ADVANTEST®
株式会社アドバンテスト

取 扱 説 明 書

R3265/3271シリーズ
オプション15
レファレンス

MANUAL NUMBER OJC00 9306

適用機種
R3265
R3365
R3271
R3371
R3271MS

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

初版1991年10月30日
Printed in Japan

緒言

1．R3265/3271シリーズ説明書の種類

R3365/3371として使用するときは、R3265/3271をR3365/3371と読みかえて下さい。

	種　類	内　容	備　考
1.	R3265/3271ｼﾘｰｽﾞ ｽﾍﾟｸﾄﾗﾑ･ｱﾅﾗｲｻﾞ 取扱説明書	R3265/3271の取扱方法を説明しています。 ・付属品 ・パネル面の説明 ・機能説明 ・操作説明 ・性能諸元　など	R3265/3271本体の標準付属品です。
2.	R3265/3271ｼﾘｰｽﾞ クイック・ガイド	R3265/3271のキー操作を基本から応用まで具体例で示しています。 ・ｽﾀｰﾄ/ｽﾄｯﾌﾟ周波数の設定手順 ・測定ウィンドウ内での掃引手順 ・データ・ストア手順 ・ソフト・メニューの初期化手順 ・占有周波数帯域幅測定手順　など	R3265/3271本体の標準付属品です。
3.	R3265/3271ｼﾘｰｽﾞ オプション15 取扱説明書	R3265/3271用　オプション15の取扱方法を説明しています。 ・ガイド ・レファレンス	R3265/3271オプション15の標準付属品です。
4.	R3265/3271ｼﾘｰｽﾞ オプション73 取扱説明書	R3265/3271用　オプション73の取扱方法を説明しています。 ・第１部：GPIB概要 ・第２部：GPIBコマンド拡張モード１ 8562のコマンドをサポート ・第３部：GPIBコマンド拡張モード２ 8566のコマンドをサポート	R3265/3271オプション73の標準付属品です。

2．本書は、R3265/3271シリーズ用オプション15を詳しく説明しています。

関連マニュアル

・R3265/3271シリーズ　オプション15（ガイド）
　オプション15の機能と操作方法を、Q&A形式でやさしく説明しています。本書より先にお読み下さい。

3．このオプション15に接続可能な外部端末は、VG-920およびVT-220相当品です。

4．本書は、以下に示す略語を用いています。

略　語		意　　味
PC	：	日本電気株式会社製、PC9801のパーソナル・コンピュータを表わします。
HP	：	ヒューレット・パッカード社製のパーソナル・コンピュータを表わします。
VG-920	：	ビクターデータシステムズ社製の端末を表わします。
VT-220	：	DEC製の端末を表わします。
パソコン	：	パーソナル・コンピュータを表わします。

5．本書は、3部構成になっています。

第　1　部
1.　概要
2.～5.　計測器に外部端末を接続した場合の説明
6.～9.　計測器のみの場合の説明

第　2　部
1.　オプション15（BASIC GPIB コントローラ）
2.　コマンドとステートメントの文法と解説
3.　ビルトイン関数

付　　録
A.1　機能別コマンドとステートメント一覧
A.2　機能別ビルトイン関数とグラフィック機能一覧
A.3　パラメータ指定一覧
A.4　エラー・メッセージ一覧

6．製品概要

本オプション15は、スペクトラム・アナライザR3265/3271に内蔵するコントローラ機能です。

コントローラの言語はわかりやすいBASIC言語を使用し、本器自身（R3265/3271）のコントロールはもちろんのこと、GPIBで接続された他のGPIB機器のコントロールも可能です。またパラレルI/Oで、パラレル入出力機器の制御も可能です。

プログラムの作成には、外部端末、本器自身、ダウンロードなどいくつかの方法があります。

作成したプログラムやデータなどはICメモリ・カードに保存できます。保存されたプログラムを使用すると、本器のみでプログラムの実行が可能となります。

第１部　ａｔｅエディタ

第１部は、計測器に外部端末(VG-920)を用いた場合と、計測器のみの場合に分けてateエディタを説明しています。

2～5 章　計測器に外部端末(VG-920)を接続した場合の説明
- 2章　特徴
- 3章　起動
- 4章　機能一覧
- 5章　詳細解説

6～9 章　計測器のみの場合の説明
- 6章　特徴
- 7章　起動
- 8章　機能一覧
- 9章　機能解説

第１部の目次

図　一　覧

1．　概要

ateエディタは、計測器上でBASICプログラミングを容易に行えるように開発されたフルスクリーンエディタです。

計測器の背面パネルにあるRS-232Cコネクタに接続した外部端末(VG-920)を用いて編集でき、操作性が大変向上しました。また、計測器のみでも編集でき、現場でのプログラムの修正などポータブル性も兼ね備えています。

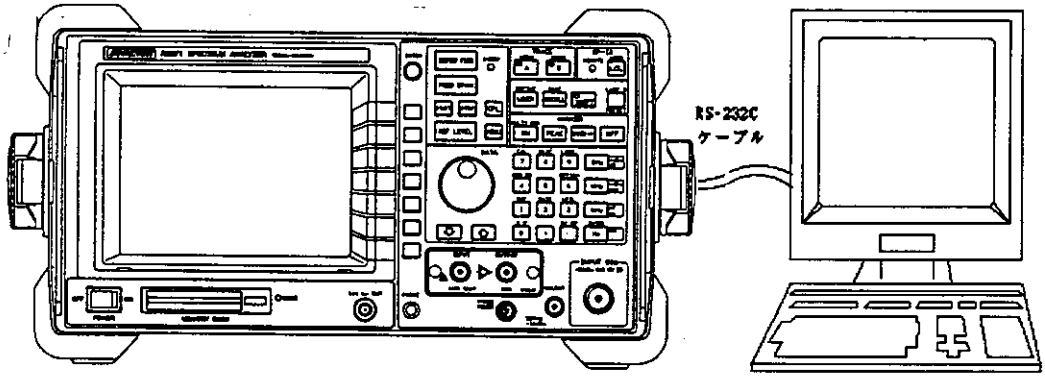
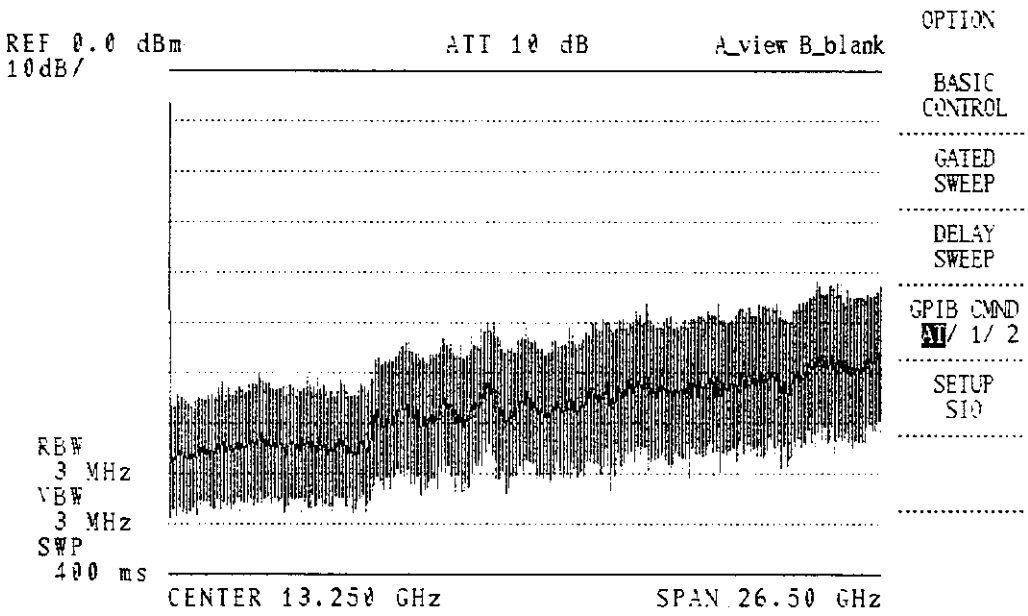


図1-1　R3265/3271とVG-920の接続

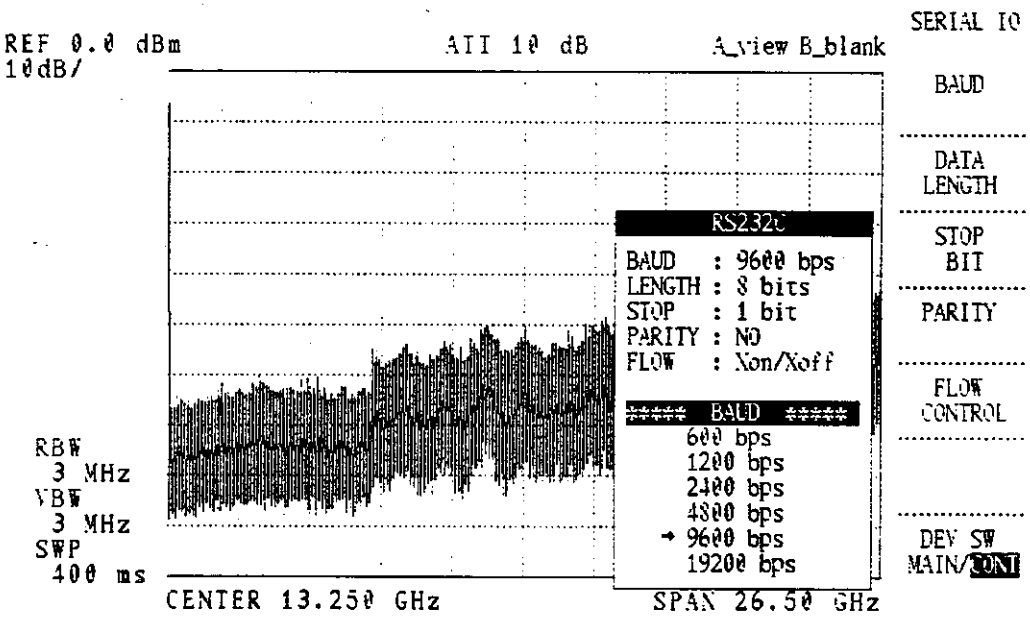
RS-232C

オプション１５で外部端末を使用する場合にRS-232Cポートを設定します。

SIFT ＋ OPTION 6 を押すと以下のソフトメニューになります。



RS-232Cは [SETUP SIO] を押すと以下の設定画面が表示され、設定できます。



設定項目

BAUD

転送速度(ﾎﾞｰﾚｰﾄ)を指定します。選択できる速度は以下に示します。

600 bps
1200 bps
2400 bps
4800 bps
9600 bps （初期値）
19200 bps

DATA LENGTH

ﾃﾞｰﾀ長を指定します。選択できるﾃﾞｰﾀ長は以下に示します。

7 bits
8 bits （初期値）

STOP BIT

ｽﾄｯﾌﾟ･ﾋﾞｯﾄを指定します。選択できるｽﾄｯﾌﾟ･ﾋﾞｯﾄは以下に示します。

1 bit （初期値）
1.5 bits
2 bits

PARITY

ﾊﾟﾘﾃｨを指定します。選択できるﾊﾟﾘﾃｨは以下に示します。

NO　　（初期値）
ODD
EXEN

FLOW COTROL

X on/offを指定します。

HARD　（ｵﾌﾟｼｮﾝ15指定時は、HARDをｻﾎﾟｰﾄしません。）
Xon/off（初期値）

DEV SW MAIN/CONT

RS-232C切り換え

MAIN　　ｵﾌﾟｼｮﾝ02使用時は、MAINを選択する。
CONT　　ｵﾌﾟｼｮﾝ15使用時は、CONTを選択する。

MEMO

2．　特長（計測器＋外部端末の場合）

ateエディタは、高度な編集機能を備えたフルスクリーンエディタです。
機能としてカーソル制御、テキストの挿入、削除、コピー、移動、置換、検索、ウィンドウ、ファイル、BASICプログラムの実行などがあります。 また、これらの機能を円滑にするためにポップアップメニュー、ヘルプメニューがあります。

2．1　キーボード

ateエディタの基本的な機能は端末のキーボードに割り当てられています。複雑な編集作業が簡単なキー操作でできます。

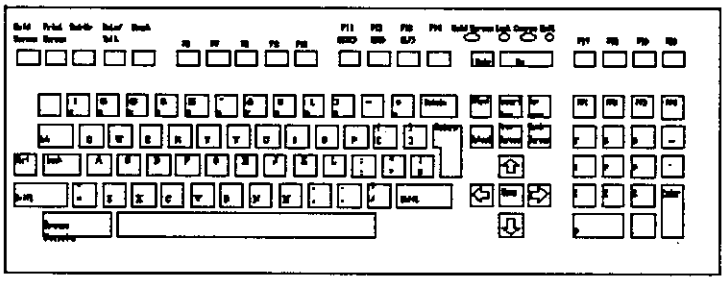


図2-1　VG-920のキーボード

2．2　ポップアップメニュー

ateエディタの機能すべてがポップアップメニューで実行できます(カーソル移動は除く）。
実行したい機能をポップアップメニュー内よりカーソルキーで選択し、<Return>キーを押すと、その機能が実行されます。

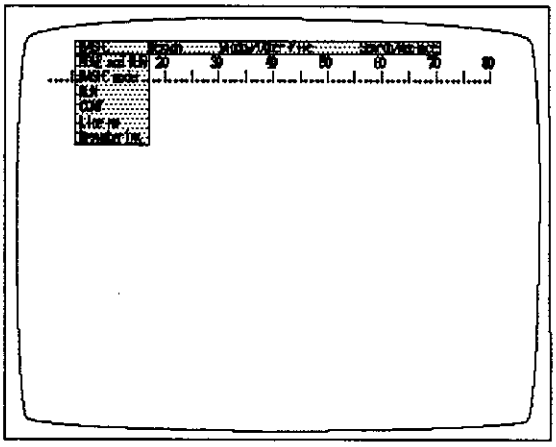


図2-2　ポップアップメニュー

２．３　BASIC実行環境

ateエディタで編集したBASICプログラムをそのままエディタ上で実行できます。
BASICのRUN, CONT, SCRATCHコマンドは、キーボードまたはポップアップメニューによって実行します。それ以外のコマンドはBASICモードのミニウィンドウ内で実行します。

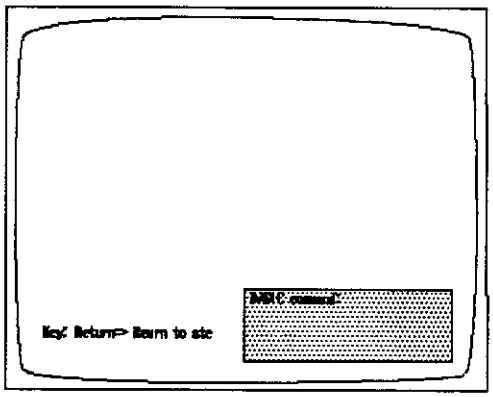


図2-3　BASICモード

２．４　ラベルの定義

ateエディタでは、行番号は省略しラベルを定義します。これにより、サブルーチン・コールなどを使うとき、わかりやすいプログラムを作成できます。

注）ラベルは文字列の先頭に必ず＊（アスタリスク）をつけます。

```
・・・・|・・・・1・・・・|・・・・2
   A=0
*LABEL
   A=A+1
   IF A >= 10 THEN
      GOTO *ABCD
   END IF
   GOTO *LABEL
*ABCD
```

図2-4　ラベルの定義

２．５　ファイル

作成したプログラムは、ファイル単位で本体のメモリ・カードへセーブできます。
また、メモリ・カードからファイルをロードして編集もできます。

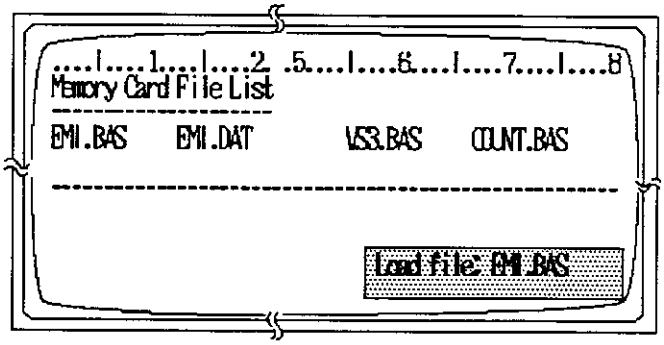


図2-5　ファイルのロード

２．６　ヘルプメニュー

VG-920のキーボードに割り当てられている機能が一覧表示されます。

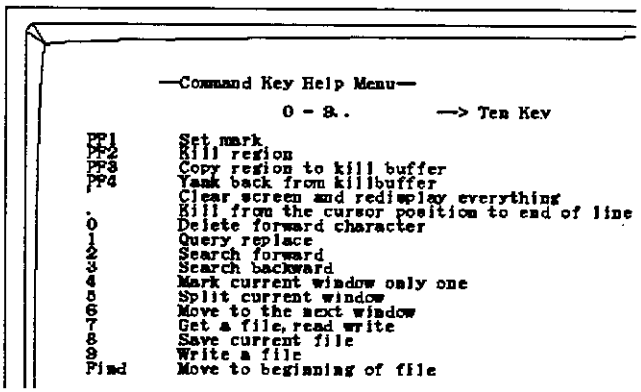


図2-6　ヘルプメニュー

MEMO

3．　起動（計測器＋外部端末の場合）

（1）　操作

計測器に端末（VG-920)を接続し電源を入れると以下に示す初期画面が表示され、ateエディタが起動します。

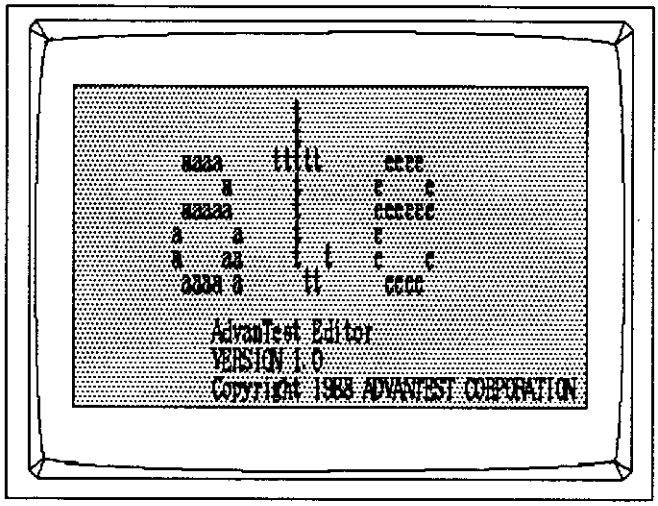


図3-1　ateの初期画面

次に何かキーを押すと、以下に示す画面に変わります。

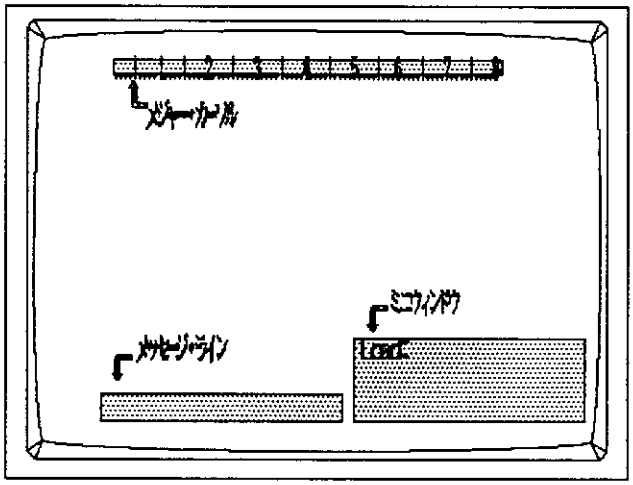


図3-2　ateの動作画面

（２）　画面表示の説明

画面の表示機能を説明します。

・メジャーカーソル
　　編集時にカーソル位置を把握するためのメジャー表示です。

・メッセージライン
　　コマンドの実行手順、実行結果、エラーメッセージを表示します。

・ミニウィンドウ
　　コマンドを実行する場合、パラメータが必要なときに使います。バッファ名、ファイル名、文字列などパラメータを要求するコマンドに応じて様々です。
　　表示されるプロンプトに従ってパラメータを入力します。パラメータの後に必ず＜Return＞キーを入力して下さい。プロンプトの後ろの［　］で囲まれた部分はデフォルトパラメータです。このデフォルトはパラメータを入力しないで＜Return＞キーを押すとパラメータになります。

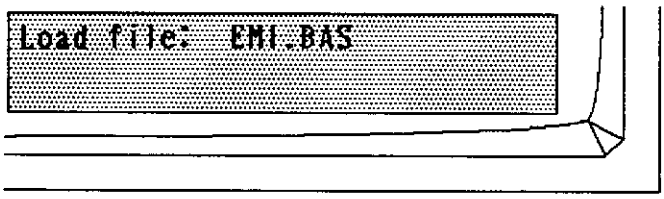


図3-3　ミニウィンドウ

4．　機能一覧（計測器＋外部端末の場合）

ateエディタの各機能は一部を除いてキーボード上にメイン・キーパッド、編集キーパッド、補助キーパッド、ファンクション・キーパッドと割り当てられています。

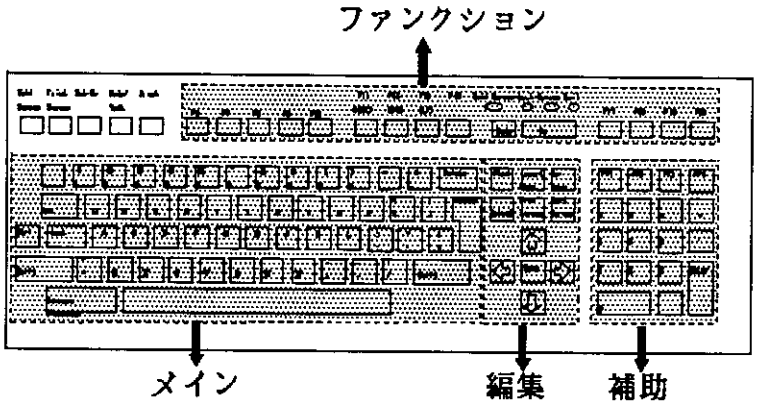


図4-1　キーパッド配置図

4．1　カーソル制御

→	次の文字に移動	（編集キーパッド）
←	前の文字に移動	（編集キーパッド）
↑	前の行に移動	（編集キーパッド）
↓	次の行に移動	（編集キーパッド）
Find	ファイルの先頭に移動	（編集キーパッド）
Insert-Here	ファイルの末尾に移動	（編集キーパッド）
Remove	行の先頭に移動	（編集キーパッド）
Select	行の末尾に移動	（編集キーパッド）
Prev-screen	前に一画面分移動	（編集キーパッド）
Next-screen	次に一画面分移動	（編集キーパッド）

4．2　挿入

Return	改行文字を挿入	（メイン・キーパッド）
Tab	タブを挿入	（メイン・キーパッド）
上記以外のメインキーパッド		
	一般文字	

4．3　削除

Delete	カーソル前の文字を削除	
0	カーソル位置の文字を削除	（補助キーパッド）
.	カーソル位置から行末までを削除	（補助キーパッド）

４．４　コピー、移動、削除（範囲指定）

PF1	リージョン処理に必要なマークをカーソル位置にセットする（補助キーパッド）
PF2	マークからカーソル位置までを削除し、バッファに入れる（補助キーパッド）
PF3	マークからカーソル位置までをバッファに入れる（補助キーパッド）
PF4	PF2キーまたはPF3キーによりバッファに入れられた部分をカーソル位置にコピーする　（補助キーパッド）

４．５　ウィンドウ

4	ウィンドウを１つにする　（補助キーパッド）
5	ウィンドウを２つに分割する　（補助キーパッド）
6	カーソルを次のウィンドウに移動（補助キーパッド）
.	画面をクリアして再表示する　（補助キーパッド）

４．６　ファイル

7	メモリカードから指定したファイルをロードする（補助キーパッド）
8	メモリカードにファイルをセーブする（補助キーパッド）
9	メモリカードに指定したファイル名でセーブする（補助キーパッド）
－	ファイル名一覧ウィンドウへカーソルを移動する（補助キーパッド）

４．７　検索

2	前方に文字列を検索（補助キーパッド）
3	後方に文字列を検索（補助キーパッド）

４．８　置換

1	文字列を置き換え　（補助キーパッド）

４．９　BASICモード

F13	自動的な行番号挿入を設定（ファンクションキーパッド）
F14	行番号の再定義　（ファンクションキーパッド）
F17	プログラムをBASICバッファに転送後、実行する（ファンクションキーパッド）
F18	BASICモードへ移行する　（ファンクションキーパッド）
F19	すでに転送済みのプログラムを実行する（ファンクションキーパッド）
F20	Ctrl-Cで中断したプログラムを続行する（ファンクションキーパッド）

４．１０　ヘルプ

Help	エディタ機能とキー割当の一覧　（ファンクションキーパッド）

４．１１　ポップアップメニュー

Do（Ctrl-A）	ポップアップメニューを表示する（ファンクションキーパッド）

４．１２　キャンセル

F11（Ctrl-Z）	エディタの機能を中止する　（ファンクションキーパッド）

４．１３　エディタの再起動

F12	現エディタを終了し、バッファを初期化して再起動する （ファンクションキーパッド）

MEMO

5．　機能解説（計測器＋外部端末の場合）

5．1　カーソル制御

カーソルは編集位置を示すものであり、編集はカーソルを基準にして行われます。

カーソル制御とは、画面上でカーソルを特定の場所に移動することを言います。まず、1文字ずつ上下左右に移動する機能があります。これらの機能はそれぞれカーソルキーに割り当てられており、現在の場所から新しい場所へカーソルを移動します。

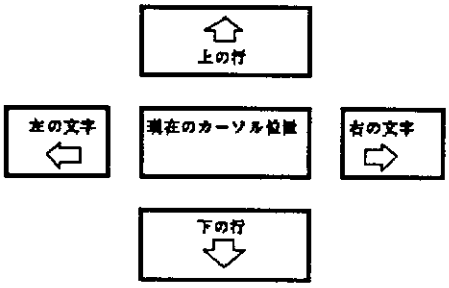


図5-1　カーソルキー

これらは基本的なカーソル移動機能をもち、頻繁に使います。
画面の上限や下限を超えて移動しようとすると、その方向にあるテキストがスクロールし、カーソルは常に画面内に存在します。

この他に編集キーパッドに割り当てられている機能があります。

Find	ファイルの先頭に移動する
Insert-Here	ファイルの末尾に移動する
Remove	行の先頭に移動する
Select	行の末尾に移動する
Next-Screen	次のページに進む
Prev-Screen	前のページに戻る

図5-2　カーソル移動に関するキーパッド

５．２　挿入

ateエディタでは、常にテキストの挿入ができる状態になっているので、キーボードを叩くと制御文字以外はすべて挿入されます。文字は一般文字と制御文字の2つに分けられます。一般文字とは'A','1'など目に見える文字を言い、制御文字とは'ESC','CTRL'など目に見えない文字を言います。

一行が画面の幅以上に長くなると右端に'$'記号を表示して、その行が継続されていることを示します。

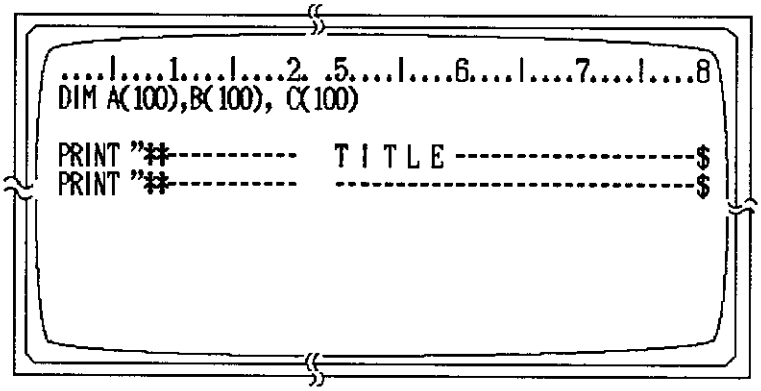


図5-3　文字の継続を表す

行の終わりは<Return>キーを入力すると改行文字で表わされます。また、文字の位置を調節するタブ機能は<Tab>キーを入力します。タブサイズは8文字分です。

Return	改行文字を挿入する
Tab	タブを挿入する
上記以外のメインキーパッド	一般文字を挿入する

図5-4　挿入に関するキーパッド

５．３　削除

削除には、一文字削除、一行削除、範囲指定削除があります。これらの機能を説明します。

５．３．１　一文字削除

一文字削除には、カーソル直前の文字を削除するものと、カーソル位置の文字を削除するものの2通りの方法があります。

（１）カーソル直前の文字を削除する場合
カーソル直前の文字を削除するには、<Delete>キーを入力します。

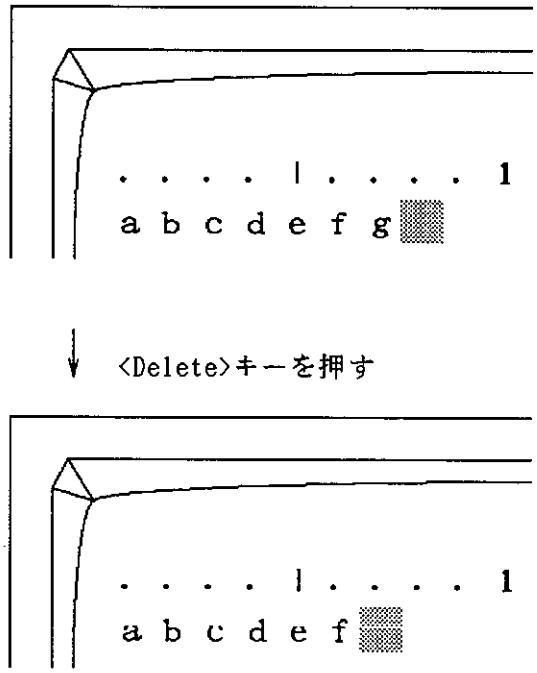


図5-5　カーソル直前の文字を削除

（２）カーソル位置の文字を削除する場合

カーソル位置の文字を削除するには、<0>キーを入力します。

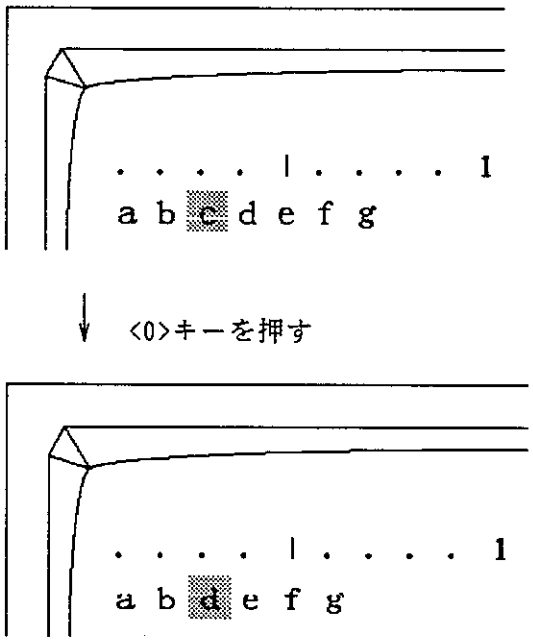


図5-6　カーソル位置の文字を削除

５．３．２　一行削除

カーソル位置から行末までを削除するには、<.>キーを入力します。最初の<.>キーでその行の内容が削除され、再度<.>キーを入力するとその行自身が削除されます。

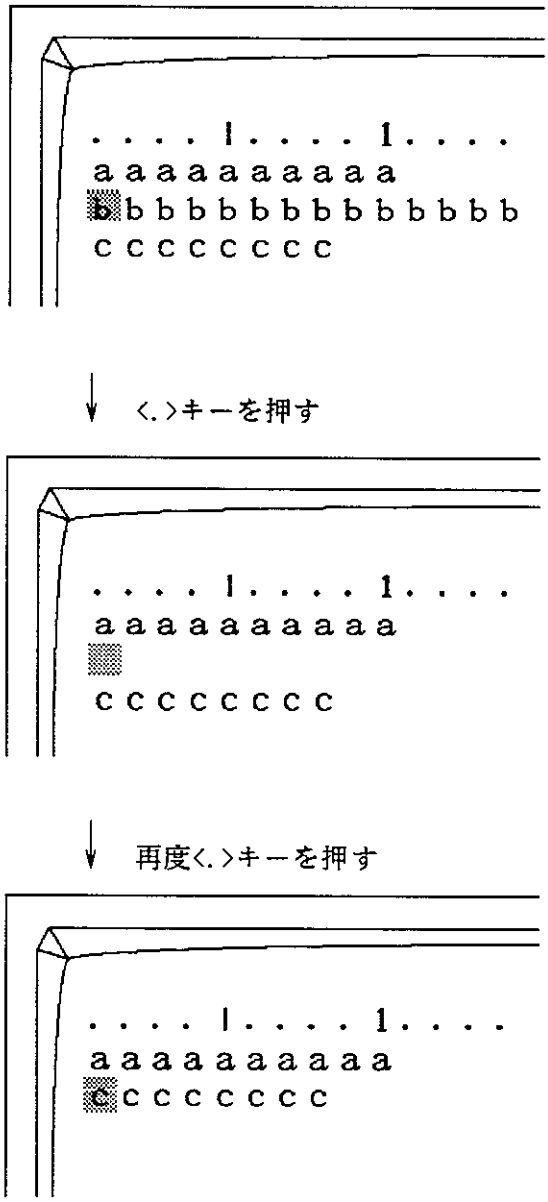


図5-7　カーソルから行末までを削除

５．３．３　範囲指定削除

範囲を指定して削除する方法は、テキストの保存と関連があるため[5.4節]で説明します。

Delete	カーソル直前の文字を削除
0	カーソル位置の文字を削除
.	カーソルから行末までを削除

図5-8　削除に関するキーパッド

５．４　コピー、移動、削除（範囲指定）

コピー、移動、削除をするために範囲を指定してテキストを保存します。

５．４．１　テキストの削除

（１）マークセット

削除、移動をするときは削除する範囲の先頭位置にカーソルを移動して、<PF1>キーまたはポップアップメニューの"Mark set"でマークを設定します。

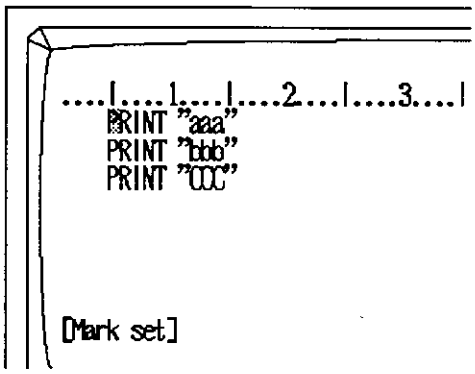


図5-9　マークセット

（２）テキストの削除

削除する範囲の最終位置よりも1文字分先へカーソルを移動し、<PF2>キーまたはポップアップメニューの"Kill region"を入力します。すると、(1)でマークをセットした位置から現在のカーソル位置の手前までのテキストを削除します。

削除したテキストは内部バッファに保存されているので後述のテキストの移動、復旧、コピーなどは(1)，(2)の操作をした後、実行します。

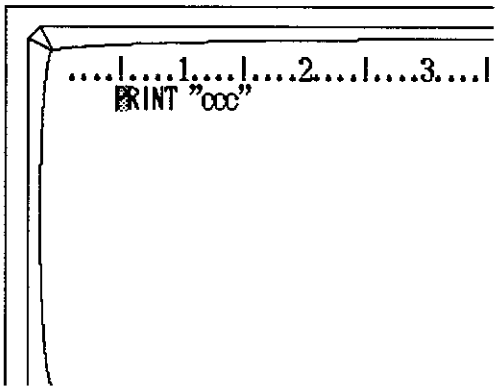


図5-10　テキストの保存（削除）

５．４．２　テキストの復旧、移動

削除したテキストを元に戻すには現在のカーソル位置で<PF4>キーまたは、ポップアップメニューの"Yank"を入力します。テキストを移動する場合は移動したい位置へカーソルを移動し、<PF4>キーまたはポップアップメニューの"Yank"を入力します。

このように<PF4>キーと、ポップアップメニューの"Yank"は、保存したテキストを現在のカーソル位置に挿入する機能です。

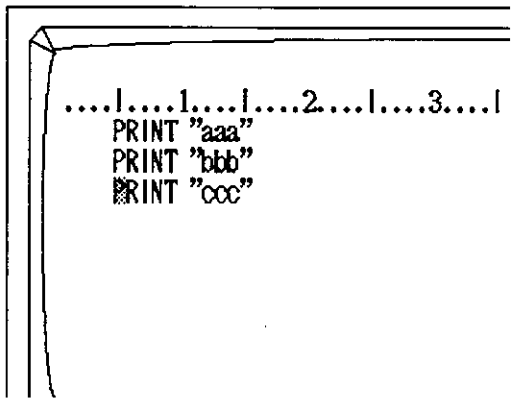


図5-11　テキストの復旧

５．４．３　テキストのコピー

テキストをコピーするには範囲の先頭位置を削除する場合と同様に設定し最終位置にカーソルを移動し、<PF3>キーまたはポップアップメニューの"Copy region"を入力します。すると範囲内は何も変わったように見えませんが、指定したテキストが内部バッファにセーブされているので、コピー位置にカーソルを移動し、<PF4>キーまたはポップアップメニューの"Yank"を入力します。

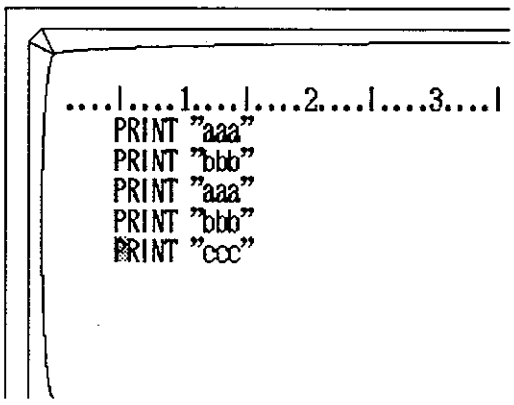


図5-12　テキストのコピー

<PF4>キーまたはポップアップメニューの"Yank"によって復旧されるテキストは、内部バッファ内の最新のテキストになります。

PF1	範囲指定に必要なマークをカーソル位置にセットする
PF2	マークからカーソル位置までを削除し、テキストを保存する
PF3	マークからカーソル位置までのテキストを保存する
PF4	保存したテキストをカーソル位置に挿入する

図5-13　リージョン処理に関するキーパッド

５．５　ウィンドウ

通常のウィンドウはオーバーラッピング型とタイル型に分けられますが、ateエディタのウィンドウは画面を上下に分割するタイル型になります。

通常のウィンドウは1つですが、ateエディタは2つのウィンドウを持つことができ、それぞれに対してテキストを表示します。例えば、画面内に表示しきれない2つの部分を同時に参照できます。

注）オープンできるファイルは1つに限られているため、2つのウィンドウにそれぞれ別のファイルをオープンできません。

５．５．１　ウィンドウの分割

ウィンドウを2つに分割するには、<5>キーまたはポップアップメニューの"Split"を入力します。

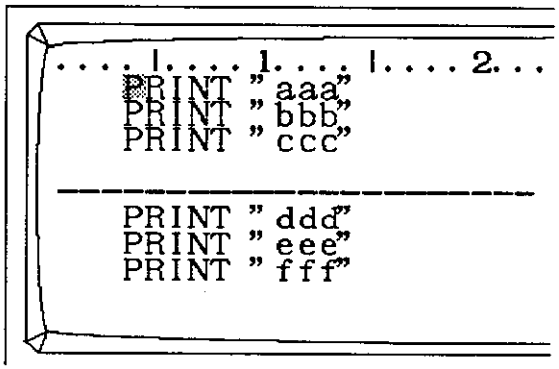


図5-14　上下に分割されたウィンドウ

５．５．２　ウィンドウの初期化

上下に分割されたウィンドウを1つに戻すには、<6>キーまたはポップアップメニューの"Next"を入力して、残したいウィンドウにカーソルを移動します。次に、<4>キーまたはポップアップメニューの"Only"を入力すると、ウィンドウは1つに戻ります。

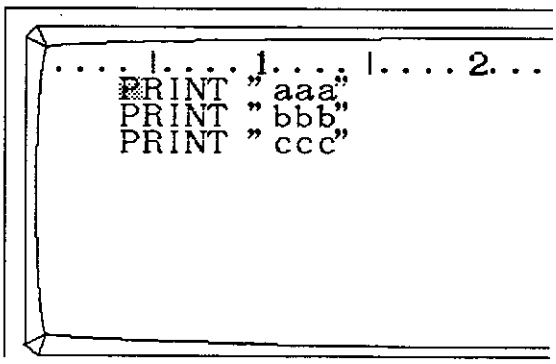


図5-15　ウィンドウを１つにする

５．５．３　画面の再表示

編集中のテキストを再表示したいときは、<⁂>キーまたはポップアップメニューの"Redisplay"を入力して、テキストを再表示します。

4	他のウィンドウを消去して、カーソルのあるウィンドウ１つにする
5	ウィンドウを上下に分割し、カーソルを上のウィンドウに置く
6	カーソルを次のウィンドウに移動する
,	画面をクリアして、テキストの再表示を行う

図5-16　ウィンドウに関するキーパッド

５．６　ファイル

ateエディタでは、テキストをファイルという単位として、本体のメモリ・カードへのセーブ、メモリ・カードからのロードができます。編集したテキストは必ずメモリ・カードへセーブするようにします。

注）セーブしないままエディタを終了すると、編集したテキストは失われます。

ここでは、ファイルのセーブ／ロードを説明します。

５．６．１　ファイルのセーブ

編集したテキストをメモリ・カードにセーブするには、<S>キーまたはポップアップメニューの"Write"を入力します。ここでセーブするファイルに名前をつけます。ファイル名は最大10文字です。

セーブが終了すると、書き込んだ行数をメッセージラインに表示します。

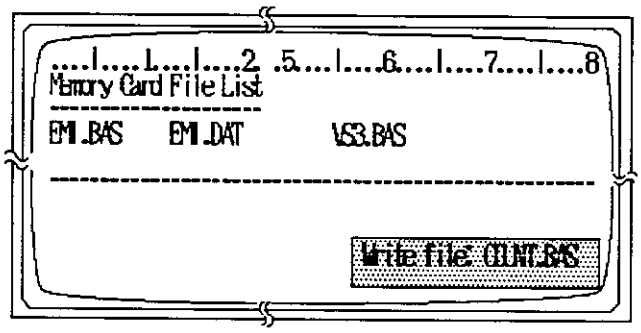


図5-17　テキストのセーブ（ファイル名を入力）

５．６．２　ファイルのロード

セーブされているファイルをロードするには、<↑>キーまたはポップアップメニューの"Load"を入力します。すると、画面上に現在メモリ・カードにセーブされているファイルが一覧表示され、どのファイルをロードするかを尋ねてきます。ここでファイル名を入力するか、または一覧表示されているウィンドウへ<↓>キーを押してカーソルを移動させます。次に、カーソルキーにてファイル名を選択し、最後に<Return>キーを入力します。

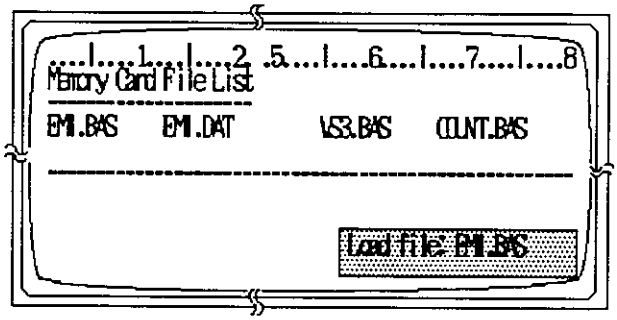


図5-18　ファイル名の入力

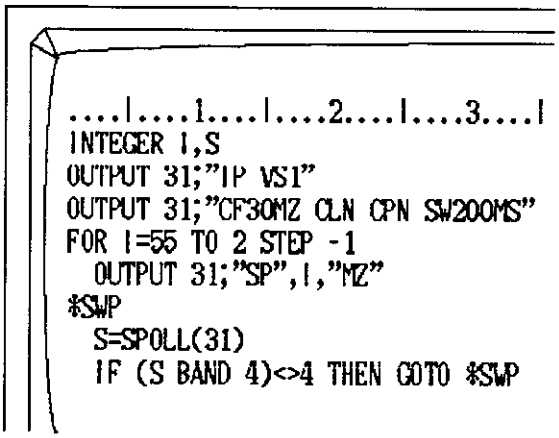


図5-19　ファイルのロード

メモリ・カードからファイルをロードしている最中は、本体のメモリ・カード挿入口上部のLEDが点灯します。ロードが終了すると、画面上にテキストが表示され、メッセージラインに読み込んだ行数が表示されます。

このとき、行番号がついているものは自動行番号挿入(AUTO)機能が設定されます。

また、新しいファイルを作成するときは、そのファイル名を入力します。

メモリ・カードが挿入されていないときは、ブザー音またはエラーメッセージが表示されます。

５．６．３　ファイルの更新

同じファイル名でセーブするには、<8>キーまたはポップアップメニューの"Save"を入力します。すると、セーブするかどうかを尋ねてきます。
テキストに何の変更もない場合は"No change"というメッセージを表示します。

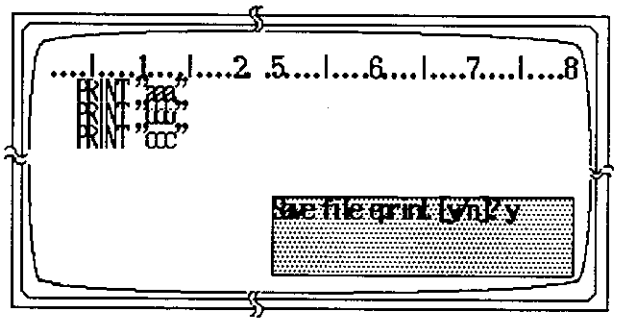


図5-20　テキストの更新（同一ファイル名）

<y> <Return>と入力すると、テキストはファイルに書き出されます。セーブが終了すると、そのファイルの行数がメッセージラインに表示されます。
<n> <Return>と入力すると、テキストのセーブは実行しません。

7	メモリ・カードから指定したファイルをロードする
8	メモリ・カードに同一ファイル名でセーブする
9	メモリ・カードに指定したファイル名でセーブする
−	ファイル表示一覧のウィンドウへ移動する

図5-21　ファイルに関するキーパッド

5．7　検索

文字列をテキスト内で前方または後方に向かって探す機能です。
カーソル位置以降を検索するには、〈2〉キーまたはポップアップメニューの"Forward search"を入力します。カーソル位置以前を検索するには、〈3〉キーまたはポップアップメニューの"Backward search"を入力します。両者は検索の方向が異なるだけですべて同じ働きをします。

以下にカーソル位置以降の文字列の検索手順を示します。
〈2〉キーまたはポップアップメニューの"Forward search"を入力します。画面にミニウィンドウが表示され、検索する文字列を尋ねてきたら、探したい文字列を入力します。

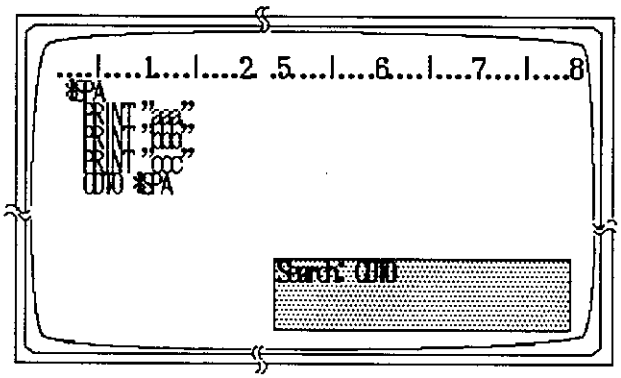


図5-22　カーソル位置以降の文字列検索

検索した文字列の先頭にカーソルが移動し、検索は終了します。同じ文字列を続けて検索したいときは、同様の操作を行いますが、文字列の入力は省略できます。

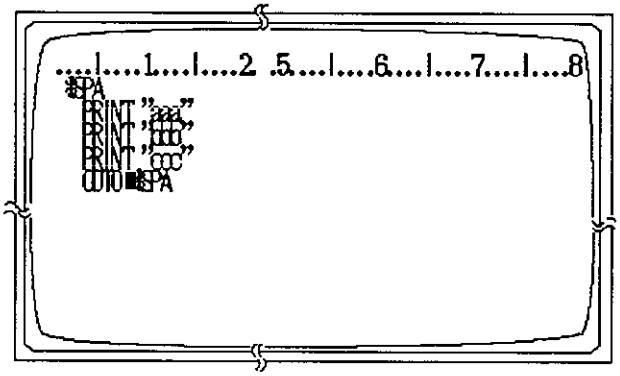


図5-23　実行結果

検索できないときは、"Not found"というメッセージが表示されます。

2	カーソル位置以降に向かって検索する
3	カーソル位置以前に向かって検索する

図5-24　検索に関するキーパッド

5．8　置換

カーソル位置以降のテキスト中の任意の文字列を変更する機能です。

それでは、文字列を置き換えてみます。<①>キーまたはポップアップメニューの"Query replace"を入力します。画面にミニウィンドウが表示され、置換したい文字列を尋ねてきたら、対象となる文字列を入力します。

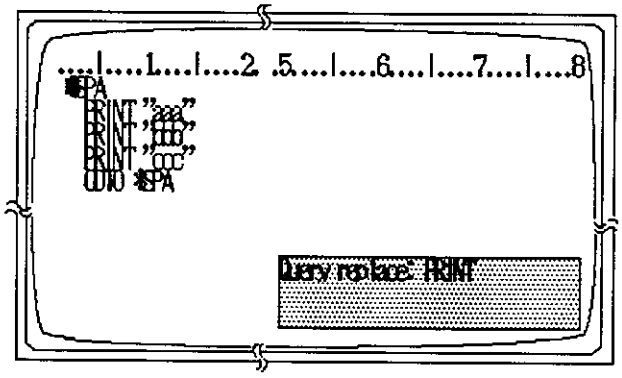


図5-25　置換対象文字列の入力

次に新たに変更する文字列を尋ねてきたら、任意の文字列を入力します。

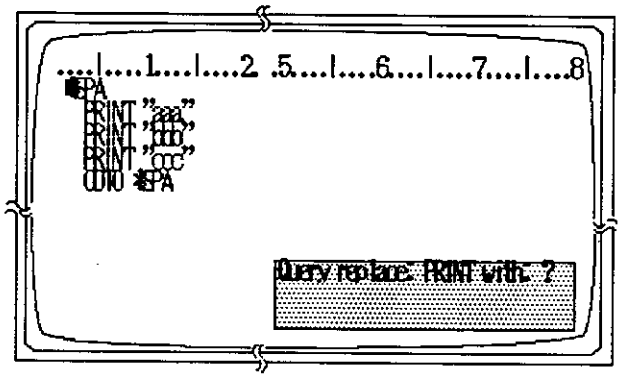


図5-26　置換文字列の入力

該当する文字列を検索すると、カーソルがその文字列の先頭に移動し、置換するかどうか待ち状態になります。ここで置換するときには<space>キーを入力し、置換しないときには<delete>キーを入力します。また、置換して終了するときには<.>キーを入力し、置換しないで終了するときには<F1>または<Ctrl-Z>キーを入力します。

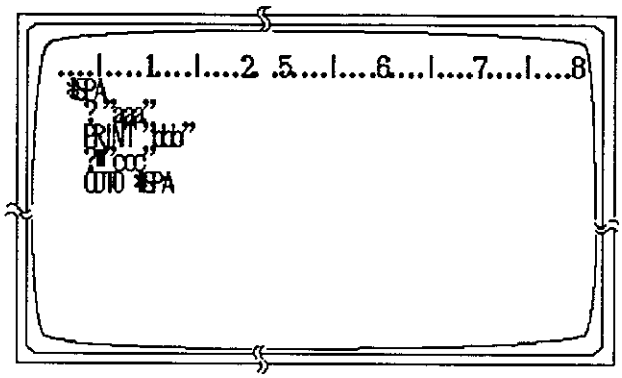


図5-27　実行結果

1	文字列を置き換える
space	置換する
delete	置換しない
.	置換して、終了する
F11(Ctrl-Z)	終了する

図5-28　置換に関するキーパッド

5．9　BASIC

ateエディタは、計測器上でのBASICプログラミングを行うために開発されたポータブル・エディタです。よって編集しながらBASICの実行、デバッグが容易にできます。その他にBASICモードに移行してから様々なBASICコマンドの実行ができます。

ここではエディタ上から実行できるBASICコマンドを説明します。

5．9．1　行番号の設定

今までBASICプログラムを作成するときは行番号を使いました。ateエディタでは行番号の代わりにラベルを用いて、プログラムを分かりやすくします。

```
A=1:B=0
*LOOP
        B=B+A^2
        IF B>10000000 THEN GOTO *ENDLOOP
        PRINT B
        GOTO *LOOP
*ENDLOOP
STOP
```

図5-29　ラベルを用いたプログラム

また、自動行番号挿入（AUTO）機能により行番号をつけながら編集もできます。<F13>キーまたはポップアップメニューの"Line no."を入力すると、ミニウィンドウが表示され、開始番号と間隔を指定します。これで編集時に<Return>キーを入力すると、行番号が自動的に出力されます。

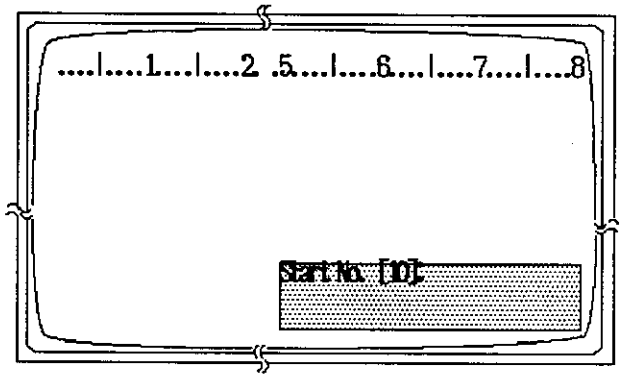


図5-30　開始番号の指定

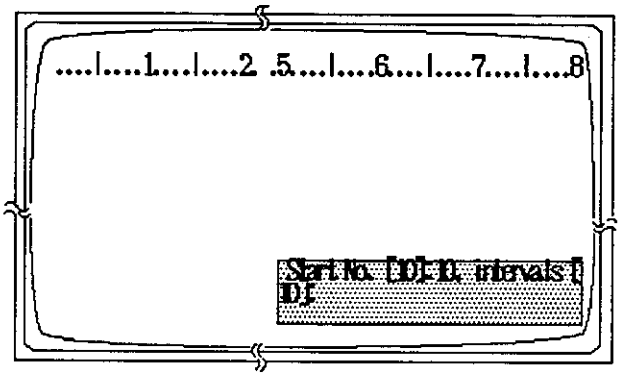


図5-31　行番号間隔の指定

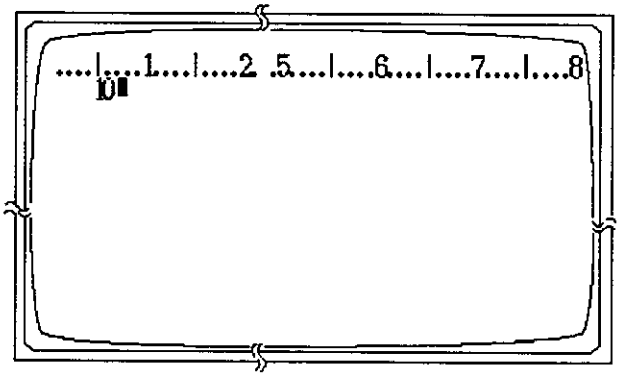


図5-32　自動行番号機能

また、編集途中の場合でも、この機能により行番号を追加することができます。
行間での挿入を行ったときは、上の行番号にプラス１した数が行番号となります。

注）一度この機能を実行すると、エディタの初期化(5.13節)をしない限り継続されます。

自動行番号挿入機能と同様、行番号のリナンバリング機能があります。これは行番号を一定の指定にしたがって付け直すものです。<F14>キーまたはポップアップメニューの"Renumbering"を入力すると、自動行番号挿入機能と同様に、開始番号と間隔を指定されます。

５．９．２　ＢＡＳＩＣの実行

プログラムを実行する前に、ateエディタとBASICインタプリタの関係を説明します。
プログラムの編集はateエディタで制御し、完成したプログラムはateエディタの内部バッファに格納されています。これとは別にBASICインタプリタも内部バッファを持っていて、実行するためにはこのバッファにBASICプログラムが存在しなければなりません。

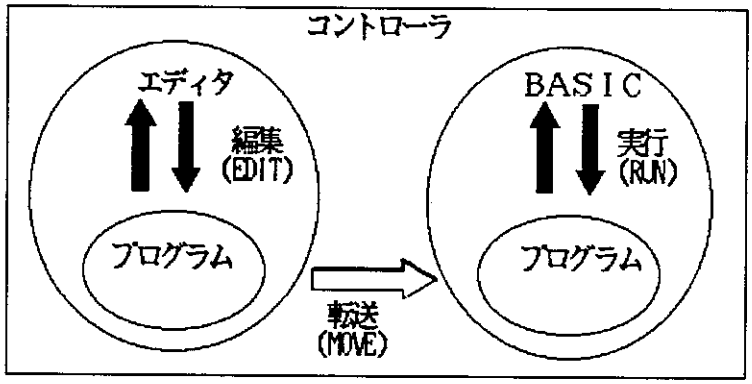


図5-33　エディタとインタプリタとの関係

よってBASICプログラムを実行する前には、エディタからインタプリタにプログラムを転送します。<F17>キーまたはポップアップメニューの"MOVE and RUN"を入力すると、エディタからプログラムが転送され、実行します。このとき、前回実行したプログラムは消去されます。また、実行中にエラーが起こると、エディタに戻り、カーソルがエラー行に移動します。

