

---

---

# ADVANTEST®

株式会社アドバンテスト

---

R3265A/71A シリーズ

コントローラ機能

取扱説明書

MANUAL NUMBER FOJ-8335209C01

---

適用機種

R3265A	R3271A
R3365A	R3371A
R3265AP	R3271AP
R3271MS	

禁無断複製転載

© 1993 年 株式会社アドバンテスト

初版 1993 年 12 月 20 日

Printed in Japan

---



## 本器を安全に取り扱うための注意事項

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

本器の操作・保守等の作業を行う場合、誤った方法で使用すると本器の保護機能がそこなわれることがあります。常に安全に心がけてご使用頂くようお願い致します。

### ■危険警告ラベル

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

危険警告ラベルのシグナル・ワードとその定義は、以下のとおりです。

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

### ■基本的注意事項

火災、火傷、感電、怪我などの防止のため、以下の注意事項をお守り下さい。

- 電源電圧に応じた電源ケーブルを使用して下さい。ただし、海外で使用する場合は、それぞれの国の安全規格に適合した電源ケーブルを使用して下さい。また、電源ケーブルの上には重いものをのせないで下さい。
- 電源プラグをコンセントに差し込むときは、電源スイッチを OFF にしてから奥までしっかり差し込んで下さい。
- 電源プラグをコンセントから抜くときは、電源スイッチを OFF にしてから、電源ケーブルを引っぱらずにプラグを持って抜いて下さい。このとき、濡れた手で抜かないで下さい。
- 電源投入前に、本器の電源電圧が供給電源電圧と一致していることを確認して下さい。
- 電源ケーブルは、保護導体端子を備えた電源コンセントに接続して下さい。保護導体端子を備えていない延長コードを使用すると、保護接地が無効になります。
- 3ピン - 2ピン変換アダプタ（弊社の製品には添付していません）を使用する場合は、アダプタから出ている接地ピンをコンセントのアース端子に接続し、大地接地して下さい。また、アダプタの接地ピンの短絡に注意して下さい。
- 電源電圧に適合した規格のヒューズを使用して下さい。
- ケースを開けたままで本器を使用しないで下さい。

## 本器を安全に取り扱うための注意事項

- 規定の周囲環境で本器を使用して下さい。
- 製品の上に物をのせたり、製品の上から力を加えたりしないで下さい。また、花瓶や薬品などの液体の入った容器を製品のそばに置かないで下さい。
- 通気孔のある製品については、通気孔に金属類や燃えやすい物などを差し込んだり、落としたりしないで下さい。
- 台車に載せて使用する場合は、ベルト等によって落下防止を行って下さい。
- 周辺機器を接続する場合は、本器の電源を切ってから接続して下さい。





### ■取扱説明書中の注意表記

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

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

### ■製品上の安全マーク

アドバンテストの製品には、以下の安全マークが付いています。

- ： 取扱注意を示しています。人体および製品を保護するため、取扱説明書を参照する必要がある場所に付いています。
- ： アース記号を示しています。感電防止のため機器を使用する前に、接地が必要なフィールド・ワイヤリング端子を示しています。
- ： 高電圧危険を示しています。1000V以上の電圧が入力または出力される場所に付いています。
- ： 感電注意を示しています。

### ■寿命部品の交換について

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

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

主な寿命部品と寿命

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

■ハード・ディスク搭載製品について

使用上の留意事項を以下に示します。

- 本器は、電源が入った状態で持ち運んだり、衝撃や振動を与えないで下さい。  
ハード・ディスクの内部は、情報を記録するディスクが高速に回転しながら、情報の読み書きを行っているため、非常にデリケートです。
- 本器は、以下の条件に合う場所で使用および保管をして下さい。  
 極端な温度変化のない場所  
 衝撃や振動のない場所  
 湿気や埃・粉塵の少ない場所  
 磁石や強い磁界の発生する装置から離れた場所
- 重要なデータは、必ずバックアップを取っておいて下さい。  
 取扱方法によっては、ディスク内のデータが破壊される場合があります。また、使用条件によりますが、ハード・ディスクには、その構造上、寿命があります。  
 なお、消失したデータ等の保証は、いたしかねますのでご了承下さい。

■本器の廃棄時の注意

製品を廃棄する場合、有害物質は、その国の法律に従って適正に処理して下さい。

- 有害物質： (1) PCB (ポリ塩化ビフェニール)  
 (2) 水銀  
 (3) Ni-Cd (ニッケル-カドミウム)  
 (4) その他

シアン、有機リン、六価クロムを有する物およびカドミウム、鉛、砒素を溶出する恐れのある物 (半田付けの鉛は除く)

例： 蛍光管、バッテリー

■使用環境

本器は、以下の条件に合う場所に設置して下さい。

- 腐食性ガスの発生しない場所
- 直射日光の当たらない場所
- 埃の少ない場所
- 振動のない場所
- 最大高度 2000 m

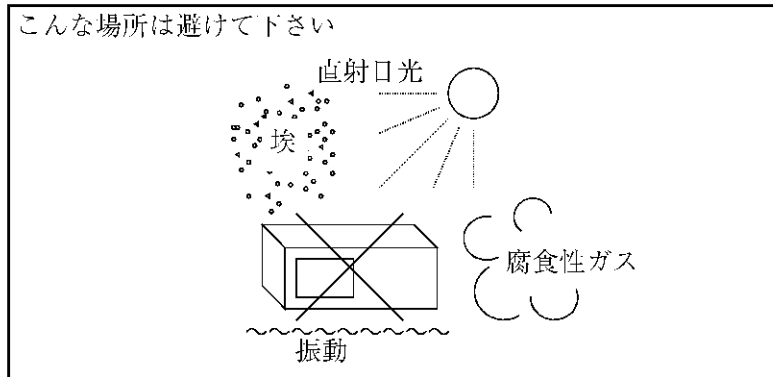


図-1 使用環境

●設置姿勢

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

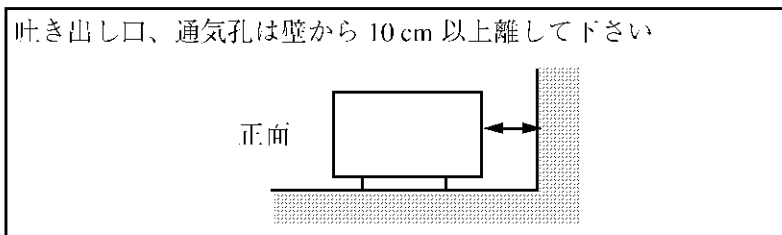


図-2 設置

●保管姿勢

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

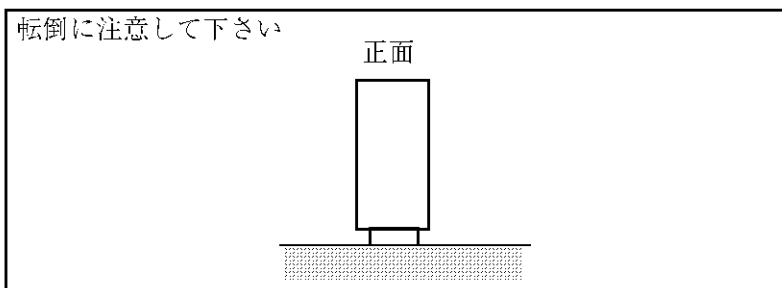
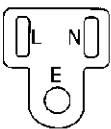

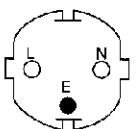
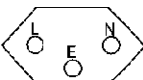
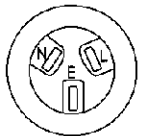
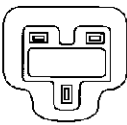
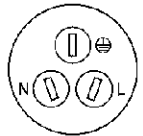


図-3 保管

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

■電源ケーブルの種類

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

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





R3265A/71A シリーズ

コントローラ機能

ガイド



## 結 言

1. 本書は、コントローラ機能をQ&A型式でわかりやすく説明しています。  
1章から順にお読みになると、コントローラ機能の詳細や実践的な操作方法が理解できます。

2. 本書の構成は、以下のようになります。

1. 使用開始前の準備
2. プログラムの作成
3. プログラムの実行
4. 応用操作

詳しい説明は別冊の「R3265A/3271Aシリーズ コントローラ機能（レファレンス）」を参照して下さい。

3. R3265A/3271Aシリーズ コントローラ機能に接続可能な外部キーボードは、101型キーボード（コネクタ形状 ミニDIN6ピン）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。
パソコン	: パーソナル・コンピュータを表わします。
外部キーボード	: 101型キーボード



# 目 次

1 . 使用開始前の準備 .....	1- 1
Q 1 プログラムを作るには、何を用意するのですか? .....	1- 1
Q 2 コントローラの起動は、どうするのですか? .....	1- 1
Q 3 メモリ・カードの初期化は、どうするのですか? .....	1- 2
2 . プログラムの作成 .....	2- 1
Q 4 プログラムの作成は、どうするのですか? .....	2- 1
Q 5 ホップアップメニューは、どう使うのですか? .....	2- 2
Q 6 このエディタは、何行のプログラムが作成できますか? .....	2- 2
Q 7 英小文字で作成したプログラムの実行結果が正しくありません。 .....	2- 3
Q 8 作成したプログラムをメモリ・カードにセーブするには、 どうするのですか? .....	2- 3
Q 9 ファイルがメモリ・カードにセーブできません。 .....	2- 4
Q 10 メモリ・カードからファイルをロードするには、どうするのですか? .....	2- 5
3 . プログラムの実行 .....	3- 1
Q 11 プログラムの実行は、どうするのですか? .....	3- 1
Q 12 実行結果を計測器の画面に表示するには、どうするのですか? .....	3- 2
Q 13 プログラムのソースリストをGPIBプリンタを印字するには、 どうするのですか? .....	3- 3
Q 14 実行結果をGPIBプリンタに印字するには、どうするのですか? .....	3- 4
Q 15 メモリ・カードから複数のプログラムを自動的にロードして実行するには どうするのですか? また、プログラムを内部メモリに常駐させて実行する には、どうするのですか? .....	3- 4
4 . 実戦テクニック .....	4- 1
Q 16 測定部へのデータ設定は、どうするのですか? .....	4- 1
Q 17 測定データを計測部から読み出すには、どうするのですか? .....	4- 1
Q 18 会話形式でプログラムを実行するには、どうするのですか? .....	4- 2
Q 19 他の計測器をGPIBで制御するには、どうするのですか? .....	4- 3
Q 20 プログラム実行時のデータをメモリ・カードにセーブするには、 どうするのですか? .....	4- 3
Q 21 メモリ・カードにセーブしてあるデータをロードするには、 どうするのですか? .....	4- 4
Q 22 トレース・データを変数に読み込むには、どうするのですか? .....	4- 4
Q 23 サービス・リクエスト (SRQ) は、どのように使うのですか? .....	4- 8
Q 24 パソコンで作成したプログラムをateエディタに転送するには、 どうするのですか? .....	4-10



## 目 次

図 1	外部キーボードの接続	1- 1
図 2	コントローラ (エディタ) の動作画面	1- 1
図 3	外部キーボード	2- 1
図 4	簡単なプログラム例	2- 1
図 5	ポップアップメニュー一覧	2- 2
図 6	ファイルのセーブ	2- 3
図 7	CATコマンドの実行	2- 4
図 8	ファイルのロード	2- 5
図 9	プログラムの実行結果	3- 1
図10	表示状態の設定	3- 2
図11	背面パネルの GPIBポート	3- 3
図12	ソースリストのプリンタ出力	3- 3
図13	実行結果のプリンタ出力	3- 4
図14	"OUTPUT 31"のサンプル・プログラム	4- 1
図15	"ENTER 31"のサンプル・プログラム	4- 1
図16	実行時会話形式のサンプル・プログラム	4- 2
図17	プロッタを制御するサンプル・プログラム	4- 3
図18	変数をセーブするサンプル・プログラム	4- 3
図19	データをロードするサンプル・プログラム	4- 4
図20	トレース・データを読み込むサンプル・プログラム	4- 4
図21	トレース・データを書き込むサンプル・プログラム	4- 5
図22	トレースデータとメモリ・カードとの読み書きのサンプル・プログラム	4- 5
図23	トレースとメモリ・カードとのデータ転送の関係	4- 5
図24	SRQを使ったサンプル・プログラム	4- 8
図25	背面パネルの GPIBポート	4-11





## 1. 使用開始前の準備

Q 1. プログラムを作るには、何を用意するのですか？

A 1. コントローラ機能 (ateエディタ+BASICインタプリタ) を使ってBASICプログラムを作るには、計測器と外部キーボードが必要です。計測器の背面パネルにあるKEYBOARDコネクタに外部キーボードを接続します。

