ADVANTEST®

株式会社アドバンテスト

R3465 シリーズ OPT15

スペクトラム・アナライザ

プログラミング・マニュアル

MANUAL NUMBER FFJ-8311286B03

適用機種

R3465
R3272
R3263

初版 1995 年 4 月 18 日
Printed in Japan

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、誤った方法で使用すると本器の保護機能がそこなわれることがあります。常に安全に心がけてご使用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、傷つけ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。

危険警告ラベルのシグナル・ワードとその定義は、以下のとおりです。

危険： 死または重度の障害が差し迫っている。
警告： 死または重度の障害が起こる可能性がある。
注意： 軽度の人身障害あるいは物損が起こる可能性がある。

■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もの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、保護導体端子を備えた電源コンセントに接続して下さい。保護導体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（弊社の製品には添付していません）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端子に接続し、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。

●製品の上に物をのせたり、製品の上から力を加えたりしないで下さい。また、花瓶や薬品などの液体の入った容器を製品のそばに置かないで下さい。

●通気孔のある製品については、通気孔に金属類や燃えやすい物などを差し込んだり、落としたりしないで下さい。

●台車に載せて使用する場合は、ベルト等によって落下防止を行って下さい。

●周辺機器を接続する場合は、本器の電源を切ってから接続して下さい。

■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の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。


： 感電注意を示しています。

■寿命部品の交換について

計測器に使用されている主な寿命部品は以下のとおりです。
製品の性能、機能を維持するために、寿命を目安に早めに交換して下さい。
ただし、製品の使用環境、使用頻度および保存環境により記載の寿命より交換時期が早くなる場合がありますので、ご了承下さい。
なお、ユーザによる交換はできません。交換が必要な場合は、当社または代理店へご連絡下さい。

製品ごとに個別の寿命部品を使用している場合があります。
本書、寿命部品に関する記載項を参照して下さい。

主な寿命部品と寿命

部品名称	寿命
ユニット電源	5 年
ファン・モータ	5 年
電解コンデンサ	5 年
液晶ディスプレイ	6 年
液晶ディスプレイ用バックライト	2.5 年
フロッピー・ディスク・ドライブ	5 年
メモリ・バックアップ用電池	5 年

■ハード・ディスク搭載製品について

使用上の留意事項を以下に示します。

●本器は、電源が入った状態で持ち運んだり、衝撃や振動を与えないで下さい。
ハード・ディスクの内部は、情報を記録するディスクが高速に回転しながら、情報の読み書きを行っているため、非常にデリケートです。

●本器は、以下の条件に合う場所で使用および保管をして下さい。
極端な温度変化のない場所
衝撃や振動のない場所
湿気や埃・粉塵の少ない場所
磁石や強い磁界の発生する装置から離れた場所

●重要なデータは、必ずバックアップを取っておいて下さい。
取扱方法によっては、ディスク内のデータが破壊される場合があります。また、使用条件によりますが、ハード・ディスクには、その構造上、寿命があります。
なお、消失したデータ等の保証は、いたしかねますのでご了承下さい。

■本器の廃棄時の注意

製品を廃棄する場合、有害物質は、その国の法律に従って適正に処理して下さい。
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）
例： 蛍光管、バッテリ

■使用環境

本器は、以下の条件に合う場所に設置して下さい。
●腐食性ガスの発生しない場所
●直射日光の当たらない場所
●埃の少ない場所
●振動のない場所
●最大高度 2000 m

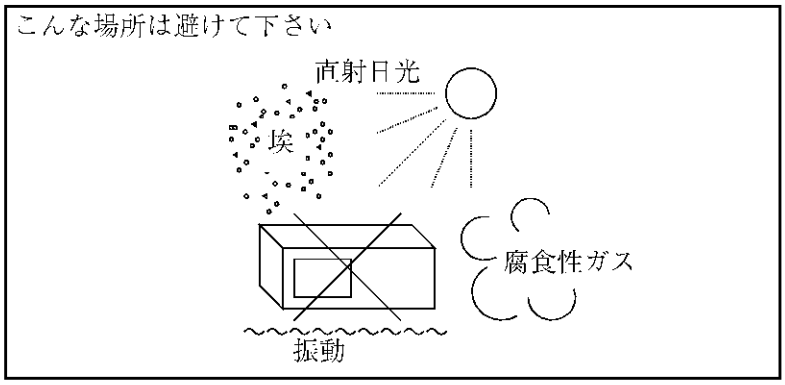


図 -1 使用環境

●設置姿勢
本器は、必ず水平状態で使用して下さい。
本器は内部温度上昇をおさえるため、強制空冷用のファンを搭載しております。
ファンの吐き出し口、通気孔をふさがないで下さい。

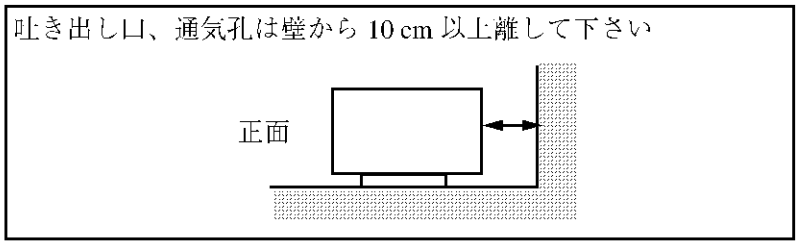


図 -2 設置

●保管姿勢
本器は、なるべく水平状態で保管して下さい。
本器を立てた状態で保管する場合、または運搬時、一時的に立てた状態で置く場合、転倒しないよう注意して下さい。衝撃・振動により転倒する恐れがあります。

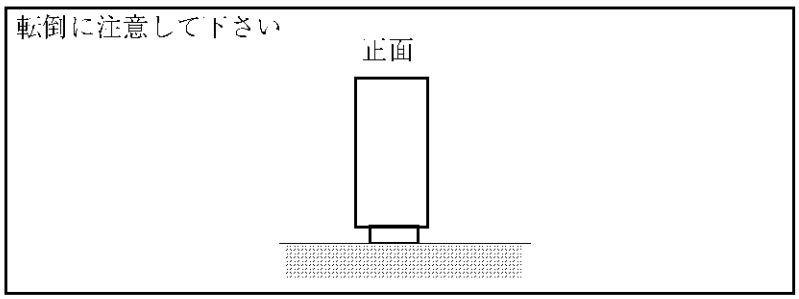
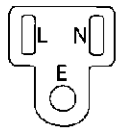
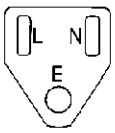
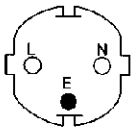
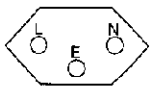
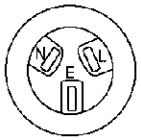
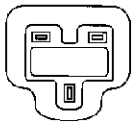
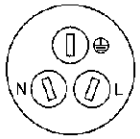


図 -3 保管

● IEC61010-1 で定義される、主電源に典型的に存在する過渡過電圧および汚染度の分類は、以下のとおりです。
IEC60364-4-443 の耐インパルス（過電圧）カテゴリ II
汚染度 2

■電源ケーブルの種類

「電源ケーブルの種類」の記述が本文中にある場合には、以下の表に置き替えてお読みください。

プラグ	適用規格	定格・色・長さ	型名（オプション No.）
	PSE: 日本 電気用品安全法	125V/7A 黒、2m	ストレート・タイプ A01402 アングル・タイプ A01412
	UL: アメリカ CSA: カナダ	125V/7A 黒、2m	ストレート・タイプ A01403 (オプション 95) アングル・タイプ A01413
	CEE: ヨーロッパ DEMKO: デンマーク NEMKO: ノルウェー VDE: ドイツ KEMA: オランダ CEBEC: ベルギー OVE: オーストリア FIMKO: フィンランド SEMKO: スウェーデン	250V/6A 灰、2m	ストレート・タイプ A01404 (オプション 96) アングル・タイプ A01414
	SEV: スイス	250V/6A 灰、2m	ストレート・タイプ A01405（オプション 97） アングル・タイプ A01415
	SAA: オーストラリア ニュージーランド	250V/6A 灰、2m	ストレート・タイプ A01406（オプション 98） アングル・タイプ ----
	BS: イギリス	250V/6A 黒、2m	ストレート・タイプ A01407（オプション99） アングル・タイプ A01417
	CCC: 中国	250V/10A 黒、2m	ストレート・タイプ A114009（オプション94） アングル・タイプ A114109

目次

図一覧

表一覧

1. はじめに

本器に内蔵されている BASIC言語は、汎用の BASICコマンド、GPIB制御用コマンドおよび専用ビルトイン関数を備え、小規模GPIBシステムを簡単に構築できます。

●コマンドとステートメントの構文について

本器で使われるコマンドとステートメントの構文は、本書の 3章と 4章で説明しますが、直観的に理解できるように図式表現と記述式表現を並記して解説しています。

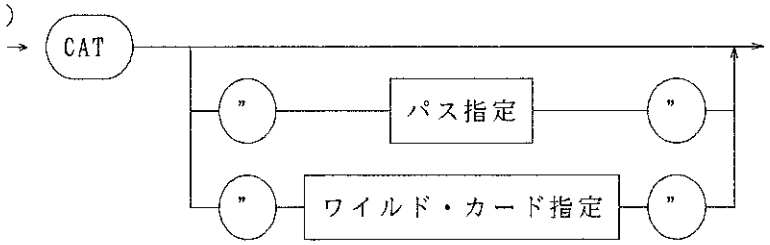
注意

各コマンドや各ステートメントの 構文 の読み方

(1) 図式表現

構文を各要素に分解し、直線で結んで表わしています。
文は、必ず矢印の方向に進みます。途中で2つ以上に分岐している場合は、それらのうちのいずれかに進みます。また、矢印の方向がループを構成している場合は、何回でもそのループを通ることができます。

記述例）



(2) 記述式表現に用いている記号の意味

- 記号〔 〕で囲まれた部分 ： 省略できる。
- 記号＜ ＞で囲まれた部分 ： 省略できない。
- 記号｛ ｝で囲まれた部分 ： 0 回以上繰り返し用いることができる。
- 記号 | ： 「または」の意味。
  （例： A | B … AまたはB を用いる。）

記述例） CAT 〔"〔ドライブ名:〕〔ディレクトリ/〕〔ワイルド・カード〕"〕

(3) 図式表現、記述式表現に用いている単語の意味

- 数値表現式 ： 数値定数、数値変数、数値配列変数、数式のいずれか。
- 文字列表現式 ： 文字列定数、文字列変数、文字列配列変数、サブ・ストリングで構成される式。
- 装置アドレス ： GPIBに接続されている装置のアドレス。

●GPIBモードについて

本器は、2 種類のモード（ADDRESSABLE モードとSYSTEM CONTROLLER モード）で動作します。モードの切り換えは、CONTROL コマンドで設定します。

CONTROL コマンドについては、[3. BASICコマンド] を参照して下さい。

(1) ADDRESSABLE モード

通常のモードで、外部コントローラによりコントロールされます。
このモードで内蔵BASIC プログラムを実行すると、以下の 2種類の動作になります。

① BASIC コマンドの「CONTROL 7;4 」が設定されていない場合

内蔵BASIC と外部コントローラの間で、データの送受信ができます。
ただし、内蔵BASIC の ENTERとOUTPUT命令が優先されるため、外部コントローラからのGPIBコマンドによる設定はできなくなります。
外部コントローラからのGPIBコマンドによる設定をしたい場合は、内蔵BASIC プログラムを停止させるか、「CONTROL 7;4 」を設定して下さい。

② BASIC コマンドの「CONTROL 7;4 」が設定されている場合

①とは逆で、外部コントーラからのGPIBコマンドによる設定ができます。
つまり、内蔵BASIC 停止中の動作と同様です。ただし、内蔵BASIC と外部コントローラの間のデータ送受信はできません。

(2) SYSTEM CONTROLLER モード

内蔵BASIC プログラムにより、測定機能および外部接続した機器をコントロールすることができます。

2.　基本操作

2.1　操作概要

BASIC プログラムのロード、実行および停止等の操作は、ソフトキーおよび外部キーボードからのコマンド・エントリで行います。また、GPIBを介して外部コンピュータによるプログラムの入力、実行等も行うこともできます。

2.2　パネル操作

2.2.1　プログラムの入力・実行・停止

BASIC プログラムの作成は、本器に外部キーボードを接続してプログラミングするか、他のパーソナル・コンピュータで作成したASCII ファイルのBASIC プログラムをメモリ・カードにセーブして行います。
作成済みのBASIC プログラムのロード、実行および停止は、以下の手順で操作して下さい。

●手順

① 実行したいBASIC プログラムがセーブされているメモリ・カードを、本器のメモリ・カード・ドライブに挿入します。

② パネルのディスプレイ下側にあるCNTRLR ON キーを押します。
BASIC 専用のソフトキー・メニューがディスプレイ上に現れます。

③ ①で挿入したメモリ・カードのデバイスをソフトキーのDeviceにて選択します。このソフトキーを押すたびに、キー内のアクティブ表示が切り換わります。このアクティブ表示は、MAがメモリ・カード Aドライブ、MBがメモリ・カード Bドライブを意味しています。

④ ソフトキーのLOADを押すと、ファイル選択ウィンドウが表示されます。ロータリ・ノブまたはステップ・キー↑、↓で、任意のファイル名を選択して下さい。

注意

BASIC 専用画面では、ファイル選択ウィンドウが見えません。 Change Screen キーを押して波形画面状態にし、 LOAD キーを押して下さい。

⑤ ロータリ・ノブまたは ENTERキーを押すと、選択したファイルがロードされます。

⑥ ソフトキーの RUNを押すと、プログラムが実行されます。

⑦　STOPキーを押すと、プログラムの実行が停止されます。

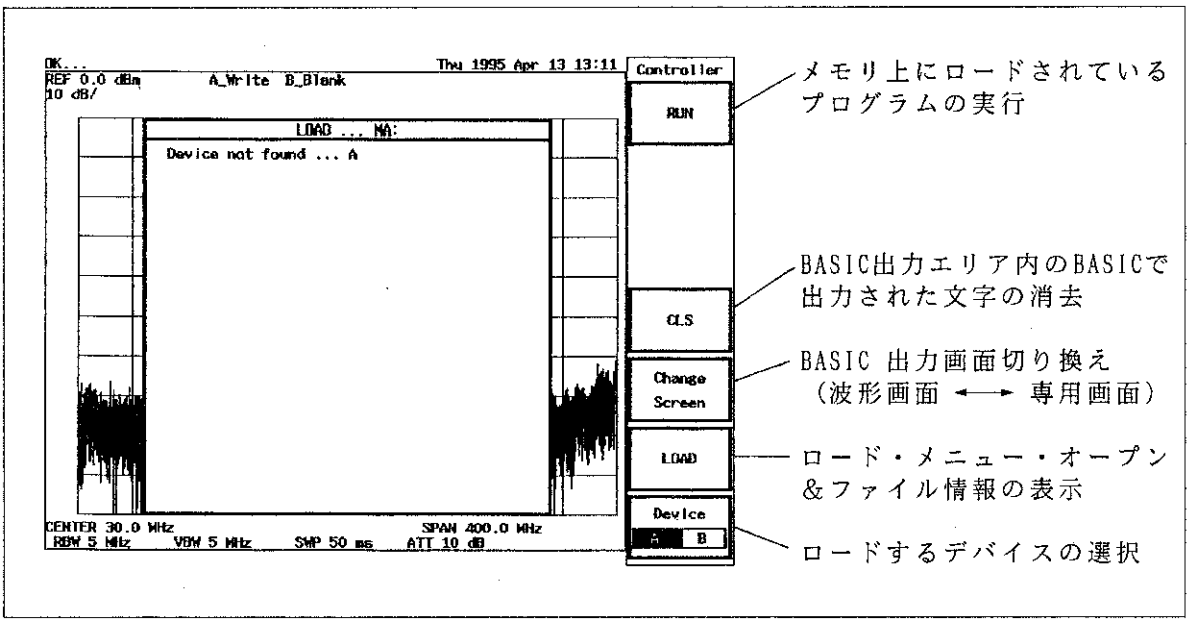


図 2 - 1　プログラムのロード／実行

2.2.2　データ入力キー

INPUT 文に対する入力は、テン・キー(0～9)、. キー、－キーを使用し、最後にENTER キーを押すと入力終了となります。

なお、ENTER キーを押す前ならば、－(BS)キーで入力データを 1文字ずつ消去することができます。

2.2.3　ファンクション・キー

ON KEY... 文に対するキー割り込みの発生は、ソフトキー(1～7)を使用します。

2.3　メモリ・カード

メモリ・カードには、設定条件、BASIC プログラムとBASIC プログラム内からのデータ・ファイルなどを記録/ 再生できます。
フォーマット形式は、MS-DOSに準拠しているため、MS-DOS対応のパーソナル・コンピュータにて、プログラム作成やデータ解析などができます。
本器では、以下のメモリ・カードが使用できます。

2.3.1　使用可能なメモリ・カード

● JEIDA Ver4.0以上に適合（68ピン 2ピース・コネクタ）
TYPEI

● メモリ・タイプは以下のものに限ります。
コモン・メモリ　：　SRAM
アトリビュート・メモリ　：　SRAM, EPROM, MASKROM, EEPROM, OTPROM, フラッシュ・メモリのいずれか

● フォーマット形式
MS-DOSフォーマット
各種メモリ・サイズに対応

注意

本器で使用可能なメモリ・カードは、㈳日本電子工業振興協会(JEIDA) のPCカード・ガイドライン Ver 4.0もしくは米国規格である PCMCIA Release 2.0 以上に適合するメモリ・カードに限られます。必ず、使用するメモリ・カードが上記の規格に適合していることを確認したうえで使用して下さい。
詳細については、「R3465 取扱説明書」または「R3272 取扱説明書」を参照して下さい。

2.3.2　メモリ・カード仕様

表 2 - 1　メモリ・カード仕様

仕様	メモリ・カード
コネクタ	68ピン 2ピース・コネクタ
インタフェース	JEIDA Ver4.0準拠
外形寸法	54（幅）×86（長さ）×3.3(厚み) mm
環境条件	結露しないこと 使用周囲環境 :　0 ～55℃ 保存温度範囲 :　-20 ～60℃ 相対湿度 :　95%以下
ライト・プロテクト	スイッチにて、ON/OFFを切り換えます。 ON側にすると書き込み不可となります。

2.3.3　メモリ・カードの取扱い上の注意

- コネクタ穴に埃などが入らないようにして下さい。
  接触不良やコネクタ破損の原因になります。

- コネクタに金属針のようなもので触れないようにして下さい。
  静電気破壊をおこすことがあります。

- 曲げたり、強い衝撃を加えないで下さい。

- 水に濡らさないで下さい。

2.3.4　メモリ・カードの挿抜方法

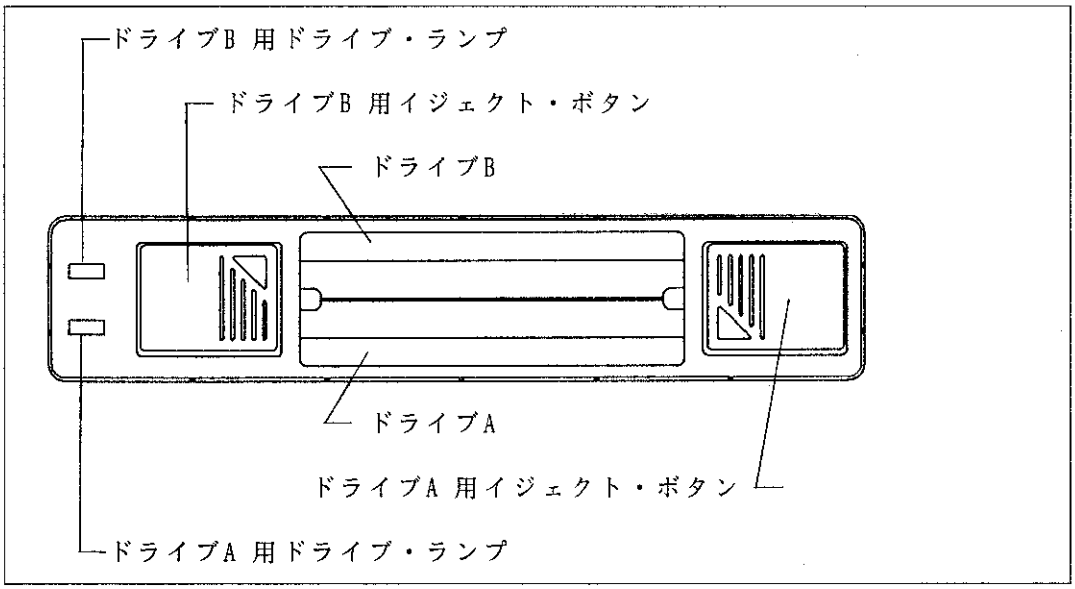


図 2 - 2　メモリ・カードのドライブ・スロット

メモリ・カードのドライブ・スロットは、正面パネル右上にあります。

① メモリ・カードの印刷のある面を上に向けて挿入して下さい。

② メモリ・カードを挿入すると、ドライブ・ランプが黄色に点灯します。

③ メモリ・カードを取り出す場合は、ドライブ・ランプが黄色に点灯していることを確認してから、イジェクト・ボタンを押して下さい。

注意

カード・アクセス中は、ドライブ・ランプが赤色に点灯します。ドライブ・ランプが赤色に点灯しているときに、イジェクト・ボタンを押して、メモリ・カードを抜かないで下さい。
ドライブ・ランプが赤色に点灯中にメモリ・カードを抜いた場合、カード内のデータは保証されません。

2.4　ファイルの管理

2.4.1　概要

(1)　ファイル

通常、ひとまとまりの情報を「ファイル」と呼びます。パーソナル・コンピュータ上で編集したBASIC プログラムや内蔵BASIC で作成したデータなどは、すべてファイルとして保存されます。

(2)　ドライブ

ファイルの保存場所は、メモリ・カードです。メモリ・カードを読み書きする装置を「ドライブ」と呼び、ドライブごとにメモリ・カードを管理することができます。本器の内蔵BASIC では、以下の2 つのドライブのアクセスが可能です。

MA：メモリ・カード・ドライブA
MB：メモリ・カード・ドライブB

（注）ドライブ名は、必ず大文字で指定して下さい。

(3)　ファイルの保存と呼び出し

BASIC でファイルを保存する場合や読み出す場合、任意のドライブとファイル名を指定します。
ドライブの指定は、ドライブ名の英文字にコロン(:) をつけます。ドライブ名は省略できますが、この場合、カレント・ドライブ（カレント・ディレクトリ）が自動的に選択されます。

【書式】　実行コマンド "〔ドライブ:〕＜ファイル名＞"

(4)　メモリ・カードの初期化

メモリ・カードの初期化用のコマンドは、本BASIC では用意していません。メモリ・カードを初期化する場合は、本器のSAVE/RECALL のソフト・メニュー構成内にある Format Card A またはFormat Card B 機能を使用して下さい。詳細は、各機種の取扱説明書を参照して下さい。

2.4.2 ファイルの管理

(1) CHDIR

CHDIR コマンドで現在のカレント・ドライブ（カレント・ディレトクリ）を切り換えます。
このコマンドは、プログラム実行中は使用できません。（ただし、プログラム行中に明記した場合は使用できます。）

```
CHDIR
```

現在のカレント・ドライブのホーム・ディレクトリにカレントを変更します。

```
CHDIR "ADVAN"
```

現在のカレント・ドライブのディレクトリADVAN にカレントを変更します。

```
CHDIR "MA:"
```

メモリ・カードA ドライブ内のカレント・ディレトクリにカレントを変更します。

```
CHDIR "MA:/"
```

メモリ・カードA ドライブ内のホーム・ディレトクリにカレントを変更します。

```
CHDIR "../"
```

現在のカレント・ディレトクリの親ディレクトリにカレントを変更します。

(2) CAT

CAT コマンドでドライブ中に挿入されているメモリ・カードのファイルおよびディレクトリ情報を表示します。

```
CAT
```

カレント・ドライブ（カレント・ディレクトリ）内を全表示します。

```
CAT "MA:"
```

メモリ・カードA ドライブ内を全表示します。

```
CAT "MB:*.BAS"
```

メモリ・カードB ドライブ内のファイル拡張子が'.BAS'のファイルを表示します。

```
CAT "../"
```

現在のカレント・ディレクトリから 1つ下がったディレクトリ内を表示します。

2.4.3 ファイルの保存

SAVEコマンドで、現在メモリ上に存在するプログラムに任意のファイル名を付けて指定ドライブのデバイスに保存します。
すでにデバイス内に存在するファイル名と同一のファイル名を指定すると、ファイルの内容は更新されるので、注意して下さい。

```
SAVE "AAA.BAS"
```

カレント・ドライブ（カレント・ディレクトリ）に保存します。

```
SAVE "MA:BBB.BAS"
```

メモリ・カードA ドライブに保存します。

2.4.4 ファイルの読み込み

LOADコマンドで、指定のドライブ（ディレクトリ）に保存済みのBASIC プログラム・ファイルをメモリ上に読み出します。

```
LOAD "AAA.BAS"
```

カレント・ドライブ（カレント・ディレクトリ）から読み出します。

```
LOAD "MA:BBB.BAS"
```

メモリ・カードA ドライブから読み出します。

注意

パーソナル・コンピュータで作成したプログラムをロードする際に、BASIC で扱える文字以外の文字がプログラム・ステートメント中（ダブル・コーテーションでくくられた場合を除く）に現れると"Not available char(yyy)" のメッセージを出力し、ロードを中断します。

2.4.5　ファイルの消去

PURGE コマンドで、不必要なファイルを消去します。

```
PURGE "AAA.BAS"
```

カレント・ドライブ（カレント・ディレクトリ）内のAAA.BAS ファイルを消去します。

```
PURGE "MA:BBB.BAS"
```

メモリ・カードA ドライブのBBB.BAS ファイルを消去します。

2.4.6　ファイル名の変更

RENAMEコマンドで、既存のファイル名を別のファイル名に変更します。変更されるのはファイル名のみで、ファイル内容に変更はありません。

```
RENAME "AAA.BAS", "BBB.BAS"
```

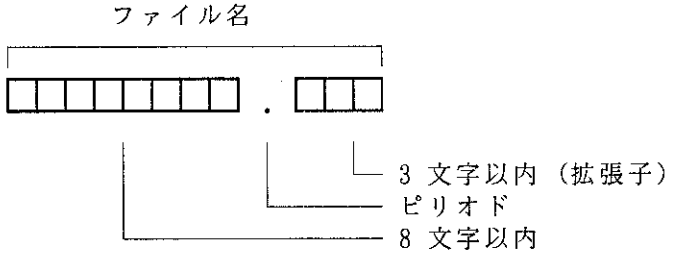
カレント・ドライブ（カレント・ディレクトリ）内のファイル名 AAA.BASファイルを BBB.BASファイルにファイル名を変更します。

```
RENAME "MA:AAA.BAS", "BBB.BAS"
```

メモリ・カードA ドライブのファイル名 AAA.BASファイルを BBB.BASファイルにファイル名を変更します。

注意

ファイル名は、数字、英字および記号（ダブル・クォーテーション(") は除く）を使い、以下のように指定します。



2.5　画面構成

内蔵BASIC を使用したときの本器の画面構成は、以下のようになります。
本器の内蔵BASIC で使用する画面には、波形画面と合成して使用する「BASIC 波形画面」（図2-3を参照）と、波形画面を隠してBASIC のみの表示を行う「BASIC 専用画面」（図2-4を参照）の 2種類をサポートしています。
2 種類の画面の切り換えは、内蔵BASIC 用ソフトキーのChange Screen キーまたはBASIC コマンドの「CONTROL 8;0 」（BASIC波形画面）、「CONTROL 8;1 」（BASIC専用画面）によって行います。

- BASIC メッセージ表示用エリア：
  内蔵BASIC が出力するエラー・メッセージ等の専用エリアです。

- BASIC 出力エリア：　内蔵BASIC で実行されたコマンドによって出力される文字列が表示されるエリアです。
  LISTコマンド、CAT コマンド等により出力される内容がこのエリアに表示されます。

- コマンド入力エリア：本器に外部キーボードを接続したときのキーボードからのキー入力や、本器のパネル上のテン・キーからの入力に対応した文字が表示されるエリアです。
  外部キーボードを使用したときの内蔵BASIC に対するコマンド入力時などに使用します。

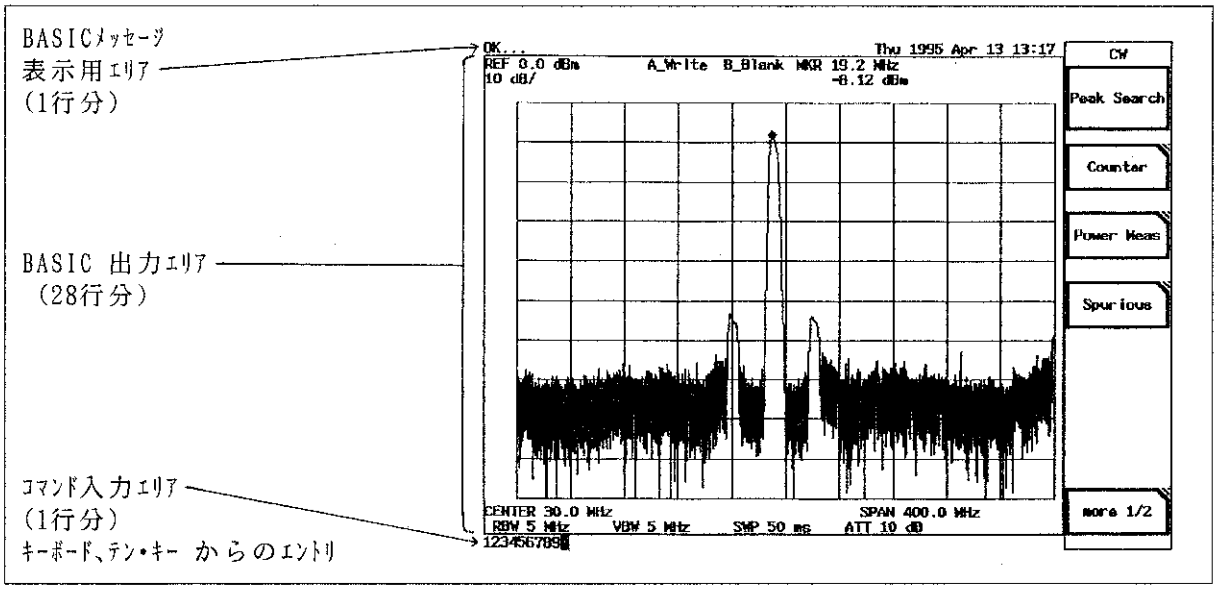


図 2 - 3　BASIC 波形画面（波形画面＋BASIC 画面）

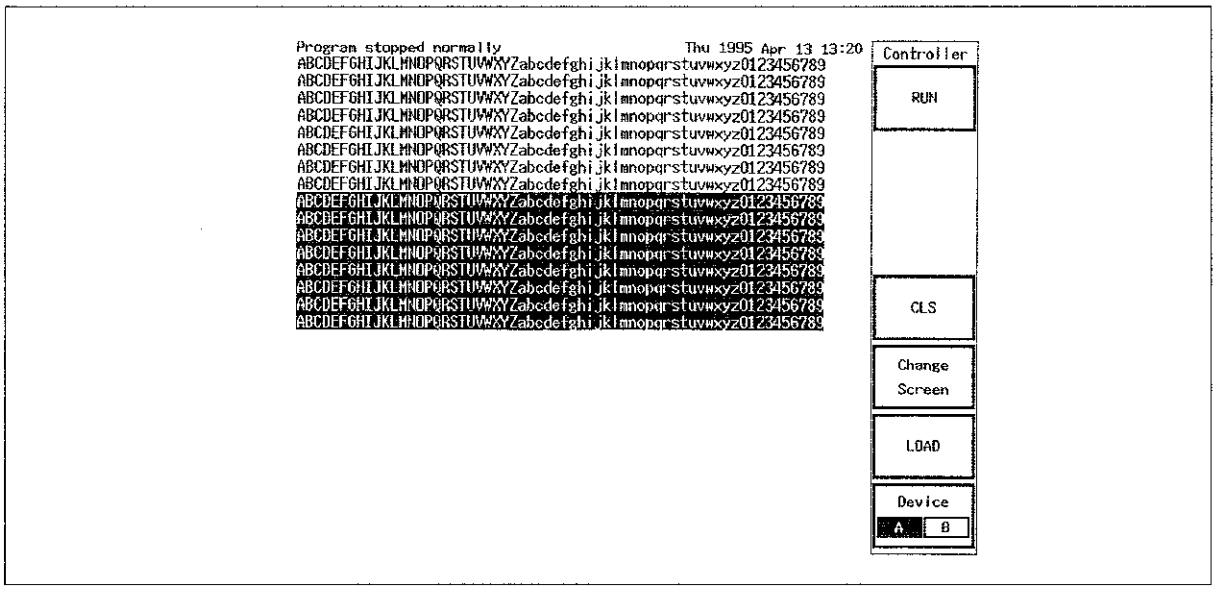


図 2 - 4 BASIC 専用画面(BASIC画面)