すでに転送済みのプログラムを実行するときは、<F19>キーまたはポップアップメニューの"RUN"を入力します。プログラムを転送する前に<F19>キーまたはポップアップメニューの"RUN"を実行したときは、"Program is not exist"というエラーメッセージが表示されます。
実行が終了すると、ミニウィンドウに"BASIC command:"のプロンプトが表示され、デバッグ環境になります。このままエディタへ戻るには、<Return>を入力します。

５．９．３　ＢＡＳＩＣの停止

実行中のBASICを停止するには、<Ctrl-C>を入力します。

５．９．４　ＢＡＳＩＣモード

BASICモードとは、BASICに対してのコマンドを実行できる環境を言います。たとえば、前述のデバッグ環境です.<F18>キーまたはポップアップメニューの"BASIC mode"を入力すると、ミニウィンドウが表示されます。

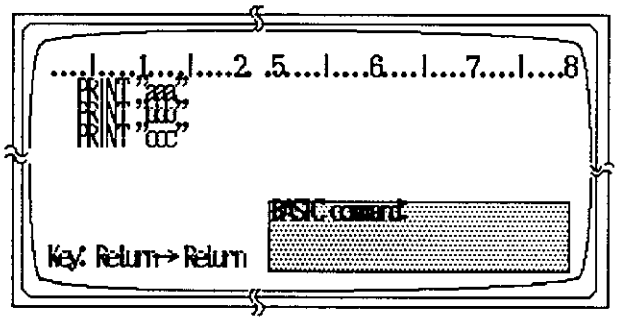


図5-34　BASICモード

実行できるコマンドは、RUN, LIST, CONTなどです。(コマンドの説明は［第2部システム・コントローラ］を参照)

注）プログラムの内容を変更する場合は必ずエディタに戻ってから行います。

５．９．５　ＢＡＳＩＣの継続

<Ctrl-C>またはPAUSE命令で一時停止したプログラムを続けて実行するときは、<F20>キーまたはポップアップメニューの"CONT"を入力します.これはBASICコマンドの"CONT"と同じです。停止した次の文から実行し、実行するプログラムがない場合は、"Program cannot be continued"というエラーメッセージが表示されます。

F13	自動的に行番号を挿入する
F14	行番号のリナンバリング
F17	プログラムを転送後、実行する
F18	BASICモードへ移行する
F19	すでに転送済みのプログラムを実行する
F20	Ctrl-CまたはPAUSE命令で中断したプログラムを続行する

図5-35　BASICに関するキーパッド

5．10　ヘルプ

エディタの機能とキーボード（VG-920）の関係が画面上で確認できます。

<HELP>キーまたはポップアップメニューの"Help"を入力すると、どのキーがどんな機能を実行するか、簡単な説明が表示されます。

メニュー画面のスクロールなどカーソルを移動させる機能は編集機能と同様です。ただし、左右の移動機能はありません。

<F11>または<Ctrl-Z>キーを入力すると、ヘルプ機能を中止し、元の編集画面に戻ります。

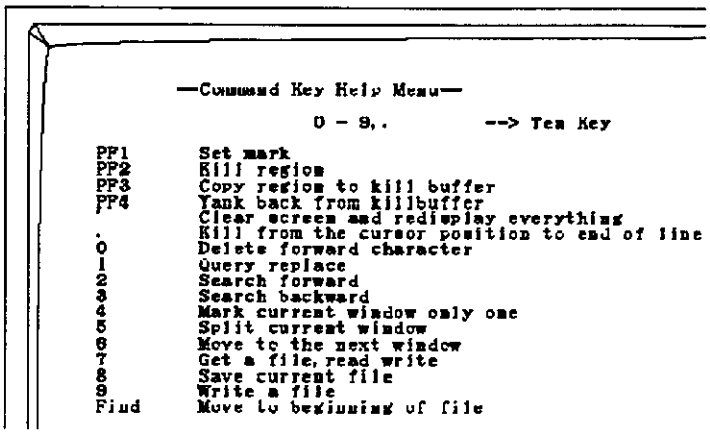


図5-36　ヘルプメニュー

HELP	ヘルプメニューを表示する
F11(Ctrl-Z)	ヘルプメニューを中止する

図5-37　ヘルプに関するキーパッド

5．11　ポップアップメニュー

前項まで説明してきたateエディタの機能は、すべてキーボードを操作して実行しました。これから説明するポップアップメニューを用いても同様の操作ができます。

カーソルキー以外のエディタ機能はすべてポップアップメニュー内に表示され、その中から選択し、実行します。これによりキー割当がわからないときや、本体単独で使うときなどにとても役立ちます。

<Do>または<Ctrl-A>キーを入力すると、画面最上部にポップアップメニューが表示されます。

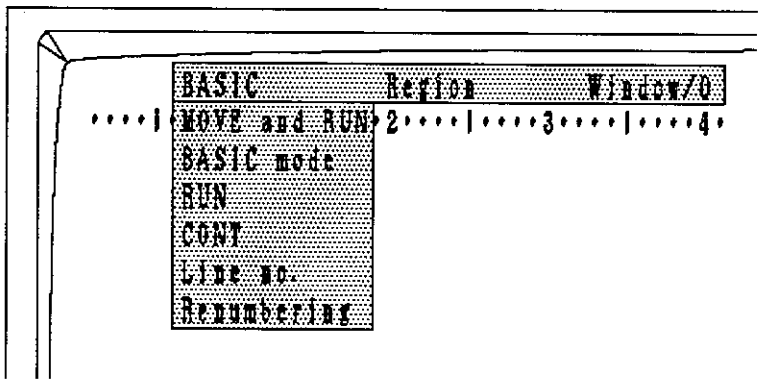


図5-38　ポップアップメニュー

BASIC	Region	Window/Other	File	Search/Replace
MOVE and RUN	Set mark	Only	Load	Forward search
BASIC mode	Kill region	Split	Save	Backward search
RUN	Copy region	Next	Write	Query replace
CONT	Yank	Redisplay		
Line No.		Help		
Renumbering		SCRATCH		

図5-39 ポップアップメニュー一覧

各項目間の移動は<→>,<←>または<F6>～<F10>キーで行います。<→>,<←>キーは左右に移動し、<F6>～<F10>キーはそれぞれの項目に割当たっています。

F6	BASICメニュー
F7	Regionメニュー
F8	Window/Otherメニュー
F9	Fileメニュー
F10	Search/Replaceメニュー

図5-40　ポップアップメニューに関するキーパッド

該当する項目のメニューが表示されたら、<↑>,<↓>キーで実行する機能を選択します。

Do(Ctrl-A)	ポップアップメニューの表示
F11(Ctrl-Z)	ポップアップメニューの中止

図5-41　ポップアップメニューに関するキーパッド

メニュー	説　　明	参照先
MOVE and RUN	BASICプログラムを転送し、実行する	5.9.2項
BASIC mode	BASICコマンドを実行する	5.9.4項
RUN	すでに転送済みのプログラムを実行する	5.9.2項
CONT	Ctrl-Cにより停止したプログラムを継続する	5.9.5項
Line no.	行番号を設定する	5.9.1項
Renumbering	行番号を再設定する	5.9.1項
Set mark	テキストの削除、コピーのための先頭位置をセットする	5.4.1項
Kill region	テキストのマークからカーソル位置までを削除する	5.4.1項
Copy region	テキストのマークからカーソル位置までを保存する	5.4.2項
Yank	保存されたテキストをコピー（復旧）する	5.4.3項
Only	ウィンドウを一つにする	5.5.2項
Split	ウィンドウを上下に分割する	5.5.1項
Next	他のウィンドウにカーソルを移動する	5.5 節
Redisplay	テキストの再表示を行う	5.5.3項
Help	VG-920に割り当てられたエディタ機能を表示する	5.10節
SCRATCH	エディタを初期化する	5.13節
Load	メモリ・カードからファイルをロードする	5.7.2項
Save	ファイルを更新する	5.7.3項
Write	メモリ・カードへファイルをセーブする	5.7.1項
Forward search	カーソル位置から後方へ向かって文字列を検索する	5.7 節
Backward search	カーソル位置から前方へ向かって文字列を検索する	5.7 節
Query replace	文字列の置き換えを行なう	5.8 節

図5-42　　ポップアップメニューの説明

５．１２　ポップアップメニューの中止

ポップアップメニューを中止するときには、<F11>または<Ctrl-Z>キーを入力します。

注）BASICバッファへテキスト転送中や、ファイルのセーブ／ロード中に中止できません。

F11(Ctrl-Z)　　エディタ機能の中止

図5-43　キャンセルに関するキーパッド

５．１３　エディタの初期化

現在起動しているエディタを初期化します。<F12>キーまたはポップアップメニューの"SCRATCH"を入力すると、ミニウィンドウが表示され、編集中のテキストをセーブするかどうかを尋ねてきます。（テキストを変更した場合のみ）

それに返答すると、エディタを初期化するかどうかを尋ねてきます。

<y> <Return>と入力すると、エディタは初期化され、<n> <Return>と入力すると初期化は中止し、元のエディタ画面に戻ります。

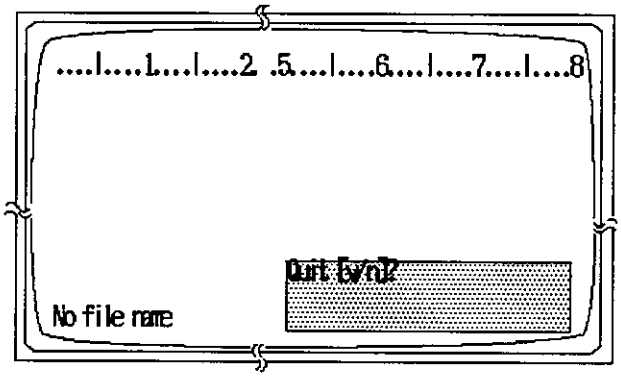


図5-44　エディタの終了ミニウィンドウ

<y> <Return> を入入力する

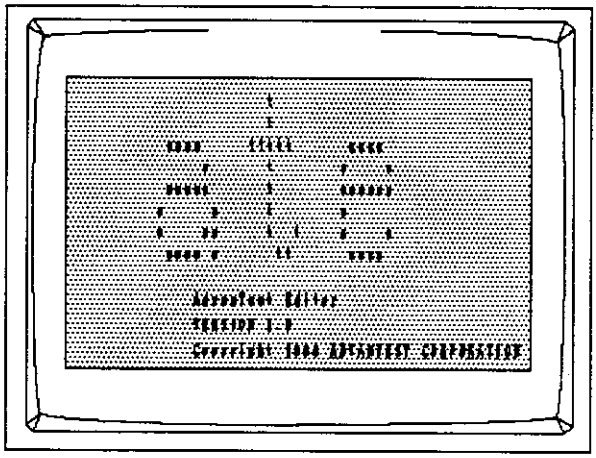


図5-45　エディタの初期化

F12	ateエディタを初期化する

図5-46　エディタの初期化に関するキーパッド

6．　特長（計測器のみの場合）

ateエディタは、高度な編集機能を備えたフルスクリーンエディタです。
ateエディタの機能には、カーソル制御、テキストの挿入、削除、コピー、移動、置換、検索、ウィンドウ、ファイル、BASICプログラムの実行、などがあります。また、これらの機能を円滑にするためにポップアップメニューがあります。
以下に外部端末を用いた場合との相違点を表します。

	計測器＋VG-920	計測器のみ
最大カラム数	８０	７２
最大行数	２３	１５
キーボード	全機能の割当	一部機能の割当
文字入力	キーボードより入力	ラベル機能により入力

図6-1　外部端末使用時との相違点

6．1　キーボード

計測器に外部端末を接続した場合、全機能が端末のキーボードに割り当てられていたのに対し、計測器のみの場合、一部の機能だけが正面パネルに割り当てられています。

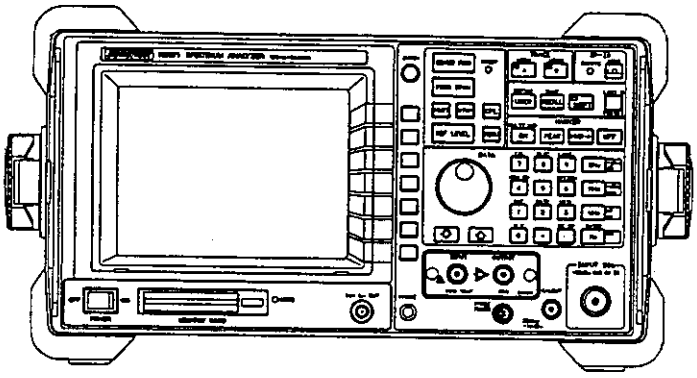


図6-2　R3265/3271正面パネル

6．2　ポップアップメニュー

外部端末を使用したときと同様に、カーソル移動を除くすべての機能がポップアップメニューで実行します。キーボードがないときは、このポップアップメニューでの編集が主になります。

実行したい機能をポップアップメニュー内よりカーソルキーで選択し、<単位>キーを押すと、その機能が実行されます。

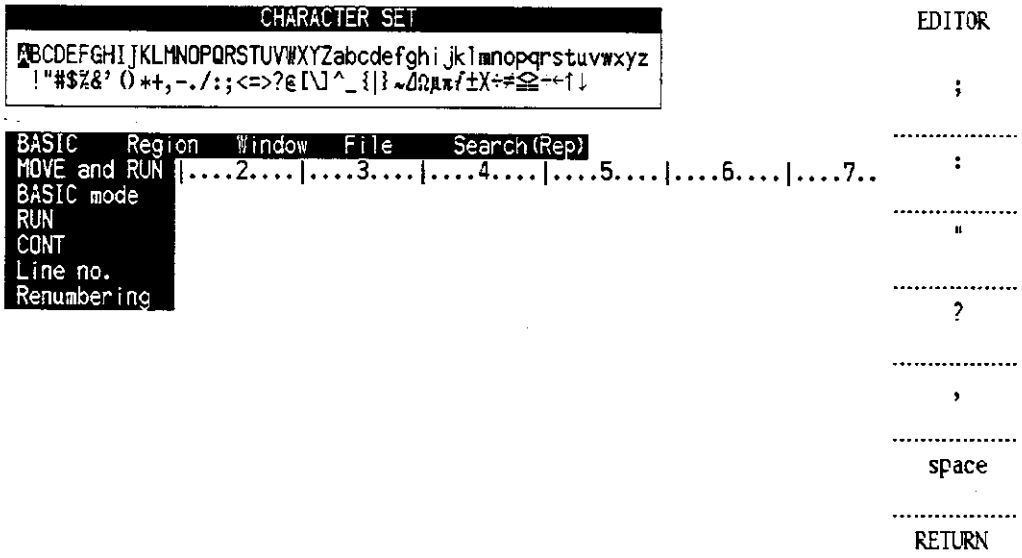


図6-3　ポップアップメニュー

6．3　BASIC実行環境

ateエディタで編集したBASICプログラムをそのままエディタ上で実行できます。
BASICのRUN, CONT, SCRATCHコマンドは、ソフトキーまたはポップアップメニューで実行します。それ以外のコマンドはBASICモードのミニウィンドウ内で実行します。

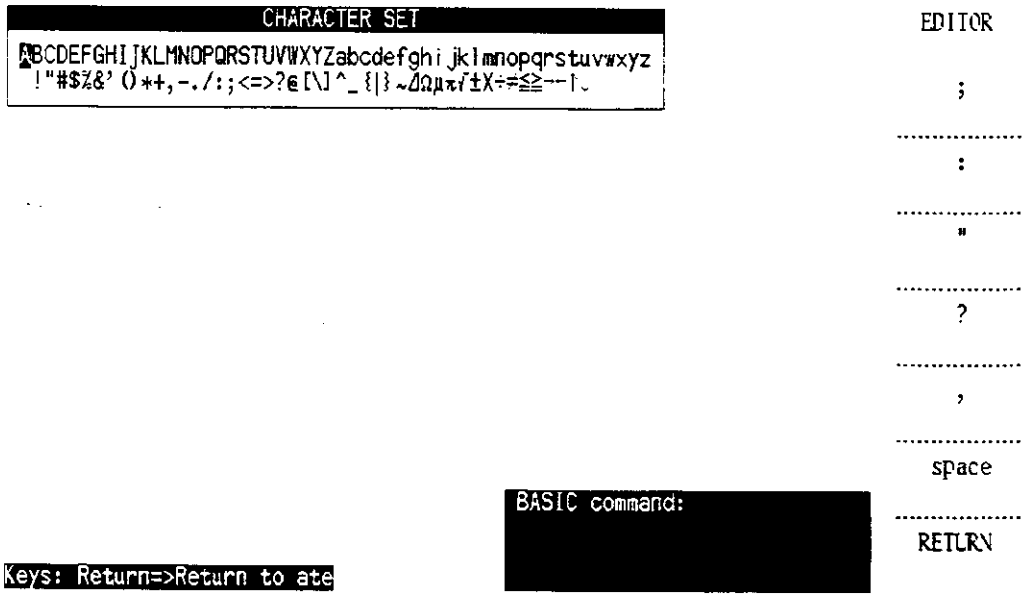


図6-4　BASICモード

6．4　ラベルの定義

ateエディタでは、行番号は省略し、ラベルを定義します。これにより、サブルーチン・コールなどを使うとき、わかりやすいプログラムが作成できます。

注）ラベルは文字列の先頭に必ず＊（アスタリスク）をつけます。

```
A=1:B=0
*LOOP
        B=B+A^2
        IF B>10000000 THEN GOTO *ENDLOOP
        PRINT B
        GOTO *LOOP
*ENDLOOP
STOP
```

図6-5　ラベルの定義

6．5　ファイル

作成したプログラムは、ファイル単位で本体のメモリ・カードへセーブできます。また、メモリ・カードからファイルをロードして、編集もできます。

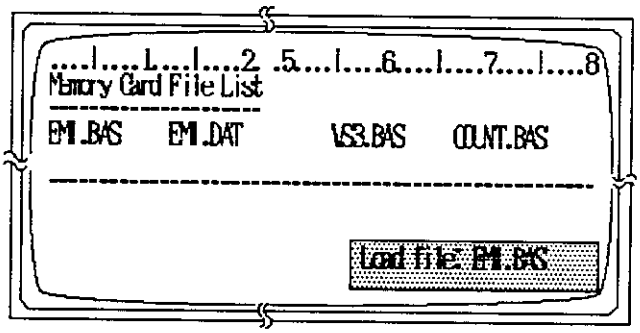


図6-6　ファイルのロード

6．6　文字入力

計測器に外部端末を接続したときは、キーボードから入力できますが、計測器のみではラベル機能を用いて入力します。このとき、<SHIFT>キーを押してからラベル機能の要領で文字を入力します。

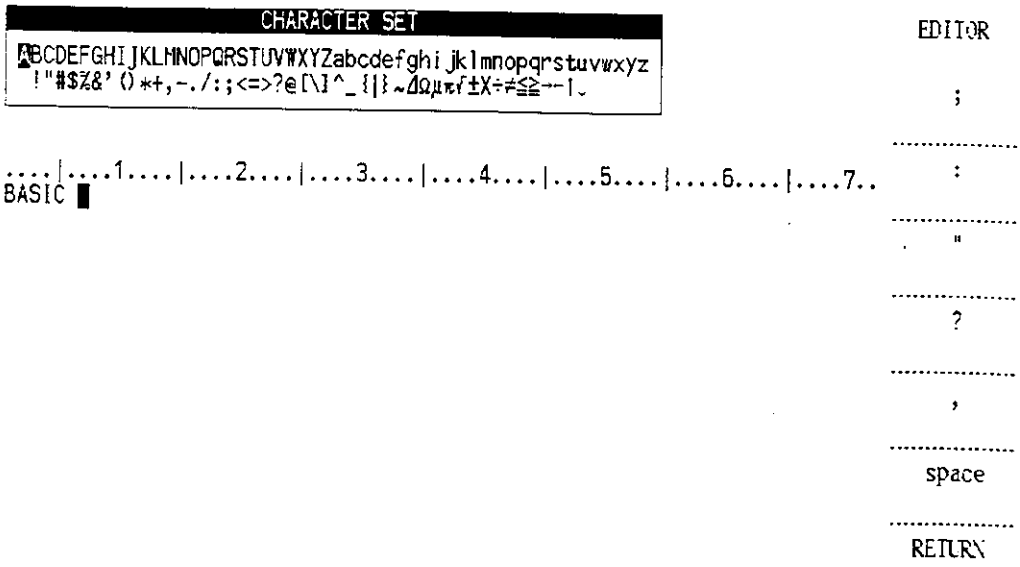


図6-7　文字入力

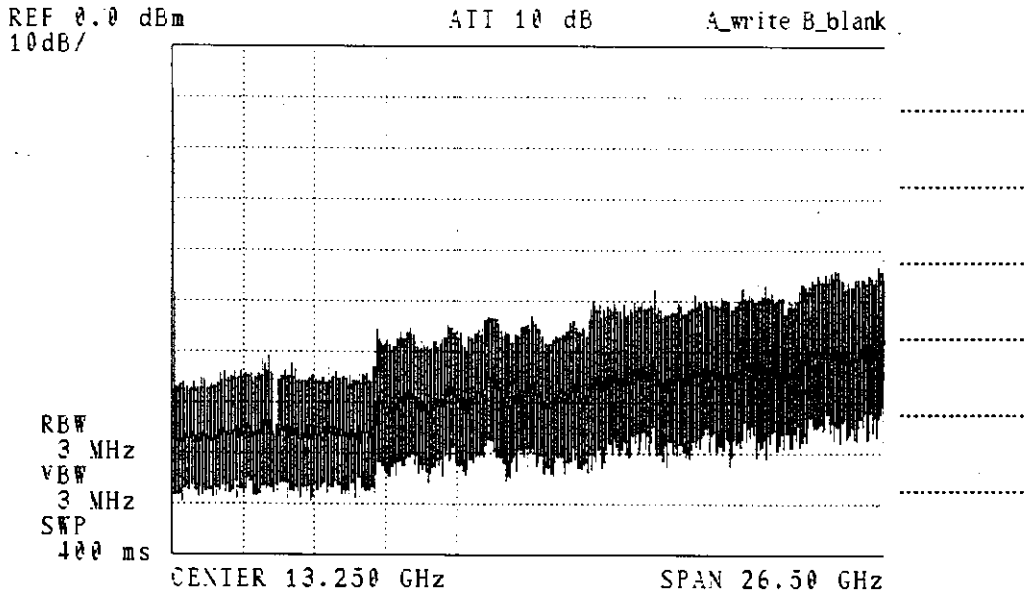
6．7　ヘルプメニュー

計測器に外部端末(VG-920)を接続したときは、割り当てられている機能を一覧表示しますが、計測器のみではサポートしていません。

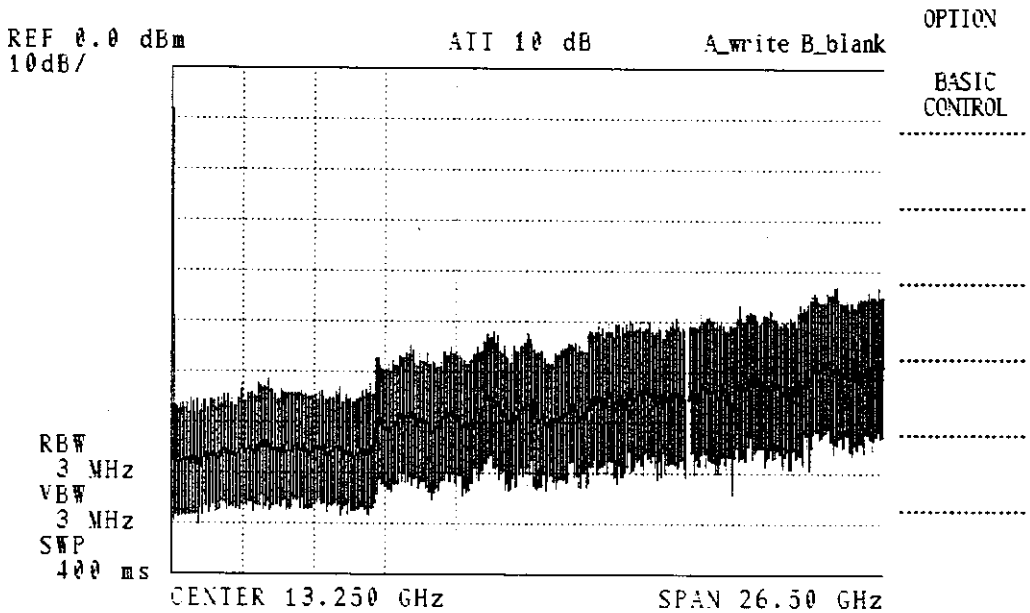
7．　起動（計測器のみの場合）

（1）操作

計測器の電源を入れ、以下に示す操作をすると、初期画面が表示され、ateエディタが起動します。



SHIFT　[6]　と押す



SYSTEM CONTROL　を押す

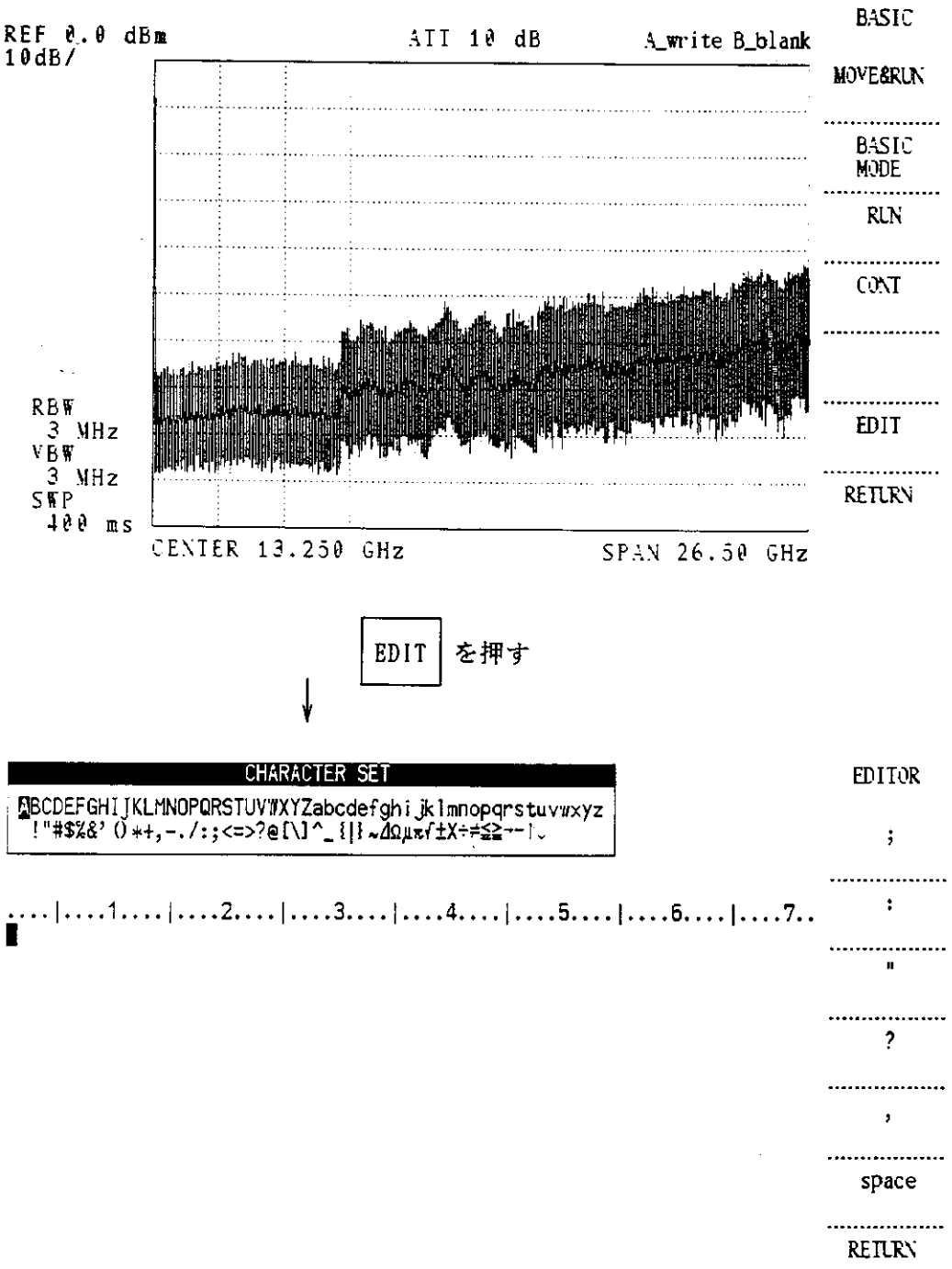


図7-1　ateの起動方法

（２）画面表示の説明

画面の表示機能を説明します。

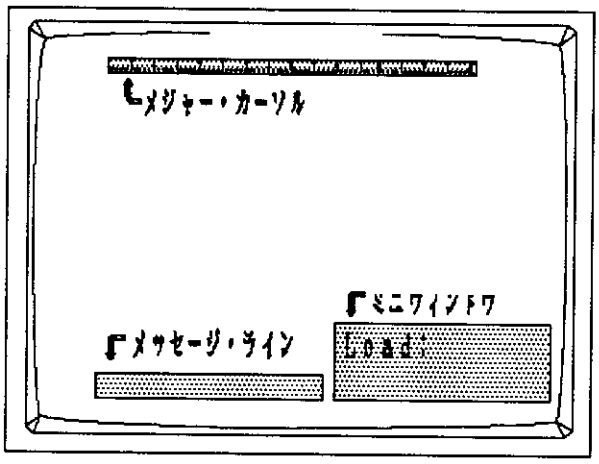


図7-2　ateの動作画面

・メジャーカーソル

編集時にカーソル位置を把握するためのメジャー表示です。

・メッセージライン

コマンドの実行手順、実行結果、エラーメッセージを表示します。

・ミニウィンドウ

コマンドを実行する場合、パラメータが必要なときに使います。バッファ名、ファイル名、文字列などパラメータを要求するコマンドに応じて様々です。

表示されるプロンプトに従ってパラメータを入力します。パラメータの後に必ず＜単位＞キーを入力して下さい。プロンプトの後ろの［　］で囲まれた部分はデフォルトパラメータです。このデフォルトはパラメータを入力しないで＜単位＞キーを押すとパラメータになります。

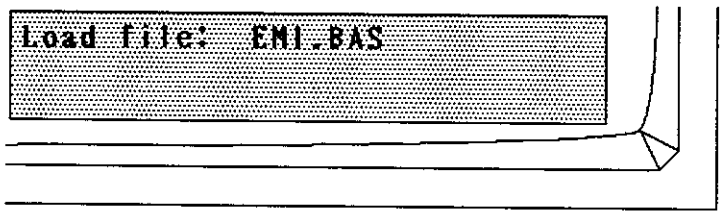


図7-3　ミニウィンドウ

MEMO

8．　機能一覧（計測器のみの場合）

キーボードがないため、エディタの各機能の実行はポップアップメニューによって行います。しかし、一部の機能は、以下に示すように本体の正面パネルに割り当てられています。

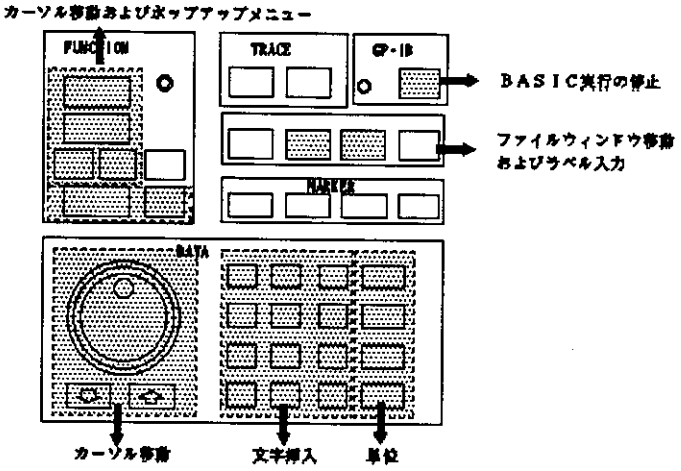


図8-1　正面パネルのキー配置

8．1　カーソル制御

◎（→ノブ）	次の文字に移動
◎（←ノブ）	前の文字に移動
↑	前の行に移動
↓	次の行に移動
STARTキー	ファイルの先頭に移動
STOPキー	ファイルの末尾に移動
SPANキー	前に一画面分移動
CENTERキー	次に一画面分移動

8．2　挿入

ラベル機能	一般文字を挿入
数値キー	数字を挿入
単位キー	改行文字を挿入
SPACEキー	スペースを挿入
ピリオドキー	ピリオドを挿入

8．3　削除

BK SPキー	カーソル前の文字を削除

８．４　コピー、移動、削除（範囲指定）

Set Mark	リージョン処理に必要なマークをカーソル位置にセットする（ポップアップ）
Kill Region	マークからカーソル位置までを削除し、バッファに入れる（ポップアップ）
Copy Region	マークからカーソル位置までをバッファに入れる（ポップアップ）
Yank	<kill>または<copy>によりバッファに入れられた部分をカーソル位置にコピーする　（ポップアップ）

８．５　ウィンドウ

Only	ウィンドウを１つにする　（ポップアップ）
Split	ウィンドウを２つに分割する　（ポップアップ）
Next	カーソルを次のウィンドウに移動（ポップアップ）
redisplay	画面をクリアして再表示する　（ポップアップ）
COUPLEキー	画面をクリアして再表示する

８．６　ファイル

Load	メモリカードから指定したファイルをロードする（ポップアップ）
Save	メモリカードにファイルをセーブする（ポップアップ）
Write	メモリカードに指定したファイル名でセーブする（ポップアップ）
RECALLキー	ファイル名一覧ウィンドウへカーソルを移動する

８．７　検索

Forward search	前方に文字列を検索（ポップアップ）
Backward search	後方に文字列を検索（ポップアップ）

８．８　置換

Query replace	文字列を置き換え　（ポップアップ）

８．９　BASICモード

Line no.	自動的な行番号挿入を設定（ポップアップ）
Renumbering	行番号の再定義　（ポップアップ）
MOVE and RUN	プログラムをBASICバッファに転送後、実行する（ポップアップ）
BASIC mode	BASICモードへ移行する　（ポップアップ）
RUN	すでに転送済みのプログラムを実行する（ポップアップ）
CONT	<LOCAL>キーで中断したプログラムを続行する（ポップアップ）

MOVE & RUNキー	プログラムをBASICバッファに転送後、実行する
BASIC MODEキー	BASICモードへ移行する
RUNキー	すでに転送済みのプログラムを実行する
CONTキー	<LOCAL>キーで中断したプログラムを続行する

８．１０　ポップアップメニュー

REF LEVELキー	ポップアップメニューを表示する

８．１１　キャンセル

MENUキー	ポップアップメニューを中止する

８．１２　エディタの再起動

SCRATCH	現エディタを終了し、バッファを初期化して再起動する（ポップアップ）

MEMO

9．　機能解説（計測器のみの場合）

9．1　カーソル制御

カーソルは編集位置を示すものであり、編集はカーソルを基準にして行われます。

カーソル制御とは、画面上でカーソルを特定の場所に移動することをいいます。まず、1文字づつ上下左右に移動する機能があります。これらの機能はそれぞれカーソルキーに割り当てられており、現在の場所から新しい場所へカーソルを移動します。

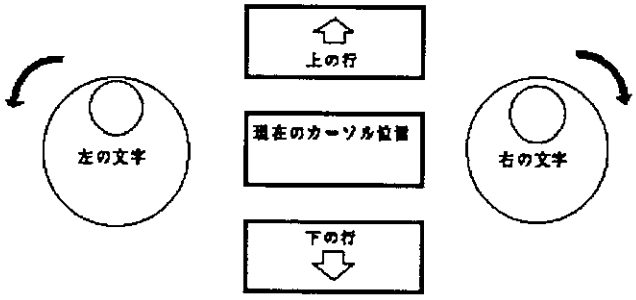


図9-1　カーソルキー

これらは基本的なカーソル移動機能をもち、頻繁に使います。
画面の上限や下限を超えて移動しようとすると、その方向にあるテキストがスクロールし、カーソルは常に画面内に存在します。

この他に正面パネルに割り当てられている機能があります。

STARTキー	ファイルの先頭に移動する
STOPキー	ファイルの末尾に移動する
CENTERキー	次のページに進む
SPANキー	前のページに戻る

図9-2　カーソル移動に関する正面パネル

9．2　挿入

ateエディタでは、常にテキストの挿入ができる状態となっているので、ラベル機能と同様に文字を入力すると、すべて挿入されます。文字は一般文字と制御文字の2つに分けられます。一般文字とは'A','1'など目に見える文字を言い、制御文字とは'ESC','CTRL'など目に見えない文字を言います。

一行が画面の幅以上に長くなると、右端に'$'記号を表示して、その行が継続されていることを示します。

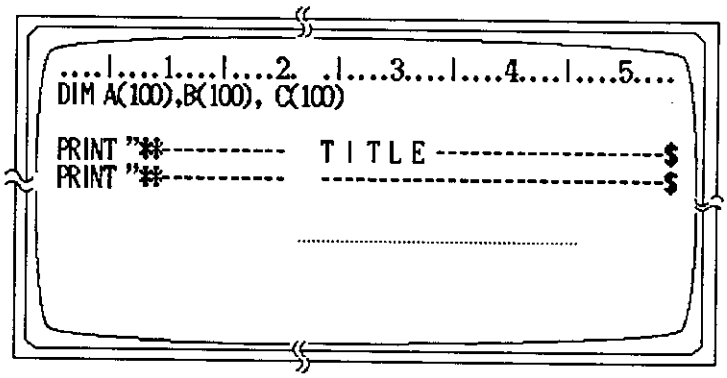


図9-3　文字の継続を表す

行の終わりは正面パネルの<単位>キーを入力すると、改行文字で表わされます。

ラベル機能	一般文字を挿入
数値キー	数字を挿入
単位キー	改行文字を挿入
SPACEキー	スペースを挿入
ピリオドキー	ピリオドを挿入

図9-4　挿入に関するキーパッド

注意

通常、正面パネルはateエディタの機能が割当てられているので、文字の挿入はできません。端末のキーボードと同様に文字を挿入するには、<SHIFT>キーを入力してから、ラベル機能または数値キーなどで文字を挿入します。

９．３　削除

削除には、一文字削除、一行削除、範囲指定削除があります。ここでは一文字削除を説明します。その他の削除方法は次項で説明します。

９．３．１　一文字削除

一文字削除とは、カーソル直前の文字を削除することを言います。

カーソル直前の文字を削除するには、正面パネルの<BK SP>キーを入力します。

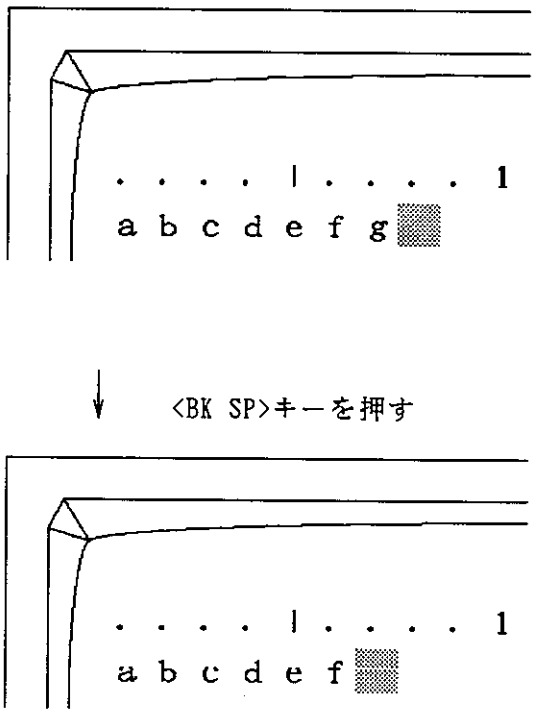


図9-5　カーソル直前の文字を削除

BK SP	カーソル直前の文字を削除

図9-6　削除に関する正面パネル

９．４　コピー、移動、削除（範囲指定）

コピー、移動、削除をするために範囲を指定して、テキストを保存します。

９．４．１　テキストの削除

（１）マークセット

削除、移動をするとき、削除する範囲の先頭位置にカーソルを移動して、ポップアップメニューの"Mark set"でマークを設定します。

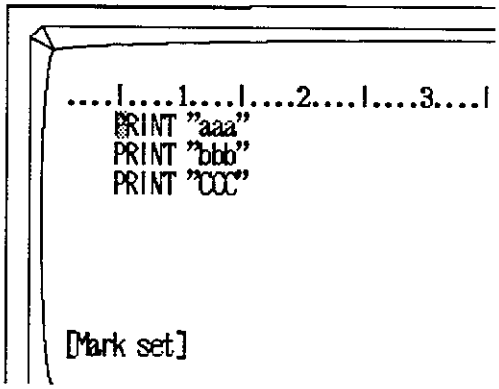


図9-7　マークセット

（２）テキストの削除

削除する範囲の最終位置よりも1文字分先へカーソルを移動し、ポップアップメニューの"Kill region"を入力します。すると、(1)でマークをセットした位置から現在のカーソルの手前までのテキストを削除します。

削除したテキストは、内部バッファに保存されているので、後述のテキストの移動、復旧、コピーなどは(1)，(2)の操作をした後、実行します。

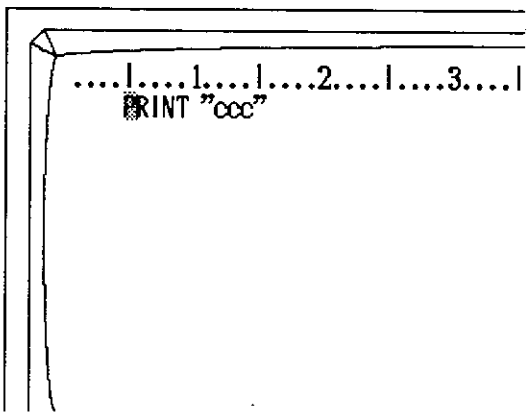


図9-8　テキストの保存（削除）

９．４．２　テキストの復旧、移動

削除したテキストを元に戻すには、現在のカーソル位置でポップアップメニューの"Yank"を入力します。テキストを移動する場合は、移動したい位置へカーソルを移動し、同様にポップアップメニューの"Yank"を入力します。

このようにポップアップメニューの"Yank"は、保存したテキストを現在のカーソル位置に挿入する機能です。

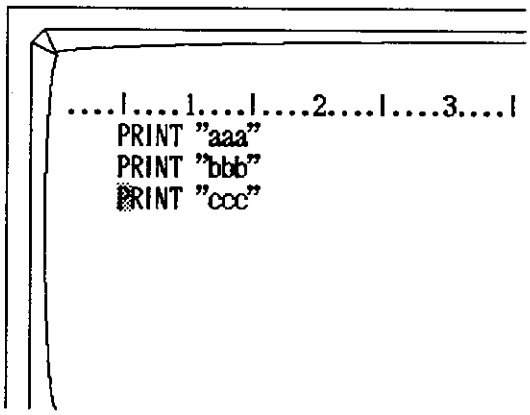


図9-9　テキストの復旧

９．４．３　テキストのコピー

テキストをコピーするには、範囲の先頭位置を削除する場合と同様に設定し、最終位置にカーソルを移動し、ポップアップメニューの"Copy region"を入力します。すると範囲内は何も変わったように見えませんが、指定したテキストが内部バッファにセーブされているので、コピー位置にカーソルを移動し、ポップアップメニューの"Yank"を入力します。

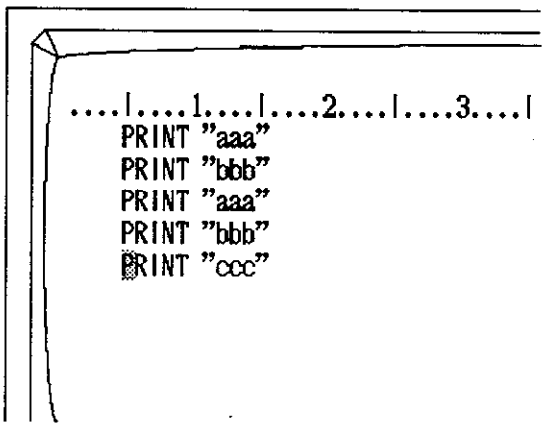


図9-10　テキストのコピー

ポップアップメニューの"Yank"によって復旧されるテキストは、内部バッファ内の最新のテキストになります。

Set mark	範囲指定に必要なマークをカーソル位置にセットする
Kill region	マークからカーソル位置までを削除し、テキストを保存する
Copy region	マークからカーソル位置までのテキストを保存する
Yank	保存したテキストをカーソル位置に挿入する

図9-11　リージョン処理に関するポップアップメニュー

９．５　ウィンドウ

通常のウィンドウはオーバーラッピング型とタイル型に分けられますが、ateエディタのウィンドウは画面を上下に分割するタイル型になります。

通常のウィンドウは1つですが、ateエディタは2つのウィンドウを持つことができ、それぞれに対してテキストを表示します。例えば、画面内に表示しきれない2つの部分を同時に参照できます。

注）オープンできるファイルは1つに限られているため、2つのウィンドウにそれぞれ別のファイルをオープンできません。

９．５．１　ウィンドウの分割

ウィンドウを2つに分割するには、ポップアップメニューの"Split"を入力します。

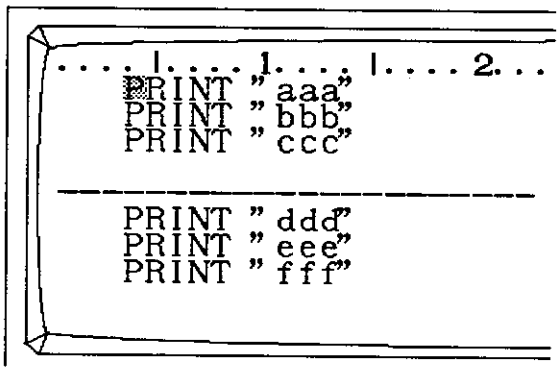


図9-12　上下に分割されたウィンドウ

９．５．２　ウィンドウの初期化

上下に分割されたウィンドウを1つに戻すには、ポップアップメニューの"Next"を入力して、残したいウィンドウにカーソルを移動します。次に、ポップアップメニューの"Only"を入力すると、ウィンドウは1つに戻ります。

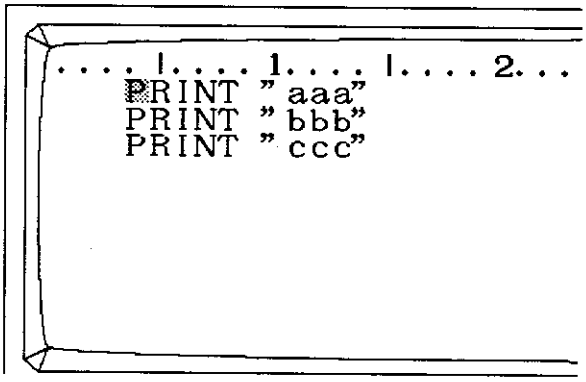


図9-13　ウィンドウを１つにする

９．５．３　画面の再表示

編集中のテキストを再表示したいときは、正面パネルの<COUPLE>キーまたはポップアップメニューの"Redisplay"を入力して、テキストを再表示します。

Only	他のウィンドウを消去して、カーソルのあるウィンドウ1つにする
Split	ウィンドウを上下に分割し、カーソルを上のウィンドウに置く
Next	カーソルを次のウィンドウに移動する
Redisplay	画面をクリアして、テキストの再表示を行う

図9-14　ウィンドウに関するポップアップメニュー

COUPLEキー	画面をクリアして、テキストの再表示を行う

図9-15　ウィンドウに関する正面パネル

9．6　ファイル

ateエディタでは、テキストをファイルという単位として、本体のメモリ・カードへのセーブ、メモリ・カードからのロードができます。編集したテキストは必ずメモリ・カードへセーブするようにします。

注）セーブしないままエディタを終了すると、編集したテキストは失われます。

ここでは、ファイルのセーブ／ロードを説明します。

9．6．1　ファイルのセーブ

編集したテキストをメモリ・カードにセーブするには、ポップアップメニューの"Write"を入力します。ここでセーブするファイルに名前をつけます。ファイル名は最大10文字です。
セーブが終了すると、書き込んだ行数をメッセージラインに表示します。

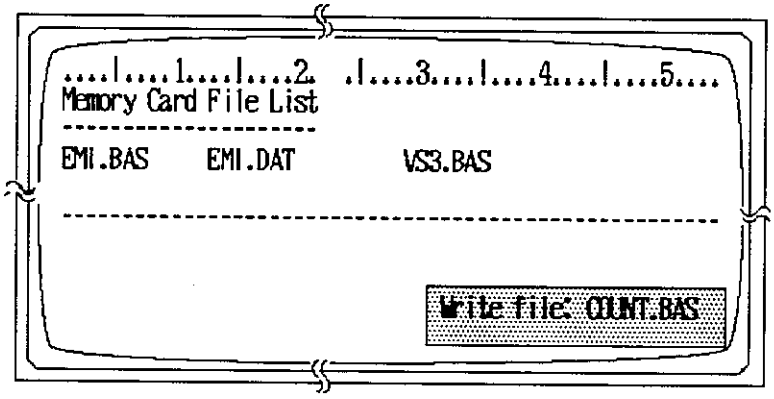


図9-16　テキストのセーブ（ファイル名を入力）

９．６．２　ファイルのロード

セーブされているファイルをロードするには、ポップアップメニューの"Load"を入力します。すると、画面上に現在メモリ・カードにセーブされているファイルが一覧表示され、どのファイルをロードするかを尋ねてきます。ここでファイル名を入力するか、または一覧表示されているウィンドウへ正面パネルの<RECALL>キーを押してカーソルを移動させ、その後、カーソルキーまたはデータ・ノブにてファイル名を選択し、最後に<単位>キーを入力します。

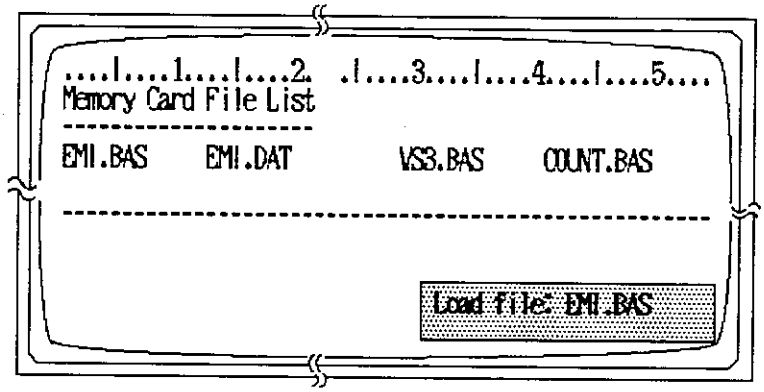


図9-17　ファイル名の入力

```
....|....1....|....2....|....3....|
INTEGER I,S
OUTPUT 31;"IP VS1"
OUTPUT 31;"CF30MZ CLN CPN SW200MS"
FOR I=55 TO 2 STEP -1
  OUTPUT 31;"SP",I,"MZ"
%SWP
  S=SPOLL(31)
  IF (S BAND 4)<>4 THEN GOTO %SWP
```

図9-18　ファイルのロード

メモリ・カードからファイルをロードしている最中は、本体のメモリ・カード挿入口上部のLEDが点灯します。ロードが終了すると、画面上にテキストが表示され、メッセージラインに読み込んだ行数が表示されます。

このとき、行番号がついているものは自動行番号挿入(AUTO)機能が設定されます。

また、新しいファイルを作成するときは、そのファイル名を入力します。

メモリ・カードが挿入されていないときは、ブザー音またはエラーメッセージが表示されます。

9．6．3　ファイルの更新

同じファイル名でセーブする場合には、ポップアップメニューの"Save"を入力します。すると、セーブするかどうかを尋ねてきます。テキストに何の変更もない場合は"No change"というメッセージを表示します。

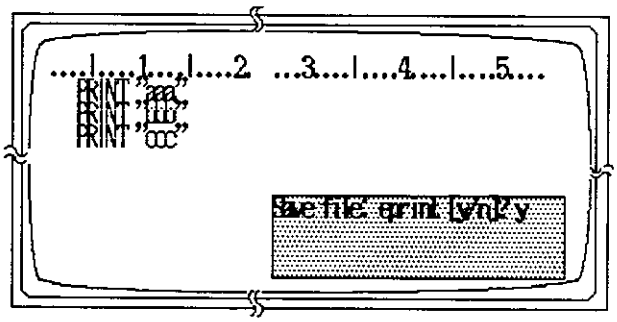


図9-19　テキストの更新（同一ファイル名）

〈y〉〈単位〉キーと入力すると、テキストはファイルに書き出されます。セーブが終了すると、そのファイルの行数がメッセージラインに表示されます。
〈n〉〈単位〉キーと入力すると、テキストのセーブは実行しません。

Load	メモリ・カードから指定したファイルをロードする
Save	メモリ・カードに同一ファイル名でセーブする
Write	メモリ・カードに指定したファイル名でセーブする

図9-20　ファイルに関するポップアップメニュー

RECALLキー	ファイル名一覧ウィンドウへカーソルを移動する

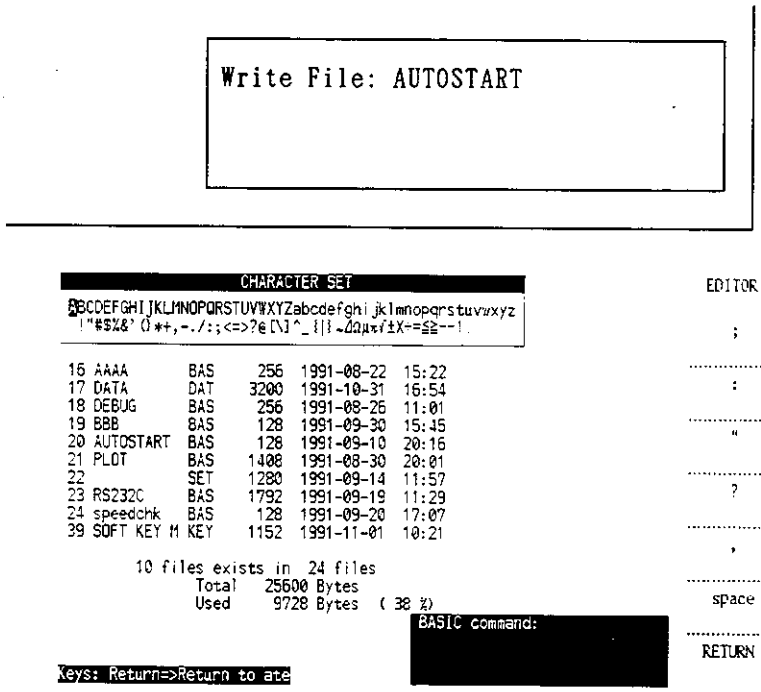
図9-21　ファイルに関する正面パネル

９．６．４　プログラム・オート・スタート機能

電源オン時にプログラムを自動的にメモリ・カード（読み込み）、RUN（起動）させることができます。

作成したプログラムをAUTOSTARTというファイル名でメモリカードに登録（SAVE）します。

ポップアップメニューの［Write］にてファイル名AUTOSTARTでメモリカードに登録します。



次回電源オン時からファイル名"AUTOSTART"を探し、AUTOSTARTが存在したら、そのファイルをロードして起動します。

注意

AUTOSTART　のファイル名は必ず大文字で登録して下さい。
AUTOSTART　はメモリーカード１枚につき、１ファイルしか存在できません。
AUTOSTART　の起動は電源オン後、１回のみ有効です。

9．7　検索

文字列をテキスト内で前方または後方に向かって探す機能です。

カーソル位置以降を検索するには、ポップアップメニューの"Forward search"を入力します。カーソル位置以前を検索するには、ポップアップメニューの"Backward search"を入力します。両者は検索の方向が異なるだけですべて同じ働きをします。

以下にカーソル位置以降の文字列の検索手順を示します。

ポップアップメニューの "Forward search"を入力します。画面にミニウィンドウが表示され、検索する文字列を尋ねてきたら、探したい文字列を入力します。

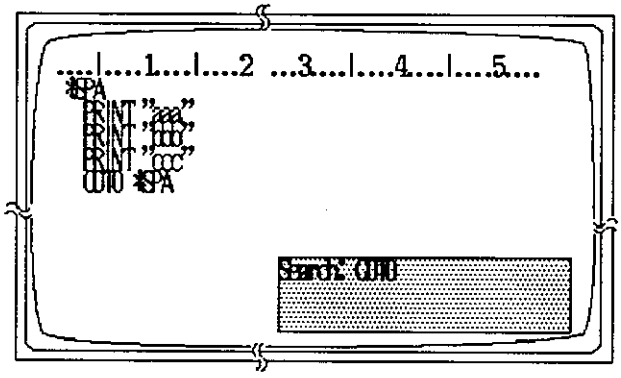


図9-22　カーソル位置以降の文字列検索

検索した文字列の先頭にカーソルが移動し、検索は終了します。同じ文字列を続けて検索したいときは、同様の操作を行いますが、文字列の入力は省略できます。

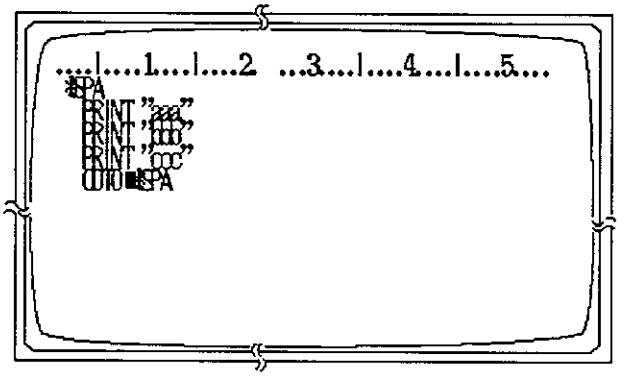


図9-23　実行結果

検索できないときは、"Not found"というメッセージが表示されます。

Forward search	カーソル位置以降に向かって検索する
Backward search	カーソル位置以前に向かって検索する

図9-24　検索に関するポップアップメニュー

９．８　置換

カーソル位置以降のテキスト中の任意の文字列を変更する機能です。

それでは、文字列を置き換えてみます。ポップアップメニューの"Query replace"を入力します。画面にミニウィンドウが表示され、置換したい文字列を尋ねてきたら、対象となる文字列を入力します。