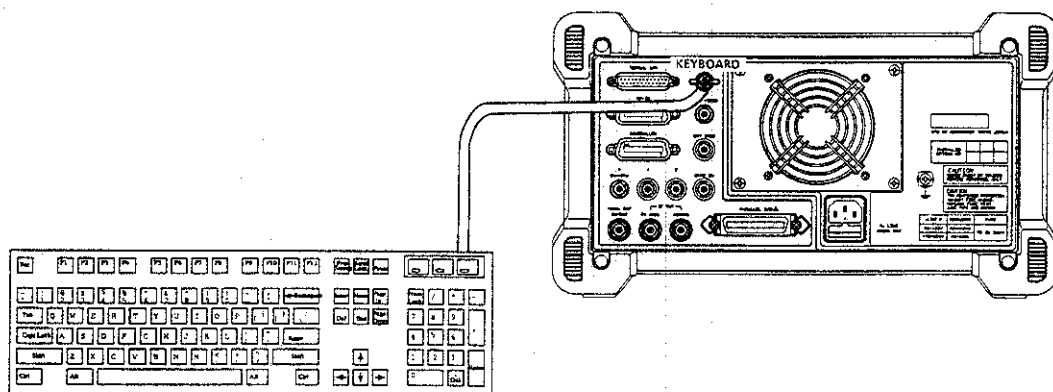


図1 外部キーボードの接続

その他にプログラムをセーブするメモリ・カード (標準装備32Kバイト)、 GPIBケーブルを用意します。

Q 2. コントローラの起動は、どうするのですか？

A 2. 計測器に外部キーボードを接続し電源を入れます。つぎに、**CTRL ON** キーを押すと以下に示す画面に変わり、コントローラ (エディタ) が起動します。

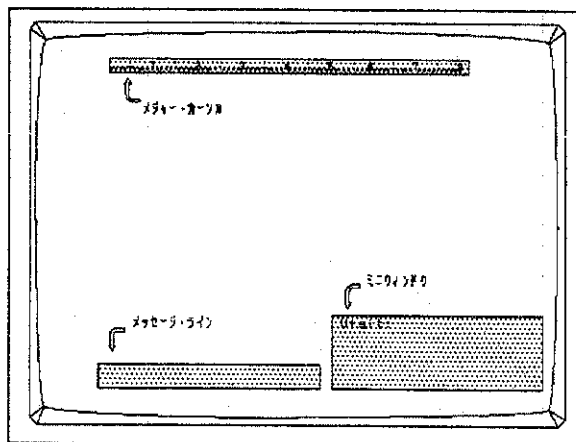


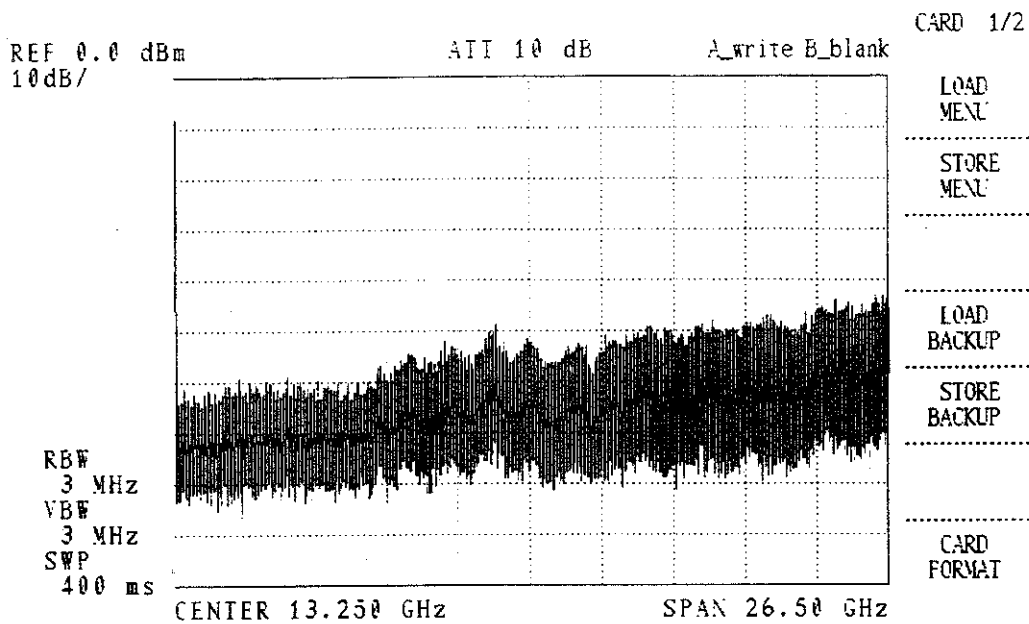
図2 コントローラ (エディタ) の動作画面

Q 3. メモリ・カードの初期化は、どうするのですか？

A 3. 計測器にメモリ・カードを挿入し、以下に示すように操作します。ただし、すでにデータがセーブされているカードを初期化すると、セーブされているすべてのデータが消去されてしまいますので注意して下さい。

メモリ・カードの初期化手順

① **SHIFT** **4** と押します。



② **CARD FORMAT** を押します。

③再確認をしてきますので、初期化を行うならばCONFIRM。行わなければCANCELを押して下さい。

## 2. プログラムの作成

Q4. プログラムの作成は、どうするのですか？

A 4. 計測器の電源を入れた時点でコントローラは起動していますので、キーボードから文字を入力します。

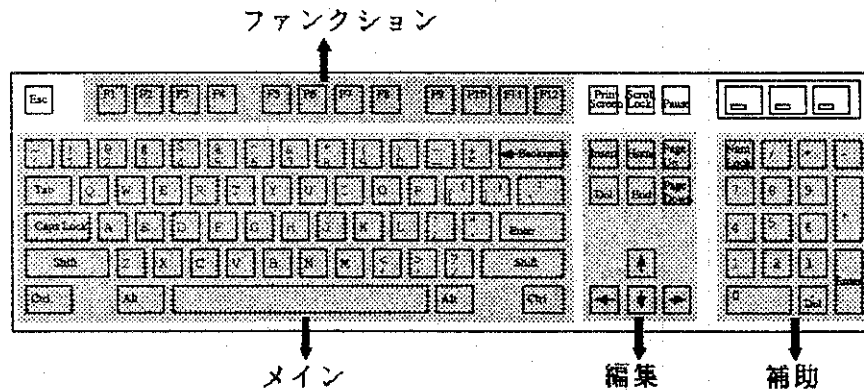


図3 外部キーボード

簡単なプログラム例を以下に示します。

```
TOTAL=0
FOR I=1 TO 100
  TOTAL=TOTAL+I
NEXT I
PRINT "1-100 TOTAL = ",TOTAL
STOP
END
```

図4 簡単なプログラム例

**Q 5. ポップアップメニューは、どう使うのですか？**

A 5. <Alt>キーまたは<Ctrl-A>キーを入力します。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。ここでカーソルキーにて実行するメニューを選択し、<Enter>キーを入力します。

ファンクションキーパッドの配置は、図6を参照して下さい。

BASIC	Region	Window/Other	File	Search/Replace
COMPILE & 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 ポップアップメニュー一覧

**Q 6. このエディタは、何行のプログラムが作成できますか？**

A 6. 1行40文字、行番号なしと考えて、約2500行のプログラムを作ることができます。

**Q7. 英小文字で作成したプログラムの実行結果が正しくありません。**

A 7. プログラム中の英小文字は変数と認識されますので、英小文字で書かれたプログラムはみなさんが期待する結果とは違う結果を出力します。  
ですから、プログラムを作るときはすべて英大文字を使うか、変数部のみ英小文字を使うようにします。

**Q8. 作成したプログラムをメモリ・カードにセーブするには、どうするのですか？**

A 8. まず、メモリ・カードを計測器に挿入します。次にポップアップメニューの"Write"または<Shift+F3>(shiftキーを押しながらF3キーを押す)を入力します。すると、画面右下にミニ・ウィンドウが表示されますので、ここでセーブするファイル名を入力します。  
正常にファイルをセーブしますと、書き込んだ行数を表示します。

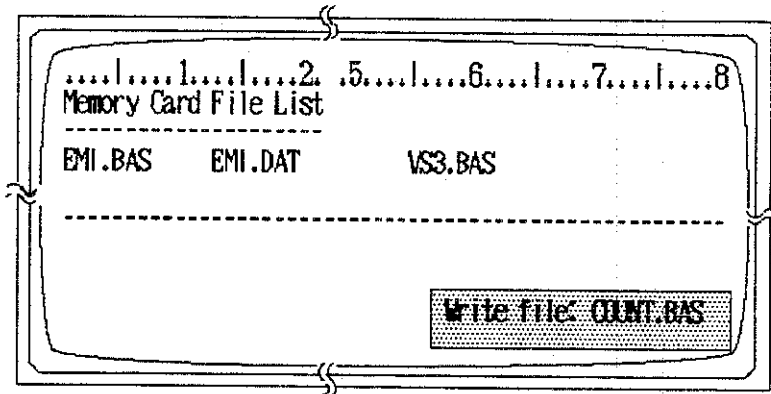


図6 ファイルのセーブ

Q9. ファイルがメモリ・カードにセーブできません。

A 9. メモリ・カードのWrite protectスイッチがONになっていませんか？

またはメモリ・カードの空き領域がなくなっていないですか？ メモリ・カードの空き領域はBASIC mode内でCATコマンドにて確認できます。

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
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

Total 25600 Bytes

Used 25476 Bytes (96 %)

図7 CATコマンドの実行

Q10. メモリ・カードからファイルをロードするには、どうするのですか？

A10. まず、メモリ・カードを計測器に挿入します。次にポップアップメニューの"Load"または<shift+F1>キーを入力します。すると、画面が上下に分割し、上画面にファイルの一覧表示、下画面にミニ・ウィンドウが表示されます。ここで、ロードするファイル名を入力します。

もし、該当するファイルが存在しないときはエラーメッセージが表示されます。

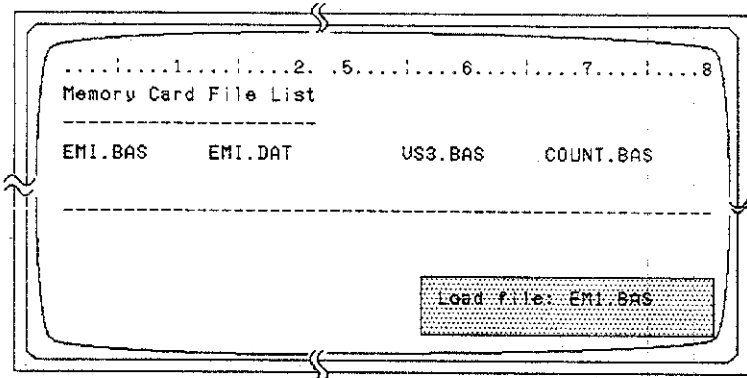



図8 ファイルのロード





### 3. プログラムの実行

Q11. プログラムの実行は、どうするのですか？

A11. ポップアップメニューの"COMPILE and RUN"または計測器の  または、  
ファンクションキーの<F1>キーを入力します。すると、プログラムが実行され、実行結果が  
画面に表示されます。

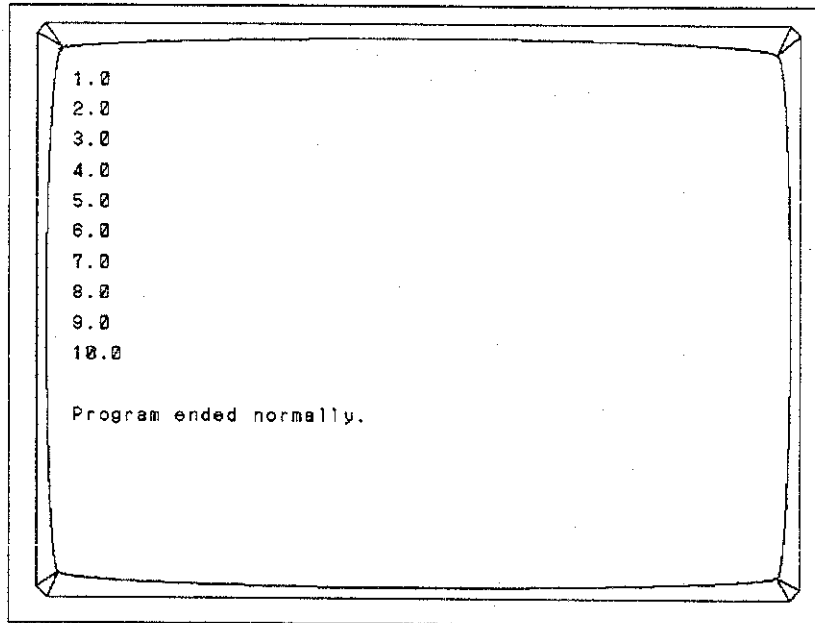


図9 プログラムの実行結果

Q12. 実行結果を計測器の画面に表示するには、どうするのですか？

A12. 以下に示す表示状態を設定し、PRINT、グラフィック命令によって画面に表示させます。

設定命令	表示状態	掃引状態	画面クリア命令
SCREEN 0	スペクトラム波形画面	測定中	
SCREEN 1	スペクトラム波形+BASICの合成画面	測定中	CLS
SCREEN 2	BASIC画面	測定中	CLS, CLS 1, CLS 2
SCREEN 3	BASIC画面	測定停止	CLS, CLS 1, CLS 2
SCREEN 4	スペクトラム波形+BASICの2画面	測定中	CLS

図10 表示状態の設定

(注) スペクトラム波形画面時に、CLS 1またはCLS 2のグラフィック画面クリア命令を実行するとスペクトラム波形画面の格子もクリアします。

これらの画面設定は"IP"コマンドにて初期化されるので、"IP"コマンド実行後は再設定して下さい。

なお、画面設定は、OUTPUT 31;"VS○"でも可能です。詳細は、第2部 システム・コントローラの3-2ページ (2) グラフィック機能の項を参照して下さい。