2.6　外部キーボード

本器で簡単なプログラムの編集を行う場合には、外部キーボードが必要です。外部キーボードを接続すると、内部にロードしたプログラムを 1行単位で修正／追加などの編集作業が行えます。

本器で使用可能なキーボードは、 101型キーボードまたは 106型キーボード（コネクタ形状ミニDIN6ピン）相当品です。

101 型/106型の切り換えは、本器のLCL キーを押した後、GPIB & Others ソフトキーを押して現れるダイアログ・ボックス内のExt.Keyboardの項で選択できます。ロータリ・ノブで101key/106key のいずれかを選択し、ロータリ・ノブまたはENTER キーで確定します。

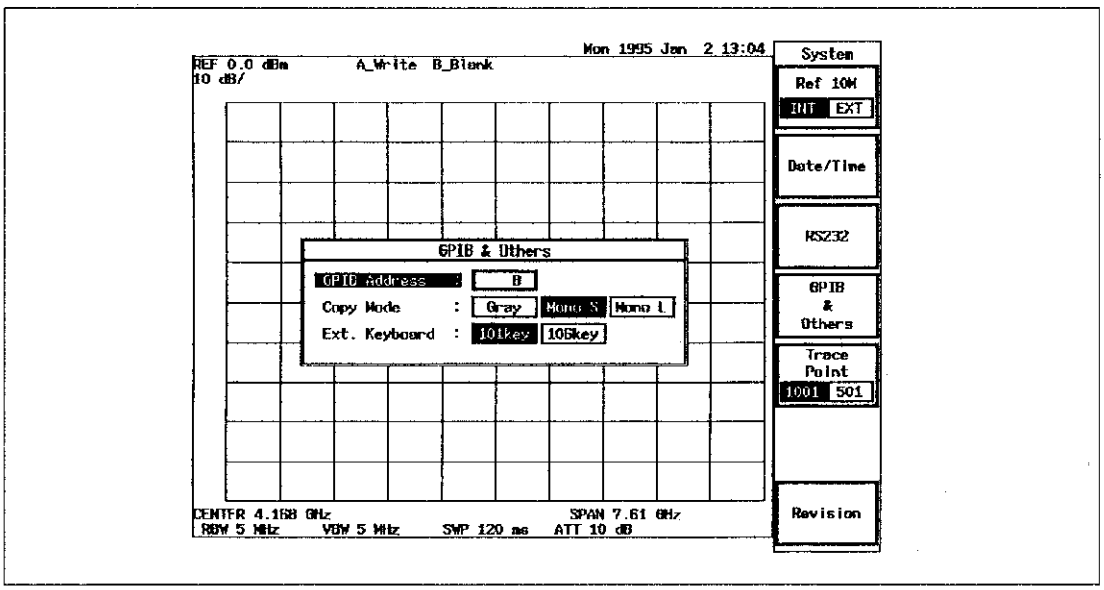
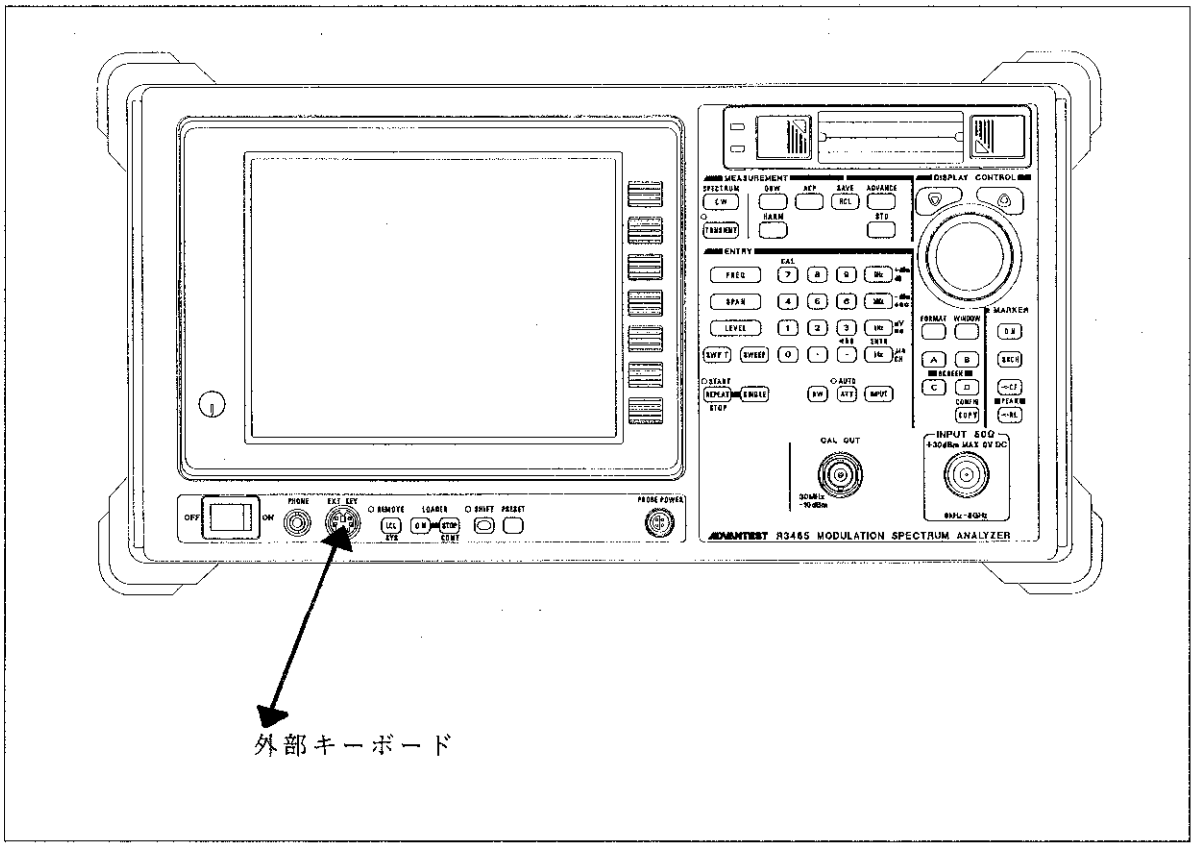


図 2 - 5　101 型/106型の切り換え(GPIB & Others)

2.6.1　外部キーボードの接続

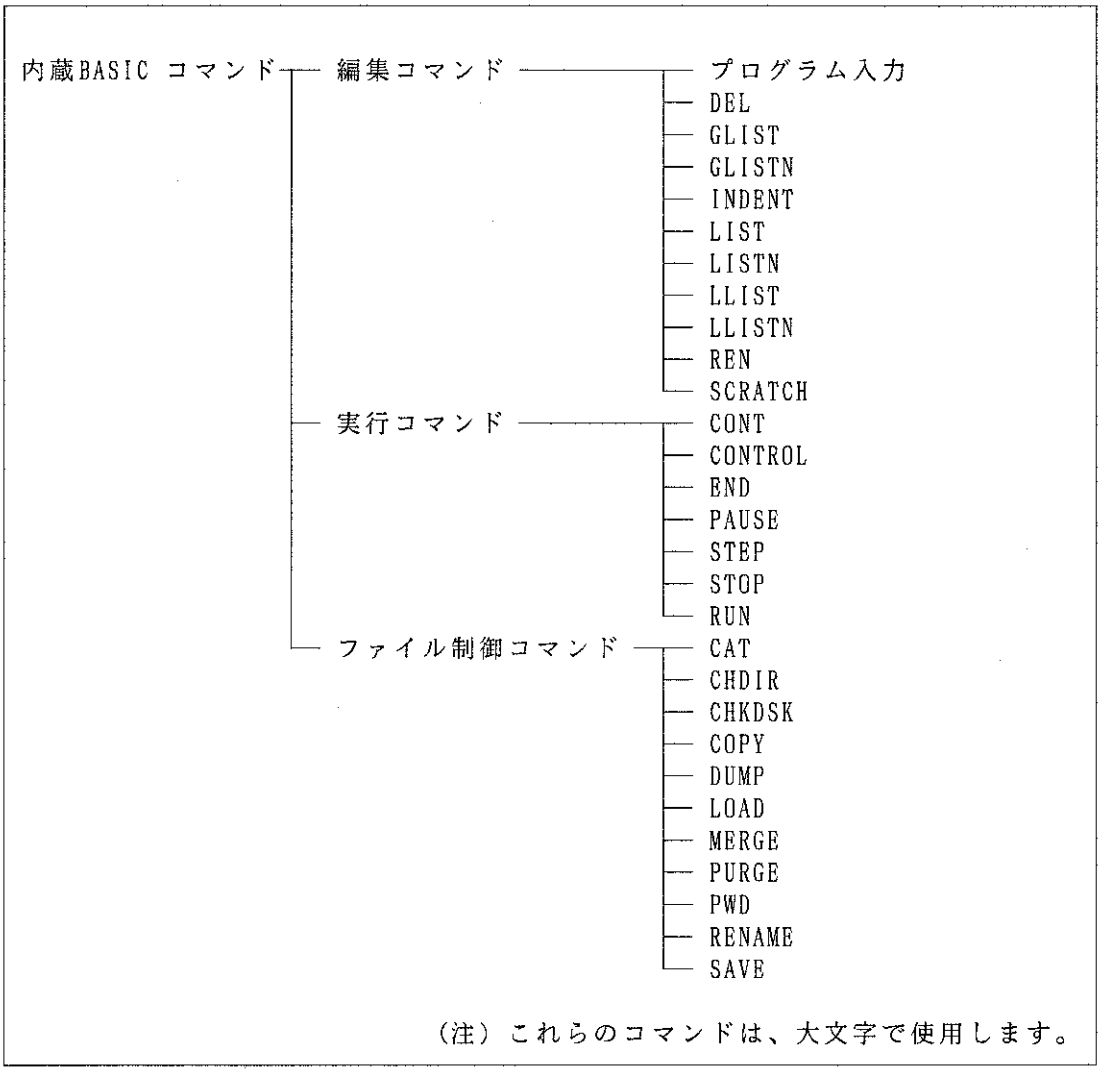
外部キーボードは、本器のEXT KEY コネクタに接続します。



3.　BASIC コマンド

3.1　各種コマンド

内蔵BASIC には、プログラムの編集、実行およびファイル操作を行うためのコマンドがあります。内蔵BASIC コマンドの構成を以下に示します。



（注）これらのコマンドは、大文字で使用します。

3.1.1 コマンド機能一覧

	コマンド	機能
編集コマンド	プログラム入力	ステートメントをプログラムとして格納
	DEL	指定行番号を削除
	GLIST	プログラム・リストをGPIBに出力
	GLISTN	プログラム・リストをGPIBに出力
	INDENT	プログラム・リストの字下げ
	LIST	プログラム・リストをディスプレイ上に表示
	LISTN	プログラム・リストをディスプレイ上に表示
	LLIST	プログラム・リストをパラレル・ポートに出力
	LLISTN	プログラム・リストをパラレル・ポートに出力
	REN	行番号の変更
	SCRATCH	すでに入力されているプログラムを消去
実行コマンド	CONT	プログラム実行を再開
	CONTROL	BASIC のコントロール変数を設定（環境設定）
	END	プログラム実行を終了　　(CONT、STEP不可)
	PAUSE	プログラム実行を一時停止（CONT、STEP可）
	STEP	プログラムを一行実行
	STOP	プログラム実行を停止　　(CONT、STEP可)
	RUN	プログラムを実行
ファイル制御コマンド	CAT	ﾄﾞﾗｲﾌﾞ内のﾒﾃﾞｨｱに記録されている ﾌｧｲﾙ情報をﾃﾞｨｽﾌﾟﾚｲ上に表示
	CHDIR	指定されたﾄﾞﾗｲﾌﾞ、およびﾃﾞｨﾚｸﾄﾘにｶﾚﾝﾄを変更
	CHKDSK	ドライブ内のメディア情報をディスプレイ上に表示
	COPY	指定されたﾒﾃﾞｨｱに記録されているファイルを複写
	DUMP	ファイル内容のASCII ダンプ出力
	LOAD	BASIC ﾌﾟﾛｸﾞﾗﾑ･ﾌｧｲﾙをメモリ上に読み出す
	MERGE	BASIC プログラムの合成／消去
	PURGE	ﾄﾞﾗｲﾌﾞ内のﾒﾃﾞｨｱに記録されているファイルを削除
	PWD	ｶﾚﾝﾄ･ﾄﾞﾗｲﾌﾞ のパス名をディスプレイ最上行に表示
	RENAME	指定されたﾒﾃﾞｨｱに記録されているファイル名を変更
	SAVE	プログラムをメモリ・カードへセーブ（保存）

3.1.2 コマンド文法一覧

	コマンド	文法
編集コマンド	プログラム入力	行番号 ｽﾃｰﾄﾒﾝﾄ
	DEL	DEL ＜開始行＞〔, 終了行〕
	GLIST	GLIST〔開始行〕〔, 〔終了行〕〕
	GLISTN	GLISTN〔開始行〕〔, 〔行数〕〕
	INDENT	INDENT ＜開始位置, 増分＞
	LIST	LIST 〔開始行〕〔, 〔終了行〕〕
	LISTN	LISTN 〔開始行〕〔, 〔行数〕〕
	LLIST	LLIST〔開始行〕〔, 〔終了行〕〕
	LLISTN	LLISTN〔開始行〕〔, 〔行数〕〕
	REN	REN 〔〔旧行番号〕〔, ＜新行番号＞〔, ＜増分値＞〕〕〕
	SCRATCH	SCRATCH 〔1｜2〕
実行コマンド	CONT	CONT〔行番号｜ラベル式〕
	CONTROL	CONTROL ＜レジスタ番号＞；＜値＞
	END	END
	PAUSE	PAUSE
	STEP	STEP〔行番号｜ラベル式〕
	STOP	STOP
	RUN	RUN 〔行番号｜ラベル式〕
ファイル制御コマンド	CAT	CAT 〔"〔ﾄﾞﾗｲﾌﾞ名:〕〔ﾃﾞｨﾚｸﾄﾘ/〕〔ﾜｲﾙﾄﾞ･ｶｰﾄﾞ〕"〕
	CHDIR	CHDIR"〔ドライブ名:/〕ディレクトリ名"
	CHKDSK	CHKDSK〔"ドライブ名:"〕
	COPY	COPY "旧ファイル名", "新ファイル名"
	DUMP	DUMP "〔ドライブ名:〕〔ディレクトリ/〕ファイル名"
	LOAD	LOAD "〔ドライブ名:〕〔ディレクトリ/〕ファイル名"
	MERGE	MERGE "〔ドライブ名:〕〔ディレクトリ/〕ファイル名" ,〔合成開始行番号〕,〔開始行｜ラベル式〕 MERGE DEL
	PURGE	PURGE "〔ドライブ名:〕〔ディレクトリ/〕ファイル名"
	PWD	PWD
	RENAME	RENAME "旧ファイル名", "新ファイル名"
	SAVE	SAVE "〔ドライブ名:〕〔ディレクトリ/〕ファイル名"

3.1.3 コマンドの共通注意事項

内部BASIC コマンドには、以下に示す共通の注意事項があります。

(1) パラメータ

コマンドのパラメータには、文字列表現式または数値表現式が指定できます。つまり、BASIC で使用する変数が使えます。ここで、実数の場合は小数点以下が切り捨てられます。
しかし各コマンドの解説では、見やすさのため整数、文字列などの表現を使っています。

(2) 式の境界

原則としてBASIC コマンドは、式を複数個続けて指定する場合、その式の境界が構文上解釈できれば、カンマ(,) はスペースにしても構いません。

(3) 行番号

行番号の設定範囲は 1～65535 です。
0 またはプログラムの先頭行よりも小さい値を指定した場合、プログラムの先頭行を指定したと解釈されます。
65535 またはプログラムの最終行よりも大きい値を指定した場合、プログラムの最終行を指定したと解釈されます。

3.2　コマンド文法と活用

1.　プログラム入力

3 章と4 章に記載するコマンドとステートメントは、行番号を付けるとプログラムとして入力できます。

入力済のプログラムに同一の行番号が存在する場合は、置き換えとなります。また、存在しない場合は、追加または挿入となります。

2.　CAT

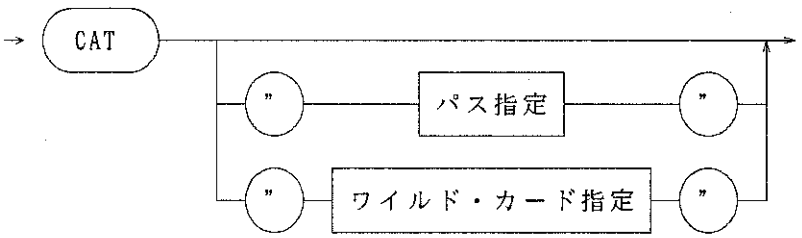
概　要

ドライブ名（ディレクトリ）が指定された場合はそのドライブ（ディレクトリ）内の、指定されなかった場合は現在のカレント・ドライブ（カレント・ディレクトリ）内のメディアに記録されているファイル情報をディスプレイ上に表示します。
出力されるファイルの情報は、左側から登録番号(変数)、ファイル名（ディレクトリ名）、使用バイト数、ファイル属性、作成日時の順となっています。メディアの抜き差しが検知された場合には、ルート・ディレクトリにカレント・ディレクトリが移動します。

ファイル属性	1 : READ ONLY 16 : DIRECTORY 32 : ARCHIVE FILE

構　文

(1)-1



(1)-2
CAT 〔"〔ドライブ名:〕〔ディレクトリ/〕〔ワイルド・カード〕"〕

解　説

CAT　：カレント・ドライブ（カレント・ディレクトリ）内を全表示する

CAT "MA:"　：メモリ・カードA ドライブ内を全表示する

CAT "MB:*.BAS"　：メモリ・カードB ドライブ内の拡張子が'.BAS'のファイルを表示する

CAT "../"　：現在のカレント・ディレクトリの親ディレクトリ内を表示する

3. CHDIR

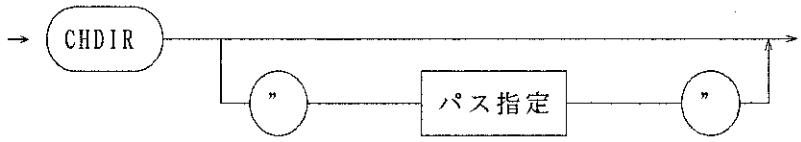
概 要

指定されたドライブおよびディレクトリにカレントを変更します。
この際、変更後のカレント・ドライブ（カレント・ディレクトリ）のパス名がディスプレイの 1行目に表示されます。CHDIR で変更されたカレント・パス名は、PWD ステートメントで確認できます。(本節 21. PWDを参照)

このコマンドは、プログラム実行中は使用できません（ただし、プログラム中での使用は可能です）。

構 文

(1)-1



(1)-2
CHDIR "〔ドライブ名:/〕ディレクトリ|../〔../../..〕 "

解 説

CHDIR : 現在のカレント・ドライブのホーム・ディレクトリにカレントを変更する

CHDIR "ADVAN" : 現在のカレント・ドライブのディレクトリADVAN にカレントを変更する

CHDIR "MA:" : メモリ・カードA ドライブ内のカレント・ディレクトリにカレントを変更する

CHDIR "MA:/" : メモリ・カードA ドライブ内のホーム・ディレクトリにカレントを変更する

CHDIR "../" : 現在のカレント・ディレクトリの親ディレクトリにカレントを変更する

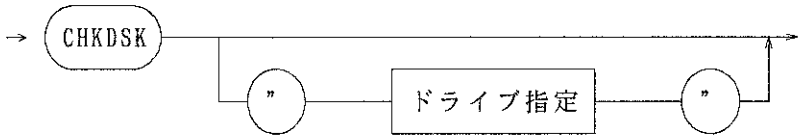
4.　CHKDSK

概　要

ドライブ名が指定された場合はそのドライブ内の、指定されなかった場合は現在のカレント・ドライブ内のメディア情報をディスプレイ上に表示します。

構　文

(1)-1



(1)-2

CHKDSK〔" ドライブ名:"〕

解　説

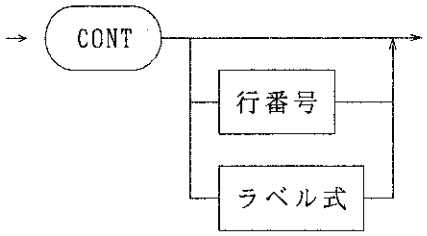
CHKDSK　　：現在のカレント・ドライブに挿入されているメディア情報を表示する

CHKDSK "MA:"　　：メモリ・カードA ドライブに挿入されているメディア情報を表示する

5. CONT

概 要 PAUSE(本節 18. PAUSEを参照) で一時停止したBASIC プログラムの実行を再開させます。

構 文 (1)-1



(1)-2
CONT〔行番号 | ラベル式〕

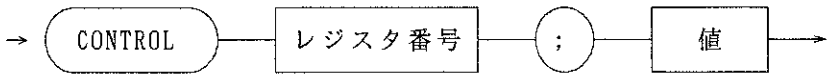
解 説

CONT	:	PAUSE で停止された次の行から実行を再開する
CONT 100	:	プログラムの 100行目からプログラムを再開する
CONT *Lb1	:	プログラムの*Lb1行目からプログラムを再開する

6. CONTROL

概 要 内蔵BASIC のコントロールに関する細かな部分の値（環境変数）を設定します。

構 文 (1)-1



(1)-2
CONTROL　レジスタ番号　;　値

解 説 ＜レジスタ１＞　初期値79（現在未使用なレジスタ）
シリアルI/O ポートの設定を行います。以下に示す値の和で指定します。
下線のある値は、電源投入時に設定されている値を示しています。

●ボーレート
0 ; 1200ボー、1 ; 2400ボー、2 ; 4800ボー、　3 ; 9600ボー

●キャラクタ長
0 ; 5 ビット、4 ; 6 ビット、8 ; 7 ビット、　12; 8 ビット

●パリティ
0 ; なし、　16; 奇数、　48;偶数

●ストップ・ビット数
0 ; なし、　64; 1 ビット、128; 1 1/2ビット、192; 2ビット

例）ボーレート9600ボー、キャラクタ長8 ビット、パリティ偶数、ストップ・ビット数 2ビットの場合

```
CONTROL 1;3+12+48+192
または
CONTROL 1;255
```

＜レジスタ２＞　初期値0
LLIST, LLISTN, GLIST, GLISTNコマンドで印字する際の左端からの印字スペース数を指定します。

例）リスト出力を右に 5文字分寄せたい場合

```
CONTROL 2;5
```

＜レジスタ３＞　初期値0
BASIC コマンドをリスト表示する際、通常表示にするか、短縮表示にするかを指定します。

0 : 通常表示（フルネーム）
1 : 短縮表示（ショートネーム）

＜レジスタ４＞　初期値1
プログラム・リストする際、変数、ラベル等を大文字表現にするか、頭1文字大文字で以降を小文字表現にするかを指定します。

0 : 大文字表現
1 : 頭1 文字大文字で以降を小文字表現
（コマンド、予約後は大文字表現）

＜レジスタ５＞　初期値0
ディスプレイ上にカーソルを表示するかどうかを設定します。

0 : カーソルを表示しない
1 : カーソルを表示する

＜レジスタ７＞　初期値0
GPIB関係の設定です。以下の値は、それぞれに設定可能です。

0 : GPIBモードをADDRESSABLE モードに設定します。
1 : GPIBモードをSYSTEM CONTROLLLERモードに設定します。
2 : REQUEST CONTROL（コントローラ権の要求）を送信します。
4 : BASIC 実行中に、外部コントローラからのGPIBコマンド設定を許可します。

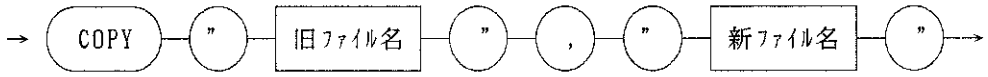
＜レジスタ８＞　初期値0
ディスプレイの表示画面を切り換えます。

0 : 波形画面（通常の波形表示上に出力文字が合成されます。）
1 : BASIC 画面（波形は表示されません。）

7. COPY

概　要　指定されたメディアに記録されているファイルを複写します。

構　文　(1)-1



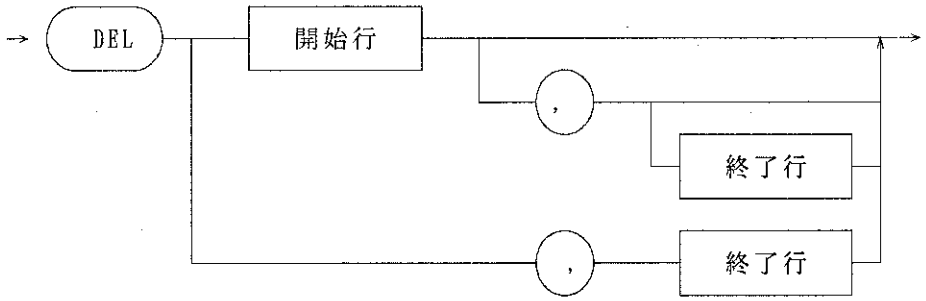
(1)-2
COPY "旧ファイル名", "新ファイル名"

解　説
- 旧ファイル名の内容を新ファイル名に複写します。
- 新ファイル名が既に存在する場合、旧ファイル名の内容が上書きされます。
- 2 つのファイル名とも文字列表現式を使って指定できます。

8. DEL

概 要 プログラム中の行を削除します。

構 文 (1)-1



(1)-2
DEL <開始行〔, 〔終了行〕>|<,終了行>

注) カンマ(,) をスペースにしても構いません。
行番号の設定範囲は 1～65535 です。

解 説
・開始行から終了行までプログラムを削除します。
・行番号の指定がない場合は実行しません。

例

DEL 10	行番号10のみ削除
DEL 10 ,	行番号10～最終行まで削除
DEL 10, 100	行番号10～100 まで削除
DEL , 100	先頭行～行番号100 まで削除

9.　END

概　要　BASIC プログラムの実行を終了します。

構　文　(1)-1

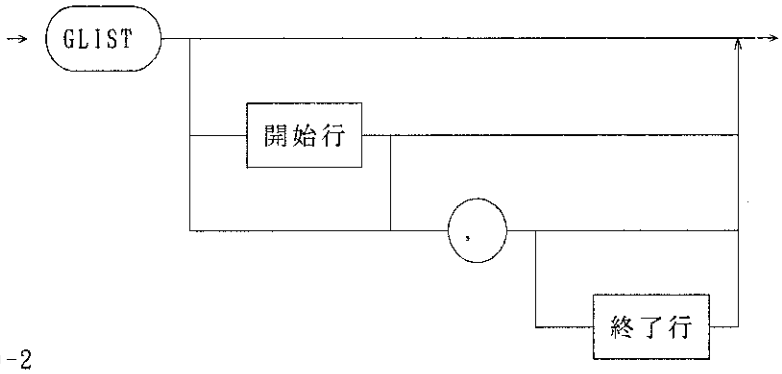
(1)-2
END

解　説
・BASIC プログラムの実行を終了、またはBASIC プログラム自身で終了させます。

10. GLIST

概 要 GPIB経由でプリンタ等にプログラム・リストを出力します。

構 文 (1)-1



(1)-2

GLIST 〔開始行〔, 〔終了行〕〕 | 〔, 〔終了行〕〕

注) カンマ(,) をスペースにしても構いません。
行番号の設定範囲は 1～65535 です。

解 説

・GPIBに接続されたプリンタ等に、BASICプログラム・リストを出力します。

・プリンタのGPIBアドレスは、PRINTER 文で、または本器のパネル操作で設定します。

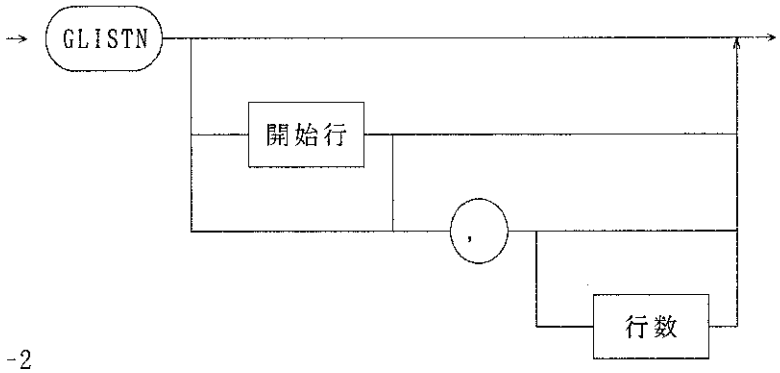
例

GLIST	全行を出力
GLIST 100	100 行目のみ出力
GLIST 100,	100 行目～最終行まで出力
GLIST 100, 200	100 行目～200 行目まで出力
GLIST ,	全行を出力 (GLISTと同じ)
GLIST ,200	先頭行～200 行目まで出力

