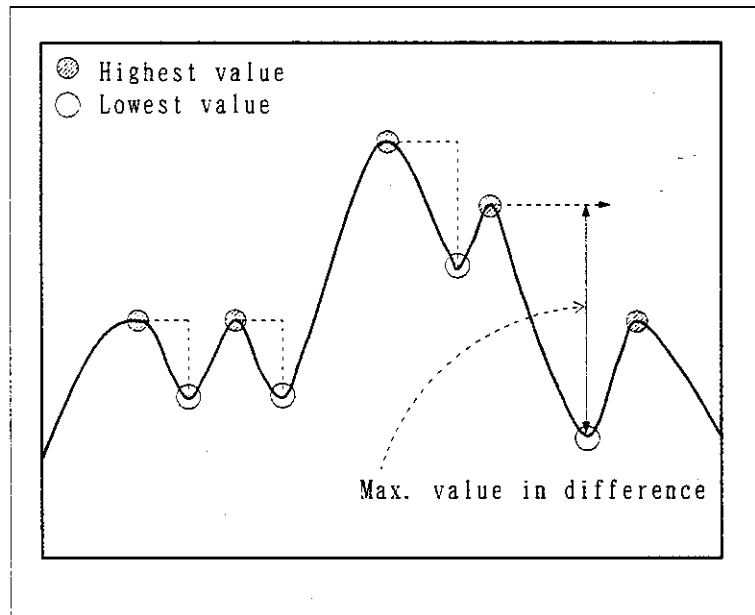


図 11 - 8 RPL2

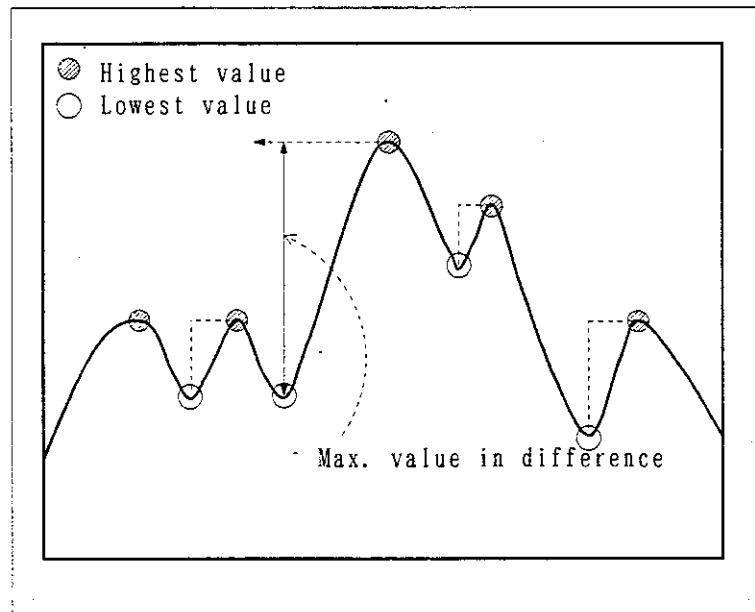


図 11 - 9 RPL4

以下にプログラム例を示します。

```
100 P = PMAX(0, 1200, 0)      ! 最大レスポンスの測定ポイントを取得する
110 RMax = RPL2(0, P, 1, 0.5, 0) ! 右側をサーチする
120 LMax = RPL4(0, P, 1, 0.5, 0) ! 左側をサーチする
```

(3) RPL3

RPL3関数は、隣接する極大値と極小値の差、極小値と極大値の差の2つの差を計算した値の最大値を取得します。

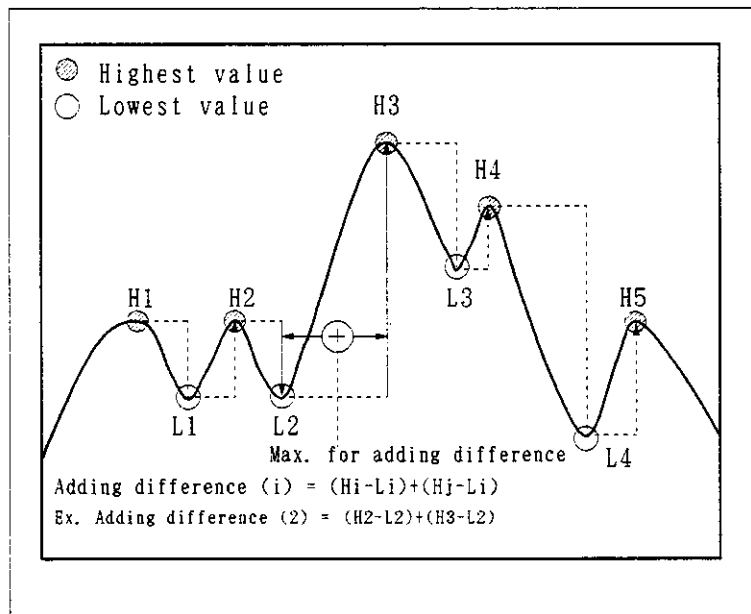


図 11 - 10 RPL3

以下に加算値の最大値を取得するプログラム例を示します。

```
100 MaxAdding = RPL3(0, 1200, 1, 0.5, 0)
```

(4) RPL5, RPL6

これらの関数は、極大値の最大値と最小値を取得します。リップル・スプリアスを解析する場合などに使用します。

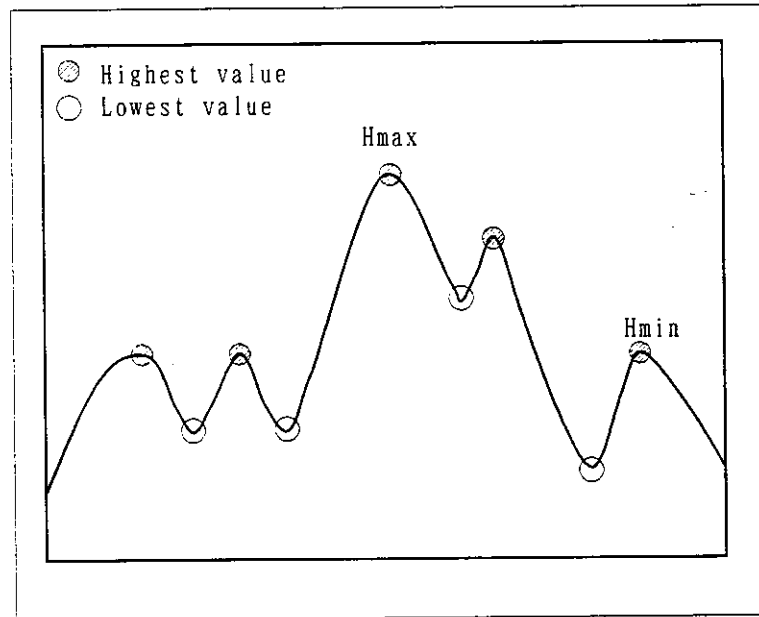


図 11 - 11 極大値の最大値と最小値

以下にプログラム例を示します。

```
100 P0 = POINT1(10.0E6,0)      ! 解析の開始範囲 10MHz
110 P1 = POINT1(20.0E6,0)      ! 解析の終了範囲 20MHz
120 Hmax = RPL5(P0,P1,1,0.5,0) ! 極大値の最大値を取得する
130 Hmin = RPL6(P0,P1,1,0.5,0) ! 極大値の最小値を取得する
```

(5) RPLF, RPLR, RPLH, RPLL, FRPLH, FRPLL, PRPLH, PRPLL

これらの関数は、最初に見つけた極大点および極小点のリップルを解析します。
RPLF, RPLRは極大点と極小点のレスポンス差（または周波数差）について計算し、
RPLH, RPLLは極大点または極小点のレスポンス値を、FRPLH, FRPLLは極大点または
極小点の周波数値を、PRPLH, PRPLLは極大点または極小点の測定ポイントを取得し
ます。

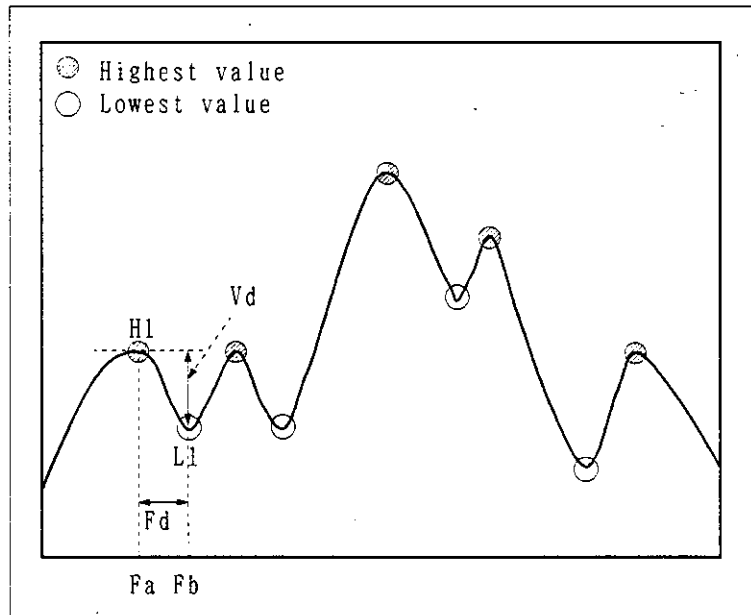


図 11 - 12 リップルのレスポンスおよび周波数

以下にプログラム例を示します。

```

100 Fd = RPLF(0, 1200, 1, 0.5, 0) ! 極大点と極小点の周波数差を取得する
110 Vd = RPLR(0, 1200, 1, 0.5, 0) ! 極大点と極小点のレスポンス差を取得する
120 H1 = RPLH(0, 1200, 1, 0.5, 0) ! 極大点のレスポンスを取得する
130 L1 = RPLL(0, 1200, 1, 0.5, 0) ! 極小点のレスポンスを取得する
140 Fa = FRPLH(0, 1200, 1, 0.5, 0) ! 極大点の周波数を取得する
150 Fb = FRPLL(0, 1200, 1, 0.5, 0) ! 極小点の周波数を取得する
160 Pa = PRPLH(0, 1200, 1, 0.5, 0) ! 極大点の測定ポイントを取得する
170 Pb = PRPLL(0, 1200, 1, 0.5, 0) ! 極小点の測定ポイントを取得する
    
```

なお、このプログラムは実用的ではありません。毎回、ビルトイン関数を呼び出すため、サーチのオーバーヘッドがかかります。極大点と極小点の測定ポイントが分かれば、FREQ, VALUE 関数を使用して周波数とレスポンス値が計算できるので、実際には次のようなプログラムに変更して使います。

| | |
|-----------------------------------|-----------------------|
| 100 Pa = RPLH(0, 1200, 1, 0.5, 0) | ! 極大点の測定ポイントを取得する |
| 110 Pb = RPLL(0, 1200, 1, 0.5, 0) | ! 極小点の測定ポイントを取得する |
| 120 Fa = FREQ(Pa, 0) | ! 極大点の周波数を計算する |
| 130 Fb = FREQ(Pb, 0) | ! 極小点の周波数を計算する |
| 140 H1 = VALUE(Pa, 0) | ! 極大点のレスポンスを計算する |
| 150 L1 = VALUE(Pb, 0) | ! 極小点のレスポンスを計算する |
| 160 Fd = Fb - Fa | ! 極大点と極小点の周波数差を計算する |
| 170 Vd = H1 - L1 | ! 極大点と極小点のレスポンス差を計算する |

11.6.5 リップル解析関数の使用例 (2)

以下に示すビルトイン関数は、最初に解析する全リップルを取得し、リップル番号を指定して解析します。複数のリップルについて解析する場合は、ここで述べるリップル解析関数を使用します。

| NRPLH NRPLL | 極大点の個数を取得する 極小点の個数を取得する |
|----------------|----------------------------|
| PRPLHN | n 番目の極大点の測定ポイントを取得する |
| PRPLLN | n 番目の極小点の測定ポイントを取得する |
| FRPLHN | n 番目の極大点の周波数値を取得する |
| FRPLLN | n 番目の極小点の周波数値を取得する |
| VRPLHN | n 番目の極大点のレスポンス値を取得する |
| VRPLLN | n 番目の極小点のレスポンス値を取得する |
| PRPLHM | 複数の極大点の測定ポイントを取得する |
| PRPLL M | 複数の極小点の測定ポイントを取得する |
| FRPLHM | 複数の極大点の周波数値を取得する |
| FRPLL M | 複数の極小点の周波数値を取得する |
| VRPLHM | 複数の極大点のレスポンス値を取得する |
| VRPLL M | 複数の極小点のレスポンス値を取得する |

(1) NRPLH, NRPLL

これらの関数は、極大点または極小点の個数を解析します。リップル番号を指定して解析するビルトイン関数を使用する場合は、あらかじめNRPLH, NRPLLによってリップル数を取得します。

(2) PRPLHN, PRPLLN, FRPLHN, FRPLLN, VRPLHN, VRPLLN

これらの関数は、NRPLH, NRPLLで取得したリップルの中で番号を指定して特定のリップルの解析を行います。

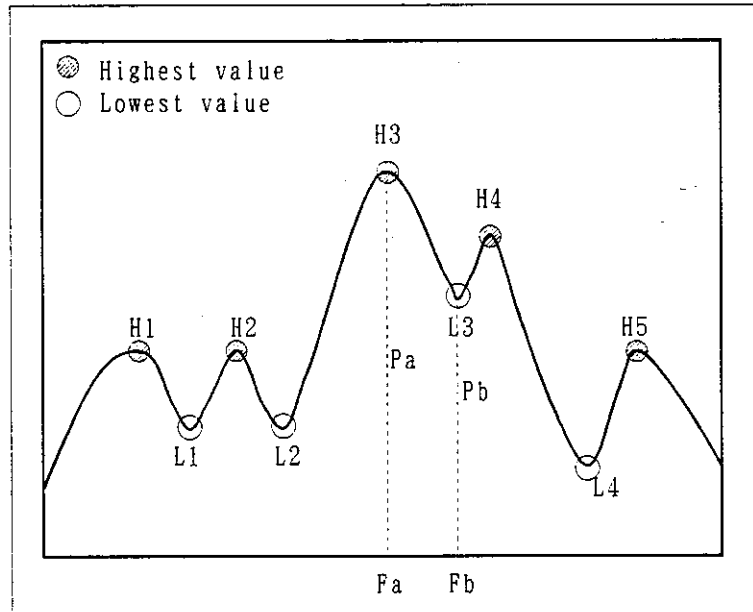


図 11 - 13 特定リップルの解析

以下にプログラム例を示します。

```

100 Nh = NRPLH(0.1200, 1, 0.5, 0)! 極大点をサーチし、番号指定解析を可能にする
110 Nl = NRPLL(0.1200, 1, 0.5, 0)! 極小点をサーチし、番号指定解析を可能にする
120 Pa = PRPLHN(3, 0)             ! 3 番目の極大点の測定ポイントを取得する
130 Fa = FRPLHN(3, 0)             ! 3 番目の極大点の周波数を取得する
140 H3 = VRPLHN(3, 0)             ! 3 番目の極大点のレスポンス値を取得する
150 Pb = PRPLLN(3, 0)            ! 3 番目の極小点の測定ポイントを取得する
160 Fb = FRPLLN(3, 0)            ! 3 番目の極小点の周波数を取得する
170 L3 = VRPLLN(3, 0)            ! 3 番目の極小点のレスポンス値を取得する
    
```

(3) PRPLHM, PRPLL, FRPLHM, FRPLL, VRPLHM, VRPLL

これらの関数は、NRPLH, NRPLLで取得した全リップルの解析値を取得します。

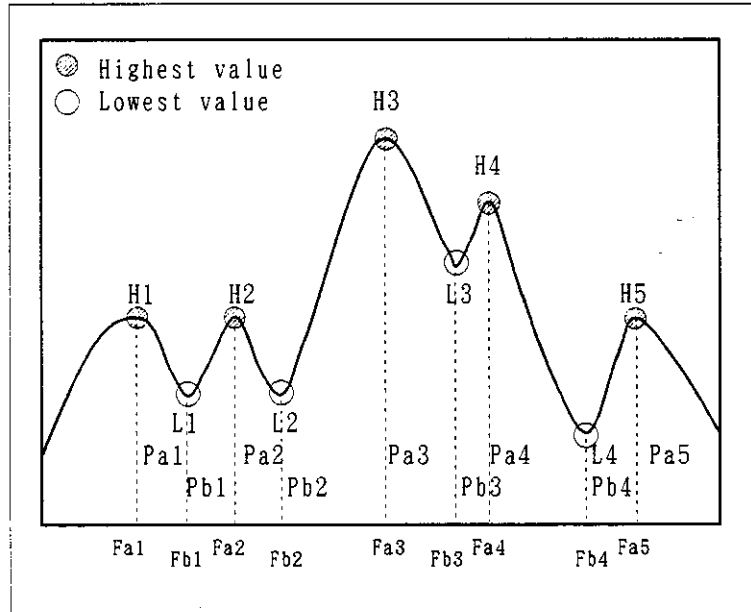


図 11 - 14 全リップルの解析

以下にプログラム例を示します。

```

100 INTEGER Pa(300), Pb(300)
110 DIM Fa(300), Fb(300)
120 DIM Va(300), Vb(300)
130 Nh = NRPLH(0, 1200, 1, 0.5, 0)! 極大点をサーチし、番号指定解析を可能にする
140 Nl = NRPLL(0, 1200, 1, 0.5, 0)! 極小点をサーチし、番号指定解析を可能にする
150 Na = PRPLHM(Pa(1), 0) ! 全極大点の測定ポイントを取得する
160 Nb = PRPLL(Pb(1), 0) ! 全極小点の測定ポイントを取得する
170 Na = FRPLHM(Fa(1), 0) ! 全極大点の周波数を取得する
180 Nb = FRPLL(Fb(1), 0) ! 全極小点の周波数を取得する
190 Na = VRPLHM(Va(1), 0) ! 全極大点のレスポンス値を取得する
200 Nb = VRPLL(Vb(1), 0) ! 全極小点のレスポンス値を取得する
    
```

11.6.6 ダイレクト・サーチ関数の使用例

以下に示すビルトイン関数は、指定された範囲内で与えられたレスポンス値をサーチします。

| | |
|----------|----------------------------|
| DIRECT | 指定レスポンスのアドレス・ポイントを取得する |
| DIRECTL | 指定レスポンスに対応する左側の測定ポイントを取得する |
| DIRECTH | 指定レスポンスに対応する右側の測定ポイントを取得する |
| CDIRECT | 指定レスポンスの周波数を取得する |
| CDIRECTL | 指定レスポンスに対応する左側の実周波数を取得する |
| CDIRECTH | 指定レスポンスに対応する右側の実周波数を取得する |
| DDIRECT | 指定レスポンスのアドレス・ポイント幅を取得する |
| CDDIRECT | 指定レスポンスの帯域幅を取得する |
| ZEROPHS | 最初にゼロ位相となる周波数をサーチする |

(1) DIRECT, DIRECTL, DIRECTH, CDIRECT, CDIRECTL, CDIRECTH

これらの関数は、レスポンス値を指定してそのレスポンス値に一致する所をサーチします。

関数名の最初が Cで始まるDIRECT関数はサーチ範囲を周波数で指定し、それ以外はアドレス・ポイントで指定します。

関数名の最後が Lの関数は低周波数から高周波数方向へ、最後が Hの関数は高周波数から低周波数方向へサーチし、実際の測定ポイントに対応した解析値を求めます。但し、一致する測定ポイントが存在しなければ、指定レスポンス値を超えた直後の測定ポイントとなります。

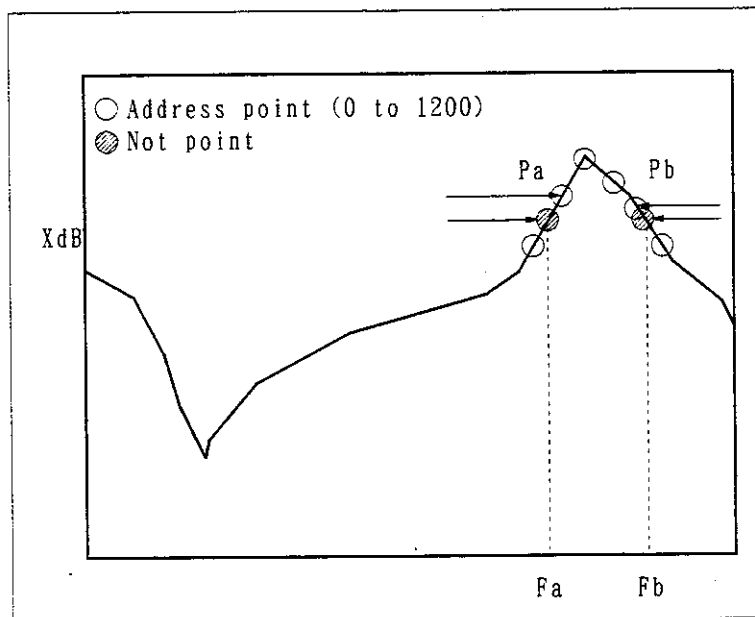


図 11 - 15 ダイレクト・サーチ

以下にプログラム例を示します。

```

100 P = DIRECT(0, 1200, -10, 0)    ! レスポンス値 -10dBのアドレス・ポイント
110 Pa = DIRECTL(0, 1200, -10, 0)  !
120 Pb = DIRECTH(0, 1200, -10, 0)  !
130 F = CDIRECT(5, 500.0E6, -10, 0) ! レスポンス値 -10dBの周波数
140 Pa = CDIRECTL(5, 500.0E6, -10, 0)!
150 Pb = CDIRECTH(5, 500.0E6, -10, 0)!
    
```

(2) DDIRECT, CDDIRECT

これらの関数は、指定されたレスポンス値に対応する 2つの測定ポイントをサーチし、そのポイント幅を取得します。

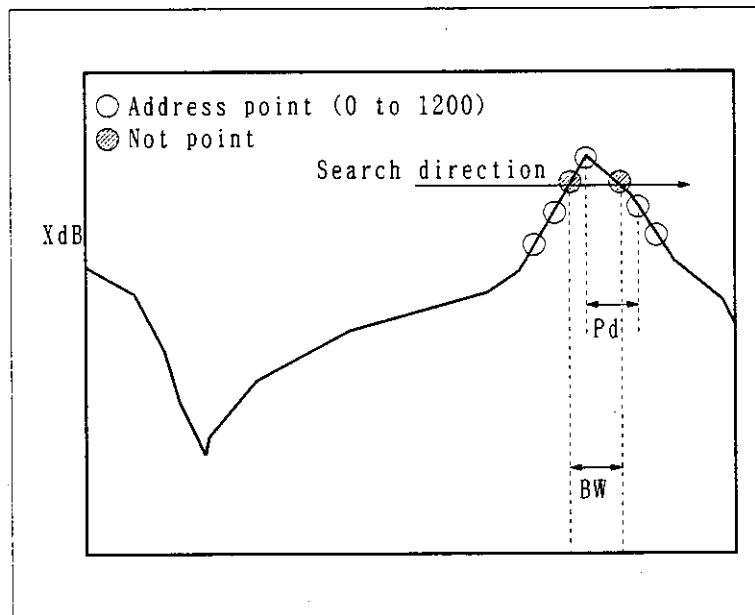


図 11 - 16 レスポンスに対応する帯域幅

以下にプログラム例を示します。

```

100 Pd = DDIRECT(0, 1200, -10, 0)    ! アドレス・ポイント幅
110 BW = CDDIRECT(5, 500.0E6, -10, 0) ! 帯域幅 (周波数で補間する)
    
```