Q13. プログラムのソースリストをGPIBプリンタに印字するには、どうするのですか？

A13. まず、プリンタ側のGPIBポートと計測器側のGPIBポート（コントローラ側）を接続し、プリンタの電源を入れます。

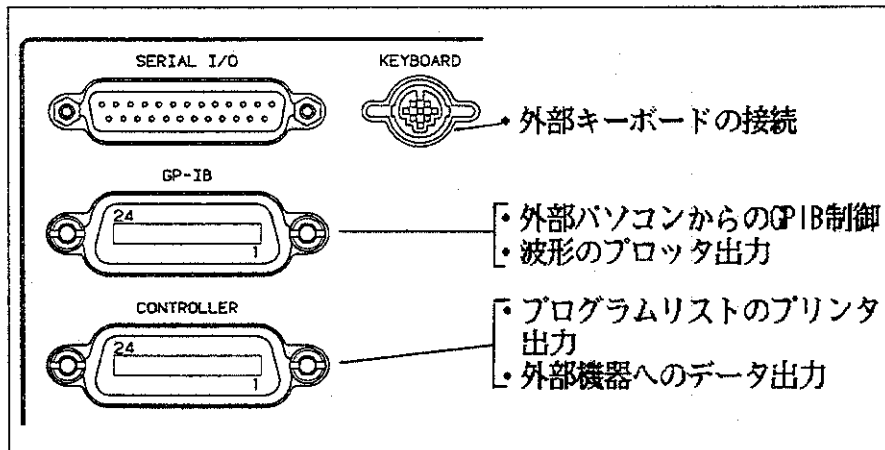


図11 背面パネルのGPIBポート

次にポップアップメニューの"BASIC mode"またはファンクションキーの<F2>を入力します。すると、画面右下にミニ・ウィンドウが表示されますので、ここで、以下に示すように入力します。

ただし、すでに"COMPILE & RUN"にて実行されていない限りはなりません。

入力方法	説明
PRINTER 3 GLIST	プリンタの指定 (3はGPIBアドレス) GPIBプリンタへソースリストを出力

図12 ソースリストのプリンタ出力

**Q14. 実行結果をGPIBプリンタに印字するには、どうするのですか？**

A14. まず、A16.と同様にGPIBケーブルでプリンタと計測器を接続します。次にBASIC mode内で以下に示すように入力します。

入力方法	説明
PRINTER 3 GPRINT	プリンタの指定 (3はGPIBアドレス) GPIBプリンタへ実行結果を出力

図13 実行結果のプリンタ出力

**Q15. メモリ・カードから複数のプログラムを自動的にロードして実行するには、どうするのですか？  
また、プログラムを内部メモリに常駐させて実行するには、どうするのですか？**

A15. 両方とも実行できません。  
ateエディタは、2つ以上のプログラムをロードしません。  
またBASICインタプリタも、ateエディタにロードしてあるプログラムのみ実行します。

#### 4 . 実戦テクニック

Q16. 計測部へのデータ設定は、どうするのですか？

A16. OUTPUTコマンドを使います。このときのアドレスは31で、これは外部のGPIBケーブルを介さず、内部で通信します。以下にサンプル・プログラムを示します。

```
OUTPUT 31;"IP"           ! プリセット
OUTPUT 31;"CF200MZ"      ! 中心周波数を200MHzにセット
OUTPUT 31;"SP100KZ"      ! 周波数スパンを100kHzにセット
WAIT 200                 ! 約200msのウエイト
OUTPUT 31;"PS"           ! ピーク・サーチ
```

図14 "OUTPUT 31"のサンプル・プログラム

Q17. 計測データを計測部から読み出すには、どうするのですか？

A17. ENTERコマンドを使います。このときのアドレスは31で、これは外部のGPIBケーブルを介さず、内部で通信します。また、読み出すデータをOUTPUTコマンドで指定しておきます。書式は、OUTPUT 31;"MF?"のようにGPIBコマンドの最後に'?'マークをつけます。以下にサンプル・プログラムを示します。

```
OUTPUT 31;"IP"           ! プリセット
OUTPUT 31;"CF200MZ"      ! 中心周波数を200MHzにセット
OUTPUT 31;"SP100KZ"      ! 周波数スパンを100kHzにセット
WAIT 200                 ! 約200msのウエイト
OUTPUT 31;"PS"           ! ピーク・サーチ
OUTPUT 31;"MF?"          ! ピーク周波数の読み出し
ENTER 31;MKRF            ! ピーク周波数の読み出し
PRINT "MARKER FREQ = ",MKRF !
OUTPUT 31;"ML?"          ! ピークレベルの読み出し
ENTER 31;MKRL            ! ピークレベルの読み出し
PRINT "MARKER LEVEL = ",MKRL !
```

図15 "ENTER 31"のサンプル・プログラム

Q18. 会話形式でプログラムを実行するには、どうするのですか？

A18. ON KEYまたはINPUTコマンドを使います。ON KEYコマンドは、1~7まで指定できます。INPUTコマンドは、<Enter>または<単位>キーを入力することにより確定します。数値キーの入力は、キーボードの有無に関係なくできますが、文字および負数の入力は、キーボードからのみでき、計測器のみではできません。以下にサンプル・プログラムを示します。

```
ON KEY 1 GOTO *KEY1      ! ON KEYコマンドの宣言
ON KEY 2 GOTO *KEY2
ON KEY 3 GOTO *KEY3
*L
  ENABLE INTR           ! 割り込み許可
  PRINT ">> key in !"
*LL
  GOTO *LL              ! 割り込み待ち
!
*KEY1
  DISABLE INTR: INPUT "----> key 1 OK? (Y=1/N=0)",I
                          ! 割り込み禁止およびINPUTコマンドの実
                          行
  IF I=0 THEN GOTO *KEY1
  GOTO *L                !
*KEY2
  DISABLE INTR: INPUT "----> key 2 OK? (Y=1/N=0)",I
  IF I=0 THEN GOTO *KEY2
  GOTO *L
*KEY3
  DISABLE INTR: INPUT "----> key 3 OK? (Y=1/N=0)",I
  IF I=0 THEN GOTO *KEY3
  GOTO *L
STOP
```

図16 実行時会話形式のサンプル・プログラム

Q19. 他の計測器を GPIB で制御するには、どうするのですか？

A19. OUTPUT コマンドを使います。このときのアドレスは 0-30 です。以下にサンプル・プログラムを示します。

```
XS=1500:YS=600          ! (1500,600)から(2000,600)まで実線を書く
OUTPUT 5;"PU;PA"&XS&","&YS&";"
                          ! GPIBアドレスは5
XS=2000:YS=600
OUTPUT 5;"PD;PA"&XS&","&YS&";"
```

図17 プロッタを制御するサンプル・プログラム

Q20. プログラム実行時のデータをメモリ・カードにセーブするには、どうするのですか？

A20. OPEN/CLOSE および OUTPUT コマンドを使います。ファイルがすでに存在する場合は、消去してから実行します。以下にサンプル・プログラムを示します。

詳細は、OPEN/CLOSE 命令の項を参照して下さい。

```
DIM A(100)
FILES="ABC"          ! ファイル名は"ABC"
INPUT "Is ¥"ABC¥" file new ? (Y=1/N=0) -> ",NEW
IF NEW=0 THEN PURGE FILES
OPEN FILES FOR OUTPUT AS #FD;ASCII
                          ! "ABC"ファイルをオープンする
FOR I=1 TO 100
  A(I)=I
  OUTPUT #FD;A(I)        ! データを書き込む
  PRINT "Save = ",A(I)
NEXT I
CLOSE #FD              ! "ABC"ファイルをクローズする
STOP
```

図18 変数をセーブするサンプル・プログラム

Q21. メモリ・カードにセーブしてあるデータをロードするには、どうするのですか？

A21. OPEN/CLOSEおよびENTERコマンドを使います。ロードするファイルはすでに存在しなければなりません。また、書き込んだフォーマットと同一でなければなりません。以下にサンプル・プログラムを示します。

詳細は、OPEN/CLOSE命令の項を参照して下さい。

```
DIM B(100)
FILES="ABC"           ! ファイル名は"ABC"
OPEN FILES FOR INPUT AS #FD;ASCII
                       ! "ABC"ファイルをオープンする
FOR I=1 TO 100
  ENTER #FD;B(I)       ! データを読み込む
  PRINT "Load = ",B(I)
NEXT I
CLOSE #FD              ! "ABC"ファイルをクローズする
STOP
```

図19 データをロードするサンプル・プログラム

Q22. トレース・データを変数に読み込むには、どうするのですか？

A22. GTA/GTBコマンドおよびビルトイン関数RTRACE( )を使います。GTA/GTBコマンドは、計測器内部の作業エリアにトレース・データを読み込むコマンドで、ビルトイン関数RTRACE( )を使う前に必ず実行しなければなりません。以下にサンプル・プログラムを示します。

```
INTEGER T1(701)       ! 格納エリアの宣言
!
OUTPUT 31;"GTA"       ! トレースAを作業エリアに読み込む
FOR I=1 TO 701
  T1(I)=RTRACE(I-1,0) ! 変数への読み込み
NEXT I
STOP
```

図20 トレース・データを読み込むサンプル・プログラム



また、トレース・データを書き込む場合は、PTA/PTBコマンドおよびビルトイン関数 WTRACE ( ) を使います。PTA/PTBコマンドは、ビルトイン関数 WTRACE ( ) によって作業エリアに書き込まれたデータを計測器内部のトレース・メモリへ転送するコマンドです。以下にサンプル・プログラムを示します。

```

INTEGER T1 (701)           ! 変数の宣言
!
FOR I=1 TO 701
  WTRACE(T1(I),I-1,1)     ! データを作業エリアに書き込む
NEXT I
OUTPUT 31;"PTB"           ! 作業エリアのデータを
                           !   トレースBに書き込む
OUTPUT 31;"BV"            ! B viewにする
STOP

```

図21 トレース・データを書き込むサンプル・プログラム

トレース A のデータをメモリ・カードにセーブし、そのデータをトレース B に書き込むサンプル・プログラムを、以下に示します。参考にして下さい。

```

! trace A                 -> Memory card (FILES)
! Memory card (FILES) -> trace B
INTEGER T1 (701), T2 (701)
!
GOSUB *SETUP              ! セットアップ
GOSUB *FSAVE              ! メモリ・カードへセーブ
GOSUB *FLOAD              ! メモリ・カードからロード
STOP
!
*SETUP
  OUTPUT 31;"VS2":CLS
  PRINT "##### OPEN/CLODE #####"
  BUZZER 1000,500
  CURSOR 5,5:PRINT ">> Please, write protect switch is off !"
  CURSOR 5,7:INPUT ">> Save file name = ?",FILES
                           ! ファイル名の入力
  CURSOR 5,9:INPUT ">> New file ? (Y=1/N=0) ",NEW
  IF NEW=0 THEN PURGE FILES
                           ! 既存のファイルは消去
  OUTPUT 31;"IP VS1":CLS
  OUTPUT 31;"CLN CF30MZ SP1MZ RE-10DB RB100KZ"
  RETURN
!
*FSAVE
  OUTPUT 31;"VS1":CLS
  BUZZER 500,500
  PRINT "trace A -> Card saving .. (file) = ",FILES
  OPEN FILES FOR OUTPUT AS #FD;ASCII
                           ! ファイルのオープン

```

図22 トレースデータとメモリ・カードとの読み書きのサンプル・プログラム (1/2)

```

OUTPUT 31;"GTA"          !   トレース A の読み込み
FOR I=1 TO 701
  T1(I)=RTRACE(I-1,0)    !   変数に格納
  OUTPUT #FD;T1(I)       !   メモリ・カードへセーブ
NEXT I
CLOSE #FD                !   ファイルのクローズ
OUTPUT 31;"AB"          !   A blank
CLS:BUZZER 500,500
RETURN
!
*FLOAD
OUTPUT 31;"VS1":CLS
BUZZER 500,500
OUTPUT 31;"CWB BV"      !   B claer & view
CLS:PRINT "Card loading .. -> trace B (file) = ",FILES
OPEN FILE$ FOR INPUT AS #FD:ASCII
                          !   ファイルのオープン

FOR I=1 TO 701
  ENTER #FD;T2(I)        !   メモリ・カードからロード
  WTRACE(T2(I),I-1,1)    !   作業エリアへ書き込み
NEXT I
OUTPUT 31;"PTB"         !   トレース B へ書き込み
OUTPUT 31;"BV"          !   B view
CLOSE #FD               !   ファイルのクローズ
CLS:BUZZER 500,500
RETURN

```

図22 トレースデータとメモリ・カードとの読み書きのサンプル・プログラム (2/2)