11. GLISTN

概 要 GPIB経由でプリンタ等にプログラム・リストを出力します。

構 文 (1)-1



(1)-2
GLISTN〔開始行〔,〔行数〕〕|〔,〔行数〕〕

注) カンマ(,) はスペースにしても構いません。
行番号の設定範囲は 1～65535 です。

解 説

・GPIBに接続されたプリンタ等に、BASIC プログラム・リストを出力します。

・プリンタのGPIBアドレスは、PRINTER 文で、または本器のパネル操作で設定します。

・開始行で指定した行番号から行数で指定した行数分のプログラム・リストを出力します。

・行数が負の数の場合、行番号の小さい方に向かって指定行数がとられます。

例

GLISTN	全行を出力
GLISTN 100	100 行目のみ出力
GLISTN 100,	100 行目～最終行まで出力
GLISTN 100, 20	100 行目～20行出力
GLISTN ,	全行を出力 (GLISTN と同じ)
GLISTN ,20	先頭行～20行出力

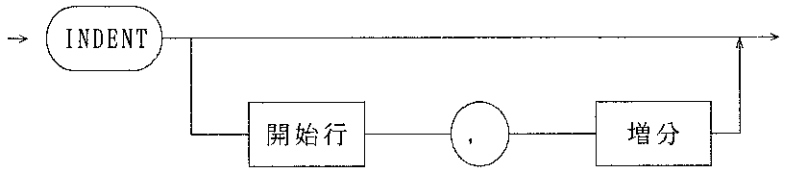
12. INDENT

概 要

メモリ上のプログラム・リストを、構文構造を反映するように整形しなおします。開始位置は、各行番号以降の最初のステートメントの、最初の文字の位置です。増分桁数は、プログラムのネフトの段階が変わったときに最初の文字が右や左に動く文字数です。

構 文

(1)-1



(1)-2

INDENT〔開始位置, 増分〕

指定範囲

開始位置 : 1～32

増分 : 0～10

解 説

・INDENT → 開始位置 6文字目で増分桁数 2文字分の字下げをします。

・INDENT 8, 3 → 開始位置 8文字目で増分桁数 3文字分の字下げをします。

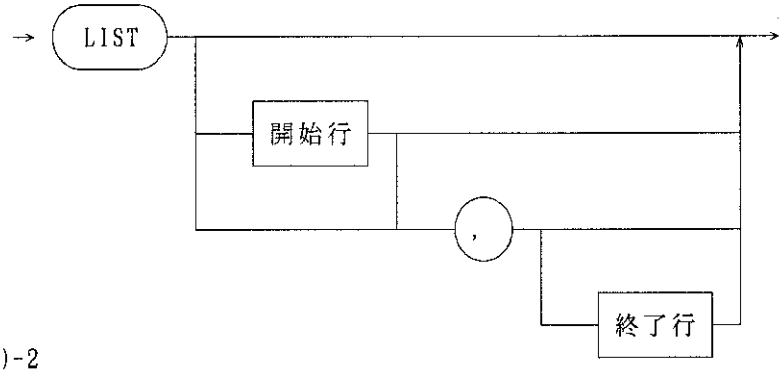
注 意

INDENTコマンドにより字下げが行われるステートメントは、IF(ELSE, ELSE IF)、FOR およびSELECT(CASE, CASE ELSE) のいずれかですが、IFの単文に限っては、そのときの構文により字下げされません。
また、END IF、NEXTおよびEND SELECTステートメントは、字下げを反転します。

13. LIST

概 要 ディスプレイ上にプログラム・リストを表示します。

構 文 (1)-1



(1)-2

LIST〔開始行〔,〔終了行〕〕|〔,〔終了行〕〕

注）カンマ(,)をスペースにしても構いません。
行番号の設定範囲は 1～65535 です。

解 説

・ディスプレイ上にパラメータで指定した範囲の BASICプログラム・リストを表示します。

・STOPキーで表示を中止できます。しかし、プログラムの実行と異なるため、中断した行位置から再開できません。

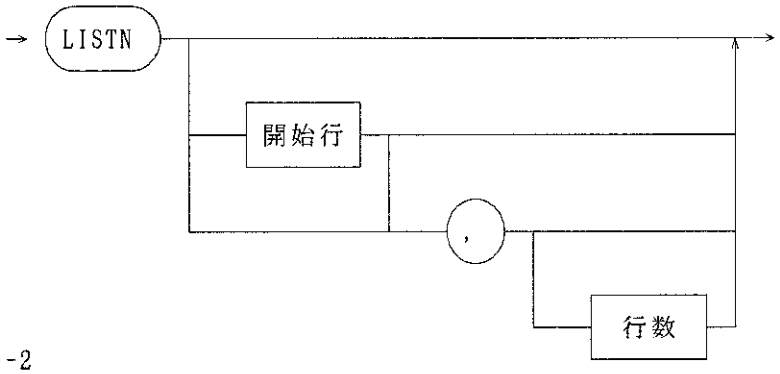
例

LIST	全行を出力
LIST 100	100 行目のみ出力
LIST 100,	100 行目～最終行まで出力
LIST 100, 200	100 行目～200 行目まで出力
LIST ,	全行を出力 (LIST と同じ)
LIST ,200	先頭行～200 行目まで出力

14.　LISTN

概　要　ディスプレイ上にプログラム・リストを表示します。

構　文　(1)-1



(1)-2
LISTN 〔開始行〔, 〔行数〕〕〕｜〔, 〔行数〕〕

注）カンマ(,) はスペースにしても構いません。
　　行番号の設定範囲は 1～65535 です。

解　説　・ディスプレイにパラメータで指定した範囲のBASIC プログラム・リストを表示します。

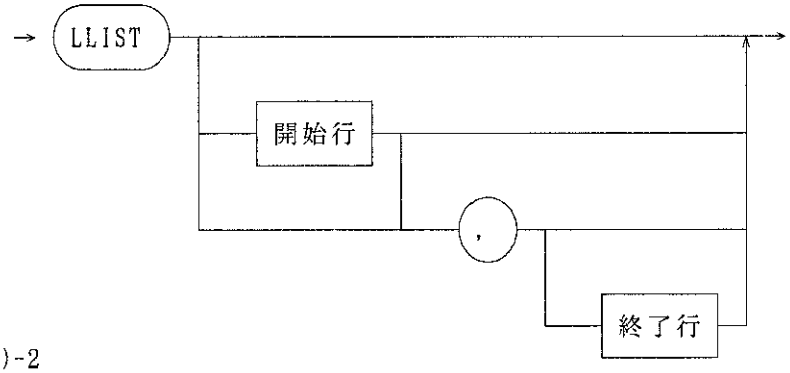
例

LISTN	全行を出力
LISTN 100	100 行目のみ出力
LISTN 100,	100 行目～最終行まで出力
LISTN 100, 20	100 行目～20行出力
LISTN ,	全行を出力　(LISTNと同じ)
LISTN ,20	先頭行～20行出力

15.　LLIST

概　要　パラレル・ポート経由でプリンタ等にプログラム・リストを出力します。

構　文　(1)-1



(1)-2

LLIST 〔開始行〔, 〔終了行〕〕〕｜〔, 〔終了行〕〕

注）カンマ(,) をスペースにしても構いません。
　　行番号の設定範囲は 1～65535 です。

解　説　・パラレル・ポートに接続されたプリンタ等に、BASIC プログラム・リストを出力します。

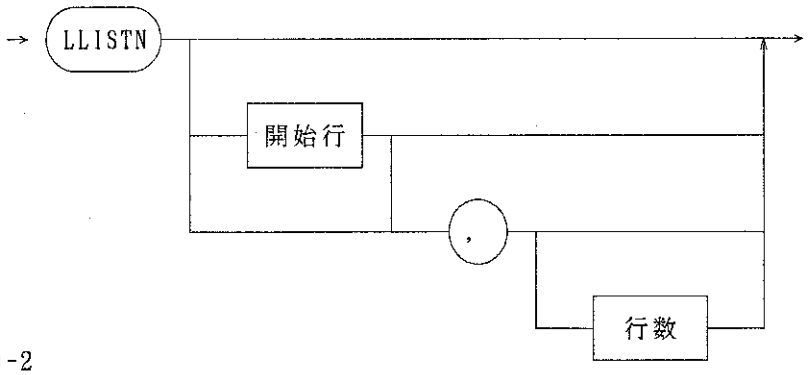
例

LLIST	全行を出力
LLIST 100	100 行目のみ出力
LLIST 100,	100 行目～最終行まで出力
LLIST 100, 200	100 行目～200 行目まで出力
LLIST ,	全行を出力　(LLISTと同じ)
LLIST ,200	先頭行～200 行目まで出力

16.　LLISTN

概　要　パラレル・ポート経由でプリンタ等にプログラム・リストを出力します。

構　文　(1)-1



(1)-2

LLISTN〔開始行〔,〔行数〕〕｜〔,〔行数〕〕

注）行番号の設定範囲は 1～65535 です。

解　説
- パラレル・ポートに接続されたプリンタ等に、BASIC プログラム・リストを出力します。
- 開始行で指定した行番号から行数で指定した行数分のプログラム・リストを出力します。
- 行数が負の数の場合、行番号の小さい方に向かって指定行数がとられます。

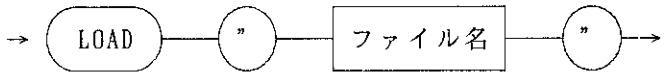
例

LLISTN	: 全行を出力
LLISTN 100	: 100 行目のみ出力
LLISTN 100,	: 100 行目～最終行まで出力
LLISTN 100, 20	: 100 行目～20行出力
LLISTN ,	: 全行を出力 （LLISTN と同じ）
LLISTN ,20	: 先頭行～20行出力

17. LOAD

概 要 ドライブに登録されたファイルを呼び出します。

構 文 (1)-1



(1)-2
LOAD "〔ドライブ名:〕〔ディレクトリ/〕ファイル名"

解 説
・ファイル名に指定されたファイルを呼び出します。BASIC 以外のファイル（システム・ファイル等）は呼び出せません。

例

LOAD "AAA.BAS"	:	現在のカレント・ドライブ（カレント・ディレクトリ）のメディアに記録されている AAA.BAS ファイルの内容をロードする
LOAD "MA:BBB.BAS"	:	メモリ・カードA ドライブ内のメモリ・カードに記録されているBBB.BAS ファイルの内容をロードする

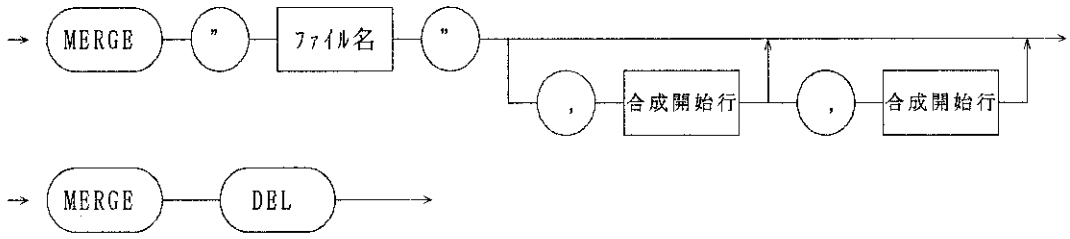
※ ファイルの取り扱いについては、「2.4 ファイルの管理」を参照して下さい。

18. MERGE

概 要

ドライブに登録されたファイルを呼び出し、合成します。
また、合成したプログラムをメモリから消去します。

構 文 (1)-1



(1)-2
MERGE "〔ドライブ名:〕〔ディレクトリ/〕ファイル名",
〔合成開始行番号〕〔, 行番号｜ラベル式〕

解 説

メモリ上のプログラムにドライブ名が指定された場合はそのドライブ内の、指定されなかった場合は現在のカレント・ドライブ内のメディアに記録されている指定ファイルを合成します。

メモリ上のプログラムとメディアから読み出す（マージ）プログラムに同一の行番号があった場合は、メディアから読み出される行が採用されます。

行番号が一致しない行は、その行をメモリ上のプログラムに追加または挿入という形になります。

なお、行番号なしのプログラム・ファイルをMARGE する場合のみ、第2 引数の合成開始行番号が有効となります。合成開始行を指定すると、合成開始行から、また指定がない場合は、現在メモリ上に存在するプログラムの最終行+1の行から 1刻みで自動的に行番号が割り当てられます。

第3 引数の行番号－ラベル式は、プログラムを合成後の実行開始行番号、またはラベル式を指定します。この引数の省略時、または合成されたプログラム間に存在しない行が指定された場合、実行開始位置は自動的に合成されたプログラムの先頭行となります。

また、合成済みのプログラム行を消去する場合は、MARGE コマンドに続きDEL と明記します。DEL コマンドによる合成プログラムの削除は行えません。このコマンドの実行により、合成されたプログラム部はメモリ上のプログラムから消去され、プログラムに実行位置（行）は、MARGE コマンドが実行された次の行へと移ります。

MERGE コマンドは、LOADコマンドと異なり、ロードする前にBASIC バッファの初期化は行われません。
SCRATCH と MERGEの組み合わせがLOADの機能と同様になります。

例

MERGE "AAA.BAS"	:	現在のカレント・ドライブ（カレント・ディレクトリ）のメディアに記録されているAAA.BAS ファイルの内容を読み出す
MERGE "MA:BBB.BAS"	:	メモリ・カードA ドライブ内のメモリ・カードに記録されているBBB.BAS ファイルの内容を読み出す
MERGE "BBB",1000	:	メモリ上のプログラムに行番号1000からBBBファイルのプログラムを合成後、合成行の先頭から実行を継続する
MERGE "BBB",100,200	:	メモリ上のプログラムに行番号100 からBBBファイルのプログラムを合成後、 200行目から実行を継続する
MERGE "BBB",,*Lb1	:	メモリ上のプログラムにBBB ファイルのプログラムを合成後、*Lb1行目から実行を継続する
MERGE DEL	:	合成済みのプログラム行を消去する

注 意

・メモリ上のプログラムがプロテクト・ファイルの場合、このファイルの最終行番号以内にプログラム行の合成を行うことはできません。
このとき、以下のメッセージが表示されます。

(Cannot execute protected file)

・合成開始行にラベルを指定することはできません。

・合成開始行指定は、合成プログラムに行番号がない場合のみ有効となります。合成プログラムに行番号がある場合は、そのまま合成されます。

・実行開始行が省略、または指定された行番号が合成されたプログラム行以外だった場合は、合成されたプログラムの先頭行から実行を継続します。

・MERGE により合成されるプログラムが、メモリ上のMERGE コマンドが実行された次の行を更新した場合、MERGE DEL 実行時にそのプログラム行は消去されます。したがってプログラムの継続実行は行われません。

19.　PAUSE

概　要　プログラムの実行を一時停止させます。

構　文　(1)-1

(1)-2
PAUSE

解　説
・BASIC プログラムの実行を一時的に停止、またはBASIC プログラム自身で一時的に停止させます。

・停止した次の行から、CONTによりプログラムを継続できます。

例

```
10  FOR I=1 TO 9
20    GOTO 60
30    GOTO *PRT
40  NEXT I
50  PAUSE
60  !
70  X = I * I
80  GOTO 30
90  *PRT
100 PRINT I;"*";I;"=";X
110 GOTO 40
```

20. PRINTER

4.3節の[55. PRINTER] を参照して下さい。

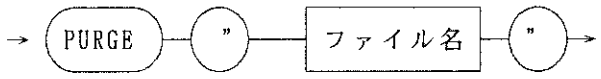
21. PURGE

概 要

ドライブ名が指定された場合はそのドライブ内の、指定去れなかった場合は現在のカレント・ドライブ内のメディアに記録されている指定ファイルを消去します。

構 文

(1)-1



(1)-2
PURGE "〔ドライブ名:〕〔ディレクトリ名/〕ファイル名"

解 説

・既存のファイルで、不要なものを消去します。

例

PURGE "AAA.BAS" : 現在のカレント・ドライブ（カレント・ディレクトリ）のメディアに記録されているAAA.BAS ファイルの内容を消去する

PURGE "MA:BBB.BAS" : メモリ・カードA ドライブ内のメモリ・カードに記録されているBBB.BAS ファイルを消去する

22. PWD

概 要 現在のカレント・ドライブ（カレント・ディレクトリ）のパス名をディスプレイ最上行に表示します。

構 文 (1)-1

(1)-2
PWD

解 説

PWD : 現在のカレント・ドライブ（カレント・ディレクトリ）のパス名をディスプレイ最上行に表示する

・PWD は、CHDIR ステートメントで変更されるカレント位置のパス名を表示します。（本節 3. CHDIR を参照）

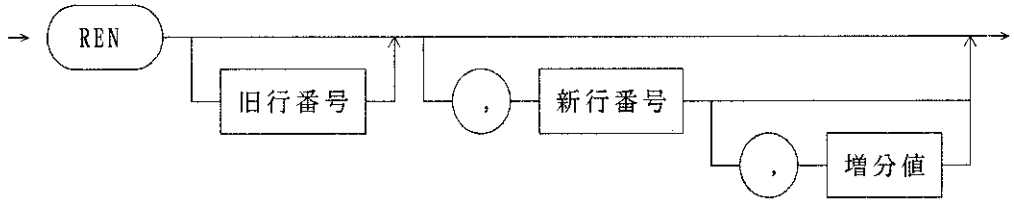
23. REN

概 要

プログラムの行番号を新しく付け直します。
このコマンドは、プログラム実行中は使用できません。

構 文

(1)-1



(1)-2
REN 〔〔旧行番号〕〔, 新行番号〔, 増分値〕〕〕

注) カンマ(,)をスペースにしても構いません。
旧行番号、新行番号、増分値の設定範囲は 1～65535 です。
新行番号と増分値は、省略すると10に設定されます。

解 説

・旧行番号は、番号の付け替えが始められる現在のプログラム中の行番号です。

・新行番号は、新しく付ける先頭の行番号です。

・増分値は、新しく付ける行番号の増分量を示します。

・REN は、GOTO, GOSUB などで使っている行番号にも、新しい行番号に対応して変更します。

・REN は、65535 を超える行番号を使えません。また、プログラム行の順序を変えるような指定をしてはいけません。

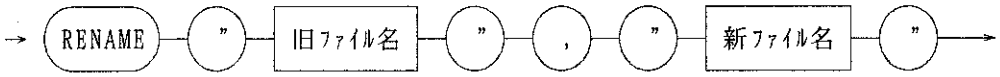
例

REN	: プログラムの先頭行を10にして、10ステップで最終行まで行番号を付け直す
REN 50,100	: プログラムの行番号50を 100にして、それ以降を10ステップで最終行まで行番号を付け直す
REN 10,50,5	: プログラムの行番号10を50にして、それ以降を 5ステップで最終行まで行番号を付け直す

24. RENAME

概 要 指定されたメディアに記録されているファイル名を変更します。

構 文 (1)-1



(1)-2
RENAME "〔ドライブ名:〕旧ファイル名", "〔ドライブ名:〕新ファイル名"

解 説
- ファイル名だけの変更で、内容は変更しません。
- 新ファイル名と同じ名前のファイルが、既に存在する場合や、新ファイル名と旧ファイル名が同じ場合は、動作しません。

例

RENAME "AAA.BAS", "BBB.BAS": 現在のカレント・ドライブ（カレント・ディレクトリ）のメディアに記録されているファイル名AAA.BAS をBBB.BAS にファイル名を変更する

RENAME "MA:AAA.BAS", "BBB.BAS"
: メモリ・カードA ドライブのメディアに記録されているファイル名AAA.BAS を BBB.BAS にファイル名を変更する

※ ファイルの取り扱いについては、「2.4 ファイルの管理」を参照して下さい。

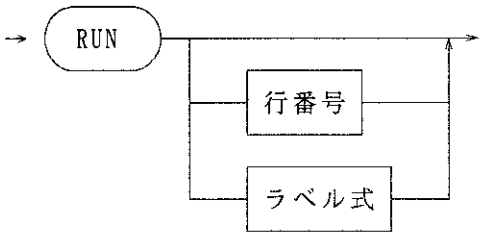
25. RUN

概 要

メモリ上にあるBASIC プログラムを実行します。
また、ドライブ内のメディアからプログラムをメモリ上にロードし、そのプログラムを実行します。

構 文

(1)-1



(1)-2

RUN 〔行番号 | ラベル式 | "〔ﾄﾞﾗｲﾌﾞ名:〕〔ﾃﾞｨﾚｸﾄﾘ名/〕ﾌｧｲﾙ名"〕

解 説

・BASIC プログラムを指定した行から実行させます。

・行番号を指定しないときは、先頭行から実行します。

・RUN を実行すると、プログラムを実行する前にすべての変数をクリアし、配列宣言なども強制的に無設定状態になります。

例

RUN	:	現在メモリ上にあるプログラムを実行する
RUN 100	:	現在メモリ上にあるプログラムの行番号100から実行する
RUN *Lbl	:	現在メモリ上にあるプログラムの*Lbl行から実行する
RUN "AAA.BAS"	:	現在のカレント・ドライブ（カレント・ディレクトリ）のメディアに記録されているAAA.BAS ファイルをロード後、実行する
RUN "MA:BBB.BAS"	:	メモリ・カードA ドライブ内のメモリ・カードに記録されているBBB.BAS ファイルをロード後、実行する

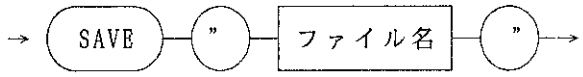
26.　SAVE

概　要

ドライブ名が指定された場合はそのドライブ内の、指定されなかった場合は現在のカレント・ドライブ内のメディアに記録されている指定ファイルを保存します。

構　文

(1)-1



(1)-2

SAVE "〔ドライブ名:〕〔ディレクトリ名/〕ファイル名"

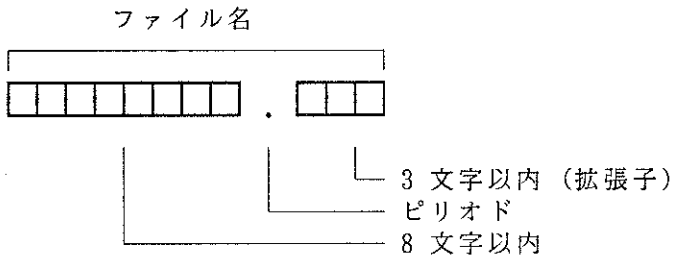
解　説

・編集したプログラム（行番号の付いている文の先頭から最後まで）をファイル名で指定した名前でファイルに登録します。

・既存のファイル名を指定すると、同一ファイルの更新とみなされ、その内容が更新されます。

注意

ファイル名は、数字、英字および記号（ダブル・クォーテーション(") は除く）を使い、以下のように指定します。



例

SAVE "AAA.BAS"　：現在のカレント・ドライブ（カレント・ディレクトリ）のメディアに記録されているAAA.BAS ファイルの内容を保存する

SAVE "MA:BBB.BAS"　：メモリ・カードA ドライブ内のメモリ・カードに記録されているBBB.BAS ファイルの内容を保存する

※　ファイルの取り扱いについては、「2.4 ファイルの管理」を参照して下さい。

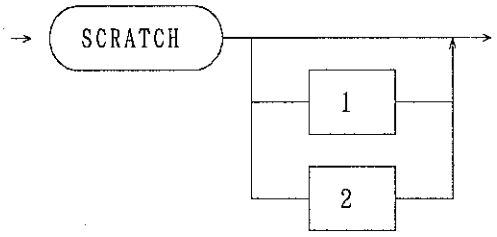
27.　SCRATCH

概　要

メモリ上にあるプログラムを消去します。
このコマンドは、プログラム実行中は使用できません。

構　文

(1)-1



(1)-2

SCRATCH 〔1 | 2 〕

例

SCRATCH　　：現在メモリ上にあるプログラムを消去します。（データ、プロシージャ）

SCRATCH　1　　：プログラム・データ（変数確保値等）のみ初期化します。（データ）

SCRATCH　2　　：現在メモリ上にあるプログラム・リストを消去します。（プロシージャ）

28. STEP

概 要

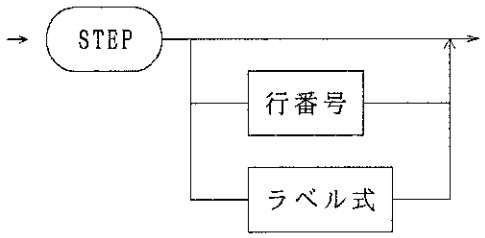
メモリ上にあるプログラムを 1行実行します。行番号指定がある場合はその行を 1行実行し、ない場合は直前に終了した行の次の行を実行します。

プログラムがRUN 状態でプログラム中のPAUSE 文、またはコマンド・ラインからの"PAUSE" 入力によるプログラム一時停止状態からSTEPコマンドでステップ実行を行い、その後、任意の行からのCONTコマンドによるプログラム実行の継続が可能です。

また、プログラムがRUN 状態でない場合にもステップ実行を行うことができます。ただし、この場合、STEPコマンドによる実行の最初だけは、STEP開始行（開始ラベル）と実行開始行を明確にして下さい。以降は、通常のSTEPコマンドで 1行ずつ実行を行うことができます。
なお、この場合、CONTコマンドによる現在の実行行からの実行の継続はできません。

構 文

(1)-1



(1)-2
STEP〔行番号｜ラベル式〕

解 説

STEP : 直前に終了した行の次の行を 1行実行する

STEP 100 : 行番号 100の行を 1行実行する

STEP *Lb1 : *Lb1行を 1行実行する

29. STOP

概 要 BASIC プログラムの実行を停止させます。

構 文 (1)-1

(1)-2
STOP

解 説 ・BASIC プログラムの実行を停止、またはBASIC プログラム自身で停止させます。
CONTまたはSTEPコマンドで停止行からの実行を再開することができます。

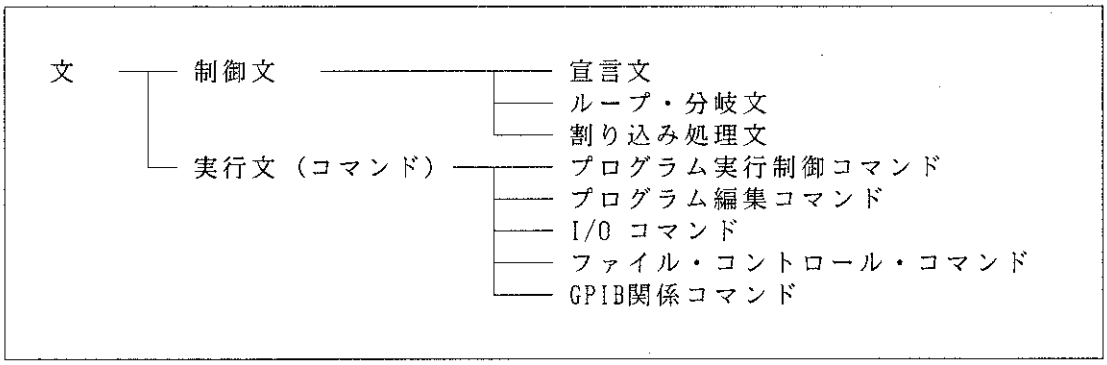
4.　BASIC ステートメント

4.1　プログラミングのきまり

4.1.1　プログラム構造

(1)　文

BASIC プログラムは各種の文の集まりです。
文は大きく分けて、制御文と実行文（コマンド）によって構成されています。
各文は、キー・ワードと式から構成されます。この構成を取り決めたものが文法の構文規則です。



(2)　キー・ワード

BASIC で、あらかじめ意味と用途を定めている言葉を「キー・ワード」と呼びます。キー・ワードと同じ名前を、その他の目的に使用できません。
キー・ワードの名前（フルネーム）で、使用頻度が高く、かつ長い名前のものにはショート・ネームがあります。
表示をフルネームからショート・ネームに変更するには、CONTROL コマンドでコントロール・レジスタ3 を 1にします。フルネームに戻すには、コントロール・レジスタ3 を 0にします。

キー・ワード一覧表は、[表4-1]を参照して下さい。
フルネームとショート・ネームの対応表は、[表4-2]を参照して下さい。

表 4 - 1　キー・ワード一覧

AND	AS	ASCII	USING	WAIT	XOR
BOR	BREAK	BXOR	BAND	BINARY	BNOT
CLS	CMD	COLOR	CASE	CLEAR	CLOSE
COUNT	CSR	COMMON	CONSOLE	CONTINUE	COS
DATE	DELAY	CSRLIN	CSRPOS	CURSOR	DATA
ELSE	ENABLE	DELIMITER	DIM	DISABLE	DSTAT
ERRM	ERRN	END	ENT	ENTER	ENTERF
FORMAT	FRE	ERROR	EVENT	EXP	FOR
GPRINT	IF	GLIST	GLISTN	GOSUB	GOTO
INTERFACE	INTR	INKEY	INP	INPUT	INTEGER
LINE	LISTEN	ISRQ	KEY	LABEL	LEN
LOCKOUT	LOG	LISTN	LLIST	LLISTN	LOCAL
OFF	ON	LPRINT	NEXT	NOT	NUM
OUTPUTF	PI	OPEN	OR	OUTPUT	OUT
PRINTF	READ	POS	PRINT	PRINTER	PRF
RESTORE	REQUEST	REAL	REM	REMOTE	REN
SPRINTF	SQR	RETURN	SELECT	SEND	SIN
THEN	TIME	SRQ	TALK	TAN	TEXT
UNT	USE	TIMER	TO	TRIGGER	UNL

表 4 - 2　フルネームとショートネームの対応表

フルネーム	ショートネーム
COMMON	COM
CURSOR	CSR
ENTER	ENT
INPUT	INP
OUTPUT	OUT
PRINTF	PRF
SPRINTF	SPRF
USING	USE
PRINT	?