(3) ZEROPHS

ZEROPHS 関数は、指定されたアドレス・ポイント間で位相の値が 0度となる最初の周波数をサーチします。

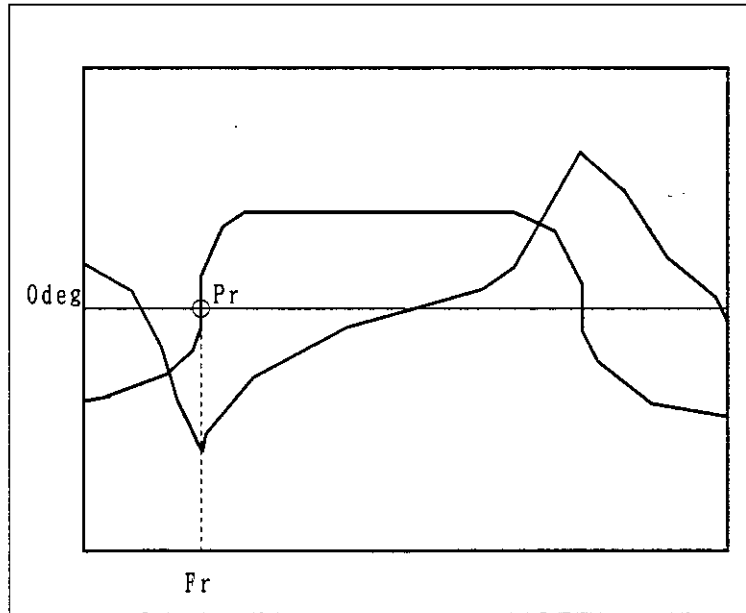


図 11 - 17 ゼロ位相のサーチ

以下にプログラム例を示します。

| | |
|-----------------------------|-----------------------|
| 100 Pr = PMAX(0, 1200, 0) | ! 最大レスポンスの測定ポイントを取得する |
| 110 Pa = PMIN(0, 1200, 0) | ! 最小レスポンスの測定ポイントを取得する |
| 120 Fr = ZEROPHS(Pr, Pa, 0) | ! ゼロ位相の周波数を取得する |

11.6.7 データ転送

以下に示すビルトイン関数は、ビルトインと内蔵BASIC との間でチャンネル・データ転送を行います。

| | |
|--------|-----------------------|
| TRANSR | 解析チャンネルのメモリからデータを読み込む |
| TRANSW | 解析チャンネルのメモリへデータを書き込む |

以下にプログラム例を示します。

| | |
|---------------------------------------|----------------------|
| 100 DIM Buf(2, 1200) | ! データ配列の定義 |
| 110 N = TRANSR(0, 1200, Buf(1, 1), 0) | ! CH1 の第 1波形データを読み込む |
| 120 N = TRANSW(0, 1200, Buf(2, 1), 8) | ! CH1 の第 2波形データを読み込む |

11.7 リミット・ラインの設定

ここでは、リミット・ラインの設定を行うプログラム例を説明します。
試料(DUT)には、880MHzのバンドパス・フィルタを使用します。
セットアップ後、ノーマライズを行い、以下のようなリミット・ラインを設定します。

| セグメント | 0 | 1 | 2 | 3 | 4 |
|-------|--------|--------|--------|--------|--------|
| 周波数 | 780MHz | 820MHz | 866MHz | 898MHz | 960MHz |
| 上限値 | -40dB | -40dB | -10dB | -10dB | -40dB |
| 下限値 | -65dB | -65dB | -30dB | -30dB | -65dB |

(注) R3752/53では使用できません。

プログラム例を [例11-6] に示します。

例 11-6 リミット・ラインの設定

```

100 ! *****
110 !
120 !     SET LIMIT LINE TABLE
130 !
140 ! *****
150 !
160 INTEGER I
170 OUTPUT 31;"OLDC OFF"
180 OUTPUT 31;"SYST:PRES"
190 OUTPUT 31;"DISP:ACT 2"
200 OUTPUT 31;"FREQ:CENT 880MHZ;SPAN 200MHZ"
210 !
220 CLS:CURSOR 0,16
230 INPUT "Connect THRU, then press [X1].",D$
240 OUTPUT 31;"CORR:COLL NORM;*OPC?"
250 ENTER 31;DUMMY$
260 !
270 FOR I=0 TO 4
280 READ ST,UP,LO
290 OUTPUT 31;"DISP:LIM:SEGM",I,"STIM ",ST,"MHZ;UPP ",UP,";LOW ",LO
300 OUTPUT 31;"DISP:LIM:SEGM",I,"COL 3;WCOL 6"
310 NEXT
320 OUTPUT 31;"DISP:LIM:STAT ON;LINE ON"
330 CLS
340 STOP
350 !
360 DATA 780,-40,-65
370 DATA 820,-40,-65
380 DATA 866,-10,-30
390 DATA 898,-10,-30
400 DATA 960,-40,-65

```

リミット・ラインの設定には、
`DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:DATA <block>`
 で全セグメントを一度に設定する方法や、
`DISPlay[:WINDow[<chno>]]:LIMit[<parano>]:SEGMENT<n> <block>`
 でセグメントごとにまとめて設定する方法があります。
 ここでは、各セグメントのパラメータごとに各々設定しています。

以下に、[例11-6] のプログラムの解説をします。

| 例11-6のプログラムの解説 | |
|----------------|-------------------------------|
| 100 | |
| { | コメント行 |
| 150 | |
| 160 | 変数I を整数型にする (セグメントを整数で指定するため) |
| 170 | R3762/63互換コマンド・モードを解除する |
| 180 | ネットワーク・アナライザの設定を初期化する |
| 190 | チャンネル2 をアクティブにする |
| 200 | 掃引周波数を中心880MHz、スパン100MHzにする |
| 210 | コメント行 |
| 220 | 画面上の文字を消去し、カーソルを移動しておく |
| 230 | メッセージを表示して入力を待つ |
| 240 | ノーマライズ・データを取得し、終了通知を要求する |
| 250 | 取得が終るのを待つ |
| 260 | コメント行 |
| 270 | セグメント番号I を 0から4 まで順次変える |
| 280 | 周波数、上限値、下限値のDATAを読み取る |
| 290 | セグメントI に、周波数、上限値、下限値を設定する |
| 300 | リミット・ライン色、波形色を設定する |
| 310 | 次のセグメントに移る |
| 320 | リミット・テスト判定、リミット・ライン表示をオンにする |
| 330 | 画面を消去する |
| 340 | 終了 |
| 350 | コメント行 |
| 360 | |
| { | 各セグメントごとの周波数、上限値、下限値 |
| 400 | |

11.8 全 Sパラメータの 4画面表示

ここでは、全 Sパラメータの 4画面表示を行うプログラム例を説明します。
試料(DUT)には、880MHzのバンドパス・フィルタを使用します。
セットアップ後、2ポート・フルキャリブレーションを行い、以下のような 4画面を表示させます。

| | |
|--------------------------------------|--|
| [CH1] S11 スミス・チャート SMITH(R+jX) | [CH2] S12 振幅/位相 LOG MAG & PHASE |
| [CH3] S22 スミス・チャート SMITH(R+jX) | [CH4] S21 振幅/群遅延時間 LOG MAG & DELAY |

(注) R3752/53では使用できません。また Sパラメータ測定は、R3764/65/66/67では Cタイプ、またはAタイプ+Sパラメータ・テストセット使用のときのみ可能です。

プログラム例を [例11-7] に示します。

例 11-7 全 Sパラメータの 4画面表示

(1/2)

```

100 ! *****
110 !
120 !     2-PORT FULL CALIBRATION
130 !     AND 4 CHANNELS DISPLAY
140 !
150 ! *****
160 !
170 *MAIN
180   GOSUB *SETUP
190   GOSUB *CAL
200   GOSUB *DISP4CH
210   STOP
220   !
230 *SETUP
240   OUTPUT 31;"OLDC OFF"
250   OUTPUT 31;"SYST:PRES"
260   OUTPUT 31;"FREQ:CENT 880MHZ;SPAN 100MHZ"
270   OUTPUT 31;"BAND 100HZ"
280   OUTPUT 31;"DISP:FORM ULOW"
290   CLS:CURSOR 0,16
300   RETURN
310   !
320 *CAL
330   INPUT "Connect OPEN to port 1, then press [X1].",D$
340   OUTPUT 31;"CORR:COLL S110"
350   GOSUB *SWPEND
360   INPUT "Connect SHORT to port 1, then press [X1].",D$

```

(2/2)

```
370 OUTPUT 31;"CORR:COLL S11S"  
380 GOSUB *SWPEND  
390 INPUT "Connect LOAD to port 1, then press [X1].",D$  
400 OUTPUT 31;"CORR:COLL S11L"  
410 GOSUB *SWPEND  
420 INPUT "Connect OPEN to port 2, then press [X1].",D$  
430 OUTPUT 31;"CORR:COLL S22O"  
440 GOSUB *SWPEND  
450 INPUT "Connect SHORT to port 2, then press [X1].",D$  
460 OUTPUT 31;"CORR:COLL S22S"  
470 GOSUB *SWPEND  
480 INPUT "Connect LOAD to port 2, then press [X1].",D$  
490 OUTPUT 31;"CORR:COLL S22L"  
500 GOSUB *SWPEND  
510 !  
520 INPUT "Connect THRU between port 1 and 2, then press [X1].",D$  
530 OUTPUT 31;"CORR:COLL GTHRU"  
540 GOSUB *SWPEND  
550 !  
560 OUTPUT 31;"CORR:COLL OIS"  
570 GOSUB *SWPEND  
580 !  
590 OUTPUT 31;"CORR:COLL:SAVE"  
600 OUTPUT 31;"BAND:AUTO ON"  
610 CLS  
620 RETURN  
630 !  
640 *DISP4CH  
650 OUTPUT 31;"DISP:DUAL ON;FORM ULOW"  
660 OUTPUT 31;"FUNC1:POW S11"  
670 OUTPUT 31;"FUNC2:POW S12"  
680 OUTPUT 31;"FUNC3:POW S22"  
690 OUTPUT 31;"FUNC4:POW S21"  
700 OUTPUT 31;"CALC1:FORM SCH"  
710 OUTPUT 31;"CALC2:FORM MLOP"  
720 OUTPUT 31;"CALC3:FORM SCH"  
730 OUTPUT 31;"CALC4:FORM MLOD"  
740 RETURN  
750 !  
760 *SWPEND  
770 OUTPUT 31;"*OPC?"  
780 ENTER 31;D$  
790 RETURN
```

サブ・メジャー (チャンネル3,4)を表示するには、測定モード指定コマンド
[SENSe:]FUNction<chno>[:ON] "<input>"

または、

[SENSe:]FUNction<chno>:POWER <input>
のチャンネル指定<chno>に 3または4 を指定します。チャンネル3,4 の測定フォーマットの指定などは、あらかじめ測定モード指定でそのチャンネルを表示させてから行います。

以下に、[例11-7] のプログラムの解説をします。

(1/2)

| 例11-7のプログラムの解説 | |
|----------------|-----------------------------|
| 100 | |
| 1 | コメント行 |
| 160 | |
| 170 | メイン・ルーチンのラベル MAIN |
| 180 | 初期設定ルーチンSETUP を呼び出す |
| 190 | 校正ルーチンCAL を呼び出す |
| 200 | 4画面表示ルーチンDISP4CH を呼び出す |
| 210 | 終了 |
| 220 | コメント行 |
| 230 | 初期設定ルーチンのラベル SETUP |
| 240 | R3762/63互換コマンド・モードを解除する |
| 250 | ネットワーク・アナライザの設定を初期化する |
| 260 | 掃引周波数を中心880MHz、スパン100MHzにする |
| 270 | 分解能帯域幅を100Hzにする |
| 280 | 上下2画面分割表示にする |
| 290 | 画面上の文字を消去し、カーソルを移動しておく |
| 300 | 初期設定ルーチンを抜ける |
| 310 | コメント行 |
| 320 | 校正ルーチンのラベル CAL |
| 330 | メッセージを表示して入力を待つ (以下同様) |
| 340 | 校正データ(S11 OPEN)を取得 |
| 350 | 取得の終了を待つ (以下同様) |
| 360 | |
| 1 | 校正データ(S11 SHORT) を取得 |
| 380 | |
| 390 | |
| 1 | 校正データ(S11 LOAD)を取得 |
| 410 | |
| 420 | |
| 1 | 校正データ(S22 OPEN)を取得 |
| 440 | |
| 450 | |
| 1 | 校正データ(S22 SHORT) を取得 |
| 470 | |

例11-7のプログラムの解説

```

480  \  校正データ(S22 LOAD)を取得
500
510  コメント行
520  \  校正データ(GROUP THRU)を取得
540
550  \  校正データを取得(ISOLATION校正省略)
560
580  コメント行
590  校正データから誤差係数を算出
600  分解能帯域幅を自動設定にする
610  画面上の文字を消去
620  校正ルーチンを抜ける
630  コメント行
640  4画面表示ルーチンのラベル DISP4CH
650  両チャンネル表示・上下 2分割表示にする
660  チャンネル1 の測定モードをS11 にする
670  チャンネル2 の測定モードをS12 にする
680  チャンネル3 の測定モードをS22 にする
690  チャンネル4 の測定モードをS21 にする
700  チャンネル1 の測定フォーマットをスミス・チャート(R+jX)にする
710  チャンネル2 の測定フォーマットを振幅/位相にする
720  チャンネル3 の測定フォーマットをスミス・チャート(R+jX)にする
730  チャンネル4 の測定フォーマットを振幅/群遅延時間にする
740  4画面表示ルーチンを抜ける
750  コメント行
760  掃引終了待ちルーチンのラベル SWPEND
770  動作終了の通知を要求する
780  通知を取得する
790  掃引終了待ちルーチンを抜ける

```

12. 外部コントローラの使用例

GPIBを使ってコンピュータと本体を接続し、その間でデータのやりとりをするには、そのコンピュータの言語とプログラム作りに通じている必要があります。

本章ではBASIC(N88-BASIC, QuickBasic, HP-BASIC)、Cで説明しますが、これらがすべて使えなくても結構です。

しかし、BASICのプログラムができないとGPIBを使いこなすのは困難です。これから述べるGPIBに関するプログラミングの前に、使用するコンピュータの言語を習得し、ある程度使えるようにしておいて下さい。

プログラミングの際には、以下のものを用意しておくとい良いでしょう。

- R3764/66, R3765/67 取扱説明書
- R3764/66, R3765/67 プログラミング・マニュアル
- GPIBアドレス割り当て表
- パーソナル・コンピュータのマニュアル
- GPIBインタフェース・ボードのマニュアル

12.1 プログラミングの前に

GPIBとは、ネットワーク・アナライザ（本体）と他のコントローラ、または周辺機器をGPIB用のケーブルで接続できるインタフェース・システムです。

ここでは、本体と外部コントローラをGPIBケーブルで接続し、外部コントローラ（パーソナル・コンピュータ）で本体をコントロールするプログラムを作ります。

電源を入れるとすぐにGPIBの命令を使うことができるコンピュータもありますが、コントローラによっては専用のプログラムをフロッピー・ディスクから読み込む必要があります。また、読み込みの前後にメモリの使用場所を指定したり、入出力ポートを開設するための命令を出す必要のあるプログラムなどもあります。

使用するコントローラのマニュアルをよく読んで、セットアップして下さい。

本章では、NEC社純正のGPIBインタフェース・ボードが備え付けられたPC-9801コンピュータを外部コントローラとした例を中心に、プログラムの作り方を述べます。使用する言語はN88日本語BASICです。

外部コントローラのセットアップが完了したら、本体を所定の条件に設定します。外部コントローラから本体を制御可能にするには、本体のGPIBモードを設定し、GPIBケーブルで接続し、本体のGPIBアドレスを設定しなければなりません。（[12.1.3項]参照）

12.1.1 GPIBのモード

本体のGPIBには、以下の2種類のモードがあります。

- SYSTEM CONTROLLER モード

内蔵の BASICプログラムにより、測定機能や本体に接続した機器をコントロールすることができます。

- TAKER/LISTENERモード

外部コントローラにより本体をコントロールできます。
内蔵の BASICインタプリタを併用することにより、外部コントローラの負荷を軽減できます。

12.1.2 本体との接続

本体の背面パネルにあるGPIBコネクタと外部コントローラのGPIBコネクタを別売のGPIBケーブルで接続します。

接続するときは、インタフェース・ボードおよび使用するコンピュータの取扱説明書をよく読んでから始めて下さい。

(注) 外部コントローラのGPIBコネクタは、GPIBインタフェース・ボードに付いています。NEC-9800シリーズやIBM-PC互換機では、GPIBインタフェース・ボードを購入する必要があります。

GPIBケーブルは長さによって型名が異なります。以下に型名と長さを示します。

表 12 - 1 GPIBケーブル (別売)

| 型名 | 長さ |
|-----------|------|
| 408JE-1P5 | 0.5m |
| 408JE-101 | 1m |
| 408JE-102 | 2m |
| 408JE-104 | 4m |

12.1.3 GPIBアドレスの設定

GPIBを使って、外部コントローラで本体を制御する場合、本体のGPIBアドレスを設定する必要があります。

アドレスを設定すると本体の不揮発性メモリに保存されます。アドレスを変更する場合を除いては、再度設定する必要はありません。

GPIBのアドレスの設定方法は R3764/66 とR3765/67で異なります。次にそれぞれの設定方法を示します。

(注) GPIBアドレスの設定は、それぞれの正面パネル・キーで行います。

- R3764/66でのアドレス設定 (R3764/66 取扱説明書[4.1.5項]を参照)

CONFIG

- ① プログラム・モード上で を押して、CONFIGモードにします。
蛍光表示管にシステム変数の一覧が表示されます。

CONFIG

- ② ADDRESS のラベルが反転表示されるまで を押し続けます。
- ③ 数値キーを使ってアドレスを入力し を押します。
- ④ 設定値を本体内部の不揮発性メモリに保存するには、再度 を押します。
蛍光表示管に確認のメッセージが表示されるので、保存するときは を押します。

- R3765/67でのアドレス設定 (R3765/67 取扱説明書[7.10.2]を参照)

- ① を押して、GPIBメニュー・モードにします。
- ② メニューから を選び、SET ADDRESSES メニュー・モードに切り替えます。
- ③ 次のメニューから を選ぶと、現在の設定アドレスがアクティブ・エリアに表示されます。
- ④ ここで、数値キーを使ってアドレスを入力し を押します。
すると、本体のアドレスが設定され、不揮発性メモリに保存されます。

(注) GPIBアドレスを設定する場合、外部コントローラに割り付けたアドレスや接続されている他の機器のアドレスと重ならないように注意して下さい。

また、ここで指定したアドレスは、外部コントローラを使用して本体を制御するときを使うアドレスで、内蔵 BASICで本体を制御するときを使うアドレスは31です。

12.2 プログラムの書き方

今までは、本体 BASIC言語を使ってプログラム例を説明してきました。本体 BASIC言語で書かれたプログラムは、R3764/66とR3765/67の全シリーズで実行できます。

しかし、外部コントローラ上で実行するプログラムは、コンピュータや組み込む GPIB インタフェース・ボードの動作条件、どのような言語を使ってプログラムを書くかによって異なります。つまり、使用するコントローラ的环境によってプログラムの書き方（スタイル）が変わってきます。

本章では、主な外部コントローラとして以下のものを取り上げます。

表 12 - 2 主な外部コントローラの特長

| 使用言語 | コンピュータ | GPIBボード |
|--------------|---------|----------|
| N88-日本語BASIC | PC-9801 | 純正ボード |
| HP-BASIC | HP-9000 | (内蔵) |
| QuickBASIC | PC/AT | NI-488.2 |
| MicrosoftC | PC/AT | NI-488.2 |

この節では、これらの外部コントローラで実行するプログラムの簡単なプログラムの書き方について説明します。以下に作成するプログラムの概要を示します。

【プログラムの概要】

- (1) コントローラの初期化
- (2) 本体の測定条件を設定
- (3) 本体の内蔵 BASICで測定データを検索（準備）
- (4) 本体の内蔵 BASICから測定データを読み込む
- (5) 測定データをコンピュータのディスプレイに表示
- (6) プログラムを終了

12.2.1 N88-BASIC でのプログラムの書き方

PC-9801 を BASICモードにして、以下のプログラムを入力して下さい。

例 12 - 1 PC-9801 上での GPIBコントロール・プログラム (N88-BASIC) (1/2)

```

1000 * *****
1010 * *
1020 *          GPIB CONTROL PROGRAM          *
1030 * *
1040 * TARGET:   PC-9801(PURE)                *
1050 * LANGUAGE: N88-BASIC                     *
1060 * FILE:     N88STYLE.BAS                  *
1070 * *****
1080 *

```


(2/2)

```
1090 ' (1) INITIALIZE
1100 '
1110 ISET IFC
1120 ISET REN
1130 NA=11
1140 '
1150 ' (2) SETUP
1160 '
1170 PRINT @NA;"OLDC OFF"
1180 PRINT @NA;"FREQ:CENT 150MAHZ"
1190 PRINT @NA;"FREQ:SPAN 300MAHZ"
1200 '
1210 ' (3) SEARCH DATA BY BUILTIN
1220 '
1230 PRINT @NA;"@AP=POINT1(1.5e+8,0)"
1240 PRINT @NA;"@FR=FREQ(AP,0)"
1250 PRINT @NA;"@LV=VALUE(AP,0)"
1260 PRINT @NA;"@PH=VALUE(AP,8)"
1270 '
1280 ' (4) GETTING DATA
1290 '
1300 PRINT @NA;"@OUTPUT 11;FR"
1310 INPUT @NA;F
1320 PRINT @NA;"@OUTPUT 11;LV"
1330 INPUT @NA;L
1340 PRINT @NA;"@OUTPUT 11;PH"
1350 INPUT @NA;P
1360 '
1370 ' (5) DISPLAY DATA
1380 '
1390 FR=F/10^6
1400 PRINT USING "FREQ = ####.### [MHz]";FR
1410 PRINT USING "LEVEL = ####.### [dB]";L
1420 PRINT USING "PHASE = ####.### [deg]";P
1430 '
1440 ' (7) ENDING
1450 '
1460 END
```

次にRUN を入力し、 を押してプログラムを実行します。

結果は、以下ようになります。

【実行結果】

```
FREQ = 150.000 [MHz]
LEVEL = -3.855 [dB]
PHASE = 148.070 [deg]
```

GPIBを使う場合、最初にGPIBのインタフェース・クリアとリモート・イネーブル信号を出力する必要があります。

N88-BASIC では、ISET IFC命令とISET REN命令を使います。

● ISET IFC

IFC(インタフェース・クリア)を送出します。これはGPIBのインタフェースを初期化するものです。

GPIBで外部コントローラから本体を制御するときには必ず指定する必要があります。

● ISET REN

REN(リモート・イネーブル)を送出して、本体をリモート状態にします。

本体がリモート状態になると、本体正面パネルの ● REMOTE LEDが点灯します。

本体がリモート状態でないとき(ローカル状態)は、GPIBコマンドを送っても実行されません。必ずリモート状態にして下さい。

N88-BASIC でGPIBコマンドを本体に送信するにはPRINT @ 命令を、受信する場合はINPUT @ 命令を使います。この命令は、本体内蔵 BASICのOUTPUTと INPUT命令に対応します。@(アット・マーク)の後に、本体のGPIBアドレスを指定します。

[例12-1] では、本体のアドレスは11となっています。

本体の内蔵 BASICにコマンドを送る場合は、BASIC コマンドの先頭に @を付けます。

@ 付きのコマンドは、測定条件などを設定するGPIBコマンドと別の経路で処理されます。

[例12-1] では、内蔵 BASICを使って測定データを取り込みます。

(詳細は、[12.3.2 "@" 付き内蔵 BASICコマンドの転送]を参照)

12.2.2 HP-BASICでのプログラムの書き方

HP-9000 で本体を制御する場合は、以下のようなプログラムとなります。

例 12 - 2 HP9000上でのGPIBコントロール・プログラム(HP-BASIC) (1/2)

```

1000 ! *****
1010 ! *
1020 ! * GPIB CONTROL PROGRAM *
1030 ! *
1040 ! * TARGET: HP-9000(PURE) *
1050 ! * LANGUAGE: HP-BASIC *
1060 ! * FILE: HPSTYLE.BAS *
1070 ! *****
1080 !
1090 ! (1) INITIALIZE
1100 !
1110 ASSIGN @Na TO 711
1120 !