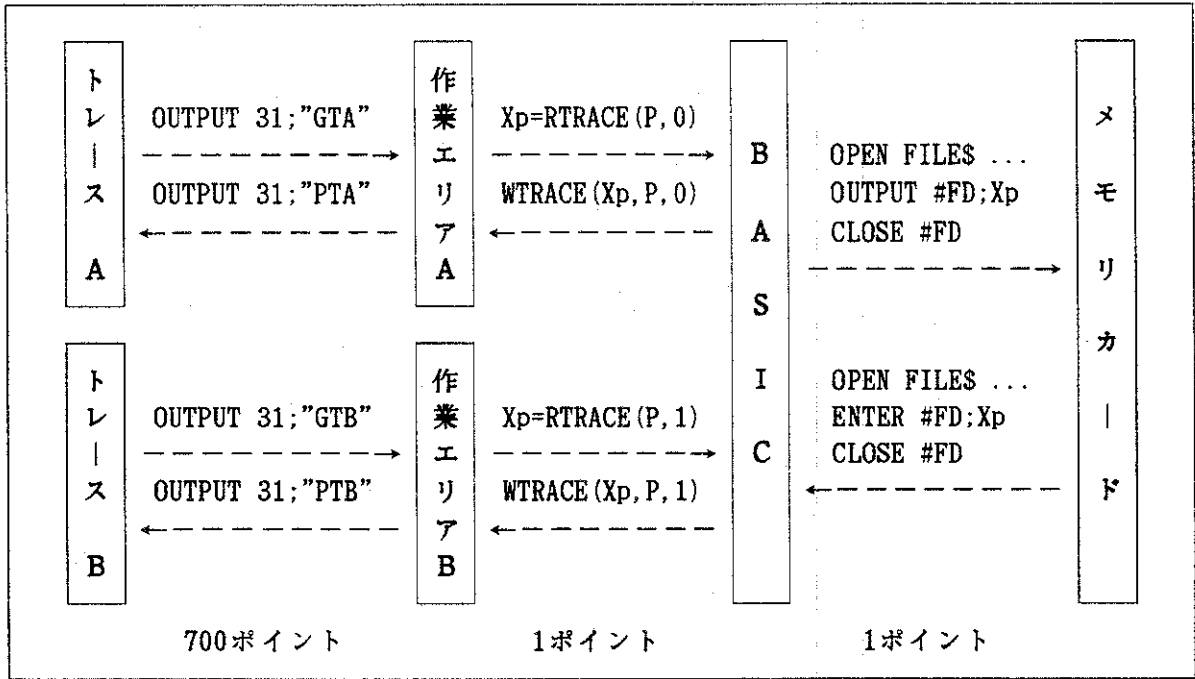


図23 トレースとメモリ・カードとのデータ転送の関係

Q23. サービス・リクエスト (SRQ) は、どのように使うのですか？

A23. SPOLLおよびビット演算子を使います。このとき、割り込みを使う方法もあります。以下に割り込みを使ったサンプル・プログラムを示します。

```
INTEGER S
SPA=31
ON ISRQ GOSUB *SRQCHK      ! 割り込み宣言
OUTPUT SPA;"IP"
OUTPUT SPA;"VS1 S0"
!----- Calibration -----!
DATA "CLG","IT0","IT1","IT2","IT3","IT4"
FUNCS="CAL"
RESTORE
FOR I=0 TO 5
  READ CALS
  OUTPUT SPA;CALS
  GOSUB *SWAIT
  WAIT 1000
NEXT I
!----- Sweep end -----!
FUNCS="SWP"
OUTPUT SPA;"IP CLN CF30MZ SP500KZ SW2SC SI"
FOR I=0 TO 5
  OUTPUT SPA;"SP DN SR"
  GOSUB *SWAIT
```

図24 SRQを使ったサンプル・プログラム (1/2)

```

NEXT I
STOP
!
!----- SRQ interrupt wait -----!
*SWAIT
  FLAG=0
  ENABLE INTR          ! 割り込み許可
*LL
  IF FLAG=1 THEN RETURN ! 割り込み待ち
  GOTO *LL
!----- SRQ status check -----!
*SRQCHK
  DISABLE INTR        ! 割り込み禁止
  S=SPOLL(SPA)        ! ステータスのポーリング
  IF FUNC$="CAL" AND (S BAND 2) <> 0 THEN
    GOSUB *CALEND: RETURN
  END IF
  IF FUNC$="SWP" AND (S BAND 4) <> 0 THEN
    GOSUB *SWPEND: RETURN
  END IF
  ENABLE INTR          ! 割り込み許可
  RETURN
!
!----- Calibration end -----!
*CALEND
  BUZZER 500,50
  PRINT "CAL. end -->", CAL$
  FLAG=1
  RETURN
!
!----- Sweep end -----!
*SWPEND
  BUZZER 500,50
  FLAG=1
  FREQ=BND(PMAX(0,700,0),10,0)
  PRINT "10 dB down band = ", FREQ/1000, "kHz"
  RETURN

```

図24 SRQを使ったサンプル・プログラム (2/2)

Q24. パソコンで作成したプログラムを、ateエディタに転送するには、どうするのですか？

A24. まず、転送するプログラム内で行番号を使って分岐しているもの（例：GOTO 100など）をラベルに変更し、ファイルにASCIIタイプでセーブします。  
次に制御プログラムを作成します。制御プログラムとは、転送するプログラムを以下の手順で、計測器に転送するものです。

計測器をダウンロードモードにする。( GPIBコード "LOAD START" )  
転送するプログラムを1行読み込む。  
行番号を削除する。  
計測器に1行、転送する。  
計測器を通常モードにする。( GPIBコード "LOAD END" )

注) ダウンロードモードを使用した場合、必ず通常モードに戻してください。

外部パソコンから転送する準備ができれば、計測器側の準備をします。  
まず、計測器の背面パネルにある GPIBポートと外部パソコンの GPIBポートを接続します。

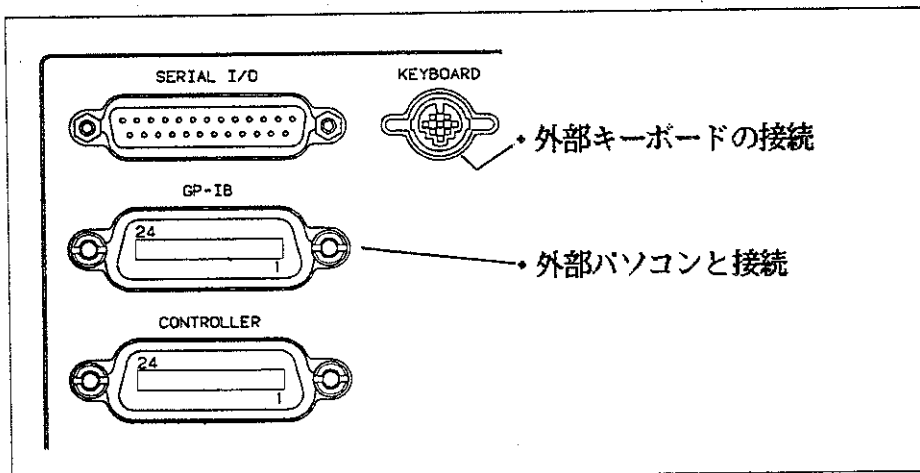


図25 背面パネルの GPIBポート

次に計測器の電源を入れ、コントローラを起動させてから外部パソコン上で作成した制御プログラムを実行します。これで外部パソコンから計測器へのプログラム転送が始まります。

なお、転送する前にポップアップメニューの "SCRATCH" を実行し、エディタを初期化して下さい。

最後に制御プログラムのサンプルを用意しましたので参考にして下さい。

PC版制御プログラム

```

10 DIM FILES(10)
20 DIM PROGS(128)
30 '
40 SPA=8 ! GPIBアドレス
50 ISET IFC
60 ISET REN
70 CMD DELIM=0
80 '
90 INPUT "Down load file name = ? ";FILES
! ファイル名の入力
100 OPEN "B:"+FILES+".BAS" FOR INPUT AS #1
! ファイルをオープン
110 WHILE EOF(1)=0
120 LINE INPUT #1,PROGS ! 1行ロード
130 LL=LEN(PROGS) ! 行番号の削除 (~140行まで)
140 PROGS=MID$(PROGS,6,LL-5) !
150 PROGS="@"+PROGS ! 行の先頭に '@' を付加する
160 PRINT PROGS
170 PRINT @SPA;PROGS ! 1行転送
180 FOR I=0 TO 3000: NEXT I ! 待ち時間をとる
190 WEND
200 CLOSE ! ファイルをクローズ
210 PRINT " Down load end !! "
220 END

```









R3265A/71A シリーズ

コントローラ機能

レファレンス



## 緒言

### 1. R3265A/3271Aシリーズ説明書の種類

R3365A/3371Aとして使用するときは、R3265A/3271AをR3365A/3371Aと読みかえて下さい。

種 類	内 容	備 考
1. R3265A/3271Aシリーズ スペクトラム・アナライザ 取扱説明書	R3265A/3271Aの取扱方法を説明しています。  <ul style="list-style-type: none"> <li>・付属品</li> <li>・パネル面の説明</li> <li>・機能説明</li> <li>・操作説明</li> <li>・性能諸元 など</li> </ul>	標準付属品です。
2. R3265A/3271Aシリーズ クイック・ガイド	R3265A/3271Aのキー操作を基本から応用まで具体例で示しています。  <ul style="list-style-type: none"> <li>・スタート/ストップ周波数の設定手順</li> <li>・測定ウィンドウ内での掃引手順</li> <li>・データ・ストア手順</li> <li>・ソフト・メニューの初期化手順</li> <li>・占有周波数帯域幅測定手順 など</li> </ul>	標準付属品です。
3. R3265A/3271Aシリーズ コントローラ機能 取扱説明書	コントローラ機能の取扱方法を説明しています。  <ul style="list-style-type: none"> <li>・ガイド</li> <li>・レファレンス</li> </ul>	標準付属品です。
4. R3265/3271シリーズ GPIBコマンド拡張機能 取扱説明書	GPIBコマンド 拡張機能の取扱方法を説明しています。  <ul style="list-style-type: none"> <li>・第1部：GPIB概要</li> <li>・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</li> <li>・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</li> </ul>	別売です。

2. 本書は、R3265A/3271Aシリーズ用コントローラ機能を詳しく説明しています。

関連マニュアル

・R3265A/3271Aシリーズ コントローラ機能 (ガイド)

コントローラ機能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. R3265A/3271Aシリーズ コントローラ機能に接続可能な外部キーボードは、101型キーボード (コネクタ形状 ミニDIN6ピン) 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
パソコン	: パーソナル・コンピュータを表わします。
外部キーボード	: 101型キーボード

5. 本書は、3部構成になっています。

第 1 部	
1.	概要
2. ~5.	ateエディタの説明

第 2 部	
1.	コントローラ機能 (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

本コントローラ機能は、スペクトラム・アナライザR3265A/3271A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 (R3265A/3271A) 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。

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 a t e エディタ

- |    |      |
|----|------|
| 1章 | 概要   |
| 2章 | 特徴   |
| 3章 | 起動   |
| 4章 | 機能一覧 |
| 5章 | 詳細解説 |





## 第 1 部の目次

1. 概要	1 - 1
2. 特徴	2 - 1
2.1 キーボード	2 - 1
2.2 ポップアップメニュー	2 - 1
2.3 BASIC実行環境	2 - 2
2.4 ラベルの定義	2 - 2
2.5 ファイル	2 - 3
2.6 ヘルプメニュー	2 - 3
2.7 ユーザ定義メニュー	2 - 4
3. 起動	3 - 1
4. 機能一覧	4 - 1
4.1 カーソル制御	4 - 1
4.2 挿入	4 - 1
4.3 削除	4 - 1
4.4 コピー、移動、削除 (範囲指定)	4 - 2
4.5 ウィンドウ	4 - 2
4.6 ファイル	4 - 2
4.7 検索	4 - 2
4.8 置換	4 - 2
4.9 BASICモード	4 - 2
4.10 ヘルプ	4 - 3
4.11 ポップアップメニュー	4 - 3
4.12 キャンセル	4 - 3
4.13 エディタの再起動	4 - 3
4.14 コントローラの起動	4 - 3
5. 機能解説	5 - 1
5.1 カーソル制御	5 - 1
5.2 挿入	5 - 2
5.3 削除	5 - 3
5.3.1 一文字削除	5 - 3
5.3.2 一行削除	5 - 5
5.3.3 範囲指定削除	5 - 5
5.4 コピー、移動、削除 (範囲指定)	5 - 6
5.4.1 テキストの削除	5 - 6
5.4.2 テキストの復旧、移動	5 - 7
5.4.3 テキストのコピー	5 - 7
5.5 ウィンドウ	5 - 9
5.5.1 ウィンドウの分割	5 - 9
5.5.2 ウィンドウの初期化	5 - 9
5.5.3 画面の再表示	5 - 10
5.6 ファイル	5 - 11
5.6.1 ファイルのセーブ	5 - 11
5.6.2 ファイルのロード	5 - 12
5.6.3 ファイルの更新	5 - 13
5.6.4 プログラム・オート・スタート機能	5 - 14

5.7	検索	5 - 15
5.8	置換	5 - 16
5.9	BASIC	5 - 18
5.9.1	行番号の設定	5 - 18
5.9.2	BASICの実行	5 - 20
5.9.3	BASICの停止	5 - 20
5.9.4	BASICモード	5 - 21
5.9.5	BASICの継続	5 - 21
5.10	ヘルプ	5 - 21
5.11	ポップアップメニュー	5 - 22
5.12	ポップアップメニューの中止	5 - 25
5.13	エディタの初期化	5 - 26