(3)　式

式は、オブジェクトと演算子からなり、文法上、式を指定できるところならば、どこにでも置くことができます。（ただし、従来のBASIC との互換性を考えて、IF文の条件式では＝を等号と解釈するため、代入式を書くことはできません。）
式には、演算結果の最終値がどのデータ型を取るかによって、以下の 4通りあります。

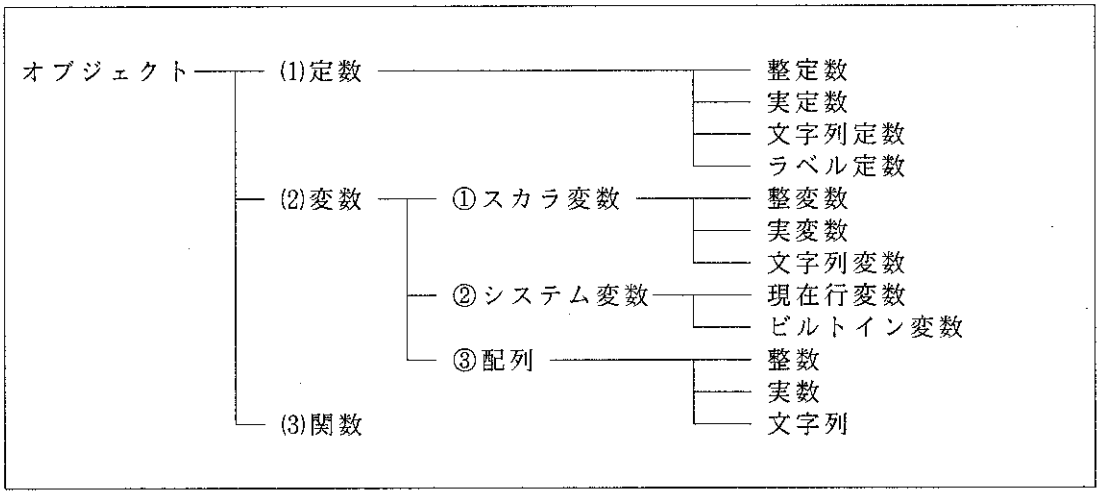
〈算術式 〉〈 文字列式 〉〈 論理式 〉〈 ラベル 〉

算術式 ： 整数値または実数値になる。

論理式 ： 内部に論理演算子を含むということには関係なく構文で決まり、最終値を論理値として評価する（つまり、0 が偽、0 以外が真になる）。

4.1.2　オブジェクト

BASICが処理する対象となるものを「オブジェクト」と呼びます。オブジェクトには定数、変数および関数があり、各データ型は以下のようになっています。



(1)　定数

・整定数

プログラム内部で小数点のない定数は整数とみなし、内部で 4バイト使い表現するので、－2,147,483,648 ～＋2,147,483,647 までの数値を表現できます。

・実定数

小数点付や1E＋20のような浮動小数点表現の定数は実数とみなし、内部で 8バイト(IEEE)を使って表現するので、約－1E＋308 ～1E＋308 まで表現でき、15桁の精度をもちます。

・文字列定数

文字列を表現するには、その文字列をダブル・クォーテーション(") で囲みます。
文字列は "" の空文字列から最大 128文字まで指定できます。構成する文字の単位は 8ビットで、 0～255 までの 256種類の文字単位を表現できます。
文字コードは ASCIIを使用し、 128～255 までは特殊なシンボルが登録されています。
コードがキー・ボードに割り当てられていないものをプログラムで表現するために、またはINPUT 文に対する入力のために、＼を使用して＼f(フォーム・フィールド) という方法があります。同様にして、ダブル・クォーテーション (")を文字列に含めるために＼" と書くことができます。
ASCII の制御文字を表現するために、以下のエスケープ・シーケンスが用意されています。

ｴｽｹｰﾌﾟ･ｼｰｹﾝｽ	意味	8進	10進
＼b	バック・スペース	010	8
＼t	水平タブ	011	9
＼n	ライン・フィード(ニューライン)	012	10
＼v	垂直タブ	013	11
＼f	フォーム・フィード(クリア・スクリーン)	014	12
＼r	キャリッジ・リターン	015	13

・ラベル定数

文番号に変わるものです。プログラムの先頭でアスタリスク (*)を付けて宣言します。
使用できる文字は変数と同じですが、変数ではないので代入できません。またラベルが書ける位置は構文的に限られていて、[4.3　ステートメント文法と活用] で説明するラベル行番号、または分岐先と書かれている部分です。

(2)　変数

変数名は、文字を先頭とするアルファニューメリックで構成し、最大20文字です。

変数名の最後が $の場合　　：文字列変数になる。
変数名の最後が(整数値)の場合　：配列型の変数とみなされる。
　変数は特に INTEGER文で宣言されなければ実数型になる。

表 4 - 3　アルファニューメリック

```
1, 2, 3, 4, 5, 6, 7, 8, 9, 0
a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t
u, v, w, x, y, z
A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T
U, V, W, X, Y, Z
 _
```

例）変数の型

```
Value, A123          : 数値変数 (実数型)
Str$, K67$           : 文字列変数 (18文字分)
INTEGR Val, M00      : 数値変数 (整数型)
INTEGR Ary(10)       : 数値配列変数 (整数型)
DIM Ss$[100]         : 文字列変数 (100 文字分)
DIM Sarr$(5)[10]     : 文字列配列変数 (10文字分を 5個)
DIM Array(3)         : 数値配列変数 (実数型)
```

① スカラ変数

・整変数
・実変数
・文字列変数

数値型の変数は、特に初期化しない限り 0が割りあてられます。したがって、特定の値に初期化すべき変数は、プログラム内で明示的に値を代入する必要があります。
各データ型に格納できる値は、定数の場合と同じ大きさです。
文字列には、文字列定数と同様に長さの属性があります。長さを宣言するには、DIM 文を使用します。

```
DIM  string$[100]
```

宣言をせずに参照すると、18文字の文字列であるとみなされます。
文字列はサブ・ストリング演算子 ([ ])を使用して、文字列の一部を扱うことができます。

4.1.3 項の [(7)　サブ・ストリング演算子] を参照して下さい。

```
string$ ="ADVANTEST CORPORATION"
PRINT string$[1,14] ;"."
```

結果

```
ADVANTEST CORP.
```

② システム変数

システム変数には、現在行記憶変数 (@)と組み込み変数 (PI) が用意されています。現在行記憶変数 (@)は、現在実行中のプログラム行番号を保持している変数です。また、組み込み変数 (PI) は、円周率の近似値3.14159265359 を返します。
なお、これらの変数は予約語なので値の代入は行えません。

③ 配列

配列の宣言は、DIM およびINTEGER ステートメントを使用します。数値配列変数および文字列配列変数は、配列宣言をしないと使用できません。数値配列変数は、DIM またはINTEGER ステートメントで、文字列配列変数はDIM ステートメントで、必ず使用前に必要領域を宣言して下さい。宣言なしに使用すると、エラー・メッセージが表示されます。
数値配列変数、文字列配列変数、共に指定できる配列の次元数は 5次元までとなっています。この際、添字は必ず 1から割り当てられます。

```
INTEGR A(3,5)      :  数値配列変数 (整数型の 2次元配列)
DIM A(3,4,5,6)     :  数値配列変数 (実数型の 4次元配列)
DIM S$(5)          :  文字列配列変数 (18文字分を 5個の 1次元配列)
DIM S$(3,5)[128]   :  文字列配列変数(128文字分を 5×3 個の 2次元配列)
```

(3)　関数

関数はすべて組み込み型です。関数は、その戻り値で整数型、実数型または文字列型に分けられます。また、この関数呼び出しを演算式の中に記述できるため、変数と同じようになります。

```
string$ ="ADVANTEST"
PRINT string$
A = NUM("A")
a = NUM("a")
FOR idx = 1 TO LEN(string$);
      b = NUM(string$[idx;1]) - A + a
      string$[idx;1]=CHR$(b)
NEXT idx
PRINT string$
```

結果

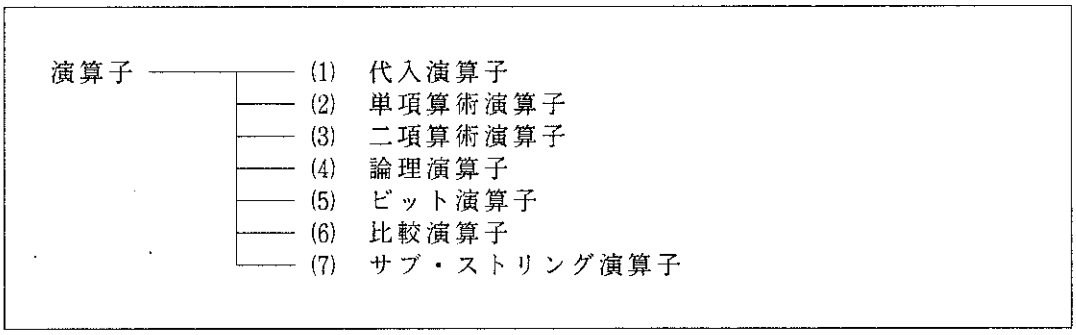
```
ADVANTEST
advantest
```

・組み込み関数

関数	機能
SIN(算術式) COS(算術式) TAN(算術式) ATN(算術式)	正弦 (sin) 余弦 (cos) 正接 (tan) 逆正接 ($\tan^{-1}$)　　角度の単位= ラジアン
LOG(算術式)	自然対数
SQR(算術式)	平方根
ABS(算術式)	絶対値
NUM(文字列式)	文字列式の先頭の 1文字の ASCIIコードを返す。 例)　NUM("A") ---> 65
CHR$ (算術式)	算術式の値に対応する ASCII文字 1文字の文字列式を返す。 例)　CHR$(65) ---> "A"
LEN(文字列)	文字列式の長さを返す。 例)　LEN("ADVANTEST") ---> 9
POS(文字列式1,文字列式2)	文字列式1の中から、文字列式2がある位置の先頭の位置を返す。 例)　POS("ADVANTEST", "AN") ---> 4

4.1.3　演算子

オブジェクトを操作するものが演算子です。演算子とオブジェクトの組み合わせで式を構成します。



(1)　代入演算子

従来の BASICにあった LETというキー・ワードはありません。代入式はそれ自体の値をもち、 1つの式となります。

```
PRINT a=1                ---> 1.0
PRINT a$="ADVANTEST"     --->"ADVANTEST"
PRINT (a=1)+a            ---> 2.0
```

代入演算子を以下に示します。

```
=       : 普通の代入
          文字列変数への代入では、右辺の有効文字数分だけ転送します。
          例)   DIM string$ [20]
                PRINT LEN(string$ ="12345")
              結果
                        5
=       : =の左辺のデータ型に変換して代入
          例)   string$ = 123.456   ---> "123.456"
                numeric = "123"     ---> 123
                integer = 123.456   ---> 123
+=      :  a  += 10   --->  a = a  + 10
-=      :  a  -= 10   --->  a = a  - 10
*=      :  a  *= 10   --->  a = a  * 10
/=      :  a  /= 10   --->  a = a  / 10
%=      :  a  %= 10   --->  a = a  % 10
=<      : 文字列を左詰めで代入
=>      : 文字列を右詰めで代入
```

(2)　単項算術演算子

```
－       : マイナス符号
＋       : プラス符号
＋＋     : 前置/後置インクリメント
           前置 b = ++a     ···· aに 1を加えてから、 bに代入する。
           後置 b = a++     ···· bに代入してから、 aに 1を加える。
－－     : 前置/後置デクリメント
           前置 b = --a     ···· aから 1を引いてから、 bに代入する。
           後置 b = a--     ···· bに代入してから、 aから 1を引く。
```

例）　a = 10: PRINT a++: PRINT a: PRINT --a: PRINT --a: PRINT a

結果

```
10.0
11.0
10.0
9.0
9.0
```

(3)　二項算術演算子

＋	: 加算	％	: モジュロ（余り）
－	: 減算	^	: 累乗
＊	: 乗算	&	: 文字列の連結
／	: 割算		

(4)　論理演算子

NOT　　AND　　OR　　XOR

(5)　ビット演算子

数式は整数型のみ有効で、実数型ではエラーになります。

BNOT　　BAND　　BOR　　BXOR

(6)　比較演算子

比較演算子には以下のものがあり、真ならば 1を、偽ならば 0を取ります。BASICの構文上で比較演算を行うところでは、演算結果として最終的に 0になったとき、偽と判定します。それ以外の値はすべて真になります。

```
＝       : イコール
＜＞     : ノットイコール
＜
＞
＜＝
＞＝
```

この比較演算はIF文の条件では、必らず論理演算を行うので、演算子の＝は無条件に比較演算子であるとみなします。したがって、代入式をIF文の条件式内に含めることはできません。

(7)　サブ・ストリング演算子

文字列式の一部を文字列として指定できます。

文字列式［算術式1,　算術式2］：文字列式の先頭から、算術式1 が表現する値だけ進んだ所から算術式2 が表わす値の所までをサブ・ストリングとする。

"ADVANTEST" [1,5]　---> "ADVAN"

文字列式［算術式1 ; 算術式2］：文字列式の先頭から、算術式1 が表現する値だけ進んだ所から算術式2 が表わす値の文字数分をサブ・ストリングとする。

"ADVANTEST" [6;4]　---> "TEST"

●演算子の優先順位一覧表

演算子	優先順位	種別
( )	1	
= += -= *= /= %= =< =>	2	代入演算子
OR XOR	3	論理演算子
AND	4	論理演算子
BOR	5	ビット演算子
BXOR	6	ビット演算子
BAND	7	ビット演算子
= == <>	8	比較演算子
< > <= >=	9	比較演算子
- + &	10	二項算術演算子
* / %	11	二項算術演算子
NOT	12	論理演算子
- +	13	単項算術演算子
^	14	二項算術演算子
A++ A--	15	単項算術演算子
++A --A	16	単項算術演算子
BNOT	17	ビット演算子

●注意すべき使用例

```
10 A=1 : B=2
20 IF ((A=1) AND (B=2)) THEN  …  ○
20 IF A=1 AND B=2 THEN        …  ×
```

上記の比較例において、×側の書式では優先順位の関係上、AND が =より先に処理が行われるため、期待する結果は得られません。必ず括弧を使用して処理の優先順位を明確に記述して下さい。

4.2　各種ステートメント

4.2.1　ステートメント機能一覧

(1)　基本ステートメント　(1/2)

ステートメント	機能
ABS	絶対値を返す
ATN	逆正接を返す
CHR$	指定したキャラクタ・コードを持つ値を返す
CLS	ディスプレイ表示をクリア
COLOR	ディスプレイ上の文字色を指定
COMMON	合成プログラムの変数渡し
CONSOLE	スクロール範囲を指定
COS	余弦値を返す
CSRLIN	位置座標y を返す
CSRPOS	位置座標x を返す
CURSOR	カーソルを移動
DATA	READ文で読み込むための数値、文字列を定義
DATE$	本器に内蔵している時計(RTC) の日付の値を読み出す
DIM	配列変数または文字列変数を定義
DISABLE INTR	割り込みの受付を禁止
ENABLE INTR	割り込みの受付を許可
ERRM$	エラー・メッセージを返す
ERRN	エラー番号を返す
EXP	定数e の引数乗を求め、その値を返す
FOR-TO-STEP, NEXT, BREAK, CONTINUE	ループ処理を行う
FRE	BASIC プログラム・バッファの残量を返す
GOSUB, RETURN	サブルーチンへの分岐、復帰
GOTO	指定行への分岐
GPRINT	数値または文字列をGPIBへ出力
IF-THEN, ELSE, END IF	条件付分岐
INKEY$	パネル・キーのコードを返す
INPUT	キーからの入力
INTEGER	変数が整数型であることを定義
LEN	文字列の文字桁数を返す
LOG	自然対数値を返す
LPRINT	数値または文字列をシリアル・ポートへ出力
NUM	指定した文字のアスキ・コードを返す
OFF ERROR	BASIC エラーを検出したときの分岐の解除
OFF ISRQ	ISRQによる割り込み分岐を解除
OFF KEY	キー入力による割り込み分岐を解除
OFF SRQ	SRQ による割り込み分岐を解除
ON DELAY	指定時間経過後に分岐
ON ERROR	BASIC エラーを検出したときの分岐の定義
ON ISRQ	本器内部要因による割り込み分岐を定義
ON KEY	キー入力による割り込み分岐を定義
ON SRQ	GPIB外部SRQ 信号による割り込み分岐を定義
PRINT [USING]	数値または文字列を表示

(2/2)

ステートメント	機能
PRINTER	プリンタのGPIBアドレスを設定
PRINTF	数値または文字列を表示
READ	DATA文の定数を変数に代入
REM	注釈
RESTORE	次のREAD文で読み込むDATA行を指定
SELECT, CASE, END SELECT	式の値を条件として複数の分岐を行う
SPRINTF	PRINTFの書式に従った結果を文字列へ代入
TIME$	本器に内蔵されている時計(RTC) の時刻の値を返す
TIMER	内蔵システム時間の読み出しおよびリセット
WAIT	指定時間だけ待つ
WAIT EVENT	指定したイベントが発生するまで待つ

(2)　GPIB制御用ステートメント

ステートメント	機能
CLEAR	デバイス・クリア
DELIMITER	ブロック・デリミタを指定
ENTER	GPIBからの入力
INTERFACE CLEAR	GPIBインタフェース・クリア
LOCAL	リモート・コントロールを解除
LOCAL LOCKOUT	ローカル・ロックアウト
OUTPUT	GPIBへの出力
REMOTE	リモート・コントロール
REQUEST	ステータス・バイトを設定
SEND	GPIBへコマンド、データなどを出力
SPOLL	ステータス・バイトの読み出し
TRIGGER	グループ・エグゼキュート・トリガを出力

(3)　ファイル制御用ステートメント

ステートメント	機能
CLOSE	ファイルを閉じる
DSTAT	ﾌﾛｯﾋﾟｰ･ﾃﾞｨｽｸのﾃﾞｨﾚｸﾄﾘの内容を BASIC変数に取り込む
ENTER [USING]	ファイルからデータを読み込む
OFF END	ON ENDで指定した処理を解除
ON END	エンド・オブ・ファイル時の処理を定義
OPEN	ファイルをオープン
OUTPUT [USING]	ファイルにデータを出力（書き込む）

4.2.2　ステートメント文法一覧

(1)　基本ステートメント　(1/2)

ステートメント	文法
ABS	ABS(数値表現式)
ATN	ATN(数値表現式)
CHR$	CHR$ (数値表現式)
CLS	CLS
COLOR	COLOR<文字色>[, 反転モード]
COMMON	COMMON 数値変数 | 文字列変数 | 配列変数
CONSOLE	CONSOLE <開始行>,<終了行>
COS	COS(数値表現式)
CSRLIN	CSRLIN
CSRPOS	CSRPOS
CURSOR	CURSOR <X軸>,<Y軸>
DATA	DATA 数値定数 | 文字列定数 {, 数値定数 | 文字列定数}
DATE$	(1) DATE$ (2) DATE$ ="YY/MM/DD"
DIM	DIM <B> | <C> {, <B> | <C>}
DISABLE INTR	DISABLE INTR
ENABLE INTR	ENABLE INTR
ERRM$	ERRM$(エラー番号)
ERRN	ERRN
FOR-TO-STEP,NEXT,BREAK,CONTINUE	FOR 数値変数= 数値表現式 TO 数値表現式 [STEP数値表現式] [BREAK] [CONTINUE] NEXT [数値変数]
EXP	EXP[(数値表現式)]
FRE	FRE(数値)
GOSUB,RETURN	GOSUB 整数 | ラベル式 RETURN
GOTO	GOTO 整数 | ラベル式
GPRINT	GPRINT [A {, | ;A}]
IF-THEN,ELSE,END IF	(1) IF <条件式> THEN <文> (2) IF <条件式> THEN [ELSE IF <条件式> THEN] [複文] [ELSE] [複文] END IF
INKEY$	INKEY$
INPUT	INPUT ["< 文字列>",] A {,A}
INTEGER	INTEGER <B> {,<B>}
LEN	LEN(文字列表現式)

A : 数値表現式 \| 文字列表現式
B : 数値変数名 [(数値表現式 {, 数値表現式})]
C : 文字列変数 [数値表現式]
D : 数値変数= 数値表現式
E : 文字列変数= \| =< \| =>文字列表現式

(2/2)

ステートメント	文法
LOG	LOG(数値表現式)
LPRINT	LPRINT [A {, | ;A}]
NUM	NUM(文字表現式)
OFF ERROR	OFF ERROR
OFF ISRQ	OFF ISRQ
OFF KEY	OFF KEY [キーコード]
OFF SRQ	OFF SRQ
ON DELAY	ON DELAY 時間 GOTO | GOSUB 整数 | ラベル式
ON ERROR	ON ERROR GOTO | GOSUB 整数 | ラベル式
ON ISRQ	ON ISRQ GOTO | GOSUB 整数 | ラベル式
ON KEY	ON KEY キーコード GOTO | GOSUB 整数 | ラベル式
ON SRQ	ON SRQ GOTO | GOSUB 整数 | ラベル式
PRINT [USING]	(1) PRINT [A {, | ;A}] (2) PRINT USING 書式設定式 ; {,A}
PRINTER	PRINTER 数値表現式
PRINTF	PRINTF 書式表現式 {,A}
READ	READ 入力項目 {, 入力項目}
REM	REM [文字列] または ![文字列]
RESTORE	RESTORE 整数 | ラベル式
SELECT, CASE, END SELECT	SELECT <数式 | 文字列式> CASE <数式 | 文字列式> 複文 [CASE ELSE] [複文] END SELECT
SPRINTF	SPRINTF 文字列変数 書式指定 {,A}
TIMER	TIMER(0 | 1)
TIME$	(1) TIME$ (2) TIME$ ="HH:MM:SS"
WAIT	WAIT 時間
WAIT EVENT	WAIT EVENT <イベント番号>

A : 数値表現式 | 文字列表現式

●PRINT USING の書式指定は、以下に示すイメージ仕様をｺﾝﾏ (,) で区切って指定します。

ｲﾒｰｼﾞ仕様

D : D の数で出力桁数を指定する。指定フィールドの余った部分にはスペースが入る。
Z : Z の数で出力桁数を指定する。指定フィールドの余った部分には0 が入る。
K : 式の値をそのまま表示する。
S : S の位置に+ または- のサイン・フラグを付けて表示する。
M : M の位置に、負のときは- 、正のときはスペースを付けて表示する。
. : . の位置に小数点がくるように位置を合わせて表示する。
E : 指数形式 (e, 符号, 指数) で表示する。
H : K と同じ。ただし、小数点にカンマ(,) を使う。
R : . と同じ。ただし、小数点にカンマ(,) を使う。
* : * の数で出力桁数を指定する。指定フィールドの余った部分には *が入る。
A : 1 文字を表示する。

k　　：文字列式をそのまま表示する。
X　　：スペース 1文字を表示する。
ﾘﾃﾗﾙ　：書式指定式にリテラルを書くときは＼”で囲む。
B　　：式の値をASCII コードとして表示する。
@　　：改ページする。
+　　：表示の位置を同じ行の先頭に移動させる。
-　　：改行する。
#　　：最後に改行しない。
n　　：数字で各イメージ仕様の繰り返し回数を指定できる。

●PRINTFの書式指定は、以下に示す方法で %に続けて指定します。

%〔-〕〔0〕〔m〕〔. n〕文字

-　　：指定されたフィールド内で左詰めにする（指定がなければ右詰め）。
0　　：指定フィールドの余った部分に詰める文字を 0にする。
m　　：m 文字分のフィールドを取る。
. n　：n 桁の精度で出力する。文字列の場合は、この値が実際の文字列の長さになる。
文字　：d ; 符号付10進数　　s ; 文字列
　　o ; 8 進数　　e ; 浮動小数点表現（指数形式）
　　x ; 16進数　　f ; 浮動小数点表現

(2)　GPIBステートメント

ステートメント	文法
CLEAR	CLEAR [装置ｱﾄﾞﾚｽ {, 装置ｱﾄﾞﾚｽ}]
DELIMITER	DELIMITER 数値表現式
ENTER	ENTER 装置ｱﾄﾞﾚｽ ; B {,B}
INTERFACE CLEAR	INTERFACE CLEAR
LOCAL	LOCAL [装置ｱﾄﾞﾚｽ {, 装置ｱﾄﾞﾚｽ}]
LOCAL LOCKOUT	LOCAL LOCKOUT
OUTPUT	OUTPUT 装置ｱﾄﾞﾚｽ {, 装置ｱﾄﾞﾚｽ} ; A {,A}
REMOTE	REMOTE [装置ｱﾄﾞﾚｽ {, 装置ｱﾄﾞﾚｽ}]
REQUEST	REQUEST 整数
SEND	SEND CMD | LISTEN | TALK[数値表現式, ...]
	SEND DATA [数値表現式 | 文字表現式 | EOI]
	SEND UNL | UNT
SPOLL	SPOLL (装置ｱﾄﾞﾚｽ)
TRIGER	TRIGER [装置ｱﾄﾞﾚｽ {, 装置ｱﾄﾞﾚｽ}]

A : 数値表現式 | 文字列表現式
B : 数値変数 [文字列変数

(3)　ファイル制御用ステートメント

ステートメント	文法
CLOSE	CLOSE #FD ｜*
DSTAT	(1) DSTAT 0 〈ファイル数〉
	(2) DSTAT 〈index〉〈ﾌｧｲﾙ名〉〈属性〉〈ｻｲｽﾞ〉〈ｾｸﾀ数〉 〈年〉〈月〉〈日〉〈時〉〈分〉〈開始ｾｸﾀ〉
	(3) DSTAT ;SELECT 〈文字列〉 COUNT 〈変数〉
ENTER [USING]	(1) ENTER #FD ; 入力項目 {, 入力項目}
	(2) ENTER #FD USING "ｲﾒｰｼﾞ仕様" ; 入力項目 {, 入力項目} }
OFF END	OFF END #FD
ON END	ON END #FD GOTO ｜GOSUB 整数｜ラベル式
OPEN	OPEN "ﾌｧｲﾙ名" FOR 処理ﾓｰﾄﾞ AS #FD [; ﾀｲﾌﾟ]
OUTPUT [USING]	(1) OUTPUT #FD ; 出力項目 {, 出力項目}
	(2) OUTPUT #FD USING " ｲﾒｰｼﾞ仕様" ; 出力項目 {, 出力項目}

FD　　：ファイル・ディスクリプタ
処理モード　：INPUT ｜OUTPUT
タイプ　　：BINARY｜TEXT｜ASCII

●ENTER USING のイメージ仕様
ｲﾒｰｼﾞ仕様
D　：D の数を入力桁と解釈して読み込み、入力項目の変数に代入する。
Z　：D と同じ。
K　：1 行読み込み、数値データに変換し、入力項目の変数に代入する。
S　：D と同じ。
M　：D と同じ。
.　：D と同じ。
E　：K と同じ。
H　：K と同じ。ただし、小数点にカンマ(,) を使う。
*　：D と同じ。
A　：A の数分の文字を読み込み、文字列変数に代入する。
k　：1 行読み込み文字列変数に代入する。
X　：1 文字のデータを読み飛ばす。
ﾘﾃﾗﾙ　：＼" で囲まれた文字列の数のデータを読み飛ばす。
B　：1 文字読み込み、入力項目に ASCIIコードとして代入する。
@　：1 バイトのデータを読み飛ばす。
+　：@ と同じ。
-　：@ と同じ。
#　：ENTER では無視される。
n　：数字で各イメージ仕様の繰り返し回数を指定できる。

●OUTPUT USINGのイメージ仕様

ｲﾒｰｼﾞ仕様

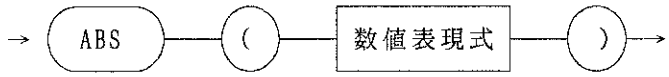
D	: D の数で出力桁数を指定する。指定フィールドの余った部分にはスペースが入る。
Z	: Z の数で出力桁数を指定する。指定フィールドの余った部分には0 が入る。
K	: 式の値をそのまま表示する。
S	: S の位置に+ または- のサイン・フラグを付けて出力する。
M	: M の位置に、負のときは- 、正のときはスペースを付けて出力する。
.	: . の位置に小数点がくるように位置を合わせて出力する。
E	: 指数形式（e, 符号, 指数）で出力する。
H	: K と同じ。ただし、小数点にカンマ(,) を使う。
R	: . と同じ。ただし、小数点にカンマ(,) を使う。
*	: * の数で出力桁数を指定する。指定フィールドの余った部分には* が入る。
A	: 1 文字を出力する。
k	: 文字列式をそのまま出力する。
X	: スペース 1文字を出力する。
ﾘﾃﾗﾙ	: ＼”で囲まれた文字列を出力項目とは無関係にそのまま出力する。
B	: 式の値をASCII コードとして出力する。
@	: フォーム・リード（改ページ）を出力する。
+	: キャリッジ・リターンを出力する。
-	: ライン・フィード（改行）を出力する。
#	: 最後の項目の後ろにライン・フィードを付けない。
n	: 数字で各イメージ仕様の繰り返し回数を指定できる。

4.3　ステートメント文法と活用

1.　ABS

概　要　絶対値を返す組み込み関数です。

構　文　(1)-1



(1)-2
ABS(数値表現式) →

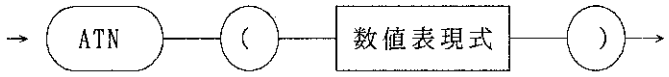
例

```
PRINT ABS(-2)
A=ABS(B)
```

2.　ATN

概　要　逆正接（アークタンジェント）を返す組み込み関数です。

構　文　(1)-1



(1)-2
ATN(数値表現式) →

例

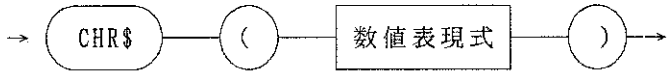
```
PRINT ATN(PI/2)
A=ATN(-3.14/B)
```

注）　ATNにて返される値は、ラジアンで $-\pi/2$から$\pi/2$までの範囲です。

3.　CHR$

概　要　指定したキャラクタ・コードを持つ値を返す組み込み関数です。

構　文　(1)-1



(1)-2
CHR$(数値表現式) →

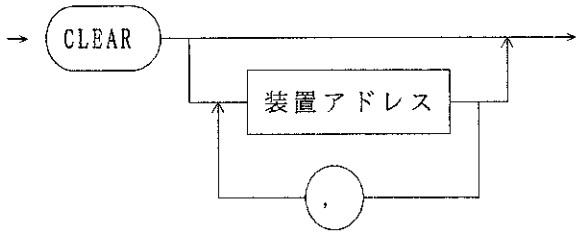
例

```
PRINT CHR$(65)
A$=CHR$(B)
```

4.　CLEAR

概　要　GPIB上に接続されたすべての装置または選択された特定の装置を初期状態にします。

構　文　(1)-1



(1)-2
CLEAR 〔装置アドレス｛，装置アドレス｝〕

解　説
- 装置アドレスを指定せずにCLEAR だけを実行すると、GPIB上にユニバーサル・コマンドのデバイス・クリア(DCL) を送ります。これによって、GPIBに接続されているすべての装置を初期状態にできます。
- CLEAR に続いて装置アドレスを指定すると、装置アドレスで指定された装置のみをアドレスし、アドレス・コマンドのセレクト・デバイス・クリア(SDC) を送ります。これによって、特定の装置のみを初期状態にします。なお、装置アドレスは複数指定できます。
- CLEAR で各装置に設定される初期状態の定義は、各装置に依存します。

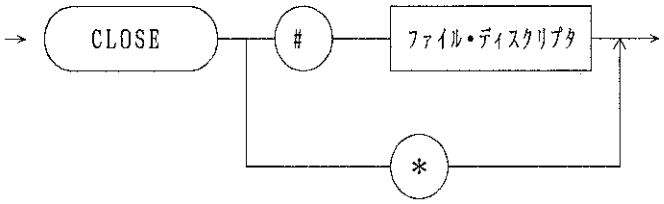
例
```
10  CLEAR
20  CLEAR 2
30  CLEAR 1, 3, 5, 7
```

注　意　ADDRESSABLE モードでは機能しません。

5.　CLOSE

概　要　ファイル・ディスクリプタに割り当てられているファイルをクローズします。

構　文　(1)-1



(1)-2
CLOSE ＜#ファイル・ディスクリプタ | ＊＞

解　説

・OPENコマンドでオープンしたファイルは、メモリ・カードを抜く前や、装置の電源をOFF する前に、必ずすべてのファイルをクローズしなければなりません。クローズしないとファイルは破壊されます。

・BASICプログラムでは、PAUSEやSTOPキーで停止させたときはファイルを自動的にはクローズしません。それ以外のときはプログラムの終了とともにすべてのファイルをクローズします。エラー終了時もクローズしますが、ON ERRORの設定がある場合は、クローズしません。

以上のような理由からエラー終了時には、以下の方法（コマンドですべてのファイルをクローズする指定方法）で明示的にクローズ動作を実行して下さい。

CLOSE *

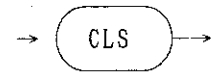
・ファイルは、SCRATCH やLOADなどを実行したときにも自動的にクローズします。

※　ファイルの取り扱いについては、「2.4 ファイルの管理」を参照して下さい。

6.　CLS

概　要　ディスプレイの表示をクリアします。

構　文　(1)-1



(1)-2
CLS

解　説
・ディスプレイに表示されているキャラクタをクリアすると同時に、カーソルをホーム・ポジションに戻します。

・CONSOLE によって指定されたスクロール範囲内のみクリアします。

例　10 CLS

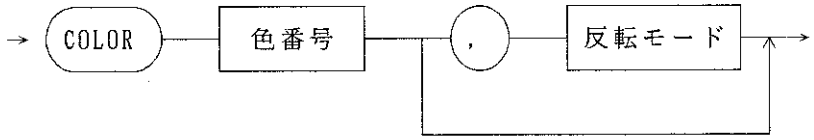
7.　COLOR

概　要

ディスプレイ（BASIC 画面選択時のみ）に出力する際の文字色を指定します。このコマンドにて指定された色、または反転モードは、改めて設定されるまで継続します。

構　文

(1)-1



(1)-2

COLOR　〈文字色〉[,　反転モード]

指定範囲

文字色　　：[0] 黒　[1]赤　[2] 緑　[3] 黄色
　　[4] 青　[5]紫　[6] 水色　[7] 白

反転モード：[0] 標準表示、[1] 反転表示

解　説

COLOR 3 →　文字色を黄色に指定します。

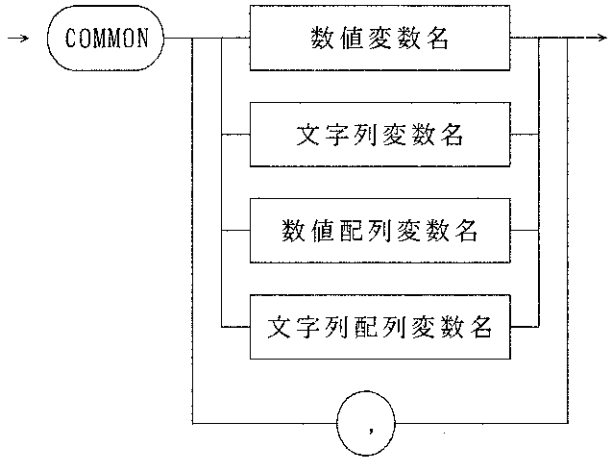
8.　COMMON

概　要

メモリ上のプログラムから、MERGE コマンドによりメモリ上のプログラムに合成されたプログラムに変数を引き渡します。COMMONは、宣言されたプログラム行以降で有効となります。

構　文

(1)-1



(1)-2

COMMON　数値変数 | 文字列変数 | 数値配列変数 | 文字配列変数[, ...]