```

(2/2)

```
1130 ! (2) SETUP
1140 !
1150 OUTPUT @Na;"OLDC OFF"
1160 OUTPUT @Na;"FREQ:CENT 150MAHZ"
1170 OUTPUT @Na;"FREQ:SPAN 300MAHZ"
1180 !
1190 ! (3) SEARCH DATA BY BUILTIN
1200 !
1210 OUTPUT @Na;"@AP=POINTI(1.5e+8,0)"
1220 OUTPUT @Na;"@FR=FREQ(AP,0)"
1230 OUTPUT @Na;"@LV=VALUE(AP,0)"
1240 OUTPUT @Na;"@PH=VALUE(AP,8)"
1250 !
1260 ! (4) GETTING DATA
1270 !
1280 OUTPUT @Na;"@OUTPUT 11;FR"
1290 ENTER @Na;F
1300 OUTPUT @Na;"@OUTPUT 11;LV"
1310 ENTER @Na;L
1320 OUTPUT @Na;"@OUTPUT 11;PH"
1330 ENTER @Na;P
1340 !
1350 ! (5) DISPLAY DATA
1360 !
1370 Fr=F/10^6
1380 PRINT "FREQ [MHz] = ";
1390 PRINT USING "DDDD.DDD";Fr
1400 PRINT "LEVEL [dB] = ";
1410 PRINT USING "DDDD.DDD";L
1420 PRINT "PHASE [deg] = ";
1430 PRINT USING "DDDD.DDD";P
1440 !
1450 ! (7) ENDING
1460 !
1470 END
```

HP-BASICで本体のアドレスを指定するには、最初にASSIGN命令で I/O経路を定義する必要があります。このプログラムでは、1110行目 ASSIGN @Na TO 711 で、I/O 経路名の @Naを作ってアドレス11の本体に割り当てています。

作成したI/O 経路名は、本体にコマンドを送ったり、データを受信する場合に使用します。コマンドの送信はOUTPUT命令で、データの受信は ENTER命令で行います。

但し、必ず命令の後に I/O経路名を記述します。

12.2.3 QuickBASICでのプログラムの書き方

QuickBASICを使用したプログラム例を以下に示します。

(注) このプログラムでは、GPIBインタフェース・ボードとして PC/AT用のNI-488.2を使用します。

※NI-488.2: ナショナル・インストルメンツ (株) の登録商標

例 12 - 3 PC/AT 上でのGPIBコントロール・プログラム(QuickBASIC) (1/4)

```

* *****
*
*          GPIB CONTROL PROGRAM
*
* * TARGET:  PC/AT(NI-488.2)
* * LANGUAGE: QuickBASIC
* * FILE:    QBSTYLE.BAS
* *****

REM $INCLUDE: 'qbdecl.bas'

DECLARE SUB naout (na%, msg$)
DECLARE SUB nainp (na%, dat$)
DECLARE SUB naerr (msg$)
DECLARE SUB gpiberr (msg$)

' (1) INITIALIZE
'
BDNAME$ = "GPIB0"
dvname$ = "DEV11"

CALL IBFIND(BDNAME$, brd0%)
IF (brd0% < 0) THEN CALL gpiberr("ibfind1 error")

CALL IBSIC(brd0%)
IF (IBSTA% AND EERR) THEN CALL gpiberr("ibsic error")
CALL IBSRE(brd0%, 1)
IF (IBSTA% AND EERR) THEN CALL gpiberr("ibsre error")

CALL IBFIND(dvname$, na%)
IF (na% < 0) THEN CALL gpiberr("ibfind2 error")

' (2) SETUP
'
CALL naout(na%, "OLDC OFF")

CALL naout(na%, "FREQ:CENT 150MAHZ")
CALL naout(na%, "FREQ:SPAN 300MAHZ")

```

(2/4)

```

' (3) SEARCH DATA BY BUILTIN
,
CALL naout(na%, "@AP=POINT1(1.5e+8,0)")
CALL naout(na%, "@LV=VALUE(AP,0)")
CALL naout(na%, "@FR=FREQ(AP,0)")
CALL naout(na%, "@LV=VALUE(AP,0)")
CALL naout(na%, "@PH=VALUE(AP,8)")

' (4) GETTING DATA
,
CALL naout(na%, "@OUTPUT 11;FR")
CALL nainp(na%, fdat$)
CALL naout(na%, "@OUTPUT 11;LV")
CALL nainp(na%, ldat$)
CALL naout(na%, "@OUTPUT 11;PH")
CALL nainp(na%, pdat$)

' (5) DISPLAY DATA
,
F = VAL(fdat$)
l = VAL(ldat$)
p = VAL(pdat$)
fr = F / 10^6
PRINT USING "FREQ = ###.### [MHz]"; fr
PRINT USING "LEVEL = ###.### [dB]"; l
PRINT USING "PHASE = ###.### [deg]"; p

' (7) ENDING
,
CALL IBONL(na%, 0)
CALL IBONL(brd0%, 0)
END

' This routine prints the result of status variables.
,
SUB gpiberr (msg$) STATIC
PRINT msg$
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 " EEND";
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 RREM THEN PRINT " RREM";
IF IBSTA% AND CIC THEN PRINT " CIC";
IF IBSTA% AND AATN THEN PRINT " AATN";

```

```

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 listner>"
IF IBERR% = EADR THEN PRINT " EADR <Address error>"
IF IBERR% = EARG THEN PRINT " EARG <Invalid argment>"
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 <Fils 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 IBONL(na%, 0)
CALL IBONL(brd0%, 0)
STOP
END SUB

' This routine would notify you that the na returned an invalid serial poll
' response byte.
SUB naerr (msg$) STATIC
PRINT msg$
PRINT "Status Byte = "; SPR%
CALL IBONL(na%, 0)
CALL IBONL(brd0%, 0)
STOP
END SUB

' This routine nainps the data from device
SUB nainp (na%, dat$) STATIC
RD$ = SPACE$(27)

CALL IBRD(na%, RD$)
IF (IBSTA% AND EERR) THEN CALL gpiberr("ibrd error")
dat$ = LEFT$(RD$, IBCNT%)
END SUB

```

(4/4)

```
' This routine naouts the command to device
,
SUB naout (dsc%, msg$) STATIC
  CALL ibwrt(dsc%, msg$)
  emsg$ = "ibwrt error:" + msg$
  IF (IBSTA% AND BERR) THEN CALL gpiberr(emsg$)
END SUB
```

NI-488.2には、インタフェース・ボードの他に、プログラム開発に必要なさまざまなツールが含まれています。NI-488.2システム購入時に、マニュアルをよく読んでコンピュータへインストールしておいて下さい。

関数や変数の先頭文字がibで始まるものは、NI-488.2システムで供給されるライブラリ関数や変数です。このプログラムでは、IEEE488.1レベルのライブラリ関数を使用します。

以下に、ここで使用するNI-488.2のライブラリおよびファイルについて述べます。

【NI-488.2のライブラリおよびファイル】

● qbdecl.bas

QuickBASICプログラムの関数宣言ファイルです。

NI-488.2をインストールすると、QBASICディレクトリが作成され、このディレクトリの下にサンプル・プログラムとともにコピーされます。

このファイルを自分の作業ディスクへコピーして下さい。

● ibfind

GPIBインタフェース・ボードやそれに接続されているGPIBデバイスを検索し、それらに固有な値を割り付けます。NI-488.2ライブラリでは、ibfindで取得する値を引数として使用します。

[例12-3] では、brd0% にボード(0)を、na% に本体(アドレス11)を割り付けています。

ライブラリ関数では、ボード・レベルで操作する関数とデバイス指定で操作する関数の2系統があります。

● ibsic

IFC(インタフェース・クリア)メッセージを出します。この操作はGPIBボードに対して行います。

ibsic を実行すると、GPIBインタフェース・ボードを初期設定し、CIC(コントローラ・イン・チャージ)状態にします。

● ibsre

REN(リモート・イネーブル) 信号を制御します。この操作は GPIB ボードに対して行います。

ibsre(brd\$0,1)を実行すると、本体はリスン・アドレスに指定されると同時にリモート状態になります。

● ibwrt

指定されたデバイスに GPIB メッセージを送ります。

[例 12-3] では、output サブ・プロシージャの中で本体に GPIB コマンドを送るために呼び出しています。

● ibrd

指定されたデバイスから GPIB メッセージを受け取ります。

[例 12-3] では、input サブ・プロシージャの中で本体からデータを受け取るために呼び出しています。

● ibonl

ibfind で割り付けたデバイスを解放します。

プログラムを終了するときに、必ずコールして下さい。

● ibsta%, iberr%, ibcnt%

これらは、ライブラリ関数が設定するステータス変数です。

これらの変数やこの変数にセットされる値は、qbdecl.bas ファイルで定義されています。

ライブラリ関数をコールした直後に、必ずステータスを読み出して、操作が正常にできたかどうかをチェックしなければなりません。

N88-BASIC や HP-BASIC では、GPIB 操作命令が BASIC 言語に組み込まれているため、GPIB プログラムは通常のステートメントと同様に記述することができます。また、エラーが発生した場合は、BASIC 言語システムで捕捉されるため、特にエラーの処理をプログラムする必要はありません。

しかし、QuickBASIC では異なります。GPIB 命令は言語に組み込まれておらず、CALL 文を使ってリンクされた外のライブラリ関数を呼び出します。

このため、QuickBASIC は GPIB 操作で起きたエラーを捕捉することができません。つまり、ライブラリ関数をコールするたびにステータス変数をチェックしなければなりません。このチェックをすべての関数コールの直後に記述すると、プログラム行が長く、読みにくくなります。コールされる頻度が少ないライブラリ関数は問題ありませんが、頻度が高いライブラリ関数では好ましくありません。

[例 12-3] では、ibewrt と ibrd をコールする専用のサブ・プロシージャを記述していません。output と input がそうです。本体にコマンドを送るときは output を、データを受信するときには input を呼び出すだけで済みます。これらのサブ・プロシージャが実際にライブラリを呼び出し、必要なエラーの処理を行います。

また、実際にエラー処理は、サブ・プロシージャ gpiberrと baerrで行います。
エラー処理をサブ・プロシージャで記述すると、プログラム全体が読みやすくなります。

[例12-3] のプログラムをMS-DOS上で実行するには、以下の手順で実行ファイルを作ります。

【実行ファイル作成手順】

- ① 以下のようにコマンドを入力します。

```
LIB QBIB.LIB + QBIB.OBJ;
```

NI-488.2システムで添付されるQuickBASIC用のオブジェクト・ファイルQBIB.OBJからライブラリQBIB.LIBを作ります。
ライブラリの作成は、MS-DOSの LIBコマンドを使います。

- ② 以下のようにコマンドを入力します。

```
BC qbstyle.bas;
```

QuickBASICのBCコマンドを使って、プログラム・ファイル qbstyle.basをコンパイルします。
この結果、オブジェクト・ファイル QBSTYLE.OBJができます。

- ③ 以下のようにコマンドを入力します。

```
LINK QBSTYLE.OBJ,.,QBIB.LIB;
```

MS-DOSのLINKコマンドを使って、 QBSTYLE.OBJとQBIB.LIBとをリンクします。
以上の操作で実行ファイル QBSTYLE.EXEが作成されます。

- ④ QBSTYLE と入力し、

| |
|-------|
| Enter |
|-------|

 を押すと実行できます。

12.2.4 C でのプログラムの書き方

ANSI-Cを使用したプログラム例を以下に示します。

(注) このプログラムでは、GPIBインタフェース・ボードとして PC/AT用のNI-488.2を使用します。

例 12 - 4 PC/AT 上でのGPIBコントロール・プログラム(ANSI-C) (1/3)

```
/*
 *      GPIB CONTROL PROGRAM
 *
 * TARGET:  PC/AT(NI-488.2)
 * LANGUAGE: C (ANSI-C STYLE)
 * FILE:    CSTYLE.C
 */

#include <stdio.h>
#include <stdlib.h>
#include <setjmp.h>
#include <errno.h>
#include "decl.h"

#define DATSIZ 27

jmp_buf jmpbuf;

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 & SRQI) printf("SRQI");
    if (ibsta & RQS ) printf("RQS");
    if (ibsta & CMPL) printf("CMPL");
    if (ibsta & LOK ) printf("LOK");
    if (ibsta & CIC ) printf("CIC");
    if (ibsta & TACS) printf("TACS");
    if (ibsta & LACS) printf("LACS");
    if (ibsta & DTAS) printf("DTAS");
    if (ibsta & DCAS) printf("DCAS");
    printf(" >\n");

    printf("iberr= %d ", iberr);
    switch (iberr)
    {
        case EDVR: printf("EDVR <DOS Error>"); break;
        case ECIC: printf("ECIC <Not CIC>"); break;
    }
}
```

(2/3)

```
case ENOL: printf("ENOL <No listner>"); break;
case EADR: printf("EADR <Address error>"); break;
case EARG: printf("EARG <invalid argment>"); break;
case ESAC: printf("ESAC <Not Sys Ctrlr>"); break;
case EABO: printf("EABO <Op. aborted>"); break;
case ENEB: printf("ENEB <No GPIB board>"); break;
case EOIP: printf("EOIP <Async I/O in prg>"); break;
case ECAP: printf("ECAP <No capability>"); break;
case EFSO: printf("EFSO <Fils sys. error>"); break;
case EBUS: printf("EBUS <Command error>"); break;
case ESTB: printf("ESTB <Status byte lost>"); break;
case ESRQ: printf("ESRQ <SRQ stuck on>"); break;
case ETAB: printf("ETAB <Table Overflow>"); break;
}
printf("ibcntl= %d\n\n", ibcntl);
longjmp(jmpbuf, EIO);
}

void outstr(int dsc, char *cmd)
{
    ibwrt(dsc, cmd, strlen(cmd));
    if (ibsta & ERR) gpiberr("ibwrt error");
}

void inpstr(int dsc, char *cmd, char *buf, unsigned bufsiz)
{
    if (cmd && *cmd)
    {
        ibwrt(dsc, cmd, strlen(cmd));
        if (ibsta & ERR) gpiberr("ibwrt error");
    }
    ibrd(dsc, buf, bufsiz);
    if (ibsta & ERR) gpiberr("ibrd error");
    buf[ibcnt] = '\0';
}

main(int argc, char **argv)
{
    char sf[DATSIZ+1];
    char sl[DATSIZ+1];
    char sp[DATSIZ+1];
    int bd = -1;
    int na = -1;
    int err;
```

```
if (err = setjmp(jmpbuf))
{
    if (na >= 0) ibonl(na, 0);
    if (bd >= 0) ibonl(bd, 0);
    exit(1);
}

/* (1) INITIALIZE
*/
if ((bd = ibfind("GP1B0")) < 0) gpiberr("ibfind error");
if ((na = ibfind("DEV11")) < 0) gpiberr("ibfind error");
if (ibsic(bd) & ERR) gpiberr("ibsic error");
if (ibsre(bd, 1) & ERR) gpiberr("ibsre error");

/* (2) SETUP
*/
outstr(na, "OLDC OFF");
outstr(na, "FREQ:CENT 150MAHZ");
outstr(na, "FREQ:SPAN 300MAHZ");

/* (3) SEARCH DATA BY BUILTIN
*/
outstr(na, "@AP=POINT1(1.5e+8,0)");
outstr(na, "@LV=VALUE(AP,0)");
outstr(na, "@FR=FREQ(AP,0)");
outstr(na, "@LV=VALUE(AP,0)");
outstr(na, "@PH=VALUE(AP,8)");

/* (4) GETTING DATA
*/
inpstr(na, "@OUTPUT 11;FR", sf, DATSIZ);
inpstr(na, "@OUTPUT 11;LV", sl, DATSIZ);
inpstr(na, "@OUTPUT 11;PH", sp, DATSIZ);

/* (5) DISPLAY DATA
*/
printf("FREQ = %4.4f MHz\n", atof(sf)/1.0e6);
printf("LEVEL = %4.4f dB\n", atof(sl));
printf("PHASE = %4.4f deg\n", atof(sp));

/* (6) ENDING
*/
ibonl(na, 0);
ibonl(bd, 0);
}
```

NI-488.2パッケージには、Cプログラミングのためのライブラリが用意されています。このプログラムでは、それらのライブラリを使用しています。使用するライブラリ関数については、前項の [例12-3] のプログラムと同じです。

このプログラムをMicrosoftCでコンパイルする場合は、以下の手順で行います。

【コンパイル手順】

- ① NI-488.2の CパッケージからDECL.HとMCIB.OBJを自分の作業ディスクにコピーします。
- ② [例12-4] のプログラムをコンパイルします。
以下のようにコマンドを入力します。

```
CL MCSTYLE.C MCIB.OBJ
```

以上の操作で実行ファイル MCSTYLE.EXEが作成されます。

- ③ MCSTYLE と入力し、 を押すと実行できます。

12.3 外部コントローラによるリモート・コントロール

外部コントローラから本体をリモート・コントロールするには、 GPIBコマンドを使用します。 GPIBコマンドは、本体の機能をコマンドに置き換えたものです。

波形データの解析などを行うビルトイン関数は、通常の GPIBコマンドでは実行できません。このような場合、内蔵 BASIC にアット・マーク (@) 付きのコマンドを送って処理します。転送コマンドの最初に @ を付けて送ると、内蔵 BASIC へ送られて処理されます。

12.3.1 通常の GPIB コマンドの転送

以下のプログラムは、本体の測定フォーマットを LOGMAG にします。

例 12 - 5 N88-BASIC での GPIB コマンド転送

```
1000 ISET IFC
1010 ISET REN
1020 NA=11
1030 PRINT @NA;"OLDC OFF"
1040 PRINT @NA;"CALC:FORM MLOG"
1050 END
```

例 12 - 6 HP-BASIC での GPIB コマンド転送

```
1000 ASSIGN @Na TO 711
1010 OUTPUT @Na;"OLDC OFF"
1020 OUTPUT @Na;"CALC:FORM MLOG"
1030 END
```

例 12 - 7 QuickBASIC での GPIB コマンド転送

```
REM $INCLUDE: 'qbdecl.bas'

CALL ibfind("GPIBO", bd%)
CALL ibfind("DEVII", na%)
CALL ibsic(bd%)
CALL ibsre(bd%, 1)
CALL ibwrt(na%, "OLDC OFF")
CALL ibwrt(na%, "CALC:FORM MLOG")
CALL ibonl(na%, 0)
CALL ibonl(bd%, 0)
END
```

例 12 - 8 C での GPIB コマンド 転送

```
#include <stdio.h>
#include <stdlib.h>
#include "decl.h"

main(int argc, char **argv)
{
    int bd, na;

    bd = ibfind("GPIBO");
    na = ibfind("DEV11");
    ibsic(bd);
    ibsre(bd, 1);
    ibwrt(na, "OLDC OFF", 8);
    ibwrt(na, "CALC:FORM MLOG", 13);
    ibonl(na, 0);
    ibonl(bd, 0);
}
```

12.3.2 "@" 付き内蔵 BASIC コマンドの転送

アット・マーク(@) 付きのコマンドは、波形解析機能などを行う場合に使います。
内蔵 BASIC で処理の一部を実行し、外部コントローラの負荷を下げるすることができます。

ここで示すプログラムは、@ 付きのコマンドを用いて、測定データの最大値を求め、
その後解析データを受信します。

例 12 - 9 N88-BASIC での BASIC コマンド 転送

```
1000 ISET IFC
1010 ISET REN
1020 NA=11
1030 PRINT @NA;"OLDC OFF"
1040 PRINT @NA;"@V=MAX(0,1200,0)"
1050 PRINT @NA;"@OUTPUT 11;V"
1060 INPUT @NA;V
1070 PRINT V
1080 END
```

例 12 - 10 HP-BASIC での BASIC コマンド 転送

```
1000 ASSIGN @Na TO 711
1010 OUTPUT @Na;"OLDC OFF"
1020 OUTPUT @Na;"@V=MAX(0,1200,0)"
1030 OUTPUT @Na;"@OUTPUT 11;V"
1040 ENTER @Na;V
1050 PRINT V
1060 END
```

例 12 - 11 QuickBASICでの BASICコマンド転送

```
REM $INCLUDE: 'qbdecl.bas'  
  
CALL ibfind("GPIB0", bd%)  
CALL ibfind("DEV11", na%)  
CALL ibsic(bd%)  
CALL ibsre(bd%, 1)  
CALL ibwrt(na%, "OLDC OFF")  
CALL ibwrt(na%, "@V=MAX(0, 1200, 0)")  
CALL ibwrt(na%, "@OUTPUT 11;V")  
dat$=SPACE$(27)  
CALL ibrd(na%, dat$)  
dat$=LEFT$(dat$, ibcnt%)  
PRINT dat$  
CALL ibonl(na%, 0)  
CALL ibonl(bd%, 0)  
END
```

例 12 - 12 C での BASICコマンド転送

```
#include <stdio.h>  
#include <stdlib.h>  
#include "decl.h"  
  
main(int argc, char **argv)  
{  
    char dat[28];  
    int bd, na;  
  
    bd = ibfind("GPIB0");  
    na = ibfind("DEV11");  
    ibsic(bd);  
    ibsre(bd, 1);  
    ibwrt(na, "OLDC OFF", 8);  
    ibwrt(na, "@V=MAX(0, 1200, 0)", 16);  
    ibwrt(na, "@OUTPUT 11;V", 12);  
    ibrd(na, dat, 27);  
    dat[ibcnt] = '\0';  
    printf(dat);  
    ibonl(na, 0);  
    ibonl(bd, 0);  
}
```


12.4 掃引終了の検出

この節では、外部コントローラが掃引の終了を検出する方法について説明します。

まず、本体のトリガ・モードを INIT:CONT OFFにします。この状態でINITコマンドを送ると、1回だけ掃引を実行します。

次に、掃引の終了をステータス・レジスタ（機器の現在の状態を報告するレジスタ）のビット状態により検出します。

ステータス・レジスタの1つ、Standard Operation Event Status RegisterのSweepingというステータス・ビットは、掃引が完了したときに1にセットします。