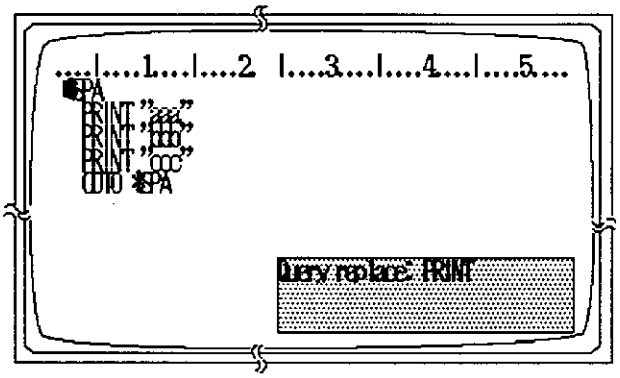


図9-25　置換対象文字列の入力

次に新たに変更する文字列を尋ねてきたら、任意の文字列を入力します。

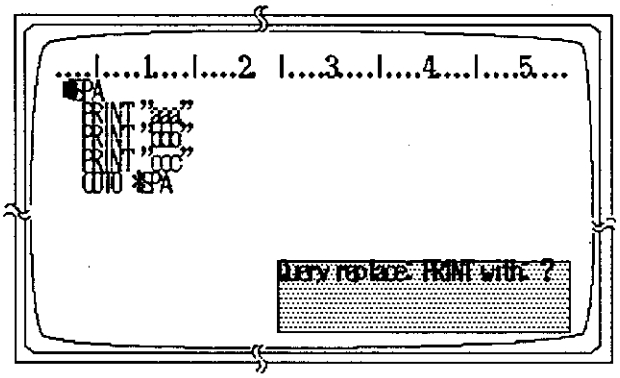


図9-26　置換文字列の入力

該当する文字列を検索すると、カーソルがその文字列の先頭に移動し、置換するかどうか待ち状態になります。ここで置換するときにはソフトキーの<SPACE>キーを入力します。置換しないときには正面パネルの<BK SP>キーを入力します。また、置換して終了するときには正面パネルの<.>キーを入力します。置換しないで終了するときには正面パネルの<MENU>キーを入力します。

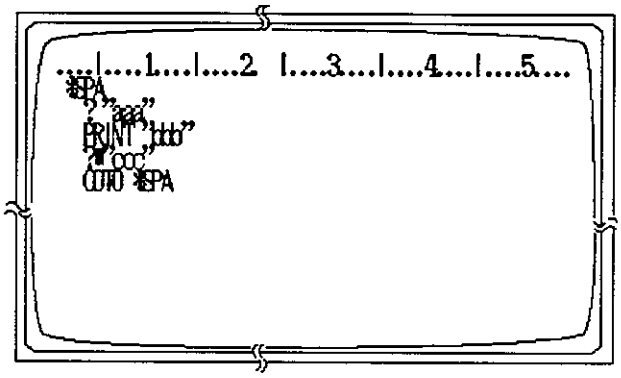


図9-27　実行結果

Query replace　文字列を置き換える

図9-28　置換に関するポップアップメニュー

SPACEキー	置換する
BK SPキー	置換しない
ピリオドキー	置換して、終了する
MENUキー	終了する

図9-29　置換に関する正面パネル

９．９　ＢＡＳＩＣ

ateエディタは、計測器上でのBASICプログラミングを行うために開発されたポータブル・エディタです。よって編集しながらBASICの実行、デバッグが容易にできます。その他にBASICモードに移行してから様々なBASICコマンドが実行できます。

ここではエディタ上から実行できるBASICコマンドを説明します。

９．９．１　行番号の設定

今までBASICプログラムを作成するときには行番号を使いました。ateエディタでは行番号の代わりにラベルを用いて、プログラムを分かりやすくします。

```
A=1:B=0
*LOOP
        B=B+A^2
        IF B>10000000 THEN GOTO *ENDLOOP
        PRINT B
        GOTO *LOOP
*ENDLOOP
STOP
```

図9-30　ラベルを用いたプログラム

また、自動行番号挿入(AUTO)機能により行番号をつけながら編集もできます。ポップアップメニューの"Line no."を入力すると、ミニウィンドウが表示され、開始番号と間隔を指定します。これで編集時に<単位>キーを入力すると、行番号が自動的に出力されます。

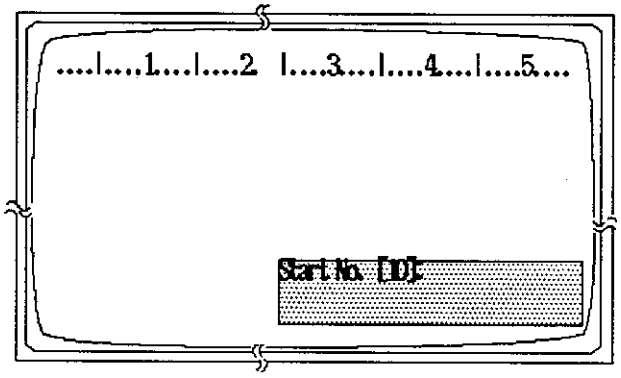


図9-31　開始番号の指定

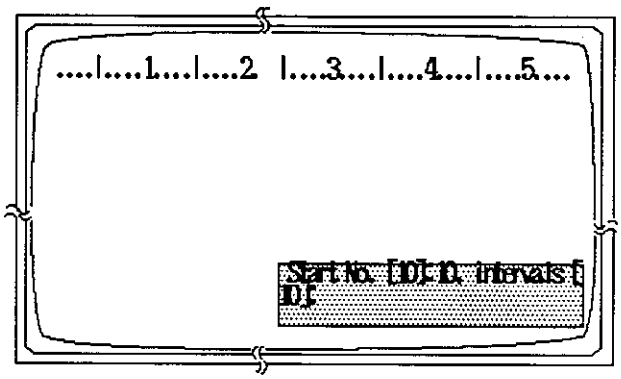


図9-32　行番号間隔の指定

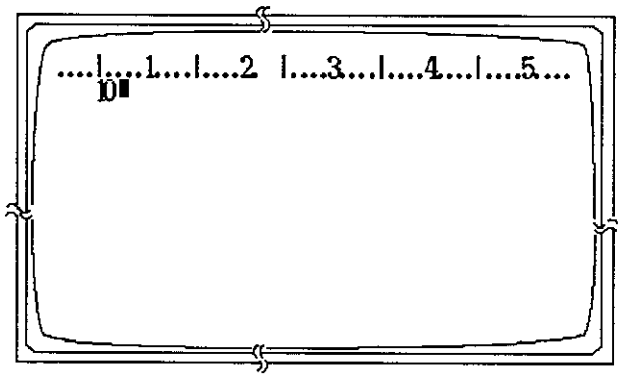


図9-33　自動行番号機能

また、編集途中の場合でも、この機能により行番号を追加することができます。
行間での挿入を行ったときは、上の行番号にプラス１した数が行番号となります。

注）一度この機能を実行すると、エディタの初期化（9.13節）をしない限り継続されます。

自動行番号挿入機能と同様、行番号のリナンバリング機能があります。これは行番号を一定の指定にしたがって付け直すものです。ポップアップメニューの"Renumbering"を入力すると、自動行番号挿入機能と同様に、開始番号と間隔を指定されます。

９．９．２　ＢＡＳＩＣの実行

プログラムを実行する前に、ateエディタとBASICインタプリタの関係を説明します。
プログラムの編集はateエディタで制御し、完成したプログラムはateエディタの内部バッファに格納されます.これとは別にBASICインタプリタも内部バッファを持っていて、実行するためにはこのバッファにBASICプログラムが存在しなければなりません。

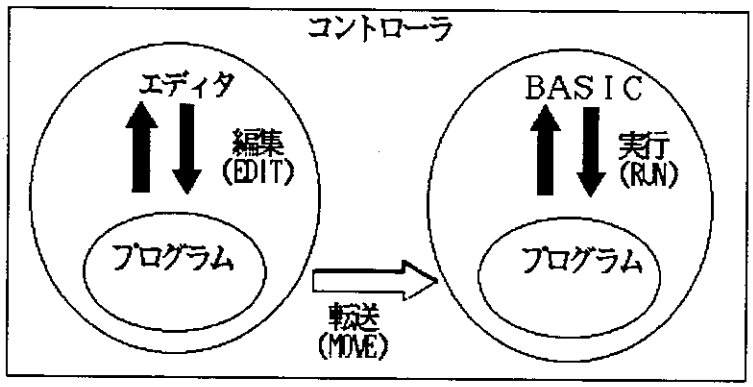


図9-34　エディタとインタプリタとの関係

よってBASICプログラムを実行する前に、エディタからインタプリタにプログラムを転送します。ソフトキーの<MOVE & RUN>* キー、またはポップアップメニューの"MOVE and RUN"を入力すると、エディタからプログラムが転送され、実行します。このとき、前回実行したプログラムは消去されます。また、実行中にエラーが起こると、エディタに戻り、カーソルがエラー行に移動します。

すでに転送済みのプログラムを実行するときは、ソフトキーの<RUN>* キー、またはポップアップメニューの"RUN"を入力します。プログラムを転送する前に、上記の手順で実行したときは、"Program is not exist"というエラーメッセージが表示されます。

* MOVE&RUN, RUNは、 SHIFT 6 BASIC CONTROL と押すと、画面右側に表示されます。

９．９．３　ＢＡＳＩＣの停止

実行中のBASICを停止するには、<LOCAL>キーを入力します。

９．９．４　ＢＡＳＩＣモード

BASICモードとは、BASICに対してのコマンドを実行できる環境を言います。ソフトキーの<BASIC MODE>キー、またはポップアップメニューの"BASIC mode"を入力すると、ミニウィンドウが表示されます。

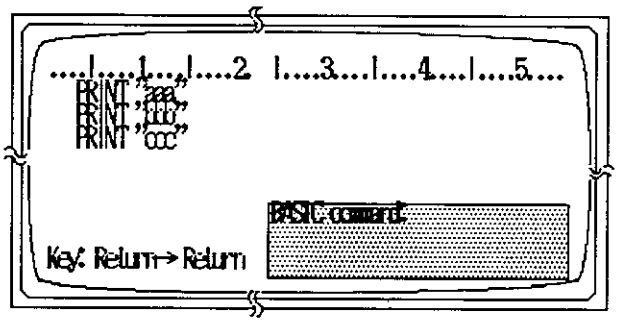


図9-35　BASICモード

実行できるコマンドは、RUN, LIST, CONTなどです。（コマンドの説明は［第2部　システム・コントローラ］を参照）

注）プログラムの内容を変更する場合は必ずエディタに戻ってから行います。

９．９．５　ＢＡＳＩＣの継続

<LOCAL>キーまたはPAUSE命令で一時停止したプログラムを続けて実行するときは、ソフトキーの<CONT>キー、またはポップアップメニューの"CONT"を入力します。これはBASICコマンドの"CONT"と同じです。停止した次の文から実行し、実行するプログラムがない場合は、"Program cannot be continued"というエラーメッセージが表示されます。

Line no.	自動的に行番号を挿入する
Renumbering	行番号のリナンバリング
MOVE and RUN	プログラムを転送後、実行する
BASIC mode	BASICモードへ移行する
RUN	すでに転送済みのプログラムを実行する
CONT	<LOCAL>キーまたはPAUSE命令で中断したプログラムを続行する

図9-36　BASICに関するポップアップメニュー

MOVE & RUNキー	プログラムを転送後、実行する
BASIC MODEキー	BASICモードへ移行する
RUNキー	すでに転送済みのプログラムを実行する
CONTキー	<LOCAL>キーまたはPAUSE命令で中断したプログラムを続行する

図9-37　BASICに関する正面パネル

9．10　ヘルプ

エディタの機能とキーボード（VG-920）の関係は画面上で確認できますが、正面パネルについてのヘルプメニューはサポートしていません。

9．11　ポップアップメニュー

ateエディタの機能は、カーソル移動機能を除いて、ほとんどがポップアップメニューによって実行できます。

カーソルキー以外のエディタ機能は、ほとんどポップアップメニュー内に表示され、その中から選択し、実行します。

正面パネルの<REF LEVEL>キーを入力すると、画面上部にポップアップメニューが表示されます。

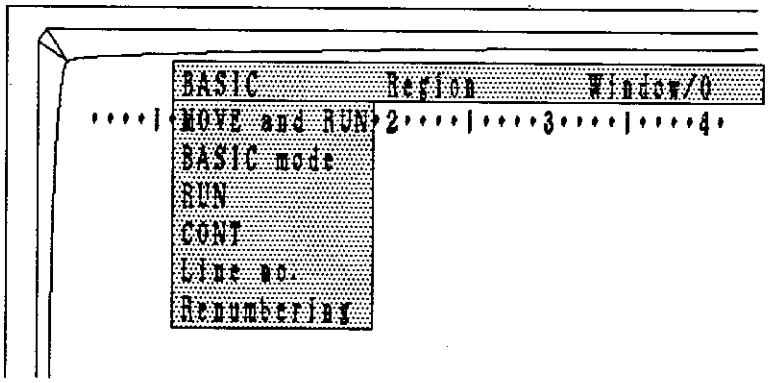


図9-38　ポップアップメニュー

BASIC	Region	Window/Other	File	Search/Replace
MOVE and RUN	Set mark	Only	Load	Forward search
BASIC mode	Kill region	Split	Save	Backward search
RUN	Copy region	Next	Write	Query replace
CONT	Yank	Redisplay		
Line No.		Help		
Renumbering		SCRATCH		

図9-39　ポップアップメニュー一覧

各項目間の移動は正面パネルの<REF LEVEL>キーを続けて入力すると、トグルとなって移動します。

該当する項目のメニューが表示されたら、正面パネルの<↑>,<↓>キーで実行する機能を選択します。

REF LEVELキー	ポップアップメニューの表示
MENUキー	ポップアップメニューの中止

図9-40　ポップアップメニューに関する正面パネル

メニュー	説明	参照先
MOVE and RUN	BASICプログラムを転送、実行する	9.9.2項
BASIC mode	BASICコマンドを実行する	9.9.4項
RUN	すでに転送済みのプログラムを実行する	9.9.2項
CONT	<LOCAL>キーにより停止したプログラムを継続する	9.9.5項
Line no.	行番号を設定する	9.9.1項
Renumbering	行番号を再設定する	9.9.1項
Set mark	テキストの削除、コピーのための先頭位置をセットする	9.4.1項
Kill region	テキストのマークからカーソル位置までを削除する	9.4.1項
Copy region	テキストのマークからカーソル位置までを保存する	9.4.2項
Yank	保存されたテキストをコピー（復旧）する	9.4.3項
Only	ウィンドウを1つにする	9.5.2項
Split	ウィンドウを上下に分割する	9.5.1項
Next	他のウィンドウにカーソルを移動する	9.5 節
Redisplay	テキストの再表示を行なう	9.5.3項
Help	（サポートしていません）	9.10節
SCRATCH	エディタを初期化する	9.13節
Load	メモリ・カードからファイルをロードする	9.7.2項
Save	ファイルを更新する	9.7.3項
Write	メモリ・カードへファイルをセーブする	9.7.1項
Forward search	カーソル位置から後方へ向かって文字列を検索する	9.7 節
Backward search	カーソル位置から前方へ向かって文字列を検索する	9.7 節
Query replace	文字列の置き換えを行う	9.8 節

図9-41　ポップアップメニューの説明

９．１２　ポップアップメニューの中止

ポップアップメニューを中止するときには、正面パネルの<MENU>キーを入力します。

注）BASICバッファへテキスト転送中や、ファイルのセーブ/ロード中は中止できません。

MENUキー	エディタ機能の中止

図9-42　キャンセルに関する正面パネル

９．１３　エディタの初期化

現在起動しているエディタを初期化します。ポップアップメニューの"SCRATCH"を入力すると、ミニウィンドウが表示され、編集中のテキストをセーブするかどうかを尋ねてきます。（テキストを変更した場合のみ）

それに返答すると、エディタを初期化するかどうかを尋ねてきます。

〈y〉 〈単位〉キーと入力すると、エディタは初期化され、〈n〉 〈単位〉キーと入力すると初期化は中止し、元のエディタ画面に戻ります。

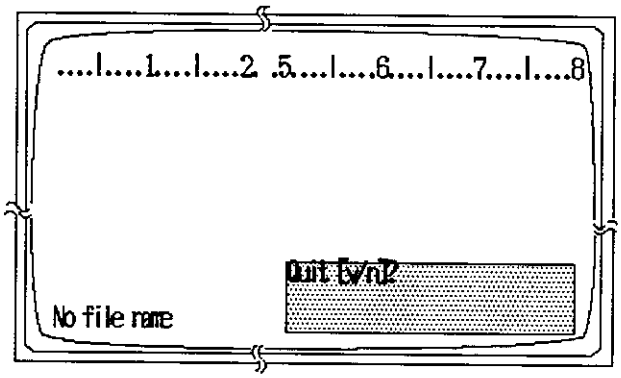


図9-43　エディタの終了ミニウィンドウ

↓　〈y〉 〈単位〉キーを入力します。

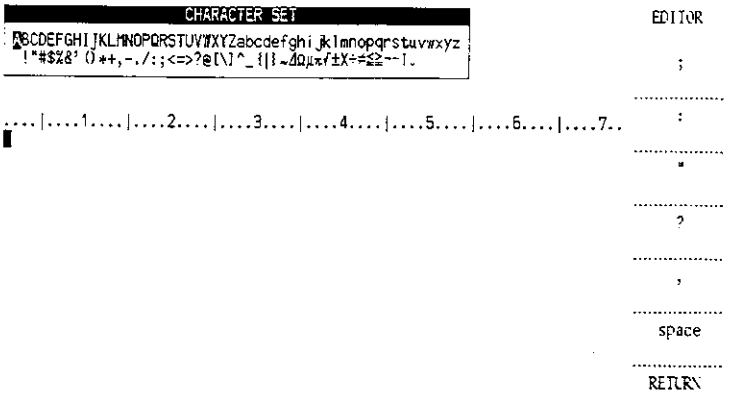


図9-44　エディタの初期化

SCRATCH	ateエディタを初期化する。

図9-45　エディタの初期化に関するポップアップメニュー

第2部　システム・コントローラ

第２部の目次

1．　オプション１５（ＢＡＳＩＣ　ＧＰＩＢコントローラ）

1.1　概要

オプション１５のＢＡＳＩＣ言語は、汎用のＢＡＳＩＣコマンドのほか、ＧＰＩＢ制御用コマンドを備えており、小規模ＧＰＩＢシステムを構築でき、測定専用ビルトイン関数を使用すると測定を簡単に、かつ高速に処理できます。

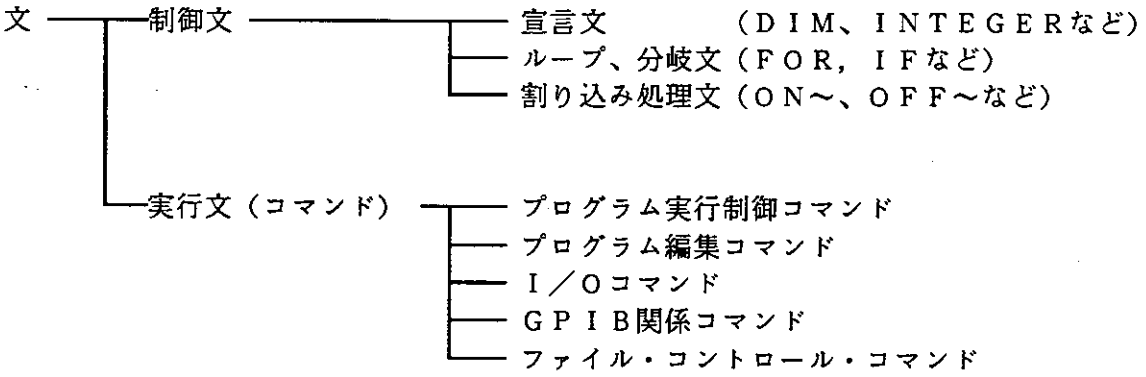
注）オプション１５コントローラのＧＰＩＢアドレスは、ＢＡＳＩＣのＣＯＮＴＲＯＬ命令で設定します。

1.2 ＢＡＳＩＣプログラミング

1.2.1 プログラム構造

文はＢＡＳＩＣが処理する最小単位です。　その文の集まりがＢＡＳＩＣプログラムになります。

文は大きく分けて制御文と実行文（コマンド）で構成されています。



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。

このＢＡＳＩＣで予め、その意味およびその用途を定めている言葉をキーワードと言います。変数名などと同じ名前では使えません。

1.2.2 キーワード

【キーワード一覧】

AND	AS	ASCII	BAND	BINARY	BNOT
BOR	BREAK	BUZZER	BXOR	CASE	CAT
CLEAR	CLOSE	CLS	CMD	CONT	CONTINUE
CONTROL	CSR	CURSOR	DATA	DELIMITER	DIM
DISABLE	ELSE	ENABLE	END	ENT	ENTER
ERROR	FOR	GLIST	GLISTN	GOSUB	GOTO
GPRINT	IF	INIT	INITIALIZE	INP	INPUT
INTEGER	INTERFACE	INTR	ISRQ	KEY	LISTEN
LISTN	LLIST	LLISTN	LOCAL	LOCKOUT	LPRINT
NEXT	NOT	OFF	ON	OPEN	OR
OUT	OUTPUT	PAUSE	PRF	PRINT	PRINTER
PRINTF	PURGE	READ	REM	REMOTE	RENAME
REQUEST	RESTORE	RETURN	RUN	SCRATCH	SELECT
SEND	SPRINTF	SRQ	STEP	STOP	TALK
TEXT	THEN	TO	TRIGGER	UNL	UNT
USE	USING	WAIT	XOR		
実行できないキーワード群					
APPEND	BASIC	CHKDSK	COPY	COPYFILES	COUNT
DEL	DSTAT	ENTERF	FORMAT	LABEL	LOAD
MERGE	NEWVERSION	REN	SAVE	SYSTEM	TIME
POKE	DUMP				

注）実行できないキーワードとは、キーワードとしては登録してあるが命令の実行はできないものです。

1.2.3 ショート・ネーム

キーワードの入力時にショートネームが使えます。これは使用頻度が高く、かつ長い名前を選んで決められています。これらもキーワードになります。

ショートネームの出力（ＬＩＳＴ、ＧＬＩＳＴ等）は、ＣＯＮＴＲＯＬコマンドで指定します。（エディタの　［ＢＡＳＩＣ　ｍｏｄｅ］　にて設定）

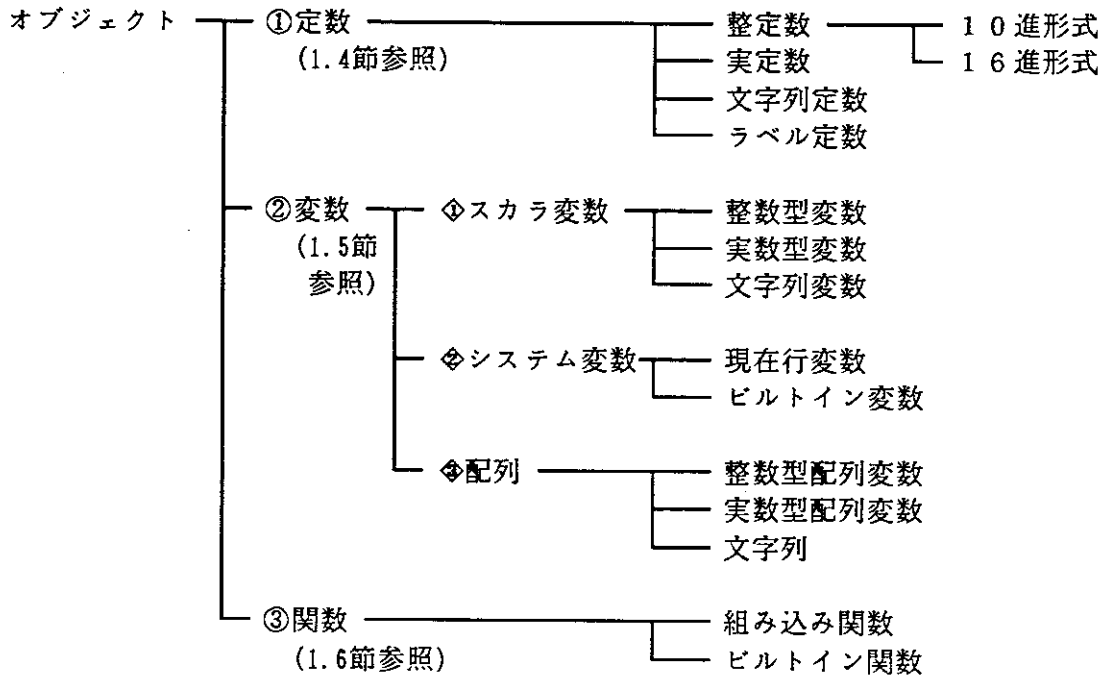
```
CONTROL  3 ; 0      ---  フルネーム出力
         ; 1      ---  ショートネーム出力
```

【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
CURSOR	CSR
ENTER	ENT
INITIALIZE	INIT
INPUT	INP
OUTPUT	OUT
PRINTF	PRF
USING	USE

1.3 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言います。オブジェクトには変数、定数、および関数があり、それぞれにデータ型があります。
データ型には整数型、実数型、および文字列型があります。



1.4 定数

1.4.1 整定数

プログラムで、小数点のない数値は整定数とみなします。
整定数では、１０進、１６進で表現できます。
内部では４バイトにて表現するので、ＢＡＳＩＣでは１０進数で
　－２，１４７，４８３，６４８　～　＋２、１４７、４８３、６４７
までの数値を表現できます。
　先頭に０ｘを付けることにより１６進で表現できます。

１０進表現
　例）
　　Ａ＝１２３
　　ＰＲＩＮＴ　４５６

１６進表現
　例）
　　Ａ＝０ｘ１０ＡＢ
　　ＰＲＩＮＴ　０ｘＦＦＦＦ

1.4.2 実定数

小数点付きや１Ｅ＋２０のような浮動小数点表現の数値は実数とみなします。
内部では８バイトで表現するので、ＢＡＳＩＣでは
　約　－１Ｅ＋３０８　～　１Ｅ３０８
まで表現でき、１５桁の精度を持ちます。
例）
　Ａ＝１２３．０
　Ｂ＝１．２３４５６７８９
　Ｃ＝１．２３Ｅ６－９．８７６５４

注）定数同士の演算

　ＰＲＩＮＴ　３／２　の場合、整数同士の演算とみなして計算するので、値は１．０になります。
　実数で演算させたい場合は、ＰＲＩＮＴ　３．０／２．０　のように両方、またはどちらか一方に小数点を付けて演算させます。

1.4.3　文字列定数

２５５文字以内の文字群の前後をダブル・クォーテーション（”）で囲んだものを文字列定数と言います。
文字列は、”” の空文字列から最大２５５文字まで表現できます。

キーボードに割当られていないコードをプログラムで表現するために、（＼）を使って表現できます。またＡＳＣＩＩコード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。

注）”＼”を使うときは８進数、または下記のエスケープ・シーケンスで指定します。

エスケープシーケンス	８進	１０進	
＼b	０１０	８	バック・スペース
＼t	０１１	９	水平タブ
＼n	０１２	１０	ライン・フィード
＼v	０１３	１１	垂直タブ
＼f	０１４	１２	フォーム・フィード
＼r	０１５	１３	キャリッジ・リターン

例）

```
A$=”ABCDE”
CR$=”＼r”
NL$=”＼n”
B$=””
PRINT ”ABCD＼014”
PRINT ”AB＼007”
PRINT ”ABC＼n”
```

1.4.4　ラベル定数

文番号に代わるもので、ラベル名の先頭に＊（アスタリスク）をつけて宣言します。
ラベル名で使用できる文字は変数と同じですが、ラベルは変数ではないので数値を入力できません。またラベルが指定できる位置は構文的に限られていて、[2.3　コマンドとステートメントの説明] にある＜ラベル＞と書かれている部分です。

1.5 変数

変数名は、英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。

注）変数名は、キーワードと同等では使用できません。

【アルファニューメリック】

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 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 ＿（アンダー・バー）

変数名の最後に＄を付けると文字列変数になります。
変数名の最後に（数字）を付けると配列変数になります。
変数は特にＩＮＴＥＧＥＲ宣言されていなければ実数型になります。

例）

ＶＡＬ		：実数型変数
ＳＴＲＧ＄		：文字列型変数
ＡＲＲＹ１（４）		：実数型配列変数
ＩＮＴＥＧＥＲ	ｃｏｄｅ２	：整数型変数
ＩＮＴＥＧＥＲ	ｗｋ（７）	：整数型配列変数

1.5.1 スカラ変数

・整数型変数
・実数型変数
・文字列型変数

のうち数値型、の変数はＢＡＳＩＣプログラム起動時に０で初期化されます。
従って特定の値に初期化する変数の場合は、プログラム内で明示的に値を入力する必要があります。

各データ型の変数に入力できる値の大きさは、定数で紹介した値と同じです。

文字列変数には、配列がありません。

文字列変数には文字列定数と同様に長さの属性があり、その長さを宣言するには、ＤＩＭ文を使います。

例）

```
ＤＩＭ　　ＳＴＲＧ＄［１００］　　：長さ１００文字の変数宣言
```

宣言をしなくても１８文字分が予め宣言されています。

1.5.2　システム変数

・現在行変数　@

現在実行しているプログラムの行番号を格納しています。
値を入力することはできません。

例）

```
ＬＩＳＴ　@　　：現在実行している行を表示します。
```

・ビルトイン変数

ＢＡＳＩＣの起動時に自動的に登録される変数で、固有の値で初期化されます。また任意の値を入力して値を変更できます。起動時の値に戻すには明示的にその値を入力するか、またはプログラムをＭＯＶＥ ａｎｄ ＲＵＮで初期化する必要があります。

```
ＰＩ　　：３．１４１５９２．．．．
ＥＸＰ　：２．７１８２８１．．．．
```

・エラー番号変数

ＢＡＳＩＣのエラー番号を保持しているシステム変数です。
ＢＡＳＩＣの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。

例）

```
ＰＲＩＮＴ　ＥＲＲＮ
```

エラー番号は内部で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　　；ビンルトイン関数関係
　　　　　　４　　；ＢＡＳＩＣ構文関係
```

1.5.3 配　列

配列の宣言には、ＤＩＭ，ＩＮＴＥＧＥＲ文を使用します。
配列変数を宣言する際、複数の添字をコンマで区切って指定すると、その配列変数は添字の個数に応じた次元数をもつことになります。
（次元数は、最大１０次元まで、ただしメモリ容量が許す限り）

・数値型配列

宣言しないで参照すると、その配列の大きさ、つまり要素数は１０になります。以下のように宣言したものと同じになります。
添字は必ず　１　から始まります。
ＤＩＭ宣言時に指定した添字が最大値となります。ＤＩＭ宣言できる範囲は３２７６７までとなります。

```
ＤＩＭ　　　ＡＢ（１０）
ＩＮＴＥＧＥＲ　ＣＤ（１０）
```

例）

```
ＤＩＭ　　　ＲＬ（３０）　　　：実数型配列変数宣言
ＩＮＴＥＧＥＲ　ＩＴ（１０，２０）　：整数型配列変数宣言
                                （２次元）
```

・文字列

文字列変数に格納される文字数をＤＩＭ文で宣言できます。

例）

```
ＤＩＭ　Ａ＄［１００］　　：１００文字まで入力可能
ＤＩＭ　Ｂ＄［５０］　　：５０文字まで入力可能
```

注）文字列変数の２次元配列はできません。

1.6 関数

1.6.1 組み込み関数

組み込み関数	説　　明
ＥＲＲＭ＄ （エラー番号）	パラメータで指定されたエラーメッセージを返します。 パラメータを０で指定すると、直前に出力されたエラーメッセージを返します。 エラー番号は内部で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 しかし、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。 例） ＰＲＩＮＴ　ＥＲＲＭ＄（ＥＲＲＮ） ：　直前のエラーメッセージを出力します。
ＮＵＭ （文字列式）	文字列式の先頭の１文字のＡＳＣＩＩコードを返します。 例） ＮＵＭ（”ＡＢＣ”）　➡　６５ Ａ＄＝”ＸＹＺ” ＮＵＭ（Ａ＄）　➡　８８
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します。 例） ＣＨＲ＄（６５）　➡　”Ａ” Ａ＝８８ ＣＨＲ＄（Ａ）　➡　”Ｘ”
ＬＥＮ （文字列式）	文字列式の長さを返します。 例） ＬＥＮ（”ＡＤＶＡＮＴＥＳＴ”）　➡　９ Ａ＄＝”ＣＯＲＰ．” ＬＥＮ（Ａ＄）　➡　５
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します。 例） Ａ＄＝”ＡＮ” ＰＯＳ（”ＡＤＶＡＮＴＥＳＴ”，Ａ＄）　➡　４

組み込み関数	説　　明
ＡＢＳ （算術式）	算術式の値の絶対値を返します。 例） ＡＢＳ（－１．２）　　➡　１．２
ＡＴＮ （算術式）	算術式の値に対する逆正接値を返します。 （算術式はラジアン） 例） ＡＴＮ（ＰＩ）　　➡　１．２６２６２．．．
ＣＯＳ （算術式）	算術式の値に対する余弦値を返します。 （算術式はラジアン） 例） ＣＯＳ（ＰＩ）　　➡　－１．０
ＦＲＥ （算術式）	ＢＡＳＩＣのメモリの残り容量（バイト）を返します。 例） ＦＲＥ（０）　　➡　３０１４５８ （電源ＯＮ時：ＰＲＩＮＴ　ＦＲＥ（０） をＢＡＳＩＣ　ｍｏｄｅにて実行）
ＬＯＧ （算術式）	算術式の値に対する自然対数（ｅを底とした対数）を返します。 例） ＬＯＧ（ＥＸＰ）　　➡　１．０
ＳＩＮ （算術式）	算術式の値に対する正弦値を返します。 （算術式はラジアン） 例） ＳＩＮ（ＰＩ）　　➡　０．０
ＳＰＯＬＬ （算術式）	ＧＰＩＢ装置へのシリアル・ポールを行い、ステータス・バイト値を返します。（算術式は０～３１） 例） ＳＰＯＬＬ（２）　：外部接続ＧＰＩＢ装置のアドレス２のシリアル・ポールを行う。 ＳＰＯＬＬ（３１）　：本器の計測部へのシリアル・ポールを行う。
ＳＱＲ （算術式）	算術式の値に対する平方根を返します。 例） ＳＱＲ（２）　　➡　１．４１４２１３５６．．．
ＴＡＮ （算術式）	算術式の値に対する正接値を返します。 （算術式はラジアン） 例） ＴＡＮ（ＰＩ）　　➡　１．０

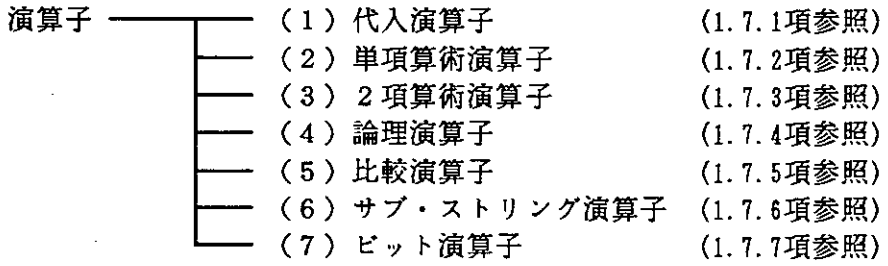
1.6.2　ビルトイン関数（ビルトイン関数の章　参照）　(1/2)

機　　能	No	関　　数
周波数／ポイントを求める	1	F＝FREQ（P）
	2	F＝DFREQ（P1，P2）
	3	P＝POINT（F）
	4	P＝DPOINT（F1，F2）
レベル／ポイントを求める	5	L＝LEVEL（T）
	6	L＝DLEVEL（T1，T2）
	7	T＝LVPOINT（L）
	8	T＝LVDPOINT（L1，L2）
	9	L＝VALUE（P，M）
	10	L＝DVALUE（P1，P2，M）
	11	L＝CVALUE（F，M）
	12	L＝DCVALUE（F1，F2，M）
最大／最小を求める	13	F＝FMAX（P1，P2，M）
	14	F＝FMIN（P1，P2，M）
	15	P＝PMAN（P1，P2，M）
	16	P＝PMIN（P1，P2，M）
	17	L＝MAX（P1，P2，M）
	18	L＝MIN（P1，P2，M）
バンド幅を求める	19	F＝BND（P，X，M）
	20	F＝BNDL（P，X，M）
	21	F＝BNDH（P，X，M）
	22	F＝CBND（F，X，M）
	23	F＝CBNDL（F，X，M）
	24	F＝CBNDH（F，X，M）
極大／極小を求める	25	N＝NRPLH（P1，P2，Dx，Dy，M）
	26	N＝NRPLL（P1，P2，Dx，Dy，M）
	27	P＝PRPLHN（N，M）
	28	P＝PRPLLN（N，M）
	29	F＝FRPLHN（N，M）
	30	F＝FRPLLN（N，M）
	31	L＝VRPLHN（N，M）
	32	L＝VRPLLN（N，M）
	33	L＝RPL1（P1，P2，Dx，Dy，M）
上下限の判定を行う	34	C＝LMTMD1（Dd，S，Ds）
	35	C＝LMTMD2（P，S，Ds、M）
	36	C＝LMTUL1（Dd，Up，Lo）
	37	C＝LMTUL2（P，Up，Lo，M）
電力を求める	38	W＝POWER（P1，P2，M）
トレース・データの read／write	39	T＝RTRACE（P，M）
	40	WTRACE（T，P，M） 注）この関数は値を返しません。

項　　目		ビ ル ト イ ン 関 数
グラフィック	1	GADRS (M0, X, Y)
	2	GFLRECT (D, X1, Y1, X2, Y2)
	3	GLINE (S, D, X1, Y1, X2, Y2)
	4	GMKR (MK, D, X, Y)
	5	GPOINT (D, X, Y)
	6	GRECT (S, D, X1, Y1, X2, Y2)
	7	GSTR (C, X, Y, STR)
	8	GSTYLE (dash, space, dot)

1.7 演算式

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合せで式を構成します。



1.7.1 代入演算子

代入式は式自体に値を持ちます。

例）

```
A＝１２３
ＰＲＩＮＴ　Ａ
ＰＲＩＮＴ　Ｂ＄＝”ＡＤＶＡＮＴＥＳＴ”
ＰＲＩＮＴ　（Ａ＝２）＋Ａ
```

【実行結果】

```
１２３．０
ＡＤＶＡＮＴＥＳＴ
４．０
```

代入演算子を以下に示します。

代入演算子	使用例	意味
＝	Ａ＝１２３	普通の代入
＋＝	Ａ　＋＝　５	Ａ　＝　Ａ　＋　５　と同じ意味
－＝	Ａ　－＝　５	Ａ　＝　Ａ　－　５　と同じ意味
＊＝	Ａ　＊＝　５	Ａ　＝　Ａ　＊　５　と同じ意味
／＝	Ａ　／＝　５	Ａ　＝　Ａ　／　５　と同じ意味
％＝	Ａ　％＝　５	Ａ　＝　Ａ　％　５　と同じ意味
＝＞	Ａ＄＝＞”ＡＢＣＤ”	文字列を右詰めで入力
＝＜	Ａ＄＝＜”ＡＢＣＤ”	文字列を左詰めで入力

注）　％　はモジュロ（余り）

例)

```
DIM S$[15]
A = 5        :PRINT A
A += 10      :PRINT A
A -= 3       :PRINT A
A *= A       :PRINT A
A /= 2       :PRINT A
A %= 5       :PRINT A
PRINT "1234567890123456"
S$ => "TEST"     :PRINT S$
S$ =< "TEST"     :PRINT S$
```

【実行結果】

```
5.0
15.0
12.0
144.0
72.0
2.0
1234567890123456
                TEST
TEST
```

1.7.2 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（変数に１を加える） ・前置インクリメント 例） A　＝　2　　Aに２を入力 B　＝　＋＋A　Aに１を加えてからBに入力 ・後置インクリメント 例） A　＝　2　　Aに２を入力 B　＝　A＋＋　BにAを入力してからAに１を加える
－－	前置／後置　デクリメント（変数から１を引く） ・前置デクリメント 例） A　＝　2　　Aに２を入力 B　＝　－－A　Aから１を引いてからBに入力 ・後置デクリメント 例） A　＝　2　　Aに２を入力 B　＝　A－－　BにAを入力してからAから１を引く

インクリメント演算子（＋＋）を変数の前に置いたときは、変数の値が使用される前に＋１の加算が行われます。変数の後に置いたときは、変数の値が使われてから＋１の加算が行われます。デクリメント演算子（－－）の場合も変数の値が１ずつ減算となる以外は同じです。

例）

```
A ＝ １０
A＋＋                    ：A＝A＋1　と同じ
＋＋A                    ：A＝A＋1　と同じ
A－－                    ：A＝A－1　と同じ
－－A                    ：A＝A－1　と同じ
ＰＲＩＮＴ ”１”，Ａ＋＋  ：Aの値を表示してからAに1加える
ＰＲＩＮＴ ”２”，Ａ
Ｂ ＝ －－Ａ              ：Aから1引いてからBに入力
Ｃ ＝ Ａ＋＋              ：AをCに入力してからAに1加える
ＰＲＩＮＴ ”３”，Ａ
ＰＲＩＮＴ ”４”，Ｂ
ＰＲＩＮＴ ”５”，Ｃ
Ｂ ＝ Ａ－－              ：AをBに入力してからAから1を引く
ＰＲＩＮＴ ”６”，Ａ
ＰＲＩＮＴ ”７”，Ｂ
Ｃ ＝ －１２３
Ｄ ＝ Ｃ ＋ （－２３）－ （＋５０）
ＰＲＩＮＴ ”８”，Ｃ
ＰＲＩＮＴ ”９”、Ｄ
```

【実行結果】

```
1          １０．０
2          １１．０
3          １１．０
4          １０．０
5          １０．０
6          １０．０
7          １１．０
8          －１２３．０
9          －１９６．０
```

1.7.3　2項算術演算子

＋	加算
－	減算
＊	積算
／	割り算
％	モジュロ（余り）
^	累乗
＆	文字列の連結

例）

```
ＰＲＩＮＴ　１０＋２
Ａ＝１０
ＰＲＩＮＴ　Ａ－５
ＰＲＩＮＴ　Ａ＊Ａ
ＰＲＩＮＴ　２０／Ａ
ＰＲＩＮＴ　Ａ％３
Ｂ＝３^４
ＰＲＩＮＴ　Ｂ
Ｓ＄＝”ＡＢＣＤ”
Ｓ１＄＝Ｓ＄＆”ＥＦＧ”
ＰＲＩＮＴ　Ｓ１＄
```

【実行結果】

```
１２
５．０
１００．０
２．０
１
８１．０
ＡＢＣＤＥＦＧ
```

1.7.4 論理演算子

複数の比較演算子を結合して複合条件の判定などに使用します。

NOT	否定	X		NOT X
		0		1
		1		0
AND	論理積	X	Y	X AND Y
		0	0	0
		0	1	0
		1	0	0
		1	1	1
OR	論理和	X	Y	X OR Y
		0	0	0
		0	1	1
		1	0	1
		1	1	1
XOR	排他的論理和	X	Y	X XOR Y
		0	0	0
		0	1	1
		1	0	1
		1	1	0

例）

```
IF NOT A THEN GOTO *MI
```

Aが0ならば、*MIへジャンプします。

```
IF X<100 OR 199<X THEN GOTO *LA
```

Xが100より小さい、または199より大きければ、*LAへジャンプします。

```
IF 0<=X AND X<=100 THEN PRINT X
```

Xが0以上、かつ100以下のとき、Xをプリントします。

```
IF A XOR B THEN PRINT A, B
```

Aが真のときBが偽、Aが偽のときBが真であれば、AとBをプリントします。

1.7.5 比較演算子

比較演算子は２つの数値を比較するときに使用します。比較結果は、真（１）偽（０）のどちらかで得られ、条件判断文（ＩＦ文）などのプログラムの流れを分岐させたりするのに使用します。

ＩＦ文内の条件では、必ず論理演算を行うので、”＝”の演算子は無条件に比較演算子とみなします。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。

ＩＦ文の条件式以外で比較演算を行うには、代入演算子の”＝”と区別するために”＝＝”をイコールとして使用します。

例）

```
A＝（Ｂ＄＝＝”ＡＤＶＡＮ”）
```

文字変数Ｂ＄が”ＡＤＶＡＮ”ならば、変数Ａには１が代入される。

記号	意味	例
＝（または　＝＝）	等しい	X＝Y，X＝＝Y
＜＞	等しくない	X＜＞Y
＜	小さい	X＜Y
＞	大きい	X＞Y
＜＝	小さいか等しい	X＜＝Y
＞＝	大きいか等しい	X＞＝Y

注）　＜＝、＞＝の記号は　＝＜、＝＞にはなりません。

例）

```
A＝1
B＝2
ＩＦ　A＝1　ＡＮＤ　Ｂ＞1　ＴＨＥＮ　ＰＲＩＮＴ　”Ａ”
ＩＦ　A＜＞1　ＯＲ　Ｂ＝5　ＴＨＥＮ　ＰＲＩＮＴ　”Ｂ”
ＩＦ　ＮＯＴ　Ａ　ＴＨＥＮ　ＰＲＩＮＴ　”Ｃ”
ＩＦ　Ａ　ＸＯＲ　Ｂ＞＝3　ＴＨＥＮ　ＰＲＩＮＴ　”Ｄ”
ＩＦ　A＝＝（Ｂ－1）　ＴＨＥＮ　ＰＲＩＮＴ　”Ｅ”
```

【実行結果】

```
A
D
E
```

1.7.6 サブ・ストリング演算子

文字列式の１部を参照できます。

・文字列式［算術式１，算術式２］
①文字列式の算術式１番目の文字から算術式２番目の文字までを、抜き出します。

例）

```
A$=" ABCDEFG"
PRINT  A$[3, 5]
PRINT  "*ADVANTEST*"[2, 6]
PRINT  "*ADVANTEST*"[7, 10]
```

【実行結果】

```
CDE
ADVAN
TEST
```

・文字列式［算術式１；算術式２］
②文字列式の算術１番目の文字から、算術式２桁の文字列を抜き出します。

例）

```
A$=" ABCDEFG"
PRINT  A$[3;4]
PRINT  "*ADVANTEST*"[7;4]
PRINT  "*ADVANTEST*"[2;5]
```

【実行結果】

```
CDEF
TEST
ADVAN
```

1.7.7　ビット演算子

ビット演算子はデータを構成する各ビットに対して直接働く演算子です。
ビット演算子は各ビットごとに論理演算（ＡＮＤ，ＯＲ，ＸＯＲなど）を行います。
ビット演算は、必ず整数型にて行います。ビット演算ができる範囲は０～６５５３５の１６ビット内です。負数を指定した場合、エラーになります。
（ＮＯ　ｏｐｅｒａｎｄ　ｉｎ　．．．）
数値変数で使用するときは、2.3節(24)ＩＮＴＥＧＥＲ命令で整数型を宣言して下さい。

ＢＮＯＴ	（１の補数） ＢＮＯＴ　０　➡　６５５３５ ＢＮＯＴ　６５５３５　➡　０
ＢＡＮＤ	（論理積） ６５５３５　ＢＡＮＤ　２５５　➡　２５５ ２５５　ＢＡＮＤ　１０２４　➡　０
ＢＯＲ	（論理和） ２５５　ＢＯＲ　１０２４　➡　１２７９ １　ＢＯＲ　２　➡　３
ＢＸＯＲ	（排他的論理和） ２５５　ＢＸＯＲ　１２８　➡　１２７ １　ＢＸＯＲ　３　➡　２

例）

```
INTEGER S
*L
S=SPOLL(31)
IF S BAND 4 THEN PRINT "SWEEP END"
GOTO *L
```

1.8. 演算の優先順位

種々の演算には優先順位があります。
実行時には下表の番号順に処理されます。

優先順位	演算子	優先順位	演算子
1	( ) で囲まれた式	１０	ＢＮＯＴ
2	関数	１１	ＢＡＮＤ
3	^	１２	ＢＯＲ
4	符号	１３	ＢＸＯＲ
5	＋＋，－－	１４	ＮＯＴ
6	＊　，　／	１５	ＡＮＤ
7	％	１６	ＯＲ
8	＋　，　－	１７	ＸＯＲ
9	関係演算子		

1.9 文字列演算

ＢＡＳＩＣでは、文字列に対して演算ができます。

1.9.1 文字列の連結

文字列は、”＆”で連結できます。

例）
```
A$=" ADVAN" &" TEST"
B$=A$&"  co. "
PRINT  A$
PRINT  B$
```

実行結果）
```
ADVANTEST
    ADVANTEST  co.
```

1.9.2 文字列の比較

文字も、数値の比較と同様に、関係演算子を用いて比較できます。

＝，　＜，　＞，　＜＞，　＜＝，　＞＝

ＩＦ文以外で、＝を比較するときは　＝＝を使用します。

例）　ＰＲＩＮＴ　Ａ＝＝Ｂ

文字列の最初から１文字ずつ比較します。相互に同じ長さの文字列の場合は、その文字のＡＳＣＩＩコードが大きい方の文字列が大きいと判断します。文字列の片方が短い場合は、短い方の文字列を小さいと判断します。
文字列の場合は、スペースなども意味を持つので注意して下さい。

例）”ＡＡ”＝”ＡＡ”　　➡ 真（ｔｕｒｅ）
　　”ＡＡ”＜”ａａ”　　➡ 真（ｔｕｒｅ）
　　”ＡＡＡ”＞”ＡＡ”　　➡ 真（ｔｕｒｅ）

1.9.3 型変換

文字列表現式から数値変数、数値表現式から文字列変数への入力は、直接行えます。
文字列表現式から数値変数の場合、文字列の最初に数字があったところから数字がなくなったところまでとします。

例）
```
A=" １２３．４"
B=" ＡＢＣ４５６．７ＤＥＦ８９"
C$=１２３
D$=B
PRINT  A
PRINT  B$
PRINT  C$
PRINT  D$
```

実行結果）　１２３．４
　　４５６．７
　　１２３
　　４５６．７

MEMO

2. コマンドとステートメントの文法と解説

2.1 概要

この章では、本器で使われるコマンドやステートメントの構文を理解できるように、記述式表現で解説します。

2.2 はじめに

2.2.1 説明の構成

各命令の解説は次の構成で説明します。

B U Z Z E R	－－	命令の名前（ショート・ネーム：省略形）
【概要】	－－	命令の機能説明
【書式】	－－	命令の記述の仕方（記述表現式）
【解説】	－－	命令の使用法や詳しい機能の説明
【例】	－－	命令の文例
【注意】	－－	注意書き
【プログラム例】	－－	その命令を使用したプログラム例
【実行結果】	－－	プログラム例の実行結果

(1) 書式

【書式】の記述式表現には、以下に示す記号を使用しています。

< > : この記号で囲まれた部分は、ユーザが指定します。
[ ] : この記号で囲まれた部分は、省略できます。
{ } : この記号で囲まれた部分は、繰り返し用いることができます。
, : コンマで複数の指定ができます。
| : または の意味

例）

＜A＞|＜B＞ ・・・ ＜A＞または＜B＞を用いる、ということです。

(2) 解説

【解説】に使用している単語の意味を説明します。

数値表現式　：数値定数、数値変数、数式のいずれかを示します。

文字列表現式　：文字列定数、文字列変数、文字列関数、サブ・ストリングで構成されている文字列式

装置アドレス　：ＧＰＩＢに接続されている装置のＧＰＩＢアドレス

ファイル・ディスクリプタ
：変数と同等の扱いで、データの入出力文につけます。

ラベル　：ラベル名（行番号も含む）
ラベル名は、アスタリスク（＊）を先頭につけ、英文字で構成します。

（3）機能別コマンドとステートメント　(1/3)

機　能	ｺﾏﾝﾄﾞとｽﾃｰﾄﾒﾝﾄ	内　　容	ページ
コマンド	CONT	プログラム停止後の再実行を行う。	2- 18
	CONTROL	各制御に関する値を設定する。	2- 19
	LIST	プログラム・リストを出力する。	2- 59
	LLIST	プログラム・リストを出力する。（ＲＳ２３２Ｃ）	2- 61
	LISTN	プログラム・リストを出力する。	2- 60
	LLISTN	プログラム・リストを出力する。（ＲＳ２３２Ｃ）	2- 62
	RUN	プログラムを実行させる。	2-107
	SCRATCH	ＢＡＳＩＣ内のプログラムを消去する。	2-109
	STEP	プログラムを１行実行する。	2-118
算術関数	ABS	与えられた値の絶対値を求める。	1- 11
	ATN	与えられた値の逆正接値を求める。	1- 11
	COS	与えられた値の余弦値を求める。	1- 11
	LOG	与えられた値の自然対数を求める。	1- 11
	SIN	与えられた値の正弦値を求める。	1- 11
	SQR	与えられた値の平方根を求める。	1- 11
	TAN	与えられた値の正接値を求める。	1- 11
ビット演算	BAND	ビットＡＮＤを求める。	1- 23
	BNOT	ビットＮＯＴを求める。	1- 23
	BOR	ビットＯＲを求める。	1- 23
	BXOR	ビットＸＯＲを求める。	1- 23
割り込み制御	ENABLE INTR	割り込み受信許可状態にする。	2- 29
	DISABLE INTR	割り込み受信禁止状態にする。	2- 27
	ON END	ＥＯＦ割り込みの分岐を定義する。	2- 75
	ON KEY	キー割り込みの分岐を定義する。	2- 79
	ON ISRQ	本体計測部ＳＲＱ割り込みの分岐を定義する。	2- 81
	ON SRQ	ＧＰＩＢのＳＲＱ割り込みの分岐を定義する。	2- 81
	ON ERROR	エラー発生割り込みの分岐を定義する。	2- 77
	OFF END	ＥＯＦ割り込みの分岐定義を解除する。	2- 70
	OFF KEY	キー割り込みの分岐定義を解除する。	2- 72
	OFF ISRQ	本体計測部ＳＲＱ割り込みの分岐定義を解除する。	2- 74
	OFF SRQ	ＧＰＩＢのＳＲＱ割り込みの分岐定義を解除する	2- 74
	OFF ERROR	エラー発生割り込みの分岐定義を解除する。	2- 71
文字列操作	NUM	文字列の先頭の文字のＡＳＣＩＩコードを求める。	1- 10
	CHR$	数値をＡＳＣＩＩ文字に変換する。	1- 10
	LEN	文字列の長さを求める。	1- 10
	POS	文字列２の中から文字列１がある位置を求める。	1- 10
	SPRINTF	書式を決め文字列変数に入力する。	1-115

注）各コマンドとステートメントの説明は2.3節を参照して下さい。

機　能	ｺﾏﾝﾄﾞとｽﾃｰﾄﾒﾝﾄ	内　　容	ページ
メモリカード制御	CAT	メモリ・カードの内容を出力する。	2- 11
	CALL	サブ・プログラムをロードする。	2- 9
	CLOSE #	ファイルを閉じる。	2- 16
	ENTER #	ファイルからデータを読み込む。	2- 30
	OPEN #	ファイルに開く。	2- 84
	OUTPUT #	ファイルにデータを書き込む。	2- 86
	INITIALIZE(INIT)	メモリ・カードを初期化する。	2- 53
	PURGE	指定ファイルを消去する。	2-108
	RENAME	ファイル名を変更する。	2-105
画面制御	CURSOR(CSR)	指定位置にカーソルを移動する。	2- 21
	CLS	画面を消去する。	2- 14
ステートメント	BUZZER	ブザーを鳴らす。	2- 7
	DIM	配列変数を宣言する。	2- 25
	FOR TO STEP NEXT	繰り返し処理を設定する。	2- 36
	BREAK	繰り返し処理から抜ける。	2- 36
	CONTINUE	繰り返し処理を先頭に戻す。	2- 36
	GOSUB	サブ・ルーチンに分岐する。	2- 43
	RETURN	サブ・ルーチンからの復帰する。	2- 43
	GOTO	指定位置へ分岐する。	2- 45
	IF THEN ELSE END IF	条件判断をして処理する。	2- 50
	INPUT(INP)	変数への入力を行う。	2- 57
	INTEGER	整数型変数の宣言する。	2- 54
	LPRINT	プリンタ(RS232C)に出力する。	2- 66
	LPRINT USING(USE)	プリンタ(RS232C)に出力する。（書式指定）	2- 68
	PAUSE	一時停止する。	2- 91
	PRINT(?)	文字を画面に出力する。	2- 95
	PRINT USING(USE)	文字を画面に出力する。（書式指定）	2- 97
	PRINTER	ＧＰＩＢプリンタ装置のアドレス指定を行う。	2- 99
	PRINTF(PRF)	文字を画面に出力する。（書式指定）	2- 92
	READ DATA	ＤＡＴＡ文からデータを読み取り、変数へ入力する。	2-101
	RESTORE	ＲＥＡＤ文で読むＤＡＴＡ文を指定する。	2-101
	REM(!)	注釈文	2-103
	SELECT CASE END SELECT	条件判断を行い処理する。	2-110
	STOP	プログラムの実行を停止する。	2-119
	WAIT	指定時間、実行停止する。	2-121

注）各コマンドとステートメントの説明は2.3節を参照して下さい。

機　能	ｺﾏﾝﾄﾞとｽﾃｰﾄﾒﾝﾄ	内　　容	ページ
G	CLEAR	ＤＣＬ、ＳＤＣを送出する。	2- 13
	DELIMITER	デリミタを設定する。	2- 23
P	ENTER(ENT)	ＧＰＩＢデータを入力する。(パラレルI/O含む)	2- 34
	GLIST	ＧＰＩＢプリンタへプログラム・リストを出力する。	2- 39
I			
	GLISTN	ＧＰＩＢプリンタへプログラム・リストを出力する。	2- 41
B			
	GPRINT	データをＧＰＩＢプリンタへ出力する。	2- 46
関	GPRINT USING(USE)	データをＧＰＩＢプリンタへ出力する。（書式指定）	2- 48
係	INTERFACE CLEAR	ＩＦＣを送出する。	2- 56
	LOCAL	指定装置をローカル状態にする。	2- 64
コ	LOCAL LOCKOUT	指定装置をローカル・ロックアウト状態にする。	2- 65
	OUTPUT(OUT)	データをＧＰＩＢに出力する。（パラレルI/O含む）	2- 89
マ			
	REMOTE	指定装置をリモート状態にする。	2-104
ン	REQUEST	標準ＧＰＩＢにＳＲＱを出力する。	2-106
	SEND	ＧＰＩＢデータを１つずつ出力する。	2-112
ド	SPOLL	指定装置のシリアル・ポールを行う。	2-114
	TRIGGER	ＧＥＴを送出する。	2-120

注）各コマンドとステートメントの説明は2.3節を参照して下さい。

2.3 各種コマンドとステートメントの説明

以下の順（アルファベット順）に説明します。

(1) ＢＵＺＺＥＲ

【概要】

本器内臓のブザーを鳴らします。

【書式】

ＢＵＺＺＥＲ　＜音程＞，＜時間＞

【解説】

指定された＜音程＞を指定された＜時間＞で鳴らします。

音程　：　整数　０～６５５３５（Ｈｚ）
時間　：　整数　０～６５５３５（ｍｓｅｃ）

【　音程　】

２６１	：ド	Ｃ	２７７	：ド＃	Ｃ＃
２９４	：レ	Ｄ	３１１	：レ＃	Ｄ＃
３３０	：ミ	Ｅ			
３４９	：ファ	Ｆ	３７０	：ファ＃	Ｆ＃
３９２	：ソ	Ｇ	４１５	：ソ＃	Ｇ＃
４４０	：ラ	Ａ	４６６	：ラ＃	Ａ＃
４９４	：シ	Ｂ			

数値を２倍にすると１オクターブ上がり、$\frac{1}{2}$倍にすると１オクターブ下がります。数値を大きくすると高音になります。

【例】

ＢＵＺＺＥＲ　４４０，１０００　　：ラの音を１秒間鳴らす。

【注意】

ＢＵＺＺＥＲ命令は、ブザーを鳴らす回路に設定するだけなので、ブザーは、指定時間鳴りますが、この命令の実行はすぐに終了します。
（１秒間で設定してもＢＵＺＺＥＲ命令は、設定後すぐに次の命令を実行します。）
連続してブザーを鳴らすときは、ＷＡＩＴ命令をＢＵＺＺＥＲ命令の次にＢＵＺＺＥＲ命令と同じ時間だけ設定して下さい。

【プログラム例】

```
FOR I=1 TO 8
 READ S
 BUZZER S,1000
 WAIT 1000
NEXT I
DATA 261,294,330,349,392,440
DATA 494,523
```

【実行結果】

”ドレミファソラシド”を1秒ごとに音階を変えて鳴らします。

（2）ＣＡＬＬ

【概要】

作成済みのプログラムの最終行に、メモリ・カードに登録されたファイル（サブプログラム）を読み込み、追加しサブルーチンとして実行します。

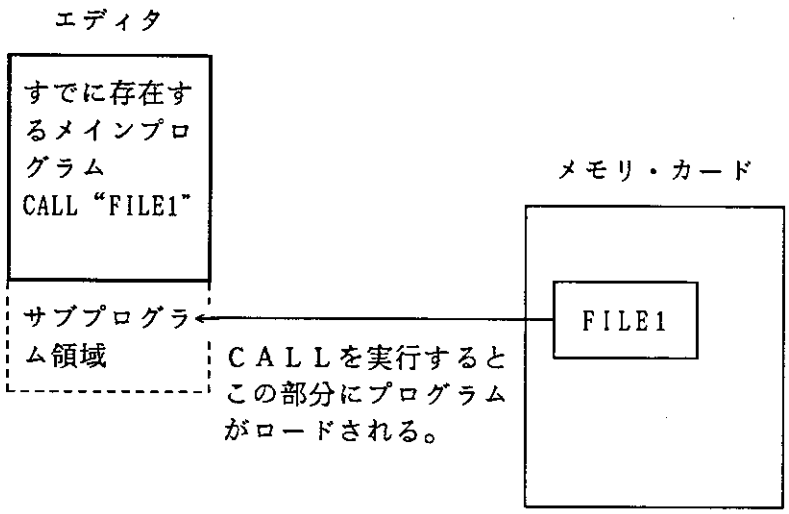
【書式】

ＣＡＬＬ　＜ファイル名＞

【解説】

すでに本器に存在するプログラムをメインプログラムとし、このメインプログラムの中のＣＡＬＬ命令でメモリ・カードから読み込み、メインプログラムの最終にプログラムを追加します。その追加されたプログラムをサブプログラムとします。（サブプログラムはサブルーチンとして扱います。）

メインプログラムからＣＡＬＬ命令で、サブプログラムを読み込みサブルーチンとして実行後、ＲＥＴＵＲＮ（復帰）し、ＣＡＬＬ命令の次の文から実行を再開します。（ＧＯＳＵＢ～ＲＥＴＵＲＮと同様）



【例】

ＣＡＬＬ“ＦＩＬＥ１”　　：ＦＩＬＥ１というプログラムを読み込み実行する。

【注意】

サブプログラム中にはＣＡＬＬ命令は使用できません。使用した場合にはエラーとなり、実行を中止します。
（ＣＡＮＮＯＴ　ｎｅｓｔ　ＣＡＬＬ）

サブプログラムは実行を終了するとロードしたサブプログラムを削除してメインプログラムにＲＥＴＵＲＮ（復帰）します。従ってエディタ上ではメインプログラム以外に何も表示されません。

変数は、メインプログラムと共用になるため、ＦＯＲ～ＮＥＸＴ等のループ処理での同じ変数名は誤動作の原因となります。サブプログラム専用の変数（ローカル変数）は作成できません。

サブプログラムにする場合は、サブプログラムをロードする際にＲＥＴＵＲＮ命令を自動的に挿入します。ＲＥＴＲＵＮがすでに存在していても動作には影響はありません。

サブプログラムに行番号がある場合は、ロードする際に行番号を無視します。メインプログラムと行番号が重複していても動作します。

【プログラム例】

```
FOR  I＝1  TO  10          :このプログラムを
  PRINT  I;                :“ＦＩＬＥ１”として
NEXT  I                    :ＳＡＶＥする。
```

```
FOR  K＝1  TO  5           :このプログラムを
  CALL  “ＦＩＬＥ１”      :メインプログラムとして
  PRINT                    :ＲＵＮさせる。
NEXT  K                    :
```

【実行結果】

```
loading...