解　説

COMMON A,B$,C(),D$()→数値変数A 、文字変数B$、数値配列変数C() 、文字列配列変数D$()の変数を合成プログラムに引き渡します。

注　意

・合成プログラム実行後、合成プログラム内で使用されたローカル変数値を、コマンド・ライン上から参照することはできません。

・数値配列変数、文字列配列変数をCOMMON宣言する場合、これ以前にDIM 、またはINTEGER ステートメントにて配列宣言をしておかなければなりません。宣言なしに使用した場合、エラー・メッセージが表示されます。

・COMMON宣言は、合成プログラム側でも宣言できますが、基本的にCOMMON宣言する変数は、すべてメモリ上に存在するプログラム側で宣言済み（使用済み）の変数にて行います。合成プログラム側では、COMMON宣言前後に出現した同一変数名の変数は、別変数扱いとなるので注意して下さい。

例）合成プログラム側

```
1000 A=100     ....ローカル変数
1010 PRINT A   ──→100.0
1020 COMMON A ....コモン変数（1000行目の変数A とは別変数扱い）
1030 PRINT A   ──→0.0
```

＊ 同じ変数名A でも、COMMON宣言前後で意味が変わります。

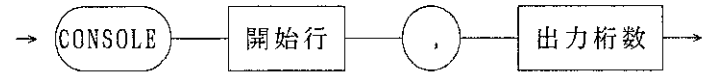
・合成プログラム側で配列変数をCOMMON宣言する場合、あらかじめメモリ上のプログラム側にてDIM 、またはINTEGER ステートメントで配列を宣言しておかなければなりません。宣言なしに使用した場合、エラー・メッセージが表示されます。

```
SELECT Value
CASE 1                  →  数値変数Value が1 の場合
  PRINT ''TYPE-1''
  Mode=100
CASE 2,3                →  数値変数Value が2 または3 の場合
  PRINT ''TYPE-2''
  Mode=200
CASE A*2+1              →  数値変数Value が （変数A*2+1)の場合
  PRINT ''TYPE-4''
  Mode=300
CASE ELSE               →  数値変数Value が上記CASE文以外の場合
  PRINT ''TYPE-OTHER''
  Mode=400
END SELECT
```

9.　CONSOLE

概　要　スクロール範囲を指定します。

構　文　(1)-1



(1)-2
CONSOLE　<開始行，出力行数>

指定範囲　開始行　：　0～26
出力行数：　1～27

解　説　CONSOLE 0，27 →最上行(0) から27行分を出力範囲とします（フル画面指定）
CONSOLE 5，10 → 6行目から10行分を出力範囲とします。

注　意
・CONSOLE コマンドにて指定されたディスプレイの出力範囲は、現在アクティブになっている画面に対してのみ有効です。つまり、波形画面、BASIC画面では、それぞれ別の出力範囲情報を持っています。

・基本的にCONSOLE にて指定した出力範囲外に出力されることはありません。CURSORにて出力範囲外が指定された場合は、直後の 1回のPRINT 等による出力に限り範囲外での出力が可能となります。なお、 1回の出力後は、再び範囲内出力に戻ります。

10.　COS

概　要

余弦値（コサイン）を返す組み込み関数です。数値表現式で指定する値の単位は、ラジアンで指定します。（4.3- 2. ATN 、68. SIN を参照。）

構　文

(1)-1

(1)-2

COS(数値表現式) →

例

```
PRINT COS(PI/2)
A$=COS(B)
```

11.　CSRLIN

概　要　現在のカーソル位置の位置座標y を返します。

構　文　(1)-1

(1)-2
CSRLIN→

指定範囲　y 座標位置：　0～26

解　説　PRINT CSRLIN　→現在のカーソル位置座標y をディスプレイに表示します。
Ypos=CSRLIN　→現在のカーソル位置座標y を数値変数Yposに代入します。

12.　CSRPOS

概　要　現在のカーソル位置の位置座標x を返します。

構　文　(1)-1

(1)-2
CSRPOS→

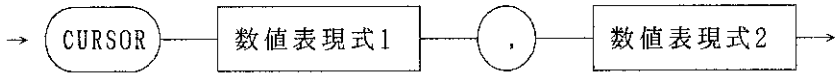
指定範囲　x 座標位置：　0～66

解　説　PRINT CSRPOS　→現在のカーソル位置座標x をディスプレイに表示します。
Xpos=CSRPOS　→現在のカーソル位置座標x を数値変数Xposに代入します。

13.　CURSOR

概　要　ディスプレイ上の指定された座標位置にカーソル（出力位置）を移動します。

構　文　(1)-1



(1)-2
CURSOR <x 座標位置, y 座標位置>

指定範囲
x 座標位置：　0～66
y 座標位置：　0～26

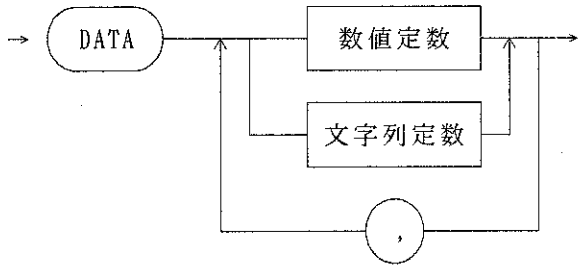
解　説
CURSOR 0, 0　→ディスプレイ左上座標にカーソルを移動します。
CURSOR 5,10　→ディスプレイのx 座標 6文字目、 y座標11行目にカーソルを移動します。

注　意　x, y座標位置ともに指定範囲を超える指定をした場合は、指定範囲内に修正されます。

14.　DATA

概　要　READ文で読み込むための数値、文字列を定義します。

構　文　(1)-1



(1)-2
DATA 〈数値定数 | 文字列定数〉 {, 〈数値定数 | 文字列定数〉 }

解　説
- DATA文は実行の対象とはならないため、どの文番号にあっても構いませんが、原則として、READ文で読み出す順序に従っている必要があります。
- READ文がプログラムの中からDATA文を捜して、対象となるデータを読み込むことになります。
- この順序を変更するには、RESTORE 文を使います。
- DATA文には、カンマ (,)かスペースで区切ることによって複数個の定数を指定できます。
  文字列は、文字列定数としてダブルクォーテーション (")で囲みます。
- DATAの後にコロン(:) によるマルチ・ステートメントは、使用できません。

注　意　DATA文に並べるパラメータは、変数を含む式ではいけません。

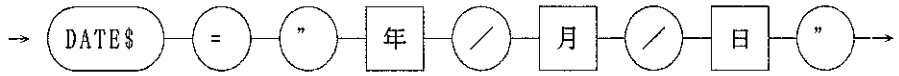
15.　DATE$

概　要　日付の読み出しと、日付の変更をします。

構　文　(1)-1

(1)-2
DATE$

(2)-1



(2)-2
DATE$=" 年／月／日"

解　説
・本器内蔵の時計（RTC)の日付を読み出します。

・読み出した日付は変更できます。
以下のように入力して下さい。

DATE$=" 1995／1／1"
または
DATE$=" 1995／01／01"

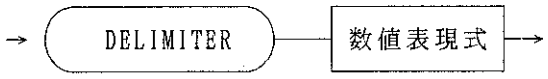
例

```
10 DIM D$[10]
20 D$=DATE$
30 PRINT "Date is ";D$
40 PRINT "Date Reset"
50 DATE$="1995/1/1"
60 STOP
```

16.　DELIMITER

概　要　4種類のデリミタを選択し、設定するステートメントです。

構　文　(1)-1



(1)-2
DELIMITER 数値表現式

解　説
・数値表現式によって示される番号に対応したデリミタを設定します。
デリミタの選択番号および種類を下表に示します。

選択番号	デリミタの種類
0	CR、LFの 2バイト・コードを出力 LF出力と同時に、単線信号EOI も出力
1	LFの 1バイト・コードを出力
2	データの最終バイトと同時に、単線信号EOI を出力
3	CR、LFの 2バイト・コードを出力

・数値表現式の結果が 0～3の範囲を越えた場合は、エラーとなります。
また、小数点以下の数値は無視し、整数として取り扱います。

・電源投入時は、DELIMITER=0 が自動的に設定されます。

例

```
10   DELIMITER 0
20   DELIMITER 1
30   DELIMITER A*10
```

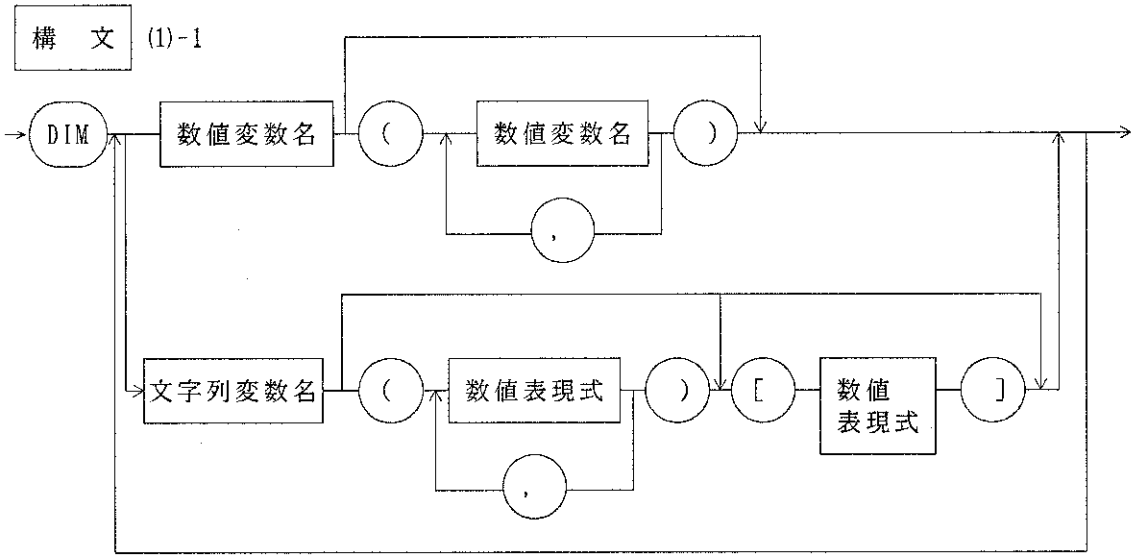
17. DIM

概　要

配列変数の要素の大きさを指定し、メモリ領域に割り当てます。数値配列変数は、実数型で10次元まで宣言できます。また、文字列配列変数も数値配列変数と同様に10次元まで指定できます。

構　文

(1)-1



(1)-2

```
DIM < 数値変数名 (数値表現式  [,  数値表現式, ...]) >
                    {, 数値変数名 (数値表現式[,  数値表現式])}
DIM < 文字変数名 (数値表現式  [,  数値表現式, ...]) 数値表現式]]>
      {, 文字変数名 (数値表現式[,  数値表現式, ...]) [[数値表現式]]}
DIM < 文字変数名 [数値表現式] >
                    {, 文字変数名 [数値表現式] }
```

解　説

```
DIM Buf(100)        →実数100 個分の領域をメモリに取得します。
DIM Buf(5,3)        →実数 3×5 個分の領域をメモリに取得します。
DIM Str$(5)         →18文字分の領域を 5個分、メモリに取得します。
DIM Str$(5,3)[10]   →10文字分の領域を 3×5 個分、メモリに取得します。
DIM Str$[128]       →128 文字分の領域をメモリに取得します。
```

注　意

・配列変数、または文字列変数を使用する場合、DIM で予め配列変数の要素の大きさを定義しなければなりません。ただし、数値配列変数の場合は宣言なしに使用すると、 1次元で10個の要素数の領域 (DIM A(10) と同様) を、文字列変数の場合は、18文字分の領域 (DIM A$[18]と同様) を取得します。
なお、文字配列変数は、宣言なしには使用できません。

・配列変数の要素の大きさを示す数値表現式に、実数変数を指定しても小数点以下を切り捨てた整数扱いとなります。

例

```
10 DIM N(5)
20 FOR I = 1 TO 5
30      N(I) = I*I/2
40 NEXT I
50 FOR I = 1 TO 5
60      PRINT N(I)
70 NEXT I
```

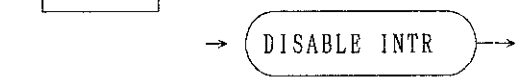
<実行結果>

```
 0.5
 2.0
 4.5
 8.0
 12.5
```

18.　DISABLE INTR

概　要　割り込みの受付を禁止します。

構　文　(1)-1



(1)-2
DISABLE INTR

解　説

・DISABLE INTRは、ENABLE INTR で許可された割り込みを禁止します。

・DISABLE INTRの実行後に、再び割り込みを許可する場合は、ENABLE INTR を実行します。このとき、ON XXXステートメントで設定された分岐の条件は、以前の状態を保っています。
ただし、割り込み分岐の条件を変更する場合は、ENABLE INTR を実行する前にON XXX、またはOFF XXX の各ステートメントを用いて設定して下さい。

・プログラムを実行した直後は、ENABLE INTR を実行するまで割り込みは禁止状態になっています。

例

```
10  ON KEY 1 GOTO 60
20  ENABLE INTR
30  ! LOOP
40  GOTO 30
50  !
60  DISABLE INTR
70  PRINT "KEY 1 INTERRUPT"
80  STOP
```

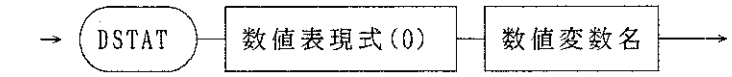
19. DSTAT

概　要

現在のカレント・ディレクトリの内容をBASIC 変数に取り込みます。カレント・ディレクトリ（カレント・ドライブ）を変更する場合は、CHDIR コマンドを使用して下さい。

構　文

(1)-1

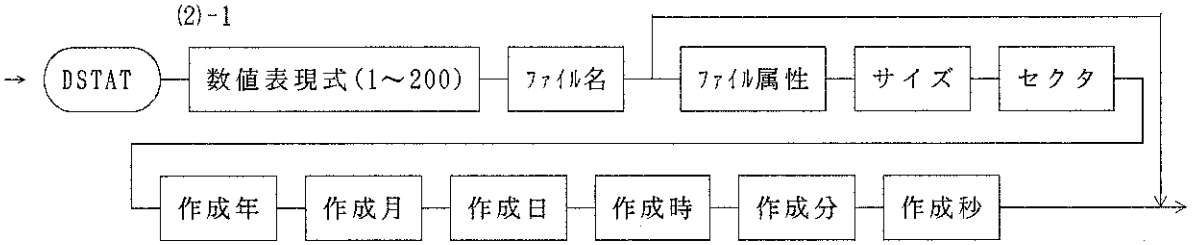


(1)-2

DSTAT < インデックス0 > 変数

インデックスが0 の場合は、現在のカレント・ディレクトリ内に記録されているファイルの数を調べて、その数を変数に代入します。

(2)-1

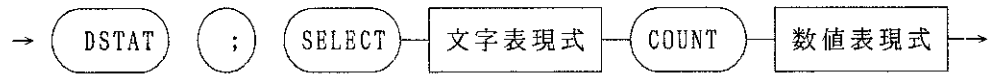


(2)-2

DSTAT < インデックス 1～200 > <ファイル名>
[,ファイル属性][, ファイルサイズ][, セクタ][, ファイル作成年]
[,月] [,日] [,時] [,分] [,秒]

インデックスが 1～200 の場合は、現在のカレント・ディレクトリ内に記録されている個々のファイル情報を各々の変数にそれぞれ代入します。ここでのインデックス番号とは、ディレクトリ内に登録されている登録番号を示します。
（CAT コマンドにて表示される順番と同一です。）

(3)-1



(3)-2

DSTAT ; SELECT <文字列> COUNT <変数>

< 文字列 >で指定された条件と一致するファイルの数を変数に代入します。
< 文字列 >内に指定できる特殊な表現文字を以下に示します。

? ：何らかの 1文字と一致する。
* ：何らかの 1文字以上の文字列と一致する。
[ ]：[ ]で囲まれた文字列のいずれか 1文字と一致する。[A-D]と指定するとA,B,C,D のいずれか 1文字と一致した場合という意味となります。

注）< 文字列 >内にドライブ名は、指定できません。

解　説

```
DSTAT 0 File_max            →ディレクトリ内のファイル数を変数
                              File_max に代入します。
DSTAT 3 File$, Atrb, Size   →ディレクトリ内の登録番号 3番目のファイ
                              ル名を変数File$ に、属性を変数Atrbに、
                              サイズを変数Sizeにそれぞれ代入します。
```

例

メディアに登録されているファイル数個分だけファイル情報を読み込み、表示します。

```
10 INTEGER I,Atr,Size,Sector,Year,Month,Day
20 DSTAT 0 Max_file
30 FOR I=1 RO Max_file
40   DSTAT I Name$,Atr,Size,Sector,Year,Month,Day
50   PRINT USING ''k,2D,k,15A,X,2D,X,6*,X,6*,X,4D,k,2Z,k,2Z'';''[',I,']''
                          ,As$,Atr,Size,Sector,Year,''/'',Month,''/'',Day
60 NEXT I
70 END
```

<実行結果>

```
 [ 1] AAA.BAS       32 **1598 *****3 1995/03/02
 [ 2] BBB.BAS       32 **6326 ****12 1994/11/25
 [ 3] ADVAN         48 **    0 *****1 1995/01/06
      .....                     ......
```

ファイル属性	1: READ ONLY　　8: FOLUME LABEL 2: HIDDEN FILE　16: DIRECTORY 3: SYSTEM FILE　32: ARCHIVE FILE
ファイル・サイズ	ファイル・サイズをバイト数で示します。
セクタ	セクタ数
年月日	ファイルの作成年月日
時間	ファイルの作成時刻

20.　ENABLE INTR

概　要　割り込みの受け付けを許可します。

構　文　(1)-1

(1)-2
ENABLE INTR

解　説

・ENABLE INTR は、割り込みの受け付けを許可し、ON XXXステートメントで定義された割り込み分岐を有効にします。

・DISABLE INTRの実行後に再び割り込みを許可する場合は、ENABLE INTR を実行します。

・プログラムを実行させた直後は、ENABLE INTR を実行するまで割り込みは禁止状態になっています。(DISABLE INTR 状態)

例

```
10  ON KEY 1 GOTO 60
20  ENABLE INTR
30  ! LOOP
40  GOTO 30
50  !
60  PRINT "KEY 1"
70  GOTO 20
```

注意

ENABLE INTR を実行しても、ON XXXで定義された割り込みが発生すると、プログラムが分岐した時点で割り込み禁止状態となります。(DISABLE INTR 実行と同等)
これは、割り込み処理中に次の割り込みが発生した場合、割り込み処理が入れ子(Nest)にならないようにしているためです。
よって、続けて割り込み分岐を有効にしたい場合は、再度ENABLE INTR により割り込みを許可する必要があります。

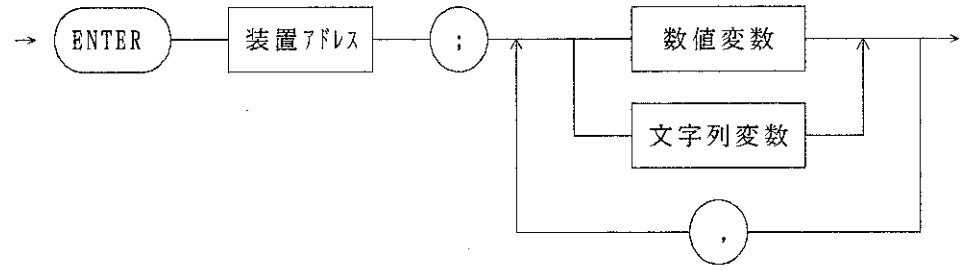
21.　ENTER

概　要

(1)GPIBからデータを取り込みます。
(2)ファイルからデータを読み込み、入力項目に代入します。

構　文

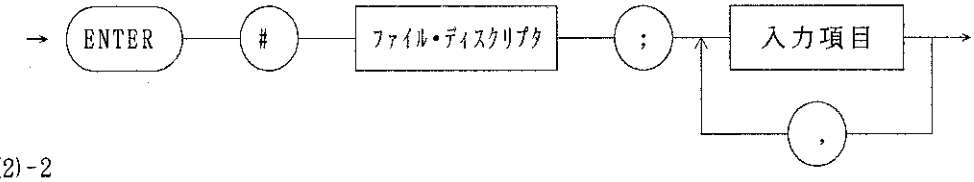
(1)-1



(1)-2
ENTER 装置アドレス;<数値変数 | 文字列変数> {,<数値変数 | 文字列変数>}

注）装置アドレス：0～30;　外部GPIB接続機器のアドレス
　　31　;　本器の測定部からのデータ入力

(2)-1



(2)-2
ENTER # ﾌｧｲﾙ･ﾃﾞｨｽｸﾘﾌﾟﾀ ;　入力項目 {, 入力項目}

解　説

(1)の構文

・装置アドレスによって指定された装置からGPIBを通してデータを入力し、数値または文字列としてBASIC の変数内に蓄えます。ただし、装置アドレスによって指定された装置にトーカ機能がない場合、コントローラはハンドシェークを完了できずに停まってしまうので、注意して下さい。
また、文字列変数を使用する場合は、あらかじめ DIM文によって文字列変数を宣言しておかなければなりません。

・文字列で入力するときは、デスティネーションに使用する文字列変数の長さが十分でないと、入力データがオーバーフローを起こし、文字列変数に入りきらないデータは無視されるので、注意して下さい。

・例
```
10 ENTER 1;A
20 DIM A$(100), B$(20)
30 ENTER 2;A$
40 ENTER 3;B$
```

・注意
SYSTEM CONTROLLER モード時は、指定アドレスの機器をトーカに指定し、データを取り込みます。

(2)の構文
・ファイル・ディスクリプタに割り当てられているファイルから、データを対応する入力項目のデータ・タイプ形式で読み込んで、その入力項目に代入します。

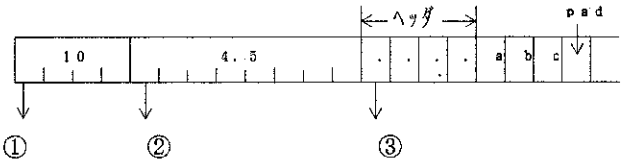
※　ファイルの取り扱いについては、「2.4 ファイルの管理」を参照して下さい。

・例1)　BINARYファイル

内部データをそのそのままの形で代入します。入力項目が整数のとき 4バイト、実数は 8バイト、文字列は 4バイトのヘッダをそれぞれ読み込んだ後、ヘッダの内容が示すバイト数のデータを読み込みます。
読み込むバイト数は入力項目の型で決まるので、OUTPUTのときと同じ型で入力しないと、データの内容が違ってしまいます。

```
10 INTEGER I
20 DIM R
30 OPEN "FILE" FOR INPUT AS #FD
40 ENTER #FD;I,R,S$
```

代入する変数のタイプによって読み込むバイト数が違ってきます。

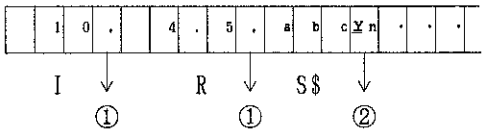


①：　変数が整数のときは、 4バイトのデータを読み込み、そのままのデータを変数に代入します。
②：　変数が実数のときは、 8バイトのデータを読み込み、そのままのデータを変数に代入します。
③：　変数が文字列のときは、ヘッダ 4バイトを読み込み、ヘッダが示す長さ分だけ読み込み、文字列変数に代入します。

・例2)　TEXTファイル

入力項目の数に関わらず、ライン・フィードまで読み込みます。カンマ(,)までが 1つのデータとなり、入力項目の型に変換して代入されます。入力項目の数が実際のデータより多いときは、多い分の変数には代入されません。
したがって、それらは前に格納されていた値が残ります。
逆に変数の数が実際のデータ数より少ないときは、データが捨てられます。

```
10 INTEGER I
20 DIM R
30 OPEN "FILE" FOR INPUT AS #FD;TEXT
40 ENTER #FD;I,R,S$
```



①：　各項目はカンマ(,)で区切られます。
②：　最後の項目の後にはライン・フィードがあります。

・例3)　ASCII ファイル

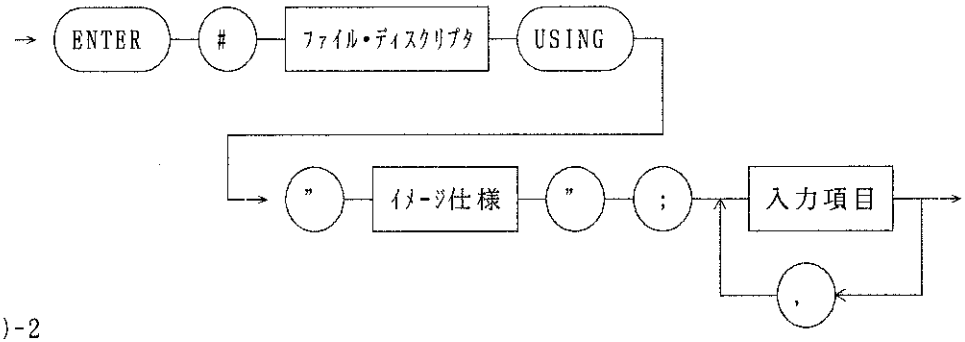
ヘッダ 2バイトを読み込み、ヘッダが示す長さのデータを読み込みます。変数に型にデータを変換し代入します。

```
10 INTEGER I
20 DIM R
30 OPEN "FILE" FOR INPUT #FD;ASCII
40 ENTER #FD;I,R,S$
```

22.　ENTER USING

概　要　ファイルからイメージ仕様のフォーマットで入力項目に入力します。

構　文　(1)-1



(1)-2

ENTER #ファイル・ディスクリプタ USING "イメージ仕様" ; 入力項目 {, 入力項目}

注）ENTERはENT 、USING はUSE と省略できます。

解　説　ファイル・ディスクリプタに割り当てられているファイルからイメージ仕様のフォーマットで入力項目にデータを入力します。

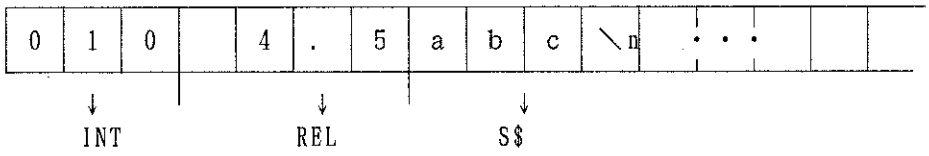
イメージ仕様

D	:	D の数を数値の桁数と解釈して数値を読み込み、入力項目の変数に代入する。
Z	:	D と同じ。
K	:	1 行読み込み、数値データに変換し、入力項目の変数に代入する。
S	:	D と同じ。
M	:	D と同じ。
.	:	D と同じ。
E	:	K と同じ。
H	:	K と同じ。ただし、小数点にカンマ(,) を使う。
*	:	D と同じ。
A	:	A の数分の文字を読み込み、文字列変数に代入する。
k	:	1 行読み込み文字列変数に代入する。
X	:	1 文字のデータを読み飛ばす。
リテラル	:	＼" で囲まれた文字列の数のデータを読み飛ばす。
B	:	1 文字読み込み、入力項目に ASCIIコードとして代入する。
@	:	1 バイトのデータを読み飛ばす。
+	:	@ と同じ。
-	:	@ と同じ。
#	:	ENTER では無視される。
n	:	数字で各イメージ仕様の繰り返し回数を指定できる。 3D.2D はDDD.DDと同じで、4AはAAAAと同じ。

※　ファイルの取り扱いについては、「2.4 ファイルの管理」を参照して下さい。

例

```
10 INTEGER INT
20 DIM     REL
30 ENTER #FD USING "ZZZ,DD.D,3A";INT,REL,S$
```

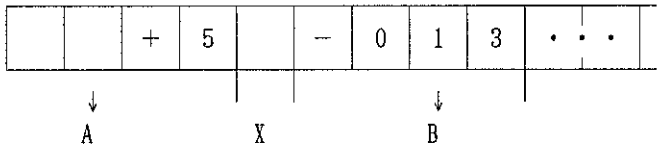


INT :　3 バイトのデータを読み込み、INT のデータタイプの整数型に変換し、INTに代入します。実行後の INTの値は10になります。

REL :　イメージ仕様の DD.D が入力項目の RELに対応します。
4 バイトのデータを読み込み、データを実数型に変換し RELに代入します。実行後の RELは 4.5になります。