図1-1	R3265A/3271Aと外部キーボードの接続	1- 1
図2-1	外部キーボード	2- 1
図2-2	ポップアップメニュー	2- 1
図2-3	BASICモード	2- 2
図2-4	ラベルの定義	2- 2
図2-5	ファイルのロード	2- 3
図2-6	ヘルプメニュー	2- 3
図3-1	ateの動作画面	3- 1
図3-2	ミニウィンドウ	3- 2
図4-1	キーパッド配置図	4- 1
図5-1	カーソルキー	5- 1
図5-2	カーソル移動に関するキーパッド	5- 1
図5-3	文字の継続を表す	5- 2
図5-4	挿入に関するキーパッド	5- 2
図5-5	カーソル直前の文字を削除	5- 3
図5-6	カーソル位置の文字を削除	5- 4
図5-7	カーソルから行末までを削除	5- 5
図5-8	削除に関するキーパッド	5- 5
図5-9	マークセット	5- 6
図5-10	テキストの保存(削除)	5- 6
図5-11	テキストの復旧	5- 7
図5-12	テキストのコピー	5- 7
図5-13	リージョン処理に関するキーパッド	5- 8
図5-14	上下に分割されたウィンドウ	5- 9
図5-15	ウィンドウを1つにする	5- 9
図5-16	ウィンドウに関するキーパッド	5-10
図5-17	テキストのセーブ(ファイル名を入力)	5-11
図5-18	ファイル名の入力	5-12
図5-19	ファイルのロード	5-12
図5-20	テキストの更新(同一ファイル名)	5-13
図5-21	ファイルに関するキーパッド	5-13
図5-22	カーソル位置以降の文字列検索	5-15
図5-23	実行結果	5-15
図5-24	検索に関するキーパッド	5-15
図5-25	置換対象文字列の入力	5-16
図5-26	置換文字列の入力	5-16
図5-27	実行結果	5-17
図5-28	置換に関するキーパッド	5-17
図5-29	ラベルを用いたプログラム	5-18
図5-30	開始番号の指定	5-18
図5-31	行番号間隔の指定	5-19
図5-32	自動行番号機能	5-19
図5-33	エディタとインタプリタとの関係	5-20

☒5-34	BASICモード .....	5-21
☒5-35	BASICに関するキーパッド .....	5-21
☒5-36	ヘルプメニュー .....	5-22
☒5-37	ヘルプに関するキーパッド .....	5-22
☒5-38	ポップアップメニュー .....	5-23
☒5-39	ポップアップメニュー一覧 .....	5-23
☒5-40	ポップアップメニューに関するキーパッド .....	5-24
☒5-41	ポップアップメニューの説明 .....	5-24
☒5-42	ファンクション・キーの割り当て .....	5-25
☒5-43	キャンセルに関するキーパッド .....	5-26
☒5-44	エディタの終了ミニウインド .....	5-27
☒5-45	エディタの初期化 .....	5-27
☒5-46	エディタの初期化に関するキーパッド .....	5-27

## 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KEYBOARDコネクタに接続した外部キーボードを用いて編集でき、操作性が大変向上しました。

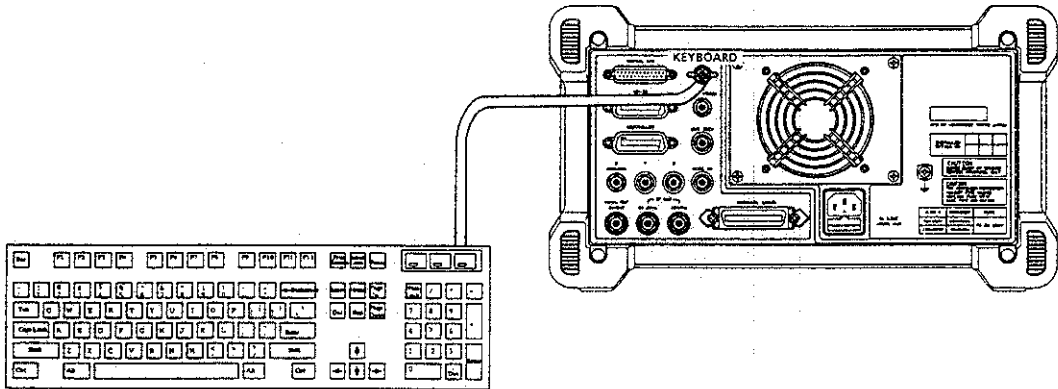


図1-1 R3265A/3271Aと外部キーボードの接続



## 2. 特長

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外部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。

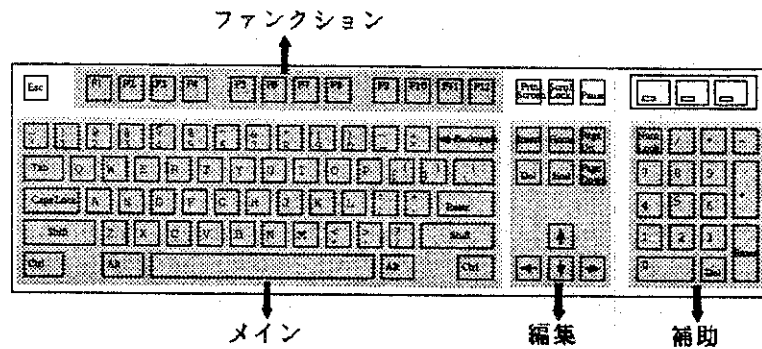


図2-1 外部キーボード

### 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、<Enter>キーを押すと、その機能が実行されます。

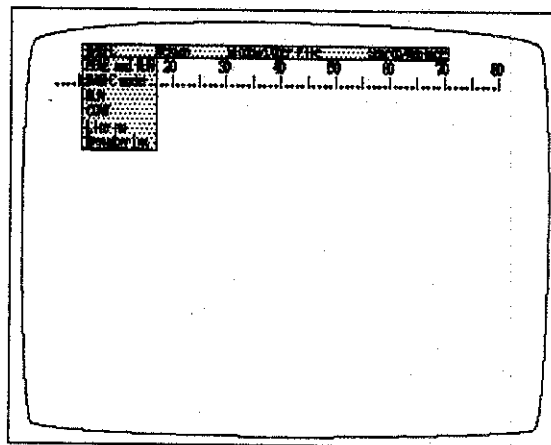


図2-2 ポップアップメニュー

### 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。

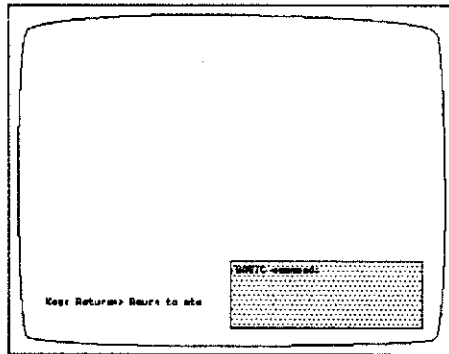


図2-3 BASICモード

### 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 1
  A = A + 1
  IF A >= 10 THEN
    GOTO * ABCD
  END IF
  GOTO * LABEL 1
* 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。

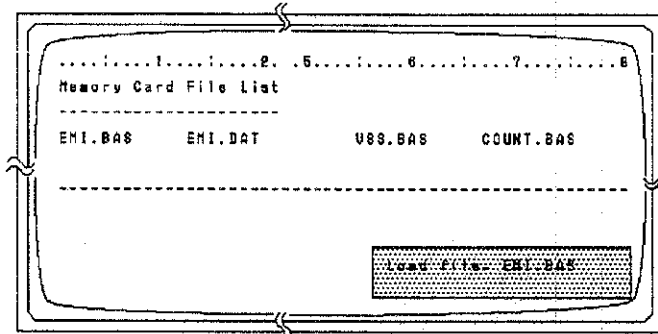


図2-5 ファイルのロード

## 2.6 ヘルプメニュー

外部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。

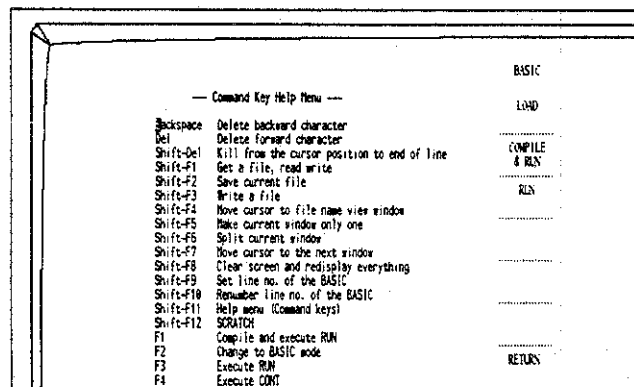


図2-6 ヘルプメニュー

## 2.7 ユーザ定義メニュー (ON KEY x LABEL)

ON KEY命令により、メニュー内の文字を任意に指定することができます。

ユーザ定義メニューは、メニュー上からON KEY 1~7 に対応してメニューを定義します。定義したメニューは、KEY ON 命令にて表示します。

ユーザ定義メニューからの割り込みは、ENABLE INTR/DISABLE INTR命令で許可/禁止を設定します。

PANEL 0 命令実行時にユーザ定義メニューが切り替わることがありますが、 キー  
を押すことによりユーザ定義メニューを表示します。 CNTRLR  
ON

例)

```
ON KEY 1 LABEL " ABC" GOTO *ABC  
ON KEY 3 LABEL " DEF" GOTO *DEF  
ON KEY 7 LABEL " XYZ" GOTO *XYZ  
KEY ON
```

ユーザ定義メニューを以下のように表示します。

ABC

-----

-----

DEF

-----

-----

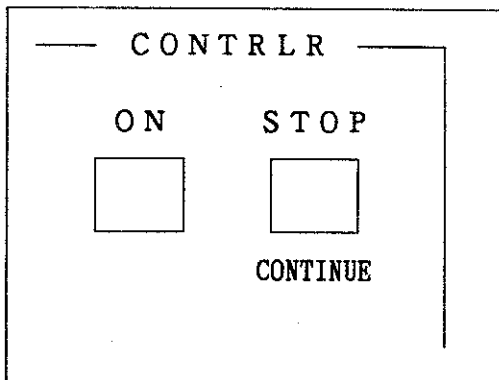
-----

-----

XYZ

### 3. 起動

#### (1) 操作



本体パネル右上部

ON



コントローラのBASICエディタをオープンします。  
また、BASIC実行中にはユーザ定義メニューを表示します。

STOP



BASICの実行を停止、または再実行させます。

CONTINUE

#### (2) BASICメニュー

LOAD

: ロードするBASICプログラムを選択します。  
(ロータリ・エンコーダにより選択し、単位キーにてロードを開始します。)

-----  
COMPILE  
& RUN

: BASICプログラムをコンパイルし終了後、実行します。

-----  
RUN

: バックアップ・メモリにあるコンパイル後のプログラムを実行します。このメニューが反転表示している場合はバックアップ・メモリにプログラムが存在することを示します。

-----  
AUTO EXEC  
CARD/MEM

: オートスタートの選択を行います。  
CARD : メモリ・カードからのオートスタート  
MEM : バックアップ・メモリからのオートスタート

-----  
ABORT

: LOADメニュー時のABORTを行います。

-----  
RETURN

: スペクトラム画面に戻ります。



#### 4. 機能一覧

ateエディタの各機能は一部を除いてキーボード上に割り当てられています。

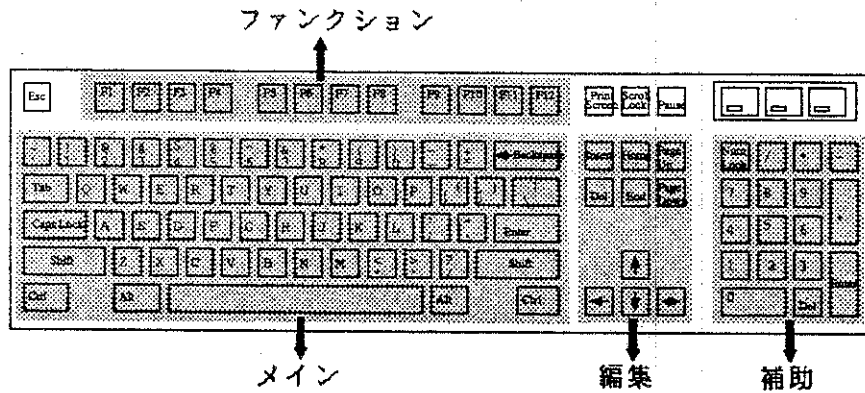


図4-1 キーボード配置図

##### 4. 1 カーソル制御

→	次の文字に移動
←	前の文字に移動
↑	前の行に移動
↓	次の行に移動
Home または Shift+↑	ファイルの先頭に移動
End または Shift+↓	ファイルの末尾に移動
Shift+←	行の先頭に移動
Shift+→	行の末尾に移動
Page Up	前に一画面分移動
Page Down	次に一画面分移動

##### 4. 2 挿入

Enter	改行文字を挿入
Tab	タブを挿入
上記以外のメインキーボード	一般文字

##### 4. 3 削除

Backspace	カーソル前の文字を削除
Delete	カーソル位置の文字を削除
Shift+Delete	カーソル位置から行末までを削除

#### 4. 4 コピー、移動、削除（範囲指定）

F5	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
F6	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
F7	マークからカーソル位置までをバッファに入れる
F8	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

#### 4. 5 ウィンドウ

Shift+F5	ウィンドウを1つにする
Shift+F6	ウィンドウを2つに分割する
Shift+F7	カーソルを次のウィンドウに移動
Shift+F8	画面をクリアして再表示する

#### 4. 6 ファイル

Shift+F1	メモ리카ードから指定したファイルをロードする
Shift+F2	メモ리카ードにファイルをセーブする
Shift+F3	メモ리카ードに指定したファイル名でセーブする
Shift+F4	ファイル名一覧ウィンドウへカーソルを移動する