12345678910
loading...

12345678910
loading...

12345678910
loading...

12345678910
loading...

12345678910
Program ended normally.
```

（3）ＣＡＴ

【概要】

メモリ・カードに格納されているファイル名を出力します。

【書式】

ＣＡＴ

【解説】

・ メモリ・カードに格納されているファイル名を出力します。

・ エディタのＢＡＳＩＣ ｍｏｄｅでＣＡＴ命令を実行します。

・ ＣＡＴを行うと以下の通り、管面に出力されます。

```
 16 FILE_1      BAS     256  1991-01-01  10:10
 17 DATA_NO1    DAT   19200  1991-02-14  12:10
 18 FILE_2      BAS     256  1991-01-15  10:20
 19 PROGRAM001  BAS     128  1990-10-04  15:35
 20 PROGRAM002  BAS     128  1990-12-24  09:04
 21 PROGRAM003  BAS    1408  1991-05-05  11:22
 22 NOISE       SET    1280  1991-03-03  13:45
```

ファイル作成時間（年月日　時分）

ファイルサイズ（バイト）

ファイルタイプ
- BAS ： ＢＡＳＩＣプログラム
- DAT ： ＢＡＳＩＣデータ
- SET ： 設定データ関係
- KEY ： ユーザ・デファイン　メニュー　データ

ファイル名

ファイル番号

```
 23 FILE_4      BAS    1792  1991-05-05  14:56
 24 FILE_5      BAS     128  1991-07-14  17:24

        9 files exists in  24 files          —— 登録ﾌｧｲﾙ数／登録可能ﾌｧｲﾙ数
             Total   25600 Bytes             —— 全カード容量
             Used    24576 Bytes (96 %)      —— 使用容量
                                                （占有率%）
```

【例】

ＣＡＴ

【注意】

・ 新しいメモリ・カードは初期化しないと使用できません。
（ＩＮＩＴＩＡＬＩＺＥ命令　参照）

(4) ＣＬＥＡＲ

【概要】

ＧＰＩＢ上に接続されたすべての装置、または指定した特定の装置を初期状態にします。

【書式】

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文だけ実行すると、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。このコマンドでＧＰＩＢ上のすべての装置を初期状態にできます。

・　ＣＬＥＡＲ文に続いて装置アドレス（０～３０）を指定すると、装置アドレスによって指定されている装置のみに、ユニバーサル・コマンドのセレクトデバイス・クリア（ＳＤＣ）を送ります。これにより特定の装置のみを初期状態にできます。なお装置アドレスは複数指定できます。

【例】

```
CLEAR

CLEAR  1

CLEAR  2，5，8
```

【注意】

・　装置アドレスで０～３０以外を指定すると、エラーになり、プログラムの実行を中止します。
（ＵＮＩＴ　ａｄｄｒ　ｅｒｒｏｒ　ｉｎ　ＣＬＥＡＲ）

・　スレーブ・モードでは機能しません。

【プログラム例】　　【実行結果】

```
CLEAR 3              ：装置アドレス３が初期状態になる
OUTPUT 3;"CF3MZ"     ：装置アドレス３にデータを送信する
CLEAR 3,5            ：装置アドレス３と５が初期状態になる
OUTPUT 3,5;"CF2MZ"   ：装置アドレス３と５にデータを送信する
CLEAR                ：接続されているすべての装置が初期状態になる
```

（5）ＣＬＳ

【概要】

外部端末、本体画面をクリアします。

【書式】

ＣＬＳ　［１｜２］

【解説】

・　外部端末、本体画面のキャラクタ画面、グラフィック画面をクリアします。

・　ＣＬＳ命令のパラメータは、以下のようになります。

指定なし　：　キャラクタ画面のみ
１　：　グラフィック画面のみ
２　：　キャラクタ、グラフィックの両画面

【例】

ＣＬＳ

ＣＬＳ　１

ＣＬＳ　２

【注意】

・　グラフィック画面を使う前には、必ずＯＵＴＰＵＴ　３１命令でＧＰＩＢコードの”ＶＳ３”を送って下さい。

【プログラム例】

例　１

```
OUTPUT 31;"VS3"
GLINE(0,1,0,0,1023,439)
PRINT "TEST"
WAIT 2000
CLS
```

例　２

```
OUTPUT 31;"VS3"
GLINE(0,1,0,0,1023,439)
PRINT "TEST"
WAIT 2000
CLS 1
```

例　3

```
OUTPUT 31;"VS3"
GLINE(0,1,0,0,1023,439)
PRINT "TEST"
WAIT 2000
CLS 2
```

【実行結果】

例　1

文字列”ＴＥＳＴ”が消去され、グラフィックが残ります。

例　2

グラフィックが消去され、文字列”ＴＥＳＴ”が残ります。

例　3

グラフィック、文字列”ＴＥＳＴ”がともに消去されます。

(6) 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。

【書式】

ＣＬＯＳＥ　＜＃ファイル・ディスクリプタ＞

【解説】

・　ＯＰＥＮ命令でオープンしたファイルは、メモリ・カードを抜く前や、本器の電源をＯＦＦする前に、必ずすべてのファイルをクローズさせないと、ファイルが破壊されます。特に書き込みとしてオープンしたファイルは、必ずクローズして下さい。

・　ＢＡＳＩＣプログラムでは、ＰＡＵＳＥ命令やＳＴＯＰキーでプログラムの実行を停止させたときは、ファイルを自動的にクローズしません。
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自動的にクローズします。
エラー終了時もクローズしますが、ＯＮ　ＥＲＲＯＲの定義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で明示的にクローズ動作を実行して下さい。

```
CLOSE *
```

このコマンドですべてのファイルをクローズします。

【例】

```
CLOSE #FD

CLOSE *
```

【注意】

・　ファイルをオープンしたときには、最後にクローズを必ず実行して下さい。

【プログラム例】

```
OPEN "FFF" FOR OUTPUT AS #FD
FOR I=1 TO 100
 OUTPUT #FD;I
NEXT I
CLOSE #FD
OPEN "FFF" FOR INPUT AS #FD
FOR I=1 TO 100
 ENTER #FD;N
 PRINT N
NEXT I
CLOSE #FD
```

【実行結果】

１から１００までの実数を”ＦＦＦ”というファイル名で書き込み、終了したら読み出し、書き込んだデータを表示します。（１．０～１００．０までプリントします。）

(7) ＣＯＮＴ

【概要】

ＢＡＳＩＣプログラムの実行を再開させます。

【書式】

ＣＯＮＴ［＜ラベル＞］

【解説】

・ ラベルを省略すると、プログラムの実行が停止した次の行から実行を再開します。ただしプログラムの実行が終了しているときは、エラーになります。
（Ｐｒｏｇｒａｍ　ＣＡＮＮＯＴ　ｂｅ　ｃｏｎｔｉｎｕｅｄ）

・ ラベルを指定すると、そのラベルの位置の次の命令から実行を開始します。

【例】

ＣＯＮＴ

ＣＯＮＴ　＊ＡＢＣ

【注意】

・ ラベル指定のＣＯＮＴ命令の実行は、ＢＡＳＩＣ　ｍｏｄｅで行って下さい。（その他の実行は、外部端末のＦ２０キー、または本体ＣＲＴのＣＯＮＴキーを押して下さい。）

・ ＣＯＮＴ命令での実行は、変数の初期化をしません。

・ プログラムの実行を一時停止させるには、(47)ＰＡＵＳＥ命令を使います。

(8) ＣＯＮＴＲＯＬ

【概要】

ＢＡＳＩＣの制御に関係する値を設定します。

【書式】

ＣＯＮＴＲＯＬ　＜レジスタ番号＞；＜数値表現式＞

＜レジスタ番号＞：２、３、４のいずれかを指定をします。

【解説】

レジスタ番号　２　：プリンタ出力時の左マージンを指定をします。
　　３　：プリンタ出力時にショート・ネームで出力するか否かを指定します。
　　４　：ＧＰＩＢアドレスを指定します。

・レジスタ　２
　ＬＩＳＴの出力を指定数分だけスペースをあけ出力を右に寄せます。

・レジスタ　３
　ＬＩＳＴ出力をショート・ネームで出力するか、フル・ネームで出力するかを設定します。

　０　：　フル・ネーム出力
　１　：　ショート・ネーム出力

・レジスタ　４
　ＧＰＩＢアドレスを設定します。
　０から３０までの数値を設定します。
　＊電源ＯＮ時の初期値は、０となっています。

【例】

レジスタ　２
プリンタ出力でスペースを５個入れたいとき

ＣＯＮＴＲＯＬ　３；５

で設定します。

```
_____10 PRINT "ADVANTEST"
_____30   PRINT I
_____40 NEXT I
```

レジスタ　3

ショート・ネームでプリンタに出力します。

CONTROL　3；1

で設定します。

```
10 CSR 10,10
20 OUT 31;"CF1.23MZ"
30 ENT 31;A
40 PRF "%f",A
```

フル・ネームでプリンタに出力します。

CONTROL　3；0

で設定します。

```
10 CURSOR 10,10
20 OUTPUT 31;"CF1.23MZ"
30 ENTER 31;A
40 PRINTF "%f",A
```

レジスタ　4

ＧＰＩＢアドレス１０を設定します。

CONTROL　4；10

で設定します。

（9）ＣＵＲＳＯＲ（ＣＳＲと省略可）

【概要】

画面のカーソルを指定位置へ移動させます。

【書式】

ＣＵＲＳＯＲ　＜Ｘ：カラム＞，＜Ｙ：行＞

ＣＳＲ　＜Ｘ：カラム＞，＜Ｙ：行＞

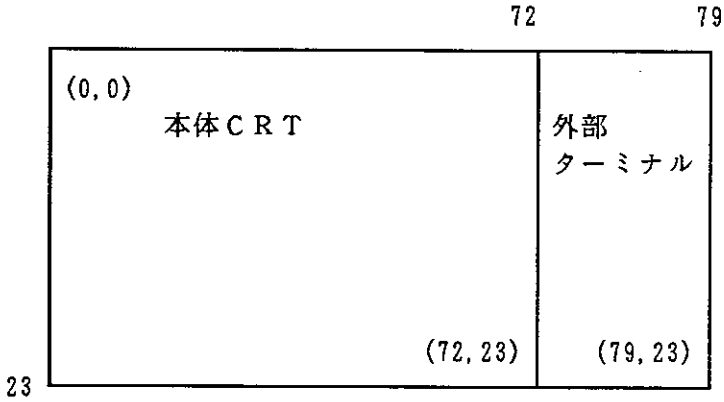
【解説】

カラム、行で指定された位置へカーソルを移動します。

	端末あり	端末なし
＜Ｘ：カラム＞（整数）	０≦Ｘ≦７９	０≦Ｘ≦７２
＜Ｙ：行＞　（整数）	０≦Ｙ≦２３	０≦Ｙ≦２３

端末あり：外部ターミナルによる起動
端末なし：本体管面による起動

どちらのＣＲＴでも左上が（０，０）になります。



【例】

ＣＵＲＳＯＲ　１０，２０　　：１０カラム、２０行目にカーソルを移動
ＣＳＲ　５０，１１　　：５０カラム、１１行目にカーソルを移動

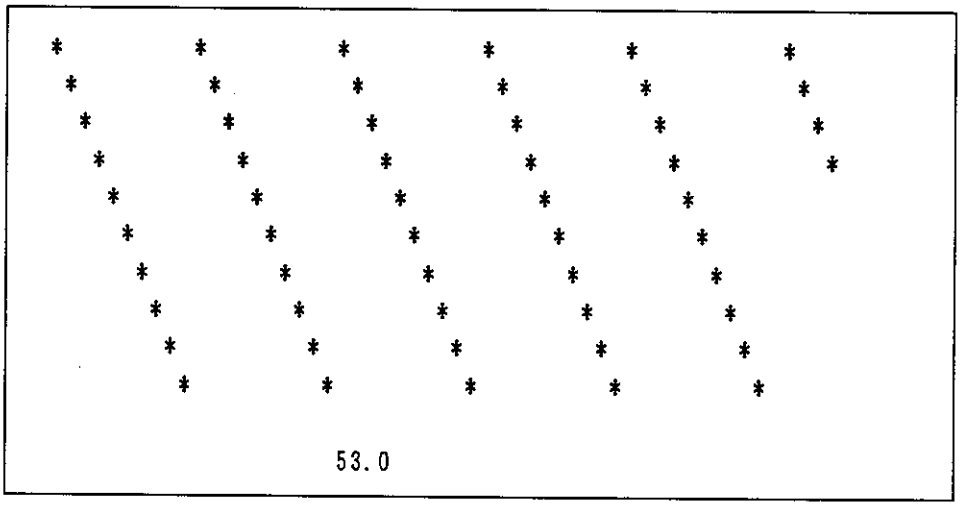
【注意】

端末なし（本器ＣＲＴ）の場合、ＢＡＳＩＣで使用できる画面サイズのカラム（Ｘ軸）が小さくなります。

【プログラム例】

```
OUTPUT 31;"VS2"
FOR X=0 TO 53
 CURSOR X, X%10
 PRINT "*";
 CSR 20, 15:PRINT X
NEXT X
```

【実行結果】



(10) ＤＥＬＩＭＩＴＥＲ

【概要】

４種類のデリミタを選択し、設定します。

【書式】

ＤＥＬＩＭＩＴＥＲ　＜Ｘ＞

Ｘ：数値表現式　０、１、２、３のいずれか

【解説】

・　ＯＵＴＰＵＴ　０～３０命令で出力時のターミネータ（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”＋単線信号ＥＯＩの２バイトを出力
1	”ＬＦ”の１バイトを出力
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バイトを出力

（◎印は電源ＯＮ時のデフォルト値）

【例】

ＤＥＬＩＭＩＴＥＲ　０

ＤＥＬＩＭＩＴＥＲ　１

ＤＥＬＩＭＩＴＥＲ　２

ＤＥＬＩＭＩＴＥＲ　３

【注意】

・ 選択番号０～３以外を指定すると、エラーになり、プログラムの実行を中止します。
(invalid value in DELIMITER)

・ ＤＥＬＩＭＩＴＥＲ文のデフォルト値は３です。

【プログラム例】　　【実行結果】

```
DELIMITER 0          :デリミタをＣＲ，ＬＦ＋ＥＯＩに設定する
OUTPUT 3;"CF3MZ"     :装置アドレス３にデータを出力
DELIMITER 1          :デリミタをＬＦに設定
OUTPUT 3;"CF4MZ"     :装置アドレス３にデータを出力
DELIMITER 2          :デリミタを最終バイト＋ＥＯＩに設定する
OUTPUT 3;"CF5MZ"     :装置アドレス３にデータを出力
DELIMITER 3          :デリミタをＣＲ，ＬＦに設定する
OUTPUT 3;"CF6MZ"     :装置アドレス３にデータを出力
```

（１１）ＤＩＭ

【概要】

配列変数または文字列変数の大きさを宣言します。

【書式】

ＤＩＭ　＜Ｘ＞｛，＜Ｘ＞｝

Ｘ：変数名（＜数値表現式＞｛，＜数値表現式＞｝）　｜
文字列変数名”［”＜数値表現式＞”］”

【解説】

・　配列変数の大きさおよび文字列変数の長さを宣言します。配列の要素を示す添字は最大値を指定します。（宣言をしないで使用すると、配列では１次元につき最大値１０の要素数になり、文字列では１８文字の長さとなります）

・　ＤＩＭ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行うとメモリ領域が足らなくなり、エラーとしてプログラムの実行を中止します。
（ｍｅｍｏｒｙ　ｓｐａｃｅ　ｆｕｌｌ）

・　配列変数のときの大きさを示す数値表現式は、実数表現になっていても、小数点以下を切り捨てて整数として扱い、宣言、参照します。

・　文字列変数のときの大きさを示す数値表現式は文字列の長さを宣言します。

・　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。
（次元数は、メモリ容量が許す限り）

【例】

```
ＤＩＭ　Ａ（２０）　　　　　：実数型配列変数Ａを２０個確保する。
ＤＩＭ　Ｂ（３０），Ｃ（２５，５）：実数型配列変数Ｂを３０個、実数型２次元
　　　　　　　　　　　　　　　配列変数Ｃを２５＊５個確保する。
ＤＩＭ　Ｓ＄［５０］　　　　：文字列変数Ｓ＄を５０文字分確保する。
ＤＩＭ　Ｄ（１５），Ｔ＄［５０］：実数型配列変数Ｄを１５個、文字列変数
　　　　　　　　　　　　　　　Ｔ＄を５０文字分確保する。
ＤＩＭ　Ｅ（３．８，５，２）　：実数型３次元配列変数Ｅを３＊５＊２個確
　　　　　　　　　　　　　　　保する。
```

【注意】

・ 配列変数の添字の最小値は、１です。

・ 配列宣言の最大値は、32767までとなりますが、プログラムの大きさにより確保できる量が変化します。

・ 文字列変数の長さは、最大１２８です。

・ 整数型変数を配列宣言するときは、ＩＮＴＥＧＥＲ命令を参照して下さい。

・ 添字が指定範囲外、０または負数で配列変数を参照すると、エラーになります。
（Ａｒｒａｙ’ｓ ｒａｎｇｅ ｅｒｒｏｒ または
Ｉｎｖａｌｉｄ ｄｉｍｅｎｓｉｏｎ ｐａｒａｍｅｔｅｒ）

・ 文字列変数における多次元配列宣言はできません。

【プログラム例】

```
DIM A(20),B(5,4),S$[20]          ：実数型配列宣言、文字列変数宣言
FOR I=1 TO 20                    ：Ｉが１からＩ＞２０になるまでループ
 A(I)=I                          ：配列変数にＩを入力
NEXT I                           ：ＮＥＸＴ　Ｉ
FOR X=1 TO 5                     ：Ｘが１からＸ＞５になるまでループ
 FOR Y=1 TO 4                    ：Ｙが１からＹ＞４になるまでループ
  B(X,Y)=A((Y-1)*5+X)            ：配列Ｂに配列Ａを入力
 NEXT Y                          ：ＮＥＸＴ　Ｙ
NEXT X                           ：ＮＥＸＴ　Ｘ
S$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"  ：Ｓ＄に文字列２６文字を入力
FOR X=1 TO 5                     ：Ｘが１からＸ＞５になるまでループ
 FOR Y=1 TO 4                    ：Ｙが１からＹ＞４になるまでループ
   CURSOR X*5,Y:PRINT B(X,Y)     ：配列Ｂを出力
 NEXT Y                          ：ＮＥＸＴ　Ｙ
NEXT X                           ：ＮＥＸＴ　Ｘ
PRINT S$                         ：文字列変数Ｓ＄を出力
```

【実行結果】

```
        1.0   2.0   3.0   4.0   5.0
        6.0   7.0   8.0   9.0   10.0
        11.0  12.0  13.0  14.0  15.0
        16.0  17.0  18.0  19.0  20.0
ABCDEFGIHJKLMNOPQRST
```

(12) ＤＩＳＡＢＬＥ　ＩＮＴＲ

【概要】

ＯＮ　（ＫＥＹ／ＳＲＱ／ＩＳＲＱ）命令によって生じた割り込みの受信を禁止します。

【書式】

ＤＩＳＡＢＬＥ　ＩＮＴＲ

【解説】

・　ＥＮＡＢＬＥ　ＩＮＴＲ命令による割り込み受信許可状態を禁止状態にします。

【注意】

・　割り込みによる分岐が生じた場合、分岐先で割り込みが入れ子（ｎｅｓｔ）にならないように分岐先でＤＩＳＡＢＬＥ　ＩＮＴＲ命令を実行して下さい。

・　割り込み処理をサブ・ルーチンにした場合は、サブ・ルーチン内のＲＥＴＵＲＮ命令の直前にＥＮＡＢＬＥ　ＩＮＴＲ命令を入れると円滑に処理ができます。

【プログラム例】

```
INTEGER S                    :整数型変数宣言
ON ISRQ GOSUB *SWPEND        :割り込み（ＩＳＲＱ）分岐先定義
OUTPUT 31;"IP SW1SC S0" : ＩＰ、掃引１秒、割り込み出力設定
*L                           : ┐（ループ）
 GOSUB *ONESWP               :  │１回掃引サブ・ルーチン
 M=MAX(0,700,0)              :  │最大レベル（ビルトイン関数）
 PRINT M                     :  │最大レベル出力
 GOTO *L                     : ┘ラベル　＊Ｌへジャンプ
!                            :
*ONESWP                      :
 F=0                         :Ｆ＝０
 ENABLE INTR                 :割り込み受信許可
 OUTPUT 31;"SI"              :シングル掃引スタート
 *LL                         :    （ループ）
  IF F THEN RETURN           : ┐Ｆが真ならばリターン
  GOTO *LL                   :  │ラベル　＊ＬＬへジャンプ
!                            : ┘
*SWPEND                      :
 DISABLE INTR                :割り込み受信禁止
 S=SPOLL(31)                 :本体計測部へのシリアル・ポール
 IF S BAND 4 THEN F=1        :３ビット目がオンならばＦ＝１を入力
 RETURN                      :リターン
```

【実行結果】

1掃引ごとの最大レベルを出力します。

（13）ＥＮＡＢＬＥ　ＩＮＴＲ

【概要】

ＯＮ　（ＫＥＹ／ＳＲＱ／ＩＳＲＱ）命令によって生じた割り込みの受信を許可します。

【書式】

ＥＮＡＢＬＥ　ＩＮＴＲ

【解説】

・　ＤＩＳＡＢＬＥ　ＩＮＴＲ命令による割り込み受信禁止状態を許可状態にします。

【注意】

・　割り込みによる分岐が生じた場合、分岐先で割り込みが入れ子（ｎｅｓｔ）にならないように分岐先でＤＩＳＡＢＬＥ　ＩＮＴＲを実行して下さい。

・　割り込み処理をサブ・ルーチンにした場合は、サブ・ルーチン内のＲＥＴＵＲＮ命令の直前にＥＮＡＢＬＥ　ＩＮＴＲ命令を入れると円滑に処理ができます。

・　プログラムのＲＵＮ後は、割り込み受信禁止状態になっているので、割り込み処理を使う場合は、ＥＮＡＢＬＥ　ＩＮＴＲ命令を実行して下さい。

【プログラム例】

```
ON KEY 1 GOSUB *K1
ON KEY 2 GOSUB *K2
ENABLE INTR
*L
 GOTO *L
!
*K1
  DISABLE INTR
  PRINT "KEY1"
  ENABLE INTR
  RETURN
*K2
  DISABLE INTR
  PRINT "KEY2"
  ENABLE INTR
  RETURN
```

【実行結果】

フル・キーボードの１または２、あるいは本体パネルのテンキーの１、２を押すと、ＫＥＹ１、ＫＥＹ２と出力します。

（14）ＥＮＴＥＲ　＃

【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からデータを入力（読み込み）します。

【書式】

ＥＮＴＥＲ　＜＃ファイル・ディスクリプタ＞　；　＜Ｘ＞［，＜Ｘ＞］

Ｘ：入力項目（数値変数，文字列変数）

【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データを、対応する入力項目のデータ・タイプの形式で読み込み、その入力項目に入力します。

・　ＯＰＥＮ命令で指定したタイプにより以下の書式で読み込まれます。

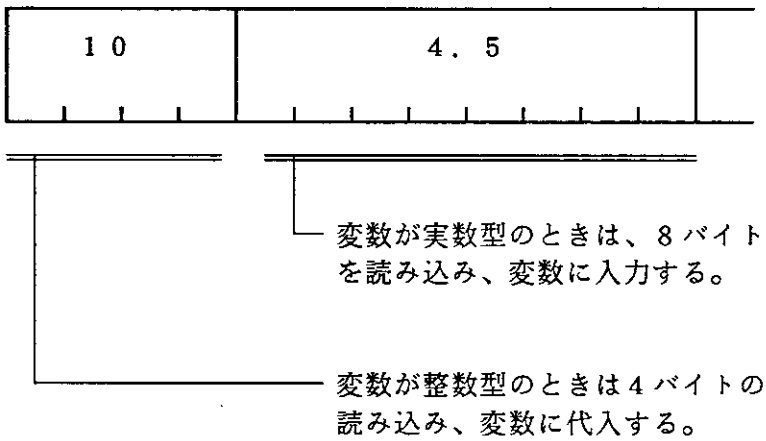
ＢＩＮＡＲＹ　タイプ

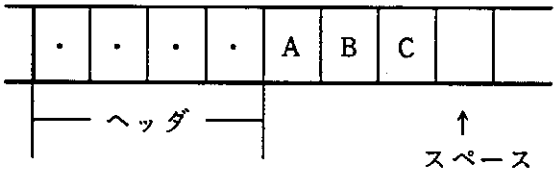
データを内部表現と同じ型で入力（読み込み）します。

整数型　：　４バイト
実数型　：　８バイト
文字列　：　４バイトのヘッダが示すバイト数のデータ

例）

```
ＩＮＴＥＧＥＲ　Ｉ
ＯＰＥＮ　”ＦＩＬＥ”　ＦＯＲ　ＩＮＰＵＴ　ＡＳ　＃ＦＤ
ＥＮＴＥＲ　＃ＦＤ；Ｉ，Ｒ，Ｓ＄
```





変数が文字列のときは、ヘッダを読み込みヘッダが示す長さ分だけ読み込み、文字列変数に入力します。

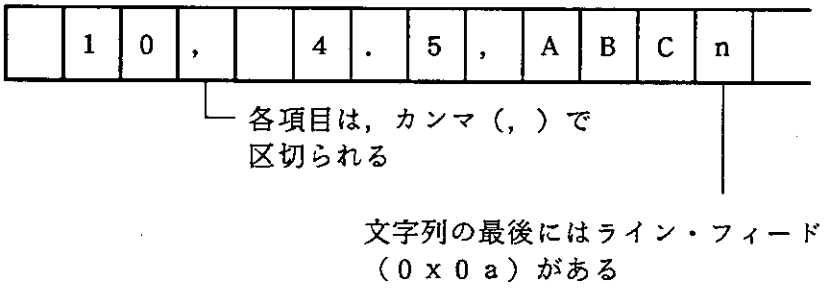
TEXT タイプ

入力項目の数にかかわらずライン・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入力されます。

データをＡＳＣＩＩコードに変換して出力します。数値データは、スペースか符号がデータの先頭に付きます。文字列データの最後にはライン・フィード（０ｘ０ａ）が付きます。

例）

```
INTEGER I
OPEN "FILE" FOR INPUT AS #FD;TEXT
ENTER #FD;I,R,S$
```

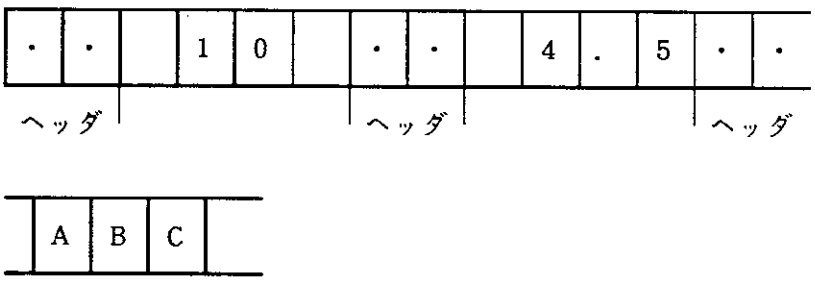


ASCII タイプ

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変数の型にデータを変換し入力します。

例）

```
INTEGER I
OPEN "FILE" FOR INPUT AS #FD;ASCII
ENTER #FD;I,R,S$
```



【例】

```
ENTER #FD;S$

ENTER #FD;A, B
```

【注意】

- ファイル・ディスクリプタは、ファイル・オープン時に指定したものを使います。
- ヘッダは各項目の区切りとしての目印となり、データの長さを持ちます。
- 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、入力されません。したがって、それらは前に格納されていた値がそのまま残り、逆に変数の数が実際のデータの数よりも少ない場合は、データが読み込まれずに捨てられます。
- 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きます。
- ＯＰＥＮ命令でファイルをオープンしないで本命令を使うとエラーになります。（ｆｉｌｅ　ｉｓ　ＮＯＴ　ｏｐｅｎ）

【プログラム例】

以下のプログラムを実行する前に2-90ページのプログラムを実行して下さい。

```
OPEN "A" FOR INPUT AS #FA;BINARY
INTEGER S
FOR I=1 TO 10
 ENTER #FA;R,S
 PRINT R,S
NEXT I
CLOSE #FA

OPEN "AA" FOR OUTPUT AS #FB;TEXT
INTEGER S
FOR I=1 TO 10
 ENTER #FB;R,S
 PRINT R,S
NEXT I
CLOSE #FB

OPEN "AAA" FOR OUTPUT AS #FC;ASCII
INTEGER S
FOR I=1 TO 10
 ENTER #FC;R,S
 PRINT R,S
NEXT I
CLOSE #FC
```

【実行結果】

上記プログラムを実行すると各プログラムとも以下のようにプリントします。

```
１．０        １
２．０        ２
３．０        ３
４．０        ４
５．０        ５
６．０        ６
７．０        ７
８．０        ８
９．０        ９
１０．０      １０
```

（15）ＥＮＴＥＲ（ＥＮＴと省略可）

【概要】

ＧＰＩＢ、本器測定部、またはパラレルＩ／Ｏからデータを取り込みます。

【書式】

ＥＮＴＥＲ　＜装置アドレス＞｜３１｜３２；＜Ｘ＞

装置アドレス　０～３０：ＧＰＩＢ接続の機器からのデータ入力
３１：本器の計測部からのデータ入力
３２：パラレルＩ／Ｏからのデータ入力

Ｘ：［数値変数｜文字列変数　{，数値変数｜文字列変数}］

【解説】

・　装置アドレス　０～３０は、背面パネルのＧＰＩＢ（下側：ＣＯＮＴＲＯＬＬＥＲ）上にある指定された装置からデータを変数に入力します。

・　３１を設定すると本器の計測部から内部メモリを通してデータを変数に入力します。

・　３２を設定すると１６ビットパラレルＩ／Ｏからデータを変数に入力します。※１
３２指定時は数値変数のみになります。文字列変数で指定すると、エラーになり、プログラムの実行を中止します。
（invalid type in assnum）

・　変数をカンマ（，）で区切ることにより変数を複数、指定できます。
（ＥＮＴＥＲ　３２に対しては無効）
この場合、受信したデータがカンマで区切られているごとに変数に入力します。受信したデータより変数が多い場合、データが割当たらない変数には、何も入力されません。逆にデータ数が多い場合、そのデータは無視されます。ＥＮＴＥＲ命令で受信できる文字は１０２４文字です。その数より多い場合は無視します。

※１　2.4節に示すパラレルＩ／Ｏを参照

【例】

```
ENTER  5 ; A
ENTER  5 ; A , B
ENTER  5 ; S $

ENTER  3 1 ; A
ENTER  3 1 ; A , B
ENTER  3 1 ; S $

ENTER  3 2 ; A
```

【注意】

・ アドレス番号０～３２以外を指定すると、エラーになり、プログラムの実行を中止します。
(UNIT addr error in ENTER)

・ 入力されたデータと変数の型が合わないと、エラーになります。

・ スレーブ・モードで使用するときは、[4. マスタ／スレーブ・モード] の章を参照して下さい。

【プログラム例】

例１

```
DIM B$[80]
OUTPUT 3;"CF?"
ENTER 3;A
OUTPUT 5;"SP?"
ENTER 5;B$
PRINT A
PRINT B$
```

例２

```
DIM S$[80]
OUTPUT 31;"CF1.23MZ"
OUTPUT 31;"CF?"
ENTER 31;A
ENTER 31;S$
PRINT A
PRINT S$
```

例３

```
INTEGER A
ENTER 32;A
IF A BAND 4 THEN PRINT "BIT 3"
```

（16）ＦＯＲ・・・ＴＯ・・・ＳＴＥＰ　ＢＲＥＡＫ／ＣＯＮＴＩＮＵＥ～ＮＥＸＴ

【概要】

ＦＯＲ文からＮＥＸＴ文までの間にある命令を繰り返して実行します。

【書式】

ＦＯＲ　＜数値変数＞＝＜初期値＞　ＴＯ　＜最終値＞
［ＳＴＥＰ　＜増分値＞］

ＢＲＥＡＫ

ＣＯＮＴＩＮＵＥ

ＮＥＸＴ　＜数値変数＞

初期値、　最終値、　増分値　：数値表現式

【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分値ずつ変化させていきます。カウンタの値が最終値を超えたとき、ループは終了しＮＥＸＴ文の次の命令へと移ります。

・　カウンタの増減はＮＥＸＴ文を使います。（数値変数に増分値を加え、ＦＯＲ文に戻ります。）

・　ＳＴＥＰ　＜増分値＞を省略した場合、増分値は１となります。

・　ＦＯＲ～ＮＥＸＴ文は、入れ子（ｎｅｓｔ）にできます。

・　一対で使用するＦＯＲ～ＮＥＸＴ文の＜数値変数＞は、同じものでなくてはいけません。変数名が異なっているとエラーになります。
（ＮＥＸＴ　ｗｉｔｈｏｕｔ　ＦＯＲ）

・　ＢＲＥＡＫ文を使用することにより、ＦＯＲ～ＮＥＸＴのループから抜け出すことができます。

・　ＣＯＮＴＩＮＵＥ文を使用することにより、ＮＥＸＴ文に達しなくても＜数値変数＞に増分値を加え、ループ処理を先頭に戻します。

【例】

```
FOR  I=1  TO  5                      :1-1
   PRINT  I                          :1-2
NEXT  I                              :1-3

FOR  J=1  TO  20  STEP  3            :2-1
   PRINT  J                          :2-2
   IF  J>12  THEN  BREAK             :2-3
NEXT  J                              :2-4

FOR  K=10  TO  0  STEP  -.5          :3-1
   IF  K>3  THEN  CONTINUE           :3-2
   PRINT  K                          :3-3
NEXT  K                              :3-4
```

1－1　：カウンタ　Ｉに１を入力、Ｉ＞５になるまでループ
1－2　：Ｉをプリント
1－3　：ＮＥＸＴ（１－１に戻る）

2－1　：カウンタ　Ｊに１を入力、ステップ３でＪ＞２０になるまでループ
2－2　：Ｊをプリント
2－3　：Ｊ＞１２ならばＢＲＥＡＫ（このループから抜ける）
2－4　：ＮＥＸＴ（２－１に戻る）

3－1　：カウンタ　Ｋに１０を入力、ステップ　－．５でＫ＜０になるまでループ
3－2　：Ｋ＞３ならば、ＣＯＮＴＩＮＵＥ（３－１に戻る）
3－3　：Ｋをプリント
3－4　：ＮＥＸＴ（３－１に戻る）

【注意】

・　ＦＯＲ～ＮＥＸＴ文でループ処理中に、ループ・カウンタに使用している数値変数の値を変えると、無限ループ等、正常なループの処理を行いません。

・　ＦＯＲ～ＮＥＸＴ文のループ内へＧＯＴＯ文などで入ってきたり、また逆にループ内から外に抜け出すようなプログラムは、その動作が保証されません。

・　ＢＲＥＡＫ、ＣＯＮＴＩＮＵＥ文は、ＦＯＲ～ＮＥＸＴ文中でのみ使用できます。

・　以下の場合は、ＦＯＲ～ＮＥＸＴが実行されずに、ＮＥＸＴ文の次へ実行が移ります。（実行されなくても、数値変数には初期値が入力されます。）
　　1　増分値が正の値で、＜初期値＞が＜最終値＞より大きい場合
　　2　増分値が負の値で、＜初期値＞が＜最終値＞より小さい場合

【プログラム例】

上記【例】のプログラム

【実行結果】

1のプログラム

```
1．0
2．0
3．0
4．0
5．0
```

2のプログラム

```
1．0
4．0
7．0
10．0
13．0
```

3のプログラム

```
3．0
2．5
2．0
1．5
1．0
0．5
0．0
```

（17）ＧＬＩＳＴ

【概要】

ＧＰＩＢプリンタにプログラム・リストを出力します。

【書式】

ＧＬＩＳＴ　［＜ラベル＞［，］］

【解説】

・　ＧＰＩＢプリンタにラベルで指定した位置のプログラム・リストを出力します。また、ラベルを省略すると、プログラムの先頭から最終行までを出力します。

・　ラベルを指定し、ラベルのあとにカンマ（，）を指定すると、そのラベル位置からプログラムの最後までを出力します。

・　プリンタへの出力は、１度ＢＡＳＩＣプログラムを転送し（ＭＯＶＥ　ａｎｄ　ＲＵＮ）、プログラムを停止させ、ＢＡＳＩＣ　ｍｏｄｅでＰＲＩＮＴＥＲ命令、ＧＬＩＳＴ命令を行って下さい。また、プログラム中に置くこともできます。

【例】

```
ＧＬＩＳＴ

ＧＬＩＳＴ　＊ＡＢＣ，
```

【注意】

・　行番号指定でも構いませんが書式が上記と異なります。

ＧＬＩＳＴ　［＜出力開始行＞］［，［＜出力終了行＞］］

例）

```
ＧＬＩＳＴ　１００
ＧＬＩＳＴ　１００，
ＧＬＩＳＴ　，２００
ＧＬＩＳＴ　１００，２００
```

・　接続するＧＰＩＢコネクタは、背面パネルのＣＯＮＴＲＯＬＬＥＲ側を使います。

・　プリンタのＧＰＩＢアドレス指定は、（５１）ＰＲＩＮＴＥＲ命令を参照して下さい。

・　装置アドレスが間違っていたり、指定したアドレスに装置が接続されていない場合、本命令を無視して次へ進みます。

・　プログラム・リストの出力は、ＢＡＳＩＣインタプリタ側のバッファの最後に転送（ＭＯＶＥ　ａｎｄ　ＲＵＮ）されたものが出力されます。

【プログラム例】

その１

リスト出力させたいプログラムをエディタ上に入力し、ポップアップ・メニューのＭＯＶＥ　ａｎｄ　ＲＵＮを選択します。

転送が終了しプログラムの実行が開始されたら、コントロールＣでプログラムの実行を停止させます。（最後まで実行させてもよい。）

ＢＡＳＩＣ　ｍｏｄｅのミニ・ウィンドウが画面右下に表示されたら（ポップアップ・メニューで選択してもよい）、ＰＲＩＮＴＥＲ命令とＧＬＩＳＴ命令を実行します。

例）

```
ＰＲＩＮＴＥＲ　３
ＧＬＩＳＴ
```

例２

リスト出力させたいプログラムの先頭にあらかじめ
ＰＲＩＮＴＥＲ命令
ＧＬＩＳＴ命令
ＳＴＯＰ命令
を入力しておき、ＭＯＶＥ　ａｎｄ　ＲＵＮを実行します。

例）

```
ＰＲＩＮＴＥＲ　３
ＧＬＩＳＴ
ＳＴＯＰ
  ・
  ・　以下プログラム
  ・
```

【実行結果】

例１

ＧＰＩＢアドレス３のプリンタにリストを出力します。

例２

ＭＯＶＥ　ａｎｄ　ＲＵＮを実行すると、ＧＰＩＢアドレス３のプリンタにリストを出力します。

（18）ＧＬＩＳＴＮ

【概要】

ＧＰＩＢプリンタにプログラム・リストを出力します。

【書式】

ＧＬＩＳＴＮ　［＜ラベル＞］［，＜行数＞］

【解説】

・　ＧＰＩＢプリンタにラベルで指定した位置のプログラム・リストを出力します。

・　ラベルを指定し、ラベルのあとにカンマ（，）行数を指定すると、そのラベル位置から行数分を出力します。

・　ラベルを省略し、＜行数＞を指定すると＜行数＞が正の値だとプログラムの先頭から行数分、負の値ならばプログラムの最終行から行数分、出力されます。

・　出力の方法は（１７）ＧＬＩＳＴ命令と同等です。

【例】

```
ＧＬＩＳＴＮ

ＧＬＩＳＴＮ　＊ＡＢＣ，１０

ＧＬＩＳＴＮ　＊ＡＢＣ，－１０

ＧＬＩＳＴＮ　，２０

ＧＬＩＳＴＮ　，－２０
```

【注意】

・　行番号指定でも構いませんが、書式が上記と異なります。

ＧＬＩＳＴＮ　［＜出力開始行＞］［，［＜行数＞］］

例）

```
ＧＬＩＳＴＮ　１００
ＧＬＩＳＴＮ　１００，１５
ＧＬＩＳＴＮ　１００，－１５
ＧＬＩＳＴＮ　，２０
ＧＬＩＳＴＮ　，－２０
```

・　プリンタのＧＰＩＢアドレス指定は、（５１）ＰＲＩＮＴＥＲ命令を参照して下さい。

・　装置アドレスが間違っていたり、指定したアドレスに装置が接続されていない場合、本命令を無視して次へ進みます。

・　本命令の使用方法は（１７）ＧＬＩＳＴ命令と同様です。

(19) 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。

【書式】

ＧＯＳＵＢ　＜ラベル＞

ＲＥＴＵＲＮ

【解説】

・　ＧＯＳＵＢ　＜ラベル＞で指定されたサブ・ルーチンへ処理の制御を移します。

・　サブ・ルーチン内のＲＥＴＵＲＮ文によりジャンプしたＧＯＳＵＢ文の次の命令へと処理の制御を戻します。

・　ＧＯＳＵＢ～ＲＥＴＵＲＮ文は、入れ子（ｎｅｓｔ）にできるので、サブ・ルーチンから別のサブ・ルーチンへ分岐できます。

注）入れ子（ｎｅｓｔ）が多過ぎるとメモリ容量が足らなくなり、エラーになることがあります。
（ＧＯＳＵＢ　ｎｅｓｔ　ｏｖｅｒｆｌｏｗ）

【例】

```
ＧＯＳＵＢ　＊Ｓ１

＊Ｓ１
    Ａ＝Ａ＊２
    ＲＥＴＵＲＮ
```

【注意】

・　ＧＯＳＵＢ　＜ラベル＞でジャンプするときに、その対象となるラベルがないとエラーになり、プログラムの実行を中止します。
（Ｕｎｄｅｆｉｎｅｄ　ＬＡＢＥＬ）

【プログラム例】

```
FOR I=2 TO 5                          : Ｉに２を入力、Ｉ＞５になるまでループ
 A=2                                  : Ａ＝２
 B=A                                  : Ｂ＝Ａ
 GOSUB *SUB1                          : サブ・ルーチン　＊ＳＵＢ１へジャンプ
 PRINTF "2^%2d = %5d\n\r",I,B         : データを出力
NEXT I                                : ＮＥＸＴ
STOP                                  : プログラム終了
!                                     :
*SUB1                                 : ラベル　＊ＳＵＢ１
  FOR C=1 TO I-1                      : Ｃ＝２、Ｃ＞Ｉ－１になるまでループ
   GOSUB *SUB2                        : サブ・ルーチン　＊ＳＵＢ２へジャンプ
  NEXT C                              : ＮＥＸＴ
  RETURN                              : リターン
!                                     :
*SUB2                                 : ラベル　＊ＳＵＢ２
  B *= A                              :  Ｂ　＝　Ｂ　＊　Ａ
  RETURN                              : リターン
```

【実行結果】

```
2 ^  2  =     4
2 ^  3  =     8
2 ^  4  =    16
2 ^  5  =    32
```

（２０）ＧＯＴＯ

【概要】

指定されたラベルへの分岐します。

【書式】

ＧＯＴＯ　＜ラベル＞

【解説】

・　指定された＜ラベル＞へ無条件で分岐します。

【例】

```
ＧＯＴＯ　＊ＬＡ
```

【注意】

・　ＧＯＴＯ　＜ラベル＞でジャンプするときに、その対象となるラベルがないとエラーになり、プログラムの実行を中止します。
（Ｕｎｄｅｆｉｎｅｄ　ＬＡＢＥＬ）

【プログラム例】

```
Ｉ＝０                          ：　変数Ｉに０を入力
＊Ｌ                            ：　ラベル名
  Ｉ＝Ｉ＋１                    ：　変数Ｉに１を加えて変数Ｉに入力
  ＩＦ　Ｉ＝１０　ＴＨＥＮ　ＳＴＯＰ　：　もし、Ｉが１０になったら終了
  ＰＲＩＮＴ　Ｉ                ：　Ｉを出力
  ＧＯＴＯ　＊Ｌ                ：　＊Ｌ　にジャンプ
```

【実行結果】

```
１．０
２．０
３．０
４．０
５．０
６．０
７．０
８．０
９．０
```

（21）ＧＰＲＩＮＴ

【概要】

ＧＰＩＢで接続されているプリンタに数値、文字列を出力します。

【書式】

ＧＰＲＩＮＴ　[＜Ｘ＞　{[　，｜；　＜Ｘ＞]}][＜；＞]

Ｘ：数値表現式｜文字列表現式

【解説】

・　（５１）ＰＲＩＮＴＥＲ命令で指定されたＧＰＩＢアドレスの装置に数値、文字列を出力します。

・　数値表現式、文字列表現式はセミ・コロン（；）またはカンマ（，）で区切って複数にわたり指定できます

・　ＧＰＲＩＮＴ文の最後にセミ・コロン（；）を置いた場合は、プリント出力が終わっても改行されません。したがって、次のＰＲＩＮＴ文は以前にプリントした行に続いてプリントします。

【例】

```
Ｓ＄＝”ＤＥＦ”                    :  ①
ＧＰＲＩＮＴ　”ＡＢＣ”；          :  ②
ＧＰＲＩＮＴ　Ｓ＄                 :  ③
ＧＰＲＩＮＴ　”Ａ＝”，Ａ           :  ④
ＧＰＲＩＮＴ　”ＣＦ”，ＣＦＱ，”ＫＨｚ”  :  ⑤
ＧＰＲＩＮＴ　Ａ＋１００           :  ⑥
```

① 文字列変数Ｓ＄に”ＤＥＦ”を入力
② プリンタに”ＡＢＣ”を改行しないで出力
③ 文字列変数Ｓ＄を出力
④ 文字列定数と、数値変数を出力
⑤ 文字列定数と、数値変数を出力
⑥ 数値変数に１００を加え出力

【注意】

・　ＧＰＩＢプリンタに出力する前は、必ず、（５１）ＰＲＩＮＴＥＲ命令でＧＰＩＢプリンタのアドレスを設定して下さい。（ＧＰＩＢプリンタとのアドレスが合わないとプリンタに出力されません。）

・　装置アドレスが間違っていたり、指定したアドレスに装置が接続されていない場合、本命令を無視して次へ進みます。

【プログラム例】

```
PRINTER 5
FOR I=1 TO 10
 GPRINT "I= ";
 GPRINT I
NEXT I
```

【実行結果】

ＧＰＩＢのアドレス５のプリンタに１から１０までを出力します。

（２２）ＧＰＲＩＮＴ　ＵＳＩＮＧ　（ＵＳＥと省略可）

【概要】

数値、文字列などを編集し、ＧＰＩＢへ出力します。

【書式】

ＧＰＲＩＮＴ　ＵＳＩＮＧ　＜イメージ仕様＞；　{，＜Ｘ＞}

ＧＰＲＩＮＴ　ＵＳＥ　＜イメージ仕様＞；　{，＜Ｘ＞}

Ｘ：数値表現式｜文字列表現式

【解説】

・　ＧＰＩＢポートに数値、文字列を編集し、ＡＳＣＩＩコードで出力します。

・　（５１）のＰＲＩＮＴＥＲ命令で指定されたＧＰＩＢアドレスの装置に数値、文字列を出力します。

・　イメージ仕様の指定は、文字列表現式にてイメージ仕様をカンマ（，）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。）

【イメージ仕様一覧】

Ｄ	指定フィールドの余った部分にスペースを出力する
Ｚ	指定フィールドの余った部分に０を詰める
Ｋ	数値をそのまま出力する
Ｓ	常に＋／－の符号を付ける
Ｍ	－の符号を付けるか値が正のときはスペースを付ける
.（小数点）	小数点を出力する
Ｅ	指数形式（ｅ、符号、指数）で出力する
Ｈ	数値、文字列をそのまま出力するが、小数点（．）がヨーロッパ・タイプ（，）になる
Ｒ	ヨーロッパ・タイプの小数点になる
＊	指定フィールドの余った部分に＊印を出力する
Ａ	１文字、出力する
ｋ	文字列をそのまま出力する
Ｘ	スペースを出力
リテラル	書式指定にリテラルを書くときは、　＼”　で囲む
Ｂ	数値をＡＳＣＩＩコードとして出力する
@	改ページする
＋	出力位置を同じ行の先頭に移動させる
－	出力位置を次の行に移動させる
＃	最後に改行を行わない
ｎ	ｎ桁の精度で出力する。文字列に対して指定すると、出力の値は、その文字列の長さになる

【例】

GPRINT USING "DDD.DD"; 1.2

GPRINT USE "ZZZ.ZZ"; 1.2

【注意】

・ GPRINT USING命令の出力は、復改はされません。改行だけ行うので、復改をGPRINT USINGの中に入れて下さい。

GPRINT USING "K,k";123,"\r"

・ イメージ仕様で指定した数がパラメータで指定した数より多い場合、エラーになり、プログラムの実行を中止します。
(Unmatched IMAGE-spec in USING)

・ GPIBプリンタに出力する前には、必ず、(51) PRINTER命令でGPIBプリンタのアドレスを設定して下さい。(GPIBプリンタとのアドレスが合わないとプリンタに出力されません。)

・ 装置アドレスが間違っていたり、指定したアドレスに装置が接続されていない場合、本命令を無視して次へ進みます。

【プログラム例】

```
PRINTER 5
GPRINT USING "DDD.DD,k"; 1.2,"\r"
GPRINT USE "ZZZ.ZZ,k"; 1.2,"\r"
GPRINT USE "***.ZZ,k"; 1.2,"\r"
A$="K,5X,B,DDDRDD,k"
GPRINT USE A$; 2.34,65,1.2,"\r"
```

【実行結果】

GPIBアドレス5のプリンタに以下のように出力されます。