S$:　3 バイトのデータを読み込み、S$に代入します。
実行後のS$は abcです。

```
10  DIM A,B
20  ENTER #FD USING "SDDD,X,MZZZ";A,B
```



A,B : 4 バイトのデータを読み込み、実数型に変換して A,Bに代入します。
実行結果、A=5.0 , B=-13.0
イメージ仕様のXは1バイトを読み込みますが、変数へ代入しません。
SDDDのフォーマットで入力するデータを実数型に変換して Aに代入します。
X は変数を必要とせず、1 文字分が読み飛ばされます。
MZZZZ が Bに対応し、4 バイトを入力し実数に変換して Bに代入されます。

```
10  DIM A
20  ENTER #FD USING "K";A
```

S	T	R	I	N	G	1	2	3	.	5	#	#	＼n	・・

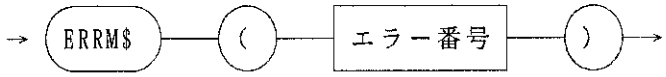
実行結果　　A=123.5

STRING123.5## を読み込んで入力変数A の実数型に変換します。入力項目が実数型の場合、先行する数値、符号(+, -)、指数のE,e以外の文字は無視され、数値のみを取り込みます。数値を検出してから、数値でない文字になったところで数値への変換をやめます。イメージ仕様の K, E, k, H は、ライン・フィードがターミネータになるので、現在のファイル・ポインタからライン・フィードまでを 1つのデータとして変数に代入します。

23.　ERRM$

概　要　指定する番号のエラーメッセージを返すシステム関数です。

構　文　(1)-1



(1)-2
ERRM$(エラー番号)

解　説

・パラメータで指定されたエラー・メッセージを返します。
特にパラメータとして0を指定すると、直前に表示されたエラー・メッセージを返します。

・エラー番号は、以下のような構造になっています。

エラークラス * 256 + エラー・メッセージ番号

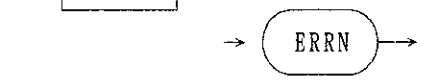
エラークラス : 1 ; データ入出力関係
2 ; データ演算処理関係
3 ; ビルトイン関数関係
4 ; BASIC 構文関係
5 ; その他

・エラークラスを含む番号を指定しても、内部ではエラーメッセージ番号だけを参照します。したがって、エラー番号にERRNを指定できます。

24.　ERRN

概　要　エラー番号を保持しているシステム変数です。

構　文　(1)-1



(1)-2
ERRN

解　説

・BASICプログラムの実行の際に発生したエラーの番号を保持するシステム変数です。

・BASIC プログラムの開始時に 0に初期化され、エラーが発生するとその値が代入されます。この値は、明示的に 0を代入するか、BASIC プログラムを再び実行させると 0に初期化します。

・エラー番号は、以下のような構造になっています。

エラークラス * 256 + エラー・メッセージ番号

エラークラス : 1 ;　データ入出力関係
2 ;　データ演算処理関係
3 ;　ビルトイン関数関係
4 ;　BASIC 構文関係
5 ;　その他

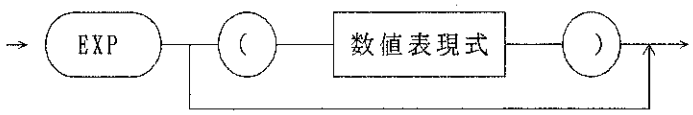
25.　EXP

概　要

定数e(自然対数の底）の引数乗を求め、その値を返す組み込み関数です。

構　文

(1)-1



(1)-2

EXP[(数値表現式)]→

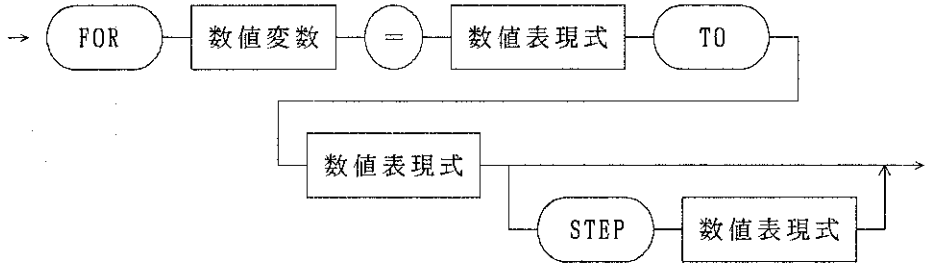
解　説

```
PRINT EXP      →EXP(1)と同じ値を返します。
PRINT EXP(2)
A=EXP(B)
```

26.　FOR - TO - STEP, NEXT, BREAK, CONTINUE

概　要　FOR 文とNEXT文の一対でプログラムのループ（繰り返し処理）を構成します。

構　文　(1)-1



(1)-2
FOR 数値変数＝数値表現式　TO　数値表現式〔STEP　数値表現式〕

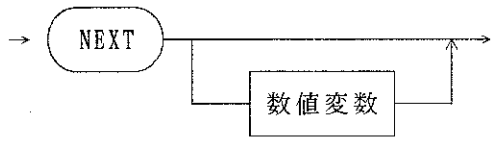
(2)-1

(2)-2
BREAK

(3)-1

(3)-2
CONTINUE

(4)-1



(4)-2
NEXT〔数値変数〕

解　説

・指定された数値変数をループ（繰り返し）のカウンタとして用い、初期値から最終値まで増加分ずつ変化させていきます。カウンタの値が最終値より大きくなったとき、ループは終了します。カウンタの増減はNEXT文で行います。したがって、FOR 文～NEXT文までの間に組まれたプログラムを繰り返し処理します。

・初期値、最終値、増加分は、以下のように指示します。
FOR A=(初期値)　　TO (最終値)　　STEP (増加分)

・STEP(増加分)を省略した場合、自動的に+1となります。

・FOR 文～NEXT文は入れ子(Nest)にできます。

・一対のFOR 文とNEXT文で使用するループ・カウンタの数値変数名は、同じにして下さい。数値変数名が異なっているとエラーになります。

・FOR 文～NEXT文で繰り返し処理をしているときに、ループ・カウンタに使っている数値変数の値を変えると、正常な繰り返し処理をしなくなりますので注意して下さい。

・NEXT文の後の数値変数を省略した場合、自動的に直前の FOR文と対応します。

・FOR-NEXTのループは、BREAK 文で抜け出すことができます。

・CONTINUE文では、FOR-NEXTのループ内で次のステップ値のループに分岐します。

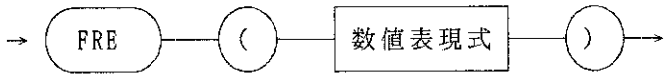
例

```
10 FOR R=11 TO 0 STEP -5
20   FOR I=0 TO PI STEP PI/180
30     X=SIN(I)*R+23
40     Y=COS(I)*R+15
50     CURSOR X,Y:PRINT "*"
60   NEXT I
70 NEXT R
80 STOP
```

27. FRE

概　要　BASICバッファのメモリ残量を返すシステム関数です。

構　文　(1)-1



(1)-2
FRE(数値)

解　説
・数値表現式に 0を指定するとBASIC が使用できるメモリのおおよその残りの容量をバイト単位で返します。また、 1を指定するとビルトイン関数用のメモリの残りの容量をバイト単位で返します。

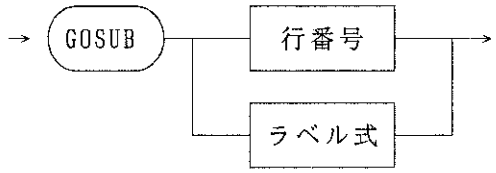
・これはおおよその容量を判断するためのもので、厳密にメモリ内の再構成はしません。したがって、一度セーブしてロードし直すと、より大きな容量を得ることがあります。

例　PRINT FRE(0)

28.　GOSUB, RETURN

概　要　指定されたサブルーチンへの分岐、復帰を行います。

構　文　(1)-1



(1)-2
GOSUB <行番号 | ラベル式>

(2)-1

(2)-2
RETURN

解　説

・行番号またはラベル式によって指示された行番号から始まるサブルーチンへ処理の制御を移し、RETURN文でGOSUB文の次の文へ戻ります。

・サブルーチンの最後には必ずRETURN文を入れて、処理をメイン・プログラムへ戻して下さい。

・サブルーチンの分岐をせずにRETURN文を実行するとエラーになります。

・GOSUB 文～RETURN文は入れ子(Nest)にできるので、サブルーチンの中から別のサブルーチンへ分岐できます。ただし、あまり入れ子を大きくするとメモリ容量がなくなり、エラーになることがあります。

・GOTO、GOSUB などでラベル式を書いたとき、その対象となるラベル式の行がない場合、または誤ってそのラベル式の行を削除した場合、GOTOやGOSUB のラベル式の値が 0になってしまいます。このままで実行しようとすると、以下のメッセージが表示されます。

Undefined line

このままでは実行できません。GOTO、GOSUB の行を正しいラベル式に直して下さい。

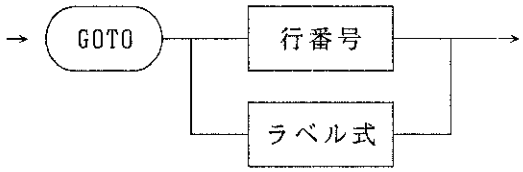
例

```
10   FOR I=1 TO 9
20     GOSUB 60
30     GOSUB *PRT
40   NEXT I
50   STOP
60   ! SUB ROUTINE
70   X = I * I
80   RETURN
90   *PRT ! SUB ROUTINE
100  PRINT I;"*";I;"=";X
110  RETURN
```

29.　GOTO

概　要　指定された行番号へ分岐します。

構　文　(1)-1



(1)-2
GOTO 〈行番号 | ラベル式〉

解　説　・指定された行番号またはラベル行へ無条件の分岐をします。

例

```
10  FOR I=1 TO 9
20    GOTO 60
30    GOTO *PRT
40  NEXT I
50  STOP
60  !
70  X = I * I
80  GOTO 30
90  *PRT
100 PRINT I;"*";I;"=";X
110 GOTO 40
```

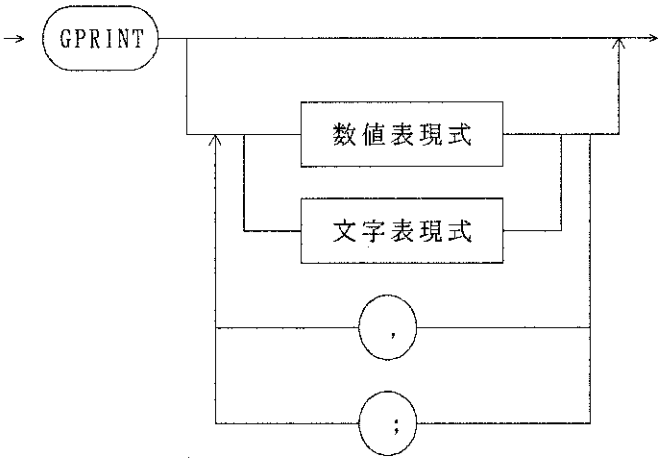
30.　GPRINT

概　要

数値または文字列を出力します。
GPRINT :　GPIB 出力

構　文

(1)-1



(1)-2
GPRINT〔<数値表現式｜文字表現式>｛,｜;<数値表現式｜文字表現式>｝

解　説

・GPRINTで指定された数値、文字列を表示します。

・数値、文字列をカンマ(,)で区切って複数を指定すると、改行せずに数値、文字列を次々に出力します。

・GPRINTの最後に、セミコロン(;)を置いた場合は、プリント出力が終っても改行されません。
したがって、次のGPRINTを実行すると、以前にプリントした行に続いてプリントします。

<進んだ使い方>
本器がADDRESSABLE モード（CONTROL 7;0)時、外部コントローラ（パソコン等）からBASIC で使用している変数の内容を読み込むことができます。

例）変数Aa, Bb$ の内容を読み込みたい場合（本器GPIBアドレスは8 とします。）

現在変数Aa=123.456、Bb$=''ADVANTEST'' となっているとします。
外部コントローラから以下のコマンドを実行します。(NEC-PC98 N88BASICの場合)

例1
```
PRINT @8;''@GPRINT AA''
INPUT @8;ENT1
PRINT EXT1              .....実行結果：123.456
```

例2
```
PRINT @8;''@GPRINT AA,BB$''
INPUT @8;ENT2$
PRINT EXT2$             .....実行結果：123.456 ADVANTEST
```

注）GPRINTの後ろに複数個の変数を指定した（例2)場合、外部コントローラ側で複数個の変数にて読み込みを行っても、 2番目以降の変数には期待値は入りません。これは、GPRINTはGPIBに 1つの文字列データとして出力しているためです。

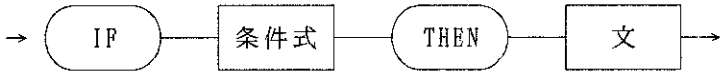
例
```
100 PRINTER 1
110 FOR I=0 TO 20
120   GPRINT I
130 NEXT I
140 STOP
```

31.　IF-THEN, ELSE, END IF

概　要　条件判断による分岐、指定された文の実行をします。

構　文

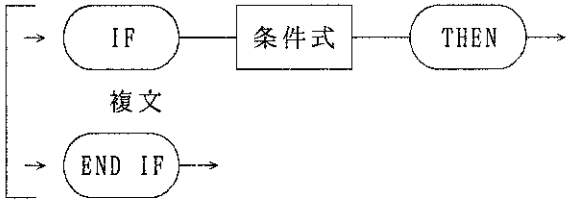
(1)-1



(1)-2

```
IF　条件式　THEN　文
```

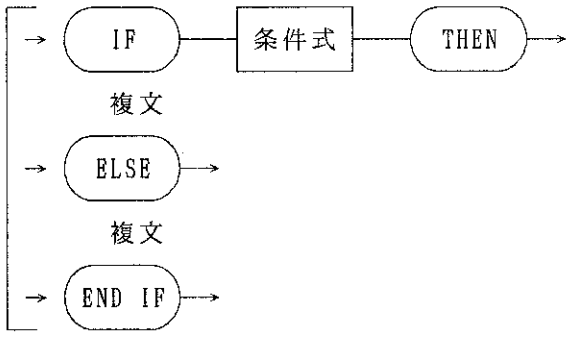
(2)-1



(2)-2

```
IF　条件式　THEN
  複文
END IF
```

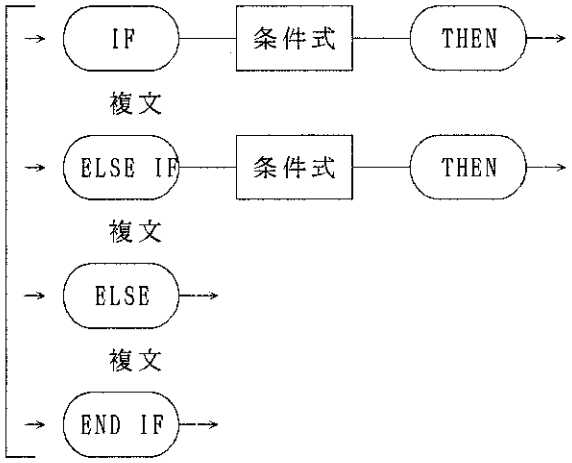
(3)-1



(3)-2

```
IF　条件式　THEN
  複文
ELSE
  複文
END IF
```

(4)-1



(4)-2

```
IF  条件式  THEN
  複文
ELSE IF 条件式  THEN
  複文
ELSE
  複文
END IF
```

解　説

・条件式は論理式ですが、ここには比較演算子を用いた論理式以外に数値表現式を書くこともできます。この場合、演算結果が 0になったときのみ(false)とし、それ以外の値は全て真(true)と解釈します。

・論理式の条件によってプログラムの分岐、処理などを行います。

・論理式の関係が成立すると、THEN文を実行します。THEN文には文を続けることができ、次文を実行します。

・論理式の関係が不成立の場合は、そのまま次の行に進みます。

・比較演算子には、以下の 6種類があります。

A=B	AとBが等しいとき成立
A>B	AがBより大きいとき成立
A<B	AがBより小さいとき成立
A>=B	AがBと等しいか大きいとき成立
A<=B	AがBと等しいか小さいとき成立
A<>B	AとBが等しくないとき成立

上の論理式でA, Bはともに数値表現式で構成できます。
ただし、数値表現式と文字列表現式を比較することもできます。

例

```
10 FLG = 0
20 FOR I=0 TO 10
30  PRINT I;
40  IF (I % 2) =0 THEN FLG = 1
50  IF FLG = 1 THEN
60                    PRINT "  EVEN";
70                    FLG = 0
80                    END IF
90  PRINT
100 NEXT I
110 STOP
```

32.　INKEY$

概　要　直前に押したパネル・キーのコードを返します。なお、一度参照するとこの変数の内容はクリアされます。

構　文　(1)-1

(1)-2

INKEY$→

解　説

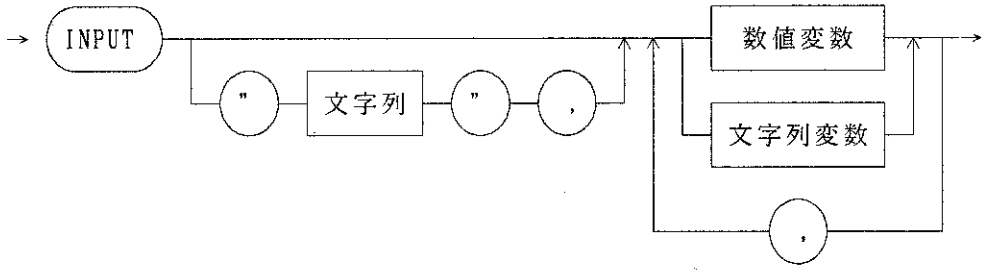
PRINT INKEY$　→直前に入力されたキー・コードを読み出します。
（結果 :A）

Key$=INKEY$　→直前に入力されたキー・コードを読み出します。

33.　INPUT（INP）

概　要　キー入力されたデータを変数に代入します。

構　文　(1)-1



(1)-2
INPUT〔“文字列”,〕<数値変数｜文字列変数>{, 数値変数｜文字列変数}

解　説

```
INPUT A                    →数値変数A にキー入力データを代入します。
INPUT A,B,C                →数値変数A,B,C にキー入力データを代入します。
INPUT "Please input",A$    →'Please input'をディスプレイに表示し、文字列変数A$にキー入力データを代入します。
```

注）INPUT を実行するとプログラムはパネル、またはキーボードからの入力待ちは、データ入力後にENTER キーが押されるまで続きます。

注　意

・変数を複数個指定した場合には、入力するデータもコンマ（,）で区切って指定した変数の数だけ入力しなければなりません。

・数値変数にて入力待ちしているときに数字以外の文字（英文字、記号等）が入力された場合、数字以外の文字は無視します。もし数字が 1文字もない場合には、数値変数には 0が代入されます。

・文字列変数にて入力待ちしている場合で、文字定数を入力するときは、ダブル・クォテーション(") で文字列を囲む必要はありません。

・数字、または文字入力が 1文字もなく、ENTER キーだけが押される場合は、変数への代入は行われません。つまり、INPUT ステートメント実行前の値がそのまま保持されています。

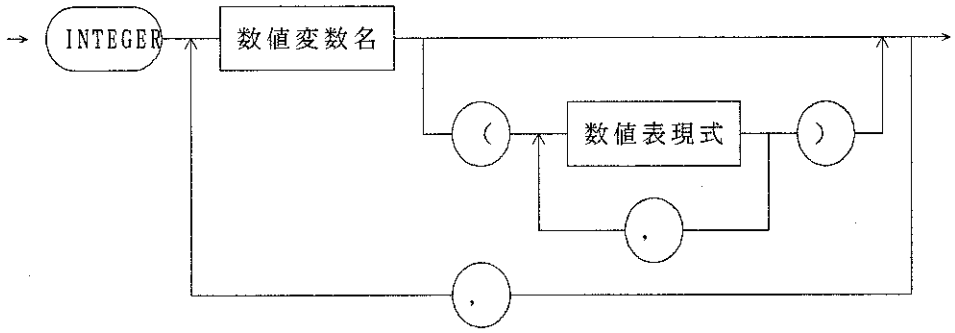
例

```
 10  OUTPUT 31;"IP"
 20  INPUT  "CENTER FREQUENCY(MHz)?",Cf
 30  INPUT  "SPAN FREQUENCY(KHz)?",Sf
 40  OUTPUT 31;"CF",Cf,"MZ"
 50  OUTPUT 31;"SP",Sf,"KZ"
 60  OUTPUT 31;"TS"
 70  OUTPUT 31;"PS"
 80  OUTPUT 31;"ML?"
 90  ENTER 31;Mkr$
100  PRINT  "LEVEL =",Mkr$
110  STOP
```

34.　INTEGER

概　要　変数または配列変数が整数型であることを宣言します。

構　文　(1)-1



(1)-2

INTEGER　A〔B〕 {, A〔B〕}

A : 数値変数名
B : (数値表現式 {, 数値表現式})

解　説

・INTEGER 文で、数値変数、配列変数を指定すると、以後その変数は、整数型となります。

・整数型変数で扱える数値は、整定数で扱える範囲と同じです。
-2147483648～+2147483647

・整数しか扱わない変数では、INTEGER 文で宣言した方が処理時間が短くなります。

・INTEGER 命令で配列宣言すると、指定された大きさの配列変数をメモリ上に確保します。したがって、大きすぎる配列宣言をするとメモリ領域が足りなくなり、エラーとしてプログラムの実行を中止します。
(Memory space full)

・添字を複数個指定すると個数分の次元を持つ配列変数の指定となります。
(指定可能な次元数は、最大10次元まで)

例

```
10  INTEGER ARRAY(2,3)
20  PRINT "J/I";
30  PRINT USING "X,3D,3D,3D";1,2,3
40  PRINT " ";
50  FOR I = 1 TO 2
60    FOR J = 1 TO 3
70        ARRAY(I,J) = I*10 + J
80    NEXT J
90  NEXT I
100 FOR I = 1 TO 2
110 PRINT
120 PRINT USING "2D,2X,#";I
130   FOR J = 1 TO 3
140       PRINT USING "3D,#";ARRAY(I,J)
150   NEXT J
160 NEXT I
```

<実行結果>

```
J/I   1  2  3

 1   11 12 13
 2   21 22 23
```

注意

1. INTEGER 文で一度整数型に指定された変数はDEL やコメント文で命令を削除しても、整数型のままです。

2. 実数型に再指定したい場合は、DIM 命令を追加するか、SAVE/LOAD を一度行ってからRUN させます。（数値変数の場合は、REAL命令を実行することにより実数型に再指定されます。）

35.　INTERFACE CLEAR

概　要　本器に接続されているすべてのGPIBインタフェースを初期化します。

構　文　(1)-1

(1)-2
INTERFACE CLEAR

解　説　・INTERFACE CLEAR を実行すると、GPIBの単線信号IFC を約100μs の間出力します。
本器のGPIBに接続されている装置のすべてのGPIBインタフェースは、IFC信号を受け取ると、トーカまたはリスナの状態が解除されます。

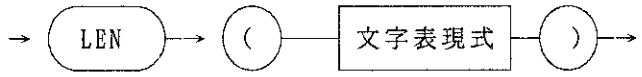
例　10 INTERFACE CLEAR

注　意　ADDRESSABLE モードでは機能しません。

36.　LEN

概　要　文字列の文字桁数を返す組み込み関数です。

構　文　(1)-1



(1)-2
LEN (文字列表現式) →

解　説

```
PRINT LEN("ABCDE")
A=LEN(Str$)
```

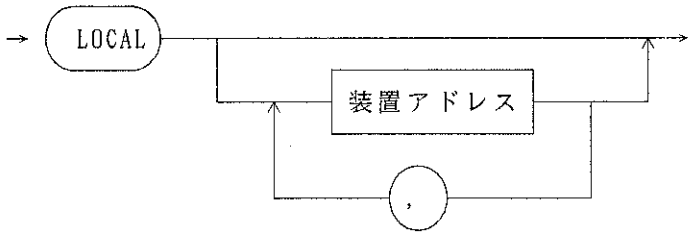
37. LOCAL

概　要

指定した装置をリモート状態から解除するか、またはリモート・イネーブル（REN)ラインを偽にします。

構　文

(1)-1



(1)-2

LOCAL 〔装置アドレス｛，装置アドレス｝〕

解　説

・装置アドレスを指定せずにLOCAL だけを実行した場合、GPIBリモート・イネーブル（REN)ラインが偽（High level）となり、GPIB上のすべての装置がローカル状態となります。
RENが偽のときは、OUTPUT命令でのGPIB機器の設定はできなくなる(GPIBでコントロールできない）ので注意が必要です。

・再び RENを真（Low level)にするためには、REMOTEを実行して下さい。

・LOCAL に続いて装置アドレスを指定した場合、装置アドレスで指定された装置のみをアドレスし、リモート状態を解除します。

例

```
10 LOCAL
20 LOCAL 1
30 LOCAL 1, 2, 3
```

注　意

ADDRESSABLE モードでは機能しません。

38. LOCAL LOCKOUT

概　要

GPIBに接続されている装置を、パネル面からローカル状態にする機能を禁止します。

構　文

(1)-1

(1)-2
LOCAL LOCKOUT

解　説

・GPIB上の各装置がリモート状態（GPIBによってリモート・コントロールされている）のときは、各装置のパネル・キーは LOCALキーを除きロックされ、パネルからデータ設定ができません。

リモート状態のときLOCAL キーを押すと、各装置は自分自身をローカル状態にするので、データ設定が可能な状態になります。このため、リモート制御中に種々の障害が生じ、正確なコントロールができなくなります。

この場合にLOCAL LOCKOUT を実行すると、GPIB上の全装置のローカル・キーをロックして、完全に各装置のパネル面からの設定を禁止します。

・LOCAL LOCKOUT を実行すると、GPIBにユニバーサル・コマンドのローカル・ロックアウト (LLO)を送ります。

・ローカル・ロックアウト状態を解除するには、LOCAL コマンドを用いてRENラインを偽(High level)にして下さい。

例

```
10 LOCAL LOCKOUT
```

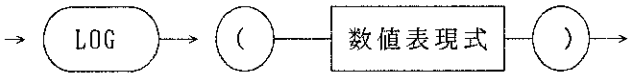
注　意

ADDRESSABLE モードでは機能しません。

39.　LOG

概　要　自然対数値を返す組み込み関数です。

構　文　(1)-1



(1)-2
LOG (数値表現式) →

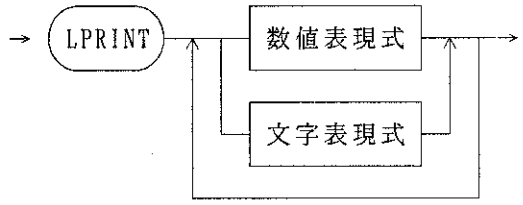
解　説

```
PRINT LOG(10)
A=LOG(B)
```

40.　LPRINT

概　要　数値、または文字列等をパラレル・ポート経由にて出力します。

構　文　(1)-1



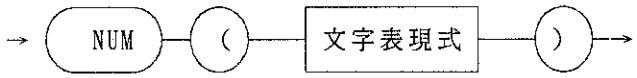
(1)-2
LPRINT　数値表現式｜文字列表現式 {, ｜; 数値表現式｜文字列表現式}

解　説
LPRINT"ADVAN" →パラレル・ポート経由にて文字列ADVAN を出力します。
LPRINT 123.456→パラレル・ポート経由にて123.456 を出力します。
LPRINT A,B,C$ →パラレル・ポート経由にて変数A,B,C$の内容を出力します。

41.　NUM

概　要　指定した文字のアスキ・コードを返す組み込み関数です。

構　文　(1)-1



(1)-2
NUM(文字表現式) →

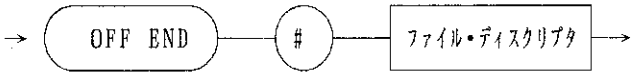
例

```
PRINT NUM("a")
A=NUM(B$)
```

42.　OFF END

概　要　ON END文で指定したエンド・オブ・ファイル時の処理を解除します。

構　文　(1)-1



(1)-2
OFF END　#ファイル・ディスクリプタ

解　説　・ファイル・ディスクリプタに定義してあった分岐先を解除した後に、エンド・オブ・ファイルが起こった場合、以下のエラー・メッセージを表示して終了します。

End of "ファイル名" file

※　ファイルの取り扱いについては、「2.4 ファイル管理」を参照して下さい。

43.　OFF ERROR

概　要　エラーが発生したときの分岐の機能、定義を解除します。

構　文　(1)-1

(1)-2

```
OFF ERROR
```

解　説　・ON ERRORステートメントによって定義されたエラー分岐を禁止します。

例

```
10  ON ERROR GOTO 100
      :
100  OFF ERROR
110  PRINT "Error Code",ERRN
120  STOP
```

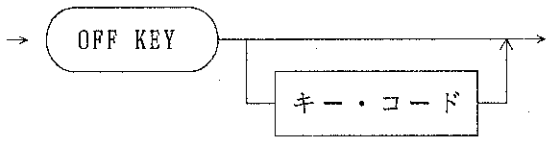
44.　OFF KEY

概　要

パネルのソフト・キー(1～7)によるキー割り込み分岐を禁止します。このコマンド文を実行するとディスプレイのソフト・キー・メニュー上のボタン表示が消去されます。全ソフト・キーの割り込み分岐が禁止されるとソフト・キー・メニュー自体も消去されます。

構　文

(1)-1



(1)-2

OFF KEY 〔キー・コード〕

解　説

・ON KEYステートメントによって許可された本器の KEY入力割り込みによる分岐を禁止します。

例

```
10   ON KEY 2 GOTO 100
20   ENABLE INTR
30   ! LOOP
40   GOTO 30
100  OFF KEY
110  PRINT "OFF KEY"
120  STOP
```

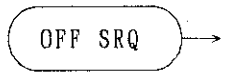
45. OFF SRQ, OFF ISRQ

概　要

SRQ またはISRQの割り込みによる分岐の機能、定義を解除します。
(OFF SRQはコントローラ・モード時のみ有効)

構　文

(1)-1



(1)-2
OFF SRQ

(2)
OFF ISRQはOFF SRQ と同様です。

解　説

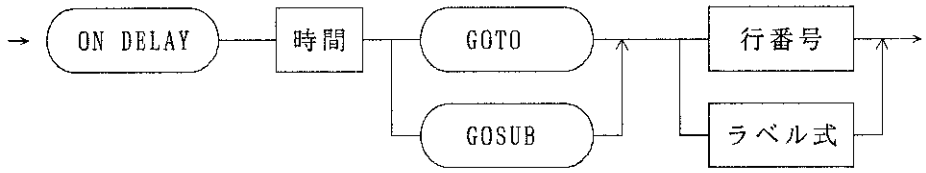
・OFF SRQ
ON SRQによって許可された割り込みによる分岐を禁止します。