#### 4. 7 検索

F9	前方に文字列を検索
F10	後方に文字列を検索

#### 4. 8 置換

F11	文字列を置き換え
-----	----------

#### 4. 9 BASICモード

Shift+F9	自動的な行番号挿入を設定
Shift+F10	行番号の再定義
F1	プログラムをBASICバッファに転送後、実行する
F2	BASICモードへ移行する
F3	すでに転送済みのプログラムを実行する
F4	Ctrl-Cで中断したプログラムを続行する

#### 4. 10 ヘルプ

Shift+F11	エディタ機能とキー割当の一覧
-----------	----------------

#### 4. 11 ポップアップメニュー

Alt(ctrl-A)	ポップアップメニューを表示する
-------------	-----------------

#### 4. 12 キャンセル

ESC(ctrl-Z)	エディタの機能を中止する
-------------	--------------

#### 4.13 エディタの再起動

Shift+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

#### 4.14 コントローラの起動

F12

コントローラを起動し、エディタ画面を表示する





## 5. 機能解説

### 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。

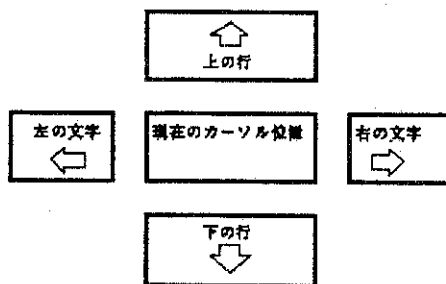


図5-1 カーソルキー

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時、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。

この他に以下のような機能があります。

Home または Shift+↑	ファイルの先頭に移動する
End または Shift+↓	ファイルの末尾に移動する
Shift+←	行の先頭に移動する
Shift+→	行の末尾に移動する
Page Up	次のページに進む
Page Down	前のページに戻る

図5-2 カーソル移動に関するキーボード

## 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。

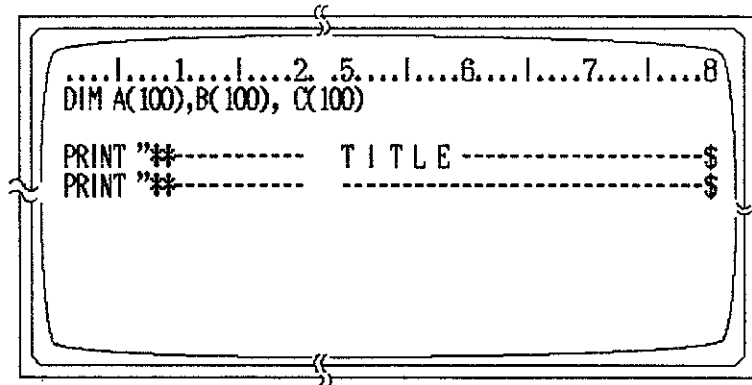


図5-3 文字の継続を表す

行の終わりは<Enter>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。

Enter	改行文字を挿入する
Tab	タブを挿入する
上記以外のキーパッド	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、<Backspace>キーを入力します。

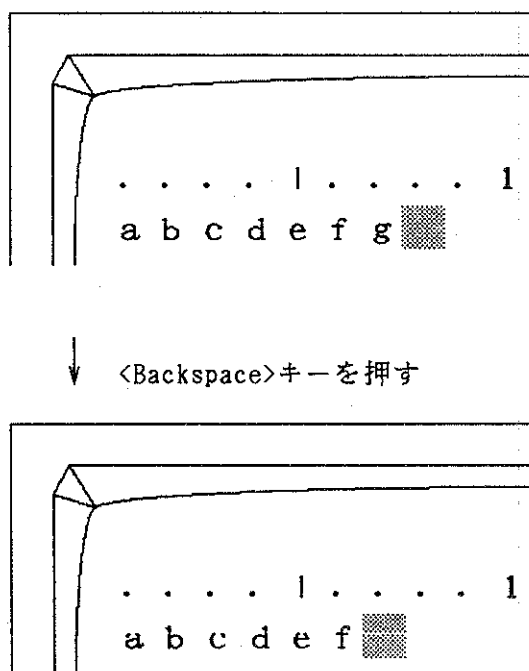


図5-5 カーソル直前の文字を削除

- (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、<Delete>キーを入力します。

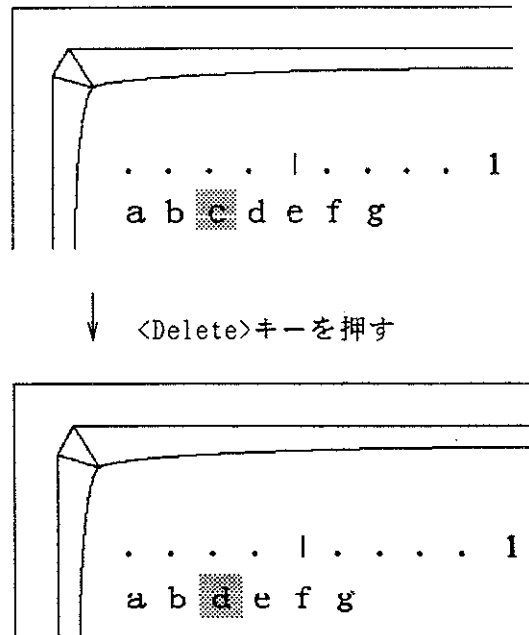


図5-6 カーソル位置の文字を削除

### 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、`<Shift+Delete>`キーを入力します。最初の`<Shift+Delete>`キーでその行の内容が削除され、再度`<Shift+Delete>`キーを入力するとその行自身が削除されます。

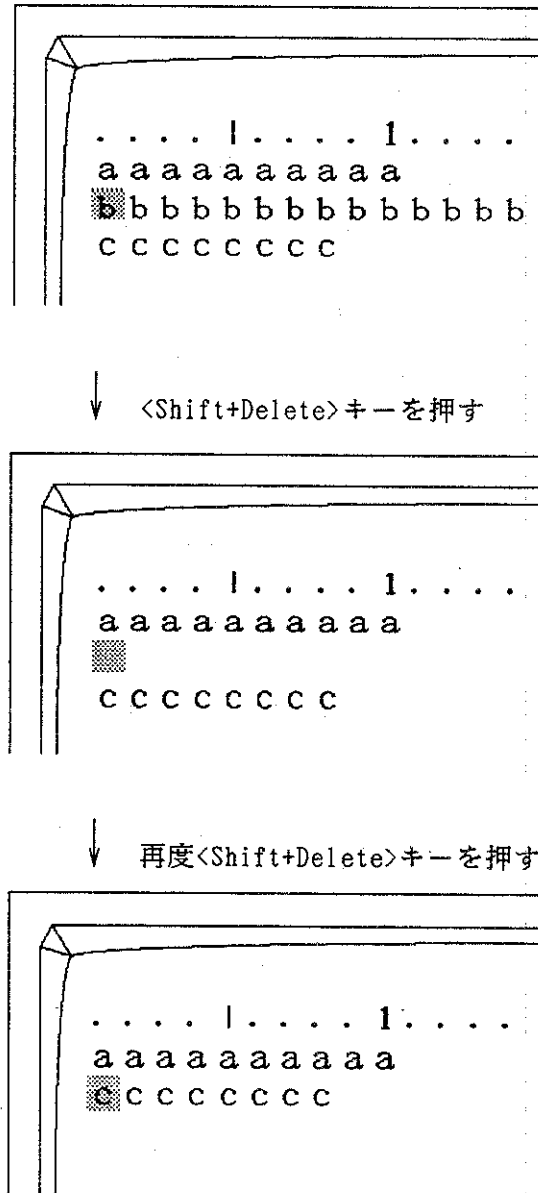


図5-7 カーソルから行末までを削除

### 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。

Backspace	カーソル直前の文字を削除
Delete	カーソル位置の文字を削除
Shift+Delete	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、〈F5〉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。

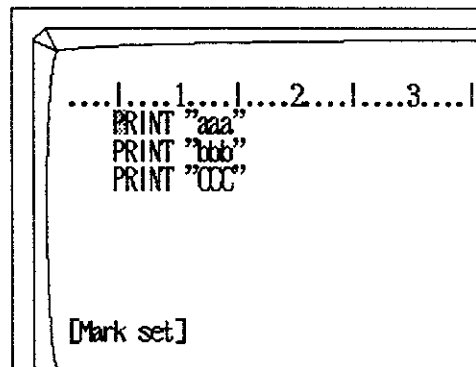


図5-9 マークセット

#### (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、〈F6〉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。

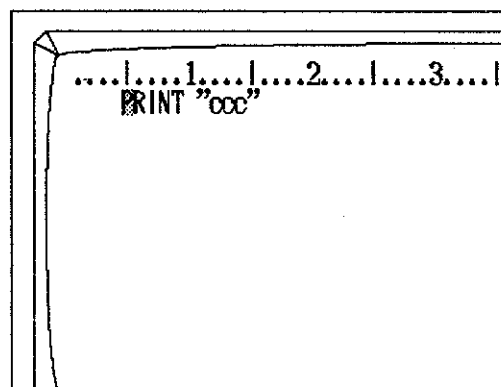


図5-10 テキストの保存（削除）

#### 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<F8>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、<F8>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<F8>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。

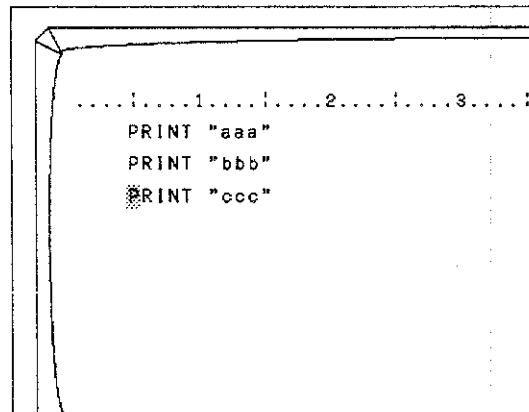


図5-11 テキストの復旧

#### 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、<F7>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、<F8>キーまたはポップアップメニューの"Yank"を入力します。

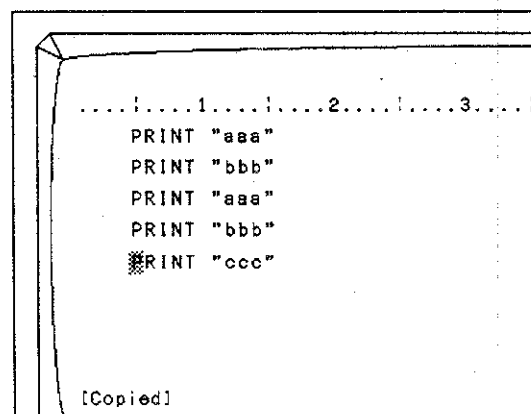


図5-12 テキストのコピー

<F8>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。

F5	範囲指定に必要なマークをカーソル位置にセットする
F6	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
F7	マークからカーソル位置までのテキストを保存する
F8	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、`<Shift+F6>`キーまたはポップアップメニューの"`Split`"を入力します。

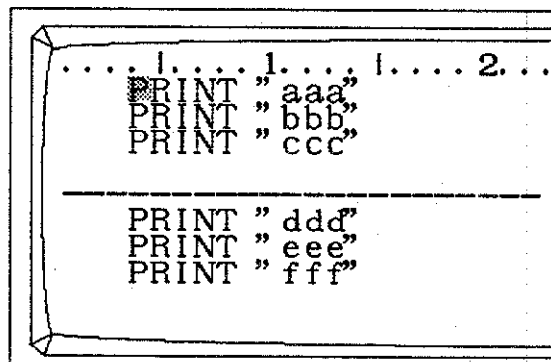


図5-14 上下に分割されたウィンドウ

### 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、`<Shift+F7>`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、`<Shift+F5>`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。

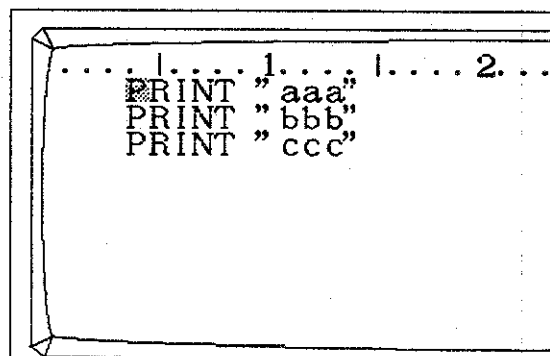


図5-15 ウィンドウを1つにする

### 5. 5. 3 画面の再表示

編集集中のテキストを再表示したいときは、〈Shift+F8〉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。

Shift+F5	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
Shift+F6	ウィンドウを上下に分割し、カーソルを上ウィンドウに置く
Shift+F7	カーソルを次のウィンドウに移動する
Shift+F8	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、~~Shift+F3~~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。

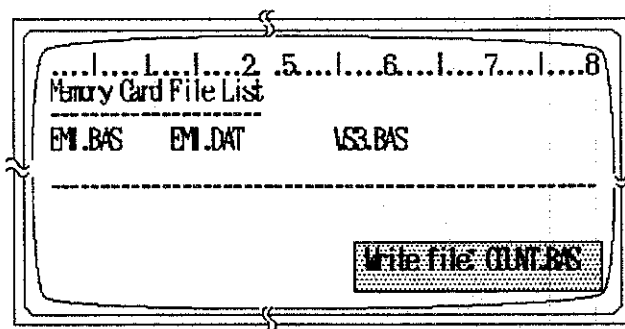


図5-17 テキストのセーブ (ファイル名を入力)

## 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、**<Shift+F1>**キーまたは計測器の **CNTRL ON LOAD** 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 **<Shift+F4>** 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 **<Enter>** キーを入力します。