```
1.20
001.20
**1.20
2.34        A    1,20
```

(23) IF THEN ELSE／ELSE IF／END IF

【概要】

条件判断による分岐、または指定された文を実行します。

【書式】

・ IF <論理式> THEN <文>

・
```
IF <論理式> THEN
        <複文>
END IF
```

・
```
IF <論理式> THEN
        <複文>
ELSE
        <複文>
END IF
```

・
```
IF <論理式> THEN
        <複文>
ELSE IF <条件式> THEN
        <複文>
ELSE IF <条件式> THEN
        <複文>
        .
        .
ELSE
        <複文>
END IF
```

【解説】

・ <論理式>の条件によりプログラムの実行を制御します。<論理式>の結果が0になったときを偽（false）とし、0以外のときを真（true）と判断します。

・ <論理式>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のときを偽（false）、0以外は真（true）と判断します。

・ <論理式>の判断が真（true）のときは、THEN文を実行します。THEN文の後ろには文を続けることができます。（このときは、IF文は1行でしか書けません。）

・ THEN以下の文を複数行続けたい場合は、THEN文で一旦区切り、次の行から複数行にわたり文を書き、最後にEND IFを置きます。また、ELSE文を置くと、偽（false）のための文も書けます。

・ ＥＬＳＥ　ＩＦを使用すると、＜論理式＞を複数書くことができます。

・ 比較演算子には以下のものがあります。

記号	意味	例
＝（または　＝＝）	等しい	Ｘ＝Ｙ，Ｘ＝＝Ｙ
＜＞	等しくない	Ｘ＜＞Ｙ
＜	小さい	Ｘ＜Ｙ
＞	大きい	Ｘ＞Ｙ
＜＝	小さいか等しい	Ｘ＜＝Ｙ
＞＝	大きいか等しい	Ｘ＞＝Ｙ

注）　＜＝、＞＝の記号は　＝＜、＝＞にはなりません。

・ 論理演算子を使い、比較演算子をつなげることもできます。

```
NOT
AND
OR
XOR
```

【例】

```
IF  A=1  THEN  PRINT  "A=1"

IF  B>10  THEN
    PRINT  "B>10"
END  IF

IF  B>2  THEN
    PRINT  "B>2"
ELSE
    PRINT  "B<=2"
END  IF

IF  1<C  AND  C<5  THEN
    PRINT  "1<C<5"
ELSE  IF  C=10  THEN
    PRINT  "C=10"
ELSE
    PRINT  "C:ELSE"
END  IF
```

【注意】

・ ＩＦ文を複数行にわたり書く場合、必ずＩＦ文の最後には、ＥＮＤ　ＩＦを置いて下さい。ＩＦ～ＥＮＤ　ＩＦ文の数が一致しないと、エラーになります。
（Ｕｎｂａｌａｎｃｅｄ　ＩＦ　ｂｌｏｃｋ）

【プログラム例】

```
FOR I=1 TO 100              : Ｉ＝１、Ｉ＞１００になるまでループ
 IF 10<=I AND I<=20 THEN    :  ①１０以上２０以下ならば以下の処理
   PRINT "10<=I<=20"        :    文字列出力
 END IF                     :  ①のＩＦ文の終わり
 IF I=50 OR I=60 THEN       :  ②Ｉ＝５０、またはＩ＝６０ならば以下の処理
   IF I=50 THEN             :    ③Ｉ＝５０ならば以下の処理
     PRINT "I=50"           :          文字列出力
   ELSE                     :    ③Ｉ＝５０以外ならば以下の処理
     PRINT "I=60"           :          文字列出力
   END IF                   :    ③のＩＦ文の終わり
 ELSE IF I=70 THEN          :  ②Ｉ＝７０ならば以下の処理
   PRINT "I=70"             :        文字列出力
 ELSE IF I=90 THEN          :  ②Ｉ＝９０ならば以下の処理
   PRINT "I=90"             :        文字列出力
 END IF                     :  ②のＩＦ文の終わり
NEXT I                      : ループ戻り
```

【実行結果】

```
１０＜＝Ｉ＜＝２０
１０＜＝Ｉ＜＝２０
１０＜＝Ｉ＜＝２０
１０＜＝Ｉ＜＝２０
１０＜＝Ｉ＜＝２０
１０＜＝Ｉ＜＝２０
１０＜＝Ｉ＜＝２０
１０＜＝Ｉ＜＝２０
１０＜＝Ｉ＜＝２０
１０＜＝Ｉ＜＝２０
１０＜＝Ｉ＜＝２０
Ｉ＝５０
Ｉ＝６０
Ｉ＝７０
Ｉ＝９０
```

（２４）ＩＮＩＴＩＡＬＩＺＥ（ＩＮＩＴと省略可）

【概要】

メモリ・カードを初期化します。

【書式】

ＩＮＩＴＩＡＬＩＺＥ

ＩＮＩＴ

【解説】

・ メモリ・カードを初期化します。

・ エディタのＢＡＳＩＣ ｍｏｄｅにてＩＮＩＴＩＡＬＩＺＥ命令を実行します。

【例】

ＩＮＩＴＩＡＬＩＺＥ

ＩＮＩＴ

【注意】

・ 初期化する際はＣＡＴ命令、またはエディタのＬｏａｄを実行し、ファイル内容を確かめてから行って下さい。

・ 新しいメモリ・カードは初期化を行わないと使用できません。また、すでに初期化済みのメモリ・カードを初期化すると、メモリ・カード内のファイルはすべて消去されます。

（２５）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。

【書式】

ＩＮＴＥＧＥＲ　＜Ｘ＞｜＜Ｘ＞（数値表現式）　{，＜Ｘ＞｜
＜Ｘ＞（数値表現式）}

Ｘ：数値変数

【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になります。

・　整数型変数で扱える数値は、整定数で扱える範囲と同等です。
－２，１４７，４８３，６４８　～　＋２、１４７、４８３、６４７

・　整数しか扱わない変数では、ＩＮＴＥＧＥＲ文で宣言した方が、高速に処理できます。

・　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行うと、メモリ領域が足らなくなり、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になります。
（次元数は、メモリ容量が許す限り指定することができます。）

【例】

```
ＩＮＴＥＧＥＲ　Ａ
ＩＮＴＥＧＥＲ　Ｂ，Ｃ
ＩＮＴＥＧＥＲ　Ｄ（２０），Ｅ（３０）
```

【注意】

・　ＦＯＲ～ＮＥＸＴ等のループ変数に整数型を使うと高速に処理できます。

・　実数型変数を配列宣言するときは、（１１）ＤＩＭ命令を参照して下さい。

【プログラム例】

```
INTEGER A(20),B(5,4),I,X,Y     :整数型宣言
FOR I=1 TO 20                  :Ｉが１からＩ＞２０になるまでループ
 A(I)=I                        :配列変数にＩを入力
NEXT I                         :ＮＥＸＴ　Ｉ
FOR X=1 TO 5                   :Ｘが１からＸ＞５になるまでループ
 FOR Y=1 TO 4                  :　Ｙが１からＹ＞４になるまでループ
  B(X,Y)=A((Y-1)*5+X)          :　　配列Ｂに配列Ａを入力
 NEXT Y                        :　ＮＥＸＴ　Ｙ
NEXT X                         :ＮＥＸＴ　Ｘ
FOR X=1 TO 5                   :Ｘが１からＸ＞５になるまでループ
 FOR Y=1 TO 4                  :　Ｙが１からＹ＞４になるまでループ
  CURSOR X*5,Y:PRINT B(X,Y)    :　　配列Ｂを出力
 NEXT Y                        :　ＮＥＸＴ　Ｙ
NEXT X                         :ＮＥＸＴ　Ｘ
```

【実行結果】

```
1       2       3       4       5
6       7       8       9       １０
１１    １２    １３    １４    １５
１６    １７    １８    １９    ２０
```

(26) INTERFACE CLEAR

【概要】

背面パネルのCONTROLLER側のGPIBに接続されているすべての装置のGPIBインタフェースを初期化します。

【書式】

INTERFACE CLEAR

【解説】

・ この命令を実行すると、GPIBの単線信号IFCを約100μsの間、出力（true状態）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を解除します。

【例】

INTERFACE CLEAR

【注意】

・ スレーブ・モードでは機能しません。

【プログラム例】

```
INTERFACE CLEAR
OUTPUT 3;"CF1MZ"
```

（27）ＩＮＰＵＴ（ＩＮＰと省略可）

【概要】

キーボード、本器パネルから入力したデータを数値変数、または文字列変数に入力します。

【書式】

ＩＮＰＵＴ　[＜文字列定数＞,]　＜Ｘ＞[　,　|；　＜Ｘ＞]

Ｘ：＜数値変数＞|＜文字列変数＞

【解説】

・　ＩＮＰＵＴ文を実行すると、プログラムは一時停止して、キー入力待ちになります。キー入力待ちは、Ｒｅｔｕｒｎキー（ターミナル）、または単位キー（本器パネル）が押されるまで続きます。入力が済み、Ｒｅｔｕｒｎキーを押すと、キーから入力されたデータが変数に入力されます。

・　文字列定数（プロンプト）を指定すると、画面にプロンプトを出力してから入力待ちになります。

・　ＩＮＰＵＴ文では、数値変数、文字列変数のいずれも使えますが、数値変数を入力しようとしているときに数字以外の文字（英文字、英記号など）を入力すると、その文字は無視されます。もし数字が１文字もないときは、０が変数に入力されます。また、Ｒｅｔｕｒｎキーのみを押したときには変数への入力は行いません。つまり、ＩＮＰＵＴ文の前の値がそのまま残ります。

【例】

```
ＩＮＰＵＴ　Ａ
ＩＮＰＵＴ　”Ｂ＝　”，Ｂ
ＩＮＰ　Ｃ，Ｄ$
ＩＮＰ　”ｎａｍｅ　＆　Ｎｏ　？”，ＮＡ$，Ｎ
```

【注意】

・ 文字列を入力するときは、引用符（”）で囲む必要はありません。

・ 本体パネルからは文字列の入力はできません。（数値の入力のみ本体パネルのテンキーで行えます。）

【プログラム例】

```
INPUT "Center Freq. (MHz)? ", CF
INP "Span Freq. (MHz)? ", SF
INP "No. ? ", A$
OUTPUT 31;"CF", CF, "MZ"
OUTPUT 31;"SP", SF, "MZ"
PRINT "No. :", A$
PRINT "CF :", CF, "MHz"
PRINT "SF :", SF, "MHz"
```

【実行結果】

```
Center Freq. (MHz)? 12
Span Freq. (MHz)? 30
No. ? 5
No. :     5
CF  :     12.0      MHz
SF  :     30.0      MHz
```

（２８）Ｌ　Ｉ　Ｓ　Ｔ

【概要】

外部端末または本体の画面にプログラム・リストを出力します。

【書式】

Ｌ Ｉ Ｓ Ｔ　［＜ラベル＞［，］］

【解説】

・　画面にラベルで指定した位置のプログラム・リストを出力します。

・　ラベルを指定し、その後にカンマ（，）を指定すると、そのラベル位置からプログラムの最後までを出力します。

・　リストの出力は、一度ＢＡＳＩＣプログラムを転送し(ＭＯＶＥ ａｎｄ ＲＵＮ)、プログラムを停止させ、ＢＡＳＩＣ ｍｏｄｅでＬＩＳＴ命令を行って下さい。また、プログラム中に置くこともできます。

【例】

```
Ｌ Ｉ Ｓ Ｔ

Ｌ Ｉ Ｓ Ｔ　＊ＡＢＣ，
```

【注意】

・　行番号指定でも構いませんが書式が上記と異なります。

Ｌ Ｉ Ｓ Ｔ　［＜出力開始行＞］［，［＜出力終了行＞］］

例）

```
Ｌ Ｉ Ｓ Ｔ　１００
Ｌ Ｉ Ｓ Ｔ　１００，
Ｌ Ｉ Ｓ Ｔ　，２００
Ｌ Ｉ Ｓ Ｔ　１００，２００
```

・　電源ＯＮ時に外部端末が接続されていると外部端末に出力されます。接続されていなければ本体画面に出力されます。

【プログラム例】

エディタのＭＯＶＥ ａｎｄ ＲＵＮを実行し、コントロールＣにてプログラムを停止させ、エディタのＢＡＳＩＣ ｍｏｄｅでＬＩＳＴを実行します。

【実行結果】

画面にリストを出力します。

（２９）Ｌ　Ｉ　Ｓ　Ｔ　Ｎ

【概要】

外部端末、または本体の画面にプログラム・リストを出力します。

【書式】

ＬＩＳＴＮ　［＜ラベル＞］［，＜行数＞］

【解説】

・　画面にラベルで指定した位置のプログラム・リストを出力します。

・　ラベルを指定し、ラベルのあとにカンマ（，）行数を指定すると、そのラベル位置から行数分を出力します。

・　ラベルを省略し、＜行数＞を指定すると、＜行数＞が正の値ならばプログラムの先頭から行数分、負の値ならばプログラムの最終行から行数分出力されます。

・　出力の方法は（２８）ＬＩＳＴ命令と同じです。

【例】

```
ＬＩＳＴＮ
ＬＩＳＴＮ　＊ＡＢＣ，１０
ＬＩＳＴＮ　＊ＡＢＣ，－１０
ＬＩＳＴＮ　，２０
ＬＩＳＴＮ　，－２０
```

【注意】

・　行番号指定でも構いませんが書式が上記と異なります。

ＬＩＳＴＮ　［＜出力開始行＞］［，［＜行数＞］］

例）

```
ＬＩＳＴＮ　１００
ＬＩＳＴＮ　１００，１５
ＬＩＳＴＮ　１００，－１５
ＬＩＳＴＮ　，２０
ＬＩＳＴＮ　，－２０
```

・　電源ＯＮ時に外部端末が接続されていると外部端末に出力されます。接続されていなければ本体画面に出力されます。

（30）ＬＬＩＳＴ

【概要】

シリアルＩ／Ｏポート（ＲＳ－２３２Ｃ）に接続されている装置にプログラム・リストを出力します。

【書式】

ＬＬＩＳＴ　［＜ラベル＞［，］］

【解説】

・　シリアルＩ／Ｏポート（ＲＳ－２３２Ｃ）にラベルで指定した位置のプログラム・リストを出力します。

・　ラベルを指定し、その後にカンマ（，）を指定すると、そのラベル位置からプログラムの最後までを出力します。

・　リストの出力は、一度ＢＡＳＩＣプログラムを転送し(ＭＯＶＥ ａｎｄ ＲＵＮ)、プログラムを停止させ、ＢＡＳＩＣ ｍｏｄｅでＬＬＩＳＴ命令を行って下さい。また、プログラム中に置くこともできます。

【例】

```
ＬＬＩＳＴ

ＬＬＩＳＴ　＊ＡＢＣ，
```

【注意】

・　行番号指定でも構いませんが書式が上記と異なります。

ＬＬＩＳＴ　［＜出力開始行＞］［，［＜出力終了行＞］］

例）

```
ＬＬＩＳＴ　１００
ＬＬＩＳＴ　１００，
ＬＬＩＳＴ　，２００
ＬＬＩＳＴ　１００，２００
```

・　シリアルＩ／Ｏポート（ＲＳ２３２Ｃ）の設定は、（８）のＣＯＮＴＲＯＬ命令を参照して下さい。

【プログラム例】

エディタのＭＯＶＥ ａｎｄ ＲＵＮを実行し、コントロールＣにてプログラムを停止させ、エディタのＢＡＳＩＣ ｍｏｄｅでＬＬＩＳＴを実行します。

【実行結果】

画面にリストを出力します。

（３１）ＬＬＩＳＴＮ

【概要】

シリアルＩ／Ｏポート（ＲＳ－２３２Ｃ）に接続されている装置にプログラム・リストを出力します。

【書式】

ＬＬＩＳＴＮ　［＜ラベル＞］［，＜行数＞］

【解説】

・　シリアルＩ／Ｏポート（ＲＳ－２３２Ｃ）にラベルで指定した位置のプログラム・リストを出力します。

・　ラベルを指定し、ラベルの後にカンマ（，）行数を指定すると、そのラベル位置から行数分を出力します。

・　ラベルを省略し、＜行数＞を指定すると、＜行数＞が正の値ならばプログラムの先頭から行数分出力されます。負の値ならばプログラムの最終行から行数分出力されます。

・　出力方法は（３０）ＬＬＩＳＴ命令と同じです。

【例】

```
ＬＬＩＳＴＮ

ＬＬＩＳＴＮ　＊ＡＢＣ，１０

ＬＬＩＳＴＮ　＊ＡＢＣ，－１０

ＬＬＩＳＴＮ　，２０

ＬＬＩＳＴＮ　，－２０
```

【注意】

・ 行番号指定でも構いませんが書式が上記と異なります。

ＬＬＩＳＴＮ　[＜出力開始行＞] [, [＜行数＞]]

例）

```
ＬＬＩＳＴＮ　１００
ＬＬＩＳＴＮ　１００，１５
ＬＬＩＳＴＮ　１００，－１５
ＬＬＩＳＴＮ　，２０
ＬＬＩＳＴＮ　，－２０
```

・ シリアルＩ／Ｏポート（ＲＳ－２３２Ｃ）の設定は、（８）ＣＯＮＴＲＯＬ命令を参照して下さい。

（３２）ＬＯＣＡＬ

【概要】

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

【書式】

ＬＯＣＡＬ　［装置アドレス　{，装置アドレス}］

装置アドレス：０～３０

【解説】

・　装置アドレスを指定せずにＬＯＣＡＬ命令だけを実行した場合、ＧＰＩＢのリモート・イネーブル（ＲＥＮ）ラインが偽（ｆａｌｓｅ）となり、ＧＰＩＢ上のすべての装置がローカル状態になります。
ＲＥＮラインが偽（ｆａｌｓｅ）のときは、ＯＵＴＰＵＴ命令でのＧＰＩＢ装置への設定は不可能となります。ＲＥＮラインを真（ｔｕｒｅ）にするには（５３）のＲＥＭＯＴＥ命令を実行して下さい。

・　ＬＯＣＡＬ命令に続いて、装置アドレスを指定した場合、指定した装置アドレスの装置のみ、リモート状態を解除します。装置アドレスはカンマ（，）で区切ると複数指定できます。

【例】

```
LOCAL

LOCAL  2

LOCAL  3，4，5
```

【注意】

・　スレーブ・モードでは機能しません。

【プログラム例】

```
REMOTE  11
WAIT  2000
LOCAL  11
```

【実行結果】

装置アドレス１１をリモート状態にし（リモート・ランプがあれば点灯する）、２秒後にＬＯＣＡＬ状態になります（リモート・ランプが消灯する）。

(33) 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。

【書式】

LOCAL LOCKOUT

【解説】

・ ＧＰＩＢ上の装置がリモート状態のとき（本器コントローラにリモート・コントロールされているとき）は、各装置のパネル・キーはロックされ、パネルからデータの設定はできないようになっています。しかし、ローカル・キーだけはロックされずにいます。このキーを押すと各装置は自分自身をローカル状態にして、データの設定を可能にします。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ってしまいます。このとき、ＬＯＣＡＬ　ＬＯＣＫＯＵＴ命令を実行すると、ＧＰＩＢ上の全装置のローカル・キーをロックして、各装置のパネル面からの設定を完全に禁止します。

・ ＬＯＣＡＬ　ＬＯＣＫＯＵＴ命令を実行すると、ＧＰＩＢ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。

【例】

LOCAL LOCKOUT

【注意】

・ ローカル・ロックアウト状態の解除は、ＬＯＣＡＬ命令で行います。

・ スレーブ・モードでは機能しません。

【プログラム例】

```
REMOTE 11,12
LOCAL LOCKOUT

LOCAL
```

【実行結果】

装置アドレス１１と１２のローカル・キーが効かなくなり、ＬＯＣＡＬ命令を実行すると解除できます。

（３４）ＬＰＲＩＮＴ

【概要】

ＲＳ－２３２Ｃで接続されている装置（外部端末、プリンタ等）に数値、文字列を出力します。

【書式】

ＬＰＲＩＮＴ　[＜Ｘ＞　{[　，｜；　＜Ｘ＞]}]

Ｘ：数値表現式｜文字列表現式

【解説】

・　数値表現式、文字列表現式は、セミ・コロン（；）またはカンマ（，）で区切って複数にわたり指定できます。

・　ＬＰＲＩＮＴ文は、必ず改行となります。

【例】

```
ＳＳ＝”ＤＥＦ”                    :  ①
ＬＰＲＩＮＴ　”ＡＢＣ”            :  ②
ＬＰＲＩＮＴ　ＳＳ                 :  ③
ＬＰＲＩＮＴ　”Ａ＝”，Ａ           :  ④
ＬＰＲＩＮＴ　”ＣＦ”，ＣＦＱ，”ＫＨｚ”  :  ⑤
ＬＰＲＩＮＴ　Ａ＋１００           :  ⑥
```

① 文字列変数ＳＳに”ＤＥＦ”を入力
② ”ＡＢＣ”を改行しないで出力
③ 文字列変数ＳＳを出力
④ 文字列と、数値変数を出力
⑤ 文字列と、数値変数を出力
⑥ 数値変数に１００を加え出力

【注意】

・　外部端末を接続して使うときは、ＬＲＰＩＮＴ命令は、ＰＲＩＮＴ文と同等な働きをします。

・　ＬＰＲＩＮＴ文の最後にセミ・コロン（；）を指定しても、改行となります。

【プログラム例】

```
FOR  I＝1  TO  10        ：Iが1からI＞10になるまでループ
 LPRINT  "I＝", I        ：Iをシリアル I／Oポートに出力
NEXT  I                  ：NEXT  I
```

【実行結果】

外部端末が接続されていると、以下のように出力されます。

```
I＝          1．0
I＝          2．0
I＝          3．0
I＝          4．0
I＝          5．0
I＝          6．0
I＝          7．0
I＝          8．0
I＝          9．0
I＝          10．0
```

（35）ＬＰＲＩＮＴ　ＵＳＩＮＧ（ＵＳＥと省略可）

【概要】

数値、文字列などを編集し、ＲＳ－２３２Ｃへ出力します。

【書式】

ＬＰＲＩＮＴ　ＵＳＩＮＧ　＜イメージ仕様＞；　{，＜Ｘ＞}

ＬＰＲＩＮＴ　ＵＳＥ　＜イメージ仕様＞；　{，＜Ｘ＞}

Ｘ：数値表現式｜文字列表現式

【解説】

・　シリアルＩ／Ｏポート（ＲＳ－２３２Ｃ）に数値、文字列を編集しＡＳＣＩＩコードで出力します。

・　イメージ仕様の指定は、文字列表現式にてイメージ仕様をカンマ（，）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。）

【イメージ仕様一覧】

Ｄ	指定フィールドの余った部分にスペースを出力する
Ｚ	指定フィールドの余った部分に０を詰める
Ｋ	数値をそのまま出力する
Ｓ	常に＋／－の符号を付ける
Ｍ	－の符号を付けるか値が正のときはスペースを付ける
．（小数点）	小数点を出力する
Ｅ	指数形式（ｅ、符号、指数）で出力する
Ｈ	数値、文字列をそのまま出力するが、小数点（．）がヨーロッパ・タイプ（，）になる
Ｒ	ヨーロッパ・タイプの小数点になる
＊	指定フィールドの余った部分に＊印を出力する
Ａ	１文字、出力する
ｋ	文字列をそのまま出力する
Ｘ	スペースを出力する
リテラル	書式指定にリテラルを書くときは、　＼”　で囲む
Ｂ	数値をＡＳＣＩＩコードとして出力する
＠	改ページする
＋	出力位置を同じ行の先頭に移動させる
－	出力位置を次の行に移動させる
＃	最後に改行しない
ｎ	ｎ桁の精度で出力する。文字列に対して指定すると、出力の値は、その文字列の長さになる

【例】

```
LPRINT USING "DDD.DD"; 1.2

LPRINT USE "ZZZ.ZZ"; 1.2
```

【注意】

・ LPRINT USING命令の出力は、復改はされません。改行だけ行うので復改をLPRINT USING中に入れて下さい。

```
LPRINT USING "K,k";123,"\r"
```

・ イメージ仕様で指定した数がパラメータで指定した数より多い場合エラーになりプログラムの実行を中止します。
(Unmatched IMAGE-spec in USING)

【プログラム例】

```
LPRINT USING "DDD.DD,k"; 1.2,"\r"
LPRINT USE "ZZZ.ZZ,k"; 1.2,"\r"
LPRINT USE "***.ZZ,k"; 1.2,"\r"
A$="K,5X,B,DDDRDD,k"
LPRINT USE A$; 2.34,65,1.2,"\r"
```

【実行結果】

```
1.20
001.20
**1.20
2.34        A    1,20
```

（３６）ＯＦＦ　ＥＮＤ

【概要】

ＯＮ　ＥＮＤ命令で定義した分岐先を解除します。

【書式】

ＯＦＦ　ＥＮＤ　＜＃ファイル・ディスクリプタ＞

【解説】

・　ＯＮ　ＥＮＤ命令で＜＃ファイル・ディスクリプタ＞に定義した分岐先を解除します。

【例】

ＯＦＦ　ＥＮＤ　＃ＦＤ

【注意】

・　この命令によって解除したあとに、ＥＮＴＥＲ　＃命令を使用してＥＯＦを読み込むとエラーになり、プログラムの実行を中止します。

・　分岐先の定義は（４０）ＯＮ　ＥＮＤ命令を使います。

（３７）ＯＦＦ　ＥＲＲＯＲ

【概要】

ＯＮ　ＥＲＲＯＲ命令で定義した分岐先を解除します。

【書式】

ＯＦＦ　ＥＲＲＯＲ

【解説】

・　ＯＮ　ＥＲＲＯＲ命令で定義した分岐先を解除します。

【例】

```
ＯＦＦ　ＥＲＲＯＲ
```

【注意】

・　この命令によって解除したあとに、プログラムを実行中エラーが発生すると、実行を中止します。

・　分岐先の定義は（４１）ＯＮ　ＥＲＲＯＲ命令を使います。

（３８）ＯＦＦ　ＫＥＹ

【概要】

ＯＮ　ＫＥＹ命令で定義した分岐先を解除します。

【書式】

ＯＦＦ　ＫＥＹ　＜キー番号＞

キー番号：数値表現式（１，２，３，４，５，６，７，８，９のいずれか）

【解説】

・　ＯＮ　ＫＥＹ命令で＜キー番号＞に定義した分岐先を解除します。

・　＜キー番号＞による分岐先は、それぞれ個別に定義を解除できます。

【例】

```
ＯＦＦ　ＫＥＹ　１
ＯＦＦ　ＫＥＹ　２
ＯＦＦ　ＫＥＹ　３
ＯＦＦ　ＫＥＹ　４
ＯＦＦ　ＫＥＹ　５
ＯＦＦ　ＫＥＹ　６
```

【注意】

・　この命令によって解除したあとに、キーを押しても分岐しません。

・　分岐先の定義は（４２）ＯＮ　ＫＥＹ命令を使います。

【プログラム例】

```
ON KEY 1 GOSUB *K1      ：キー番号１の分岐先を定義（サブ・ルーチン）
ON KEY 2 GOTO *K2       ：キー番号２の分岐先を定義（ジャンプ）
!
ENABLE INTR             ：割り込み受信許可
*L                      ：ループ
GOTO *L                 ：  〃
!
*K1                     ：キー番号１の割り込み処理
PRINT "KEY 1"           ：”ＫＥＹ　１”を出力
OFF KEY 1               ：キー番号１の分岐先の定義を解除
ON KEY 2 GOTO *K2       ：キー番号２の分岐先を定義（ジャンプ）
RETURN                  ：リターン
!
*K2                     ：キー番号２の割り込み処理
PRINT "KEY 2"           ：”ＫＥＹ　１”を出力
OFF KEY 2               ：キー番号２の分岐先の定義を解除
ON KEY 1 GOSUB *K1      ：キー番号１の分岐先を定義（サブ・ルーチン）
GOTO *L                 ：＊Ｌへジャンプ
```

【実行結果】

```
ＫＥＹ　１
ＫＥＹ　２
ＫＥＹ　１
ＫＥＹ　２
    ・
    ・
    ・
```

同じキーを２回押しても、同じものが出力されません。

(39) ＯＦＦ　ＳＲＱ／ＩＳＲＱ

【概要】

ＯＮ　ＳＲＱ命令、またはＯＮ　ＩＳＲＱ命令で定義した分岐先を解除します。

【書式】

ＯＦＦ　ＳＲＱ

ＯＦＦ　ＩＳＲＱ

【解説】

・ ＯＮ　ＳＲＱ命令、またはＯＮ　ＩＳＲＱ命令で定義した分岐先を解除します。

【例】

```
ＯＦＦ　ＳＲＱ
ＯＦＦ　ＩＳＲＱ
```

【注意】

・ 分岐先の定義は（４３）ＯＮ　ＳＲＱ命令、またはＯＮ　ＩＳＲＱ命令を使います。

（４０）ＯＮ　ＥＮＤ　～
ＧＯＴＯ／ＧＯＳＵＢ

【概要】

ファイルのＥＯＦ（Ｅｎｄ　Ｏｆ　Ｆｉｌｅ）発生時に処理する分岐先を定義します。

【書式】

ＯＮ　ＥＮＤ　＜＃ファイル・ディスクリプタ＞　ＧＯＴＯ　＜ラベル＞

ＯＮ　ＥＮＤ　＜＃ファイル・ディスクリプタ＞　ＧＯＳＵＢ　＜ラベル＞

【解説】

・　ＥＮＴＥＲ　＃命令ファイルを読み込みで、ファイルの終わり（ＥＯＦ）を検出したときに分岐する分岐先を定義します。

・　ＯＮ　ＥＮＤ文にてＥＯＦ時の処理を定義しないで次々にデータを読み込むと、ＥＯＦ読み込み時にエラーとなり、実行を中止します。
（ｅｎｄ　ｏｆ　＜ファイル名＞　ｆｉｌｅ）

【例】

ＯＮ　ＥＮＤ　＃ＦＤ　ＧＯＴＯ　＊ＥＯＦ

ＯＮ　ＥＮＤ　＃ＦＤ　ＧＯＳＵＢ　＊ＥＯＦ

【注意】

・　分岐は、ＥＯＦが読み込まれたときに行われます。また、ＧＯＳＵＢ文での定義で分岐した場合の戻り先は、ＥＯＦが読み込まれたＥＮＴＥＲ命令の次の命令からの実行になります。

・　ＥＮＡＢＬＥ　ＩＮＴＲ／ＤＩＳＡＢＬＥ　ＩＮＴＲ命令に関係なく定義した分岐先にジャンプします。

・　分岐先定義の解除は(３６)ＯＦＦ　ＥＮＤ命令で行います。

・　データの書き込み、読み込みは、(４５)ＯＵＴＰＵＴ　＃、(１４)ＥＮＴＥＲ　＃を参照して下さい。

【プログラム例】

```
OPEN "FFF" FOR OUTPUT AS #FD      :”ＦＦＦ”のファイル名で書き込みオープン
FOR I=100 TO 200                  :Ｉを１００から２００までループ
  OUTPUT #FD;I                    :Ｉのデータをセーブ
NEXT I                            :ＮＥＸＴ　Ｉ
CLOSE #FD
!
ON END #FR GOTO *LA               :ＥＯＦの割り込み分岐先を定義
OPEN "FFF" FOR INPUT AS #FR       :”ＦＦＦ”のファイル名で読み出しオープン
*LOOP                             :
 ENTER #FR;N                      :ファイルからデータをロード、Ｎへ入力
 PRINT N                          :Ｎを画面出力
 GOTO *LOOP                       :ラベル　＊ＬＯＯＰへジャンプ
!
*LA                               :
 CLOSE #FR                        :ファイル・クローズ
 PRINT "EOF"                      :”ＥＯＦ”を画面出力
 STOP                             :プログラム終了
```

【実行結果】

```
１００．０
１０１．０
１０２．０
    ・
    ・    （省略）
    ・
１９９．０
２００．０
ＥＯＦ
```

（41）ＯＮ　ＥＲＲＯＲ　～　ＧＯＴＯ／ＧＯＳＵＢ

【概要】

ＢＡＳＩＣプログラムの実行中にエラーが発生したときの分岐先を定義します。

【書式】

ＯＮ　ＥＲＲＯＲ　ＧＯＴＯ　＜ラベル＞

ＯＮ　ＥＲＲＯＲ　ＧＯＳＵＢ　＜ラベル＞

【解説】

・　ＢＡＳＩＣプログラム実行中、エラー発生時の分岐先を定義します。

・　この命令を実行しておくと、エラーが起こったときに指定した＜ラベル＞位置へプログラムの実行を移します。

・　ビルトイン関数実行時のエラー処理に使用できます。

【例】

ＯＮ　ＥＲＲＯＲ　ＧＯＴＯ　＊ＥＲＲ

ＯＮ　ＥＲＲＯＲ　ＧＯＳＵＢ　＊ＥＲＲ

【注意】

・　分岐は、エラーが発生したときに行われます。また、ＧＯＳＵＢ文での定義で分岐した場合、サブ・ルーチンからの戻り先はエラーが発生した命令の次の命令からになります。

・　ＥＮＡＢＬＥ　ＩＮＴＲ／ＤＩＳＡＢＬＥ　ＩＮＴＲ　命令に関係なく定義した分岐先にジャンプします。

・　分岐先定義の解除は（３７）ＯＦＦ　ＥＲＲＯＲ命令を使います。

【プログラム例】

```
ON ERROR GOTO *ERR          :エラー割り込み分岐先定義
PRINT "START"               :" ＳＴＡＲＴ" を画面出力
PRINT 1/0                   :１÷０を実行
PRINT "END"                 :" ＥＮＤ" を出力
STOP                        :プログラム終了
*ERR
PRINT "<<ERROR>>"           :" ＜＜ＥＲＲＯＲ＞＞を出力
PRINT "  >>  ",ERRM$(0)     :エラー・メッセージを出力
```

【実行結果】

```
ＳＴＡＲＴ
＜＜ＥＲＲＯＲ＞＞
＞＞      ４０：  ０  ｄｉｖｉｄｅ
```

(42) ON KEY ～ GOTO／GOSUB

【概要】

フル・キーボード、または本体パネル（テンキー）からの割り込みを受信したときの分岐先を定義します。

【書式】

ON KEY ＜キー番号＞ GOTO ＜ラベル＞

ON KEY ＜キー番号＞ GOSUB ＜ラベル＞

キー番号：数値表現式（1，2，3，4，5，6，7，8，9のいずれか）

【解説】

・ フル・キーボード、または本体パネル（テンキー）による割り込みの分岐先を定義します。

・ 使用できるキーは、1，2，3，4，5，6，7，8，9までの数値キー、ON KEY ＜キー番号＞で定義できるキー番号も1～9までです。

・ 外部端末のテンキーを使用する前に、必ず外部端末のテンキーの設定を変更して下さい。

【例】

```
ON KEY 1 GOTO *K1
ON KEY 2 GOSUB *K2
ON KEY 3 GOTO *K3
ON KEY 4 GOSUB *K4
ON KEY 5 GOTO *K5
ON KEY 6 GOSUB *K6
```

【注意】

・ 分岐は、割り込みが発生したときに実行していた命令の処理が終了してから行われます。また、GOSUB文での定義で分岐した場合、サブ・ルーチンからの戻り先は割り込みが発生したときに実行していた命令の次の命令からになります。

・ 割り込みの受信の許可、禁止は（13）ENABLE INTR、（12）DISABLE INTRを参照して下さい。

・ 分岐先定義の解除は（38）OFF KEY命令で行います。

【プログラム例】

```
ON KEY 1 GOTO *SC
ON KEY 2 GOTO *SD
ON KEY 3 GOTO *SE
ON KEY 4 GOSUB *SF
ON KEY 5 GOSUB *SG
ON KEY 6 GOSUB *RA
ENABLE INTR
*LOOP
 GOTO *LOOP
STOP

*SC:BUZZER 261,300:GOTO *LOOP
*SD:BUZZER 294,300:GOTO *LOOP
*SE:BUZZER 330,300:GOTO *LOOP

*SF
 DISABLE INTR
 BUZZER 349,300
 ENABLE INTR
 RETURN

*SG
 DISABLE INTR
 BUZZER 392,300
 ENABLE INTR
 RETURN

*SA
 DISABLE INTR
 BUZZER 440,300
 ENABLE INTR
 RETURN
```

【実行結果】

1，2，3，4，5，6 を ド、レ、ミ、ファ、ソ、ラ と対応させて音を出します。

(43) ＯＮ　ＳＲＱ／ＩＳＲＱ　～ ＧＯＴＯ／ＧＯＳＵＢ

【概要】

本器のＧＰＩＢコントローラがＳＲＱ（サービス・リクエスト）を受信したときの分岐先を定義します。

【書式】

ＯＮ　ＳＲＱ　ＧＯＴＯ　＜ラベル＞
ＯＮ　ＳＲＱ　ＧＯＳＵＢ　＜ラベル＞
（外部ＧＰＩＢからのＳＲＱによる分岐先定義）

ＯＮ　ＩＳＲＱ　ＧＯＴＯ　＜ラベル＞
ＯＮ　ＩＳＲＱ　ＧＯＳＵＢ　＜ラベル＞
（本体計測部からのＳＲＱよる分岐先定義）

【解説】

・　ＳＲＱ（サービス・リクエスト）による割り込みの分岐先を定義します。

・　ＯＮ　ＳＲＱは、外部ＧＰＩＢ装置からのＳＲＱ（サービス・リクエスト）の分岐先を定義します。

・　ＯＮ　ＩＳＲＱは、本体計測部からのＳＲＱ（サービス・リクエスト）の分岐先を定義します。

【例】

```
ＯＮ　ＳＲＱ　ＧＯＴＯ　＊ＳＳ
ＯＮ　ＳＲＱ　ＧＯＳＵＢ　＊ＳＳ
ＯＮ　ＩＳＲＱ　ＧＯＴＯ　＊ＩＳ
ＯＮ　ＩＳＲＱ　ＧＯＳＵＢ　＊ＩＳ
```

【注意】

・　ＯＮ　ＩＳＲＱ命令を使用するときには、計測部にＳＲＱ割り込みを発生させる必要があります。ＳＲＱを発生させるには以下のＧＰＩＢコードを使用します。
ＯＵＴＰＵＴ３１命令にて実行して下さい。

ＧＰＩＢコード	内　　容
Ｓ０	計測部からコントローラに対しＳＲＱ割り込みを送信する
Ｓ１	計測部からコントローラに対しＳＲＱ割り込みを送信しない
Ｓ２	ステータス・バイトをクリアする

・ 分岐は、割り込みが発生したときに実行していた命令の処理が終了してから行われます。また、ＧＯＳＵＢ文での定義で分岐した場合、サブ・ルーチンからの戻り先は割り込みが発生したときに実行していた命令の次の命令からになります。

・ 割り込みの受信の許可、禁止は、（１３）のＥＮＡＢＬＥ　ＩＮＴＲ、（１２）ＤＩＳＡＢＬＥ　ＩＮＴＲを参照して下さい。

・ 分岐先定義の解除は（３９）ＯＦＦ　ＳＲＱ／ＩＳＲＱで行います。

【プログラム例】

例１

```
TGT=7
ON SRQ GOSUB *SS
ENABLE INTR
*MAINLOOP
 GOSUB *BEEP
 GOTO *MAINLOOP

*BEEP
 BUZZER 440,20
 WAIT 200
 RETURN

*SS
 DISABLE INTR
 S=SPOLL(TGT)
 PRINT "SPOLL",S
 ENABLE INTR
 RETURN
```

例２

```
INTEGER S
ON ISRQ GOTO *SS
*ST
 OUTPUT 31;"S0 SI"
 ENABLE INTR
 *MAINLOOP
  GOTO *MAINLOOP

*SS
 S=SPOLL(31)
 IF S BAND 4 THEN
    PRINT "MAX :",MAX(0,700,0)
    PRINT "MIN :",MIN(0,700,0)
 GOTO *ST
```

【実行結果】

例1

外部ＧＰＩＢ装置からのＳＲＱ（サービス・リクエスト）により分岐し、シリアル・ポールの結果を出力します。

例2

本体計測部にて、掃引終了ごとにＭＡＸ／ＭＩＮレベルを求めます。

(44) 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。

【書式】

ＯＰＥＮ　”ファイル名”　ＦＯＲ　＜処理モード＞　ＡＳ　＜＃ファイル・ディスクリプタ＞　［；＜タイプ＞］

＜処理モード＞

ＯＵＴＰＵＴ	：書き込み
ＩＮＰＵＴ	：読み出し

＜タイプ＞

ＢＩＮＡＲＹ	：バイナリ
ＴＥＸＴ	：ＡＳＣＩＩコード
ＡＳＣＩＩ	：ヘッダ付きＡＳＣＩＩコード

【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、指定したモードでオープンします。

・ 処理モードには　ＯＵＴＰＵＴ　と　ＩＮＰＵＴ　があります。
　ＯＵＴＰＵＴ　はファイルにデータを書き込むとき
　ＩＮＰＵＴ　は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／ＯＵＴＰＵＴ命令を使いますが、これらの命令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。

・ タイプ

タイプには、ＢＩＮＡＲＹ，ＴＥＸＴ，ＡＳＣＩＩの３種類があります。（タイプの指定がない場合は、ＢＩＮＡＲＹとなります。）
（４５）ＯＵＴＰＵＴ　＃参照

【例】

```
OPEN "FILE1" FOR OUTPUT AS #FD
OPEN "FILE2" FOR OUTPUT AS #FD;BINARY
OPEN "FILE3" FOR OUTPUT AS #FF;TEXT
OPEN "FILE4" FOR OUTPUT AS #FA;ASCII

OPEN "FILE1" FOR INPUT AS #FD
OPEN "FILE2" FOR INPUT AS #FD;BINARY
OPEN "FILE3" FOR INPUT AS #FF;TEXT
OPEN "FILE4" FOR INPUT AS #FA;ASCII
```

【注意】

・ すで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、エラーになりプログラムの実行を中止します。
（”ファイル名”　ｆｉｌｅ　ｉｓ　ａｌｒｅａｄｙ　ｅｘｉｓｔ．）
（”ファイル名”　file is already with another PATH.）

・ ファイルをオープンしたときには、最後にクローズを必ず実行して下さい。

・ １つのファイルを複数のファイル・ディスクリプタでオープンできません。

・ すでにメモリ・カードに存在するファイルをＯＵＴＰＵＴ（書き込み）モードでオープンすると、エラーになりプログラムの実行を中止します。
これは誤って必要なファイルを消してしまうのを防ぐためです。
新しくファイルを作成し直すには、ＰＵＲＧＥ命令でファイルを削除しておきます。

【プログラム例】

```
OPEN "B" FOR OUTPUT AS #FA;BINARY
FOR I=1 TO 10
 OUTPUT #FA;I
NEXT I
CLOSE #FA

OPEN "BB" FOR OUTPUT AS #FB;TEXT
FOR I=1 TO 10
 OUTPUT #FB;I
NEXT I
CLOSE #FB

OPEN "BBB" FOR OUTPUT AS #FC;ASCII
FOR I=1 TO 10
 OUTPUT #FC;I
NEXT I
CLOSE #FC
```

(45) OUTPUT

【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出力（書き込み）します。

【書式】

OUTPUT ＜#ファイル・ディスクリプタ＞ ； ＜X＞ [, ＜X＞]

X：出力項目（数値表現式，文字列表現式）

【解説】

・ OPEN命令で指定したタイプにより以下の書式で書き込まれます。

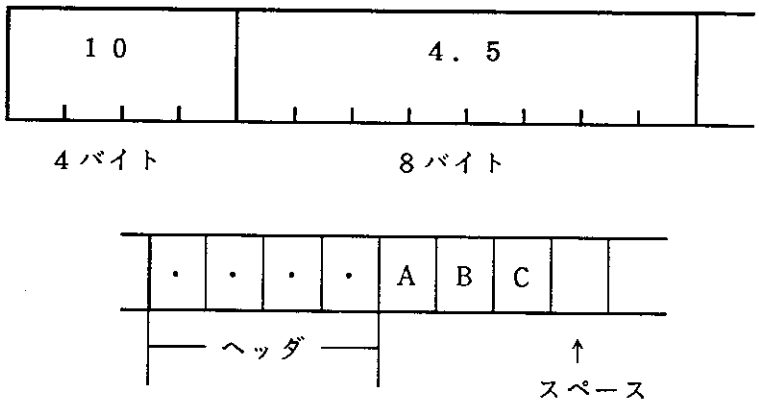
BINARY タイプ

データを内部表現と同じ型で出力（書き込み）します。

整数型 ： ４バイト
実数型 ： ８バイト
文字列 ： ４バイトのヘッダ＋文字列数バイト
（ただし、文字列数が奇数の場合は、文字列の最後にスペースを付ける。）

例）

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

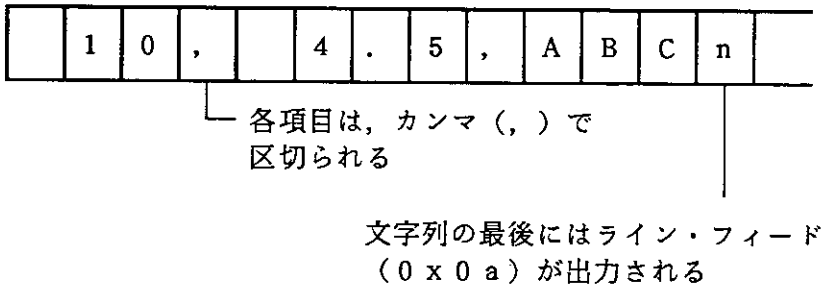


TEXT タイプ

データをＡＳＣＩＩコードに変換して出力します。数値データは、スペースか符号がデータの先頭に付きます。文字列データの最後にはライン・フィード（０ｘ０ａ）が付きます。

例）

```
OPEN "FILE" FOR OUTPUT AS #FD;TEXT
OUTPUT #FD;10, 4.5,"ABC"
```

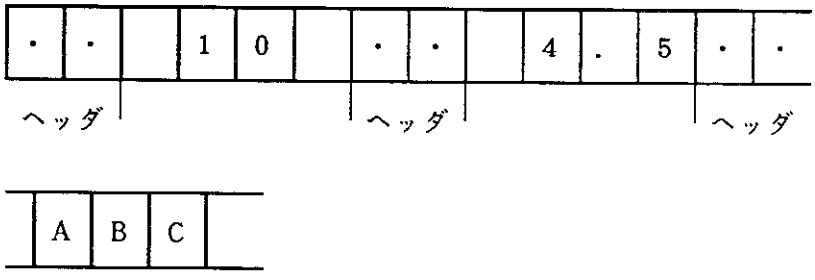


ASCII タイプ

データをＡＳＣＩＩコードに変換して出力します。数値データは、スペースか符号がデータの先頭に付きます。
各項目の先頭にはヘッダが２バイト付きます。
データ数が奇数の場合には最後にスペースが付きます。

例）

```
OPEN "FILE" FOR OUTPUT AS #FD;ASCII
OUTPUT #FD;10, 4.5,"ABC"
```



【例】

```
OUTPUT #FD;"ABC"

OUTPUT #FD;A, B, C
```

【注意】

- ファイル・ディスクリプタは、ファイル・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となるファイルに（４４）のＯＰＥＮ　＃命令でファイル・ディスクリプタを割り当てます。以後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。

- ヘッダは各項目の区切りとしての目印となり、データの長さを持ちます。

- ＯＰＥＮ命令でファイルをオープンしないで本命令を使用すると、エラーになります。（ｆｉｌｅ　ｉｓ　ＮＯＴ　ｏｐｅｎ）

【プログラム例】

```
OPEN "A" FOR OUTPUT AS #FA;BINARY
INTEGER S
FOR I=1 TO 10
 S=I
 OUTPUT #FA;I,S
NEXT I
CLOSE #FA

OPEN "AA" FOR OUTPUT AS #FB;TEXT
INTEGER S
FOR I=1 TO 10
 S=I
 OUTPUT #FB;I,S
NEXT I
CLOSE #FB

OPEN "AAA" FOR OUTPUT AS #FC;ASCII
INTEGER S
FOR I=1 TO 10
 S=I
 OUTPUT #FC;I,S
NEXT I
CLOSE #FC
```

（46）ＯＵＴＰＵＴ（ＯＵＴと省略可）

【概要】

ＧＰＩＢ、本器測定部、またはパラレルＩ／Ｏにデータを送ります。

【書式】

ＯＵＴＰＵＴ ＜Ａ {, Ａ} ｜３１｜３２＞;＜Ｘ {, Ｘ}＞

Ａ：装置アドレス　０～３０：ＧＰＩＢ接続の機器へデータ出力
３１：本器の計測部へデータ出力
３２：パラレルＩ／Ｏへデータ出力

Ｘ：数値表現式｜文字列表現式

【解説】

・ 装置アドレス(０～３０)によって指定された装置へ、数値および文字列をＡＳＣＩＩコードにしてデータを送ります。 ※１

装置アドレス（０～３０）は、カンマ（,）で区切ると複数指定できます。
数値表現式と文字列表現式は、カンマ（,）で区切ると混合して使用できます。
なお、ＲＥＮ(リモート・イネーブル)ラインが真(ｔｕｒｅ)のときにＯＵＴＰＵＴ命令の０～３０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。

・ ３１を指定しますと本器の計測部へデータを送ります。 ※１
（計測部へはトーカ／リスナのＧＰＩＢコードを送ると、外部コントローラと同じように本器を設定できます。）

・ ３２を設定すると１６ビットパラレルＩ／Ｏへデータを出力します。※２
出力できる値は、０～６５５３５までの１６ビットです。３２の指定では数値表現式のみ使用できます。文字列表現式ではエラーとなり、プログラムの実行を中止します。
（invalid type in getvi）

※１ Ｒ３２６５／３２７１シリーズの取扱説明書（別冊）にあるＧＰＩＢコード一覧を参照して下さい。

※２ 2.4節に示すパラレルＩ／Ｏを参照

【例】

```
OUTPUT 3;"CF3MZ"
OUTPUT 3,4,5;"SP",S,"MZ"

OUTPUT 31;"CF",CF,"MZ"

OUTPUT 32;128
OUTPUT 32;DT
```

【注意】

・ 0～30以外は、カンマ(,)で区切っても指定できません。
(複数のアドレスを指定するのは0～30まで)

・ 0～30指定のとき、装置アドレスが間違っていたり、指定したアドレスに装置が接続されていない場合、本命令を無視して次へ進みます。

・ スレーブ・モードで使用するときは、[4. マスタ／スレーブ・モード]の章を参照して下さい。

【プログラム例】

```
DIM S$[26],B$[20]
OUTPUT 5;"CF?"
ENTER 5;C
ENTER 5;S$
PRINT C,S$

CF=1.23
A$="CF?"
OUTPUT 31;"CF",CF,"MZ"
OUTPUT 31;A$
ENTER 31;B$
PRINT B$

INTEGER S
ENTER 32;S
IF S BAND 4 THEN OUTPUT 32;128
```

(47) PAUSE

【概要】

プログラムの実行を一時停止させます。

【書式】

PAUSE

【解説】

・ BASICプログラムの実行をBASICのプログラム自身で一時的に停止するためのコマンドです。一時停止後のプログラムの再開は、CONT命令、または外部端末の［CONT］（F20）キー、または本器パネルのCONT（ソフト・キー内）で行います。

（CONT命令は、BASIC modeで入力して下さい。）

【例】

```
PAUSE
```

【プログラム例】

```
FOR I=1 TO 5                      : カウンタIを1に初期化
  PRINT I                         : 変数Iを出力
  PRINT "Hit CONT key"            : 文字定数を出力
  PAUSE                           : PAUSE
NEXT I                            : NEXT
```

【実行結果】

変数を出力するたびに、一時停止します。

（４８）ＰＲＩＮＴＦ（ＰＲＦと省略可）

【概要】

外部端末、または本体画面へ数値、文字列を編集し出力します。

【書式】

ＰＲＩＮＴＦ　＜文字列表現式＞［，＜Ｘ＞］

Ｘ：数値表現式｜文字列表現式

【解説】

・　外部端末使用のときは外部端末画面に出力します。本体のみのときは本体画面に数値、文字列を出力します。

・　文字列表現式内に以下の書式を指定すると、引数のデータを編集し出力できます。（編集仕様は％で始まります。）

マイナス符号：編集されたパラメータをフィールド（出力領域）の左側にそろえることを指定する。
ピリオド　：フィールドの幅を示す数字列と桁数（精度）を指定する数字列との区切り。
０（ゼロ）　：余ったフィールドにゼロ・サプレスを行う。

・　編集文字とその意味

ｄ　：パラメータを１０進数に変換する
ｏ　：パラメータを符号なし８進数に変換する（先頭に０を付けない）
ｘ　：パラメータを符号なし１６進数に変換する（先頭に０ｘを付けない）
ｓ　：文字列
ｅ　：実数として受取り、［－］ｍ．ｎｎｎｎｎｎＥ［±］ｘｘの形の１０進数に変換する
（ｎの文字列の長さは精度によって指定される。標準値は６）
ｆ　：実数として受取り、［－］ｍｍｍ．ｎｎｎｎの形の１０進数に変換する
（ｎの文字列の長さは精度によって指定される。標準値は６）

・　％の後の文字は、編集文字でなければその文字が出力されます。したがって％は％％で出力します。

【例】

```
PRINTF " C=%d", C          :変数Cを10進数で出力
PRINTF " C=%5d", C         :変数Cを5桁の10進数で出力
                            (右詰め)
PRINTF " C=%-5d", C        :変数Cを5桁の10進数で出力
                            (左詰め)
PRINTF " C=%05d", C        :変数Cを5桁の10進数で出力
                            (右詰め:ゼロサプレスする)

PRINTF " H=%x", H          :変数Hを16進数で出力
PRINTF " H=%4x", H         :変数Hを4桁の16進数で出力
                            (右詰め)
PRINTF " H=%-4x", H        :変数Hを4桁の16進数で出力
                            (左詰め)
PRINTF " H=%04x", H        :変数Hを4桁の16進数で出力
                            (右詰め:ゼロサプレスする)

PRINTF " S$=%s", S$        :文字列を出力
PRINTF " S$=%8s", S$       :文字列を右詰めで8文字出力する
PRINTF " S$=%-8s", S$      :文字列を左詰めで8文字出力する
PRINTF " %20.10s", S$      :20文字のフィールド内で右詰めで
                            10文字を出力
PRINTF " %-20.10s", S$     :20文字のフィールド内で左詰めで
                            10文字を出力

PRINTF " F=%f", F          :変数Fを10進数の実数で出力
PRINTF " F=%8.2f", F       :変数Fを小数点以下2桁の全8桁で出
                            力する。小数点含む
PRINTF " F=%08.2f", F      :変数Fを小数点以下2桁の全8桁で出
                            力する。小数点含む
                            (ゼロ・サプレスする)
```

【注意】

・ PRINTF文での出力は改行されないので、改行コードをPRINTF文に入れて使用して下さい。またはPRINTF文の後にPRINT文を入れて下さい。

```
PRINTF " A=%d\n\r", A
          改行
```

・ PRINTFの編集文字とパラメータが不足していたり、型指定に誤りがあると、無意味な出力となることがあります。

【プログラム例】

```
PRINTF "%d * %4d =%05d",12,34,12*34
PRINT
PRINTF ":%-10d:%10d:",123,456
PRINT
A$="ABCD"
PRINTF ":%8s:%-8s:%8.2s:%-8.2s:",A$,A$,A$,A$
PRINT
PRINTF "%f+%8.3f - %06.3f=%e",1.23,4.56,7.89,1.23+4.56-7.89
```

【実行結果】

```
12 *   34 =00408
:123        :        456:
:    ABCD:ABCD    :       AB:AB       :
1.230000+   4.560 - 07.890=-2.100000e+00
```

（４９）ＰＲＩＮＴ（？と省略可）

【概要】

外部端末、または本体画面へ数値、文字列を出力します。

【書式】

ＰＲＩＮＴ　[＜Ｘ＞　{[　，｜；　＜Ｘ＞]}] [<，｜；>]

Ｘ：数値表現式｜文字列表現式

【解説】

・　数値表現式、文字列表現式はセミ・コロン（；）またはカンマ（，）で区切って複数にわたり指定できます。（セミ・コロンは、すぐ後に出力、カンマは、タブ間隔で出力されます。）

・　ＰＲＩＮＴ文の最後にセミ・コロン（；）を置いた場合は、出力が終わっても改行されません。してがって、次のＰＲＩＮＴ文を実行すると、以前に出力した文字に続いて出力します。

【例】

```
ＳＳ＝”ＤＥＦ”                    :  ①
ＰＲＩＮＴ　”ＡＢＣ”；            :  ②
ＰＲＩＮＴ　ＳＳ                  :  ③
ＰＲＩＮＴ　”Ａ＝”，Ａ             :  ④
ＰＲＩＮＴ　”ＣＦ”，ＣＦＱ，”ＫＨｚ”  :  ⑤
ＰＲＩＮＴ　Ａ＋１００             :  ⑥
```

① 文字列変数ＳＳに”ＤＥＦ”を入力
② 画面に”ＡＢＣ”を改行しないで出力
③ 文字列変数ＳＳを出力
④ 文字列と、数値変数を出力
⑤ 文字列と、数値変数を出力
⑥ 数値変数に１００を加え出力

【注意】

・　外部端末使用のときは外部端末画面へ、本体のみのときは本体画面に数値、文字列を出力します。

【プログラム例】

```
FOR  I=1  TO  10
  PRINT  "I=  ";
  PRINT  I
NEXT  I
```

【実行結果】

```
I=  1.0
I=  2.0
I=  3.0
I=  4.0
I=  5.0
I=  6.0
I=  7.0
I=  8.0
I=  9.0
I=  10.0
```

外部端末が接続されていれば外部端末に出力されます。接続されていなければ本体画面へ出力されます。

（５０）ＰＲＩＮＴ　ＵＳＩＮＧ（ＵＳＥと省略可）

【概要】

数値、文字列などを編集し外部端末、または本体画面へ出力します。

【書式】

ＰＲＩＮＴ　ＵＳＩＮＧ　＜イメージ仕様＞；　{，＜Ｘ＞}

ＰＲＩＮＴ　ＵＳＥ　＜イメージ仕様＞；　{，＜Ｘ＞}

Ｘ：数値表現式｜文字列表現式

【解説】

・　外部端末使用のときは外部端末画面に出力します。本体のみのときは本体画面に数値、文字列を編集し出力します。

・　イメージ仕様の指定は、文字列表現式にてイメージ仕様をカンマ（，）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。）

【イメージ仕様一覧】

Ｄ	･･･････ 指定フィールドの余った部分にスペースを出力する
Ｚ	･･･････ 指定フィールドの余った部分に０を詰める
Ｋ	･･･････ 数値をそのまま出力する
Ｓ	･･･････ 常に＋／－の符号を付ける
Ｍ	･･･････ －の符号を付けるか値が正のときはスペースを付ける
．（小数点）	･･･････ 小数点を出力する
Ｅ	･･･････ 指数形式（ｅ、符号、指数）で出力する
Ｈ	･･･････ 数値、文字列をそのまま出力するが、小数点（．）がヨーロッパ・タイプ（，）になる
Ｒ	･･･････ ヨーロッパ・タイプの小数点になる
＊	･･･････ 指定フィールドの余った部分に＊印を出力する
Ａ	･･･････ １文字、出力する
ｋ	･･･････ 文字列をそのまま出力する
Ｘ	･･･････ スペースを出力する
リテラル	･･･････ 書式指定にリテラルを書くときは、　＼”　で囲む
Ｂ	･･･････ 数値をＡＳＣＩＩコードとして出力する
＠	･･･････ 改ページする
＋	･･･････ 表示位置を同じ行の先頭に移動させる
－	･･･････ 表示位置を次の行に移動させる
＃	･･･････ 最後に改行をしない
ｎ	･･･････ ｎ桁の精度で出力する。文字列に対して指定すると、出力の値は、その文字列の長さになる

【例】