・OFF ISRQ
ON ISRQ によって許可された割り込みによる分岐を禁止します。

46.　ON DELAY

概　要　指定時間経過後に分岐します。

構　文　(1)-1



(1)-2
ON DELAY　時間　<GOTO | GOSUB> <行番号 | ラベル式>

注）時間の単位はmsecで、設定範囲は 0～65535 です。

解　説
・指定時間経過後、その後のステートメントに従って分岐します。
・ENABLE INTR で割り込みの受け付けを許可しておく必要があります。

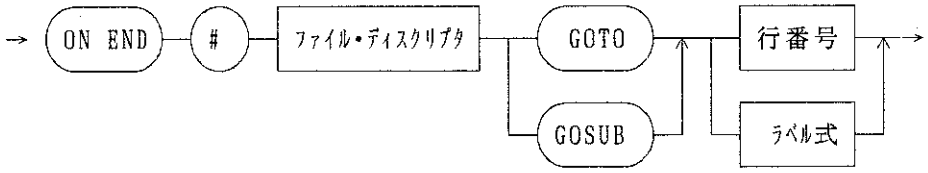
例

```
10  INTEGER T
20  T=50
30  ENABLE INTR
40  ON DELAY T GOSUB *TEST
50  STOP
100 *TEST
110   PRINT T;"[msec] Delay"
120 RETURN
```

47.　ON END

概　要　エンド・オブ・ファイル時の処理（分岐先）を定義します。

構　文　(1)-1



(1)-2
ON END #ﾌｧｲﾙ･ﾃﾞｨｽｸﾘﾌﾟﾀ <GOTO | GOSUB> <行番号 | ラベル式>

解　説
・ENTER でファイルからデータを読み込みますが、ファイルの終わりまで読み込んで入力するデータがない場合に、エンド・オブ・ファイルになります。
ON END文で処理を宣言しておかないと、ファイルをクローズした後に、エラー・メッセージを表示して実行を停止します。

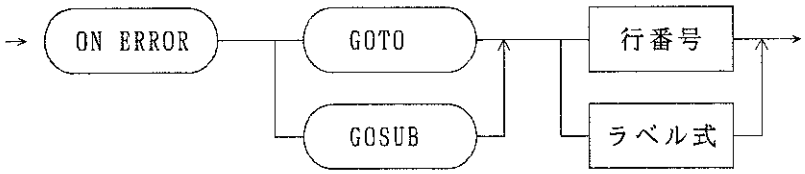
・分岐先を、数値変数、数値定数またはラベルで指定します。

※　ファイルの取り扱いについては、「2.4 ファイルの管理」を参照して下さい。

48.　ON ERROR

概　要　エラーが発生したときの分岐を許可します。

構　文　(1)-1



(1)-2
ON ERROR <GOTO | GOSUB> <行番号 | ラベル式>

解　説

・BASICプログラムの実行中にエラーが発生すると、そこの文番号とエラー・メッセージを表示してプログラムを停止します。
特に、計測器のサービスを要求するビルトイン関数のエラーの際には、エラー・メッセージを表示するだけで実行し続けます。これらを検出して分岐する場合には、ON ERROR文を使用します。

・分岐先は、数値定数、数値変数またはラベルで指定します。
発生したエラーを分類するために、エラー番号を記憶したERRNシステム変数が用意されています。

・エラーが発生した後に、そのエラー処理で確実に回復できないと永久ループになってしまいます。これを防ぐには、OFF ERROR文を入れます。

例　ON ERROR GOTO 1000

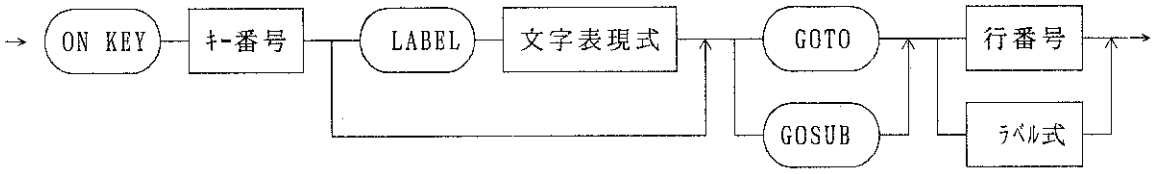
49. ON KEY

概　要

パネルのソフト・キー(1～7)によるキー割り込み分岐を許可します。
このコマンド文にてディスプレイのソフト・キー・メニュー上に任意のラベル文字をそれぞれソフト・キーごとに表示することができます。

構　文

(1)-1



(1)-2

ON KEY <ｷｰ番号> [LABEL文字列表現式]<GOTO | GOSUB><行番号 | ラベル式>

解　説

・キー番号指定範囲： 1～7

```
ON KEY 1 GOTO 100    →ソフト・キー1 を押すと行番号100 に処理が移ります。
ON KEY 2 GOTO *Lb1   →ソフト・キー2 を押すとラベル*Lb1行に処理が移ります。
ON KEY 7 LABEL "Soft-key|7"GOSUB *Sub
                     →ソフト・キー7 を押すと*Sub関数がサブルーチン・コールされます。
```

注　意

・ENABLE INTR ステートメントにて、割り込み受け付けを許可しておかない機能しません。

・LABEL に続く文字列表現式は、10文字× 3行分の指定ができます。なお、この文字列中にパイプ（|）を指定すると、そこでボタン内の表示が改行されます。
（パイプは、文字列中に 2個まで表現できます。 3個以上のパイプが表現された場合は、それ以降の文字が切り捨てられます。

例　ソフト・キーの 1～3 に処理を割り付けます。

```
10      ON KEY 1 GOTO *key1
20      ON KEY 2 LABEL ''Softkey-2'' GOSUB *key2
30      ON KEY 3 LABEL ''Softkey | key2'' GOSUB *key3
40      ENABLE INTR
50      GOTO 50
60      !
70      *Key1
80      PRINT ''Pressed'';inkey$
90      GOTO 40
100     *Key2
110      PRINT ''Softkey-2 pressed!!''
120     RETURN
130     *Key3
140     STOP
```

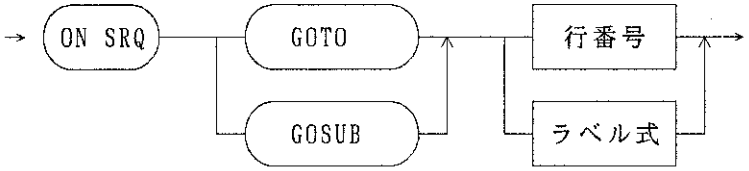
50.　ON SRQ, ON ISRQ

概　要

ON SRQは、GPIB外部SRQ 信号による割り込み分岐を許可します。
（ON SRQコントローラ・モード時のみ有効）
ON ISRQ は、内部割り込み要因が発生したときの割り込み分岐を許可します。

構　文

(1)-1



(1)-2
ON SRQ <GOTO | GOSUB> <行番号 | ラベル式>

(2)
ON ISRQ はON SRQと同様です。

解　説

・プログラム実行中の割り込みで分岐します。

・分岐は割り込みが発生したときに実行していたステートメントの処理が終了してから行われます。

・サブルーチンへ分岐した場合の戻り先は、割り込みが発生したときに実行していたステートメントの次のステートメントになります。

・ON SRQは、コントローラ・モードで実行時のみGPIB外部からのSRQ 信号で割り込み分岐します。

・ENABLE INTR で割り込みの受け付けを許可しておく必要があります。

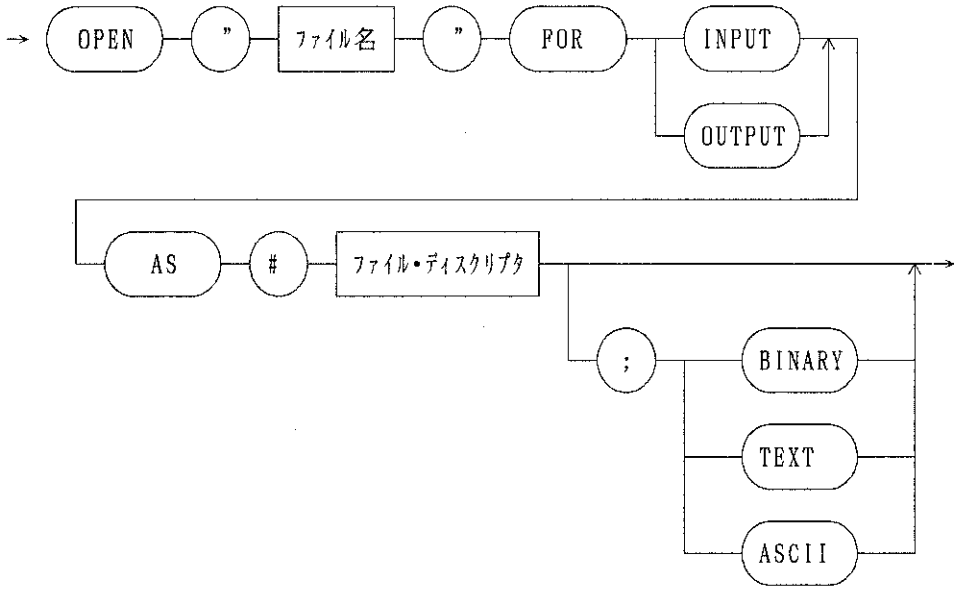
51. OPEN

概　要

ファイルに対しファイル・ディスクリプタを割り当て、指定した処理モードでオープンします。

構　文

(1)-1



(1)-2

OPEN "ファイル名" FOR 処理モード AS #ファイル・ディスクリプタ 〔; ファイル・タイプ〕

注）処理モード : INPUT | OUTPUT
　　ファイル・タイプ : BINARY | TEXT | ASCII

解　説

・ファイルをプログラムに認識させるために、ファイルに対してファイル・ディスクリプタを割り当て、指定した処理モードでオープンします。

処理モード
処理モードには、OUTPUTとINPUT があります。
OUTPUT :　ファイルにデータを書き込むときに使います。
INPUT　:　ファイルからデータを読み込むときに使います。

#ファイル・ディスクリプタ
実際のファイルに対する読み書きは、ENTER またはOUTPUTを使いますが、これらのコマンドに対して、対象となるファイルを認識させるために、ファイル・ディスクリプタを使います。
ファイル・ディスクリプタ名は #の後に英数字で記述します。

ファイル・タイプ
ファイル・タイプには、BINARY、TEXT、ASCII の 3種類あります。
ファイル・タイプを指定しないとBINARYになります。

BINARY ： データを内部の表現のまま記録します。整数のときは 4バイト、実数のときは 8バイト、文字列はヘッダ 4バイトの後にASCIIデータが続きます。データ文字数が奇数の場合はデータの後に 1 バイトのスペースをとります。

TEXT ： データをそのまま ASCIIコードに変換して出力しますが、数値の前に－かスペースをとります。
TEXTファイルでは USING指定ができます。

ASCII ： 入力、出力項目を 2バイトのヘッダの後に ASCIIで表現します。数値の前に－かスペースをとります。データ文字数が奇数の場合はデータの後に 1バイトのスペースをとります。

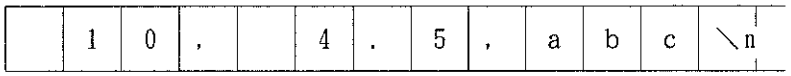
・既に他のファイルに割り当てられているファイル・ディスクリプタをオープンすると、前に割り当てられていたファイルをクローズして、指定されたファイルを新しくオープンします。

・同じファイルを同時点で複数のファイル・ディスクリプタでオープンすることはできません。

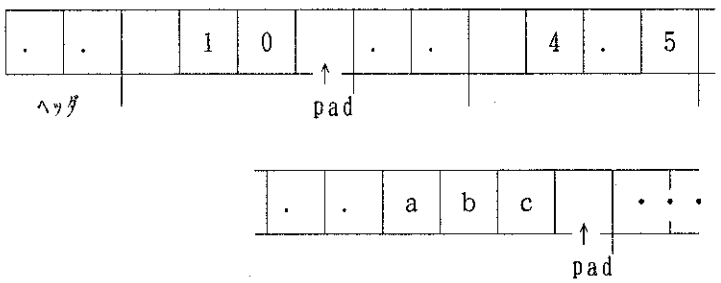
※ ファイルの取り扱いについては、「2.4 ファイルの管理」を参照して下さい。

例

```
10 OPEN "DATA.BAS" FOR OUTPUT AS #FD ; TEXT
20 OUTPUT #FD;10,4.5,"abc"
```



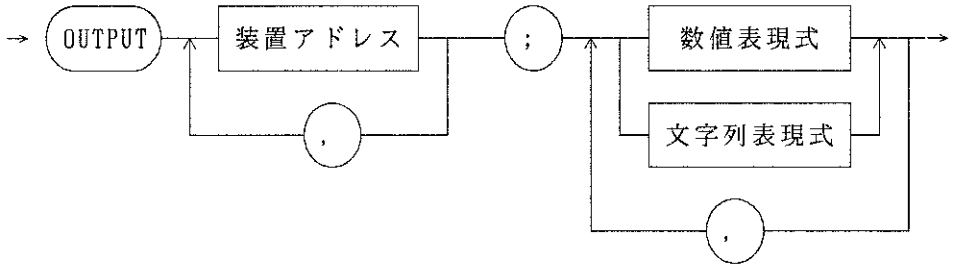

```
10 OPEN "DATA.BAS" FOR OUTPUT AS #FD ; ASCII
20 OUTPUT #FD;10,4.5,"abc"
```



52.　OUTPUT

概　要　(1)GPIBへデータを送出します。
(2)ファイルにデータを出力（書き込み）します。

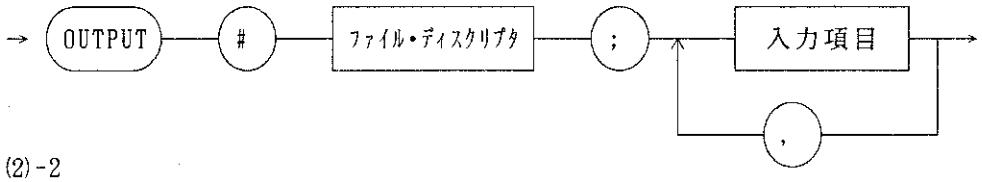
構　文　(1)-1



(1)-2
OUTPUT　装置アドレス｛，装置アドレス｝；<数値表現式｜文字列表現式>
｛，<数値表現式｜文字列表現式>｝

注）装置アドレス：0～30；　外部GPIB接続機器のアドレス
31　；　本器の測定部への出力

(2)-1



(2)-2
OUTPUT # ﾌｧｲﾙ･ﾃﾞｨｽｸﾘﾌﾟﾀ ; 入力項目｛,入力項目｝

解　説　(1)の構文

・装置アドレスによって指定された装置へ、数値および文字列をASCII データとして送ります。
装置アドレスは、カンマ（，）で区切って複数を指定できます。
また、数値表現式と文字列表現式もカンマで区切ると、混在して使えます。

・REN ラインが真（Low level)のときにOUTPUTステートメントを実行すると、装置アドレスで指定された装置は、自動的にリモート状態になります。リモート状態をプログラムで解除するときは、LOCAL ステートメントを実行して下さい。

・例
```
10 A=5
20 B=10
30 OUTPUT A;"STARTF",B,"MHz"
```

・注意
SYSYEM CFONTROLLERモード時は、指定アドレスの機器をリスナに指定し、データを出力します。

外部に指定したリスナがない場合、このコマンドは実行しません。

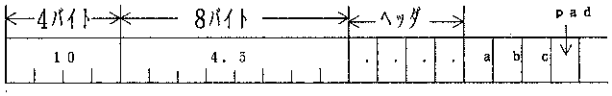
⑵の構文
・ファイル・ディスクリプタに割り当てられているファイルに、出力項目をBASIC の標準の書式に変換してから出力します。
このデータタイプの形式で読み込んで、その入力項目に代入します。

・例1)　BINARYファイル

データを内部表現と同じ型で出力します。文字列は 4バイトの文字列の長さを示すヘッダを付けて出力します。文字列が奇数の長さである場合は、最後に 1文字分のスペースをとります。

```
10 OPEN "FILE" FOR OUTPUT AS #FD
20 OUTPUT #FD;10,4.5,"abc"
```

※　ファイルの取り扱いについては、「2.4 ファイルの管理」を参照して下さい。

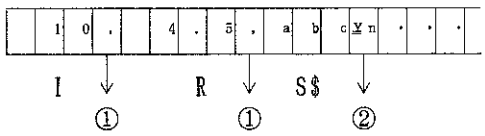


ヘッダはデータの長さを持っています。

・例2)　TEXTファイル

データをASCII コードに変換して出力します。
数値データには、スペースかマイナスの符号が頭に付きます。

```
10 OPEN "FILE" FOR  OUTUT AS #FD;TEXT
20 OUTPUT #FD;10,4.5,"abc"
```

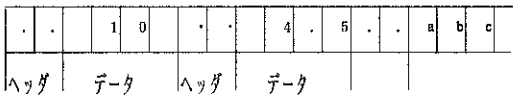


①：　各項目はカンマ(,) で区切られます。
②：　最後の項目の後にはライン・フィードが出力されます。

・例3)　ASCII ファイル

データをASCII コードに変換して出力します。
数値データには、スペースかマイナスの符号が頭に付きます。データのバイト数が奇数の場合は、最後にスペースが入ります。

```
10 OPEN "FILE" FOR INPUT #FD;ASCII
20 OUTPUT #FD;10,4.5,"abc"
```



ヘッダはデータの長さを持ちます。

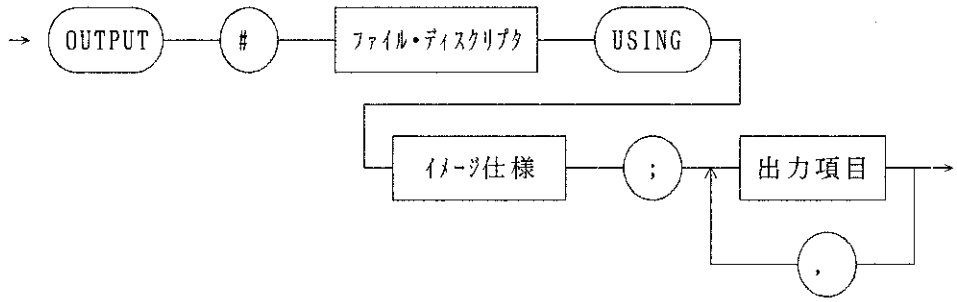
53.　OUTPUT USING

概　要

#ファイル・ディスクリプタに割り当てられているファイルにデータを指定された形式で出力（書き込み）します。

構　文

(1)-1



(1)-2

OUTPUT　#ﾌｧｲﾙ･ﾃﾞｨｽｸﾘﾌﾟﾀ　USING　イメージ仕様；出力項目 {, 出力項目}

注）OUTPUTは OUT、USING はUSE と省略できます。

解　説

・USINGとイメージ仕様を指定すると、自由に書式を変換して出力します。
イメージ仕様は、文字列式で指定します。

・ファイル・ディスクリプタは、フィイル・オープン時に指定したものを使います。オープン時に、処理の対象になるファイルに対して、ファイル・ディスクリプタを割り当てます。以後、そのファイルに対する処理はすべてこのファイル・ディスクリプタを介して行います。

イメージ仕様

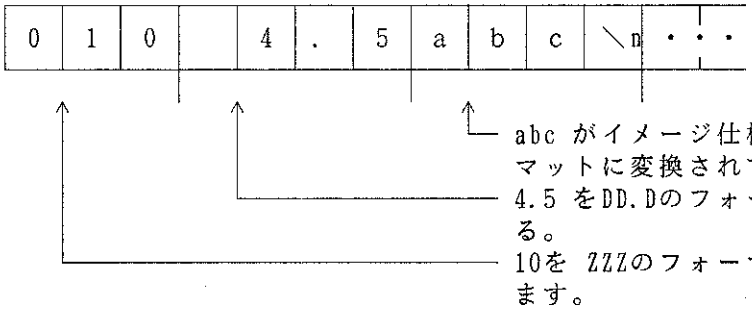
D	：	D の数で数値を出力するときの桁数を指定する。 指定したフィールドで空いた部分にはスペースが入る。
Z	：	Z の数で数値を出力するときの桁数を指定する。 指定したフィールドで空いた部分には 0が入る。
K	：	式の値を BASICの標準形式（PRINTと同じ）で出力する。
S	：	S の位置にプラス(+) かマイナス(-) を出力する。
M	：	M の位置に、負のときはマイナス(-) を、正ならばスペースを出力する。
.	：	. の位置に小数点がくるように位置を合わせる。
E	：	e 符号　指数　という書式で出力する。
H	：	K と同じ。ただし、小数点にカンマ (,)を使う。
R	：	. と同じ。ただし、小数点にカンマ (,)を使う。
*	：	* の数で数値の出力時の桁数を指定する。 指定したフィールドで空いた部分には *を出力する。
A	：	A の部分に 1文字出力する。
k	：	文字列式の値をそのまま出力する。
ﾘﾃﾗﾙ	：	＼”で囲まれた文字列を出力項目とは無関係にそのまま出力する。

X　：X の位置に 1つスペースをとる。
B　：式の値を ASCIIコードとして出力する。
@　：フォーム・フィードを出力する。
+　：キャリージ・リターンを出力する。
-　：ライン・フィードを出力する。
#　：最後の項目の後ろには、自動的にライン・フィードが付くが、このイメージ仕様を指定するとライン・フィードが付かない。
n　：数字で各イメージ仕様の繰り返し指定の回数を指定できる。
3D. 2DはDDD.DDと同じで、4AはAAAAと同じ。

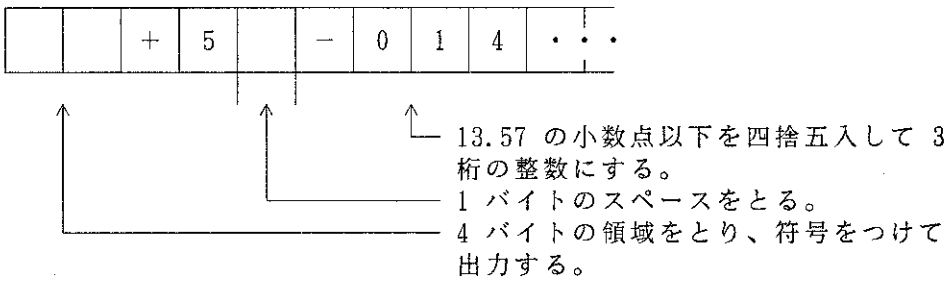
※ ファイルの取り扱いについては、「2.4 ファイルの管理」を参照して下さい。

例

```
OUTPUT #FD USING "ZZZ,DD.D,3A";10,4.5,"abc"
```




```
OUTPUT #FD USING "SDDD,X,MZZZ";+5,-13.57
```



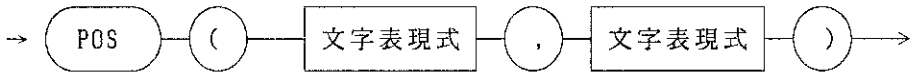
54. POS

概　要

文字列の中から指定された文字を探索し、その検出位置（何文字目か）を返す組み込み関数です。

構　文

(1)-1



(1)-2

POS(文字表現式，文字表現式) →

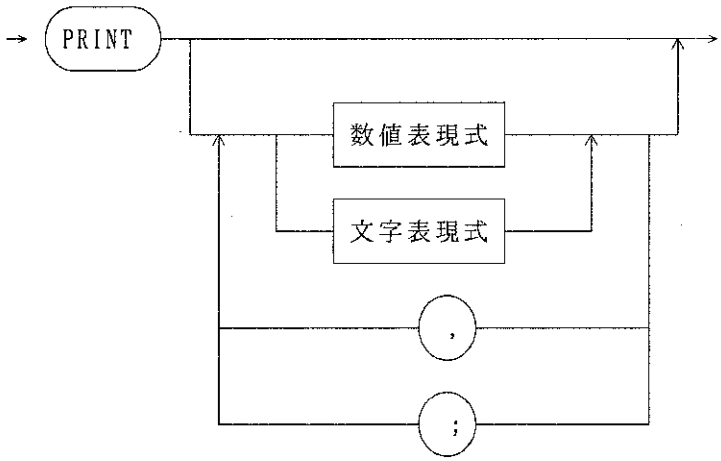
解　説

PRINT POS(''ADVANTEST'',''VA'') → 3文字目に'VA'が検出されたので 3
A=POS(A$, B$)　　を返します。

55. PRINT [USING]

概　要　数値または文字列を表示します。

構　文　(1)-1



(1)-2
PRINT〔数値表現式｜文字表現式 {, | ; 数値表現式｜文字表現式} 〕

解　説
- 指定された数値、文字列を表示します。
- 数値、文字列をカンマ(,) で区切って複数を指定すると、改行せずに数値、文字列を次々に出力します。
- PRINT の最後にセミコロン(;) を置いた場合は、プリント出力が終っても改行されません。したがって、次のPRINT を実行すると、以前にプリントした行に続いてプリントします。

例

```
10 PRINT 123*456
20 PRINT "ABC"
30 PRINT "Freq.=",A,"Hz"
40 PRINT I,
```

●PRINT USING 書式指定式 ;〔〔式〔・・・・〕〕

書式指定式は文字列表現式で、イメージ仕様をコンマで区切って、書式を指定します。最後は自動的に改行します。

ｲﾒｰｼﾞ仕様
D　：　指定フィールドの余った部分にスペースを表示する。
Z　：　指定フィールドの余った部分に 0を表示する。
K　：　式の値をそのまま表示する。
S　：　常に＋または－のサイン・フラグを付ける。
M　：　－のサイン・フラグを付けるか、正のときはスペースを取る。
.　：　小数点を表示する。
E　：　指数形式（e，符号，指数）で表示する。
H　：　式の値をそのまま表示する。ただし、小数点にカンマ(,) を使う。
R　：　ヨーロッパ・タイプの小数点を表示する（小数点にカンマ(,) を使う）。
*　：　指定フィールドの余った部分に *を表示する。
A　：　1 文字を表示する。
k　：　式の文字列をそのまま表示する。
X　：　スペースを表示する。
ﾘﾃﾗﾙ：　書式指定式にリテラルを書くときは＼" で囲む。
B　：　式の値をASCII コードとして表示する。
@　：　改ページする。（フォーム・フィード）
+　：　表示の位置を同じ行の先頭に移動させる。（キャリッジ・リターン）
-　：　表示の位置を次の行に移動させる。（ライン・フィールド）
#　：　最後に改行されない。
n　：　数字で各イメージ仕様の繰り返し回数を指定できる。
　　3D.2D はDDD.DDと同じ、4AはAAAAと同じ。

例 1

```
10  PRINT USING "4Z,2X,5D,2X,5*";123,-444,567
```

<実行結果>
```
0123    -444  **567
```

例 2

```
10  PRINT USING "S3D,X,S3D";-4.5,465
20  PRINT USING "M3Z.Z,X,M3ZR3Z";1.26,-5.452
```

<実行結果>
```
 -5 +465
001.3 -005,452
```

例 3

```
10  PRINT USING "K,X,H";5.03884e+22,4.5563
```

<実行結果>
```
5.03884e+22 4,5563
```

例 4

```
10  PRINT USING "k,#";"character:"
20  PRINT USING "B";69
```

<実行結果>
```
character:E
```

例 5

```
10  PRINT USING "\.....\",+,A";"*"
20  PRINT USING "k,-,\".END.\"";"string"
```

<実行結果>

```
*.....
string
.END.
```

例 6

```
100  PRINT USING "DDD.DD";1.2
110  PRINT USING "ZZZ.ZZ";1.2
120  PRINT USING "K";1.2
130  PRINT USING "SDDD.DD";1.2
140  PRINT USING "MDDD.DD";1.2
150  PRINT USING "MDDD.DD";-1.2
160  PRINT USING "H";1.2
170  PRINT USING "DDDRDD";1.2
180  PRINT USING "***.**";1.2
190  PRINT USING "A";"a"
200  PRINT USING "k";"string"
210  PRINT USING "B";42
220  PRINT USING "3D.2D";1.2
```

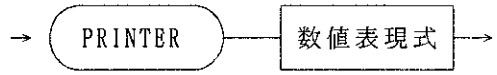
<実行結果>

```
  1.20
001.20
1.2
  +1.20
   1.20
  -1.20
1,2
  1,20
**1.20
a
string
*
  1.20
```

56.　PRINTER

概　要　プリンタに送る装置アドレスを指定します。

構　文　(1)-1



(1)-2
PRINTER 数値表現式

解　説
- GPIBに接続されるプリンタの装置アドレスを設定します。
- PRINT を実行する前に、必ずPRINTER でプリンタの装置アドレスを本器に指示して下さい。
- 装置アドレスは、0 ～30までの整数です。

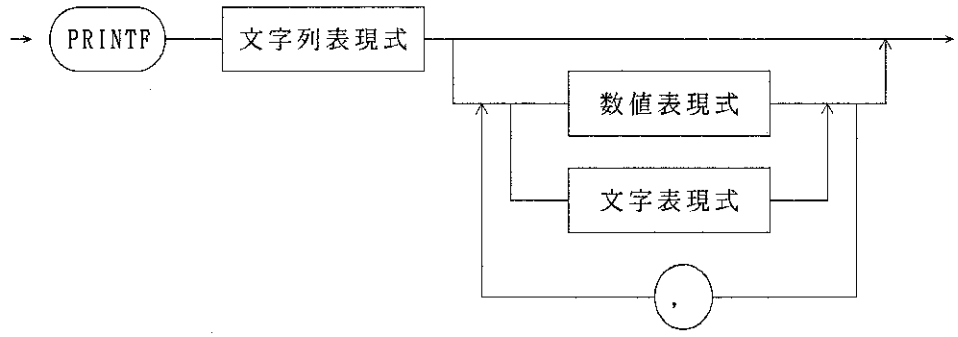
例

```
10 PRINTER 1
```

57. PRINTF

概　要　数値または文字列を表示します。

構　文　(1)-1



(1)-2
PRINTF　文字列表現式〔数値表現式｜文字表現式
{, 数値表現式｜文字表現式}〕

解　説

・指定された数値、文字列を表示します。

・数値、文字列をカンマ(,)で区切って複数を指定すると、改行せずに数値、文字列を次々に出力します。改行する場合は＼n を書式指定式の中で指定します。

・第 1パラメータの文字列表現式が、その後のパラメータの書式を指定するために使われます。

・書式指定の方法は以下の通りです。

●PRINTF　書式指定式〔〔式〔式〔･･･〕〕〕

書式指定の方法はC言語のPrintf関数に似ています。
書式指定式は文字列型であって、% に続けて以下の方法で出力の書式を指定します。この書式以外の文字列は単純に出力されます。% を出力したい場合は、%%と続けます。

% 〔-〕〔0〕〔m〕〔. n〕文字
- ……指定されたフィールド内で左詰めにする。この指定がなければ右詰めにする。
0 ……指定フィールドの余った部分に詰める文字を、スペースでなく 0にする。
m ……m 文字分のフィールドを取る。
. n ……n 桁の精度で出力する。文字列に対して指定すると、この値が実際の文字列の長さになる。
文字……d ; 符号付10進数　　s ; 文字列
o ; 8 進数　　e ; 浮動小数点表現（指数形式）
x ; 16進数　　f ; 浮動小数点表現