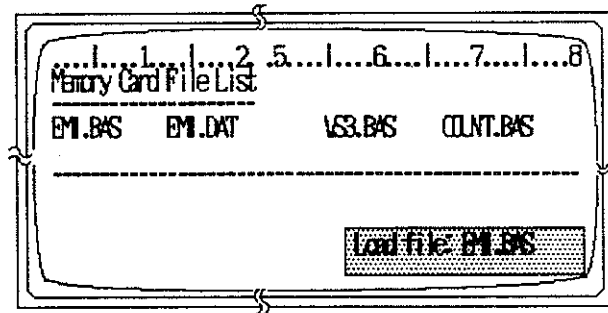


図5-18 ファイル名の入力

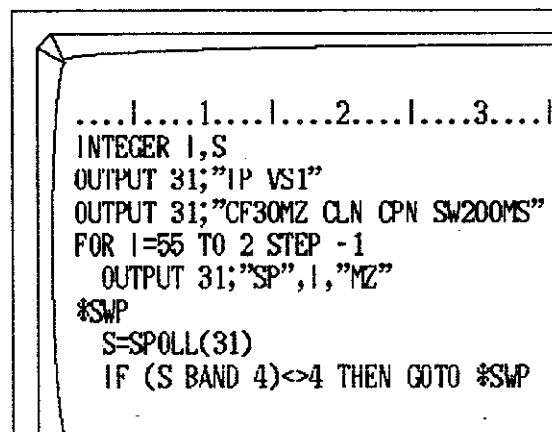


図5-19 ファイルのロード

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、<Shift+F2>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。

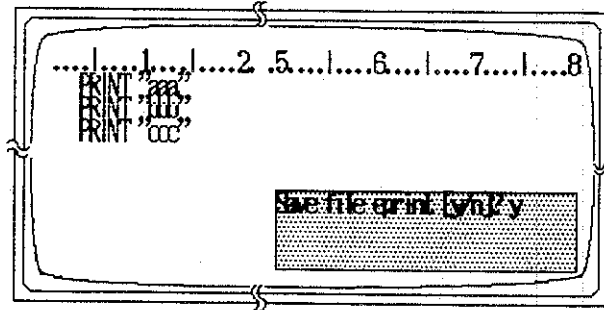


図5-20 テキストの更新（同一ファイル名）

<Y> <Enter>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> <Enter>と入力すると、テキストのセーブは実行しません。

Shift+F1	メモリ・カードから指定したファイルをロードする
Shift+F2	メモリ・カードに同一ファイル名でセーブする
Shift+F3	メモリ・カードに指定したファイル名でセーブする
Shift+F4	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

#### 5.6.4 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。

##### (1) レジューム機能

BASICプログラムの実行を行うと、コンパイル後のプログラムはバックアップ・メモリに蓄えられます。プログラム実行終了後、電源を切断してもコンパイル後のプログラムはメモリ内に保持され再度、電源を投入しても即座にプログラムの実行が可能になっています。

##### BASICメニュー

```
-----  
  RUN  
-----
```

反転表示しているとき、バックアップ・メモリにコンパイル後のプログラムが存在していることを示します。

注) BASICプログラムの保存は行いません(コンパイル後のプログラムのみ保存します)。

プログラム実行中に電源を切断すると、電源投入後にプログラムの実行ができません。プログラムの停止を確認してから電源を切断して下さい。

##### (2) オートスタート機能

電源投入後プログラムを自動的に実行します。実行するプログラムは、メモリ・カードまたはバックアップ・メモリにあるプログラムを選択して使用します。

選択方法は以下の通りです。

BASICメニューの AUTO EXEC で設定します。  
CARD/MEM

CARD : メモリ・カード内の「AUTOSTART」のファイル名が付いているプログラムをロードして実行させます。メモリ・カード内に「AUTOSTART」のファイルが存在しない、またはメモリ・カードがドライブに挿入されていない場合はスペクトラム画面で動作します。  
通常にスペクトラム画面で立ち上げる場合には、このモードを選択して下さい。

MEM : 本器コントローラ機能にあるバックアップ・メモリに格納されているプログラムを実行させます。バックアップ・メモリにプログラムが無い場合、スペクトラム画面で動作します。  
バックアップ・メモリのプログラムの有無は、BASICメニューの「RUN」表示で確認します。(レジューム機能参照)

注) ・ 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・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。

## 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、<F9>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、<F10>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。

<F9>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。

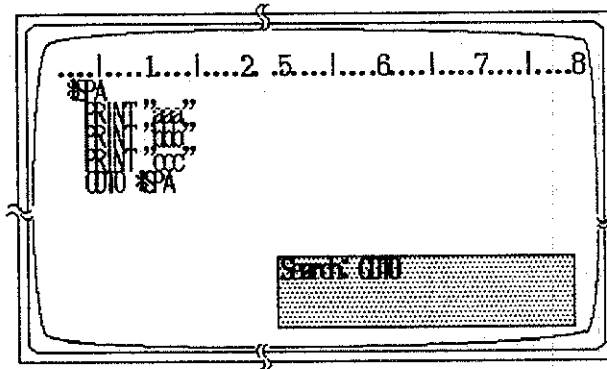


図5-22 カーソル位置以降の文字列検索

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。

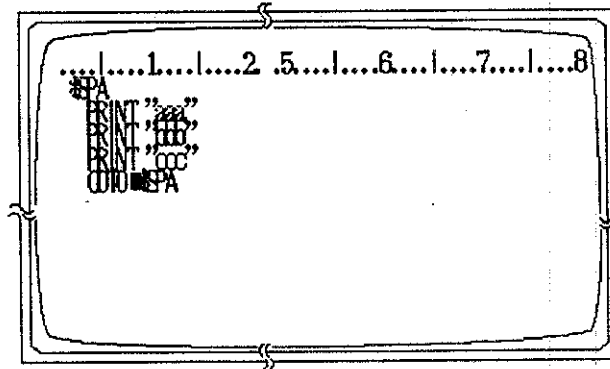


図5-23 実行結果

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。

F9	カーソル位置以降に向かって検索する
F10	カーソル位置以前に向かって検索する

図5-24 検索に関するキーボード

## 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。〈F11〉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。

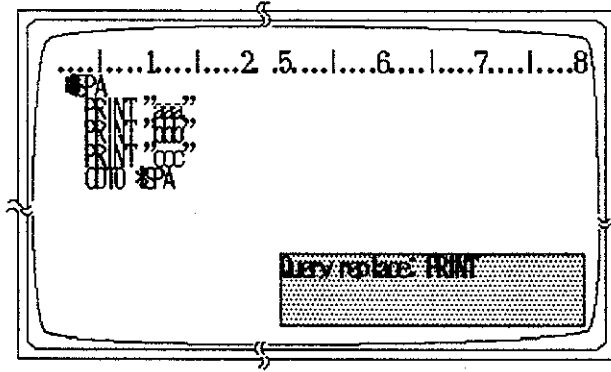


図5-25 置換対象文字列の入力

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。

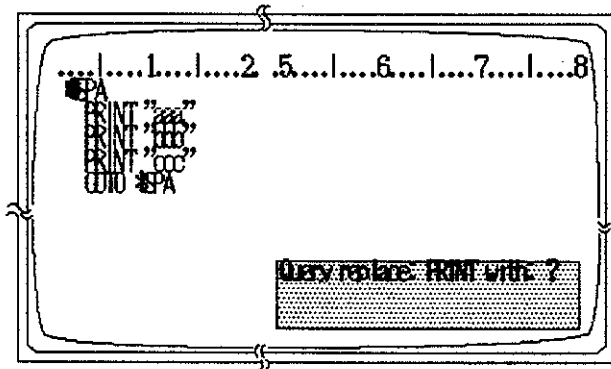


図5-26 置換文字列の入力

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〈Backspace〉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〈ESC〉または〈Ctrl-C〉キーを入力します。



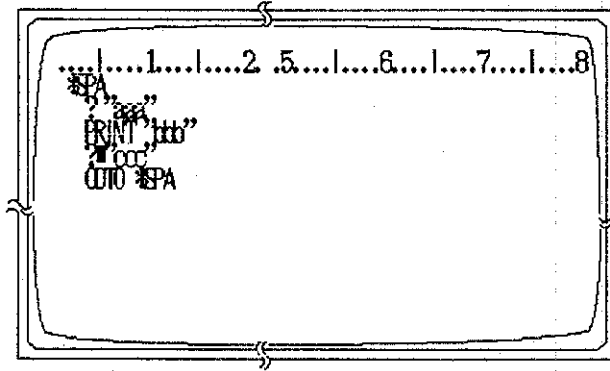


図5-27 実行結果

F11	文字列を置き換える
space	置換する
Backspace	置換しない
.	置換して、終了する
ESC(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。〈Shift+F9〉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〈Enter〉キーを入力すると、行番号が自動的に出力されます。

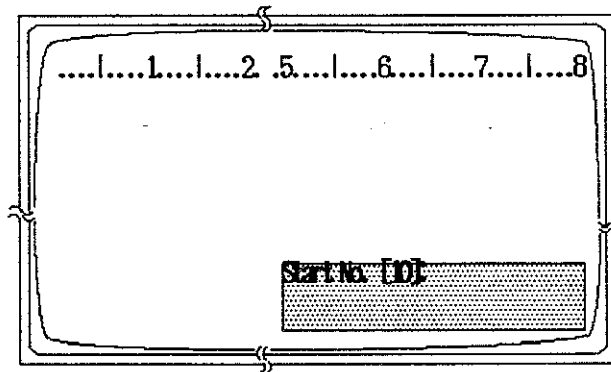


図5-30 開始番号の指定

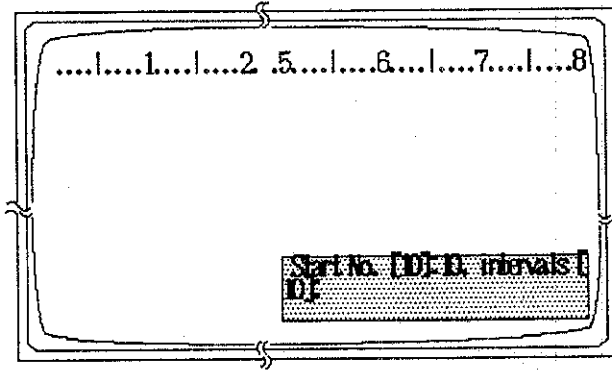


図5-31 行番号間隔の指定

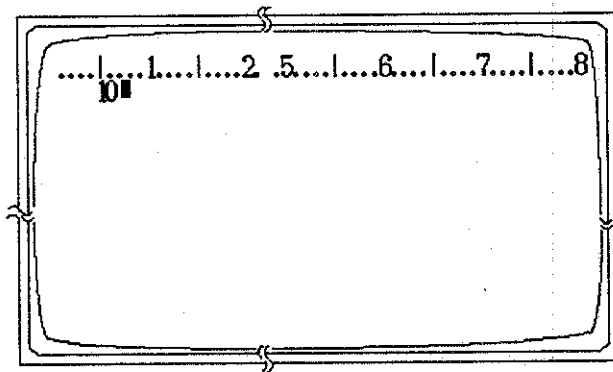


図5-32 自動行番号機能

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。〈Shift+F10〉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。

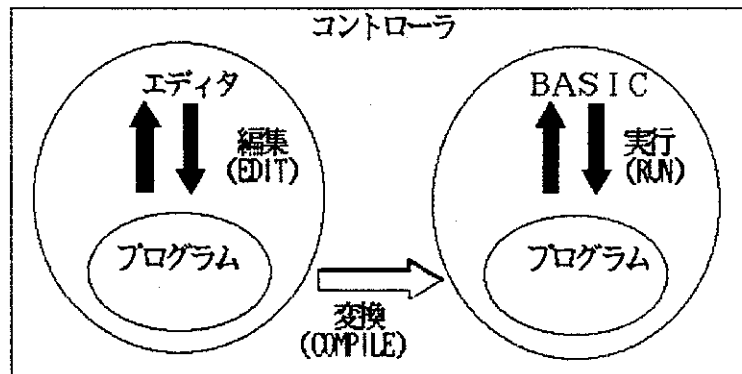

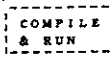




図5-33 エディタとインタプリタとの関係

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。<F1>キーまたは計測器の   キーまたは、ポップアップメニューの "COMPILE & 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、<F3>キーまたは計測器の   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<F3>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、<Break>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。<F2>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。

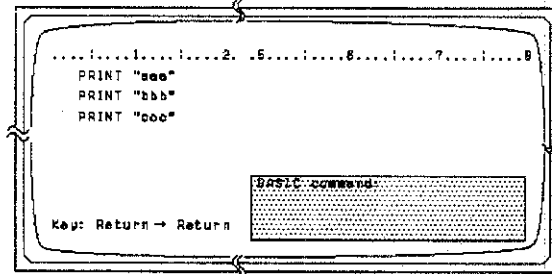


図5-34 BASICモード

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、<F4>キーまたは計測器の <sup>STOP</sup> CONTINUE 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。

Shift+F9	自動的に行番号を挿入する
Shift+F10	行番号のリナンバリング
F1	プログラムを転送後、実行する
F2	BASICモードへ移行する
F3	すでに転送済みのプログラムを実行する
F4	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に関するキーバッド

## 5. 10 ヘルプ

エディタの機能とキーボードの関係が画面上で確認できます。

<SHIFT+F11>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。

<ESC>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。

-- Command Key Help Menu --		BASIC
Backspace	Delete backward character	LOAD
Del	Delete forward character	.....
Shift-Del	Kill from the cursor position to end of line	COMPILE
Shift-F1	Get a file, read write	& RUN
Shift-F2	Save current file	.....
Shift-F3	Write a file	RUN
Shift-F4	Move cursor to file name view window	.....
Shift-F5	Make current window only one	.....
Shift-F6	Split current window	.....
Shift-F7	Move cursor to the next window	.....
Shift-F8	Clear screen and redisplay everything	.....
Shift-F9	Set line no. of the BASIC	.....
Shift-F10	Renumber line no. of the BASIC	.....
Shift-F11	Help menu (Command keys)	.....
Shift-F12	SCRATCH	.....
F1	Compile and execute RUN	.....
F2	Change to BASIC mode	.....
F3	Execute RUN	RETURN
F4	Execute CONT	.....