```
PRINT USING "DDD.DD"; 1.2

PRINT USE "ZZZ.ZZ"; 1.2
```

【注意】

・ PRINT USING命令の出力は、復改はされません。改行だけ行うので復改をPRINT USINGの中に入れて下さい。

```
PRINT USING "K,k"; 123,"\n"
```

・ イメージ

【プログラム例】

```
PRINT USING "DDD.DD,k"; 1.2,"\r"
PRINT USE "ZZZ.ZZ,k"; 1.2,"\r"
PRINT USE "***.ZZ,k"; 1.2,"\r"
A$="K,5X,B,DDDRDD,k"
PRINT USE A$; 2.34,65,1.2,"\r"
```

【実行結果】

```
1.20
001.20
**1.20
2.34        A   1,20
```

(51) ＰＲＩＮＴＥＲ

【概要】

ＧＰＩＢプリンタに出力するためのＧＰＩＢアドレスを指定します。

【書式】

ＰＲＩＮＴＥＲ ＜ＧＰＩＢアドレス＞

＜ＧＰＩＢアドレス＞： ０～３０ まで

【解説】

・ ＧＰＲＩＮＴ，ＧＬＩＳＴ，ＧＬＩＳＴＮ 等の命令の出力先のＧＰＩＢアドレスを指定します。（ＧＰＲＩＮＴ，ＧＬＩＳＴ，ＧＬＩＳＴＮ 等の命令を実行する前に、必ずＰＲＩＮＴＥＲ命令を実行して下さい。）

・ ＧＰＩＢアドレスの指定は、０～３０までです。（デフォルト値は、３１で設定してあるので、使用する際には必ずＰＲＩＮＴＥＲ命令で０～３０までのＧＰＩＢアドレスに毎回設定して下さい。）

【例】

```
ＰＲＩＮＴＥＲ 1

ＰＲＩＮＴＥＲ 5
```

【注意】

・ デフォルト値は３１で設定してありますが、ＯＵＴＰＵＴ／ＥＮＴＥＲ命令のアドレス３１とは関係がありません。

・ アドレスが設定されていないと出力されません。

【プログラム例】

```
ＰＲＩＮＴＥＲ 1                    ： 出力先をＧＰＩＢアドレス１に設定
ＧＰＲＩＮＴ " ＡＢＣＤ"            ： アドレス１に" ＡＢＣＤ" を出力
Ａ＝１２                            ： 変数Ａに１２を入力
ＰＲＩＮＴＥＲ Ａ                   ： 出力先をＧＰＩＢアドレス１２に設
ＧＰＲＩＮＴ " ｐｒｉｎｔｅｒ＝" , Ａ ： アドレス１２に文字列と変数内容を
                                     出力
```

【実行結果】

２台のＧＰＩＢプリンタのアドレスをそれぞれ１，１２に設定します。

アドレス１のプリンタには
ＡＢＣＤ

アドレス１２のプリンタには
ｐｒｉｎｔｅｒ＝　１２．０

と出力します。

（５２）ＲＥＡＤ　ＤＡＴＡ／ＲＥＳＴＯＲＥ

【概要】

ＤＡＴＡ文に書かれている定数を変数に入力します。

【書式】

ＲＥＡＤ　＜Ｘ＞ {, ＜Ｘ＞}
　　＜Ｘ＞：数値変数｜文字列変数

ＤＡＴＡ　＜Ｙ＞ {, ＜Ｙ＞}
　　＜Ｙ＞：数値定数｜文字列定数

ＲＥＳＴＯＲＥ　[＜ラベル＞]

【解説】

・　ＤＡＴＡ文で定義されている数値、文字列をＲＥＡＤ文で指定してある変数に入力します。

・　最初のＲＥＡＤでは、原則として（ＲＥＳＴＯＲＥ文で変更されていなければ）、ＤＡＴＡ文をプログラムの先頭から順に捜して、最初に発見したＤＡＴＡ文の値を変数に入力します。その後は、順に対応するＤＡＴＡ文の定数を捜して入力していきます。

・　ＤＡＴＡ文での文字列の定義は、ダブル・クォーテーション（”）で囲みます。

・　ＲＥＳＴＯＲＥ命令は、ＲＥＡＤ文で読み込むＤＡＴＡ文の先頭を指定します。（ラベルを指定すると、そのラベルの位置以降にあるＤＡＴＡ文を捜します。ラベルの指定がないとプログラムの先頭からＤＡＴＡ文を捜します。）

【例】

```
ＲＥＡＤ　Ａ
ＲＥＡＤ　Ａ，Ｂ＄

ＤＡＴＡ　１，２，３，４
ＤＡＴＡ　”ＡＢＣ”，５，”ＤＥＦ”，６

ＲＥＳＴＯＲＥ
ＲＥＳＴＯＲＥ　＊ＬＡ
```

【注意】

・ ＲＥＡＤ文の変数に対してＤＡＴＡ文の指定する定数の数が少ない場合、エラーになります。
(Unmatched DATA's values and READ variable)

・ 入力する変数の型とＤＡＴＡ文に指定してある定数の型が合わない場合、エラーになります。
(Invalid type in getdata)

【プログラム例】

```
ＲＥＳＴＯＲＥ　＊ＬＡ　　：ラベル　＊ＬＡ以降のＤＡＴＡ文から読み込む指定
ＲＥＡＤ　Ａ，Ｂ＄　　：数値変数Ａ、文字列変数Ｂ＄を読み込む
ＰＲＩＮＴ　Ａ，Ｂ＄　　：変数内容を画面へ出力
ＲＥＳＴＯＲＥ　＊ＬＢ　　：ラベル　＊ＬＢ以降のＤＡＴＡ文から読み込む指定
ＲＥＡＤ　Ａ，Ｂ＄　　：数値変数Ａ、文字列変数Ｂ＄を読み込む
ＰＲＩＮＴ　Ａ，Ｂ＄　　：変数内容を画面へ出力
ＳＴＯＰ　　：プログラム終了
！
＊ＬＢ　　：ラベル　＊ＬＡ
　ＤＡＴＡ　２，”Ｂ”　　：　データ
＊ＬＡ　　：ラベル　＊ＬＢ
　ＤＡＴＡ　１，”Ａ”　　：　データ
```

【実行結果】

```
１．０　　Ａ
２．０　　Ｂ
```

（５３）ＲＥＭ（！と省略可）

【概要】

プログラムに注釈文を置きます。

【書式】

ＲＥＭ　［＜文字列＞］

！　［＜文字列＞］

【解説】

・　プログラムに注釈文を付けたいときに使います。

・　ＲＥＭ文は非実行命令であり、ＲＥＭ文に続く文字はすべて無視します。（すべての文字、数字、記号が使用できます。）

【例】

```
ＲＥＭ　　＜＜＜ｒｅｍａｒｋ＞＞＞
！
！　　＊＊＊＊　ＡＤＶＡＮＴＥＳＴ　＊＊＊＊
```

【注意】

・　ＲＥＭ文の後ろにはコロン（：）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。

【プログラム例】

```
！　＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊　：注釈
！　＊＊　＜＜ＰＲＯＧＲＡＭ＞＞　＊＊　：注釈
！　＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊　：注釈
ＲＥＭ　　　　　　　　　　　　　　：注釈
ＲＥＭ　　　　ＡＤＶＡＮＴＥＳＴ　：注釈
！　　　　　　　　　　　　　　：注釈
ＰＲＩＮＴ　”ＴＥＳＴ”　　　　：文字列を出力
ＳＴＯＰ　　　　　　　　　　　　：プログラム終了
```

【実行結果】

```
ＴＥＳＴ
```

（５４）ＲＥＭＯＴＥ

【概要】

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

【書式】

ＲＥＭＯＴＥ　［装置アドレス　{，装置アドレス}］

装置アドレス：０～３０

【解説】

・　装置アドレスを指定せずにＲＥＭＯＴＥ命令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（ｔｕｒｅ）となりＧＰＩＢ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（ｆａｌｓｅ）にするためには、（３２）ＬＯＣＡＬ命令を実行して下さい。

・　ＲＥＭＯＴＥ命令に続いて、装置アドレスを指定した場合、指定した装置アドレスの装置のみ、リモート状態にします。装置アドレスはカンマ（，）で区切ると複数指定できます。

・　以下の命令を実行するとＲＥＭＯＴＥ命令を実行しなくてもリモート状態にします。

ＯＵＴＰＵＴ ……（４６）参照
ＳＥＮＤ　ＬＩＳＴＥＮ ……（６１）参照

【例】

```
ＲＥＭＯＴＥ
ＲＥＭＯＴＥ　６
ＲＥＭＯＴＥ　６，７，１０
```

【注意】

・　スレーブ・モードでは機能しません。

【プログラム例】

```
ＲＥＭＯＴＥ　１１
ＷＡＩＴ　２０００
ＬＯＣＡＬ　１１
```

【実行結果】

装置アドレス１１をリモート状態（リモート・ランプが点灯）にして、２秒後にＬＯＣＡＬ状態（リモート・ランプが消灯）になります。

（５５）ＲＥＮＡＭＥ

【概要】

メモリ・カード内のファイル名を変更します。

【書式】

ＲＥＮＡＭＥ　＜旧ファイル名＞，＜新ファイル名＞

【解説】

・　メモリ・カード内のプログラムファイル、データファイルのファイル名を変更します。

・　エディタのＢＡＳＩＣ　ｍｏｄｅでＲＥＮＡＭＥ命令を実行します。

・　変更指定されたファイル名が見つからないと、エラーになります。
（ＲＥＮＡＭＥ（旧ファイル名，新ファイル名）　ｅｒｒｏｒ）

【例】

ＲＥＮＡＭＥ　”ＢＡＳ１”，”ＢＡＳＩＣ１”

【注意】

・　変更する前には、ＣＡＴ命令、またはエディタのＬｏａｄを実行し、変更するファイルを確かめて下さい。

・　ファイル名は１０文字以内です。１１文字以上の場合は１１文字以降の文字は無視します。

・　メモリ・カードのライト・プロテクトがオンのときは実行できません。

(56) REQUEST

【注意】

・ スレーブ・モードで使用するときは［4. マスタ／スレーブ・モード］の章を参照して下さい。

・ マスタ・モードでは機能しません。

（５７）ＲＵＮ

【概要】

ＢＡＳＩＣプログラムの実行を開始させます。

【書式】

ＲＵＮ　［＜ラベル＞］

【解説】

・　ラベルを省略すると、プログラムの先頭から実行を開始します。

・　ラベルを指定すると、そのラベルの位置からの実行を開始します。

・　この命令は、ＭＯＶＥ ａｎｄ ＲＵＮによってＢＡＳＩＣバッファに転送済みのプログラムを再度実行（ＲＵＮのみ）させます。

【例】

ＲＵＮ

ＲＵＮ　＊ＡＢＣ

【注意】

・　ラベル指定のＲＵＮ命令の実行は、ＢＡＳＩＣ　ｍｏｄｅで行って下さい。（プログラムの実行の開始は、外部端末　Ｆ１９キー、または本体ＣＲＴのＲＵＮキーを押して下さい。プログラムはＢＡＳＩＣインタプリタに転送してあるものとします。）

・　ＣＯＮＴ命令での実行は、変数の初期化をしません。

（５８）ＰＵＲＧＥ

【概要】

メモリ・カード内のファイルを削除します。

【書式】

ＰＵＲＧＥ　＜ファイル名＞

【解説】

・　メモリ・カード内のプログラムファイル、データファイルを削除します。

・　エディタのＢＡＳＩＣ　ｍｏｄｅでＰＵＲＧＥ命令を実行します。

・　削除指定されたファイル名がないときはエラーになります。
（ＰＵＲＧＥ（ファイル名）　ｅｒｒｏｒ）

【例】

ＰＵＲＧＥ　”ＢＡＳ１”

【注意】

・　消去する前には、ＣＡＴ命令、またはエディタのＬｏａｄを実行し、消去するファイルを確かめて下さい。

・　メモリ・カードのライト・プロテクトがオンのときは実行できません。

（５９）ＳＣＲＡＴＣＨ

【概要】

ＢＡＳＩＣインタプリタ側の内部バッファを消去します。

【書式】

ＳＣＲＡＴＣＨ　［＜数値表現式＞］

＜数値表現式＞　：　１、２　のいずれか

【解説】

・　すでに転送されているＢＡＳＩＣプログラムが不要になったときに、この命令を実行します。

・　ＢＡＳＩＣインタプリタ側の内部バッファのプログラムのデータ（変数値など）のみを初期化する場合は、＜数値表現式＞に１を指定します。

・　ＢＡＳＩＣ内部バッファのプログラムのみを消去する（変数値は残す）場合は、＜数値表現式＞に２を指定します。

【例】

ＳＣＲＡＴＣＨ

ＳＣＲＡＴＣＨ　１

ＳＣＲＡＴＣＨ　２

【注意】

・　ＢＡＳＩＣプログラムがＢＡＳＩＣインタプリタ側の内部バッファに転送されていないとＳＣＲＡＴＣＨできません。

(60) ＳＥＬＥＣＴ　ＣＡＳＥ／
ＣＡＳＥ　ＥＬＳＥ／
ＥＮＤ　ＳＥＬＥＣＴ

【概要】

１つの式の値を条件として、複数の分岐をします。

【書式】

```
SELECT　＜数値表現式＞

CASE　＜数値表現式＞

END　SELECT
```

【解説】

・　ＣＡＳＥ文に指定された式の値とＳＥＬＥＣＴ文で指定した式の値が、一致するＣＡＳＥ文以下の文（複文）を実行します。

・　一致したならばＣＡＳＥ文以下を、一致しなければＣＡＳＥ　ＥＬＳＥ文以下を実行します。（ＣＡＳＥ　ＥＬＳＥは指定しなくてもよい。）

・　ＳＥＬＥＣＴ文は入れ子（ｎｅｓｔ）ができます。

【例】

```
INPUT "A= ", A
SELECT A
 CASE 1
  PRINT "1"
 CASE 3
  PRINT "3"
 CASE 5
  PRINT "5"
 CASE ELSE
  PRINT "ELSE"
END SELECT
```

【注意】

・　ＳＥＬＥＣＴ文を指定した場合、必ずＥＮＤ　ＳＥＬＥＣＴを置いて下さい。

・　ＳＥＬＥＣＴ文は、数値表現式のみ指定できます。

【プログラム例】

```
    INPUT "A= ",A              :変数Ａへ入力
    INPUT "B= ",B              :変数Ｂへ入力
    SELECT A                   :  ┐   ＡのＳＥＬＥＣＴ文
     CASE 1                    :  │   Ａ＝１ならば以下の処理
       SELECT B                :  │ ┐   ＢのＳＥＬＥＣＴ文
        CASE 10                :  │ │   Ｂ＝１０ならば以下の処理
          PRINT "A=1,B=10"     :  │ │     文字列出力
        CASE ELSE              :  │ │   Ｂ＜＞１０ならば以下の処理
          PRINT "A=1,B=ELSE"   :  │ │     文字列出力
       END SELECT              :  │ ┘   ＢのＳＥＬＥＣＴ文の終了
     CASE 2                    :  │   Ａ＝２ならば以下の処理
       PRINT "A=2"             :  │     文字列出力
     CASE 10                   :  │   Ａ＝１０ならば以下の処理
       PRINT "A=10"            :  │     文字列出力
   END SELECT                  :  ┘   ＡのＳＥＬＥＣＴ文の終了
```

【実行結果】

```
Ａ＝１
Ｂ＝１０
Ａ＝１，Ｂ＝１０

Ａ＝１
Ｂ＝２０
Ａ＝１，Ｂ＝ＥＬＳＥ

Ａ＝１０
Ｂ＝１０
Ａ＝１０
```

(61) ＳＥＮＤ

【概要】

ＧＰＩＢ上にコマンド、およびデータを出力します。

【書式】

ＳＥＮＤ　＜Ａ＞|＜Ｂ＞|＜Ｃ＞ {, ＜Ａ＞|＜Ｂ＞}

Ａ：ＣＭＤ|ＤＡＴＡ|ＬＩＳＴＥＮ [＜Ｄ＞ {, ＜Ｄ＞}]
Ｂ：ＵＮＴ|ＵＮＬ
Ｃ：ＴＡＬＫ　[＜Ｄ＞]
Ｄ：数値表現式

【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して送る命令です。

ＣＭＤ　：　アテンション(ＡＴＮ)ラインを真(ｔｕｒｅ)にして、与えられた数値を８ビットのバイナリ・データに変換してＧＰＩＢに送ります。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整数に変換されます。
　数値を指定しないで実行した場合は、アテンション（ＡＴＮ）ラインを真（ｔｕｒｅ）にします。

ＤＡＴＡ　：　アテンション(ＡＴＮ)ラインを偽(ｆａｌｓｅ)にして、与えられた数値を８ビットのバイナリ・データに変換してＧＰＩＢに送ります。ここで扱う数値は”ＣＭＤ”で扱われるものと同様です。
　数値を指定しないで実行した場合は、アテンション（ＡＴＮ）ラインを偽（ｆａｌｓｅ）にします。

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０～３０までの数値で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。数値は０～３０までの数値です。（複数の指定はできない。）

ＵＮＬ　：　アンリスン（ＵＮＬ）コマンドをＧＰＩＢ上に送ります。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、この命令によりリスナ状態が解除されます。

ＵＮＴ　：　アントーク（ＵＮＴ）コマンドをＧＰＩＢ上に送ります。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、この命令によりトーカ状態が解除されます。

【例】

```
SEND UNT UNL TALK 1 LISTEN 2,3 DATA 0x43 0x46 0x35 0x4d 0x5a 0xd 0xa
  SEND CMD
SEND DATA
```

【注意】

・ スレーブ・モードでは機能しません。

【プログラム例】

```
FOR I=1 TO 9
 A=I+0x30
 SEND UNT UNL TALK 30 LISTEN 11 DATA 0x43 0x46 A 0x4d 0x5a 0xd 0xa
 WAIT 1000
NEXT I
SEND UNL UNT
```

【実行結果】

装置アドレス１１の中心周波数を１ＭＨｚ～９ＭＨｚまで１秒ごとに設定を変えます。最後に装置アドレス１１のリスナ状態と装置アドレス３０のトーカ状態を解除します。

(62) ＳＰＯＬＬ

【概要】

ＧＰＩＢ上に接続されている指定した特定の装置にシリアル・ポールを行い、ステータス・バイトを読み込みます。

【書式】

ＳＰＯＬＬ（装置アドレス）

装置アドレス　０～３０　：　ＧＰＩＢ上の装置
　　３１　　：　本器計測部

【解説】

・　ＧＰＩＢ（ＣＯＮＴＲＯＬＬＥＲ側）に接続されている指定した装置へシリアル・ポールを行い、ステータス・バイトを読み込みます。

・　本器計測部のステータス・バイトは以下のようになります。

ビット	内　　容
0	ＵＮＣＡＬを発生したときにこのビットが立つ
1	キャリブレーションが終了したときにこのビットが立つ
2	掃引が終了したときにこのビットが立つ
3	アベレージが設定回数に達したときにこのビットが立つ
4	プロット出力が終了したときにこのビットが立つ
5	ＧＰＩＢコードに誤りがあったときにこのビットが立つ
6	ＳＲＱ割り込みが発生するモード”Ｓ０”に設定されているときビット０～５、７のいずれかのビットが立つとこのビットが立つ
7	ＲＥＱＵＥＳＴ命令を実行したときにこのビットが立つ

【例】

ＳＰＯＬＬ（１１）　：装置アドレス１１にシリアル・ポールを行う。
ＳＰＯＬＬ（３１）　：本器計測部へシリアル・ポールを行う。

【注意】

・　スレーブ・モードでは機能しません。（スレーブ・モードでは、０を返します。）

【プログラム例】

```
INTEGER S
OUTPUT 31;"S0"
ON ISRQ GOSUB *SS
ENABLE INTR
*L
 GOTO *L
*SS
 S=SPOLL(31)
 IF S BAND 4 THEN PRINT "SWEEP END"
 RETURN
```

【実行結果】

本器計測部の掃引終了毎に" ＳＷＥＥＰ　ＥＮＤ" をプリントします。

(63) ＳＰＲＩＮＴＦ

【概要】

数値、文字列を編集し文字列変数に入力します。

【書式】

ＳＰＲＩＮＴＦ　＜文字列変数＞，＜文字列表現式＞［，＜Ｘ＞］

Ｘ：数値表現式｜文字列表現式

【解説】

・　この命令はＰＲＩＮＴＦ命令とほぼ同じですが、この命令は編集した文字列を指定した文字列変数に入力できます。

・　文字列表現式に文字列表現式内に以下の書式を指定すると、引数のデータを編集し入力できます。（編集仕様は％で始まります。）

マイナス符号：編集されたパラメータをフィールド（出力領域）の左側にそろえることを指定する
ピリオド　：フィールドの幅を示す数字列と桁数（精度）を指定する数字列との区切り
０（ゼロ）　：余ったフィールドにゼロ・サプレスを行う

・　編集文字とその意味

ｄ　：パラメータを１０進数に変換する。
ｏ　：パラメータを符号なし８進数に変換する。（先頭に０を付けない）
ｘ　：パラメータを符号なし１６進数に変換する。（先頭に０ｘを付けない）
ｓ　：文字列
ｅ　：実数として受取り、［－］ｍ．ｎｎｎｎｎｎＥ［±］ｘｘの形の１０進数に変換する。
（ｎの文字列の長さは精度によって指定される。標準値は６）
ｆ　：実数として受取り、［－］ｍｍｍ．ｎｎｎｎの形の１０進数に変換する。
（ｎの文字列の長さは精度によって指定される。標準値は６）

・　％の後の文字は、編集文字でなければその文字が入力されます。したがって％は％％で入力します。

【例】

```
SPRINTF S$,"C=%d",C
```
変数Ｃを１０進数で入力する

```
SPRINTF S$,"C=%5d",C
```
変数Ｃを５桁の１０進数で入力する（右詰め）

```
SPRINTF S$,"C=%-5d",C
```
変数Ｃを５桁の１０進数で入力する（左詰め）

```
SPRINTF S$,"C=%05d",C
```
変数Ｃを５桁の１０進数で入力する
（右詰め：ゼロサプレスする）

```
SPRINTF S$,"H=%x",H
```
変数Ｈを１６進数で入力する

```
SPRINTF S$,"H=%4x",H
```
変数Ｈを４桁の１６進数で入力する（右詰め）

```
SPRINTF S$,"H=%-4x",H
```
変数Ｈを４桁の１６進数で入力する（左詰め）

```
SPRINTF S$,"H=%04x",H
```
変数Ｈを４桁の１６進数で入力する
（右詰め：ゼロサプレスする）

```
SPRINTF B$,"S$=%s",S$
```
文字列を入力する

```
SPRINTF B$,"S$=%8s",S$
```
文字列を８文字右詰めで入力する

```
SPRINTF B$,"S$=%-8s",S$
```
文字列を８文字左詰めで入力する

```
SPRINTF B$,"%20.10s",S$
```
２０文字のフィールド内で右詰めで１０文字を入力する

```
SPRINTF B$,"%-20.10s",S$
```
２０文字のフィールド内で左詰めで１０文字を入力する

```
SPRINTF B$,"F=%f",F
```
変数Ｆを１０進数の実数で入力する

ＳＰＲＩＮＴＦ　Ｂ＄，＂Ｆ＝％８．２ｆ＂，Ｆ

変数Ｆを小数点以下２桁の全８桁で入力する（小数点含む）

ＳＰＲＩＮＴＦ　Ｂ＄，＂Ｆ＝％０８．２ｆ＂，Ｆ

変数Ｆを小数点以下２桁の全８桁でゼロ・サプレスして入力する（小数点含む）

【注意】

・　ＳＰＲＩＮＴＦの編集文字とパラメータが不足していたり型指定が間違っていたりすると無意味な入力になることがあります。

【プログラム例】

```
DIM S$[80]
SPRINTF S$,"%d * %4d =%05d",12,34,12*34
PRINT S$
SPRINTF S$,":%-10d:%10d:",123,456
PRINT S$
A$="ABCD"
SPRINTF S$,":%8s:%-8s:%8.2s:%-8.2s:",A$,A$,A$,A$
PRINT S$
SPRINTF S$,"%f+%8.3f - %06.3f=%e",1.23,4.56,7.89,1.23+4.56-7.89
PRINT S$
```

【実行結果】

```
12 *   34 =00408
:123        :       456:
:    ABCD:ABCD    :      AB:AB       :
1.230000+   4.560 - 07.890=-2.100000e+00
```

（６４）ＳＴＥＰ

【概要】

ＢＡＳＩＣプログラムを１行のみ実行させます。

【書式】

ＳＴＥＰ　［＜ラベル＞］

【解説】

・　ラベルを指定した場合、指定されたラベル位置のプログラムを１行実行します。

・　ラベル指定がない場合は、直前に終了した次の行を実行します。
（連続して指定すると１行ずつ実行しながら進みます。）

・　ＰＡＵＳＥ命令で一時停止しているときにも、ＳＥＴＰ命令で１行ずつ進めることができます。

【例】

```
ＳＴＥＰ

ＳＴＥＰ　＊ＡＢＣ
```

【注意】

・　この命令は、ストップ・キー（コントロールＣ）またはＰＡＵＳＥ命令によってプログラムが停止した状態でのみ実行されます。
（ポップアップ・メニューのＢＡＳＩＣ　ｍｏｄｅ内でのみ使用できます。）

・　ＧＯＴＯ文、ＦＯＲ～ＮＥＸＴ文のループの中では、ＳＴＥＰ命令は実行しません。

(65) ＳＴＯＰ

【概要】

プログラムの実行を中止します。

【書式】

ＳＴＯＰ

【解説】

・ プログラムの実行を中止します。ＳＴＯＰ命令により、プログラムが停止すると以下のメッセージが出力されます。

(Program ended normally.)

【例】

ＳＴＯＰ

【注意】

・ ＳＴＯＰ命令でプログラムの実行を中止させたとき、ＣＯＮＴ命令では再実行しません。

【プログラム例】

```
FOR I=1 TO 10          ：カウンタＩに１を入力Ｉ＞１０になるまでループ
 IF I=5 THEN STOP      ：もし、Ｉが５ならばプログラム終了
 PRINT I               ：変数Ｉを出力
NEXT I                 ：ループ戻り
```

【実行結果】

```
１．０
２．０
３．０
４．０
Ｐｒｏｇｒａｍ　ｅｎｄｅｄ　ｎｏｒｍａｌｌｙ．
```

(66) ＴＲＩＧＧＥＲ

【概要】

ＧＰＩＢ上に接続されているすべての装置、または指定し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。

【書式】

ＴＲＩＧＧＥＲ　［装置アドレス　{，装置アドレス}］

装置アドレス：０～３０

【解説】

・　装置アドレスを指定せずにＴＲＩＧＧＥＲ命令だけを実行した場合、ＧＰＩＢにはグループ･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されてなければいけません。

・　ＴＲＩＧＧＥＲ命令に続いて装置アドレスを指定すると、指定した装置にのみＧＥＴコマンドを送ります。

【例】

```
TRIGGER

TRIGGER  2

TRIGGER  3，4，5
```

【注意】

・　スレーブ・モードでは機能しません。

【プログラム例】

```
TRIGGER  5
ENTER  5；A$
PRINT  A$
```

【実行結果】

装置アドレス５にトリガをかけ、装置アドレス５からデータを入力します。

(67) ＷＡＩＴ

【概要】

プログラムの実行を指定時間停止させます。

【書式】

ＷＡＩＴ　＜数値表現式＞

【解説】

・　数値表現式は、時間をミリ秒（ｍｓｅｃ）で指定します。

・　指定できる時間は、０～６３９９９ミリ秒（ｍｓｅｃ）です。

【例】

```
WAIT  1000
WAIT  60000
A=200
WAIT  A
```

【注意】

・　時間の指定は、ミリ秒（ｍｓｅｃ）単位です。

・　ＯＮ　（ＳＲＱ／ＩＳＲＱ／ＫＥＹ）で指定された割り込みが発生しても、この命令を実行中割り込みによる分岐は行いません。

【プログラム例】

```
FOR  I=1  TO  8
 READ  S
 BUZZER  S,1000
 WAIT  1000
NEXT  I
DATA  261,294,330,349,392,440
DATA  494,523
```

【実行結果】

”ドレミファソラシド”を１秒ごとに音階を変えて鳴します。

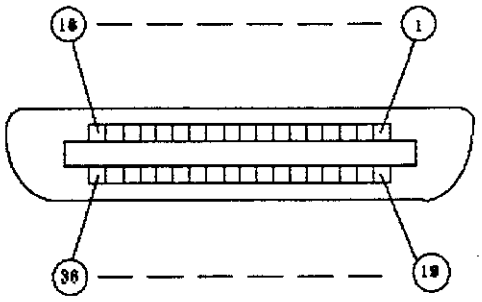
2.4 パラレルＩ／Ｏ

接続コネクタ：　３６ピン　アンフェノールコネクタ（プラグ）

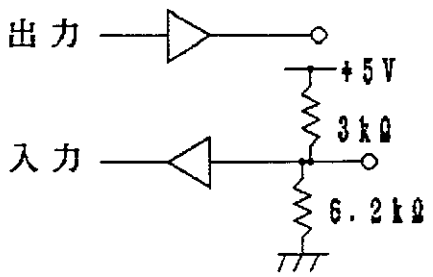
端子番号	信号名	端子番号	信号名	
1	ＧＮＤ	１９	＊ＯＥ	
2	ＩＮ０	２０	ＯＵＴ０	ＬＳＢ
3	ＩＮ１	２１	ＯＵＴ１	
4	ＩＮ２	２２	ＯＵＴ２	
5	ＩＮ３	２３	ＯＵＴ３	
6	ＩＮ４	２４	ＯＵＴ４	
7	ＩＮ５	２５	ＯＵＴ５	
8	ＩＮ６	２６	ＯＵＴ６	
9	ＩＮ７	２７	ＯＵＴ７	
１０	ＩＮ８	２８	ＯＵＴ８	
１１	ＩＮ９	２９	ＯＵＴ９	
１２	ＩＮ１０	３０	ＯＵＴ１０	
１３	ＩＮ１１	３１	ＯＵＴ１１	
１４	ＩＮ１２	３２	ＯＵＴ１２	
１５	ＩＮ１３	３３	ＯＵＴ１３	
１６	ＩＮ１４	３４	ＯＵＴ１４	
１７	ＩＮ１５	３５	ＯＵＴ１５	ＭＳＢ
１８		３６		

【３６ピン　コネクタ端子割当表】

ＩＮ０　～　ＩＮ１５　：　入力　（ＴＴＬ）
ＯＵＴ０～ＯＵＴ１５　：　出力　（ＴＴＬ）
＊ＯＥ　：　入力　出力イネーブル　負論理



回　路



・ＯＵＴＰＵＴ　３２　を使用したとき
（ＯＵＴＰＵＴ３２はパラレルＩ／Ｏへデータ出力）

ＯＵＴＰＵＴ　３２：１　　－－－＞　ＯＵＴ０が、Ｈレベルになります。

ＯＵＴＰＵＴ　３２：３　　－－－＞　ＯＵＴ０とＯＵＴ１が、Ｈレベルになります。

・ＥＮＴＥＲ　３２：を使用したとき　（ＥＮＴＥＲ３２はパラレルＩ／Ｏへデータ入力）

ＩＮ０が　Ｈレベルのとき　　ＥＮＴＥＲ　３２：Ａ　　Ａは１が代入されます。

ＩＮ１
ＩＮ２が　Ｈレベルのとき　　ＥＮＴＥＲ　３２：Ａ　　Ａは６が代入されます。

【注意】
- 電源投入時はすべてＨレベルとなります。
- データ入力／出力時は＊ＯＥをＬレベルにして下さい。

３．　ビルトイン関数

3.1 概要

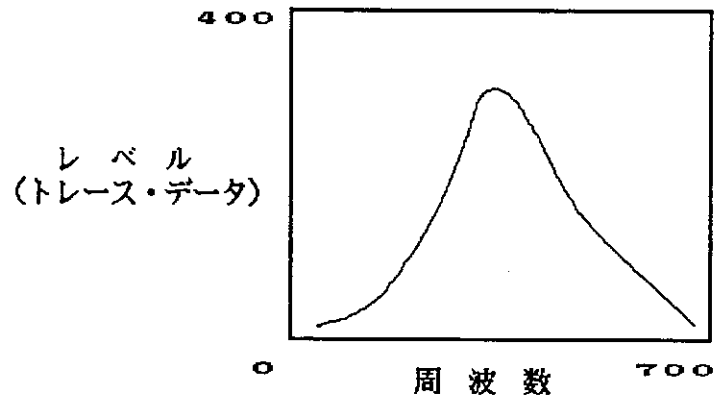
ビルトイン関数を使用することにより簡単な操作で解析処理ができ、プログラムの開発時間の短縮および高スループットを可能にします。

3.2 はじめに

3.2.1 使用の前に

（１）ポイント処理

ポイントには周波数（横軸）とレベル（縦軸）があり、下図に示すような精度を持ちます。波形の解析処理の基本となるデータです。



●周波数ポイント

横軸　７０１ポイント　　１ポイント当りの周波数は、周波数スパン／７００となります。

●レベルポイント（トレースデータ）

縦軸　４０１ポイント　　１ポイント当りのレベルは、ダイナミック・レンジ／４００となります。

（２）グラフィック機能

このＢＡＳＩＣにはグラフィック機能があります。
グラフィック機能を使用する前に画面の説明をします。

画面は以下の３つのスクリーンがあります。

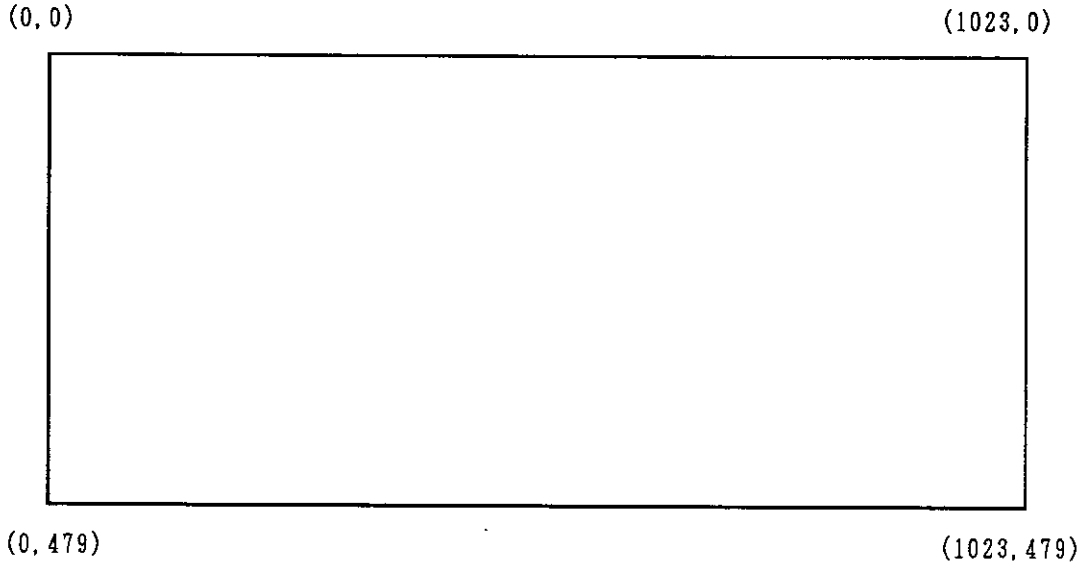
波形画面　　：プリセットを押したときに最初にでる画面
文字画面（実行結果）
　　：ＢＡＳＩＣのＰＲＩＮＴ命令等で文字が出力される画面
グラフィック画面：ＢＡＳＩＣのグラフィック関数で出力される画面

ＢＡＳＩＣプログラムで各画面へ出力する前に、ＯＵＴＰＵＴ　３１命令で下記のいずれかのＧＰＩＢコードを実行して下さい。

表示状態	ＧＰＩＢコマンド
波形画面のみ	ＯＵＴＰＵＴ　３１；”ＶＳ０”
波形画面＋文字画面	ＯＵＴＰＵＴ　３１；”ＶＳ１”
文字画面のみ	ＯＵＴＰＵＴ　３１；”ＶＳ２”
グラフィック画面	ＯＵＴＰＵＴ　３１；”ＶＳ３”

グラフィック関数を使用するときには、関数を実行する前にＯＵＴＰＵＴ　３１；”ＶＳ３”を実行して下さい。

グラフィック画面は以下の仕様になっています。

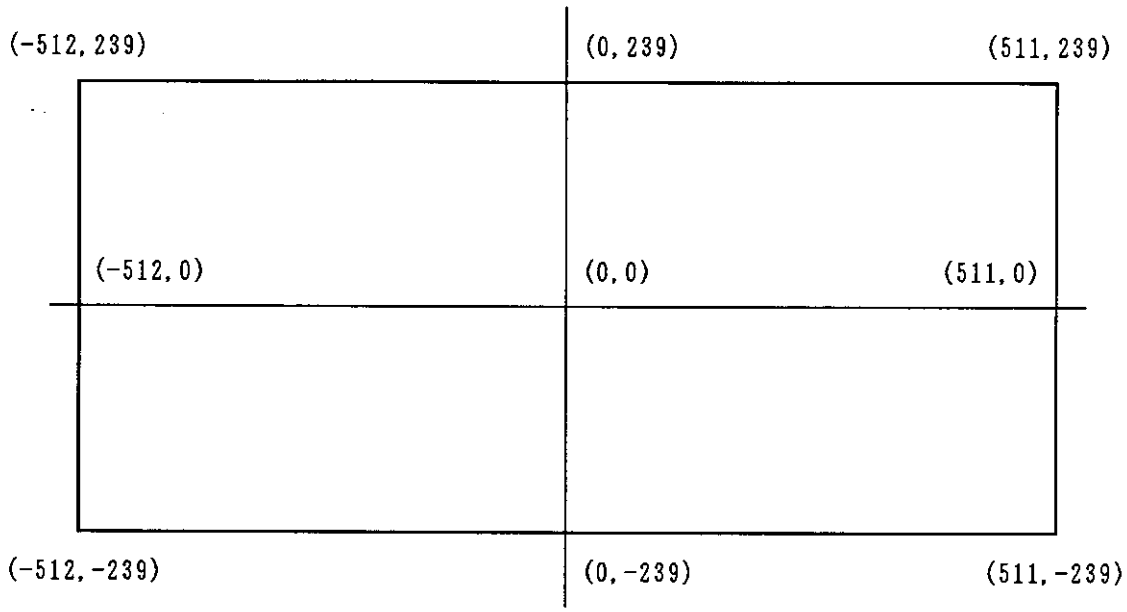


上図を絶対アドレスと言い、左上が原点（０，０）となります。

ＧＡＤＲＳ関数にてビューポート・アドレスを設定すると、前ページの図の範囲内なら原点（０，０）を任意に指定できます。

例）　ＧＡＤＲＳ（１，５１２，２４０）と設定すると、以下のようになります。

注：ビューポート・アドレス指定時は、設定時の画面範囲を超える指定をすると、エラーになります。



3.4節の（１）～（８）を参照して下さい。

3.2.2.使用上の注意

① リップル関係のデータを求めるときは、最初に極大、極小の数を求めて下さい。
以下のビルトイン関数を実行します。実行しないと、エラーになります。（ｆｕｎｃｔｉｏｎ ｅｒｒｏｒ）

ＮＲＰＬＨ －－ 極大のデータが必要なとき
ＮＲＰＬＬ －－ 極小のデータが必要なとき

注）極大、極小が必要なときは、両方行います。

上記のビルトイン関数を実行してから、以下のビルトイン関数を使って下さい。

ＰＲＰＬＨＮ
ＰＲＰＬＬＮ
ＦＲＰＬＨＮ
ＦＲＰＬＬＮ
ＶＲＰＬＨＮ
ＶＲＰＬＬＮ

② 関数のパラメータ等にエラーが生じた場合、エラー・メッセージを出力するだけで、プログラムの実行は停止しません。（ｆｕｎｃｔｉｏｎ ｅｒｒｏｒ）エラー時に求まる値は、不定値になります。

③ ＯＵＴＰＵＴ ３１，ＥＮＴＥＲ ３１を使って値を求めるよりもビルトイン関数を使用する方が高速に処理できます。（ＯＵＴＰＵＴ ３１，ＥＮＴＥＲ ３１で求めた値も、ビルトイン関数で求めた値も同じ値になります。）

④ ポイント（０～７００），トレース・データ（０～４００），トレースＡ／Ｂ（０／１）の値を、範囲外に指定すると、エラーになります。
（ｆｕｎｃｔｉｏｎ ｅｒｒｏｒ）

3.2.3 説明の構成

各関数は、以下の構成で説明します。

ＦＲＥＱ　（周波数）　－－関数名　（戻り値）
【機能】　　－－　関数の機能について
【書式】　　－－　ＢＡＳＩＣでの構文（書式）
【戻り値】　－－　関数を実行したときに、返る値
【エラー】　－－　設定パラメータの指定時、エラー等について
【注意】　　－－　設定パラメータの指定時の注意
【例】　　－－　ＢＡＳＩＣプログラム例

（１）エラー

・　ビルトイン関数にエラーが発生したときは、ＢＡＳＩＣプログラムの実行は停止しません。プログラムにおけるエラーの認識は、（４１）ＯＮ　ＥＲＲＯＲ文で認識できます。

例

```
ON ERROR GOTO *ERR
```

（２）機能別ビルトイン関数一覧

『周波数／ポイント（横軸）を求める』

No	関　数	内　容
1	ＦＲＥＱ	ポイントから周波数を求める
2	ＤＦＲＥＱ	ポイント間の周波数幅を求める
3	ＰＯＩＮＴ	周波数からポイントを求める
4	ＤＰＯＩＮＴ	周波数間のポイント幅を求める

『レベル／ポイント（縦軸）を求める』

No	関　数	内　容
5	ＬＥＶＥＬ	ポイントからレベルを求める
6	ＤＬＥＶＥＬ	ポイント間のレベル差を求める
7	ＬＶＰＯＩＮＴ	レベルからポイントを求める
8	ＬＶＤＰＯＩＮＴ	レベル間のポイント幅を求める
9	ＶＡＬＵＥ	横軸ポイント位置の波形のレベルを求める
10	ＤＶＡＬＵＥ	横軸ポイント間の波形のレベル差を求める
11	ＣＶＡＬＵＥ	周波数位置の波形のレベルを求める
12	ＤＣＶＡＬＵＥ	周波数間の波形のレベル差を求める

『最大／最小を求める』

No	関　数	内　容
13	ＦＭＡＸ	ポイント間の最大レベル位置の周波数を求める
14	ＦＭＩＮ	ポイント間の最小レベル位置の周波数を求める
15	ＰＭＡＸ	ポイント間の最大レベル位置の横軸ポイントを求める
16	ＰＭＩＮ	ポイント間の最小レベル位置の横軸ポイントを求める
17	ＭＡＸ	ポイント間の最大レベルを求める
18	ＭＩＮ	ポイント間の最小レベルを求める

『バンド幅を求める』

No	関　数	内　容
19	ＢＮＤ	横軸ポイント位置のロスレベルの帯域幅を求める
20	ＢＮＤＬ	横軸ポイント位置のロスレベルの低周波数側を求める
21	ＢＮＤＨ	横軸ポイント位置のロスレベルの高周波数側を求める
22	ＣＢＮＤ	周波数位置のロスレベルの帯域幅を求める
23	ＣＢＮＤＬ	周波数位置のロスレベルの低周波数側を求める
24	ＣＢＮＤＨ	周波数位置のロスレベルの高周波数側を求める

『極大／極小（リップル）を求める』

No	関　数	内　容
25	ＮＲＰＬＨ	すべての極大点の数を求める
26	ＮＲＰＬＬ	すべての極小点の数を求める
27	ＰＲＰＬＨＮ	左からＮ番目の極大点の横軸ポイントを求める
28	ＰＲＰＬＬＮ	左からＮ番目の極小点の横軸ポイントを求める
29	ＦＲＰＬＨＮ	左からＮ番目の極大点の周波数を求める
30	ＦＲＰＬＬＮ	左からＮ番目の極小点の周波数を求める
31	ＶＲＰＬＨＮ	左からＮ番目の極大点のレベルを求める
32	ＶＲＰＬＬＮ	左からＮ番目の極小点のレベルを求める
33	ＲＰＬ１	極大点の最大値と極小点の最小値のレベル差を求める

『上下限の判定を行う』

No	関　数	内　容
34	ＬＭＴＭＤ１	指定したデータについて基準値と上下幅により判定を行う
35	ＬＭＴＭＤ２	横軸ポイント位置の波形データについて基準値と上下幅により判定を行う
36	ＬＭＴＵＬ１	指定したデータについて上限値と下限値により判定を行う
37	ＬＭＴＵＬ２	横軸ポイント位置の波形データについて上限値と下限値により判定を行う

『電力を求める』

No	関　数	内　容
38	ＰＯＷＥＲ	横軸ポイント間の総電力を求める

『トレースデータ』

No	関　数	内　容
39	ＲＴＲＡＣＥ	指定ポイントのトレースデータを読み込む
40	ＷＴＲＡＣＥ	指定ポイントのトレースデータを書き込む

『グラフィック機能』

No	関　数	内　容
1	ＧＡＤＲＳ	グラフィック・ポイントの絶対アドレス/ビューポートアドレスを指定する
2	ＧＦＬＲＥＣＴ	指定２点間を対角線とする長方形を塗りつぶす
3	ＧＬＩＮＥ	指定２点間に直線を描く
4	ＧＭＫＲ	指定した位置にマーカ（ノーマル／デルタ）を描く
5	ＧＰＯＩＮＴ	指定した位置に点を描く
6	ＧＲＥＣＴ	指定２点間を対角線とする長方形を描く
7	ＧＳＴＲ	文字列を描く
8	ＧＳＴＹＬＥ	破線，点線，１点鎖線の要素の長さを指定する

（３）パラメータ指定一覧

＜ビルトイン関数＞

機　能	No	関　数
周波数／ポイントを求める	1	F＝FREQ（P）
	2	F＝DFREQ（P1，P2）
	3	P＝POINT（F）
	4	P＝DPOINT（F1，F2）
レベル／ポイントを求める	5	L＝LEVEL（T）
	6	L＝DLEVEL（T1，T2）
	7	T＝LVPOINT（L）
	8	T＝LVDPOINT（L1，L2）
	9	L＝VALUE（P，M）
	10	L＝DVALUE（P1，P2，M）
	11	L＝CVALUE（F，M）
	12	L＝DCVALUE（F1，F2，M）
最大／最小を求める	13	F＝FMAX（P1，P2，M）
	14	F＝FMIN（P1，P2，M）
	15	P＝PMAX（P1，P2，M）
	16	P＝PMIN（P1，P2，M）
	17	L＝MAX（P1，P2，M）
	18	L＝MIN（P1，P2，M）
バンド幅を求める	19	F＝BND（P，X，M）
	20	F＝BNDL（P，X，M）
	21	F＝BNDH（P，X，M）
	22	F＝CBND（F，X，M）
	23	F＝CBNDL（F，X，M）
	24	F＝CBNDH（F，X，M）
極大／極小を求める	25	N＝NRPLH（P1，P2，Dx，Dy，M）
	26	N＝NRPLL（P1，P2，Dx，Dy，M）
	27	P＝PRPLHN（N，M）
	28	P＝PRPLLN（N，M）
	29	F＝FRPLHN（N，M）
	30	F＝FRPLLN（N，M）
	31	L＝VRPLHN（N，M）
	32	L＝VRPLLN（N，M）
	33	L＝RPL1（P1，P2，Dx，Dy，M）
上下限の判定を行う	34	C＝LMTMD1（Dd，S，Ds）
	35	C＝LMTMD2（P，S，Ds，M）
	36	C＝LMTUL1（Dd，Up，Lo）
	37	C＝LMTUL2（P，Up，Lo，M）
電力を求める	38	W＝POWER（P1，P2，M）
トレースデータのread／write	39	T＝RTRACE（P，M）
	40	WTRACE（T，P，M）　　注）この関数は値を返しません。

Ｆ：周波数
Ｐ：ポイント（０～７００）
Ｌ：レベル
Ｔ：トレース・データ（０～４００）
Ｍ：トレースＡ／Ｂ（０／１）
Ｘ：１ｏｓｓレベル
Ｃ：チェック値（０，１，２，－１）
Ｎ：リップル数
Ｗ：ワット（電力）

Ｄｘ：横軸微分係数
Ｄｙ：縦軸微分係数
Ｓ　：基準値
Ｄｓ：基準値幅
Ｄｄ：被検データ
Ｌｏ：下限値
Ｕｐ：上限値

【パラメータ指定一覧】

機　能	No	関　数
グラフィック	1	GADRS (Mo, X, Y)
	2	GFLRECT (D, X1, Y1, X2, Y2)
	3	GLINE (S, D, X1, Y1, X2, Y2)
	4	GMKR (MK, D, X, Y)
	5	GPOINT (D, X, Y)
	6	GRECT (S, D, X1, Y1, X2, Y2)
	7	GSTR (C, X, Y, STR)
	8	GSTYLE (dash, space, dot)

C　：文字サイズ　0 ; 16×20ドット
　　1 ; 18×24ドット

D　：セット／消去　0 ; 消去
　　1 ; セット（描く）

MK　：マーカ　0 ; ノーマル・マーカ
　　1 ; △マーカ

Mo　：アドレス・モード　0 ; 絶対アドレス
　　1 ; ビューポート・アドレス

S　：ライン・スタイル　0 ; 実線
　　1 ; 破線
　　2 ; 点線
　　3 ; 一点鎖線

STR　：文字列表現式
X　：座標（横軸）
Y　：座標（縦軸）
dash　：破線部
dot　：点線部
space　：スペース部

3.3 各種ビルトイン関数の説明

以下の順に説明します。

（1）ＦＲＥＱ　（周波数）

【機能】

ポイント値を指定すると、そのポイントに相当する周波数を算出します。

【書式】

ＦＲＥＱ（ Ｐ ）

Ｐ：ポイント（０～７００）

【戻り値】

正常終了時：ポイント値を周波数に換算した値（Ｈｚ）。ゼロスパン時は、時間（Ｓｅｃ）
エラー時　：不定値

【エラー】

- 指定ポイントが０～７００の範囲外の場合、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

３５０ポイント目の周波数を求めます。

```
F＝ＦＲＥＱ（３５０）
```

４００ポイント目の周波数を求めます。

```
I＝２００
F＝ＦＲＥＱ（I＊２）
```

（２）ＤＦＲＥＱ　（周波数）

【機能】

ポイント幅（ポイント１，ポイント２）を指定すると、そのポイント幅に相当する周波数幅を算出します。

【書式】

ＤＦＲＥＱ（ Ｐ１，Ｐ２ ）

Ｐ１：指定ポイント１（０～７００）
Ｐ２：指定ポイント２（０～７００）

【戻り値】

正常終了時：指定ポイント間Ｐ１，Ｐ２の周波数幅（Ｈｚ）。ゼロスパン時は時間（Ｓｅｃ）
エラー時　：不定値

【エラー】

・　指定ポイントが０～７００の範囲外の場合、エラーになります。
・　エラーのときの戻り値は、不定値になります。
・　エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

Ｐ１＞Ｐ２の場合、Ｐ１とＰ２を入れ換えて処理します。

【例】

ポイント値　３００ と ４００ の間の周波数を求めます。

```
ＦＳ＝ＤＦＲＥＱ（３００，４００）

Ｉ＝４００
ＦＳ＝ＤＦＲＥＱ（Ｉ，３００）
```

（３）ＰＯＩＮＴ　（ポイント：横軸［０～７００］）

【機能】

周波数を指定すると、その周波数が測定器内部のポイント（０～７００）に対して、何ポイント目にあたるかを算出します。
各ビルトイン関数を高速に動作させるために重要な関数となります。

【書式】

ＰＯＩＮＴ（ Ｆ ）

Ｆ：指定周波数（Ｈｚ）。ゼロスパン時は時間（Ｓｅｃ）

【戻り値】

正常終了時：周波数を換算したポイント数（０～７００）
エラー時　：不定値

【エラー】

- 指定周波数がスタート周波数からストップ周波数の範囲外の場合、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

周波数３ＭＨｚの測定ポイントを求めます。

```
PO＝POINT（３E６）
```

周波数１０ＭＨｚの測定ポイントを求めます。

```
F＝１０E６
PO＝POINT（F）
```

（4）ＤＰＯＩＮＴ　（ポイント：横軸［０～７００］）

【機能】

周波数幅（周波数１、周波数２）を指定すると、その周波数幅が測定器内部のポイント（０～７００）に対して、周波数幅が何ポイントあるかを算出します。

【書式】

ＤＰＯＩＮＴ（ Ｆ１，Ｆ２ ）

Ｆ１：指定周波数１（Ｈｚ）。ゼロスパン時は時間１（Ｓｅｃ）
Ｆ２：指定周波数２（Ｈｚ）。ゼロスパン時は時間２（Ｓｅｃ）

【戻り値】

正常終了時：指定周波数間Ｆ１，Ｆ２のポイント数（０～７００）
エラー時　：不定値

【エラー】

- 指定周波数がスタート周波数からストップ周波数の範囲外の場合、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

Ｆ１＞Ｆ２の場合、Ｆ１とＦ２を入れ換えて処理します。

【例】

周波数３ＭＨｚと、４ＭＨｚの間のポイント数を求めます。

```
PSPAN=DPOINT（3E6，4E6）

PSPAN=DPOINT（4E6，3E6）
```

周波数３ＭＨｚと、３．５ＭＨｚの間のポイント数を求めます。

```
FA=3E6
PSPAN=DPOINT（FA，3．5E6）
```

（5）ＬＥＶＥＬ　（レベル）

【機能】

トレース・データ（縦軸［０～４００］）を指定すると、そのトレース・データに相当するレベルを算出します。

【書式】

ＬＥＶＥＬ（Ｔ）

Ｔ：トレース・データ

【戻り値】

正常終了時：トレース・データから換算したレベル（単位は、リファレンス・レベルの単位と同じ）

エラー時　：不定値

【エラー】

- 指定ポイントが０～４００の範囲外の場合、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレース・データ２００のレベル値を求めます。

```
L＝ＬＥＶＥＬ（２００）
```

トレースＡの全てトレース・データをレベル値に換えます。

```
ＤＩＭ　Ｌ［７０１］
ＯＵＴＰＵＴ　３１；”ＧＴＡ”
ＦＯＲ　Ｉ＝０　ＴＯ　７００
　Ｌ［Ｉ＋１］＝ＬＥＶＥＬ（ＲＴＲＡＣＥ（Ｉ，０））
ＮＥＸＴ　Ｉ
```

（6）ＤＬＥＶＥＬ　（レベル）

【機能】

トレース・データ［縦軸：０～４００］間（Ｔ１，Ｔ２）を指定すると、トレース・データ間に相当するレベルを算出します。

【書式】

ＤＬＥＶＥＬ（ Ｔ１，Ｔ２ ）

Ｔ１：指定トレース・データ１
Ｔ２：指定トレース・データ２

【戻り値】

正常終了時：トレース・データから換算したレベル（単位は、ｄＢまたは　Ｖ，Ｗ）
エラー時　：不定値

【エラー】

- 指定トレース・データが０～４００の範囲外の場合、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

Ｔ１＜Ｔ２の場合、Ｔ１とＴ２を入れ換えて処理します。

【例】

トレース・データ２００，３００間のレベルを求めます。

```
L＝ＬＥＶＥＬ（２００，３００）

L＝ＬＥＶＥＬ（３００，２００）
```

(7).ＬＶＰＯＩＮＴ　（トレース・データ：［０～４００］）

【機能】

レベルを指定すると、そのレベルに相当するトレース・データ値（縦軸）を算出します。

【書式】

ＬＶＰＯＩＮＴ（　Ｌ　）

Ｌ：レベル
（単位は、リファレンス・レベルの単位と同じ）

【戻り値】

正常終了時：レベルから換算したトレース・データ
エラー時　：不定値

【エラー】

- 指定レベルが最下位～ＲＥＦレベルの範囲外の場合、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

レベル値－１０ｄＢｍのトレース・データを求めます。

```
TD＝ＬＶＰＯＩＮＴ（－１０）
```

レベル値－２０ｄＢｍのトレース・データを求めます。

```
Ｌ＝－２０
ＴＤ＝ＬＶＰＯＩＮＴ（Ｌ）
```

（8）ＬＶＤＰＯＩＮＴ　（トレース・データ：［０～４００］）

【機能】

レベル間（Ｌ１，Ｌ２）を指定すると、そのレベル間のトレース・データ（縦軸）を算出します。

【書式】

ＬＶＤＰＯＩＮＴ（Ｌ１，Ｌ２）

Ｌ１：レベル１
（単位は、リファレンス・レベルの単位と同じ）
Ｌ２：レベル２
（単位は、リファレンス・レベルの単位と同じ）

【戻り値】

正常終了時：レベルから換算したトレース・データ
エラー時　：不定値

【エラー】

- 指定レベル１，２がそれぞれ最下位～ＲＥＦレベルの範囲外の場合、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

Ｌ１＜Ｌ２の場合、Ｌ１とＬ２を入れ換えて処理します。

【例】

レベル値 －５ｄＢｍ～ －１０ｄＢｍ間のトレース・データを求めます。