このステータス・ビットに対応したStandard Operation Event Enable Registerのビットを1にすることにより、掃引が完了したときにサービス要求(SRQ)を発生させることができます。（「R3764/66, R3765/67 プログラミング・マニュアル」の[4. ステータス・バイト]を参照）

12.4.1 N88-BASIC で掃引終了を検出する

以下に示すプログラムは、PC-9801 の N88-BASICで、割り込み処理により測定中の掃引の終了を検出するものです。

ステータス・レジスタにビットが立つと、サービス・リクエスト(SRQ)を発生させることができます。以下のプログラムでは、Standard Operation Event Enable RegisterとSREをイネーブルにすることで、掃引終了時にSRQが発生します。

プログラム中で割り込み処理を宣言しておくこと、SRQが発生したときにそれまで行っていた処理を中断し、割り込みで指定された処理を行います。ここでは、SRQが発生すると、無限ループを中断してラベル *MEAS.ENDへジャンプするように指定しています。

例 12 - 13 N88-BASIC での掃引終了の検出

(1/2)

```

1000 ISET IFC
1010 ISET REN
1020 NA=11
1030 POLL NA,P
1040 ON SRQ GOSUB *MEAS.END
1050 '
1060 *MEAS.SETUP
1070     PRINT @NA;"OLDC OFF"
1080     PRINT @NA;" :INIT:CONT OFF"
1090     PRINT @NA;"*CLS;*SRE 128;:STAT:OPER:ENAB 8"
1100     PRINT @NA;"INIT"
1110     SRQ ON
1120 '
1130 *MEAS.WAIT
1140     GOTO *MEAS.WAIT
1150 '

```

(2/2)

```
1160 *MEAS.END
1170     POLL NA,P
1180     P = P AND 128
1190     IF P<>128 THEN RETURN
1200     PRINT "SWEEP END"
1210     END
```

12.4.2 HP-BASICで掃引終了を検出する

以下に示すプログラムは、HP-9000 シリーズのHP-BASICで、割り込み処理により測定中の掃引の終了を検出するものです。

ON INTR 命令により割り込み分岐先 (Measend)を定義し、ENABLE INTR 命令で割り込みを許可します。

例 12 - 14 HP-BASICでの掃引終了の検出

```
1000 ASSIGN @Na TO 711
1010 !
1020 OUTPUT @Na;"OLDC OFF"
1030 OUTPUT @Na;" :INIT:CONT OFF"
1040 Stat=SPOLL(@Na)
1050 OUTPUT @Na;"*CLS;*SRE 128;;STAT:OPER:ENAB 8"
1060 OUTPUT @Na;"INIT"
1070 ON INTR 7 GOTO Measend
1080 ENABLE INTR 7;255
1090 Measwait: !
1100     GOTO Measwait
1110 !
1120 Measend: !
1130     PRINT "SWEEP END"
1140 END
```

12.4.3 QuickBASICで掃引終了を検出する

次に示すプログラムは、PC/AT 上の QuickBASIC で、NI-488.2のライブラリ関数を利用して測定中の掃引の終了を検出するものです。

spoll は、サービス要求を検出するためのサブ・プロシージャです。

NI-488.2のライブラリ関数ibwaitでシリアル要求が発生するまで待ち、NI-488.2のライブラリ関数 ibrspでステータス・バイトを読みます。

もし、何らかのエラーが発生すると、サブ・プロシージャ gpiberrをコールします。

gpiberr は、エラーが発生したときにコールされ、エラー・メッセージを表示した後STOP命令を実行します。

ライブラリ関数の使い方は、NI-488.2のマニュアルを参照して下さい。

例 12 - 15 QuickBASICでの掃引終了の検出

```
REM $INCLUDE: 'qbdecl.bas'

DECLARE SUB spoll (dsc%, spr%)
DECLARE SUB gpiberr (msg$)

CALL ibfind("GPIBO", bd%)
CALL ibfind("DEV11", na%)
CALL ibsic(bd%)
CALL ibsre(bd%, 1)
CALL ibwrt(na%, "OLDC OFF")
CALL ibwrt(na%, ":INIT:CONT OFF")
CALL ibrsp(na%, spr%)
CALL ibwrt(na%, "*CLS;*SRE 128::STAT:OPER:ENAB 8")
CALL ibwrt(na%, "SWE:TIME 0")
CALL ibwrt(na%, "INIT")
CALL spoll(na%, spr%)
PRINT "SWEEP END"
CALL ibonl(na%, 0)
CALL ibonl(bd%, 0)
END

' Error Handler
,
SUB gpiberr (msg$) STATIC
    PRINT msg$
    PRINT "ibsta=&H"; HEX$(ibsta%)
    CALL ibonl(na%, 0)
    CALL ibonl(bd%, 0)
    STOP
END SUB

' SRQ Handler
,
SUB spoll (na%, spr%) STATIC
    spr% = 0
    mask% = &H800
    CALL ibwait(na%, mask%)
    IF (ibsta% AND EERR) THEN CALL gpiberr("ibwait error")
    CALL ibrsp(na%, spr%)
    IF (ibsta% AND EERR) THEN CALL gpiberr("ibsta error")
    IF (spr% <> &HCO) THEN CALL gpiberr("R3764/66,R3765/67 SRQ error")
END SUB
```

12.4.4 C で掃引終了を検出する

以下に示すプログラムは、PC/AT 上の Microsoft C で、NI-488.2のライブラリ関数を利用して測定中の掃引の終了を検出するものです。

前項と同様に、spoll でサービス要求を検出し、成功すると 0を返します。

ライブラリ関数の使い方は、NI-488.2のマニュアルを参照して下さい。

例 12 - 16 MicrosoftCでの掃引終了の検出

(1/2)

```
#include <stdio.h>
#include <stdlib.h>
#include "decl.h"

static int spoll(int dsc, char *spr)
{
    if (ibwait(dsc, TIMO|RQS) & (ERR|TIMO))
    {
        fprintf(stderr, "ibwait error: &H%x\n", ibsta);
        return -1;
    }
    if (ibrsp(dsc, spr) & ERR)
    {
        fprintf(stderr, "ibrsp error: &H%x\n", ibsta);
        return -1;
    }
    if ((*spr & 0xff) != 0x0C0)
    {
        fprintf(stderr, " R3764/66,R3765/67 error: &H%x\n", *spr);
        return -1;
    }
    return 0;
}

main(int argc, char **argv)
{
    int  bd, na;
    int  err;
    char spr;

    bd = ibfind("GPIB0");
    na = ibfind("DEV11");
    ibsic(bd);
    ibsre(bd, 1);
    ibwrt(na, "OLDC OFF", 8);
    ibwrt(na, ":INIT:CONT OFF", 14);
    ibrsp(na, &spr);
    ibwrt(na, "*CLS:*SRE 128::STAT:OPER:ENAB 8", 31);
}
```

(2/2)

```
ibwrt(na, "SWE:TIME 0", 10);
ibwrt(na, "INIT", 4);
if ((err = spoll(na, &spr)) == -1)
{
    ibonl(na, 0);
    ibonl(bd, 0);
    exit(1);
}
printf("SWEEP END\n");
ibonl(na, 0);
ibonl(bd, 0);
}
```

12.5 トレース・データの転送

外部コントローラと内蔵 BASIC間のデータ転送は、ASCII フォーマットとバイナリ・フォーマットのいずれかで転送することができます。

また、バイナリ・フォーマットは 6種類のフォーマットの中から外部コントローラに対応したフォーマットを選択することができます。（「R3764/66, R3765/67 プログラミング・マニュアル」の[7.7 FORMat サブシステム]を参照）

ASCII フォーマットは簡単な操作でデータ転送できます。

バイナリ・フォーマットは、ASCII フォーマットよりも高速なデータ転送が行えます。

データ転送にスピードが要求されない場合、プログラムが簡単な ASCIIフォーマットが適しています。高速なデータの転送が必要な場合、バイナリ・フォーマットを使います。

この節では、ASCII とバイナリの両フォーマットによるデータ転送のプログラムについて説明します。

12.5.1 本体から PC-9801へのトレース・データ転送

以下のプログラムは、本体のトレース・データを N88-BASICの配列TRへ代入するものです。

[例12-17]に ASCIIフォーマットによるデータ転送、[例12-18]にバイナリ・フォーマットによるデータ転送例を示します。

N88-BASIC では、ブロック・データを転送する GPIB命令を持っていません。

このため、このプログラムではデータの受信部をマシン語で記述し、直接 BIOSルーチン呼び出しています。

また、バイナリ・フォーマットでは、マイクロソフト単精度浮動小数点バイナリを用いています。

例 12 - 17 PC-9801 でのデータ入力(ASCIIフォーマット) (1/2)

```

1000 * *****
1010 * *
1020 * INPUT TRACE DATA FROM NA *
1030 * (ASCII FORMAT) *
1040 * *
1050 * TARGET: PC-9801(PURE) *
1060 * LANGUAGE: N88-BASIC *
1070 * FILE: N88TINPA.BAS *
1080 * *****
1090 *
1100 DIM TR$(1201,2)
1110 *
1120 ISET IFC
1130 ISET REN
1140 NA=11
1150 *
1160 *TINP.EXEC
1170 PC98=IEEE(1) AND &H1F
1180 CMD DELIM=3

```

(2/2)

```

1190 PRINT @NA;"OLDC OFF" @
1200 PRINT @NA;"FORM:DATA ASC" @
1210 FOR N=1 TO 2
1220     PRINT @NA;"TRAC? FDAT"+CHR$(48+N) @
1230     WBYTE &H3F,&H5F,&H40+NA,&H20+PC98;
1240     GOSUB *TINP.RECEIVE
1250     WBYTE &H3F,&H5F,&H40+PC98,&H20,NA;
1260 NEXT
1270 GOSUB *TINP.PRINT
1280 WBYTE &H3F,&H5F;
1290 END
1300
1310 *TINP.PRINT
1320 PRINT @NA;"SWE:POIN?" @
1330 INPUT @NA;PTS
1340 FOR I=0 TO PTS-1
1350     PRINT I,TR$(I,1),TR$(I,2)
1360 NEXT
1370 RETURN
1380
1390 *TINP.RECEIVE
1400 I%=0
1410 A$=""
1420 *RECEIVE.NEXT
1430     RBYTE ;D%
1440     S%=IEEE(2) AND &H8
1450     IF D%=44 THEN *RECEIVE.SEPARATE
1460     A$=A$+CHR$(D%)
1470     IF S%<>0 THEN *RECEIVE.SEPARATE
1480     GOTO *RECEIVE.NEXT
1490 *RECEIVE.SEPARATE
1500     TR$(I%,N)=LEFT$(A$,22)
1510     I%=I%+1
1520     A$=""
1530     LOCATE 0,24:PRINT I%;
1540     IF S%=0 THEN *RECEIVE.NEXT
1550     PRINT
1560 RETURN

```

例 12 - 18 PC-9801 でのデータ入力 (バイナリ・フォーマット) (1/2)

```

1000 * *****
1010 * *
1020 * INPUT TRACE DATA FROM NA *
1030 * (BINARY FORMAT) *
1040 * *
1050 * TARGET: PC-9801(PURE) *
1060 * LANGUAGE: N88-BASIC *
1070 * FILE: N88TINPB.BAS *
1080 * *****
1090 *
1100 CLEAR &H100:DEF SEG=SEGPTR(2)
1110 DIM TR1!(1202),TR2!(1202)
1120 GOSUB *SETGPIB.RECEIVE
1130 *
1140 ISET IFC
1150 ISET REN
1160 NA=11
1170 *
1180 *TINP.EXEC
1190 PC98=IEEE(1) AND &H1F
1200 CMD DELIM=3
1210 PRINT @NA;"OLDC OFF" @
1220 PRINT @NA;"FORM:DATA MBIN.32" @
1230 FOR N=1 TO 2
1240 PRINT @NA;"TRAC:DATA? FDAT"+CHR$(48+N) @
1250 WBYTE &H3F,&H5F,&H40+NA,&H20+PC98;
1260 NUM%=4816
1270 IF N=1 THEN CALL RECEIVE.DATA(TR1!(0),NUM%)
1280 IF N=2 THEN CALL RECEIVE.DATA(TR2!(0),NUM%)
1290 WBYTE &H3F,&H5F,&H40+PC98,&H20+NA;
1300 NEXT
1310 GOSUB *TINP.PRINT
1320 WBYTE &H3F,&H5F;
1330 END
1340 *
1350 *TINP.PRINT
1360 PRINT @NA;"SWE:POIN?" @
1370 INPUT @NA;PTS
1380 FOR I=1 TO PTS
1390 PRINT I,TR1!(I+1),TR2!(I+1)
1400 NEXT
1410 RETURN
1420 *
1430 * Call GPIB BIOS of RECEIVE DATA
1440 * SYNTAX: CALL RECEIVE.DATA(VAR,SIZE%)
1450 *
1460 *SETGPIB.RECEIVE
1470 RECEIVE.DATA = &H0

```


(2/2)

```

1480 RESTORE *GPIB.BIOS.RECEIVE
1490 FOR ADR = 0 TO &H38
1500     READ BYTE: POKE ADR,BYTE
1510     NEXT
1520     RETURN
1530
1540 *GPIB.BIOS.RECEIVE
1550     DATA &H50           :' PUSH    AX
1560     DATA &H51           :' PUSH    CX
1570     DATA &H52           :' PUSH    DX
1580     DATA &H06           :' PUSH    ES
1590     DATA &H56           :' PUSH    SI
1600     DATA &H57           :' PUSH    DI
1610     DATA &H55           :' PUSH    BP
1620     DATA &H53           :' PUSH    BX
1630     DATA &H8B,&H4F,&H02  :' MOV    CX,2[BX]
1640     DATA &H8E,&HC1       :' MOV    ES,CX
1650     DATA &H8B,&H37       :' MOV    SI,[BX]
1660     DATA &H26,&H8B,&H0C  :' MOV    CX,ES:[SI] ; DATA LENGTH
1670     DATA &H8B,&H7F,&H04  :' MOV    DI,4[BX]  ; DATA OFFSET
1680     DATA &H8E,&H47,&H06  :' MOV    ES,6[BX]  ; SEGMENT BASE
1690     DATA &HBB,&H00,&H00  :' MOV    BX,00H    ; COMMAND LENGTH
1700     DATA &HBE,&H00,&H00  :' MOV    SI,00H    ; COMMAND OFFSET
1710     DATA &HB0,&H80       :' MOV    AL,80H    ; EOI ONLY
1720     DATA &HB4,&H05       :' MOV    AH,05H    ; RECEIVE DATA
1730     DATA &HCD,&HD1       :' INT    OD1H     ; CALL GPIB BIOS
1740     DATA &H5B           :' POP    BX
1750     DATA &H53           :' PUSH   BX
1760     DATA &H8B,&H4F,&H02  :' MOV    CX,2[BX]
1770     DATA &H8E,&HC1       :' MOV    ES,CX
1780     DATA &H8B,&H37       :' MOV    SI,[BX]
1790     DATA &H26,&H89,&H14  :' MOV    ES:[SI],DX
1800     DATA &H5B           :' POP    BX
1810     DATA &H5D           :' POP    BP
1820     DATA &H5F           :' POP    DI
1830     DATA &H5E           :' POP    SI
1840     DATA &H07           :' POP    ES
1850     DATA &H5A           :' POP    DX
1860     DATA &H59           :' POP    CX
1870     DATA &H58           :' POP    AX
1880     DATA &HCF           :' IRET
1890     TOTAL 39H byte

```

バイナリ・フォーマットを使用した場合、始めの 8バイトがヘッダとなります。ここでは、ヘッダ部を含めて配列TRへ読み込んでいます。

TRは単精度小数点の配列で、1ポイント当たりのデータ数が2つなので、1ポイント当たりのバイト数は8バイトとなります。よって、このプログラムでは配列TRの添字が2のところから本来のトレース・データが格納されることとなります。

12.5.2 PC-9801 から本体へのトレース・データ出力

以下のプログラムは、N88-BASIC のプログラム中の配列データを、本体へ転送するものです。

注意

このプログラムは、そのままでは使用できません。
このプログラムを実際に使用する場合、前項のようなトレース・データ入力プログラムによってデータを取得し、ファイルなどに保存し、そのデータを配列TRに読み込んでから行って下さい。

例 12 - 19 PC-9801 でのデータ出力(ASCIIフォーマット) (1/2)

```
1000 ' *****
1010 ' *
1020 ' * OUTPUT TRACE DATA TO NA *
1030 ' * (ASCII FORMAT) *
1040 ' *
1050 ' * TARGET: PC-9801(PURE) *
1060 ' * LANGUAGE: N88-BASIC *
1070 ' * FILE: N88TOUTA.BAS *
1080 ' *****
1090 '
1100 DIM TR$(1201,2)
1120 '
5000 '
5010 ISET IFC
5020 ISET REN
5030 NA=11
5040 '
5050 *TOUT.EXEC
5060 PC98=IEEE(1) AND &H1F
5070 CMD DELIM=3
5080 PRINT @NA;"OLDC OFF" @
5090 PRINT @NA;"FORM:DATA ASC" @
5100 FOR N=1 TO 2
5110 PRINT @NA;"TRAC:DATA FDAT"+CHR$(48)+", " @
5120 WBYTE &H3F,&H5F,&H40+PC98,&H20+NA;
5130 GOSUB *TOUT.SEND
5140 NEXT
5150 WBYTE &H3F,&H5F;
5160 END
5170 '
5180 *TOUT.SEND
5190 I%=0
5200 *SEND.NEXT
5210 IF TR$(I%+1,N)=" " GOTO *SEND.LAST
5220 PRINT @;TR$(I%,N)+CHR$(44)
```

(2/2)

```
5230 LOCATE 0,23:PRINT I%,TR$(I%,N);
5240 I%=I%+1
5250 GOTO *SEND.NEXT
5260 *SEND.LAST
5270 PRINT @;TR$(I%,N) @
5280 LOCATE 0,23:PRINT I%,TR$(I%,N)
5290 RETURN
```

例 12 - 20 PC-9801 でのデータ出力 (バイナリ・フォーマット) (1/2)

```
1000 * *****
1010 * *
1020 * OUTPUT TRACE DATA TO NA *
1030 * (BINARY FORMAT) *
1040 * *
1050 * TARGET: PC-9801(PURE) *
1060 * LANGUAGE: N88-BASIC *
1070 * FILE: N88TOUTB. BAS *
1080 * *****
1090 *
1100 CLEAR &H100:DEF SEG=SEGPTR(2)
1110 DIM TR1!(1202),TR2!(1202)
1120 GOSUB *SETGPIB.SEND
1130 *
1140 ISET IFC
1150 ISET REN
1160 NA=11
1170 *
1180 *TOUT.EXEC
1190 PC98=IEEE(1) AND &H1F
1200 CMD DELIM=3
1210 PRINT @NA;"OLDC OFF" @
1220 PRINT @NA;"FORM:DATA MBIN,32" @
1230 NUM%=4816
1240 FOR N=1 TO 2
1250 PRINT @NA;"TRAC:DATA FDAT"+CHR$(48)+"",
1260 WBYTE &H3F,&H5F,&H40+PC98,&H20+NA;
1270 IF N=1 THEN CALL SEND.DATA(TR1(0),NUM%)
1280 IF N=2 THEN CALL SEND.DATA(TR2(0),NUM%)
1290 NEXT
1300 WBYTE &H3F,&H5F;
1310 END
1320 *
1330 * Call GPIB BIOS of SEND DATA
1340 * SYNTAX: CALL SEND.DATA(VAR.SIZE%)
1350 *
1360 *SETGPIB.SEND
```

(2/2)

```

1370 SEND. DATA = &H39
1380 RESTORE *GPIB. BIOS. SEND
1390 FOR ADR = &H39 TO &H65
1400 READ BYTE: POKE ADR, BYTE
1410 NEXT
1420 RETURN
1430
1440 *GPIB. BIOS. SEND
1450 DATA &H50 : ' PUSH AX
1460 DATA &H51 : ' PUSH CX
1470 DATA &H52 : ' PUSH DX
1480 DATA &H06 : ' PUSH ES
1490 DATA &H56 : ' PUSH SI
1500 DATA &H57 : ' PUSH DI
1510 DATA &H55 : ' PUSH BP
1520 DATA &H53 : ' PUSH BX
1530 DATA &H8B, &H4F, &H02 : ' MOV CX, 2[BX]
1540 DATA &H8E, &HC1 : ' MOV ES, CX
1550 DATA &H8B, &H37 : ' MOV SI, [BX]
1560 DATA &H26, &H8B, &H0C : ' MOV CX, ES:[SI] ; DATA LENGTH
1570 DATA &H8B, &H7F, &H04 : ' MOV DI, 4[BX] ; DATA OFFSET
1580 DATA &H8E, &H47, &H06 : ' MOV ES, 6[BX] ; SEGMENT BASE
1590 DATA &HBB, &H00, &H00 : ' MOV BX, 00H ; COMMAND LENGTH
1600 DATA &HBE, &H00, &H00 : ' MOV SI, 00H ; COMMAND OFFSET
1610 DATA &HB0, &H80 : ' MOV AL, 80H ; EOI ONLY
1620 DATA &HB4, &H04 : ' MOV AH, 04H ; RECEIVE DATA
1630 DATA &HCD, &HD1 : ' INT 0D1H ; CALL GPIB BIOS
1640 DATA &H5B : ' POP BX
1650 DATA &H5D : ' POP BP
1660 DATA &H5F : ' POP DI
1670 DATA &H5E : ' POP SI
1680 DATA &H07 : ' POP ES
1690 DATA &H5A : ' POP DX
1700 DATA &H59 : ' POP CX
1710 DATA &H58 : ' POP AX
1720 DATA &HCF : ' IRET
1730 ' TOTAL 2DH byte

```

12.5.3 本体から HP-BASIC へのトレース・データ転送

以下のプログラムは、本体のトレース・データを HP-BASIC の配列Trへ代入するものです。

バイナリ・フォーマットでは、IEEE倍精度浮動小数点を選択しています。

例 12 - 21 HP-BASICでのデータ入力(ASCIIフォーマット)

```

1000 ! *****
1010 ! *
1020 ! * INPUT TRACE DATA FROM NA *
1030 ! * (ASCII FORMAT) *
1040 ! *
1050 ! * TARGET: HP-9000(PURE) *
1060 ! * LANGUAGE: HP-BASIC *
1070 ! * FILE: HPTINPA.BAS *
1080 ! *****
1090 !
1100 ASSIGN @Na TO 711
1110 DIM Tr1(1:201), Tr2(1:201)
1120 !
1130 OUTPUT @Na;"OLDC OFF"
1140 OUTPUT @Na;"SWE:POIN 201"
1150 OUTPUT @Na;"FORM:DATA ASC"
1160 !
1170 OUTPUT @Na;"TRAC? FDAT1"
1180 ENTER @Na;Tr1(*)
1190 !
1200 OUTPUT @Na;"TRAC? FDAT2"
1210 ENTER @Na;Tr2(*)
1220 !
1230 FOR I=1 TO 201
1240 PRINT "No. ";I;": ";Tr1(I),Tr2(I)
1250 NEXT I
1260 !
1270 END