例

```
10 N = 500000
20 U = LOG(1+1/N)
30 V = U - 1 / N
40 PRINTF "%7d %12.5e %16.8e＼n",N,U,V
50 PRINTF "%s＼n","end"
```

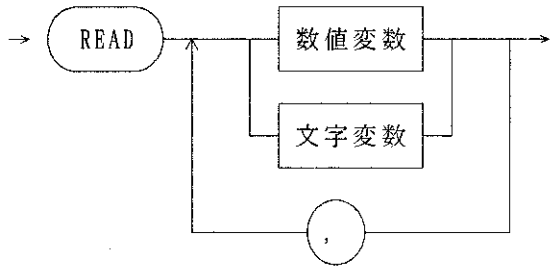
<実行結果>

```
 500000 2.00000e-06 -1.99993982e-12
end
```

58. READ

概　要　DATA文の定数を、変数に代入します。

構　文　(1)-1



(1)-2
READ 〈数値変数 | 文字変数〉 {, 数値変数 | 文字変数}

解　説

・DATA文で定義されている数値、文字列を、引数で指定してある変数に読み込みます。

・READ文が現れたところで、プログラムの中からDATA文を捜します。

・最初のREADでは、原則として（RESTORE 文で変更されていなければ）、プログラムの先頭行から行番号順に捜して、最初に発見した値を引き数並びの変数に代入します。
その後、順に対応するDATA文の定数を捜して代入します。

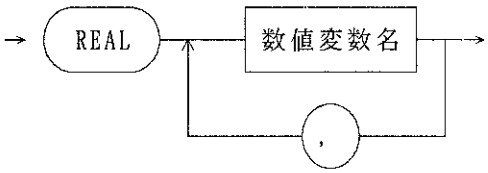
・READの変数に対して、DATAで指定する定数の数の方が少ない場合には、エラーとなります。

・対象は、READで読み出そうとする変数の数と、それに対するDATA文の定数の数で、DATA文やREAD文の行数は関係ありません。

59.　REAL

概　要　数値変数が実数型であることを宣言します。

構　文　(1)-1



(1)-2
REAL 〈数値変数〉 {, 数値変数}

解　説　REAL A,B　→数値変数A とB が実数型であることを宣言します。

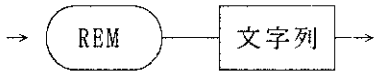
注）REAL省略時は、使用変数は実数型となります。

注　意　・INTEGER 文にて、整数型に宣言した変数は、REAL文にて明示的に実数型に型変更しない限り、整数型を保持します。

60.　REM

概　要　プログラムの注釈です。

構　文　(1)-1



(1)-2
REM　文字列

解　説
・プログラム中に注釈をつけたいときに使用します。

・REM は非実行ステートメントですから、REM に続く文字列はいかなるものでも構いません。すべての文字、数字、記号が使用できます。

・REM は、感嘆符(!) で代用できます。

・REM の後にコロン(:) によるマルチ・ステートメントは使用できません。すべて注釈文として見なされます。

例

```
10 REM "PROGRAM 1"
20 ! 1983-JUN-02
30 A=A+1:! INCREMENT A
```

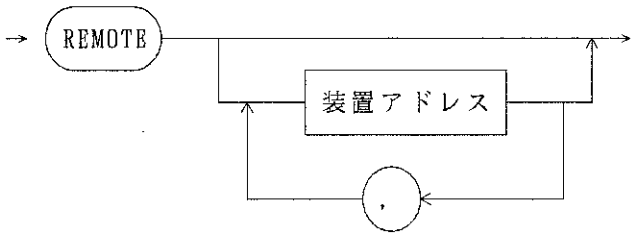
61. REMOTE

概　要

指定した装置をリモート状態にするか、またはリモート・イネーブル(REN)ラインを真にします。

構　文

(1)-1



(1)-2

REMOTE〔装置アドレス｛,装置アドレス　｝〕

解　説

・装置アドレスを指定せずにREMOTEだけを実行した場合、GPIBのリモート・イネーブル（REN)ラインが真（Low level)となり、GPIB上に接続された装置をリモート・コントロール可能な状態にします。
REN ラインを偽（Highlevel)にするためには、LOCAL を実行して下さい。

・REMOTEに続いて装置アドレスを指定した場合、装置アドレスで指定された装置のみをリモート状態にします（ただし、RENラインが真のときのみ）。
装置アドレスは複数を指定できます。
またリモート状態を解除するためには、LOCAL を実行して下さい。

・REMOTEは、選択した装置をリモート状態にするものですが、以下に示すステートメントを実行したときは、REMOTEを実行しなくても自動的に指定した装置をリモート状態にします（ただし、REN ラインが真のときのみ）。

CLEAR 〔装置ｱﾄﾞﾚｽ｛ , 装置ｱﾄﾞﾚｽ｝ 〕
OUTPUT 装置ｱﾄﾞﾚｽ｛ , 装置ｱﾄﾞﾚｽ｝ ; < 出力データ> { , <出力データ> }
REMOTE〔装置ｱﾄﾞﾚｽ｛ , 装置ｱﾄﾞﾚｽ｝ 〕
SEND LISTEN 装置ｱﾄﾞﾚｽ｛ , 装置ｱﾄﾞﾚｽ｝
TRIGGER 装置ｱﾄﾞﾚｽ｛ , 装置ｱﾄﾞﾚｽ｝

例

```
10 REMOTE 1
20 REMOTE 5
30 REMOTE 1  2  3
```

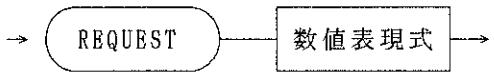
注　意

ADDRESSABLE モードでは機能しません。

62.　REQUEST

概　要　ADDRESSABLE モード時に外部GPIBコントローラへ送信するステータス・バイトを設定します。

構　文　(1)-1



(1)-2
REQUEST 数値表現式

注）整数の設定範囲は 0～255 です。

解　説
・ADDRESSABLE モード時に外部GPIBコントローラへ送信するステータス・バイトを設定します。

・サービス・リクエストを発信する場合は、64～127 または 192～255(ビット6 が1)の値を設定します。

例　10 REQUEST 65

注　意
・SYSTEM CONTROLLER モードでは機能しません。

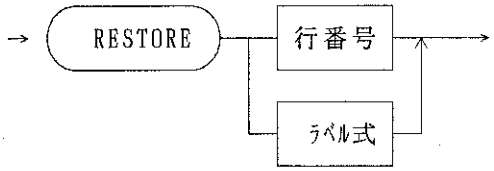
・外部コントローラからはシリアルポールを用いて読んで下さい。GPIBコマンドの*STB? では読めません。

・GPIBコマンドのSRQDが実行されている場合は、ステータス・バイトのビット6 は常に 0で送信されます。
したがって、サービス・リクエストも発信されません。

63.　RESTORE

概　要　次のREAD文で読み込むDATA行を指定します。

構　文　(1)-1



(1)-2

RESTORE　〈行番号｜ラベル式〉

解　説

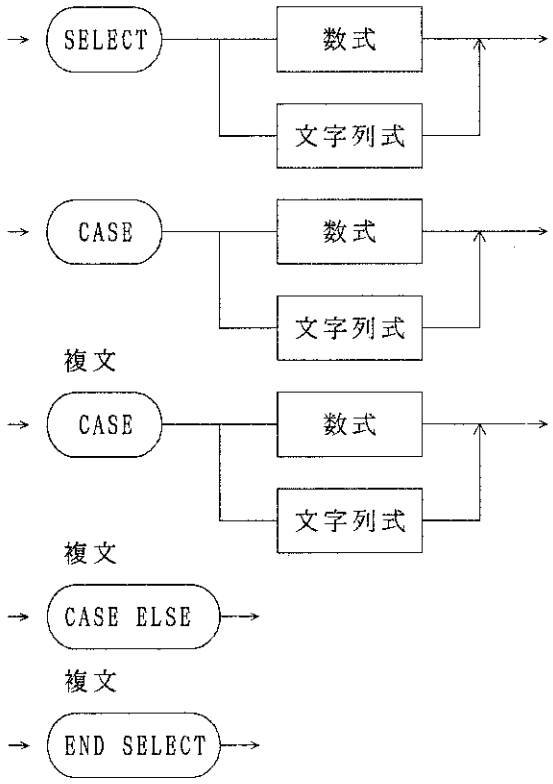
・行番号は、数式またはラベルで指定します。
特に指定がなければ、プログラムの先頭行から順番にDATA文の定数が読み込まれ、RESTORE で次のREADの対象となるDATA文を指定できます。

・引数の行番号がDATA文を捜し始める先頭行という判断をするので、その行以降の最初のDATA文が指定するものであれば構いません。

64.　SELECT, CASE, END SELECT

概　要　1つの式の値を条件として、複数の分岐を行います。

構　文　(1)-1



(1)-2

```
SELECT <数式 | 文字列式>
CASE   <数式 | 文字列式>
  複文
CASE   <数式 | 文字列式>
  複文
CASE ELSE
  複文
END SELECT
```

解　説

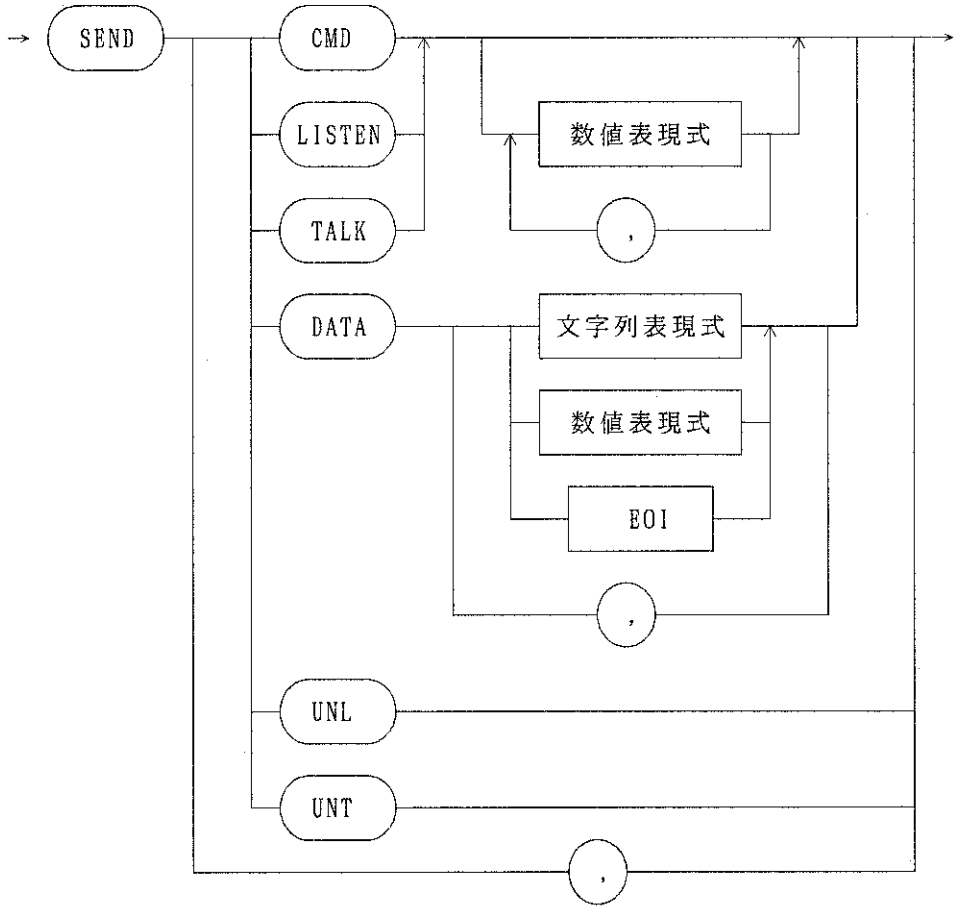
・SELECTで指定した式の値に、一致するCASE以下の文（複文）を実行します。実行の対象は、次のCASE、CASE ELSE、またはEND SELECTまでです。

・SELECT構文自体の入れ子ができます。この場合内部のSELECTは、完全に外部のものを含む形になります。

65.　SEND

概　要　GPIBにコマンドおよびデータを出力します。

構　文　(1)-1



(1)-2
SEND <種別A | 種別B | 種別C> {<, 種別A | 種別B | 種別C>}

種別A　:<CMD | DATA | LISTEN | TALK> 〔数値表現式 {, 数値表現式} 〕
種別B　:<DATA> 〔数値表現式 | 文字列表現式 {, 数値表現式 | 文字列表現式} 〕
種別C　:<UNL | UNT>

解　説

・GPIB上にユニバーサル・コマンド、アドレス・コマンド、およびデータなどを独立に送ります。

CMD　：アテンション（ATN)ラインを真（Low level)にして、与えられた数値をGPIBに送ります。ただし、数値は8ビットのバイナリ・データに変換されて、GPIBに出力されます。したがって、扱う数値は0 ～255 の範囲内で、また小数点表現の数値は自動的に整数に変換されます。

DATA　：ANT ラインを偽（High level）にして、与えられた数値をGPIBに送ります。ただし、ここで扱う数値は CMDで扱われるものと同様です。

EOI　：単線信号EOI を出力します。DATA以降に指定します。

LISTEN：与えられた数値を、リスナ・アドレス・グループ（LAG)としてGPIB上に送ります。数値は複数を指定できます。

TALK　：与えられた数値をトーカ・アドレス・グループ（TAG)としてGPIB上に送ります。ただし、数値は複数を指定できません。

UNT　：アントーク（UNT)コマンドをGPIBに送ります。このコマンドを実行する前にトーカに指定されていた装置は、トーカを解除されます。

UNL　：アンリスン（UNL)コマンドをGPIBに送ります。このコマンドを実行する前にリスナに指定されていた装置は、リスナを解除されます。

例

```
10 SEND UNT UNL LISTEN 1, 2, 3 TALK 4
20 SEND UNT CMD 63, 33 DATA 30, 54
```

注　意

ADDRESSABLE モードでは機能しません。

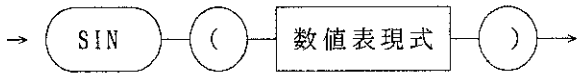
66. SIN

概　要

正弦値（サイン）を返す組み込み関数です。数値表現式で指定する値の単位は、ラジアンで指定します。

構　文

(1)-1



(1)-2

SIN(数値表現式) →

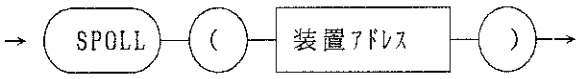
例

```
PRINT SIN(PI/2)
A=SIN(B)
```

67.　SPOLL

概　要　指定した装置のシリアル・ポールを行い、ステータス・バイトを読み込みます。

構　文　(1)-1



(1)-2
SPOLL（装置アドレス）

解　説
- 本器がSYSTEM CONTROLLER モードのとき、他のGPIB装置に対してシリアル・ポールを行います。
- 装置アドレスが 0～30のときは、各アドレスに対応した装置のシリアル・ポールを行います。
- 装置アドレスが31のとき、SYSTEM CONTROLLER モード、ADDRESSABLE モードに関係なく、本器に対してステータス・バイトを取り出します。

例　掲引終了の内部ステータスを検知します。

```
10 INTEGER Sp
20 OUTPUT 31;''IP''
30 OUTPUT 31;''CF30MZ SP100MZ SW2SC''
40 !
50 OUTPUT 31;''*CLS''     !ステータス・バイト・クリア
60 OUTPUT 31;''OPR8''     !掲引終了ビットをイネーブルにする
70 OUTPUT 31;''*SRE128'' !オペレーション・ステータス・イネーブル
80 OUTPUT 31;''SO''       !SRQ イネーブル
90 ON ISRQ GOTO 130
100 ENABLE INTR
110 GOTO 110
120 !
130 Sp=SPOLL(31)
140 PRINT ''STATUS = '';Sp
150 END
```

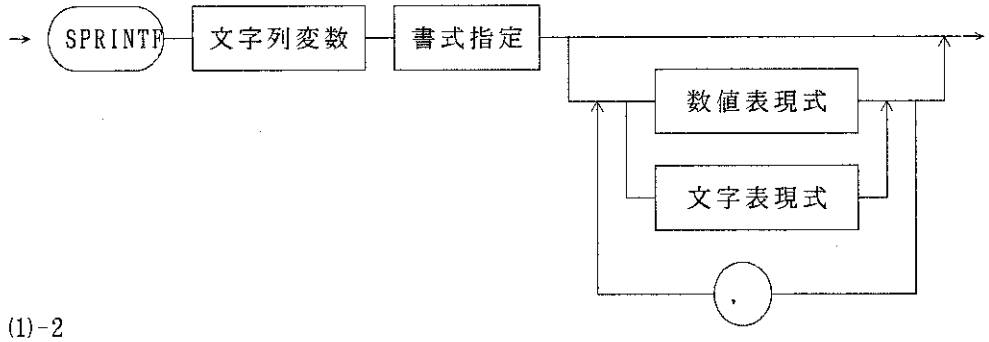
<実行結果>
STATUS = 192

注　意　ADDRESSABLE モード時に装置アドレス 0～30を指定し、SPOLL を行った場合は、0 が返ります。

68.　SPRINTF

概　要　PRINTFコマンドの書式変換仕様にしたがって書式を変換し、文字列変数に結果を代入します。

構　文　(1)-1



(1)-2
SPRINTF 文字列変数　書式指定〔数値表現式｜文字表現式
{,数値表現式｜文字表現式}〕

解　説

・PRINTFの書式変換の方法で式の値を変換して、最初のパラメータの文字列変数に結果を代入します。

・書式指定の方法と式の数、それに結果を入れる文字列変数の大きさには十分な注意が必要です。
特に結果をいれる文字列が結果に対して十分な大きさがないと、BASIC バッファを破壊する恐れがあります。

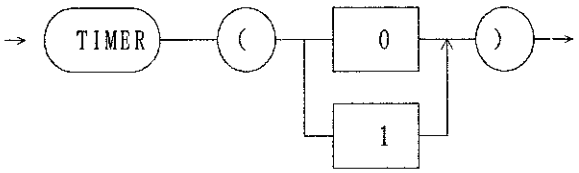
書式の指定方法は、4.3 節の[57. PRINTF]を参照して下さい。

69.　TIMER

概　要　内部システム時間の読み出しおよびリセットをします。

構　文

(1)-1



(1)-2

```
TIMER (0 | 1)
```

解　説

・内部システム時間をsec 単位の値で返す組み込み関数です。
この関数は、主に実行時間の測定などに使います。

引数 0を指定した場合 ：　内部システム時間の読み出し
引数 1を指定した場合 ：　内部システム時間のリセット

・読み出した値は10msecの分解能で、±10msecの誤差があります。

例

```
10  INTEGER I
20  TIMER(1)
30  FOR I=0 TO 10000
40  NEXT I
50  T1=TIMER(0)
60  !
70  TIMER(1)
80  FOR I=0 TO 10000
90    PRINT I
100 NEXT I
110 T2=TIMER(0)
120 !
130 PRINT "PRINT Command execute time is ";T2-T1
140 STOP
```

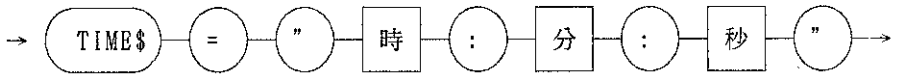
70.　TIME$

概　要　時刻の読み出しおよび設定をします。

構　文　(1)-1

(1)-2
TIME$

(2)-1



(2)-2
TIME$=" 時: 分: 秒"

解　説
- 本器内蔵の時計(RTC) の時刻を読み出します。
- 読み出した時刻は変更できます。
  以下のように入力して下さい。

```
TIME$="23:43:12"
TIME$="11:5:6"
```

例

```
10 DIM T$[10]
20 T$=TIME$
30 PRINT "Time is ";T$
40 PRINT "Time Reset"
50 TIME$="0:0:0"
60 STOP
```

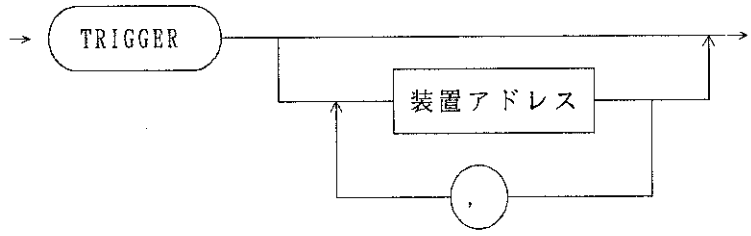
71.　TRIGGER

概　要

GPIB上に接続されているすべての装置、または選択された特定の装置にアドレス・コマンド・グループ（ACG)のグループ・エグゼキュート・トリガ（GET）を送ります。

構　文

(1)-1



(1)-2

TRIGGER 〔装置アドレス｛，装置アドレス　｝〕

解　説

・装置アドレスを指定せずにTRIGGER だけを実行すると、GPIBにはアドレス・コマンドのグループ・エグゼキュート・トリガ（Group Execute Trigger-GET)のみが送られます。この場合、トリガをかけたい装置はあらかじめリスナに設定しておかなければなりません。

・TRIGGER に続いて装置アドレスを指定すると、装置アドレスで指定された装置のみに GETコマンドを送ります。

例

```
10 TRIGGER 1
20 TRIGGER
```

注　意

ADDRESSABLE モードでは機能しません。

72.　WAIT

概　要　指定時間だけ待ちます。

構　文　(1)-1

(1)-2

WAIT　時間

解　説
・指定された時間待ちます。時間の単位はmsecです。
時間の設定範囲は 0～65535 です。

例

```
10 INTEGER T
20 T=30
30 PRINT T;"[msec] Wait !!"
40 WAIT T
50 STOP
```

73.　WAIT EVENT

概　要　指定したイベントが発生するまで待ちます。

構　文　(1)-1

(1)-2

WAIT EVENT　イベント番号

解　説　WAIT EVENT 1→掃引終了イベントまで待ちます。

例　掃引終了の内部ステータスを検知します。

```
10 OUTPUT 31;''IP''
20 OUTPUT 31;''CF30MZ SP100MZ SW2SC''
30 !
40 OUTPUT 31;''*CLS''     !ステータス・バイト・クリア
50 OUTPUT 31;''OPR8''     !掃引終了ビットをイネーブルにする
60 OUTPUT 31;''*SRE128'' !オペレーション・ステータス・イネーブル
70 WAIT EVENT 1
80 PRINT ''sweep end''
90 END
```

<実行結果>

```
sweep end
```

5.　エラー・メッセージ

5.1　エラー・メッセージを知る方法

PRINT ERRM$(0)を実行することにより、最近のエラーメッセージ（プログラム実行でのエラーの場合は行番号を含む）を表示します。

5.2　プログラムの実行位置(行)を知る方法

@ はシステム変数で、現在のプログラム実行行番号を知ることができます。

例）　PRINT @

5.3　エラー・メッセージ一覧

- 以下の表はエラー・メッセージをエラー・クラス（エラー番号）順に明記しています。なお、エラー・メッセージ中のxxx は文字列を、yyy は数値を意味しています。

- エラー・クラス内容：　1；データ入出力関係
  2；データ演算処理関係
  3；BASIC 構文解析関係
  5；その他

(1/6)

エラー・クラス (エラー番号)	エラー・メッセージ	内容
1(1)	xxx1(xxx2) error	xxx2のファイルに対してxxx1の命令が実行できない
1(2)	xxx1(xxx2, xxx3) error	xxx2, xxx3のファイルに対してxxx1の命令が実行できない
1(3)	Memory space full	BASIC 用のヒープ・メモリがない
1(4)	Bad free call	メモリ解放時に異常が検出された
1(5)	File format error	256 文字内にターミネータがない
1(6)	File is NOT open	指定のディスクリプタにファイルが登録されていない（ファイルがオープンされていない）
1(7)	Cannot read from "xxx" file	xxx のファイルから指定した文字数を読み込めない
1(8)	Cannot write to "xxx" file	xxx ファイルにデータを書き込めない

(2/6)

エラー・クラス (エラー番号)	エラー・メッセージ	内容
1(9)	"xxx" file cannot be opened	xxx のファイルがないためオープンできない
1(10)	Only one OUTPUT file can be opened	書込みモードで2 つ以上オープンしようとした
1(11)	Only one INPUT file can be opened	読込みモードで2 つ以上オープンしようとした
1(12)	xxx: "xxx" file was opened with xxx mode	オープンしたときと違うモードでアクセスした
1(13)	"xxx" file is already opened	すでにオープンされているファイルをオープンしようとした
1(14)	End of "xxx" file	EOF 検出後、更に読み込もうとした
1(15)	Abort	GPIB制御ステートメントが中断、またはGPIBバス上で異常が検出された
1(16)	Time out	GPIBタイム・アウト
1(17)	GPIB syntax error	GPIBコマンドが間違っている
1(18)	Missing GPIB code	GPIBコマンドが間違っている
1(19)	yyy: Unit address error in xxx	xxx コマンドにおいてGPIBアドレス指定が間違っている
1(20)	Device not ready	指定されたドライブにデバイスがない
1(21)	Invalid character	意味のない文字が指定された
1(22)	Device not found	指定されたデバイスが見つからない
1(23)	Too many dots	"." の数が多すぎる(CHDIRコマンド)
1(24)	Root has no parent	指定されたディレクトリが見つからない
1(25)	No such file or directory	指定されたファイル、またはディレクトリが見つからない
1(26)	Already 8 files are opened	既に8 個のファイルがオープンされている

(3/6)

ｴﾗｰ･ｸﾗｽ (ｴﾗｰ番号)	エラー・メッセージ	内容
1(27)	Cannot execute to protected file	プロテクト・ファイルなのでアクセスできない
1(28)	Illegal file name	プロテクト・ファイルをCOPY、RENAMEしようとした
2(30)	Zero divide	0 による除算が行われた
2(3i)	Substring error	サブストリング指定が間違っている
2(32)	Overflow value	数値が扱える範囲を超えている
2(33)	Invalid string constant	”” 内に文字がない
2(34)	String length is too long	文字列変数の宣言が多すぎる（最大128 文字）
2(35)	xxx: Cannot convert into string	文字列中にASCII 変換できない文字が検出された
3(40)	Program is NOT exist	プログラムがない状態でRUN しようとした
3(41)	xxx: Syntax error	文法が違う（致命的なエラー）
3(42)	Expression format error	文法の書式が違う（引数、セパレータ指定時等）
3(43)	Parameter error	引数の書式、または型が違っている
3(44)	xxx2: Invalid type in xxx1	xxx1の中に無効な型(xxx2)が指定された
3(45)	xxx2: Invalid first type in xxx1	xxx1構文の最初(xxx2)が間違っている
3(46)	xxx2: Invalid second type in xxx1	xxx1構文の 2番目(xxx2)が間違っている
3(47)	Cannot assigned into this token	指定された文字列変数に値が代入できない

(4/6)

ｴﾗｰ･ｸﾗｽ (ｴﾗｰ番号)	エラー・メッセージ	内容
3(48)	No operand in xxx	xxx の演算書式が間違っている
3(49)	Not found DATA statement	RESTORE する先にDATA文がない
3(50)	Unmatched DATA's values and READ variable	DATA文で指定した値とREADで読み込む数が合わない(READ で読み込むデータがない)
3(51)	Unmatched IMAGE-spec in USING	USING のイメージ指定が間違っている
3(52)	Not found THEN in IF	IF文の後にTHENがない
3(53)	FOR's nest is abnormal	FOR 文の入れ子が正しくない
3(54)	FOR variable does NOT exist	FOR 文のカウンタ変数がない
3(54)	NEXT variable does NOT exist	NEXT文のカウンタ変数がない
3(55)	FOR < init value > does NOT exist	FOR 文の初期値がない
3(56)	Unbalanced FOR variable in NEXT	FOR 文とNEXT文の関係が正しくない
3(57)	Unbalanced NEXT statement	FOR 文の後のNEXT文がない
3(58)	Unbalanced BREAK	BREAK 文がFOR ～ NEXT 間にない
3(59)	SELECT nesting overflow	SELECT文の入れ子が多すぎる
3(60)	GOSUB nest overflow	GOSUB ～ RETURN の入れ子が多すぎる
3(61)	Label "xxx" is already exists	既に存在するラベルを更に登録しようとした

(5/6)

エラー・クラス (エラー番号)	エラー・メッセージ	内容
3(62)	Unbalanced xxx	xxx の構文上バランスがとれない
3(63)	Unbalanced xxx block	xxx ブロック(IF, SELECT 等) が合わない
3(64)	Invalid dimension parameter	配列変数のパラメータが間違っている
3(65)	Array's range error	配列変数の添字が指定範囲外である
3(66)	Array's declaration error	配列変数の書式または使用方法が間違っている
3(67)	Unbalanced line No	行番号の指定がおかしい
3(68)	xxx: Reserved value	システム変数等の予約後に値を代入しようとした
3(69)	Undefined label	指定されたラベルがない
3(70)	Label not found	指定されたラベルが存在しない
3(71)	Unknown line No	指定された行がない
3(72)	Not found line No.yyy	行番号 yyyの行が存在しない
3(73)	Line No.yyy is out of range	マイナスまたは65535 を超える行番号が指定された
3(74)	Cannot move line	REN コマンドによる行番号変更ができない
3(75)	xxx error(s) appeared	プログラム実行時のトータル・エラー数表示
3(76)	Not available ASCII char(yyy)	無効なASCII 文字yyy(16進表示) が検出された
3(77)	Program mistaked at xxx	プログラムに間違いがある
3(78)	xxx: Undefined icode	無効な中間コードが検出された
3(79)	Undeclare value	配列宣言無しに配列変数を使おうとした
3(80)	Cannot specify USING	指定のファイル・タイプではUSING は使用できない
3(81)	Program cannot changed	プログラム実行中にプログラムを変更しようとした

(6/6)

ｴﾗｰ･ｸﾗｽ (ｴﾗｰ番号)	エラー・メッセージ	内容
3(82)	Uninstalled type (xxx)	変数の書式が間違っている
3(83)	Paramerter is out of range	指定範囲外の値が引数で渡された
3(84)	Merge program is NOT exist	MERGE していないのにMERGE DEL を実行しようとした
5(90)	Program cannot be continued	終了したプログラムを再実行しようとした
5(91)	Warning: cannot execute 'xxx'	xxx コマンドが実行できない状態である
5(92)	Warning: cannot execute 'xxx'(yyy)	yyy 実行中にxxx を実行しようとした
5(93)	Cannot execute 'xxx' (yyy)	yyy 状態時にxxx を実行しようとした
	Unmatched FOR to NEXT variable	FOR ～ NEXT のカウンタ変数が違っている
	Unmatched number of FOR to NEXT	FOR,NEXTの入れ子が正しくない

索引

―――― 数字 ――――

―――― アルファベット順 ――――

【A】

【B】

【C】

【D】

【E】

【F】

【G】

【I】

【L】

【M】

【N】

【O】

【P】

【R】

【S】

【T】

【W】

50音順

【あ】

【え】

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こと
- 許可なく複製、修正、改変を行うこと
- 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こと

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収日）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トでは、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ト 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ない場合があります。
アドバンテストでは、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に実施することで、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薦めいたします。
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ト

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
TEL 0120-919-570
FAX 0120-057-508
E-mail : icc@acs.advantest.co.jp