```
TD＝ＬＶＤＰＯＩＮＴ（－５，－１０）
```

レベル値 －５ｄＢｍ～ －２０ｄＢｍ間のトレース・データを求めます。

```
Ｌ＝－２０
TD＝ＬＶＤＰＯＩＮＴ（－５，Ｌ）
```

（9）ＶＡＬＵＥ　（レベル）

【機能】

ポイント値とトレース（Ａ／Ｂ）を指定すると、その指定トレース側での指定ポイントのレベルを求めます。

【書式】

ＶＡＬＵＥ（ Ｐ，Ｍ ）

Ｐ：指定ポイント値（０～７００）
Ｍ：トレース　０：トレースＡ
　　１：トレースＢ

【戻り値】

正常終了時：指定ポイント値のレベル（単位は、リファレンス・レベルの単位と同じ）
エラー時　：不定値

【エラー】

- トレース・メモリが、Ａ／Ｂトレース以外の値、または指定ポイントが（０～７００）以外の場合は、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレースＡの３００ポイント目のレベルを求めます。

```
L＝ＶＡＬＵＥ（３００，０）
```

トレースＢの周波数３ＭＨｚのレベルを求めます。

```
Ｆ＝３Ｅ６                    |   |
Ｐ＝ＰＯＩＮＴ（Ｆ）          | ＝ |   Ｌ＝ＣＶＡＬＵＥ（３Ｅ６，１）
Ｌ＝ＶＡＬＵＥ（Ｐ，１）      |   |
```

（左右のプログラムは、同じことを求めています。）

（１０）ＤＶＡＬＵＥ　（レベル）

【機能】

ポイント幅（ポイント１，ポイント２）の指定と、トレース（Ａ／Ｂ）を指定すると、トレース指定側の与えられた２点の周波数位置のレベル差を求めます。

【書式】

ＤＶＡＬＵＥ（Ｐ１，Ｐ２，Ｍ）

Ｐ１：指定ポイント１（０～７００）
Ｐ２：指定ポイント２（０～７００）
Ｍ　：トレース　０：トレースＡ
　　１：トレースＢ

【戻り値】

正常終了時：２点間のレベル差（単位は、ｄＢ　または　Ｖ，Ｗ）
エラー時　：不定値

【エラー】

- トレース・メモリが、Ａ／Ｂトレース以外の値、または指定ポイントが（０～７００）以外の場合は、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

Ｐ１＞Ｐ２の場合、Ｐ１とＰ２を入れ換えて処理します。

【例】

トレースＡの３０ポイント目と４０ポイント目のレベル差を求めます。

```
L=DVALUE(30,40,0)

L=DVALUE(40,30,0)
```

トレースＢの周波数２ＭＨｚと５ＭＨｚのレベル差を求めます。

```
F=2E6                          |   |
P1=POINT(F)                    |   |
P2=POINT(5E6)                  | = | L=DCVALUE(2E6,5E6,1)
L=DVALUE(P1,P2,1)              |   |
```

（左右のプログラムは、同じことを求めています。）

（１１）ＣＶＡＬＵＥ　（レベル）

【機能】

周波数とトレース（Ａ／Ｂ）を指定すると、その指定トレース側での指定周波数位置のレベルを求めます。

【書式】

ＣＶＡＬＵＥ（Ｆ，Ｍ）

Ｆ：指定周波数位置（Ｈｚ）。ゼロスパン時は時間（Ｓｅｃ）
Ｍ：トレース　０：トレースＡ
　　１：トレースＢ

【戻り値】

正常終了時：指定周波数位置のレベル（単位は、リファレンス・レベルの単位と同じ）
エラー時　：不定値

【エラー】

- トレース・メモリが、Ａ／Ｂトレース以外の値、または指定ポイントが（０～７００）以外の場合は、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレースＡの３．５ＭＨｚのレベルを求めます。

```
L＝VALUE（3．5E6，0）
```

トレースＢの周波数３ＭＨｚのレベルを求める。

```
F＝3E6                    |   |   L＝CVALUE（3E6，1）
P＝POINT（F）             | ＝ |
L＝VALUE（P，1）          |   |
```

（左右のプログラムは、同じことを求めています。）

（１２）ＤＣＶＡＬＵＥ　（レベル）

【機能】

周波数幅（周波数１、周波数２）の指定と、トレース（Ａ／Ｂ）を指定すると、トレース指定側の与えられた２点の周波数位置のレベル差を求めます。

【書式】

ＤＣＶＡＬＵＥ（Ｆ１，Ｆ２，Ｍ）

Ｆ１：指定周波数１（Ｈｚ）。ゼロスパン時は時間１（Ｓｅｃ）
Ｆ２：指定周波数２（Ｈｚ）。ゼロスパン時は時間２（Ｓｅｃ）
Ｍ　：トレース　０：トレースＡ
　　１：トレースＢ

【戻り値】

正常終了時：２点間のレベル差（単位は、ｄＢ　または　Ｖ，Ｗ）
エラー時　：不定値

【エラー】

- トレース・メモリが、Ａ／Ｂトレース以外の値、または指定ポイントが（０～７００）以外の場合は、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

Ｆ１＞Ｆ２の場合、Ｆ１とＦ２を入れ換えて処理します。

【例】

トレースＡの３ＭＨｚと４ＭＨｚのレベル差を求めます。

```
L＝ＤＣＶＡＬＵＥ（３Ｅ６，４Ｅ６，０）

L＝ＤＣＶＡＬＵＥ（４Ｅ６，３Ｅ６，０）
```

トレースＢの２ＭＨｚと５ＭＨｚのレベル差を求めます。

```
Ｆ＝２Ｅ６                          |   |
Ｐ１＝ＰＯＩＮＴ（Ｆ）              |   |
Ｐ２＝ＰＯＩＮＴ（５Ｅ６）          | ＝ | L＝ＤＣＶＡＬＵＥ（２Ｅ６，５Ｅ６，１）
Ｌ＝ＤＶＡＬＵＥ（Ｐ１，Ｐ２，１）  |   |
```

（左右のプログラムは、同じことを求めています。）

（１３）ＦＭＡＸ　（周波数）

【機能】

測定ポイント領域（ポイント１，ポイント２）と、トレースＡ／Ｂを指定すると、そのトレース側での領域内の縦軸の最大値点の周波数を算出します。

【書式】

ＦＭＡＸ（Ｐ１，Ｐ２，Ｍ）

Ｐ１：指定ポイント１（０～７００）
Ｐ２：指定ポイント２（０～７００）
Ｍ　：トレース　０：トレースＡ
　　　１：トレースＢ

【戻り値】

正常終了時：指定ポイント領域内での縦軸の最大値点の周波数（Ｈｚ）
エラー時　：不定値

【エラー】

- 指定ポイントが０～７００の範囲外の場合、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

Ｐ１＞Ｐ２の場合、Ｐ１とＰ２を入れ換えて処理します。

【例】

トレースＡの ０～７００ポイントの間のＭＡＸ値点の周波数を求めます。

```
MF＝FMAX（0，７００，0）
```

トレースＢの周波数１０～２０ＭＨｚ間のＭＡＸ値点の周波数を求めます。

```
F１＝１０Ｅ６
Ｐ１＝ＰＯＩＮＴ（Ｆ１）
ＭＦ１＝ＦＭＡＸ（Ｐ１，ＰＯＩＮＴ（２０Ｅ６），１）
ＭＦ２＝ＦＭＡＸ（ＰＯＩＮＴ（Ｆ１），ＰＯＩＮＴ（２０Ｅ６），１）
```

（ＭＦ１ と ＭＦ２は 同じ）

（１４）ＦＭＩＮ　（周波数）

【機能】

測定ポイント領域（ポイント１，ポイント２）と、トレースＡ／Ｂを指定すると、そのトレース側での領域内の縦軸の最小値点の周波数を算出します。

【書式】

ＦＭＩＮ（Ｐ１，Ｐ２，Ｍ）

Ｐ１：指定ポイント１（０～７００）
Ｐ２：指定ポイント２（０～７００）
Ｍ　：トレース　０：トレースＡ
　　　１：トレースＢ

【戻り値】

正常終了時：指定ポイント領域内での縦軸の最小値点の周波数（Ｈｚ）
エラー時　：不定値

【エラー】

- 指定ポイントが０～７００の範囲外の場合、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

Ｐ１＞Ｐ２の場合、Ｐ１とＰ２を入れ換えて処理します。

【例】

トレースＡの ０～７００ポイントの間のＭＩＮ値点の周波数を求めます。

```
MF＝FMIN（0，700，0）
```

トレースＢの周波数 １０～２０ＭＨｚ間のＭＩＮ値点の周波数を求めます。

```
F1＝10E6
P1＝POINT（F1）
MF1＝FMIN（POINT（20E6），P1，1）
MF2＝FMIN（POINT（20E6），POINT（P1），1）
```

（ＭＦ１ と ＭＦ２は 同じ）

（１５）ＰＭＡＸ　（ポイント：横軸［０～７００］）

【機能】

測定ポイント領域（ポイント１，ポイント２）と、トレースＡ／Ｂを指定すると、そのトレース側での領域内の縦軸の最大値点のポイント（横軸：０～７００）を算出します。

【書式】

ＰＭＡＸ（Ｐ１，Ｐ２，Ｍ）

Ｐ１：指定ポイント１（０～７００）
Ｐ２：指定ポイント２（０～７００）
Ｍ　：トレース　０：トレースＡ
　　　１：トレースＢ

【戻り値】

正常終了時：指定ポイント領域内での縦軸の最大値点のポイント（横軸：０～７００）
エラー時　：不定値

【エラー】

・　指定ポイントが０～７００の範囲外の場合、エラーになります。
・　エラーのときの戻り値は、不定値になります。
・　エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

Ｐ１＞Ｐ２の場合、Ｐ１とＰ２を入れ換えて処理します。

【例】

トレースＡの０～７００ポイントの間のＭＡＸ値点のポイント（横軸：０～７００）を求めます。

```
MP＝PMAX（0，７００，0）
```

トレースＢの周波数１０～２０ＭＨｚ間のＭＡＸ値点のポイント（横軸：０～７００）を求めます。

```
Ｆ１＝１０Ｅ６
Ｐ１＝ＰＯＩＮＴ（Ｆ１）
ＭＰ１＝ＰＭＡＸ（Ｐ１，ＰＯＩＮＴ（２０Ｅ６），１）
ＭＰ２＝ＰＭＡＸ（ＰＯＩＮＴ（Ｆ１），ＰＯＩＮＴ（２０Ｅ６），１）
```

（ＭＰ１　と　ＭＰ２は　同じ）

（１６）ＰＭＩＮ　（ポイント：横軸［０～７００］）

【機能】

測定ポイント領域（ポイント１，ポイント２）と、トレースＡ／Ｂを指定すると、そのトレース側での領域内の縦軸の最小値点のポイント（横軸：０～７００）を算出します。

【書式】

ＰＭＩＮ（Ｐ１，Ｐ２，Ｍ）

Ｐ１：指定ポイント１（０～７００）
Ｐ２：指定ポイント２（０～７００）
Ｍ　：トレース　０：トレースＡ
　　　１：トレースＢ

【戻り値】

正常終了時：指定ポイント領域内での縦軸の最小値点のポイント（横軸：０～７００）
エラー時　：不定値

【エラー】

・　指定ポイントが０～７００の範囲外の場合、エラーになります。
・　エラーのときの戻り値は、不定値になります。
・　エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

Ｐ１＞Ｐ２の場合、Ｐ１とＰ２を入れ換えて処理します。

【例】

トレースＡの０～７００ポイントの間のＭＩＮ値点のポイント（横軸：０～７００）を求めます。

```
MP＝PMIN（0，700，0）
```

トレースＢの周波数１０～２０ＭＨｚ間のＭＩＮ値点のポイント（横軸：０～７００）を求めます。

```
F1＝10E6
P1＝POINT（F1）
MP1＝PMIN（P1，POINT（20E6），1）
MP2＝PMIN（POINT（F1），POINT（20E6），1）
```

（ＭＰ１ と ＭＰ２は 同じ）

（１７）ＭＡＸ　（レベル）

【機能】

測定ポイント領域（ポイント１，ポイント２）と、トレースＡ／Ｂを指定すると、そのトレース側での領域内の縦軸の最大レベル値を算出します。

【書式】

ＭＡＸ（Ｐ１，Ｐ２，Ｍ）

Ｐ１：指定ポイント１（０～７００）
Ｐ２：指定ポイント２（０～７００）
Ｍ　：トレース　０：トレースＡ
　　　１：トレースＢ

【戻り値】

正常終了時：指定ポイント領域内での縦軸の最大レベル（単位は、リファレンス・レベルの単位と同じ）
エラー時　：不定値

【エラー】

- 指定ポイントが０～７００の範囲外の場合、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

Ｐ１＞Ｐ２の場合、Ｐ１とＰ２を入れ換えて処理します。

【例】

トレースＡの　０～７００ポイントの間のＭＡＸ値を求めます。

```
ML＝MAX（0，700，0）
```

トレースＢの周波数　１０～２０ＭＨｚ間のＭＡＸ値を求めます。

```
F1＝10E6
P1＝POINT（F1）
ML1＝MAX（P1，POINT（20E6），1）
ML2＝MAX（POINT（F1），POINT（20E6），1）
```

（ＭＬ１　と　ＭＬ２は　同じ）

（１８）ＭＩＮ　（レベル）

【機能】

測定ポイント領域（ポイント１，ポイント２）と、トレースＡ／Ｂを指定すると、そのトレース側での領域内の縦軸の最小レベル値を算出します。

【書式】

ＭＩＮ（Ｐ１，Ｐ２，Ｍ）

Ｐ１：指定ポイント１（０～７００）
Ｐ２：指定ポイント２（０～７００）
Ｍ　：トレース　０：トレースＡ
　　　１：トレースＢ

【戻り値】

正常終了時：指定ポイント領域内での縦軸の最小レベル（単位は、リファレンス・レベルの単位と同じ）
エラー時　：不定値

【エラー】

- 指定ポイントが０～７００の範囲外の場合、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

Ｐ１＞Ｐ２の場合、Ｐ１とＰ２を入れ換えて処理します。

【例】

トレースＡの　０～７００ポイントの間のＭＩＮ値を求めます。

```
ML=MIN(0, 700, 0)
```

トレースＢの周波数　１０～２０ＭＨｚ間のＭＩＮ値を求めます。

```
F1=10E6
P1=POINT(F1)
ML1=MIN(P1, POINT(20E6),1)
ML2=MIN(POINT(F1), POINT(20E6), 1)
```

（ＭＬ１　と　ＭＬ２は　同じ）

（１９）ＢＮＤ　（周波数）

【機能】

基準データの測定ポイントと、ＬＯＳＳレベル，トレースＡ／Ｂを指定すると、その指定トレース側での帯域幅を算出します。

【書式】

ＢＮＤ（Ｐ，Ｘ，Ｍ）

Ｐ：基準データのポイント（０～７００）
Ｘ：ＬＯＳＳレベル　（ｄＢ）
Ｍ：トレース　０：トレースＡ
　　１：トレースＢ

【戻り値】

正常終了時：指定トレース側の基準データからのＬＯＳＳレベルの帯域幅（Ｈｚ）
エラー時　：不定値

【エラー】

- 指定ポイントが０～７００の範囲外の場合、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレースＡ の ＭＡＸ値点からの３ｄＢダウンの帯域幅を求めます。

```
ＭＰ＝ＰＭＡＸ（０，７００，０）
ＢＷ１＝ＢＮＤ（ＭＰ，３，０）

ＢＷ２＝ＢＮＤ（ＰＭＡＸ（０，７００，０），３，０）

　（ＢＷ１とＢＷ２は同じ）
```

トレースＢの 中心ポイントから５ｄＢダウンの帯域幅を求めます。

```
ＢＷ＝ＢＮＤ（３５０，５，１）
```

（２０）ＢＮＤＬ　（周波数）

【機能】

基準データの測定ポイントと、ＬＯＳＳレベル，トレースＡ／Ｂを指定すると、その指定トレース側での帯域幅の低周波数側（左側）の周波数を算出します。

【書式】

ＢＮＤＬ（Ｐ，Ｘ，Ｍ）

Ｐ：基準データのポイント（０～７００）
Ｘ：ＬＯＳＳレベル（ｄＢ）
Ｍ：トレース　０：トレースＡ
　　１：トレースＢ

【戻り値】

正常終了時：指定トレース側の基準データからのＬＯＳＳレベルの帯域幅の低周波数側（左側）の周波数（Ｈｚ）
エラー時　：不定値

【エラー】

- 指定ポイントが０～７００の範囲外の場合、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレースＡのＭＡＸ値点からの５ｄＢダウンの帯域幅の低周波数側（左側）の周波数を求めます。

```
MP＝PMAX（0，７００，0）
ＢＬ１＝ＢＮＤＬ（ＭＰ，５，０）

ＢＬ２＝ＢＮＤＬ（ＰＭＡＸ（０，７００，０），５，０）

　（ＢＬ１とＢＬ２は同じ）
```

トレースＢの中心ポイントから５ｄＢダウンの帯域幅の低周波数側（左側）の周波数を求めます。

```
ＢＬ＝ＢＮＤＬ（３５０，５，１）
```

（２１）ＢＮＤＨ　（周波数）

【機能】

基準データの測定ポイントと、ＬＯＳＳレベル、トレースＡ／Ｂを指定すると、その指定トレース側での帯域幅の高周波数側（右側）の周波数を算出します。

【書式】

ＢＮＤＨ（Ｐ，Ｘ，Ｍ）

Ｐ：基準データのポイント（０～７００）
Ｘ：ＬＯＳＳレベル（ｄＢ）
Ｍ：トレース　０：トレースＡ
　　１：トレースＢ

【戻り値】

正常終了時：指定トレース側の基準データからのＬＯＳＳレベルの帯域幅の高周波数側（右側）の周波数（Ｈｚ）
エラー時　：不定値

【エラー】

・　指定ポイントが０～７００の範囲外の場合、エラーになります。
・　エラーのときの戻り値は、不定値になります。
・　エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレースＡのＭＡＸ値点からの５ｄＢダウンの帯域幅の高周波数側（右側）の周波数を求めます。

```
ＭＰ＝ＰＭＡＸ（０，７００，０）
ＢＬ１＝ＢＮＤＨ（ＭＰ，５，０）

ＢＬ２＝ＢＮＤＨ（ＰＭＡＸ（０，７００，０），５，０）

　（ＢＬ１とＢＬ２は同じ）
```

トレースＢの中心ポイントから５ｄＢダウンの帯域幅の高周波数側（右側）の周波数を求めます。

```
ＢＬ＝ＢＮＤＨ（３５０，５，１）
```

（２２）ＣＢＮＤ　（周波数）

【機能】

基準データの周波数位置と、ＬＯＳＳレベル、トレースＡ／Ｂを指定すると、その指定トレース側での帯域幅を算出します。

【書式】

ＣＢＮＤ（ Ｆ，Ｘ，Ｍ ）

Ｆ：基準データの周波数
Ｘ：ＬＯＳＳレベル（ｄＢ）
Ｍ：トレース　０：トレースＡ
　　１：トレースＢ

【戻り値】

正常終了時：指定トレース側の基準データからのＬＯＳＳレベルの帯域幅（Ｈｚ）
エラー時　：不定値

【エラー】

- 指定周波数がスタート周波数からストップ周波数の範囲外の場合、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレースＡの周波数３ＭＨｚからの３ｄＢダウンの帯域幅を求めます。

```
BW＝ＣＢＮＤ（３Ｅ６，３，０）
```

トレースＢの周波数１０ＭＨｚからの５ｄＢダウンの帯域幅を求めます。

```
Ｆ＝１０Ｅ６
Ｌ＝５
ＢＷ＝ＣＢＮＤ（Ｆ，Ｌ，１）
```

（２３）ＣＢＮＤＬ　（周波数）

【機能】

基準データの周波数位置と、ＬＯＳＳレベル、トレースＡ／Ｂを指定すると、その指定トレース側での帯域幅の低周波数側（左側）の周波数を算出します。

【書式】

ＣＢＮＤＬ（Ｆ，Ｘ，Ｍ）

Ｆ：基準データの周波数
Ｘ：ＬＯＳＳレベル（ｄＢ）
Ｍ：トレース　０：トレースＡ
　　１：トレースＢ

【戻り値】

正常終了時：指定トレース側の基準データからのＬＯＳＳレベルの帯域幅の低周波数側（左側）の周波数（Ｈｚ）
エラー時　：不定値

【エラー】

- 指定周波数がスタート周波数からストップ周波数の範囲外の場合、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレースＡの周波数３ＭＨｚからの３ｄＢダウンの帯域幅の左側を求めます。

```
BW＝ＣＢＮＤＬ（３Ｅ６，３，０）
```

トレースＢの周波数１０ＭＨｚからの５ｄＢダウンの帯域幅の左側を求めます。

```
Ｆ＝１０Ｅ６
Ｌ＝５
ＢＷ＝ＣＢＮＤＬ（Ｆ，Ｌ，１）
```

（２４）ＣＢＮＤＨ　（周波数）

【機能】

基準データの周波数位置と、ＬＯＳＳレベル、トレースＡ／Ｂを指定すると、その指定トレース側での帯域幅の高周波数側（右側）の周波数を算出します。

【書式】

ＣＢＮＤＨ（Ｆ，Ｘ，Ｍ）

Ｆ：基準データの周波数
Ｘ：ＬＯＳＳレベル（ｄＢ）
Ｍ：トレース　０：トレースＡ
　　１：トレースＢ

【戻り値】

正常終了時：指定トレース側の基準データからのＬＯＳＳレベルの帯域幅の高周波数側（右側）の周波数（Ｈｚ）
エラー時　：不定値

【エラー】

- 指定周波数がスタート周波数からストップ周波数の範囲外の場合、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレースＡの周波数３ＭＨｚからの３ｄＢダウンの帯域幅の右側を求めます。

```
BW＝ＣＢＮＤＨ（３Ｅ６，３，０）
```

トレースＢの周波数１０ＭＨｚからの５ｄＢダウンの帯域幅の右側を求めます。

```
Ｆ＝１０Ｅ６
Ｌ＝５
ＢＷ＝ＣＢＮＤＨ（Ｆ，Ｌ，１）
```

（２５）ＮＲＰＬＨ　（極大点の数）

【機能】

測定ポイント領域（ポイント１，ポイント２）と、トレースＡ／Ｂ、微分係数（Ｄｘ，Ｄｙ）を指定すれば、その指定トレース側のポイント領域内において極大を周波数軸（横軸：左側から）に対して求め、その極大点の数を算出します。

【書式】

ＮＲＰＬＨ（Ｐ１，Ｐ２，Ｄｘ，Ｄｙ，Ｍ）

Ｐ１：指定ポイント１（０～７００）
Ｐ２：指定ポイント２（０～７００）
Ｄｘ：微分係数ポイント（１～７００）
Ｄｙ：微分係数ポイント（１～４００）
Ｍ　：トレース　０：トレースＡ
　　１：トレースＢ

【戻り値】

正常終了時：極大を周波数軸（横軸：左側から）に対して求めた、その極大点の数（最大２５５）
エラー時　：不定値

【エラー】

- 指定ポイントが０～７００の範囲外の場合、エラーになります。
- 極大値がサーチできない場合は、エラーになります。
  （Ｄｘ，Ｄｙの数値を変更して下さい）
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

Ｐ１＞Ｐ２の場合、Ｐ１とＰ２を入れ換えて処理します。

ＰＲＰＬＨＮ，ＦＲＰＬＨＮ，ＶＲＰＬＨＮ　のビルトイン関数を使用する前に、必ずこの関数（ＮＰＲＬＨ）を実行して下さい。

【例】

トレースＡの０～７００ポイント内に対して、微分係数Ｄｘ＝５，Ｄｙ＝３の極大の数を求めます。

```
RH＝NRPLH（0，700，5，3，0）
```

トレースＢの１０～２０ＭＨｚ内に対して、微分係数Ｄｘ＝３，Ｄｙ＝５の極大の数を求めます。

```
STF＝10E6
SPF＝20E6
X＝3
Y＝5
STP＝POINT（STF）
SPP＝POINT（SPF）
TS＝1
RH＝NRPLH（STP，SPP，X，Y，TS）
```

（２６）ＮＲＰＬＬ　（極小点の数）

【機能】

測定ポイント領域（ポイント１，ポイント２）と、トレースＡ／Ｂ、微分係数（Ｄｘ，Ｄｙ）を指定すると、その指定トレース側のポイント領域内において、極小を周波数軸（横軸：左側から）に対して求め、その極小点の数を算出します。

【書式】

ＮＲＰＬＬ（Ｐ１，Ｐ２，Ｄｘ，Ｄｙ，Ｍ）

Ｐ１：指定ポイント１（０～７００）
Ｐ２：指定ポイント２（０～７００）
Ｄｘ：微分係数ポイント（１～７００）
Ｄｙ：微分係数ポイント（１～４００）
Ｍ　：トレース　０：トレースＡ
　　１：トレースＢ

【戻り値】

正常終了時：極小を周波数軸（横軸：左側から）に対して求めた、その極小点の数（最大２５５）
エラー時　：不定値

【エラー】

- 指定ポイントが０～７００の範囲外の場合、エラーになります。
- 極小値がサーチできない場合は、エラーになります。
  （Ｄｘ，Ｄｙの数値を変更して下さい）
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

Ｐ１＞Ｐ２の場合、Ｐ１とＰ２を入れ換えて処理します。

ＰＲＰＬＬＮ，ＦＲＰＬＬＮ，ＶＲＰＬＬＮ　のビルトイン関数を使用する前には、必ずこの関数（ＮＰＲＬＬ）を実行して下さい。

【例】

トレースＡの０～７００ポイント内に対して、微分係数Ｄｘ＝５，Ｄｙ＝３の極小の数を求めます。

```
RL＝NRPLL（0，７００，５，３，０）
```

トレースＢの１０～２０ＭＨｚ内に対して、微分係数Ｄｘ＝３，Ｄｙ＝５の極小の数を求めます。

```
ＳＴＦ＝１０Ｅ６
ＳＰＦ＝２０Ｅ６
Ｘ＝３
Ｙ＝５
ＳＴＰ＝ＰＯＩＮＴ（ＳＴＦ）
ＳＰＰ＝ＰＯＩＮＴ（ＳＰＦ）
ＴＳ＝１
ＲＬ＝ＮＲＰＬＬ（ＳＴＰ，ＳＰＰ，Ｘ，Ｙ，ＴＳ）
```

（２７）ＰＲＰＬＨＮ　（ポイント：横軸［０～７００］）

【機能】

トレースＡ／Ｂ および 周波数軸に対して、左からＮ番目の極大点のＮｏ．を指定すると、その極大点のポイント（０～７００）を算出します。

【書式】

ＰＲＰＬＨＮ（Ｎ，Ｍ）

Ｎ：周波数軸に対して左からＮ番目の極大点のＮｏ．
Ｍ：レース　０：トレースＡ
　　１：トレースＢ

【戻り値】

正常終了時：極大点を周波数軸（横軸：左側から）に対して、Ｎ番目の極大点のポイント位置（０～７００）
エラー時　：不定値

【エラー】

- Ｎ番目の極大点がない場合は、エラーになります。
  この関数を使う前に、ＮＲＰＬＨ　を実行して下さい。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレースＡの０～７００ポイント内に対して、微分係数Ｄｘ＝５，Ｄｙ＝３の全極大を求め、左から３番目の極大点のポイント位置を求めます。

```
RH＝NRPLH（0，７００，５，３，０）
P＝PRPLHN（３，０）
```

トレースＢの周波数１０～２０ＭＨｚ内に対して、微分係数Ｄｘ＝３，Ｄｙ＝５の全極大を求め、左から２番目の極大点のポイント位置を求めます。

```
ＳＴＦ＝１０Ｅ６
ＳＰＦ＝２０Ｅ６
Ｘ＝３
Ｙ＝５
ＳＴＰ＝ＰＯＩＮＴ（ＳＴＦ）
ＳＰＰ＝ＰＯＩＮＴ（ＳＰＦ）
ＴＳ＝１
RH＝NRPLH（ＳＴＰ，ＳＰＰ，Ｘ，Ｙ，ＴＳ）
Ｐ＝ＰＲＰＬＨＮ（２，ＴＳ）
```

（２８）ＰＲＰＬＬＮ　（ポイント：横軸［０～７００］）

【機能】

トレースＡ／Ｂ および 周波数軸に対して、左からＮ番目の極小点のＮｏ．を指定すると、その極小点のポイント（０～７００）を算出します。

【書式】

ＰＲＰＬＬＮ（Ｎ，Ｍ）

Ｎ：周波数軸に対して左からＮ番目の極小点のＮｏ．
Ｍ：トレース　０：トレースＡ
　　１：トレースＢ

【戻り値】

正常終了時：極小点を周波数軸（横軸：左側から）に対して、Ｎ番目の極小点のポイント位置（０～７００）
エラー時　：不定値

【エラー】

・　Ｎ番目の極小点がない場合は、エラーになります。
　この関数を使う前に、ＮＲＰＬＬ　を実行して下さい
・　エラーのときの戻り値は、不定値になります。
・　エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレースＡの０～７００ポイント内に対して、微分係数Ｄｘ＝５，Ｄｙ＝３の全極小を求め，左から３番目の極小点のポイント位置を求めます。

```
ＲＨ＝ＮＲＰＬＬ（０，７００，５，３，０）
Ｐ＝ＰＲＰＬＬＮ（３，０）
```

トレースＢの周波数１０～２０ＭＨｚ内に対して、微分係数Ｄｘ＝３，Ｄｙ＝５の全極小を求め，左から２番目の極小点のポイント位置を求めます。

```
ＳＴＦ＝１０Ｅ６
ＳＰＦ＝２０Ｅ６
Ｘ＝３
Ｙ＝５
ＳＴＰ＝ＰＯＩＮＴ（ＳＴＦ）
ＳＰＰ＝ＰＯＩＮＴ（ＳＰＦ）
ＴＳ＝１
ＲＨ＝ＮＲＰＬＬ（ＳＴＰ，ＳＰＰ，Ｘ，Ｙ，ＴＳ）
Ｐ＝ＰＲＰＬＬＮ（２，ＴＳ）
```

（２９）ＦＲＰＬＨＮ　（周波数）

【機能】

トレースＡ／Ｂおよび周波数軸に対して、左からＮ番目の極大点のＮｏ．を指定すると、その極大点の周波数を算出します。

【書式】

ＦＲＰＬＨＮ（Ｎ，Ｍ）

Ｎ：周波数軸に対して左からＮ番目の極大点のＮｏ．
Ｍ：トレース　０：トレースＡ
　　１：トレースＢ

【戻り値】

正常終了時：極大点を周波数軸(横軸：左側から)に対して、Ｎ番目の極大点の周波数（Ｈｚ）
エラー時　：不定値

【エラー】

- Ｎ番目の極大点がない場合は、エラーになります。
  （この関数を使う前に、ＮＲＰＬＨ　を実行して下さい）
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレースＡの０～７００ポイント内に対して、微分係数Ｄｘ＝５，Ｄｙ＝３の全極大を求め、左から３番目の極大点の周波数を求めます。

```
RH＝NRPLH（0，700，5，3，0）
F＝FRPLHN（3，0）
```

トレースＢの周波数１０～２０ＭＨｚ内に対して、微分係数Ｄｘ＝３，Ｄｙ＝５の全極大を求め、左から２番目の極大点の周波数を求めます。

```
STF＝10E6
SPF＝20E6
X＝3
Y＝5
STP＝POINT（STF）
SPP＝POINT（SPF）
TS＝1
RH＝NRPLH（STP，SPP，X，Y，TS）
F＝FRPLHN（2，TS）
```

（３０）ＦＲＰＬＬＮ　（周波数）

【機能】

トレースＡ／Ｂおよび周波数軸に対して、左からＮ番目の極小点のＮｏ．を指定すると、その極小点の周波数を算出します。

【書式】

ＦＲＰＬＬＮ（Ｎ，Ｍ）

Ｎ：周波数軸に対して左からＮ番目の極小点のＮｏ．
Ｍ：トレース　０：トレースＡ
　　１：トレースＢ

【戻り値】

正常終了時：極小点を周波数軸(横軸：左側から)に対して、Ｎ番目の極小点の周波数（Ｈｚ）
エラー時　：不定値

【エラー】

- Ｎ番目の極小点がない場合は、エラーになります。
  この関数を使う前に、ＮＲＰＬＬ　を実行して下さい
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレースＡの０～７００ポイント内に対して、微分係数　Ｄｘ＝５，Ｄｙ＝３の全極小を求め、左から３番目の極小点の周波数を求めます。

```
RH＝NRPLL（0，700，5，3，0）
F＝FRPLLN（3，0）
```

トレースＢの周波数１０ＭＨｚ～２０ＭＨｚ内に対して、微分係数　Ｄｘ＝３，Ｄｙ＝５の全極小を求め、左から２番目の極小点の周波数を求めます。

```
STF＝10E6
SPF＝20E6
X＝3
Y＝5
STP＝POINT（STF）
SPP＝POINT（SPF）
TS＝1
RH＝NRPLL（STP，SPP，X，Y，TS）
F＝FRPLLN（2，TS）
```

（３１）ＶＲＰＬＨＮ　（レベル）

【機能】

トレースＡ／Ｂおよび周波数軸に対して左からＮ番目の極大点のＮｏ．を指定すると、その極大点のレベルを算出します。

【書式】

ＶＲＰＬＨＮ（Ｎ，Ｍ）

Ｎ：周波数軸に対して左からＮ番目の極大点のＮｏ．
Ｍ：トレース　０：トレースＡ
　　１：トレースＢ

【戻り値】

正常終了時：極大点を周波数軸（横軸：左側から）に対して、Ｎ番目の極大点のレベル（単位は、リファレンス・レベルの単位と同じ）
エラー時　：不定値

【エラー】

・　Ｎ番目の極大点がない場合は、エラーになります。
　この関数を使う前に、ＮＲＰＬＨ　を実行して下さい
・　エラーのときの戻り値は、不定値になります。
・　エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレースＡの０～７００ポイント内に対して、微分係数Ｄｘ＝５，Ｄｙ＝３の全極大を求め、左から３番目の極大点のレベルを求めます。

```
RH＝NRPLH（0，７００，５，３，０）
L＝VRPLHN（３，０）
```

トレースＢの周波数１０～２０ＭＨｚ内に対して、微分係数Ｄｘ＝３，Ｄｙ＝５の全極大を求め、左から２番目の極大点のレベルを求めます。

```
ＳＴＦ＝１０Ｅ６
ＳＰＦ＝２０Ｅ６
Ｘ＝３
Ｙ＝５
ＳＴＰ＝ＰＯＩＮＴ（ＳＴＦ）
ＳＰＰ＝ＰＯＩＮＴ（ＳＰＦ）
ＴＳ＝１
ＲＨ＝ＮＲＰＬＨ（ＳＴＰ，ＳＰＰ，Ｘ，Ｙ，ＴＳ）
Ｌ＝ＶＲＰＬＨＮ（２，ＴＳ）
```

（３２）ＶＲＰＬＬＮ　（レベル）

【機能】

トレースＡ／Ｂおよび周波数軸に対して、左からＮ番目の極小点のＮｏ．を指定すると、その極小点のレベルを算出します。

【書式】

ＶＲＰＬＬＮ（Ｎ，Ｍ）

Ｎ：周波数軸に対して左からＮ番目の極小点のＮｏ．
Ｍ：トレース　０：トレースＡ
　　１：トレースＢ

【戻り値】

正常終了時：極小点を周波数軸（横軸：左側から）に対して、Ｎ番目の極小点のレベル（単位は、リファレンス・レベルの単位と同じ）
エラー時　：不定値

【エラー】

・　Ｎ番目の極小点がない場合は、エラーになります。
　この関数を使う前に、ＮＲＰＬＬ　を実行して下さい
・　エラーのときの戻り値は、不定値になります。
・　エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレースＡの０～７００ポイント内に対して、微分係数Ｄｘ＝５，Ｄｙ＝３の全極小を求め、左から３番目の極小点のレベルを求めます。

```
RH＝NRPLL（0，700，5，3，0）
L＝VRPLLN（3，0）
```

トレースＢの１０～２０ＭＨｚ内に対して、微分係数Ｄｘ＝３，Ｄｙ＝５の全極小を求め、左から２番目の極小点のレベルを求めます。

```
STF＝10E6
SPF＝20E6
X＝3
Y＝5
STP＝POINT（STF）
SPP＝POINT（SPF）
TS＝1
RH＝NRPLL（STP，SPP，X，Y，TS）
L＝VRPLLN（2，TS）
```

（３３.）ＲＰＬ１　（レベル）

【機能】

測定ポイント領域（ポイント１，ポイント２）と、トレースＡ／Ｂ、微分係数を（Ｄｘ，Ｄｙ）を指定すると、その指定トレース側のポイント領域内において、極大値、極小値をサーチし、極大値のうちの最大値と、極小値のうちの最小値のレベル差を算出します。

【書式】

ＲＰＬ１（Ｐ１，Ｐ２，Ｄｘ，Ｄｙ，Ｍ）

Ｐ１：指定ポイント１（０～７００）
Ｐ２：指定ポイント２（０～７００）
Ｄｘ：微分係数ポイント（１～７００）
Ｄｙ：微分係数ポイント（１～４００）
Ｍ　：トレース　０：トレースＡ
　　１：トレースＢ

【戻り値】

正常終了時：極大値のうちの最大値と、極小値のうちの最小値のレベル差（単位は、ｄＢまたはＶ，Ｗ）
エラー時　：不定値

【エラー】

- 指定ポイントＰ１，Ｐ２が０～７００の範囲外の場合、エラーになります。
- 極大値および極小値がサーチできない場合は、エラーになります。（Ｄｘ，Ｄｙの数値を変更して下さい。）
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

Ｐ１＞Ｐ２の場合、Ｐ１とＰ２を入れ換えて処理します。

【例】

トレースＡの０～７００ポイント内に対して、微分係数Ｄｘ＝５，Ｄｙ＝３の極大値のうちの最大値と、極小値のうちの最小値のレベル差を求めます。

```
ＲＰ＝ＲＰＬ１（０，７００，５，３，０）
```

トレースＢの周波数１０～２０ＭＨｚ内に対して、微分係数　Ｄｘ＝３，Ｄｙ＝５の極大値のうちの最大値と、極小値のうちの最小値のレベル差を求めます。

```
ＳＴＦ＝１０Ｅ６
ＳＰＦ＝２０Ｅ６
Ｘ＝３
Ｙ＝５
ＳＴＰ＝ＰＯＩＮＴ（ＳＴＦ）
ＳＰＰ＝ＰＯＩＮＴ（ＳＰＦ）
ＴＳ＝１
ＲＰ＝ＲＰＬ１（ＳＴＰ，ＳＰＰ，Ｘ，Ｙ，ＴＳ）
```

（３４）ＬＭＴＭＤ１　（チェック値［０，１，２］）

【機能】

基準値とその基準値からの上下限幅、および被検データを与えると、その上下限幅の領域内にあるか否かをチェックします。

【書式】

ＬＭＴＭＤ１（Ｄｄ，Ｓ，Ｄｓ）

Ｄｄ：被検データ
Ｓ　：基準値
Ｄｓ：上下限幅

（（Ｓ－Ｄｓ）≦　Ｄｄ　≦（Ｓ＋Ｄｓ））であるかを調べる。

【戻り値】

正常終了時：領域内のとき　　－－　０
　　上限値より上のとき　－－　１
　　下限値より下のとき　－－　２

【例】

入力した値が、４０～６０の間にあるかをチェックします。

```
ＩＮＰＵＴ　Ｄ
Ｔ＝ＬＭＴＭＤ１（Ｄ，５０，１０）
ＩＦ　Ｔ＝０　ＴＨＥＮ　ＰＲＩＮＴ　”ＯＫ”
ＩＦ　Ｔ＝１　ＴＨＥＮ　ＰＲＩＮＴ　”ＵＰＰＥＲ”
ＩＦ　Ｔ＝２　ＴＨＥＮ　ＰＲＩＮＴ　”ＬＯＷＥＲ”
```

基準値を入力し、トレースＡの３０ＭＨｚのレベルが、その基準値に対し±１０の範囲内であるか否かを判定します。

```
ＩＮＰＵＴ　Ｓ
ＬＶ＝ＣＶＡＬＵＥ（３０Ｅ６，０）
Ｔ＝ＬＭＴＭＤ１（ＬＶ，Ｓ，１０）
ＩＦ　Ｔ＝０　ＴＨＥＮ　ＰＲＩＮＴ　”ＯＫ”
ＩＦ　Ｔ＝１　ＴＨＥＮ　ＰＲＩＮＴ　”ＵＰＰＥＲ”
ＩＦ　Ｔ＝２　ＴＨＥＮ　ＰＲＩＮＴ　”ＬＯＷＥＲ”
```

（３５）ＬＭＴＭＤ２　（チェック値［０，１，２］）

【機能】

基準値と、その基準値からの上下限幅、およびトレースＡ／Ｂ、ポイントを与えると、そのトレース側のポイントのレベルが上下限幅の領域内にあるか否かをチェックします。

【書式】

ＬＭＴＭＤ２（Ｐ，Ｓ，Ｄｓ，Ｍ）

Ｐ　：ポイント（０～７００）
Ｓ　：基準値（単位はＲＥＦレベルと同じ）
Ｄｓ：上下限幅（ｄＢまたはＶ，Ｗ）
Ｍ　：トレース　０：トレースＡ
　　１：トレースＢ

（（Ｓ－Ｄｓ）≦　レベル　≦（Ｓ＋Ｄｓ））であるかを調べる。

【戻り値】

正常終了時：領域内のとき　　－－　０
　　上限値より上のとき　－－　１
　　下限値より下のとき　－－　２

【例】

基準値を入力し、トレースＡの３０ＭＨｚのレベルが、その基準値に対し±１０ｄＢの範囲内であるか否かを判定します。

```
ＩＮＰＵＴ　Ｓ
Ｐ＝ＰＯＩＮＴ（３０Ｅ６）
Ｔ＝ＬＭＴＭＤ２（Ｐ，Ｓ，１０，０）
ＩＦ　Ｔ＝０　ＴＨＥＮ　ＰＲＩＮＴ　”ＯＫ”
ＩＦ　Ｔ＝１　ＴＨＥＮ　ＰＲＩＮＴ　”ＵＰＰＥＲ”
ＩＦ　Ｔ＝２　ＴＨＥＮ　ＰＲＩＮＴ　”ＬＯＷＥＲ”
```

（３６）ＬＭＴＵＬ１　（チェック値［０，１，２］）

【機能】

上下限値と被検データを与えると、その上下限値の領域内にあるか否かをチェックします。

【書式】

ＬＭＴＵＬ１（Ｄｄ，Ｕｐ，Ｌｏ）

Ｄｄ：被検データ
Ｕｐ：上限値
Ｌｏ：下限値

（ Ｌｏ ≦ Ｄｄ ≦ Ｕｐ ）であるかを調べる。

【戻り値】

正常終了時：領域内のとき　　－－　０
　　上限値より上のとき　－－　１
　　下限値より下のとき　－－　２

【例】

入力した値が、３０～４０の間かをチェックします。

```
ＩＮＰＵＴ　Ｄ
Ｔ＝ＬＭＴＵＬ１（Ｄ，４０，３０）
ＩＦ　Ｔ＝０　ＴＨＥＮ　ＰＲＩＮＴ　”ＯＫ”
ＩＦ　Ｔ＝１　ＴＨＥＮ　ＰＲＩＮＴ　”ＵＰＰＥＲ”
ＩＦ　Ｔ＝２　ＴＨＥＮ　ＰＲＩＮＴ　”ＬＯＷＥＲ”
```

上下限値を入力し、トレースＡの３０ＭＨｚのレベルをチェックします。

```
ＩＮＰＵＴ　”ＵＰＰＥＲ”，Ｕ
ＩＮＰＵＴ　”ＬＯＷＥＲ”，Ｌ
ＬＶ＝ＣＶＡＬＵＥ（３０Ｅ６，０）
Ｔ＝ＬＭＴＵＬ１（ＬＶ，Ｕ，Ｌ）
ＩＦ　Ｔ＝０　ＴＨＥＮ　ＰＲＩＮＴ　”ＯＫ”
ＩＦ　Ｔ＝１　ＴＨＥＮ　ＰＲＩＮＴ　”ＵＰＰＥＲ”
ＩＦ　Ｔ＝２　ＴＨＥＮ　ＰＲＩＮＴ　”ＬＯＷＥＲ”
```

（３７）ＬＭＴＵＬ２　（チェック値［０，１，２］）

【機能】

上下限値とトレースＡ／Ｂ、ポイントを与えると、そのトレース側のポイントのレベルが上下限値の領域内にあるか否かをチェックします。

【書式】

ＬＭＴＵＬ２（Ｐ，Ｕｐ，Ｌｏ，Ｍ）

Ｐ　：ポイント（０～７００）
Ｕｐ：上限値（単位はＲＥＦレベルと同じ）
Ｌｏ：下限値（単位はＲＥＦレベルと同じ）
Ｍ　：トレース　０：トレースＡ
　　１：トレースＢ

（　Ｌｏ　≦　レベル　≦　Ｕｐ　）であるかを調べる。

【戻り値】

正常終了時：領域内のとき　　－－　０
　　上限値より上のとき　－－　１
　　下限値より下のとき　－－　２

【例】

上下限値を入力し，トレースＡの３０ＭＨｚのレベルをチェックします。

```
ＩＮＰＵＴ　”ＵＰＰＥＲ”，Ｕ
ＩＮＰＵＴ　”ＬＯＷＥＲ”，Ｌ
Ｐ＝ＰＯＩＮＴ（３０Ｅ６）
Ｔ＝ＬＭＴＵＬ２（Ｐ，Ｕ，Ｌ，０）
ＩＦ　Ｔ＝０　ＴＨＥＮ　ＰＲＩＮＴ　”ＯＫ”
ＩＦ　Ｔ＝１　ＴＨＥＮ　ＰＲＩＮＴ　”ＵＰＰＥＲ”
ＩＦ　Ｔ＝２　ＴＨＥＮ　ＰＲＩＮＴ　”ＬＯＷＥＲ”
```

（３８）ＰＯＷＥＲ　（総電力）

【機能】

ＲＥＦレベルが０ｄＢｍ時の総電力を求めます。周波数範囲は、ポイント幅（ポイント１，ポイント２）で決まります。なお縦軸の設定は１０ｄＢ／ｄｉｖに限ります。

【書式】

ＰＯＷＥＲ（Ｐ１，Ｐ２，Ｍ）

Ｐ１：指定ポイント１（０～７００）
Ｐ２：指定ポイント２（０～７００）
Ｍ　：トレース　０：トレースＡ
　　１：トレースＢ

【戻り値】

正常終了時：指定ポイント間Ｐ１，Ｐ２の総電力（ｍＷ）
エラー時　：不定値

【エラー】

- 指定ポイント値が０～７００の範囲外の場合、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

Ｐ１＞Ｐ２の場合、Ｐ１とＰ２を入れ換えて処理します。

ＲＥＦレベルを０ｄＢｍ以外に設定し、総電力を求める場合は、ＲＥＦレベルが１０ｄＢステップごとに設定されたときのみ算出可能となります。
（・・・－２０，－１０，０，１０，２０，・・・ｄＢｍのみ可能）

算出式を以下に示します。

$$\text{総電力} = \text{POWER}(P1, P2, M) * 10^{\frac{X}{10}} \quad (X\text{dBm})$$

【例】

トレースＡのポイント値　３００～４００の間の総電力を求めます。（ＲＥＦレベル＝０ｄＢｍ時）

```
AW＝POWER（３００，４００，０）
```

トレースＡのポイント値０～７００の間の総電力を求めます。（ＲＥＦレベル＝－２０ｄＢｍ時）

```
AW＝POWER（０，７００，０）／１００
```

（３９）ＲＴＲＡＣＥ　（トレース・データ：［０～４００］）

【機能】

指定したポイントのトレース・データを返します。

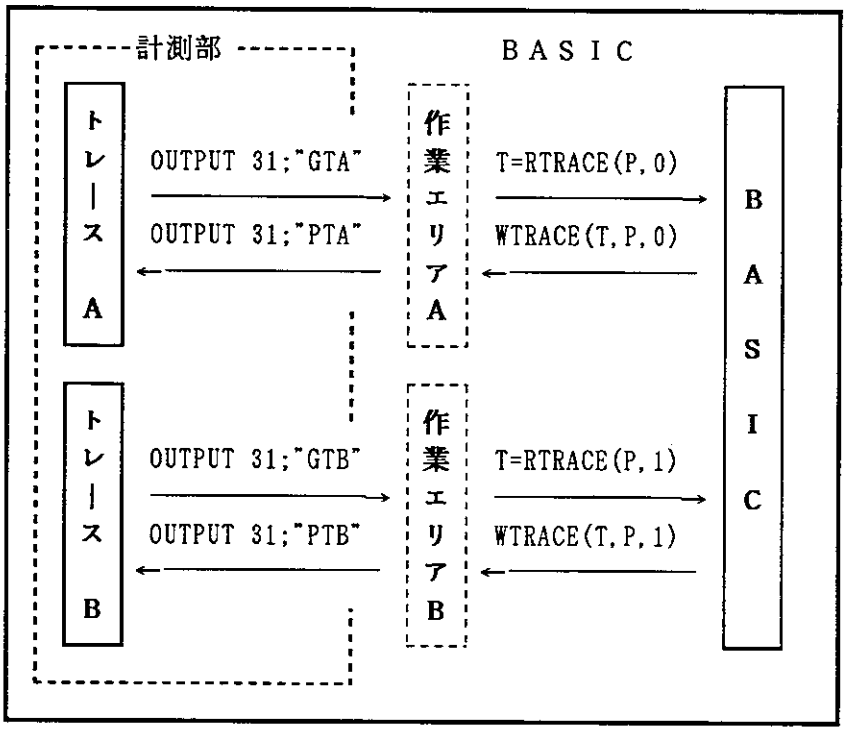
【書式】

ＲＴＲＡＣＥ（Ｐ，　Ｍ）

Ｐ：ポイント（０～７００）
Ｍ：トレース　０：トレースＡ
　　１：トレースＢ

この命令を行う前に、必ずＯＵＴＰＵＴ　３１命令で”ＧＴＡ”または”ＧＴＢ”を行います。これは計測部にあるトレース・データを作業エリアに転送するために行います。
トレースＡデータを作業エリアＡに転送するときは、ＯＵＴＰＵＴ　３１；”ＧＴＡ”
トレースＢデータを作業エリアＢに転送するときは、ＯＵＴＰＵＴ　３１；”ＧＴＢ”
を実行すると、７０１ポイントすべてを転送します。

このＲＴＲＡＣＥ関数は、作業エリアからのデータを１ポイントずつ読み込むための関数です。



トレース・データとＢＡＳＩＣのデータ転送の関係

【戻り値】

正常終了時：トレース・データ（０～４００）
エラー時　：不定値

【エラー】

・　ポイント指定で０～７００以外の数値は、エラーにならずに不定値が返ります。

【例】

トレースＡデータ（０～７００ポイント）を配列変数Ａに格納します。

```
INTEGER I,A(701)
OUTPUT 31;"GTA"
FOR I=0 TO 700
  A(I+1)=RTRACE(I,0)
NEXT I
```

【参考】

上記のプログラム例の実行速度は、約２．５秒です。
データ１ポイントあたり約３．５ミリ秒です。

（４０）ＷＴＲＡＣＥ

【機能】

指定したポイントへトレース・データを書き込みます。

【書式】

ＷＴＲＡＣＥ（Ｔ，　Ｐ，　Ｍ）

Ｍ：トレース　０：トレースＡ
　　１：トレースＢ
Ｐ：ポイント（０～７００）
Ｔ：トレース・データ（０～４００）

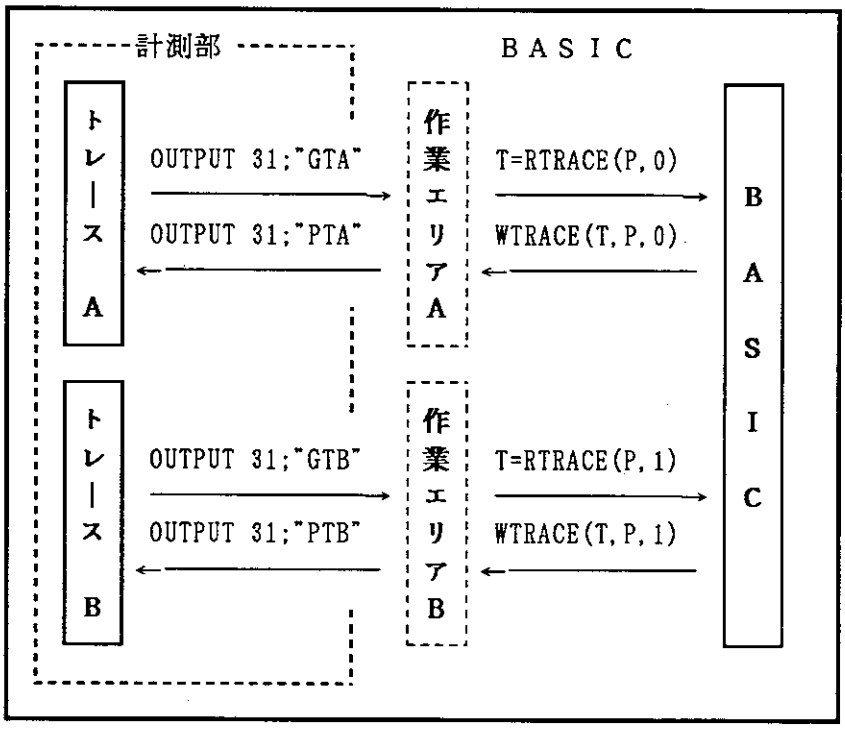
この命令を実行した後に、必ずＯＵＴＰＵＴ　３１命令で”ＰＴＡ”または”ＰＴＢ”を行います。これは作業エリアにあるトレース・データを計測部に転送するために行います。

作業エリアＡのデータをトレースＡへ書き込むときは、ＯＵＴＰＵＴ　３１；”ＰＴＡ”

作業エリアＢのデータをトレースＢへ書き込むときは、ＯＵＴＰＵＴ　３１；”ＰＴＢ”

を実行すると、７０１ポイントすべてを転送します。

このＷＴＲＡＣＥ関数は、ＢＡＳＩＣのデータを作業エリアに１ポイントずつ書き込むための関数です。



トレース・データとＢＡＳＩＣのデータ転送の関係

【エラー】

・　ポイント指定で０～７００以外の数値は、エラーにはならずに、そしてデータは書き込まれません

【例】

配列変数Ｂのデータをトレースデータ（計測部）へ転送します。
（このプログラムでは、配列変数Ｂは０です。）

```
INTEGER I,B(701)
FOR I=0 TO 700
  WTRACE(B(I+1),I,1)
NEXT I
OUTPUT 31;"PTB"
```

【参考】

上記のプログラム例の実行速度は、約２．２秒です。
データ１ポイントあたり約３．１ミリ秒です。

3.4　各種グラフィックの説明

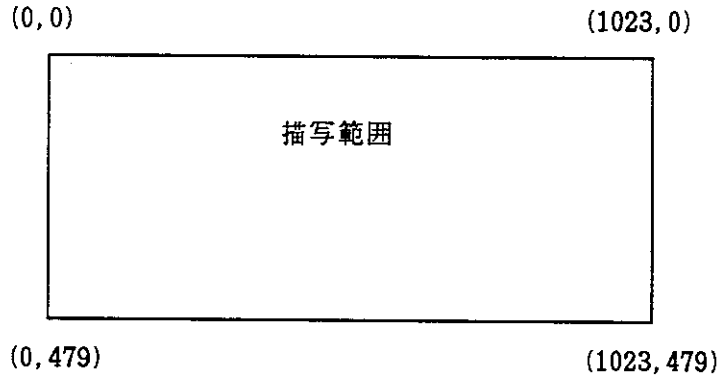
以下の順（アルファベット順）に説明します。

(1) GADRS

【機能】

描写画面のアドレス・モードを指定します。
アドレス・モードには以下の2つがあります。

0：絶対アドレス
1：ビューポート・アドレス



絶対アドレス・モードを指定すると原点は左上となり、このときのX，Y指定は無視されます。また、ビューポート・アドレスを指定したときはX，Yが原点となります。

注）X，Yは絶対アドレスで指定し、上図の範囲を超えないようにして下さい。

ビューポート・アドレスを指定したときは、原点を基準として右上が第1象限になります。

【書式】

GADRS（Mo，　X，　Y）

Mo：モード　　0：絶対アドレス
　　1：ビューポート・アドレス
X　：横軸（0～1023）
Y　：縦軸（0～479）

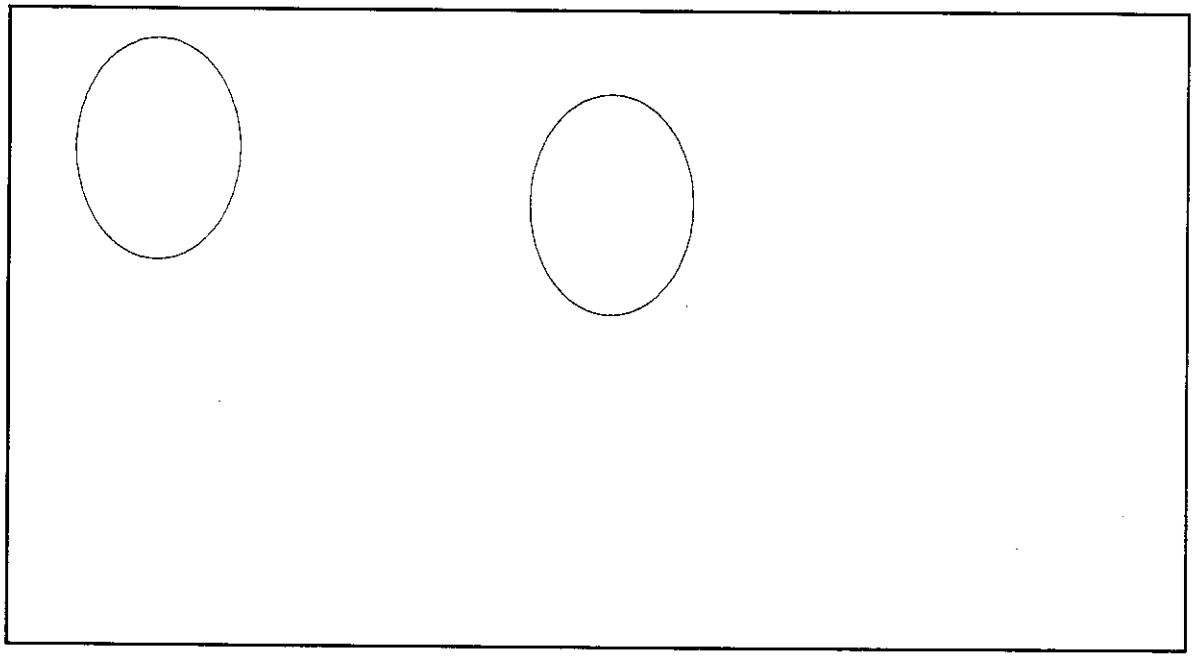
【エラー】

・　ビューポート・アドレス指定のときにX，Yが、描写画面の範囲外に設定されると、エラーになります。
・　エラー時は、メッセージを出力するだけで、実行は停止しません。

【例】