```

例 12 - 22 HP-BASICでのデータ入力 (バイナリ・フォーマット)

```

1000 ! *****
1010 ! * *
1020 ! * INPUT TRACE DATA FROM NA *
1030 ! * (BINARY FORMAT) *
1040 ! * *
1050 ! * TARGET: HP-9000(PURE) *
1060 ! * LANGUAGE: HP-BASIC *
1070 ! * FILE: HPTINPB. BAS *
1080 ! *****
1090 !
1100 ASSIGN @Na TO 711
1110 ASSIGN @Dt TO 711;FORMAT OFF ! BINARY DATA PASS
1120 DIM Tr(1:201,1:2)
1130 !
1140 OUTPUT @Na;"OLDC OFF"
1150 OUTPUT @Na;"SWE:POIN 201"
1160 OUTPUT @Na;"FORM:BORD NORM"
1170 OUTPUT @Na;"FORM:DATA REAL,64"
1180 !
1190 OUTPUT @Na;"TRAC? FDAT1"
1200 ENTER @Na USING "%.8A";Header$ ! READ HEADER STRING
1210 ENTER @Dt;Tr1(*) ! READ ALL TRACE DATA
1220 ENTER @Na USING "%.1A";Terminate$ ! READ TERMINATOR
1230 !
1240 OUTPUT @Na;"TRAC? FDAT2"
1250 ENTER @Na USING "%.8A";Header$ ! READ HEADER STRING
1260 ENTER @Dt;Tr2(*) ! READ ALL TRACE DATA
1270 ENTER @Na USING "%.1A";Terminate$ ! READ TERMINATOR
1280 !
1290 I=1
1300 WHILE I<202
1310 PRINT "No. ";I;": ";Tr1(I),Tr2(I)
1320 I=I+1
1330 END WHILE
1340 !
1350 END

```

12.5.4 本体からQuickBASICへのトレース・データ転送

以下のプログラムは、本体のトレース・データを PC/ATで実行しているQuickBASICの配列trへ代入するものです。

(注) NI-488.2インタフェース・ボードおよびライブラリ関数を利用します。

例 12 - 23 QuickBASICでのデータ入力(ASCIIフォーマット) (1/4)

```

* *****
*
* INPUT TRACE DATA FROM NA
* (ASCII FORMAT)
*
* TARGET: PC/AT(NI-488.2)
* LANGUAGE: QuickBASIC
* FILE: QBTINPA.BAS
* *****

REM $INCLUDE: 'qbdecl.bas'

DECLARE SUB gpinit (bdname$, bd%)
DECLARE SUB nainit (bd%, naname$, dv%)
DECLARE SUB tisetup (dv%, name$)
DECLARE SUB tireceive (bd%, dat!())
DECLARE SUB tiprint (dv%, dat1!(), dat2!())
DECLARE SUB prtterr (msg$)

DIM tr1!(1 TO 201), tr2!(1 TO 201)

CALL gpinit("GPIBO", bd%)
CALL nainit(bd%, "DEV11", na%)
CALL tisetup(na%, "FDAT1") ' trace 1
CALL tireceive(bd%, tr1!())
CALL tisetup(na%, "FDAT2") ' trace 2
CALL tireceive(bd%, tr2!())
CALL tiprint(na%, tr1!(), tr2!())
CALL ibonl(na%, 0)
CALL ibonl(dv%, 0)
END

' This routine open the gpib board and initialize
SUB gpinit (bdname$, bd%) STATIC

CALL ibfind(bdname$, bd%) ' OPEN BOARD
IF (bd% < 0) THEN
CALL prtterr("ibfind error")
STOP
END IF

```

(2/4)

```
CALL ibsic(bd%)          ' INTERFACE CLEAR.
IF (ibsta% AND EERR) THEN
    CALL prtterr("ibsic error")
    CALL ibonl(bd%, 0)
    STOP
END IF

CALL ibsre(bd%, 1)      ' REMOTE ENABLE
IF (ibsta% AND EERR) THEN
    CALL prtterr("ibsre error")
    CALL ibonl(bd%, 0)
    STOP
END IF

END SUB

' This routine open N.A and initialize
SUB nainit (bd%, dvname$, dv%) STATIC

    CALL ibfind(dvname$, dv%)
    IF (dv% < 0) THEN
        CALL prtterr("ibfind error")
        CALL ibonl(bd%, 0)
        STOP
    END IF

    cmd$ = "OLDC OFF"
    CALL ibwrt(dv%, cmd$)
    IF (ibsta% AND EERR) THEN
        CALL prtterr("ibwrt error")
        CALL ibonl(dv%, 0)
        CALL ibonl(bd%, 0)
        STOP
    END IF

END SUB

' This routine prints the result of status variables.
SUB prtterr (msg$) STATIC

    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 " EEND";
```


(3/4)

```
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 RREM THEN PRINT " RREM";
IF ibsta% AND CIC THEN PRINT " CIC";
IF ibsta% AND AATN THEN PRINT " AATN";
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 listner>"
IF iberr% = EADR THEN PRINT " EADR <Address error>"
IF iberr% = EARG THEN PRINT " EARG <Invalid argment>"
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 <Fils 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%

END SUB

' This routine print received data
'
SUB tiprint (dv%, dat1!(), dat2!()) STATIC

cmd$ = "SWE:POIN?"
CALL ibwrt(dv%, cmd$)
cmd$ = SPACE$(23)
CALL ibrd(dv%, cmd$)
pts% = VAL(cmd$)
FOR num% = 1 TO pts%
PRINT num%, dat1!(num%), dat2!(num%)
NEXT

END SUB
```

(4/4)

```
' This routine receives trace data
,
SUB tireceive (bd%, buf!()) STATIC

    v% = &H427
    CALL ibeos(bd%, v%)

    cmd$ = "?_K"          ' UNL UNT MLA 0 TAD 11
    CALL ibcmd(bd%, cmd$)
    IF (ibsta AND EERR) THEN
        CALL prtterr("ibcmd error")
        STOP
    END IF
    eoi% = 0
    num% = 1
    WHILE eoi% = 0
        cmd$ = SPACES(23)          ' [S#. #####.]
        CALL ibrd(bd%, cmd$)      ' READ ONE DATA
        dat$ = LEFT$(cmd$, 22)
        buf!(num%) = VAL(dat$)
        eoi% = ibsta% AND EEND
        num% = num% + 1
    WEND

    v% = &H40A
    CALL ibeos(bd%, v%)

END SUB

' This routine setups
,
SUB tisetup (dv%, name$) STATIC

    cmd$ = "SWE:POIN 201"        ' BASE COMMAND
    CALL ibwrt(dv%, cmd$)
    cmd$ = "FORM:DATA ASC::TRAC? " + name$ ' TRACE INPUT COMMAND
    CALL ibwrt(dv%, cmd$)

END SUB
```

12.5.5 本体から C へのトレース・データ転送

以下のプログラムは、本体のトレース・データを PC/AT で実行している C の配列 tr へ代入するものです。

(注) NI-488.2 インタフェース・ボードおよびライブラリ関数を利用します。

例 12 - 24 C でのデータ入力(ASCIIフォーマット)

(1/4)

```

/*
 * INPUT TRACE DATA FROM NA
 * (ASCII FORMAT)
 *
 * TARGET: PC/AT(NI-488.2)
 * LANGUAGE: C (ANSI-C STYLE)
 * FILE: MCTINPA.C
 */

#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include "decl.h"

double databuf1[201], databuf2[201]; /* data buffer */

/* prtterr - print gpib error message and status code
 */
static void prtterr(char *msg)
{
    printf("%s\n", msg);
    printf("ibsta=&H%x < ", ibsta);
    if (ibsta & ERR) printf("ERR");
    if (ibsta & TIMO) printf("TIMO");
    if (ibsta & SRQI) printf("SRQI");
    if (ibsta & RQS) printf("RQS");
    if (ibsta & CMPL) printf("CMPL");
    if (ibsta & LOK) printf("LOK");
    if (ibsta & CIC) printf("CIC");
    if (ibsta & TACS) printf("TACS");
    if (ibsta & LACS) printf("LACS");
    if (ibsta & DTAS) printf("DTAS");
    if (ibsta & DCAS) printf("DCAS");
    printf(" >\n");

    printf("iberr= %d ", iberr);
    switch (iberr)
    {
        case EDVR: printf("EDVR <DOS Error>"); break;
        case ECIC: printf("ECIC <Not CIC>"); break;
    }
}

```

```
case ENOL: printf("ENOL <No listner>"); break;
case EADR: printf("EADR <Address error>"); break;
case EARG: printf("EARG <Invalid argment>"); break;
case ESAC: printf("ESAC <Not Sys Ctrlr>"); break;
case EABO: printf("EABO <Op. aborted>"); break;
case ENEB: printf("ENEB <No GPIB board>"); break;
case EOIP: printf("EOIP <Async I/O in prg>"); break;
case ECAP: printf("ECAP <No capability>"); break;
case EFSO: printf("EFSO <Fils sys. error>"); break;
case EBUS: printf("EBUS <Command error>"); break;
case ESTB: printf("ESTB <Status byte lost>"); break;
case ESRQ: printf("ESRQ <SRQ stuck on>"); break;
case ETAB: printf("ETAB <Table Overflow>"); break;
}
printf("ibcntl= %d\n\n", ibcntl);
}

/* gpinit - open gpib board and initialize
*/
static int gpinit(char *bdname)
{
    int  bd;

    if ((bd = ibfind(bdname)) < 0)          /* open board */
    {
        prterr("ibfind error");
        return -1;
    }

    if (ibsic(bd) & ERR)                   /* interface clear */
    {
        prterr("ibsic error");
        ibonl(bd, 0);
        return -1;
    }

    if (ib sre(bd, 1) & ERR)               /* remote enable */
    {
        prterr("ibsre error");
        ibonl(bd, 0);
        return -1;
    }

    return bd;                            /* return descriptor */
}

/* nainit - open N.A port and initialize
*/
```

(3/4)

```

static int nainit(char *dvname)
{
    int  dv;

    if ((dv = ibfind("DEV11")) < 0)      /* open N.A */
    {
        prterr("ibfind error");
        return -1;
    }

    ibwrt(dv, "OLDC OFF", 9);           /* default command */
    if (ibsta & ERR)
    {
        prterr("ibwrt error");
        ibonl(dv, 0);
        return -1;
    }
    return dv;                          /* return descriptor */
}

/* tisetup - setups
*/
static int tisetup(int dv)
{
    ibwrt(dv, "SWE:POIN 201", 13);
    ibwrt(dv, "FORM:DATA ASC", 14);
    return 0;
}

/* tireceive -- receives trace data
*/
static int tireceive(int bd, double *buf, unsigned bufsiz, char *name)
{
    unsigned int len, cnt = 0;
    char  s[32], n[20];

    *n = 0;
    strcat(strcpy(n, "TRAC? "), name);
    ibwrt(bd, n, strlen(n));           /* query */
    ibeos(bd, 0x427);
    ibcmd(bd, "?_K", 4);               /* UNL UNT MLA 0 TAD 11 */

    while (cnt < bufsiz)
    {
        ibrd(bd, s, 23);               /* [S#.#####.] */
        s[ibcnt1] = 0;
    }
}

```

(4/4)

```

        *buf++ = atof(s);
        cnt++;
        if (ibsta & END) break;          /* with EOI */
    }

    ibeos(bd, 0x40A);
    ibcmd(bd, "?_+@", 4);                /* UNL UNT MLA 11 TAD 0 */
    return cnt;
}

/* tiprint - print trace data
*/
static int tiprint(double *data1, double *data2, unsigned num)
{
    unsigned i;

    for (i = 0; i < num; ++i)
    {
        printf("%4d: %1.7e\t%1.7e\n", i, *data1, *data2);
        data1++;
        data2++;
    }
}

/* main entry
*/
main(int argc, char **argv)
{
    int    bd, na;
    int    num;

    if ((bd = gpinit("GP1B0")) == -1)
        exit(1);
    if ((na = nainit("DEV11")) == -1)
    {
        ibonl(bd, 0);
        exit(1);
    }

    tisetup(na);
    num = tireceive(bd, databuf1, 201, "FDAT1");
    num = tireceive(bd, databuf2, 201, "FDAT2");
    tiprint(&databuf1[0], &databuf2[0], num);

    iboni(na, 0);
    ibonl(bd, 0);
}

```

例 12 - 25 C でのデータ入力 (バイナリ・フォーマット) (1/4)

```
/*
 *      INPUT TRACE DATA FROM NA
 *      (BINARY FORMAT)
 *
 * TARGET:   PC/AT(NI-488.2)
 * LANGUAGE: C (ANSI-C STYLE)
 * FILE:     MCTINPB.C
 */
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include "decl.h"

static int gpinit(char *bdname);
static int nainit(char *dynamname);
static int tisetup(int dv);
static int tireceive(int bd, double *buf, unsigned bufsiz);
static int tiprint(double *data, unsigned points);
static void prtterr(char *msg);

double databuf[402];          /* data buffer */

/* main entry
 */
main(int argc, char **argv)
{
    int    bd, na;
    int    num;

    if ((bd = gpinit("GPIB0")) == -1)
        exit(1);
    if ((na = nainit("DEV11")) == -1)
    {
        ibonl(bd, 0);
        exit(1);
    }

    tisetup(na);
    num = tireceive(bd, &databuf[0], 402);
    tiprint(&databuf[0], num);

    ibonl(na, 0);
    ibonl(bd, 0);
}

/* gpinit - open gpib board and initialize
 */
static int gpinit(char *bdname)
```

(2/4)

```
{
  int  bd;

  if ((bd = ibfind(bdname)) < 0)      /* open board */
  {
    prterr("ibfind error");
    return -1;
  }

  if (ibsic(bd) & ERR)                /* interface clear */
  {
    prterr("ibsic error");
    ibonl(bd, 0);
    return -1;
  }

  if (ibsr(bd, 1) & ERR)              /* remote enable */
  {
    prterr("ibsr error");
    ibonl(bd, 0);
    return -1;
  }

  return bd;                          /* return descriptor */
}

/* nainit - open N.A port and initialize
*/
static int nainit(char *dvname)
{
  int  dv;

  if ((dv = ibfind("DEV11")) < 0)    /* open N.A */
  {
    prterr("ibfind error");
    return -1;
  }

  ibwrt(dv, "OLDC OFF", 9);          /* default command */
  if (ibsta & ERR)
  {
    prterr("ibwrt error");
    ibonl(dv, 0);
    return -1;
  }

  return dv;                          /* return descriptor */
}

/* tisetup - setups
*/
```


(3/4)

```

static int tisetup(int dv)
{
    ibwrt(dv, "SWE:POIN 201", 13);
    ibwrt(dv, "FORM:BORD SWAP;DATA REAL,64", 28);
    return 0;
}

/* tireceive - receives trace data
*/
static int tireceive(int bd, double *buf, unsigned bufsiz)
{
    unsigned cnt;
    char s[32];

    ibwrt(bd, "TRAC:DATA? DATA", 16);      /* query */

    ibconfig(bd, 12, 0);                    /* disable EOS detection */
    ibwrt(bd, "?_K", 5);                    /* UNL UNT MLA 0 TAD 11 */

    ibrd(bd, s, 8);                          /* read header */
    ibrd(bd, (char *)buf, sizeof(double)*bufsiz);
    cnt = ibcnt;
    if (!(ibsta&END)) ibrd(bd, s, 1);        /* read terminator */

    ibconfig(bd, 12, '\n');                 /* enable EOS detection */
    return cnt/sizeof(double);
}

/* tiprint - print trace data
*/
static int tiprint(double *data, unsigned num)
{
    unsigned points = num>>1;
    unsigned i;

    for (i=0; i<points; ++i)
    {
        printf("%4d: %1.7e\t%1.7e\n", i, *data, *(data+1));
        data += 2;
    }
}

/* prtterr - print gpib error message and status code
*/
static void prtterr(char *msg)
{
    printf("%s\n", msg);
    printf("ibsta=&H%x < ", ibsta);
}

```

(4/4)

```
if (ibsta & ERR) printf("ERR");
if (ibsta & TIMO) printf("TIMO");
/*if (ibsta & EEND) printf("EEND");*/
if (ibsta & SRQI) printf("SRQI");
if (ibsta & RQS ) printf("RQS");
if (ibsta & CMPL) printf("CMPL");
if (ibsta & LOK ) printf("LOK");
/*if (ibsta & RREM) printf("RREM");*/
if (ibsta & CIC ) printf("CIC");
/*if (ibsta & AATN) printf("AATN");*/
if (ibsta & TACS) printf("TACS");
if (ibsta & LACS) printf("LACS");
if (ibsta & DTAS) printf("DTAS");
if (ibsta & DCAS) printf("DCAS");
printf(" >\n");

printf("iberr= %d ", iberr);
switch (iberr)
{
    case EDVR: printf("EDVR <DOS Error>"); break;
    case ECIC: printf("ECIC <Not CIC>"); break;
    case ENOL: printf("ENOL <No listner>"); break;
    case EADR: printf("EADR <Address error>"); break;
    case EARG: printf("EARG <Invalid argment>"); break;
    case ESAC: printf("ESAC <Not Sys Ctrlr>"); break;
    case EABO: printf("EABO <Op. aborted>"); break;
    case ENEB: printf("ENEB <No GPIB board>"); break;
    case EOIP: printf("EOIP <Async I/O in prg>"); break;
    case ECAP: printf("ECAP <No capability>"); break;
    case EFSO: printf("EFSO <Fils sys. error>"); break;
    case EBUS: printf("EBUS <Command error>"); break;
    case ESTB: printf("ESTB <Status byte lost>"); break;
    case ESRQ: printf("ESRQ <SRQ stuck on>"); break;
    case ETAB: printf("ETAB <Table Overflow>"); break;
}
printf("ibcnt= %d\n\n", ibcntl);
}
```

12.6 内蔵BASIC と外部コントローラを同時に使うために

この節では、外部コントローラから本体の BASICプログラムを実行させて測定を行い、外部コントローラで測定データを受信し、表示する方法を述べます。

まず、本体側の内蔵 BASICで測定プログラムを作ります。ビルトイン関数を使ってフィルタ解析を行うプログラムの例を示します。

その後で、外部コントローラ側のプログラムで、SRQ(サービス・リクエスト)を使って測定データを受信するプログラムを作ります。外部コントローラ側のプログラムは、利用するコンピュータや言語によって異なります。始めに、NEC PC-9801 の N88-BASICプログラムについて説明します。

この節で述べるプログラムの概要を以下に示します。

| 外部コントローラ側 | | 内蔵 BASIC側 |
|------------------------|---|-----------------------------------|
| (1) インタフェースを初期化する | | (1) 本体の測定条件を設定する |
| (2) 本体のプログラムをロードして実行する | → | (2) PAUSE する |
| (3) 本体のプログラムをCONTする | → | (3) 掃引を開始し、WAIT EVENTで掃引が終了するまで待つ |
| (4) SRQ がくるまでループして待つ | | (4) 測定データを解析する |
| (5) SRQ がきたらシリアル・ポルを行う | ← | (5) REQUEST 命令でSRQ を出力する |
| (6) データを受信して画面上に表示する | ← | (6) データを送信する |
| (7) (3)から繰り返す | | (7) (2)から繰り返す |

外部コントローラと本体の内蔵 BASICとの間でデータの送受信を行う場合、上記のやりとりが必要となります。

12.6.1 内蔵 BASICでの送信プログラム

本体内蔵 BASICでフィルタ解析を行い、解析データを送信するプログラムを以下に示します。

例 12 - 26 内蔵 BASICの送信プログラム

(1/2)

```

1000 !*****
1010 !*
1020 !*          DATA TRANSFER PROGRAM
1030 !*
1040 !* TARGET:NETWORK ANALYZER (to PC-9801)
1050 !* FILE:  NSEND.BAS
1060 !*****
1070 INTEGER EV
1080 DIM L(2), F(2,4)
1090 !
1100 *MAIN

```

(2/2)

```
1110 GOSUB *SETUP
1120 CLS
1130 *MEAS_LOOP
1140 CURSOR 0,0
1150 PAUSE
1160 GOSUB *MEAS
1170 GOSUB *SEND
1180 GOTO *MEAS_LOOP
1190 !
1200 *SETUP
1210 NA=31 :PC=11 :EV=1 :L(1)=3.0 :L(2)=60.0
1220 OUTPUT NA;"OLDC OFF"
1230 OUTPUT NA;"SYST:PRES::INIT:CONT OFF::STAT:OPER:ENAB 8;*SRE 128;*OPC?"
1240 ENTER NA;A
1250 OUTPUT NA;"FREQ:SPAN 20MAHZ;CENT 12MAHZ"
1260 RETURN
1270 !
1280 *MEAS
1290 SPOLL(NA)
1300 OUTPUT NA;"INIT" :WAIT EVENT EV
1310 AP=PMAX(0,1200,0)
1320 NP=MBNDI(0,1200,AP,2,L(1),F(1,1),0)
1330 QF=F(1,3)/F(1,4) ! QF = CF(3dB) / BW(3dB)
1340 SF=F(2,4)/F(1,4) ! SF = BW'(60dB) / BW(3dB)
1350 RETURN
1360 !
1370 *SEND
1380 REQUEST 65
1390 OUTPUT PC;F(1,1) :OUTPUT PC;F(1,2) :OUTPUT PC;F(1,3) :OUTPUT PC;F(1,4)
1400 OUTPUT PC;QF :OUTPUT PC;SF
1410 RETURN
```

このプログラムを入力し、フロッピー・ディスクに保存します。
保存するときファイル名は、"NSEND.PGM" とします。
ファイル名は外部コントローラからロードするとき参照されます。

このプログラムでは、初期化および本体の測定条件を行った後、自動的に PAUSE を実行します。

外部コントローラから @CONT が送られてきた後、測定を行い、測定データを解析し、そのデータを外部コントローラへ送信します。

この一連の処理を行った後、また、PAUSE を実行します。

12.6.2 N88-BASIC での受信プログラム

PC-9801 側の受信プログラムを以下に示します。

例 12 - 27 N88-BASIC での受信プログラム

(1/2)

```

1000 ' *****
1010 ' * *
1020 ' * CONTROL AND RECEIVE DATA *
1030 ' * *
1040 ' * TARGET:PC-9801 *
1050 ' * FILE: NRECEIVE.BAS *
1060 ' *****
1070 ISET IFC
1080 ISET REN
1090 NA=11
1100 POLL NA,P
1110 ON SRQ GOSUB *SRINT
1120 '
1130 A$="A:/NSEND.BAS"
1140 M$=CHR$(34)+A$+CHR$(34)
1150 L$="@LOAD "+M$
1160 PRINT @NA;"@SCRATCH"
1170 PRINT @NA;L$
1180 PRINT @NA;"@RUN"
1190 '
1200 CLS
1210 *MEAS.LOOP
1220 GOSUB *MEAS.CONT
1230 GOSUB *RECEIVE
1240 GOTO *MEAS.LOOP
1250 '
1260 *MEAS.CONT
1270 LOCATE 6,9 :PRINT "CONNECT DUT"
1280 LOCATE 6,10 :INPUT "IF OK THEN PRESS ANY KEY",D$
1290 PRINT @NA;"@CONT"
1300 URQ=0
1310 SRQ ON
1320 RETURN
1330 '
1340 *RECEIVE
1350 IF URQ=0 THEN GOTO *RECEIVE
1360 INPUT @NA;LF:INPUT @NA;RF:INPUT @NA;CF:INPUT @NA;BW
1370 INPUT @NA;QF:INPUT @NA;SF
1380 LOCATE 5,1:PRINT USING "C.F = ####.##### [MHz]";CF/10^6
1390 LOCATE 5,2:PRINT USING "L.F = ####.##### [MHz]";LF/10^6
1400 LOCATE 5,3:PRINT USING "R.F = ####.##### [MHz]";RF/10^6
1410 LOCATE 5,4:PRINT USING "BW = ####.##### [MHz]";BW/10^6
1420 LOCATE 5,5:PRINT USING "QF = ####.#####";QF
1430 LOCATE 5,6:PRINT USING "SF = ####.#####";SF

```

(2/2)

```
1440    RETURN
1450
1460 *SRINT
1470    POLL NA, P
1480    P = P AND 1
1490    IF P<>0 THEN URQ=1
1500    RETURN
```

このプログラムを PC-9801 の BASICモードで入力します。
プログラムの実行は、以下の手順で行います。

【実行手順】

- ① [例12-26]を保存したフロッピー・ディスクを本体のドライブに挿入します。
- ② PC-9801 側のプログラムを実行します。

RUN を入力して、 を押します。

PC-9801 側でプログラムを実行すると、自動的に本体側のプログラムをフロッピー・ディスクの中からロードして実行します。
実行されると、プログラムに従い、本体を設定して測定を始めます。
測定が終了して結果が出ると、それを PC-9801側に送ります。

結果は、以下のようになります。

【実行結果】

```
C. F = 146.716000 [MHz]
L. F = 137.156000 [MHz]
R. F = 157.699000 [MHz]
BW = 21.964200 [MHz]
QF = 6.679790
SF = 0.000000
```

このように PC-9801と本体の実行結果をみると、確実にデータの受け渡しが行われたことが分かります。

R3752 シリーズでは、このようなプログラムを作成することにより、マーカ機能を代用することができます。

12.6.3 HP-BASICでの受信プログラム

HP-BASICを使用した受信プログラム例を以下に示します。

例 12 - 28 HP-BASICでの受信プログラム

(1/2)

```

1000 ! *****
1010 ! *
1020 ! * CONTROL AND RECEIVE DATA *
1030 ! *
1040 ! * TARGET:HP-BASIC *
1050 ! * FILE: HPREC.BAS *
1060 ! *****
1070 !
1080 DIM L$(24)
1090 !
1100 ASSIGN @Na TO 711
1110 ON INTR 7 GOSUB Srint
1120 !
1130 A$="A:/NSEND.BAS"
1140 M$=CHR$(34)+A$+CHR$(34)
1150 L$="@LOAD "+M$
1160 !
1170 OUTPUT @Na;"@SCRATCH"
1180 OUTPUT @Na;L$
1190 OUTPUT @Na;"@RUN"
1200 !
1210 Meas_loop:!
1220 GOSUB Meas_cont
1230 GOSUB Receive
1240 GOTO Meas_loop
1250 !
1260 Meas_cont:!
1270 PRINT "CONNECT DUT"
1280 INPUT "IF OK THEN PRESS ANY KEY", D$
1290 OUTPUT @Na;"@CONT"
1300 Urq=0
1310 ENABLE INTR 7;255
1320 RETURN
1330 !
1340 Receive:!
1350 IF Urq=0 THEN GOTO *Receive
1360 DISABLE INTR 7
1370 ENTER @Na;Lf
1380 ENTER @Na;Rf
1390 ENTER @Na;Cf
1400 ENTER @Na;Bw
1410 ENTER @Na;Qf
1420 ENTER @Na;Sf
1430 PRINT "C. F [MHz] = ";

```

(2/2)

```

1440 PRINT USING "DDDD.DDDDDD";Cf/10^6
1450 PRINT "L.F [MHz] = ";
1460 PRINT USING "DDDD.DDDDDD";Lf/10^6
1470 PRINT "R.F [MHz] = ";
1480 PRINT USING "DDDD.DDDDDD";Rf/10^6
1490 PRINT "BW [MHz] = ";
1500 PRINT USING "DDDD.DDDDDD";Bw/10^6
1510 PRINT "QF = ";
1520 PRINT USING "DDDD.DDDDDD";Qf
1530 PRINT "SF = ";
1540 PRINT USING "DDDD.DDDDDD";Sf
1550 RETURN
1560 !
1570 Srint:~
1580 Stat = SPOLL(@Na) AND 1
1590 IF Stat<>0 THEN Urq = 1
1600 RETURN
1610 END

```

12.6.4 QuickBASICでの受信プログラム

QuickBASICを使用した受信プログラム例を以下に示します。

例 12 - 29 QuickBASICでの受信プログラム

(1/5)

```

' *****
' *
' * CONTROL AND RECEIVE DATA *
' *
' * TARGET: PC/AT(NI-488.2) *
' * LANGUAGE: QuickBASIC *
' * FILE: QBREC.BAS *
' *****

REM $INCLUDE: 'qbdecl.bas'

DECLARE SUB gpinit (bdname$, bd%)
DECLARE SUB nainit (bd%, naname$, dv%)
DECLARE SUB nasetup (dv%)
DECLARE SUB nacont (dv%)
DECLARE SUB nareceive (bd%)
DECLARE SUB prterr (msg$)

CALL gpinit("GPIBO", bd%)
CALL nainit(bd%, "DEV11", na%)
CALL nasetup(na%)

```



```
Measloop:
    CALL nacont(na%)
    CALL nareceive(na%)
    GOTO Measloop

CALL ibonl(na%, 0)
CALL ibonl(dv%, 0)
END

' This routine open the gpib board and initialize
'
SUB gpinit (bdname$, bd%) STATIC

    CALL ibfind(bdname$, bd%)      ' OPEN BOARD
    IF (bd% < 0) THEN
        CALL prtterr("ibfind error")
        STOP
    END IF

    CALL ibsic(bd%)                ' INTERFACE CLEAR
    IF (ibsta% AND EERR) THEN
        CALL prtterr("ibsic error")
        CALL ibonl(bd%, 0)
        STOP
    END IF

    CALL ibsre(bd%, 1)            ' REMOTE ENABLE
    IF (ibsta% AND EERR) THEN
        CALL prtterr("ibsre error")
        CALL ibonl(bd%, 0)
        STOP
    END IF

END SUB

' This routine continue the N.A BASIC PROGRAM
'
SUB nacont (dv%) STATIC

    PRINT "CONNECT DUT"
    INPUT "IF OK THEN PRESS ANY KEY", key$
    cmd$ = "@CONT"
    CALL ibwrt(dv%, cmd$)

END SUB

' This routine open N.A and initialize
'
```

```
SUB nainit (bd%, dvname$, dv%) STATIC

    CALL ibfind(dvname$, dv%)
    IF (dv% < 0) THEN
        CALL prtterr("ibfind error")
        CALL ibonl(bd%, 0)
        STOP
    END IF

    cmd$ = "OLDC OFF"
    CALL ibwrt(dv%, cmd$)
    IF (ibsta% AND EERR) THEN
        CALL prtterr("ibwrt error")
        CALL ibonl(dv%, 0)
        CALL ibonl(bd%, 0)
        STOP
    END IF

END SUB

' This routine receives data and print
'
SUB nareceive (dv%) STATIC

    mask% = &H4800
Nawait:
    CALL ibwait(dv%, mask%)
    CALL ibrsp(dv%, spr%)
    spr% = spr% AND 1
    IF (spr% = 0) GOTO Nawait

    str1$ = SPACE$(23)
    str2$ = SPACE$(23)
    str3$ = SPACE$(23)
    str4$ = SPACE$(23)
    str5$ = SPACE$(23)
    str6$ = SPACE$(23)

    CALL ibrd(dv%, str1$)
    CALL ibrd(dv%, str2$)
    CALL ibrd(dv%, str3$)
    CALL ibrd(dv%, str4$)
    CALL ibrd(dv%, str5$)
    CALL ibrd(dv%, str6$)

    PRINT USING "C.F = ###.##### [MHz]"; VAL(str3$) / 10^6
    PRINT USING "L.F = ###.##### [MHz]"; VAL(str1$) / 10^6
    PRINT USING "R.F = ###.##### [MHz]"; VAL(str2$) / 10^6
```

```

PRINT USING "BW = ####.##### [MHz]"; VAL(str4$) / 10^6
PRINT USING "QF = ####.#####"; VAL(str5$)
PRINT USING "SF = ####.#####"; VAL(str6$)

END SUB

' This routine setups
'
SUB nasetup (dv%) STATIC

    cmd$ = "@STOP"
    CALL ibwrt(dv%, cmd$)
    CALL ibwrt(dv%, cmd$)
    cmd$ = "@SCRATCH"
    CALL ibwrt(dv%, cmd$)
    cmd$ = "@LOAD " + CHR$(34) + "A:/NSEND.BAS" + CHR$(34)
    CALL ibwrt(dv%, cmd$)
    cmd$ = "@RUN"
    CALL ibwrt(dv%, cmd$)

END SUB

' This routine prints the result of status variables.
'
SUB prtterr (msg$) STATIC

    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 " EEND";
    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 RREM THEN PRINT " RREM";
    IF ibsta% AND CIC THEN PRINT " CIC";
    IF ibsta% AND AATN THEN PRINT " AATN";
    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 listner>"

```

(5/5)

```
IF iberr% = EADR THEN PRINT " EADR <Address error>"
IF iberr% = EARG THEN PRINT " EARG <Invalid argment>"
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 <Fils 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%

END SUB
```

12.6.5 ANSI-Cでの受信プログラム

C を使用した受信プログラム例を以下に示します。

例 12 - 30 C での受信プログラム

(1/5)

```
/*
 * CONTROL AND RECEIVE DATA
 *
 * TARGET: PC/AT(NI-488.2)
 * LANGUAGE: C (ANSI-C STYLE)
 * FILE: MCREC.C
 */
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include "decl.h"

/* prtterr - print gpib error message and status code
 */
static void prtterr(char *msg)
{
    printf("%s\n", msg);
    printf("ibsta=&H%x < ", ibsta);
    if (ibsta & ERR) printf("ERR");
    if (ibsta & TIMO) printf("TIMO");
    if (ibsta & SRQI) printf("SRQI");
    if (ibsta & RQS ) printf("RQS");
    if (ibsta & CMPL) printf("CMPL");
    if (ibsta & LOK ) printf("LOK");
}
```

(2/5)

```

if (ibsta & CIC ) printf("CIC");
if (ibsta & TACS) printf("TACS");
if (ibsta & LACS) printf("LACS");
if (ibsta & DTAS) printf("DTAS");
if (ibsta & DCAS) printf("DCAS");
printf(" >\n");

printf("iberr= %d ", iberr);
switch (iberr)
{
    case EDVR: printf("EDVR <DOS Error>"); break;
    case ECIC: printf("ECIC <Not CIC>"); break;
    case ENOL: printf("ENOL <No listner>"); break;
    case EADR: printf("EADR <Address error>"); break;
    case EARG: printf("EARG <Invalid argment>"); break;
    case ESAC: printf("ESAC <Not Sys Ctrlr>"); break;
    case EABO: printf("EABO <Op. aborted>"); break;
    case ENEB: printf("ENEB <No GPIB board>"); break;
    case EOIP: printf("EOIP <Async I/O in prg>"); break;
    case ECAP: printf("ECAP <No capability>"); break;
    case EFSO: printf("EFSO <Fils sys. error>"); break;
    case EBUS: printf("EBUS <Command error>"); break;
    case ESTB: printf("ESTB <Status byte lost>"); break;
    case ESRQ: printf("ESRQ <SRQ stuck on>"); break;
    case ETAB: printf("ETAB <Table Overflow>"); break;
}
printf("ibcntl= %d\n\n", ibcntl);
}

/* gpinit - open gpib board and initialize
*/
static int gpinit(char *bdname)
{
    int  bd;

    if ((bd = ibfind(bdname)) < 0)          /* open board */
    {
        prterr("ibfind error");
        return -1;
    }

    if (ibsic(bd) & ERR)                   /* interface clear */
    {
        prterr("ibsic error");
        ibonl(bd, 0);
        return -1;
    }
}

```

(3/5)

```
if (ibsre(bd, 1) & ERR) /* remote enable */
{
    prtterr("ibsre error");
    ibonl(bd, 0);
    return -1;
}
return bd; /* return descriptor */
}

/* nainit - open N.A Port and initialize */
*/
static int nainit(char *dvname)
{
    int dv;

    if ((dv = ibfind("DEV11")) < 0) /* open R3752/R3753 */
    {
        prtterr("ibfind error");
        return -1;
    }

    ibwrt(dv, "OLDC OFF", 8); /* default command */
    if (ibsta & ERR)
    {
        prtterr("ibwrt error");
        ibonl(dv, 0);
        return -1;
    }
    return dv; /* return descriptor */
}

/* nasetup - setups
*/
static int nasetup(int dv)
{
    ibwrt(dv, "@STOP", 5);
    ibwrt(dv, "@STOP", 5);
    ibwrt(dv, "@SCRATCH", 8);
    ibwrt(dv, "@LOAD \"A:/NSEND.BAS\"", 20);
    ibwrt(dv, "@RUN", 4);
    return 0;
}

/* nacont - continue INTERNAL BASIC
*/
```

```
static int nacont(int dv)
{
    int f;
    char c;

    printf("CONNECT DUT\n")
    printf("IF OK THEN PRESS KEY [y/n] ");
    fflush(stdout);

    while (1)
    {
        c = getchar();
        if (c == EOF)
        {
            f = -1;
            break;
        }
        if (c == 'y' || c == 'Y')
        {
            f = 0;
            break;
        }
        if (c == 'n' || c == 'N')
        {
            f = -1;
            break;
        }
    }
    fflush(stdin);
    ibwrt(dv, "@CONT", 5);
    return f;
}

/* nareceive - receives trace data
*/
static int nareceive(int na, int bd)
{
    char buf[6][24];
    int i;
    char s;

    while (1)
    {
        ibwait(na, (int)0x4800);
        ibrsp(na, &s);
        if (s & 1) break;          /* check user request bit */
    }
}
```

```
ibcmd(bd, "?_K", 4); /* UNL UNT MLA 0 TAD 11 */
for (i = 0; i < 6; i++)
{
    ibrd(bd, &buf[i][0], 23);
    buf[i][ibcnt1] = '\0';
}
ibcmd(bd, "?_+@", 4); /* UNL UNT MLA 11 TAD 0 */

printf("C.F = %4.6f [MHz]\n", atof(&buf[2][0])/1.0e6);
printf("L.F = %4.6f [MHz]\n", atof(&buf[0][0])/1.0e6);
printf("R.F = %4.6f [MHz]\n", atof(&buf[1][0])/1.0e6);
printf("BW = %4.6f [MHz]\n", atof(&buf[3][0])/1.0e6);
printf("QF = %4.6f\n", atof(&buf[4][0]));
printf("SF = %4.6f\n", atof(&buf[5][0]));

return 6;
}

/* main entry
*/
main(int argc, char **argv)
{
    int bd, na;
    int num;

    if ((bd = gpinit("GPIBO")) == -1)
        exit(1);
    if ((na = nainit("DEV11")) == -1)
    {
        ibonl(bd, 0);
        exit(1);
    }

    nasetup(na);
    while (1)
    {
        if (nacont(na) == -1) break;
        nareceive(na, bd);
    }
    ibonl(na, 0);
    ibonl(bd, 0);
}
```


12.7 BASICプログラムのダウン・ロード

この節では、本体用のプログラムを外部コントローラから本体側にダウン・ロードして実行する方法について述べます。

また、[12.8節]では、本体からプログラムをアップ・ロードする方法についても述べます。

最初に、ダウン・ロードとアップ・ロードについて簡単に説明します。

●ダウン・ロード

あらかじめ外部コントローラを利用して、本体のプログラム・ファイルを作り、フロッピー・ディスクにセーブします。

そのプログラム・ファイルを、 GPIB経由で本体のメモリに転送することをダウン・ロード(Download)といいます。

●アップ・ロード

ダウン・ロードとは反対に、本体側のメモリにあるプログラムを GPIB経由で外部コントローラのメモリに転送することをアップ・ロード(Upload)といいます。

ダウン・ロードを利用すると、外部コントローラで内蔵 BASIC用のプログラム・ファイルを管理している場合、フロッピー・ディスクを用いずにプログラムをロードすることができます。つまり、内蔵 BASICを、他の GPIBコマンドと同様に制御することができます。

ここで作るプログラムは、内蔵 BASICで実行するプログラムをダウン・ロードし、その後で外部コントローラ上で実行するプログラムと入れ換えて実行します。

[例12-26]、[例12-27]の 2つのプログラムを利用することができます。

【プログラムの概要】

- (1) 外部コントローラを初期化。
- (2) ダウン・ロードするプログラム・ファイルをオープンし、その内容を本体へ転送。
- (3) ダウン・ロードした後で、外部コントローラで実行するプログラムを読み込んで実行。

12.7.1 N88-BASIC でのダウン・ロード・プログラム

PC-9801 で実行するダウン・ロード・プログラムを以下に示します。

例 12 - 31 N88-BASIC 用ダウン・ロード・プログラム (1/2)

```

1000 ' *****
1010 ' *
1020 ' *          DOWN LOAD PROGRAM          *
1030 ' *
1040 ' * TARGET:PC-9801
1050 ' * FILE:  NDOWNLD.BAS
1060 ' *****
1070 NA=11
1080 ISET IFC
1090 ISET REN
1100 CMD DELIM=2
1110 CMD TIMEOUT=3
1120 '
1130 ON ERROR GOTO *ERRORMES
1140 PRINT "PROGRAM TRANSFER (PC to NA)"
1150 '
1160 *DNLD. ENTER
1170     FLAG=0
1180     INPUT "ENTER DOWNLOAD PROGRAM ";F$
1190     OPEN F$ FOR INPUT AS #1
1200     IF FLAG=1 THEN *DNLD. ENTER
1210     PRINT @NA;"@SCRATCH" @
1220 '
1230 *DNLD. LOOP
1240     LINE INPUT #1, DB$
1250     DB$="@"+DB$
1260     PRINT DB$
1270     PRINT @NA;DB$ @
1280     IF EOF(1) THEN *DNLD. EXIT ELSE *DNLD. LOOP
1290 '
1300 *DNLD. EXIT
1310     CLOSE
1320     PRINT "COMPLETE DOWNLOAD"
1330 '
1340 *EXEC. ENTER
1350     FLAG=0
1360     INPUT "ENTER STARTING PROGRAM ";F$
1370     IF F$="" THEN END
1380     OPEN F$ FOR INPUT AS #1
1390     IF FLAG=1 THEN *EXEC. ENTER
1400     CLOSE
1410     ON ERROR GOTO 0
1420     RUN F$
1430     END

```

(2/2)


```
1440 '
1450 *ERRORMES
1460     FLAG=1
1470     PRINT "ERROR: LINE=";ERL;" NO.=";ERR
1480     INPUT "RETRY? (Y/N)";A$
1490     IF A$="Y" OR A$="y" THEN RESUME NEXT
1500     ON ERROR GOTO 0
1510     END
```

このプログラムを PC-9801の BASICモードで入力します。
入力したらフロッピー・ディスクに保存して下さい。

プログラムの実行は、以下の手順で行います。

【実行手順】

- ① ダウン・ロードするプログラムを作成してフロッピー・ディスクに保存します。
ここでは、[例12-26]を使います。
- ② 外部コントローラで制御するプログラムを作成してフロッピー・ディスクに保存します。
ここでは、[例12-27]の一部を修正して使います。
1190行を 1190 ! PRINT NA;L\$ としてコメント行にします。
- ③ ダウン・ロード・プログラムを実行します。
ENTER DOWNLOAD PROGRAMと表示されて入力待ち状態になります。
- ④ ここで、ダウン・ロードするプログラムのファイル名（例えば、NSEND.BAS など）
を入力します。


ファイル名を入力して  を押すと、ファイルを読み込んでダウン・ロードを開

始します。

ダウン・ロードはプログラムの 1行ごとに行われ、転送している行を確認のため画
面上に表示します。

ダウン・ロードが終了すると、COMPLETE DOWNLOAD が画面に表示されます。

- ⑤ 次に、ENTER STARTING PROGRAMと表示されるので、外部コントローラで実行する制
御プログラムのファイル名を入力します。
入力されたプログラムが、今のダウン・ロード・プログラムと入れ換わって実行さ
れます。

ダウン・ロードだけなら、単に  を押します。

実行結果は、[例12-27]と同様になります。

12.7.2 HP-BASICでのダウン・ロード・プログラム

HP-BASICで実行するダウン・ロード・プログラムを以下に示します。

例 12 - 32 HP-BASIC用ダウン・ロード・プログラム

```
1000 ! *****
1010 ! *
1020 ! *          DOWN LOAD PROGRAM          *
1030 ! *
1040 ! * TARGET:HP-BASIC                    *
1050 ! * FILE:  HPDNLD.BAS                  *
1060 ! *****
1070 !
1080 ASSIGN @Na TO 711
1090 DIM Line$[512]
1100 !
1110 PRINT "PROGRAM TRANSFER (HP-9000 TO NA)"
1120 INPUT "ENTER DOWNLOAD PROGRAM ";Name$
1130 OUTPUT @Na;"*RST"
1140 OUTPUT @Na;"@SCRATCH"
1150 !
1160 ON ERROR GOTO Done
1170 ASSIGN @File TO Name$
1180 !
1190 LOOP
1200     Line$=""
1210     ENTER @File;Line$                    ! READ ONE LINE
1220     OUTPUT @Na;"@"+Line$                ! TRANSFER ONE LINE
1230 END LOOP
1240 !
1250 Done: !                                ! END OF FILE
1260     OFF ERROR
1270     ASSIGN @File TO *                    ! CLOSE FILE
1280 END
1290
1300
```

12.7.3 QuickBASICでのダウン・ロード・プログラム

QuickBASICで実行するダウン・ロード・プログラムを以下に示します。

例 12 - 33 QuickBASIC用ダウン・ロード・プログラム (1/3)

```

' *****
' *
' *          DOWN LOAD PROGRAM          *
' *
' * TARGET:   PC/AT(NI-488.2)           *
' * LANGUAGE: QuickBASIC                *
' * FILE:     QBDNLD.BAS                 *
' *****

REM $INCLUDE: 'qbdecl.bas'

DECLARE SUB gpinit (bdname$, bd%)
DECLARE SUB nainit (bd%, naname$, dv%)
DECLARE SUB prtterr (msg$)

DIM LineBuffer$(512)
DIM cmd$(512)

PRINT "PROGRAM TRANSFER (PC/AT TO NA)"
INPUT "ENTER DOWNLOAD PROGRAM "; Name$
CALL gpinit("GPIBO", bd%)
CALL nainit(bd%, "DEV11", na%)
OPEN Name$ FOR INPUT AS #1

DO UNTIL EOF(1)
    LINE INPUT #1, LineBuffer$ ' READ ONE LINE
    PRINT LineBuffer$          ' PRINT TO DISPLAY
    cmd$ = "@" + LineBuffer$
    CALL ibwrt(na%, cmd$)      ' TRANSFER PROGRAM TO NA
LOOP

CLOSE #1
CALL ibonl(na%, 0)
CALL ibonl(dv%, 0)
END

' This routine open the gpib board and initialize
'
SUB gpinit (bdname$, bd%) STATIC

    CALL ibfind(bdname$, bd%) ' OPEN BOARD
    IF (bd% < 0) THEN
        CALL prtterr("ibfind error")
        STOP
    END IF