```
OUTPUT 31;"VS3"
CLS 1
R=100
OFFSET=100
FOR M=0 TO 1
  GADRS(M,512,240)
  FOR I=0 TO PI*2 STEP PI/90
    X=SIN(I)*R+OFFSET
    Y=COS(I)*R+OFFSET
    GPOINT(1,X,Y)
  NEXT I
NEXT M
```

【実行結果】



（２）ＧＦＬＲＥＣＴ

【機能】

指定された２点間（座標１、座標２）を対角線とする長方形を塗りつぶして描きます。

【書式】

ＧＦＬＲＥＣＴ（Ｄ，　Ｘ１，　Ｙ１，　Ｘ２，　Ｙ２）

Ｄ　　：０：消去
　　１：描く
Ｘ１，Ｙ１：座標１
Ｘ２，Ｙ２：座標２

【エラー】

・　Ｘ，Ｙが、描写画面の範囲外に設定されると、エラーになります。（ビューポート・アドレス時は注意して下さい。）
・　エラー時は、メッセージを出力するだけで実行は停止しません。

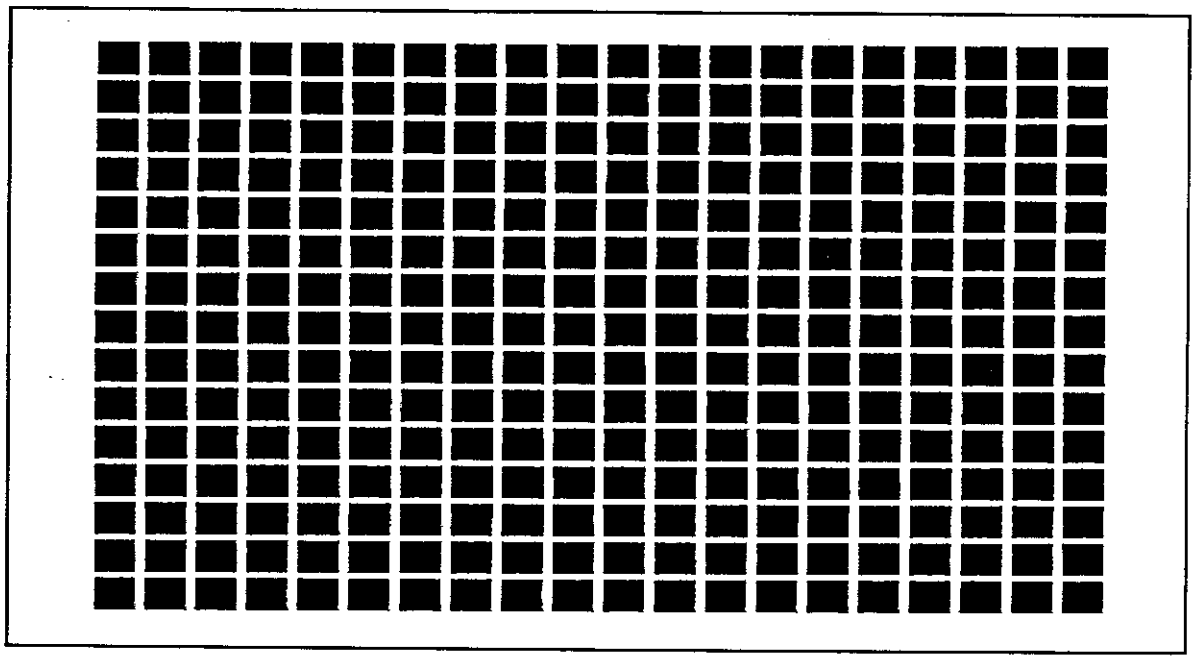
【注意】

・　座標１は必ず左上、座標２は右下をなるように指定して下さい。

【例】

```
OUTPUT 31;"VS3"
CLS 1
INTEGER X,Y
FOR X=10 TO 970 STEP 50
  FOR Y=10 TO 450 STEP 30
    GFLRECT(1,X,Y,X+40,Y+25)
  NEXT Y
NEXT X
STOP
```

【実行結果】



（3）ＧＬＩＮＥ

【機能】

指定された２点間（座標１、座標２）に線を描きます。

【書式】

ＧＬＩＮＥ（Ｓ，　Ｄ，　Ｘ１，　Ｙ１，　Ｘ２，　Ｙ２）

Ｓ　　：スタイル　０：実線
　　１：破線
　　２：点線
　　３：一点鎖線
Ｄ　　：０：消去
　　１：描く
Ｘ１，Ｙ１：座標１
Ｘ２，Ｙ２：座標２

【エラー】

・　Ｘ，Ｙが、描写画面の範囲外に設定されると、エラーになります。（ビューポート・アドレス時は注意して下さい。）
・　エラー時は、メッセージを出力するだけで、実行は停止しません。

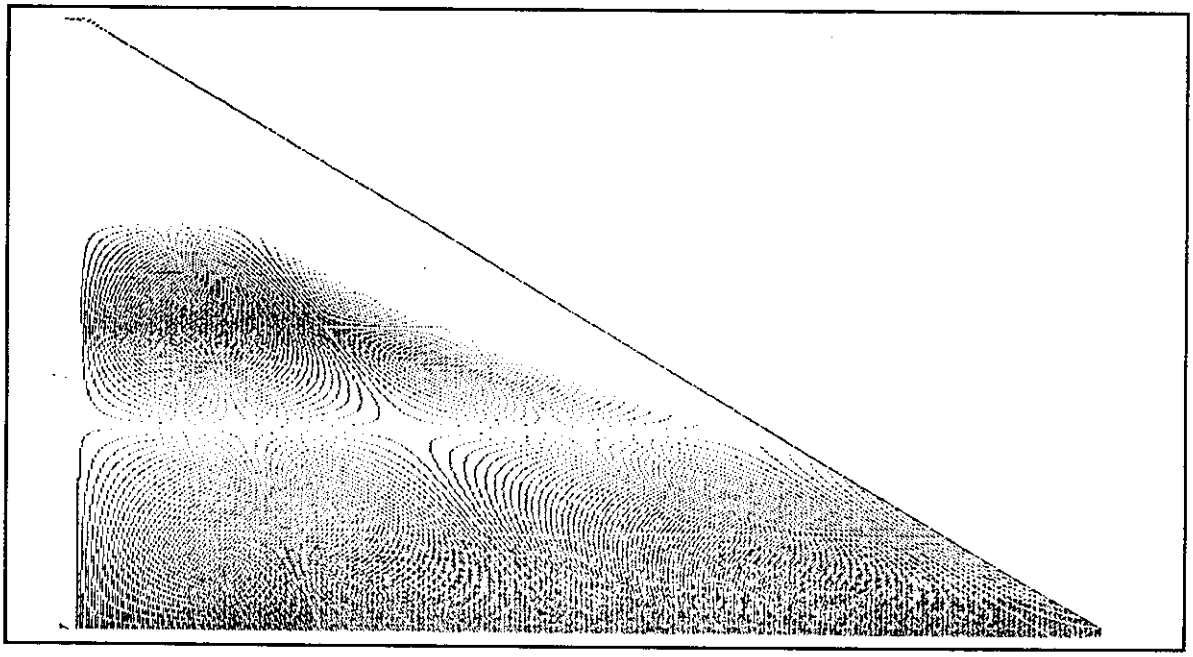
【注意】

・　実線を書く以外は、ライン・スタイル（（８）ＧＳＴＹＬＥ関数）であらかじめスタイルを指定しておきます。

【例】

```
OUTPUT 31;"VS3"
CLS 1
FOR I=2 TO 1023 STEP 3
  GLINE(0,1,2,0,I,479)
  GLINE(0,1,0,0,I-2,479)
NEXT I
STOP
```

【実行結果】



（4）GMKR

【機能】

指定した位置にマーカ（ノーマル、Δ）を描きます。

【書式】

GMKR（MK，　D，　X，　Y）

MK　：マーカ種類　0：ノーマル・マーカ
　　1：Δ　マーカ
D　：0：消去
　　1：描く
X，Y：座標

【エラー】

- X，Yが、描写画面の範囲外に設定されると、エラーになります。（ビューポート・アドレス時は注意して下さい。）
- エラー時は、メッセージを出力するだけで、実行は停止しません。

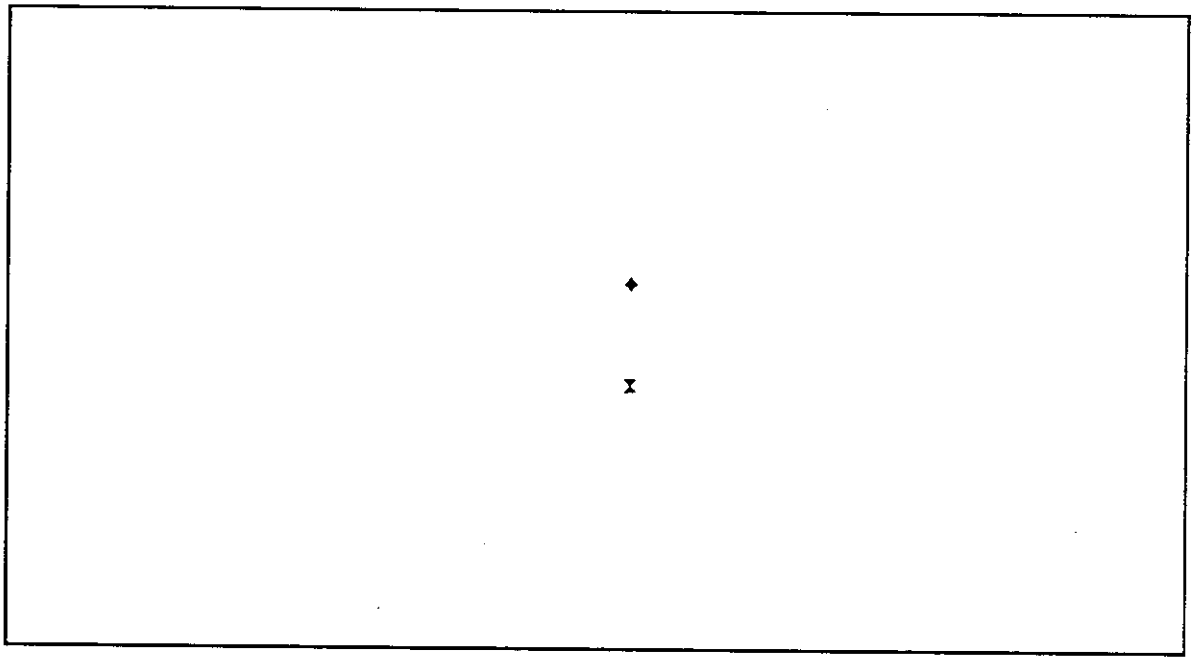
【注意】

- マーカのサイズは、１１×１１ドットです。
- マーカの中心が指定した位置に描かれます。

【例】

```
OUTPUT 31;"VS3"
CLS 1
GMKR(0,1,512,200)
GMKR(1,1,512,280)
```

【実行結果】



（5）ＧＰＯＩＮＴ

【機能】

指定した位置に点を描きます。

【書式】

ＧＰＯＩＮＴ（Ｄ，　Ｘ，　Ｙ）

Ｄ　　：０：消去
　　１：描く
Ｘ，Ｙ：座標

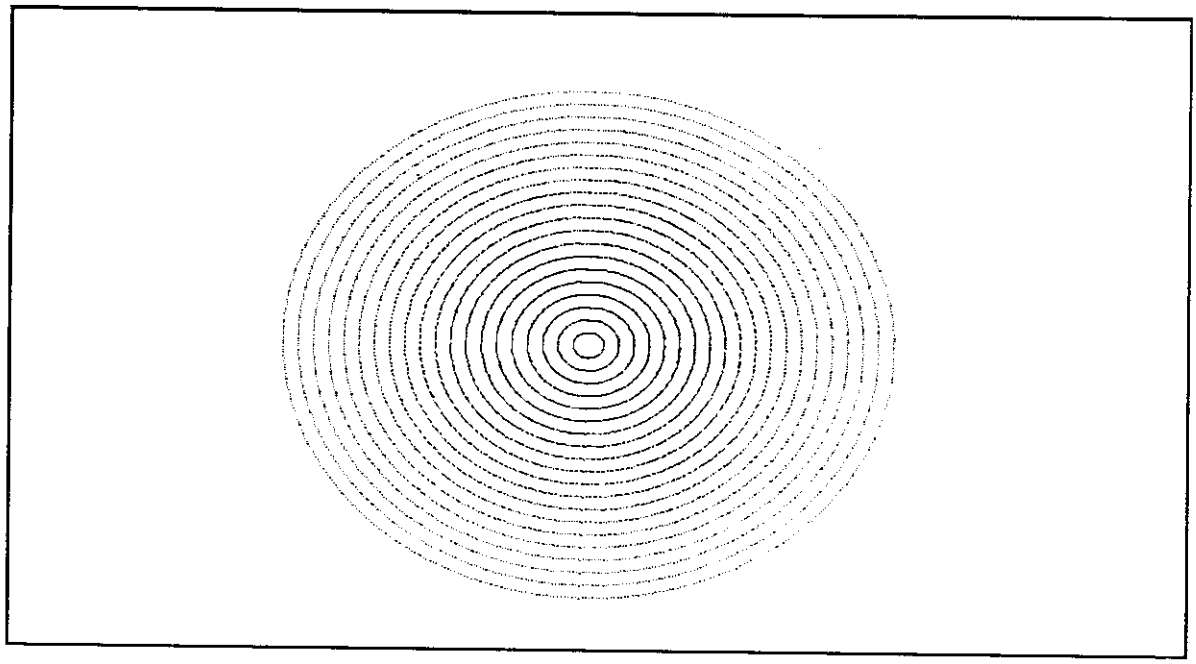
【エラー】

- Ｘ，Ｙが、描写画面の範囲外に設定されると、エラーになります。（ビューポート・アドレス時は注意して下さい。）
- エラー時は、メッセージを出力するだけで、実行は停止しません。

【例】

```
OUTPUT 31;"VS3"
CLS 1
FOR R=200 TO 1 STEP -10
  FOR I=0 TO PI*2 STEP PI/270
    X=SIN(I)*R*1.5+512
    Y=COS(I)*R+240
    GPOINT(1,X,Y)
  NEXT I
NEXT R
```

【実行結果】



（6）ＧＲＥＣＴ

【機能】

指定された２点間（座標１、座標２）を対角線とする長方形を描きます。

【書式】

ＧＲＥＣＴ（Ｓ，　Ｄ，　Ｘ１，　Ｙ１，　Ｘ２，　Ｙ２）

Ｓ　　：スタイル　０：実線
　　１：破線
　　２：点線
　　３：一点鎖線
Ｄ　　：０：消去
　　１：描く
Ｘ１，Ｙ１：座標１
Ｘ２，Ｙ２：座標２

【エラー】

・　Ｘ，Ｙが、描写画面の範囲外に設定されるとエラーになります。（ビューポート・アドレス時は注意して下さい。）
・　エラー時は、メッセージを出力するだけで、実行は停止しません。

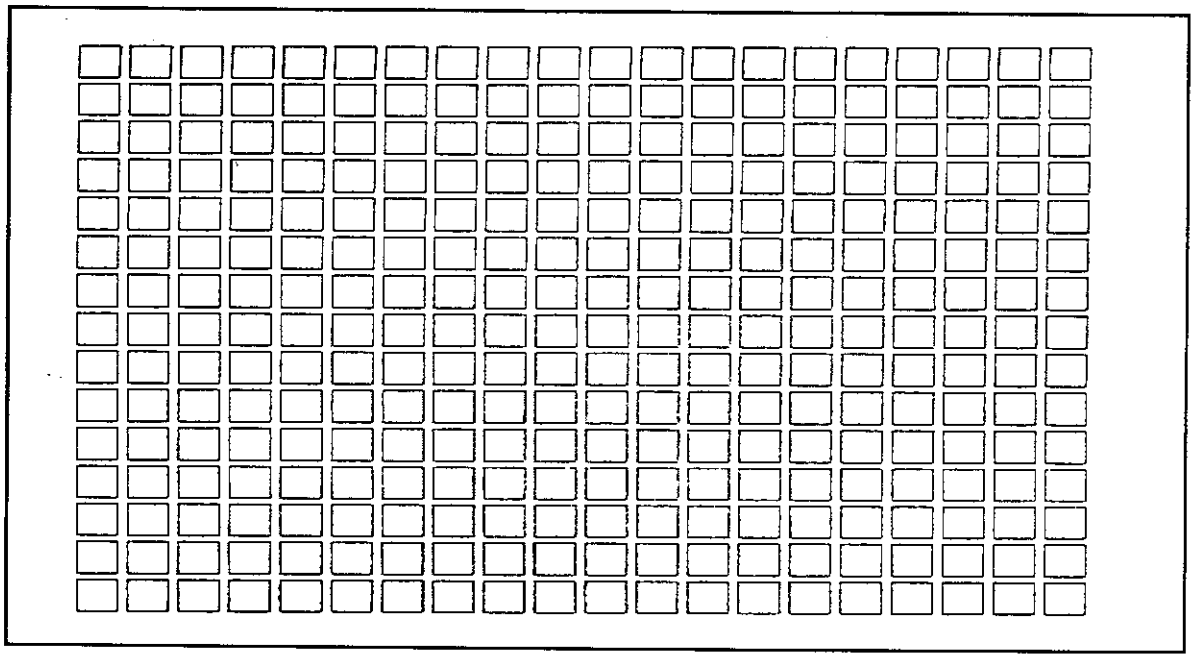
【注意】

・　実線を書く以外は、ライン・スタイル（（８）ＧＳＴＹＬＥ関数）であらかじめスタイルを指定しておきます。

【例】

```
OUTPUT 31;"VS3"
CLS 1
INTEGER X,Y
FOR X=10 TO 970 STEP 50
  FOR Y=10 TO 450 STEP 30
    GRECT(0,1,X,Y,X+40,Y+25)
  NEXT Y
NEXT X
STOP
```

【実行結果】



(7).GSTR

【機能】

グラフィック画面の指定位置に文字を描きます。

【書式】

GSTR(C, X, Y, STR)

C　：文字サイズ　0：16×20ドット
　　1：18×24ドット
X　：横軸（絶対アドレス：0～1023）
Y　：縦軸（絶対アドレス：0～479）
STR：文字列表現式（18文字まで）

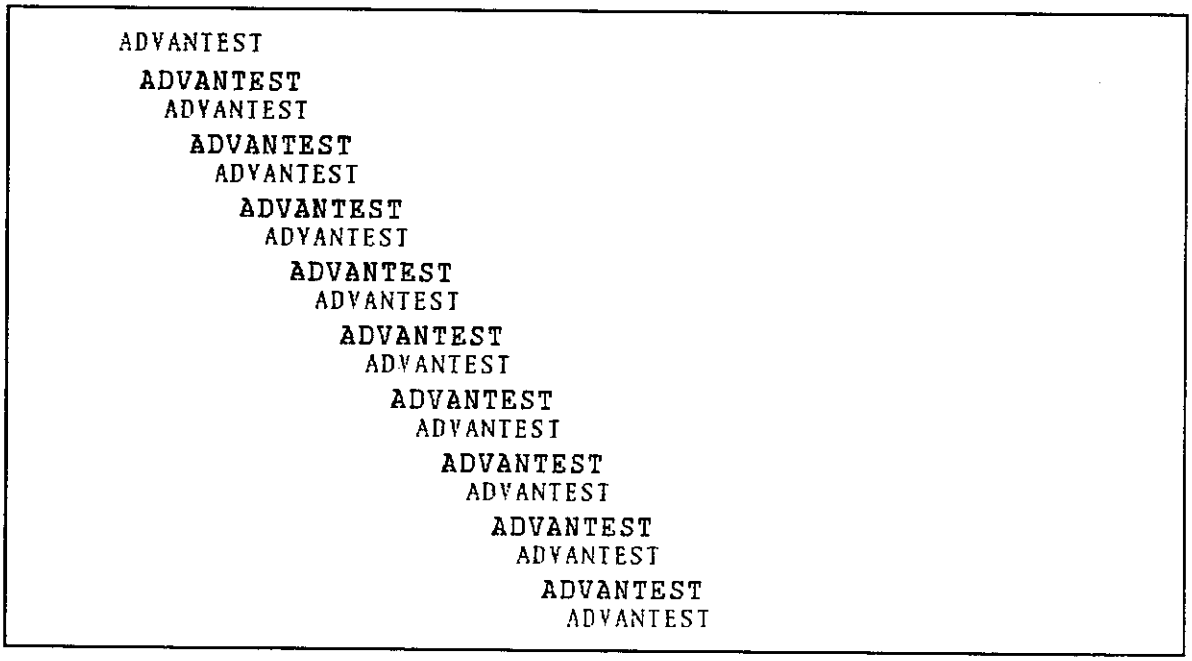
【エラー】

- X，Yが、描写画面の範囲外に設定されると、エラーになります。（ビューポート・アドレス時は注意して下さい。）
- エラー時は、メッセージを出力するだけで、実行は停止しません。

【例】

```
OUTPUT 31;"VS3"
S$="ADVANTEST"
FOR I=0 TO 450 STEP 25
 GSTR(I%2,I,I,S$)
NEXT I
STOP
```

【実行結果】



（8）ＧＳＴＹＬＥ

【機能】

破線、点線および一点鎖線のそれぞれの要素の長さを指定します。

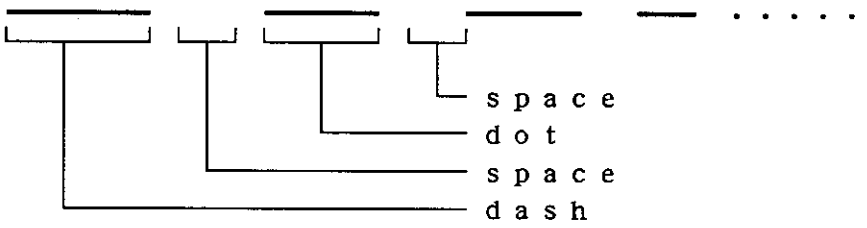
【書式】

ＧＳＴＹＬＥ（ｄａｓｈ，　ｓｐａｃｅ，　ｄｏｔ）

この関数は、破線、点線、および一点鎖線のそれぞれの要素の長さを指定するものです。それぞれの要素とは破線部、スペース部、点線部のことを差し、この要素は以下の組合せで構成されています。

破線　　：　ｄａｓｈ　＋　ｓｐａｃｅ
点線　　：　ｄｏｔ　＋　ｓｐａｃｅ
一点鎖線　：　ｄａｓｈ　＋　ｓｐａｃｅ　＋　ｄｏｔ　＋　ｓｐａｃｅ

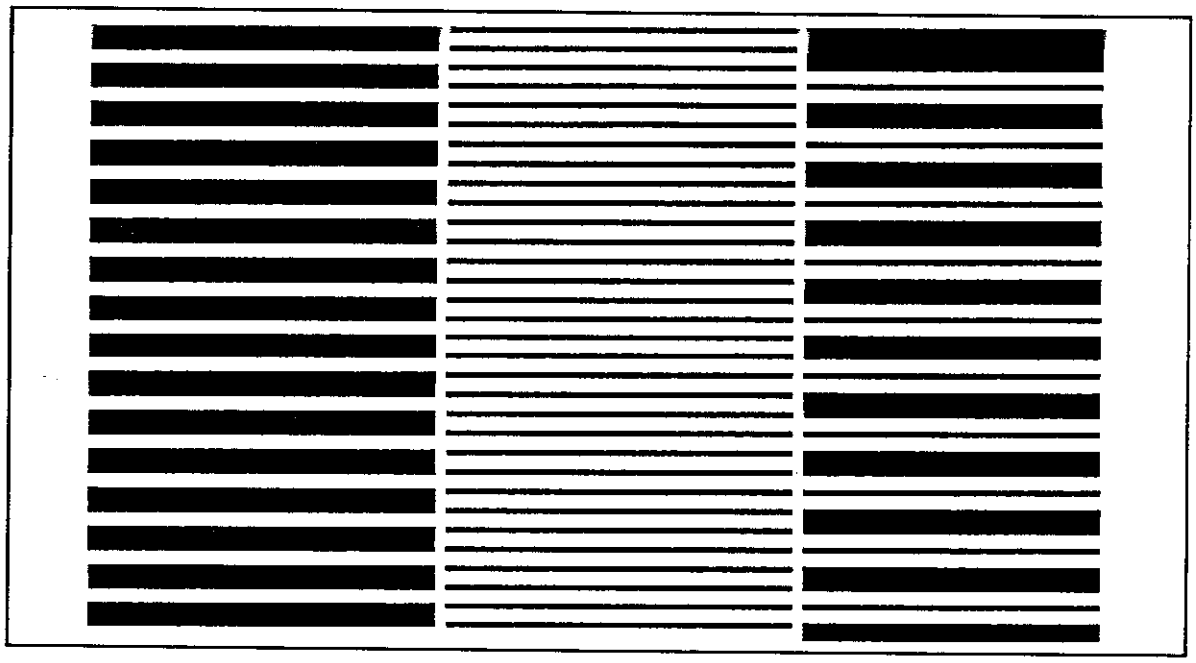
例）１点鎖線を指定のときは以下のようになります。



【例】

```
OUTPUT 31;"VS3"
CLS 1
S=0
GSTYLE(20,10,5)
FOR X=1 TO 1000
  IF X % 350 = 1 THEN ++S : X+=10
  GLINE(S,1,X,0,X,479)
NEXT X
STOP
```

【実行結果】



MEMO

４．　マスタ／スレーブ・モード

4.1 概要

マスタ・モード

このモードでは、Ｒ３２６５／３２７１がシステム・コントローラとなります。Ｒ３２６５／３２７１ＢＡＳＩＣプログラムを実行して、外部に接続されているＧＰＩＢ機器をコントロールすることができます。

スレーブ・モード

このモードでは、Ｒ３２６５／３２７１が被コントローラとなります。ＢＡＳＩＣプログラムを実行しながら外部コントローラとデータの送受信を行います。
（このモードでは、Ｒ３２６５／３２７１から外部に接続されているＧＰＩＢ機器をコントロールすることはできません。）

4.1.1 マスタ／スレーブ・モードの選択

SHIFT ＋ OPTION 6 ＋ NEXT MENU　と押すと次のソフト・メニューが表示されます。

MASTER/ SLAVE　マスタ・モードとスレーブ・モードの切り換えを行います。
（モードは、電源を切っても保持されます。）

4.2 ＢＡＳＩＣコマンド

マスタ・モード、スレーブ・モードの２つのモードでは、ＢＡＳＩＣコマンドの動作が異なります。

BASICｺﾏﾝﾄﾞ	マスタ・モード	スレーブ・モード
ＣＬＥＡＲ	デバイス・クリア	機能しません。*2
ＤＥＬＩＭＩＴＥＲ	デリミタの設定 OUTPUT,GPRINT,GLISTのデリミタを設定します。	
ＥＮＴＥＲ	指定装置をトーカに指定し、ＧＰＩＢポートからＡＳＣＩＩデータを入力します。	外部コントローラからリスナの指定を受けたとき、ＧＰＩＢポートからＡＳＣＩＩデータを入力します。*1
INTERFACE CLEAR	ｲﾝﾀﾌｪｰｽ･ｸﾘｱ	機能しません。*2
ＬＯＣＡＬ	ローカル	機能しません。*2
LOCAL LOCKOUT	ﾛｰｶﾙ･ﾛｯｸｱｳﾄ	機能しません。*2
ＯＵＴＰＵＴ	指定装置をリスナに指定し、ＧＰＩＢポートからＡＳＣＩＩデータを出力します。	外部コントローラからトーカの指定を受けたとき、ＧＰＩＢポートからＡＳＣＩＩデータを出力します。*1
ＲＥＭＯＴＥ	リモート	機能しません。*2
ＲＥＱＵＥＳＴ	機能しません。*2	外部ｺﾝﾄﾛｰﾗへSRQを出力します。
SEND-DATA-CMD-TALK-LISTEN-UNT-UNL	ＡＴＮラインの操作	機能しません。*2
ＳＰＯＬＬ	シリアル・ポール	機能しません。 必ず　０を返します。
ＴＲＩＧＧＥＲ	トリガ	機能しません。*2
ＣＯＮＴＲＯＬ	オプション１５のＢＡＳＩＣプログラム内でＧＰＩＢアドレス、マスタ／スレーブの切り換えを行います。	

*1：スレーブ・モードのOUTPUT, ENTER命令のアドレスは無視します。
*2：「機能しません。」は、何もせずに次の命令に実行を移します。

4.3 スレーブ・モードのＧＰＩＢコマンド

(1) ＯＵＴＰＵＴ

外部コントローラからリスナ指定を受けたとき、数値表現式をＡＳＣＩＩデータに変換し、ＧＰＩＢポートに送信します。

ＯＵＴＰＵＴ命令の書式は、マスタ・モードと同一です。ただし、ＧＰＩＢアドレスは無視されます。

(2) ＥＮＴＥＲ

外部コントローラからトーカ指定を受けたとき、入力されたＡＳＣＩＩデータを指定変数に格納します。

ＥＮＴＥＲ命令の書式は、マスタ・モードと同一です。ただし、ＧＰＩＢアドレスは無視されます。

注意

スレーブ・モードのＯＵＴＰＵＴ、ＥＮＴＥＲ命令は、処理が終了するまで次の命令に移りません。（外部コントローラから指定を受けるまで待機します。）

(3) ＲＥＱＵＥＳＴ

外部コントローラへＳＲＱを出力します。

ＲＥＱＵＥＳＴ　＜数値　0-255＞

128	64	32	16	8	4	2	1
	RSV						

RSQ割り込みをホスト側に送る場合

例　REQUEST　64+1　または　REQUEST 65

(4) コントローラ・モード切り換えコマンド

工場出荷時のデフォルト値は、

オプション１５　ＧＰＩＢアドレス　：　３０
　　マスタ／スレーブ・モード　：　マスタ・モード
（切り換え後は、データを保持します。）

ＢＡＳＩＣコマンドでの切り換え（ＣＯＮＴＲＯＬ命令を使用します。）

ＧＰＩＢアドレス変更　　：　ＣＯＮＴＲＯＬ　４；　［０～３０］

マスタ／スレーブ・モード変更：　ＣＯＮＴＲＯＬ　５；　［０，１］
０：スレーブ・モード
１：マスタ・モード

例）

ＣＯＮＴＲＯＬ　４；５　……　ＧＰＩＢアドレス５に変更
ＣＯＮＴＲＯＬ　５；０　……　スレーブ・モード

ＣＯＮＴＲＯＬ　４；２０……　ＧＰＩＢアドレス２０に変更
ＣＯＮＴＲＯＬ　５；１　……　マスタ・モード

4.4 外部コントローラでのコントロール

外部コントローラからＲ３２６５／３２７１オプション１５のＢＡＳＩＣプログラムを起動させる場合、以下のコマンドが使用できます。（ただし、ＯＵＴＰＵＴ、ＥＮＴＥＲ命令が動作していないときに限ります。）

コマンド	機能
"@MOVERUN"	ＭＯＶＥ＆ＲＵＮ
"@RUN"	ＲＵＮ
"@STOP"	プログラム停止
"@CR"	改行

上記のコマンドで、@マークに続けてスペースを空けずにコマンドを続けて書きます。@マークを付けないでコマンドを送るとＧＰＩＢバスが動作不能となります。（復旧の方法は、Ｒ３２６５／３２７１のＳＴＯＰキーを押すと復帰します。）

注意

１．　ＯＵＴＰＵＴ、ＥＮＴＥＲ命令実行中は上記コマンドは動作しません。
２．　外部コントローラとの通信を目的としたプログラムでは、Ｒ３２６５／３２７１側のプログラムを手動または上記コマンドにて先に起動させて下さい。

プログラム例－１

外部コントローラ（例：ＰＣ９８０１）

```
1000 ISET IFC
1010 ISET REN
1020 PRINT @9;"@MOVERUN"
1030 PRINT @9;"START"
1040 INPUT @9;A
1050 PRINT A
1060 GOTO 1040
```

R３２６５／３２７１（スレーブ・モード　アドレス９）

```
ENTER 0;A$
PRINT A$
*L
X=MAX(0,700,0)
OUTPUT 0;X
GOTO *L
```

外部コントローラ側のプログラムを起動させます。

"@MOVERUN"でＲ３２６５／３２７１側のＢＡＳＩＣプログラムが起動します。

Ｒ３２６５／３２７１側プログラムのENTER 0;A$命令で、外部コントローラ側プログラムとの同期をとります。
(A$には、"START"が入力されます。外部コントローラからデータが入力されるまで待ち状態となります。)

OUTPUT 0:Xで、Ｒ３２６５／３２７１側の最大値を送信可能状態にします。外部コントローラ側の INPUT @9;A命令でＲ３２６５／３２７１をトーカに指定するとＲ３２６５／３２７１は最大値を送信します。以後繰り返し処理をします。

注意

スレーブ・モードのためＯＵＴＰＵＴ、ＥＮＴＥＲもアドレスが０になっていますが、無視しますので動作には影響しません。

プログラム例－２

外部コントローラ（例：ＰＣ９８０１）

```
1000 ISET IFC
1010 ISET REN
1020 SPA=9:POLL SPA,RSV
1030 ON SRQ GOSUB *INTR
1040 PRINT @SPA;"@MOVERUN"
1050 PRINT @SPA;"START"
1060 *L
1070  S=-63
1080  W=.1
1090  PRINT @SPA;S
1100  PRINT @SPA;W
1110  F=1
1120  WHILE F
1130  WEND
1140  GOTO *L
1150 *INTR
1160  POLL SPA,RSV
1170  IF RSV-64 THEN INPUT @SPA;LV:PRINT "FALL LEVEL= ";LV ELSE PRINT "PASS"
1180  F=0
1190  SRQ ON:RETURN
```

プログラムの説明

1000～1030	初期設定
1040	Ｒ３２６５／３２７１側のＢＡＳＩＣプログラムを起動します。
1050	Ｒ３２６５／３２７１との同期をとるための通信します。
1070～1080	基準値、許容値を設定します。
1090～1100	基準値、許容値をＲ３２６５／３２７１へ送信します。
1110～1130	サービス・リクエスト割り込み待ちループ
1150	サービス・リクエスト割り込み処理
1160	シリアル・ポール
1170	シリアル・ポール値が６４ならばＰＡＳＳ、６４以外ならばＦＡＩＬとしてレベル値をＲ３２６５／３２７１から受信し表示します。
1180	ループフラグＯＦＦ
1190	割り込み許可：リターン

R３２６５／３２７１（スレーブ・モード　アドレス９）

```
INTEGER I
OUTPUT 31;"VS0"          "VS0"で波形画面に設定します。
ENTER 0;A$
*L
 ENTER 0;S               外部コントローラからの基準値を変数Sへ代入します。
 ENTER 0;W               外部コントローラからの許容値を変数Wへ代入します。
 M=0
 FOR I=1 TO 10           ┐
   OUTPUT 31;"TS"        │ TSコマンドを使用し、最大値を10回測定し合計します。
   M += MAX(0,700,0)     ┘
 NEXT I
 M /= 10                 10回の平均を計算します。
 J=LMTMD(M,S,W)          平均値が基準値±許容値内か判定します。
 IF J THEN
       REQUEST 64+J      許容値外であれば、リクエスト64+J（判定値）、レベル出力
       OUTPUT 0;M
 ELSE
       REQUEST 64        許容値内であれば、リクエスト64
 END IF
 GOTO *L
```

付　録

付 録 の 目 次

A.1 機能別コマンドとステートメント一覧

機　　能	コマンドとステートメント	内　　容
①コマンド	CONT	プログラム停止後の再実行を行う。
	CONTROL	各制御に関する値を設定する。
	LIST	プログラム・リストを出力する。
	LLIST	プログラム・リストを出力する。(RS-232C)
	LISTN	プログラム・リストを出力する。
	LLISTN	プログラム・リストを出力する。(RS-232C)
	RUN	プログラムを実行させる。
	SCRATCH	ＢＡＳＩＣ内のプログラムを消去する。
	STEP	プログラムを１行実行する。
②算術関数	ABS	与えられた値の絶対値を求める。
	ATN	与えられた値の逆正接値を求める。
	COS	与えられた値の余弦値を求める。
	LOG	与えられた値の自然対数を求める。
	SIN	与えられた値の正弦値を求める。
	SQR	与えられた値の平方根を求める。
	TAN	与えられた値の正接値を求める。
③ビット演算	BAND	ビットＡＮＤを求める。
	BNOT	ビットＮＯＴを求める。
	BOR	ビットＯＲを求める。
	BXOR	ビットＸＯＲを求める。
④割り込み制御	ENABLE INTR	割り込み受信許可状態にする。
	DISABLE INTR	割り込み受信禁止状態にする。
	ON END	ＥＯＦ割り込みの分岐を定義する。
	ON KEY	キー割り込みの分岐を定義する。
	ON ISRQ	本体計測部ＳＲＱ割り込みの分岐を定義する。
	ON SRQ	ＧＰＩＢ、ＳＲＱ割り込みの分岐を定義する。
	ON ERROR	エラー発生割り込みの分岐を定義する。
	OFF END	ＥＯＦ割り込みの分岐定義を解除する。
	OFF KEY	キー割り込みの分岐定義を解除する。
	OFF ISRQ	本体計測部ＳＲＱ割り込みの分岐定義を解除する。
	OFF SRQ	ＧＰＩＢ、ＳＲＱ割り込みの分岐定義を解除する。
	OFF ERROR	エラー発生割り込みの分岐定義を解除する。
⑤文字列操作	NUM	文字列の先頭の文字のＡＳＣＩＩコードを求める。
	CHR$	数値をＡＳＣＩＩ文字に変換する。
	LEN	文字列の長さを求める。
	POS	文字列２の中から文字列１がある位置を求める。
	SPRINTF	書式を決め文字列変数に入力する。

機　能	コマンドとステートメント	内　容
⑥メモリカード制御	CAT	メモリ・カードの内容を表示する。
	CALL	サブ・プログラムをロードする。
	CLOSE #	ファイルを閉じる。
	ENTER #	ファイルからデータを読み込む。
	OPEN #	ファイルに開く。
	OUTPUT #	ファイルにデータを書き込む。
	INITIALIZE(INIT)	メモリ・カードを初期化する。
	PURGE	指定ファイルを消去する。
	RENAME	ファイル名を変更する。
⑦画面制御	CURSOR(CSR)	指定位置にカーソルを移動する。
	CLS	画面を消去する。
⑧ステートメント	BUZZER	ブザーを鳴らす。
	DIM	配列変数を宣言する。
	FOR TO STEP NEXT	繰り返し処理を設定する。
	BREAK	繰り返し処理から抜ける。
	CONTINUE	繰り返し処理を先頭に戻す。
	GOSUB	サブ・ルーチン分岐する。
	RETURN	サブ・ルーチンからの復帰する。
	GOTO	指定位置へ分岐する。
	IF THEN ELSE END IF	条件判断をして処理する。
	INPUT(INP)	変数への入力を行う。
	INTEGER	整数型変数の宣言する。
	LPRINT	プリンタ(RS232C)出力する。
	LPRINT USING(USE)	プリンタ(RS232C)出力する。（書式指定）
	PAUSE	一時停止する。
	PRINT(?)	文字を画面に出力する。
	PRINT USING(USE)	文字を画面に出力する。（書式指定）
	PRINTER	ＧＰＩＢプリンタ装置のアドレス指定を行う。
	PRINTF(PRF)	文字を画面に出力する。（書式指定）
	READ DATA	ＤＡＴＡ文からデータを読み取り、変数へ入力する。
	RESTORE	ＲＥＡＤ文で読むＤＡＴＡ文を指定する。
	REM(!)	注釈文
	SELECT CASE END SELECT	条件判断を行い処理する。
	STOP	プログラム実行停止する。
	WAIT	指定時間、実行停止する。

機　　能	コマンドとステートメント	内　　容
⑨ＧＰＩＢ関係コマンド	CLEAR	ＤＣＬ、ＳＤＣを送出する。
	DELIMITER	デリミタを設定する。
	ENTER(ENT)	ＧＰＩＢデータを入力する。(パラレルI/O含む)
	GLIST	ＧＰＩＢプリンタへプログラム・リストを出力する。
	GLISTN	ＧＰＩＢプリンタへプログラム・リストを出力する。
	GPRINT	データをＧＰＩＢプリンタへ出力する。
	GPRINT USING(USE)	データをＧＰＩＢプリンタへ出力する。（書式指定）
	INTERFACE CLEAR	ＩＦＣを送出する。
	LOCAL	指定装置をローカル状態にする。
	LOCAL LOCKOUT	指定装置をローカル・ロックアウト状態にする。
	OUTPUT(OUT)	データをＧＰＩＢに出力する。（パラレルI/O含む）
	REMOTE	指定装置をリモート状態にする。
	REQUEST	標準ＧＰＩＢにＳＲＱを出力する。
	SEND	ＧＰＩＢデータを１つずつ出力する。
	SPOLL	指定装置のシリアル・ポールを行う。
	TRIGGER	ＧＥＴを送出する。

A.2 機能別ビルトイン関数とグラフィック機能一覧

『周波数／ポイント（横軸）を求める』

No	関　数	内　容
1	ＦＲＥＱ	ポイントから周波数を求める
2	ＤＦＲＥＱ	ポイント間の周波数幅を求める
3	ＰＯＩＮＴ	周波数からポイントを求める
4	ＤＰＯＩＮＴ	周波数間のポイント幅を求める

『レベル／ポイント（縦軸）を求める』

No	関　数	内　容
5	ＬＥＶＥＬ	ポイントからレベルを求める
6	ＤＬＥＶＥＬ	ポイント間のレベル差を求める
7	ＬＶＰＯＩＮＴ	レベルからポイントを求める
8	ＬＶＤＰＯＩＮＴ	レベル間のポイント幅を求める
9	ＶＡＬＵＥ	横軸ポイント位置の波形のレベルを求める
10	ＤＶＡＬＵＥ	横軸ポイント間の波形のレベル差を求める
11	ＣＶＡＬＵＥ	周波数位置の波形のレベルを求める
12	ＤＣＶＡＬＵＥ	周波数間の波形のレベル差を求める

『最大／最小を求める』

No	関　数	内　容
13	ＦＭＡＸ	ポイント間の最大レベル位置の周波数を求める
14	ＦＭＩＮ	ポイント間の最小レベル位置の周波数を求める
15	ＰＭＡＸ	ポイント間の最大レベル位置の横軸ポイントを求める
16	ＰＭＩＮ	ポイント間の最小レベル位置の横軸ポイントを求める
17	ＭＡＸ	ポイント間の最大レベルを求める
18	ＭＩＮ	ポイント間の最小レベルを求める

『バンド幅を求める』

No	関　数	内　容
19	ＢＮＤ	横軸ポイント位置のロスレベルの帯域幅を求める
20	ＢＮＤＬ	横軸ポイント位置のロスレベルの低周波数側を求める
21	ＢＮＤＨ	横軸ポイント位置のロスレベルの高周波数側を求める
22	ＣＢＮＤ	周波数位置のロスレベルの帯域幅を求める
23	ＣＢＮＤＬ	周波数位置のロスレベルの低周波数側を求める
24	ＣＢＮＤＨ	周波数位置のロスレベルの高周波数側を求める

『極大／極小（リップル）を求める』

No	関　数	内　容
25	ＮＲＰＬＨ	すべての極大点の数を求める
26	ＮＲＰＬＬ	すべての極小点の数を求める
27	ＰＲＰＬＨＮ	左からＮ番目の極大点の横軸ポイントを求める
28	ＰＲＰＬＬＮ	左からＮ番目の極小点の横軸ポイントを求める
29	ＦＲＰＬＨＮ	左からＮ番目の極大点の周波数を求める
30	ＦＲＰＬＬＮ	左からＮ番目の極小点の周波数を求める
31	ＶＲＰＬＨＮ	左からＮ番目の極大点のレベルを求める
32	ＶＲＰＬＬＮ	左からＮ番目の極小点のレベルを求める
33	ＲＰＬ１	極大点の最大値と極小点の最小値のレベル差を求める

『上下限の判定を行う』

No	関　数	内　容
34	ＬＭＴＭＤ１	指定したデータについて基準値と上下幅により判定を行う
35	ＬＭＴＭＤ２	横軸ポイント位置の波形データについて基準値と上下幅により判定を行う
36	ＬＭＴＵＬ１	指定したデータについて上限値と下限値により判定を行う
37	ＬＭＴＵＬ２	横軸ポイント位置の波形データについて上限値と下限値により判定を行う

『電力を求める』

No	関　数	内　容
38	ＰＯＷＥＲ	横軸ポイント間の総電力を求める

『トレースデータ』

No	関　数	内　容
39	ＲＴＲＡＣＥ	指定ポイントのトレースデータを読み込む
40	ＷＴＲＡＣＥ	指定ポイントのトレースデータを書き込む

『グラフィック機能』

No	関　数	内　容
1	ＧＡＤＲＳ	グラフィック・ポイントの絶対アドレス/ビューポートアドレスを指定する
2	ＧＦＬＲＥＣＴ	指定２点間を対角線とする長方形を塗りつぶす
3	ＧＬＩＮＥ	指定２点間に直線を描く
4	ＧＭＫＲ	指定した位置にマーカ（ノーマル／デルタ）を描く
5	ＧＰＯＩＮＴ	指定した位置に点を描く
6	ＧＲＥＣＴ	指定２点間を対角線とする長方形を描く
7	ＧＳＴＲ	文字列を描く
8	ＧＳＴＹＬＥ	破線，点線，１点鎖線の要素の長さを指定する

A.3　パラメータ指定一覧

＜ビルトイン関数＞

機　能	No	関　数
周波数／ポイントを求める	1	F＝FREQ（P）
	2	F＝DFREQ（P1，P2）
	3	P＝POINT（F）
	4	P＝DPOINT（F1，F2）
レベル／ポイントを求める	5	L＝LEVEL（T）
	6	L＝DLEVEL（T1，T2）
	7	T＝LVPOINT（L）
	8	T＝LVDPOINT（L1，L2）
	9	L＝VALUE（P，M）
	10	L＝DVALUE（P1，P2，M）
	11	L＝CVALUE（F，M）
	12	L＝DCVALUE（F1，F2，M）
最大／最小を求める	13	F＝FMAX（P1，P2，M）
	14	F＝FMIN（P1，P2，M）
	15	P＝PMAX（P1，P2，M）
	16	P＝PMIN（P1，P2，M）
	17	L＝MAX（P1，P2，M）
	18	L＝MIN（P1，P2，M）
バンド幅を求める	19	F＝BND（P，X，M）
	20	F＝BNDL（P，X，M）
	21	F＝BNDH（P，X，M）
	22	F＝CBND（F，X，M）
	23	F＝CBNDL（F，X，M）
	24	F＝CBNDH（F，X，M）
極大／極小を求める	25	N＝NRPLH（P1，P2，Dx，Dy，M）
	26	N＝NRPLL（P1，P2，Dx，Dy，M）
	27	P＝PRPLHN（N，M）
	28	P＝PRPLLN（N，M）
	29	F＝FRPLHN（N，M）
	30	F＝FRPLLN（N，M）
	31	L＝VRPLHN（N，M）
	32	L＝VRPLLN（N，M）
	33	L＝RPL1（P1，P2，Dx，Dy，M）
上下限の判定を行う	34	C＝LMTMD1（Dd，S，Ds）
	35	C＝LMTMD2（P，S，Ds，M）
	36	C＝LMTUL1（Dd，Up，Lo）
	37	C＝LMTUL2（P，Up，Lo，M）
電力を求める	38	W＝POWER（P1，P2，M）
トレースデータのread／write	39	T＝RTRACE（P，M）
	40	WTRACE（T，P，M）　　注）この関数は値を返しません。

Ｆ：周波数
Ｐ：ポイント（０～７００）
Ｌ：レベル
Ｔ：トレース・データ（０～４００）
Ｍ：トレースＡ／Ｂ（０／１）
Ｘ：１ｏｓｓレベル
Ｃ：チェック値（０，１，２，－１）
Ｎ：リップル数
Ｗ：ワット（電力）

Ｄｘ：横軸微分係数
Ｄｙ：縦軸微分係数
Ｓ　：基準値
Ｄｓ：基準値幅
Ｄｄ：被検データ
Ｌｏ：下限値
Ｕｐ：上限値

【パラメータ指定一覧】

機　能	No	関　数
グラフィック	1	GADRS (Mo, X, Y)
	2	GFLRECT (D, X1, Y1, X2, Y2)
	3	GLINE (S, D, X1, Y1, X2, Y2)
	4	GMKR (MK, D, X, Y)
	5	GPOINT (D, X, Y)
	6	GRECT (S, D, X1, Y1, X2, Y2)
	7	GSTR (C, X, Y, STR)
	8	GSTYLE (dash, space, dot)

C　：文字サイズ　0 ; 16×20ドット
　　1 ; 18×24ドット

D　：セット／消去　0 ; 消去
　　1 ; セット（描く）

MK　：マーカ　0 ; ノーマル・マーカ
　　1 ; △マーカ

Mo　：アドレス・モード　0 ; 絶対アドレス
　　1 ; ビューポート・アドレス

S　：ライン・スタイル　0 ; 実線
　　1 ; 破線
　　2 ; 点線
　　3 ; 一点鎖線

STR　：文字列表現式
X　：座標（横軸）
Y　：座標（縦軸）
dash　：破線部
dot　：点線部
space　：スペース部

A.4　エラーメッセージ一覧

記述上の注意　xxx：文字列を表わします
　　yy　：数値を表わします

（1）ateエディタのエラー・メッセージ一覧

エラーメッセージ	説　　明
Already auto line no. mode	すでに自動行番号挿入モードであるにもかかわらず、再度自動行番号挿入モードにしようとした
Cannot allocate バイト数 bytes	エディタ内部に確保できるメモリがない
Cannot allocate memory	エディタ内部に確保できるメモリがないため、リナンバリングできない
Cannot allocate WINDOW block	エディタ内部に確保できるメモリがないため、新しいウィンドウをオープンできない
Cannot create buffer	エディタ内部に確保できるメモリがない
Cannot find error line	BASIC実行時のエラー行が、ateエディタ内プログラムにない
Cannot open file for writing	メモリカードがない、 WRITE PROTECTがONなどにより、セーブするファイルがオープンできない
Cannot sprit a ライン数 line window	ウィンドウの行数が少ないために、スプリットできない
Line no. is out of range	行数が65535を超えている
No file name	メモリカードにセーブするとき、ファイル名がない
No mark set in this window	削除、コピーなど範囲指定のマークがない
Not found	検索する文字列がない
Not line no. mode	リナンバリングを実行したとき、自動行番号挿入モードでなかった
Too large region	削除、コピーなど範囲指定の範囲が大きすぎる
Write I/O error	メモリカードがない、電池切れ等により、メモリカードをアクセスできない

（２）.システムコントローラのエラー・メッセージ一覧

（１／４）

エラー・メッセージ	内　　容
yy error(s) appeared	ラベル、行ナンバのエラー
"xxx" file cannot be opened.	オープンしようとするファイルがない
"xxx" file is already opened with another PATH.	すでにオープンしてあるファイルをオープンしようとした
"xxx" file is already exist.	オープン時のモードと使用した命令が違う
"xxx" read error.	読み込みエラー
0 divide	０による除算（ｎ／０）が行われた
Array's range error	配列変数の添字が宣言よりも大きい
Bad free call	メモリ管理上のエラー
CANNOT assigned into this token	文字列変数に入力できない
cannot read data from "xxx" file.	読み込むべきファイルがない
cannot specify "USING"	読み込みバイト数と読み込んだバイト数が違う
cannot write data into "xxx" file.	書き込むべきファイルがない
end of "xxx" file	ＥＯＦ（Ｅｎｄ　Ｏｆ　Ｆｉｌｅ）まで読み込んだ
expression format error	書式の表現が誤っている
file format error	２５６文字以内にターミネータがない
file is NOT open.	オープンしないで書き込み／読み込みを実行した
FOR <init value> does NOT exist	ＦＯＲ文の初期値がない
FOR variable does NOT exist.	ＦＯＲ文のカウンタ変数がない
FOR's nest is abnormal.	ＦＯＲ文が入れ子（ｎｅｓｔ）できない
Invalid dimension parameter	配列変数のパラメータが誤っている
Invalid string constant	ダブル・クォーテーションがあわない
invalid type in xxx	ｘｘｘの中に無効なものがある

エラー・メッセージ	内　　容
label not found	指定したラベルがない
Label xxx is already exists	ｘｘｘのラベルはすでに存在している
Line No.yy is out of range.	行番号の指定が範囲を超えている
memory space full	メモリに空き領域がない
NO operand in xxx	ｘｘｘの演算書式が間違っている
NOT available ASCII char(xx)	アスキー・コードが有効でない
Not found DATA statement	ＲＥＳＴＯＲＥする先にＤＡＴＡ文がない
Not found THEN in xxx	ＩＦ文の後ろにＴＨＥＮがない
Only one INPUT file can be opened.	読み込みモードで２つ以上オープンしようとした
Only one OUTPUT file can be opened.	書き込みモードで２つ以上オープンしようとした
Overflow value	数値が扱える範囲を超えている
parameter error	パラメータが誤っている
Program CANNOT be continued.	終了したプログラムを再実行しようとした
Program NOT exist	プログラムが存在しないのに実行した
SELECT nesting overflow	ＳＥＬＥＣＴ文の入れ子が多すぎる
string declaration error	ダブル・クォーテーションがあわない
string length is too long	文字列変数の宣言が多すぎる(最大１２８)
Substring error	サブ・ストリングの指定が誤っている
Unbalanced BREAK	ＢＲＥＡＫ文がＦＯＲ～ＮＥＸＴ間にない
Unbalanced FOR variable in NEXT	ＦＯＲ文とＮＥＸＴ文の関係がおかしい
Unbalanced line No.	ＬＩＳＴで指定した行がない
Unbalanced NEXT statement	ＦＯＲ文があるのにＮＥＸＴ文がない

エラー・メッセージ	内　　容
Unbalanced xxx	構文上、バランス(カッコ、カギカッコ等)がとれない
Unbalanced xxx block	ｘｘｘのブロック（ＦＯＲ，ＩＦ文等）があわない
Undefined LABEL	ラベルが存在しない
undefined ON condition	ＯＮ条件が未定義の状態で、ＯＮ状態が発生した
Uninstalled type (xxx)	変数の書式が間違っている
Unknown line No.	指定行がない
Unmatched DATA's values and READ variable	ＲＥＡＤするＤＡＴＡ文がない
Unmatched IMAGE-spec in USING	ＵＳＩＮＧのイメージ使用が誤っている
xxx function error	ビルトイン関数のエラー
xxx nest overflow	入れ子（ｎｅｓｔ）が多すぎる
xxx(xxx) error	ＰＵＲＧＥ命令の指定ファイルがない
xxx(xxx,xxx) error	ＲＥＮＡＭＥ命令の指定ファイルがない
xxx: "xxx" file was opened with xxx mode.	ファイル・ディスクリプタのモード（書き込み／読み込み）が定義したものと合っていない
xxx: CANNOT convert into string	文字列に変換できない
xxx: invalid first type in xxx	コマンドの構文の最初が誤っている
xxx: invalid second type in xxx	コマンドの構文の２番目が誤っている
xxx: invalid source type in xxx	式の代入でソース側のタイプが誤っている
xxx: invalid target type in xxx	代入しようとする変数のタイプが誤っている
xxx: Invalid TERGET operand in XXX	ｘｘｘの中の書式に無効なものがある
xxx: Syntax error	文法が間違っている

エラー・メッセージ	内　　容
You cannot use POKE command	ＰＯＫＥ命令を実行しようとした
yy is invalid value in xxx	ｙｙ行目にあるｘｘｘの値が無効である
yy: Undefined Control Register	ＣＯＮＴＲＯＬ命令のレジスタ番号が誤っている
yy: UNIT addr error in xxx	ＧＰＩＢアドレス指定が誤っている

MEMO

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