```

(2/3)

```
CALL ibsic(bd%)          ' INTERFACE CLEAR.
IF (ibsta% AND EERR) THEN
    CALL prtterr("ibsic error")
    CALL ibonl(bd%, 0)
    STOP
END IF

CALL ibsre(bd%, 1)      ' REMOTE ENABLE
IF (ibsta% AND EERR) THEN
    CALL prtterr("ibsre error")
    CALL ibonl(bd%, 0)
    STOP
END IF

END SUB

' This routine open N.A and initialize
SUB nainit (bd%, dvname$, dv%) STATIC

    CALL ibfind(dvname$, dv%)
    IF (dv% < 0) THEN
        CALL prtterr("ibfind error")
        CALL ibonl(bd%, 0)
        STOP
    END IF

    cmd$ = "OLDC OFF;*RST"
    CALL ibwrt(dv%, cmd$)
    IF (ibsta% AND EERR) THEN
        CALL prtterr("ibwrt error")
        CALL ibonl(dv%, 0)
        CALL ibonl(bd%, 0)
        STOP
    END IF

    cmd$ = "@SCRATCH"
    CALL ibwrt(na%, cmd$)

END SUB

' This routine prints the result of status variables.
SUB prtterr (msg$) STATIC

    PRINT msg$
    PRINT "ibsta=&H"; HEX$(ibsta%); " <";
```

(3/3)

```
IF ibsta% AND EERR THEN PRINT " ERR";
IF ibsta% AND TIMO THEN PRINT " TIMO";
IF ibsta% AND EEND THEN PRINT " EEND";
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 RREM THEN PRINT " RREM";
IF ibsta% AND CIC THEN PRINT " CIC";
IF ibsta% AND AATN THEN PRINT " AATN";
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 listner>"
IF iberr% = EADR THEN PRINT " EADR <Address error>"
IF iberr% = EARG THEN PRINT " EARG <Invalid argment>"
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 <Fils 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%
```

END SUB

12.7.4 Cでのダウン・ロード・プログラム

Cで実行するダウン・ロード・プログラムを以下に示します。

例 12 - 34 C用ダウン・ロード・プログラム

(1/4)

```

/*
 *      DOWN LOAD PROGRAM
 *
 * TARGET:  PC/AT(NI-488.2)
 * LANGUAGE: C (ANSI-C STYLE)
 * FILE:    MCDNLD.C
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include "decl.h"

static char line[512];

/* prtterr - print gpib error message and status code
 */
static void prtterr(char *msg)
{
    printf("%s\n", msg);
    printf("ibsta=&H%x < ", ibsta);
    if (ibsta & ERR) printf("ERR");
    if (ibsta & TIMO) printf("TIMO");
    if (ibsta & SRQI) printf("SRQI");
    if (ibsta & RQS ) printf("RQS");
    if (ibsta & CMPL) printf("CMPL");
    if (ibsta & LOK ) printf("LOK");
    if (ibsta & CIC ) printf("CIC");
    if (ibsta & TACS) printf("TACS");
    if (ibsta & LACS) printf("LACS");
    if (ibsta & DTAS) printf("DTAS");
    if (ibsta & DCAS) printf("DCAS");
    printf(" >\n");

    printf("iberr= %d ", iberr);
    switch (iberr)
    {
        case EDVR: printf("EDVR <DOS Error>"); break;
        case ECIC: printf("ECIC <Not CIC>"); break;
        case ENOL: printf("ENOL <No listner>"); break;
        case EADR: printf("EADR <Address error>"); break;
        case EARG: printf("EARG <Invalid argment>"); break;
        case ESAC: printf("ESAC <Not Sys Ctrlr>"); break;
    }
}

```


(2/4)

```

case EABO: printf("EABO <Op. aborted>"); break;
case ENEB: printf("ENEB <No GPIB board>"); break;
case EOIP: printf("EOIP <Async I/O in prg>"); break;
case ECAP: printf("ECAP <No capability>"); break;
case EFSO: printf("EFSO <Fils sys. error>"); break;
case EBUS: printf("EBUS <Command error>"); break;
case ESTB: printf("ESTB <Status byte lost>"); break;
case ESRQ: printf("ESRQ <SRQ stuck on>"); break;
case ETAB: printf("ETAB <Table Overflow>"); break;
}
printf("ibcntl= %d\n\n", ibcntl);
}

/* gpinit - open gpib board and initialize
*/
static int gpinit(char *bdname)
{
    int    bd;

    if ((bd = ibfind(bdname)) < 0)          /* open board */
    {
        prterr("ibfind error");
        return -1;
    }

    if (ibsic(bd) & ERR)                   /* interface clear */
    {
        prterr("ibsic error");
        ibonl(bd, 0);
        return -1;
    }

    if (ib sre(bd, 1) & ERR)               /* remote enable */
    {
        prterr("ib sre error");
        ibonl(bd, 0);
        return -1;
    }

    return bd;                             /* return descriptor */
}

/* nainit - open N.A port and initialize
*/
static int nainit(char *dvname)
{
    int    na;

```

(3/4)

```
if ((na = ibfind(dvname)) < 0)      /* open R3752/R3753 */
{
    prterr("ibfind error");
    return -1;
}

ibwrt(na, "OLDC OFF;*RST", 14);    /* default command */
if (ibsta & ERR)
{
    prterr("ibwrt error");
    ibonl(na, 0);
    return -1;
}

ibwrt(na, "@SCRATCH", 9);
return na;
}                                     /* return descriptor */

/* main entry
*/
main(int argc, char **argv)
{
    char name[64], *s;
    FILE *fp;
    int bd, na;
    int n;

    printf("PROGRAM TRANSFER (PC/AT to NA)\n");
    printf("Enter DOWNLOAD PROGRAM ? ");
    fflush(stdout);
    fflush(stdin);
    if (scanf("%s", name) <= 0)
    {
        fprintf(stderr, "File name error\n");
        exit(1);
    }

    if ((bd = gpinit("GPIB0")) == -1)
        exit(1);
    if ((na = nainit("DEV11")) == -1)
    {
        ibonl(bd, 0);
        exit(1);
    }

    if ((fp = fopen(name, "r")) == NULL)
    {
```

(4/4)

```
fprintf(stderr, "%s: not found\n", name);
ibonl(na, 0);
ibonl(bd, 0);
exit(1);
}

line[0] = '@';
while ((s = fgets(&line[1], 510, fp)) && *s != NULL)
{
    printf("%s", line);
    ibwrt(na, line, strlen(line));
}

fclose(fp);
ibonl(na, 0);
ibonl(bd, 0);
}
```

12.8 BASICプログラムのアップ・ロード

この節では、本体側のメモリ上にあるプログラムを、外部コントローラへアップ・ロードする方法について述べます。

アップ・ロードするプログラムの外部コントローラ側でコントロールするので、使用方法はダウン・ロード・プログラムとほぼ同じですが、以下の点について注意して下さい。

- (注) この例で述べるアップ・ロード・プログラムで、本体のプログラムをアップ・ロードする場合、本体のプログラムの最後の行を 65535 END として下さい。これは、アップ・ロードするプログラムの最終行を判別するためです。

N88-BASIC で実行するアップ・ロード・プログラム例を以下に示します。

例 12 - 35 N88-BASIC 用アップ・ロード・プログラム

```

1000 ' *****
1010 ' * *
1020 ' *          UP LOAD PROGRAM *
1030 ' * *
1040 ' * TARGET:PC-9801 *
1050 ' * FILE:  NDOWNLD.N88 *
1060 ' *****
1070 NA=11
1080 ISET IFC
1090 ISET REN
1100 CMD DELIM=0
1110 CMD TIMEOUT=3
1120 '
1130 ON ERROR GOTO *ERRORMES
1140 PRINT "PROGRAM UPLOAD (NA to PC)"
1150 '
1160 *UPLD. ENTER
1170     FLAG=0
1180     INPUT "ENTER NEW FILE NAME ";F$
1190     OPEN F$ FOR OUTPUT AS #1
1200     IF FLAG=1 THEN *UPLD. ENTER
1210 '
1220     PRINT "UpLoading ... (Saving ";F$;" )"
1230     PRINT @NA;"@GLIST"
1240 '
1250 *UPLD. LOOP
1260     LINE INPUT @NA;DA$
1270     PRINT DA$
1280     PRINT #1, DA$
1290     IF DA$<>"65535 END" THEN *UPLD. LOOP
1300     CLOSE
1310     PRINT "COMPLETE UPLOAD"
1320     ON ERROR GOTO 0
1330     END
1340 '
1350 *ERRORMES
1360     FLAG=1
1370     PRINT "ERROR: LINE=";ERL;" NO.=";ERR
1380     INPUT "RETRY? (Y/N)";A$
1390     IF A$="Y" OR A$="y" THEN RESUME NEXT
1400     ON ERROR GOTO 0
1410     END

```

12.9 補正データの転送

この節では、2ポート・フルキャリブレーションの全補正データを、外部コントローラとの間で入出力するプログラムの例を示します。

補正データの転送は、トレース（波形）データと同様な方法で行います。データの転送フォーマットには、トレース・データと同様に、ASCII とバイナリがあります。バイナリ・フォーマットでは、より高速な転送を実現できます。

ここでは、バイナリ・フォーマットを用いた転送プログラムの例を、N88-BASIC および C で示します。

12.9.1 本体とPC-9801 との間の補正データ転送(N88-BASIC)

このプログラムは、本体とPC-9801 との間で、補正データの送受信を行うものです。

(注) NEC 社純正の GPIB インタフェース・ボードを使用します。

データ受信では、本体からPC-9801 に補正データを転送し、PC-9801 上のディスク・ドライブに保存します。あらかじめ、2ポート・フルキャリブレーションを行っておいて下さい。

一方、データ送信では、そのディスク・ドライブ上のデータを読み込んで、本体に転送します。ポイント数などの設定条件は、保存したときと同じにしておく必要があります。

このプログラムを実行すると、1:Receive(SAVE), 2:Send(LOAD) ? と表示されます。データを受信（保存）するときは 1を、送信（再生）するときは 2を入力して下さい。次に File name = ? と表示されますので、ファイル名を入力して下さい。

例 12-36 本体とPC-9801 との間の補正データ転送（バイナリ・フォーマット）
(1/3)

```

1000 ' *****
1010 '
1020 '     TRANSFER 2PORT CAL. DATA
1030 '
1040 ' *****
1050 '
1060 CLEAR &H100
1070 DEF SEG=SEGPTR(2)
1080 DIM TR1!(1201*2+4)
1090 '
1100 GOSUB *SETUP.GPIBCALL
1110 ISET IFC: ISET REN
1120 CMD DELIM=3
1130 PC98=IEEE(1) AND &H1F          ' my GPIB address
1140 NA=11                          ' target GPIB address
1150 BITLEN%=32                     ' bit length (32 or 64)
1160 PRINT @NA;"OLDC OFF" @
1170 PRINT @NA;"SWE:POIN?" @

```

(2/3)

```

1180 INPUT @NA;POINTS%
1190 PRINT @NA;"FORM MBIN,"+STR$(BITLEN%) @
1200 DT=12 ' number of traces
1210 NUMSET%=2*POINTS%*(BITLEN%/8)+9
1220 '
1230 *FORM. DATA
1240 DATA EDF,ESF,ERF,ELF,ETF,EXF
1250 DATA EDR,ESR,ERR,ELR,ETR,EXR
1260 '
1270 CLS
1280 INPUT "1:Receive(SAVE), 2:Send(LOAD) ";MODE
1290 IF MODE<1 OR 2<MODE THEN END
1300 INPUT "File name = ";FILENAME$
1310 ON MODE GOSUB *SAVE.CALDATA,*LOAD.CALDATA
1320 END
1330 '
1340 ' save full calibration data
1350 *SAVE.CALDATA
1360 RESTORE *FORM. DATA
1370 OPEN FILENAME$ FOR OUTPUT AS #1
1380 FOR J=1 TO DT
1390 READ FORM$
1400 GOSUB *RECEIVE.TRACE
1410 GOSUB *WRITE.TRACE
1420 NEXT
1430 CLOSE #1
1440 RETURN
1450 '
1460 ' load full calibration data
1470 *LOAD.CALDATA
1480 RESTORE *FORM. DATA
1490 OPEN FILENAME$ FOR INPUT AS #1
1500 FORM$="DATA":GOSUB *RECEIVE.TRACE
1510 FOR J=1 TO DT
1520 READ FORM$
1530 GOSUB *READ.TRACE
1540 GOSUB *SEND.TRACE
1550 NEXT
1560 CLOSE #1
1570 PRINT @NA;"CORR:CSET:STAT ON" @
1580 RETURN
1590 '
1600 ' receive data on one trace
1610 *RECEIVE.TRACE
1620 PRINT @NA;"TRAC:DATA? "+FORM$ @ ' trace data read
1630 WBYTE &H3F,&H5F,&H40+NA,&H20+PC98; ' set TALKER/LISTENER
1640 NUM%=NUMSET% ' read buffer size

```

(3/3)

```
1650 CALL RECEIVE.DATA(TR1!(0),NUM%)      ' read data
1660 RETURN
1670 '
1680 ' send data on one trace
1690 *SEND.TRACE
1700 NUM%=NUMSET%
1710 PRINT @NA;"TRAC:DATA "+FORM$+" , "
1720 CALL SEND.DATA(TR1!(0),NUM%)
1730 RETURN
1740 '
1750 ' write trace data into the file
1760 *WRITE.TRACE
1770 FOR I=0 TO 2*POINTS%-1
1780 PRINT #1,TR1!(I+2)
1790 NEXT I
1800 RETURN
1810 '
1820 ' read trace data from the file
1830 *READ.TRACE
1840 FOR I=0 TO 2*POINTS%-1
1850 INPUT #1,TR1!(I+2)
1860 NEXT I
1870 RETURN
1880 '
1890 ' setup system calls
1900 *SETUP.GPIBCALL
1910 RECEIVE.DATA=&H0: SEND.DATA=&H39
1920 RESTORE *GPIB.BIOS
1930 FOR ADR=0 TO &H65
1940 READ BYTE$: POKE ADR,VAL("&H"+BYTE$)
1950 NEXT
1960 RETURN
1970 '
1980 *GPIB.BIOS
1990 DATA 50,51,52,06,56,57,55,53, 8B,4F,02,8E,C1,8B,37,26
2000 DATA 8B,0C,8B,7F,04,8E,47,06, BB,00,00,BE,00,00,B0,80
2010 DATA B4,05,CD,D1,5B,53,8B,4F, 02,8E,C1,8B,37,26,89,14
2020 DATA 5B,5D,5F,5E,07,5A,59,58, CF,50,51,52,06,56,57,55
2030 DATA 53,8B,4F,02,8E,C1,8B,37, 26,8B,0C,8B,7F,04,8E,47
2040 DATA 06,BB,00,00,BE,00,00,B0, 80,B4,04,CD,D1,5B,5D,5F
2050 DATA 5E,07,5A,59,58,CF-
```

12.9.2 本体とPC/AT との間の補正データ転送(C言語)

このプログラムは、本体とPC/AT との間で、補正データの送受信を行うものです。

データ受信では、本体からPC/AT に補正データを転送し、PC/AT 上のディスク・ドライブに保存します。あらかじめ、2ポート・フルキャリブレーションを行っておいて下さい。

一方、データ送信では、そのディスク・ドライブ上のデータを読み込んで、本体に転送します。ポイント数などの設定条件は、保存したときと同じにしておく必要があります。

このプログラムを実行すると、1:Receive(SAVE), 2:Send(LOAD) ? と表示されます。データを受信(保存)するときには1を、送信(再生)するときには2を入力して下さい。次にFile name = ? と表示されますので、ファイル名を入力して下さい。

(注) NI-488.2インタフェース・ボードおよびライブラリ関数を使用します。

例 12-37 本体とPC/AT との間の補正データ転送 (バイナリ・フォーマット) (1/5)

```

/* Transfer two port full calibration data via GPIB
 * between R376X and DOS/V PC with NI-488 GPIB board
 * FORMat[:DATA] REAL,32
 * How to compile: bcc trans.c mcib.lib
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "decl.h"                /* NI-488.2 headers <DOS> */

#define ADDRESS 11                /* target GPIB address */
#define KIND 12                   /* kinds of cal. data */
#define COMP 2                    /* 1(FDATn) or 2(others) */
#define BYTE 4                    /* 4(REAL,32) or 8(REAL,64) */
#define HEAD 8                    /* block header length */
#define FOOT 1                    /* block footer(LF) length */
#define TRAC 9                    /* command "TRAC E**," length */
#define BUFLen (HEAD + 1201 * BYTE * COMP + FOOT)
#define EOT_CONFIG 1              /* ibeot configuration value */
#define EOS_CONFIG 0              /* ibeos configuration value */

char *form[] = {
    "EDF", "ESF", "ERF", "ELF", "ETF", "EXF",
    "EDR", "ESR", "ERR", "ELR", "ETR", "EXR"
};

```


(2/5)

```
void gpib_err(int id, char *msg)
{
    if (id == -1)
        fprintf(stderr, "%s\n", msg);
    else
    {
        fprintf(stderr, "%s: ibsta=0x%x, iberr=%d, ibcnt=%d\n",
            msg, ibsta, iberr, ibcnt);
        ibonl(id, 0);
    }
    exit(-1);
}

int gpib_init(int address)
{
    int id;

    if ((id = ibdev(0, address, 0, Tls, EOT_CONFIG, EOS_CONFIG)) < 0)
        gpib_err(id, "ibdev error");

    if (ibdma(0, 1) & ERR) /* enable DMA transfer */
        gpib_err(id, "ibdma error");

    return id; /* return device identifier */
}

void gpib_end(int id)
{
    if (ibonl(id, 0) & ERR) /* interface offline */
        gpib_err(id, "ibonl error");
}

int send_buf(int id, char *buf)
{
    int len;

    len = strlen(buf);
    if (ibwrt(id, buf, (long)len) & ERR) /* IBWRT */
        gpib_err(id, "ibwrt error");
    return ibcntl; /* return actual sent bytes */
}
```

(3/5)

```

int receive_buf(int id, char *buf)
{
    int eval, count = 0;

    while (1)
    {
        if ((eval = ibrd(id, buf, (long)BUFLen)) & ERR) /* IBRD */
            gpib_err(id, "ibrd error");
        count += ibcntl; /* sum total length */
        if (eval & END) /* END or EOS detected */
            break;
    }
    return count; /* return actual received bytes */
}

float btof(char *buf) /* 32bit raw binary to float */
{
    char tmp[4];

    tmp[3] = buf[0];
    tmp[2] = buf[1];
    tmp[1] = buf[2];
    tmp[0] = buf[3];
    return *((float *)tmp);
}

void ftob(float *f, char *buf) /* float to 32bit raw binary */
{
    buf[3] = *((char *)f + 0);
    buf[2] = *((char *)f + 1);
    buf[1] = *((char *)f + 2);
    buf[0] = *((char *)f + 3);
}

void save_caldata(int id, char *buf, char *filename)
{
    FILE *fp;
    int i, j, len;

    if ((fp = fopen(filename, "w")) == NULL)
        gpib_err(-1, "File open error");
    for (j = 0; j < KIND; j++)
    {

```

(4/5)

```

        sprintf(buf, "TRAC? %3s", form[j]);
        send_buf(id, buf);
        len = receive_buf(id, buf);

        for (i = 0; i < (len - HEAD - FOOT) / BYTE; i++)
            fprintf(fp, "%f\n", btof(buf + HEAD + i * BYTE));
    }
    fclose(fp);
}

void load_caldata(int id, char *buf, char *filename)
{
    FILE *fp;
    float f;
    int i, j, pts;

    send_buf(id, "SWE:POIN?");
    receive_buf(id, buf);
    sscanf(buf, "%d", &pts);

    if ((fp = fopen(filename, "r")) == NULL)
        gpib_err(-1, "File open error");

    for (j = 0; j < KIND; j++)
    {
        sprintf(buf, "TRAC %3s, #6%06d", form[j], pts * BYTE * COMP);

        for (i = 0; i < pts * COMP; i++)
        {
            fscanf(fp, "%f", &f);
            ftob(&f, buf + TRAC + HEAD + i * BYTE);
        }
        *(buf + TRAC + HEAD + pts * BYTE * COMP) = '\n';

        if (ibwrt(id, buf, (long)(TRAC + HEAD + pts * COMP * BYTE + FOOT)) & ERR)
            gpib_err(id, "ibwrt error");
    }

    send_buf(id, "CORR:CSET:STAT ON");
}

void main(void)
{
    int id;
    char *buf, filename[20];

```

(5/5)

```
if ((buf = malloc(BUFLen)) == NULL)
    gpib__err(-1, "Memory allocation error");

id = gpib__init(ADDRESS);
send__buf(id, "OLDC OFF");
send__buf(id, "FORM REAL,32");

while (1)
{
    printf("1:Receive(SAVE), 2:Send(LOAD) ? ");
    if (strchr("12", *gets(buf)) != NULL)
        break;
}

printf("File name = ? ");
gets(filename);

switch (*buf)
{
    case '1':
        save__caldata(id, buf, filename);
        break;
    case '2':
        load__caldata(id, buf, filename);
        break;
}

gpib__end(id);
}
```

索引

| | | |
|------------------------------|---------|---------------------------|
| ————— 50音順 ————— | | |
| 【あ】 | | 【き】 |
| アクティブ・ウィンドウのスクロール | 3 - 9 | キー操作によりコマンドを実行する |
| アクティブ・ウィンドウの変更 | 3 - 8 | 基本関数 |
| アップ・ロード | 12 - 61 | 行の挿入 |
| | | 極大値の最大値と最小値 |
| 【い】 | | 【く】 |
| インストール | 1 - 2 | クリスタル共振点の測定例 |
| インストール方法 | 1 - 2 | クリスタル共振点の測定 |
| | | クリップ・ボードについて |
| 【う】 | | 【け】 |
| ウィンドウ・サイズの変更 | 3 - 8 | 減衰レベル解析関数の使用例 |
| | 7 - 4 | 減衰レベルの解析 |
| ウィンドウの使い方 | 3 - 8 | |
| 【え】 | | 【こ】 |
| エディタ画面の構成要素 | 3 - 2 | コピー |
| エディタの起動 | 2 - 12 | コマンド・ボタン |
| | 3 - 1 | コマンドを使用したBASIC 命令 |
| エディタの基本操作 | 5 - 1 | コンパイル手順(MicrosoftC) |
| エディタの終了 | 2 - 17 | |
| エディタのパッケージ | 1 - 1 | 【さ】 |
| エディタのプログラミング環境 | 2 - 13 | 最大および最小値解析関数 |
| エディット・モードにする手順 | 3 - 1 | の使用例 |
| エディット・モードの画面 | 2 - 13 | 最大および最小値解析関数 |
| エディット・モード | 2 - 12 | |
| 【お】 | | 【し】 |
| 主な外部コントローラの特長 | 12 - 4 | システムの設定 |
| | | 実行結果の画面表示 |
| 【か】 | | (クリスタル共振点の測定) |
| カーソル位置から行末までの削除 | 2 - 21 | 実行結果の画面表示 |
| 外部コントローラによるリモート・コントロール | 12 - 18 | (バンドパス・フィルタの測定) |
| 外部コントローラの使用例 | 12 - 1 | 実行結果の画面表示 |
| 各ウィンドウの働き | 3 - 8 | 実行中のデータ入力 |
| カット | 2 - 23 | ショートカットキー操作 |
| 画面の構成要素 | 3 - 2 | ショートカットキーにより |
| | | コマンドを実行する |
| | | |
| | | 【せ】 |
| | | ゼリミック・フィルタ自動測定プログラム |
| | | ゼロ位相のサーチ |
| | | 全 Sパラメータの 4画面表示 |
| | | 全リップルの解析 |

【そ】

掃引終了の検出 12 - 21
 測定値を判定するプログラム 10 - 9
 10 - 11
 測定値を判定するプログラム
 (パラレルI/O ポート使用) 10 - 15

【た】

代入命令 2 - 5
 ダイレクト・サーチ 11 - 30
 ダイレクト・サーチ 関数の使用例 11 - 30
 ダイレクト・モード 2 - 2
 対話ボックスの構成要素 2 - 15
 対話ボックスの操作 2 - 15
 対話ボックスの使い方 2 - 15
 対話ボックスの使い方 3 - 6
 対話ボックスの働き 3 - 6
 ダウン・ロード 12 - 61

【つ】

通常の GPIB コマンドの転送 12 - 18

【て】

ディスク・フォーマット を変換する 1 - 3
 ディスク変換プログラム 1 - 3
 ディレクトリ内容の一覧表示 4 - 4
 データ転送 11 - 32
 テキスト画面を見易くする方法 2 - 2
 テキスト入力フィールド 3 - 7
 テキストのインデント 5 - 1
 テキストの選択 5 - 1
 テキストの入力 5 - 1
 テキスト・ブロックのペースト 2 - 24

【と】

特定リップルの解析 11 - 28
 トレース・データの転送 12 - 26

【な】

内蔵 BASIC の送信プログラム 12 - 47
 内蔵 BASIC での送信プログラム 12 - 47
 内蔵 BASIC と外部コントローラを
 同時に使うために 12 - 47

【ね】

ネットワーク・アナライザでの自動測定 10 - 1

【は】

波形解析プログラム例 11 - 1
 はじめに 1 - 1
 パス名指定フィールド 3 - 7
 パラレルI/O ポートに出力する 10 - 13
 判定プログラムの実行結果 10 - 13
 バンドパス・フィルタの測定 11 - 11
 バンドパス・フィルタの測定例 11 - 11

【ひ】

必要な機器 1 - 1
 ビルトイン関数の使用 10 - 7
 ビルトイン関数の使い方 11 - 15
 ビルトイン関数を使用したプログラム 10 - 8
 ビルトイン関数 10 - 7

【ふ】

ファイルの指定 4 - 3
 ファイル名が見たいとき 2 - 11
 ファイルを表示する 7 - 2
 プリントへの出力 2 - 11
 プログラミングの基礎 2 - 1
 プログラミングの前に 12 - 1
 プログラム(ダイレクト)・モードの画面 2 - 1
 プログラムの書き方 12 - 4
 プログラムの作成と実行 2 - 6
 プログラムの実行 10 - 1
 プログラムの消去 2 - 7
 プログラム入力と実行 2 - 6
 プログラムの編集 2 - 18
 プログラムの保存 2 - 9
 プログラムの読み出し 2 - 11
 プログラム・モード 2 - 1
 プログラム・モード状態 2 - 2
 プログラム・モードの起動 2 - 1
 プログラム・リストを表示する 2 - 6
 プログラムを実行する 2 - 6
 プログラムを保存する 2 - 10
 ブロック編集 2 - 22
 フロッピー・ディスクを初期化する 2 - 9

【へ】

| | |
|-----------------|--------|
| 編集コマンド一覧 | 5 - 2 |
| 編集コマンドの選び方 | 2 - 14 |
| 編集コマンドの概要 | 5 - 2 |
| 編集コマンドを直接実行する方法 | 2 - 16 |

【ほ】

| | |
|--------------------------------------|---------|
| 補正データの転送 | 12 - 73 |
| 本BASICの特長 | 1 - 1 |
| 本体からCへのトレース・データ転送 | 12 - 39 |
| 本体からHP-BASICへの トレース・データ転送 | 12 - 33 |
| 本体からPC-9801への トレース・データ転送 | 12 - 26 |
| 本体からQuickBASICへの トレース・データ転送 | 12 - 35 |
| 本体とPC-9801との間の 補正データ転送(N88-BASIC) | 12 - 73 |
| 本体とPC/ATとの間の 補正データ転送(C言語) | 12 - 76 |
| 本体との接続 | 12 - 2 |

【ま】

| | |
|---------|--------|
| マーカのセット | 2 - 22 |
|---------|--------|

【め】

| | |
|-------------------|--------|
| メッセージ・ラインの使い方 | 3 - 8 |
| メニューのオープン | 2 - 13 |
| メニューのオープンとコマンドの実行 | 2 - 13 |

【も】

| | |
|---------|--------|
| 文字と行の削除 | 2 - 19 |
| 文字の挿入 | 2 - 18 |

【り】

| | |
|-------------------|---------|
| リスト・ボックス | 3 - 7 |
| リップル解析関数の使用例(1) | 11 - 21 |
| リップル解析関数の使用例(2) | 11 - 27 |
| リップル解析プログラム | 11 - 7 |
| リップルのレスポンス および周波数 | 11 - 26 |
| リミット・ラインの設定 | 11 - 33 |

【れ】

| | |
|---------------|---------|
| レスポンスに対応する帯域幅 | 11 - 31 |
|---------------|---------|

— 記号、アルファベット順 —

【記号】

| | |
|-----------------------|---------|
| "@" 付き内蔵BASIC コマンドの転送 | 12 - 19 |
|-----------------------|---------|

【A】

| | |
|-----------------|---------|
| ANSI-Cでの受信プログラム | 12 - 56 |
|-----------------|---------|

【B】

| | |
|----------------------------------|---------|
| Backspace を使っての文字の削除 | 2 - 20 |
| BASIC エディタ画面 | 3 - 1 |
| BASIC エディタの起動 | 3 - 1 |
| BASIC エディタの機能 | 3 - 1 |
| BASIC プログラムのアップ・ロード | 12 - 71 |
| BASIC プログラムのダウン・ロード | 12 - 61 |
| BND | 11 - 17 |
| BNDH | 11 - 17 |
| BNDL | 11 - 17 |
| [Buffer list...] コマンド | 7 - 1 |
| [Buffer list...] コマンドの 対話ボックス | 7 - 2 |

【C】

| | |
|------------------------|---------|
| C で掃引終了を検出する | 12 - 24 |
| C での BASIC コマンド転送 | 12 - 20 |
| C での GPIB コマンド転送 | 12 - 19 |
| C での受信プログラム | 12 - 56 |
| C でのダウン・ロード・プログラム | 12 - 68 |
| C でのデータ入力(ASCIIフォーマット) | 12 - 39 |
| C でのデータ入力(バイナリフォーマット) | 12 - 43 |
| C でのプログラムの書き方 | 12 - 14 |
| C 用ダウン・ロード・プログラム | 12 - 68 |
| CAT | 2 - 11 |
| CBNDH | 11 - 17 |
| CBNDL | 11 - 17 |
| CBND | 11 - 17 |
| CDDIRECT | 11 - 31 |
| CDIRECT | 11 - 30 |
| CDIRECTH | 11 - 30 |
| CDIRECTL | 11 - 30 |
| [Close subfile] コマンド | 4 - 10 |
| [Continue] コマンド | 9 - 2 |
| [Copy] コマンド | 6 - 2 |
| [Cut] コマンド | 6 - 2 |
| CVALUE関数を使用したプログラム | 10 - 7 |
| CVALUE | 10 - 7 |
| | 11 - 15 |

【D】

| | |
|------------------------|---------|
| DCVALUE | 11 - 15 |
| DDIRECT | 11 - 31 |
| Deleteを使っての文字の削除 | 2 - 21 |
| DFREQ | 11 - 15 |
| DIRECT | 11 - 30 |
| DIRECTH | 11 - 30 |
| DIRECTL | 11 - 30 |
| [Download] コマンド | 9 - 2 |
| DPOINT | 11 - 15 |
| DVALUE | 11 - 15 |

【E】

| | |
|--------------------------------|--------|
| ENTER 命令 | 10 - 3 |
| [Execution display] コマンド | 7 - 4 |

【F】

| | |
|-------------------------------|---------|
| F1:File メニュー | 2 - 14 |
| | 3 - 3 |
| F1:File メニュー (ファイルメニュー) | 4 - 1 |
| F2:Edit メニュー (編集メニュー) | 6 - 1 |
| F3:View メニュー (ビューメニュー) | 7 - 1 |
| F4:Search メニュー (検索メニュー) .. | 8 - 1 |
| F5:Runメニュー (実行メニュー) | 9 - 1 |
| [Find again] コマンド | 8 - 3 |
| [Find label...] コマンドの | |
| 対話ボックス | 8 - 6 |
| [Find label...] コマンド | 8 - 6 |
| [Find word] コマンド | 8 - 3 |
| [Find...] コマンドの対話ボックス | 8 - 2 |
| [Find...] コマンド | 8 - 1 |
| FMAX | 10 - 10 |
| | 11 - 16 |
| FMIN | 11 - 16 |
| FREQ | 10 - 8 |
| | 11 - 15 |
| FRPLH | 11 - 26 |
| FRPLHM | 11 - 29 |
| FRPLHN | 11 - 28 |
| FRPLL | 11 - 26 |
| FRPLLM | 11 - 29 |
| FRPLLN | 11 - 28 |

【G】

| | |
|---------------------|--------|
| GLIST | 2 - 11 |
| GPIBアドレスの設定 | 12 - 2 |
| GPIBケーブル (別売) | 12 - 2 |
| GPIBのモード | 12 - 2 |
| GPRINT USING | 10 - 6 |

【H】

| | |
|-------------------------------|---------|
| HP-BASICで掃引終了を検出する | 12 - 22 |
| HP-BASICでの BASICコマンド転送 | 12 - 19 |
| HP-BASICでの GPIBコマンド転送 | 12 - 18 |
| HP-BASICでの受信プログラム | 12 - 51 |
| HP-BASICでの受信プログラム | 12 - 51 |
| HP-BASICでの掃引終了の検出 | 12 - 22 |
| HP-BASICでのダウン・ロード・プログラム | 12 - 64 |
| HP-BASICでのデータ入力 | |
| (ASCIIフォーマット) | 12 - 33 |
| HP-BASICでのデータ入力 | |
| (バイナリ・フォーマット) | 12 - 34 |
| HP-BASICでのプログラムの書き方 | 12 - 6 |
| HP-BASIC用ダウン・ロード・プログラム | 12 - 64 |
| HP9000上での GPIBコントロール・プログラム | |
| (HP-BASIC) | 12 - 6 |

【I】

| | |
|--------------------------------|---------|
| ibcnt% | 12 - 12 |
| iberr% | 12 - 12 |
| ibfind | 12 - 11 |
| ibonl | 12 - 12 |
| ibrd | 12 - 12 |
| ibsic | 12 - 11 |
| ib sre | 12 - 12 |
| ibsta% | 12 - 12 |
| ibwrt | 12 - 12 |
| [Initialize] コマンド | 9 - 2 |
| INITIALIZE | 2 - 9 |
| [Insert...] コマンドの対話ボックス | 4 - 6 |
| [Insert...] コマンド | 4 - 6 |
| ISET IFC | 12 - 6 |
| ISET REN | 12 - 6 |

【L】

| | |
|-------------------------|--------|
| LIST | 2 - 6 |
| LLIST | 2 - 11 |
| LOAD | 2 - 11 |
| [Lower case] コマンド | 6 - 3 |

| 【M】 | 【P】 |
|---|---|
| MAX およびMIN レベル自動測定 プログラム 11 - 1 | [Paste] コマンド 6 - 2 |
| MAX 10 - 10 | PC/AT 上での GPIB コントロール・プログラム (ANSI-C) 12 - 14 |
| MBNDI 11 - 18 | PC/AT 上での GPIB コントロール・プログラム (QuickBASIC) 12 - 8 |
| MBNDO 11 - 18 | PC-9801 から本体へのトレース・データ出力 12 - 30 |
| MicrosoftCでの掃引終了の検出 12 - 24 | PC-9801 上での GPIB コントロール・プログラム (N88-BASIC) 12 - 4 |
| MIN 11 - 16 | PC-9801 でのデータ出力 (ASCIIフォーマット) 12 - 30 |
| | PC-9801 でのデータ出力 (バイナリ・フォーマット) 12 - 31 |
| 【N】 | PC-9801 でのデータ入力 (ASCIIフォーマット) 12 - 26 |
| N88-BASIC で掃引終了を検出する 12 - 21 | PC-9801 でのデータ入力 (バイナリ・フォーマット) 12 - 28 |
| N88-BASIC での BASIC コマンド転送 12 - 19 | PMAX 11 - 16 |
| N88-BASIC での GPIB コマンド転送 12 - 18 | PMIN 11 - 16 |
| N88-BASIC での受信プログラム 12 - 49 | POINT1 10 - 8 |
| N88-BASIC での受信プログラム 12 - 49 | POINT1H 11 - 15 |
| N88-BASIC での掃引終了の検出 12 - 21 | POINT1L 11 - 15 |
| N88-BASIC でのダウンロード・プログラム 12 - 62 | POINT2 11 - 15 |
| N88-BASIC でのプログラムの書き方 12 - 4 | POINT2H 11 - 15 |
| N88-BASIC 用アップロード・プログラム 12 - 72 | POINT2L 11 - 15 |
| N88-BASIC 用ダウンロード・プログラム 12 - 62 | PRINT USING 10 - 6 |
| [New subfile] コマンドの対話ボックス 4 - 8 | [Print with GPIB] コマンド 4 - 10 |
| [New subfile] コマンド 4 - 8 | [Print with SIO] コマンド 4 - 10 |
| [New] コマンド 4 - 2 | PRINT 命令 2 - 4 |
| [Next buffer] コマンド 7 - 3 | PRPLH 11 - 26 |
| NI-488.2のライブラリ およびファイル 12 - 11 | PRPLHM 11 - 29 |
| NRPLH 11 - 27 | PRPLHN 11 - 28 |
| NRPLL 11 - 27 | PRPLL 11 - 26 |
| | PRPLLH 11 - 29 |
| 【O】 | PRPLLN 11 - 28 |
| [Open subfile] コマンドの対話ボックス 4 - 9 | |
| [Open subfile] コマンド 4 - 9 | |
| [Open...] コマンド 4 - 3 | |
| [Open...] コマンドの対話ボックス 2 - 16 | |
| 3 - 6 | |
| 4 - 3 | |
| OUTPUTとENTER 命令を使用した プログラム 10 - 1 | |
| OUTPUTとENTER 命令を使用したプログラム (ビルトイン関数使用) 10 - 2 | |
| OUTPUT命令 10 - 3 | |

【Q】

| | |
|--|---------|
| qbdecl.bas | 12 - 11 |
| QuickBASICで掃引終了を検出する .. | 12 - 22 |
| QuickBASICでの BASICコマンド転送 .. | 12 - 20 |
| QuickBASICでの GPIBコマンド転送 .. | 12 - 18 |
| QuickBASICでの受信プログラム .. | 12 - 52 |
| QuickBASICでの受信プログラム .. | 12 - 52 |
| QuickBASICでの掃引終了の検出 .. | 12 - 23 |
| QuickBASICでのダウンロードプログラム .. | 12 - 65 |
| QuickBASICでのデータ入力 (ASCIIフォーマット) | 12 - 35 |
| QuickBASICでのプログラムの書き方 .. | 12 - 8 |
| QuickBASIC用ダウンロードプログラム .. | 12 - 65 |
| 〔Quit〕コマンド | 4 - 11 |

【R】

| | |
|----------------------------|---------|
| REM 命令 | 10 - 3 |
| 〔Replace〕コマンドの対話ボックス .. | 8 - 4 |
| 〔Replace...〕コマンドの対話ボックス .. | 3 - 6 |
| 〔Replace...〕コマンド | 8 - 4 |
| RPL1 | 11 - 22 |
| RPL2 | 11 - 23 |
| RPL3 | 11 - 24 |
| RPL4 | 11 - 23 |
| RPL5 | 11 - 25 |
| RPL6 | 11 - 25 |
| RPLF | 11 - 26 |
| RPLH | 11 - 26 |
| RPLL | 11 - 26 |
| RPLR | 11 - 26 |
| RUN | 2 - 6 |

【S】

| | |
|-----------------------------|---------|
| SAVE | 2 - 10 |
| 〔Save〕コマンド | 4 - 7 |
| 〔Save and exit〕コマンド | 4 - 10 |
| 〔Save as...〕コマンド | 4 - 7 |
| 〔Save as...〕コマンドの対話ボックス .. | 4 - 7 |
| SCRATCH | 2 - 7 |
| 〔Split window〕コマンド | 7 - 3 |
| 〔Start〕コマンド | 9 - 2 |
| SWFREQ | 11 - 15 |
| SWPOINT | 11 - 15 |
| SWVALUE | 11 - 15 |
| SYSTEM CONTROLLER モード | 12 - 2 |

【T】

| | |
|-------------------------|---------|
| TAKER/LISTENERモード | 12 - 2 |
| TRANSR | 11 - 33 |
| TRANSW | 11 - 33 |

【U】

| | |
|--------------------------|--------|
| 〔Upload〕コマンド | 9 - 2 |
| 〔Upper case〕コマンド | 6 - 3 |
| USING を使用した表示プログラム | 10 - 5 |
| USING を使用したプログラム | 10 - 5 |

【V】

| | |
|--------------|---------|
| VALUE | 10 - 9 |
| | 11 - 15 |
| VRPLHM | 11 - 29 |
| VRPLHN | 11 - 28 |
| VRPLLM | 11 - 29 |
| VRPLLN | 11 - 28 |

【Z】

| | |
|---------------|---------|
| ZEROPHS | 11 - 32 |
|---------------|---------|

本製品に含まれるソフトウェアのご使用について

本製品に含まれるソフトウェア（以下本ソフトウェア）のご使用について以下のことにご注意下さい。

ここでいうソフトウェアには、本製品に含まれる又は共に使用されるコンピュータ・プログラム、将来弊社よりお客様に提供されることのある追加、変更、修正プログラムおよびアップデート版のコンピュータ・プログラム、ならびに本製品に関する取扱説明書等の付随資料を含みます。

使用許諾

本ソフトウェアの著作権を含む一切の権利は弊社に帰属いたします。

弊社は、本ソフトウェアを本製品上または本製品とともに使用する限りにおいて、お客様に使用を許諾するものといたします。

禁止事項

お客様は、本ソフトウェアのご使用に際し以下の事項は行わないで下さい。

- 本製品使用目的以外で使用する事
- 許可なく複製、修正、改変を行う事
- リバース・エンジニアリング、逆コンパイル、逆アセンブルなどを行う事

免 責

お客様が、本製品を通常の用法以外の用法で使用したことにより本製品に不具合が発生した場合、およびお客様と第三者との間で著作権等に関する紛争が発生した場合、弊社は一切の責任を負いかねますのでご了承下さい。

保証について

製品の保証期間は、お客様と別段の取り決めがある場合または当社が特に指定した場合を除き、製品の納入日(システム機器については検取日)から1年間といたします。保証期間中に、当社の責めに帰する製造上の欠陥により製品が故障した場合、無償で修理いたします。ただし、下記に該当する場合は、保証期間中であっても保証の対象から除外させていただきます。

- 当社が認めていない改造または修理を行った場合
- 支給品等当社指定品以外の部品を使用した場合
- 取扱説明書に記載する使用条件を超えて製品を使用した場合(定められた許容範囲を超える物理的ストレスまたは電流電圧がかかった場合など)
- 通常想定される使用環境以外で製品を使用した場合(腐食性の強いガス、塵埃の多い環境等による電気回路の腐食、部品の劣化が早められた場合など)
- 取扱説明書または各種製品マニュアルの指示事項に従わずに使用された場合
- 不注意または不当な取扱により不具合が生じた場合
- お客様のご指示に起因する場合
- 消耗品や消耗材料に基づく場合
- 火災、天変地異等の不可抗力による場合
- 日本国外に持出された場合
- 製品を使用できなかったことによる損失および逸失利益

当社の製品の保証は、本取扱説明書に記載する内容に限られるものとします。

保守に関するお問い合わせについて

長期間にわたる信頼性の保証、国家標準とのトレーサビリティを実現するためにアドバンテスでは、工場から出荷された製品の保守に対し、カスタマ・エンジニアを配置しています。

カスタマ・エンジニアは、故障などの不慮の事故は元より、製品の長期間にわたる性能の保証活動にフィールド・エンジニアとしても活動しています。

万一、動作不良などの故障が発生した場合には、当社のMS(計測器)コールセンターにご連絡下さい。

製品修理サービス

- **製品修理期間**
製品の修理サービス期間は、製品の納入後10年間とさせていただきます。
- **製品修理活動**
当社の製品に故障が発生した場合、当社に送っていただく引取り修理、または当社技術員が現地に出張しての出張修理にて対応いたします。

製品校正サービス

- **校正サービス**
ご使用中の製品に対し、品質および信頼性の維持を図ることを目的に行うもので、校正後の製品には校正ラベルを貼付けし、品質を保証いたします。
- **校正サービス活動**
校正サービス活動は、株式会社アドバンテス カスタマサポートに送っていただく引取り校正、または当社技術員が現地に出張しての出張校正にて対応いたします。

予防保守のおすすめ

製品にはエレクトロニクス部品およびメカニカル部品の一部に寿命を考慮すべき部品を使用しているため、定期的な交換を必要とします。適正な交換期間を過ぎて使用し発生した障害に対しては、修理および性能の保証ができません場合があります。

アドバンテスでは、このようなトラブルを未然に防ぐため、予防保守が有効な手段と考え、予防保守作業を実施する体制を整えています。

各種の予防保守を定期的実施することで、製品の安定稼働を図り、不意の費用発生を防ぐため、年間保守契約による予防保守の実施をお勧めいたします。

なお、年間保守契約は、製品、使用状況および使用環境により内容が変わりますので、最寄りの弊社営業支店にお問い合わせ下さい。

ADVANTEST

<http://www.advantest.co.jp>

株式会社アドバンテス

本社事務所
〒100-0005 千代田区丸の内1-6-2 新丸の内センタービルディング
TEL: 03-3214-7500 (代)

第4アカウント販売部(東日本)
〒100-0005 千代田区丸の内1-6-2 新丸の内センタービルディング
TEL: 0120-988-971
FAX: 0120-988-973

第4アカウント販売部(西日本)
〒564-0062 吹田市垂水町3-34-1
TEL: 0120-638-557
FAX: 0120-638-568

★計測器に関するお問い合わせ先

(製品の仕様、取扱い、修理・校正等計測器関連全般)

MS(計測器)コールセンター ☎ TEL 0120-919-570
FAX 0120-057-508

E-mail: icc@acs.advantest.co.jp