図5-36 ヘルプメニュー

Shift+F11	ヘルプメニューを表示する
ESC(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ボード

### 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に役立ちます。

<Alt>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。

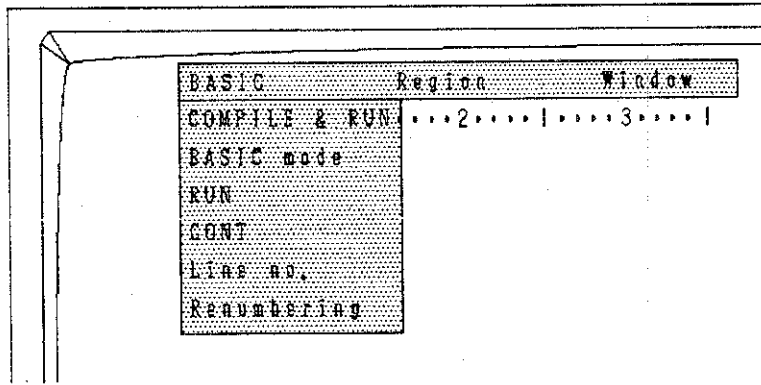


図5-38 ポップアップメニュー

BASIC	Region	Window/Other	File	Search/Replace
COMPILE & 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<⇒>,<⇐>で行います。

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。

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

図5-40 ポップアップメニューに関するキーボード

メニュー	説明	参照先
COMPILE & 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5-41 ポップアップメニューの説明



	F1	F2	F3	F4	F5	F6	F7	F8
Base Case	COMPILE and RUN	BASIC mode	RUN	CONT	Set mark	Kill region	Copy region	Yank
Shift Case	Load	Save	Write	File window	Only	Split	Next	Redisplay

	F9	F10	F11	F12
Base Case	Forward search	Backward search	Query replace	CNTRLR ON
Shift Case	Line no.	Renumbering	Help	SCRATCH

File window: ファイル表示一覧のウィンドウへ移動する。

図5-42 ファンクション・キーの割り当て

## 5. 1 2 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、<ESC>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。

ESC(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。<Shift+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> <Enter>と入力すると、エディタは初期化され、<n> <Enter>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。

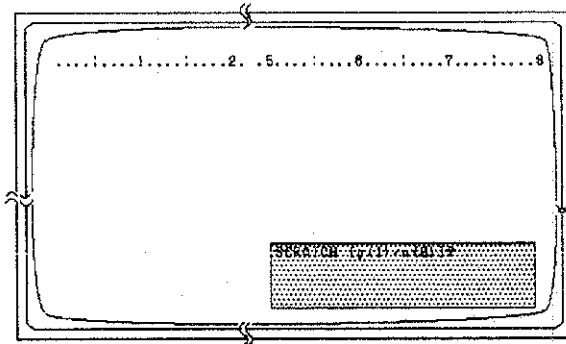


図5-44 エディタの終了ミニウィンドウ

↓  
<y> <Return> を入力する

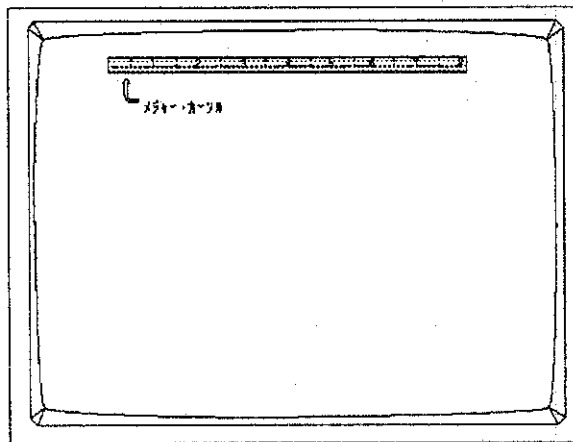


図5-45 エディタの初期化

Shift+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



第2部 システム・コントローラ



## 第 2 部の目次

1. コントローラ機能 (BASIC GPIBコントローラ)	1- 1
1.1 概要	1- 1
1.2 BASICプログラミング	1- 2
1.2.1 プログラム構造	1- 2
1.2.2 キーワード	1- 3
1.2.3 ショート・ネーム	1- 3
1.3 オブジェクト	1- 4
1.4 定数	1- 5
1.4.1 整定数	1- 5
1.4.2 実定数	1- 5
1.4.3 文字列定数	1- 6
1.4.4 ラベル定数	1- 6
1.5 変数	1- 7
1.5.1 スカラ変数	1- 7
1.5.2 システム変数	1- 8
1.5.3 配列	1- 9
1.6 関数	1- 10
1.6.1 組み込み関数	1- 10
1.6.2 ビルトイン関数	1- 12
1.7 演算式	1- 14
1.7.1 代入演算子	1- 14
1.7.2 単項算術演算子	1- 16
1.7.3 2項算術演算子	1- 18
1.7.4 論理演算子	1- 19
1.7.5 比較演算子	1- 20
1.7.6 サブ・ストリング演算子	1- 21
1.7.7 ビット演算子	1- 22
1.8 演算の優先順位	1- 23
1.9 文字列演算	1- 24
1.9.1 文字列の連結	1- 24
1.9.2 文字列の比較	1- 24
1.9.3 型変換	1- 24
2. コマンドとステートメントの文法と解説	2- 1
2.1 概要	2- 1
2.2 はじめに	2- 1
2.2.1 説明の構成	2- 1
2.3 各種コマンドとステートメントの説明	2- 6
3. ビルトイン関数	3- 1
3.1 概要	3- 1
3.2 はじめに	3- 1
3.2.1 使用開始の前に	3- 1
3.2.2 使用上の注意	3- 4
3.2.3 説明の構成	3- 5
3.3 各種ビルトイン関数の説明	3- 11
3.4 各種グラフィックの説明	3- 56

4.	マスタ/スレーブ・モード .....	4- 1
4.1	概要 .....	4- 1
4.2	CONNECTED HOSTモード .....	4- 2
4.2.1	概要 .....	4- 2
4.2.2	R3265A/3271Aコントローラの制御および変数の出力 .....	4- 3
4.2.3	R3265A/3271Aコントローラとホスト・コンピュータとの同期 .....	4- 5
4.2.4	R3265A/3271Aコントローラ制御コマンド .....	4- 7
4.3	MASTER/SLAVEモード .....	4-18
4.3.1	概要 .....	4-18
4.3.2	BASICコマンド .....	4-19
4.3.3	スレーブ・モードのGPIBコマンド .....	4-20
4.3.4	外部コントローラでのコントロール .....	4-22
5.	レジューム機能 .....	5- 1
6.	コントローラ・キー .....	6- 1



## 1. コントローラ機能 (BASIC GPIBコントローラ)

### 1.1 概要

コントローラ機能の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。

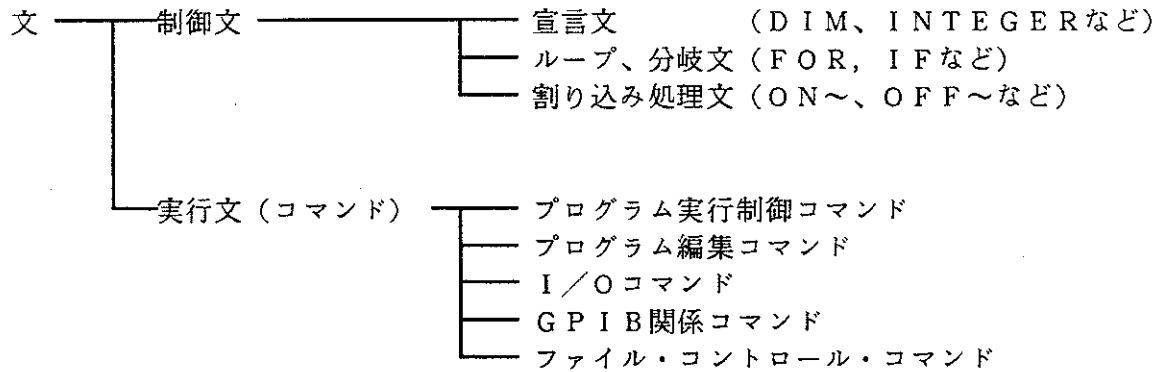
注) コントローラ機能のGPIBアドレスは、BASICのCONTROL命令で設定します。

## 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
```

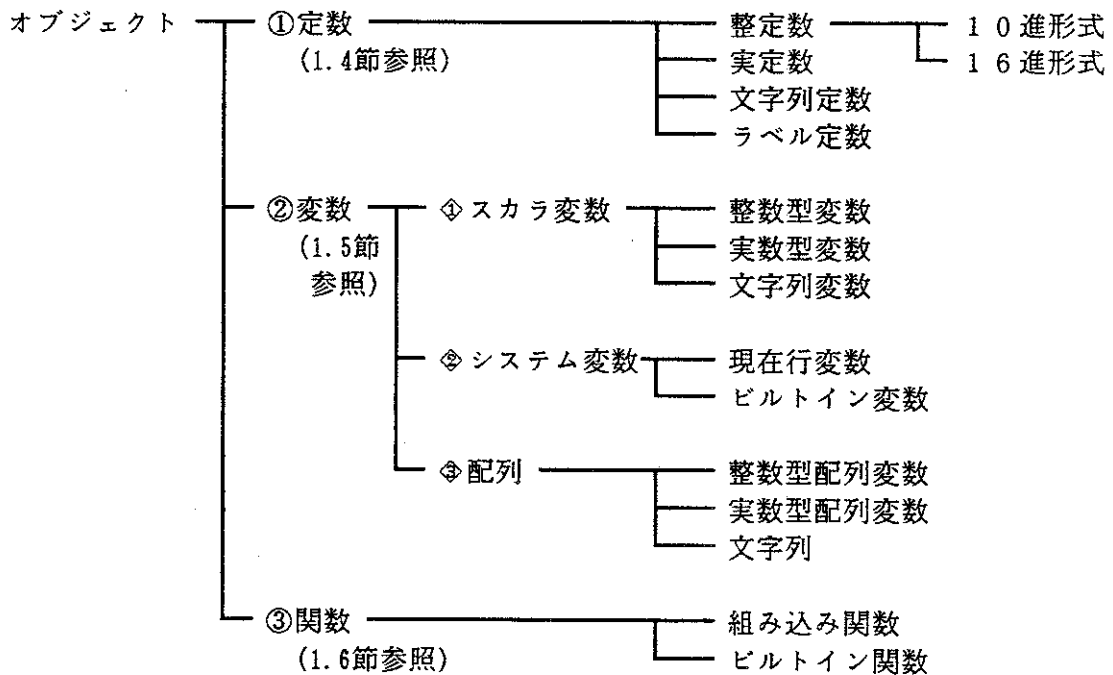
【フルネームとショートネームの対応表】

フルネーム	ショートネーム
C U R S O R	C S R
E N T E R	E N T
I N I T I A L I Z E	I N I T
I N P U T	I N P
O U T P U T	O U T
P R I N T F	P R F
U S I N G	U S E

### 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。

キーボードに割り当て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
STRGS		: 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 STRGS [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COMPILE 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一になります。

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)	<p>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。</p> <p>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</p> <p>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。</p> <p>例)</p> <pre>PRINT ERRM\$ (ERRN)</pre> <p>: 直前のエラーメッセージを出力します。</p>
NUM (文字列式)	<p>文字列式の先頭の1文字のASCIIコードを返します。</p> <p>例)</p> <pre>NUM ("ABC") → 65</pre> <pre>A\$ = "XYZ"</pre> <pre>NUM (A\$) → 88</pre>
CHR\$ (算術式)	<p>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。</p> <p>例)</p> <pre>CHR\$ (65) → "A"</pre> <pre>A = 88</pre> <pre>CHR\$ (A) → "X"</pre>
LEN (文字列式)	<p>文字列式の長さを返します。</p> <p>例)</p> <pre>LEN ("ADVANTEST") → 9</pre> <pre>A\$ = "CORP."</pre> <pre>LEN (A\$) → 5</pre>
POS (文字列式1 文字列式2)	<p>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。</p> <p>例)</p> <pre>A\$ = "AN"</pre> <pre>POS ("ADVANTEST", A\$) → 4</pre>
ABS (算術式)	<p>算術式の値の絶対値を返します。</p> <p>例)</p> <pre>ABS (-1.2) → 1.2</pre>

組み込み関数	説明
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 (0)	BASICのメモリの残り容量(バイト)を返します。 例) FRE (0) → 502718 (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
RAND (0)	0~1までの乱数を発生します。 例) RAND (0) → .375
LGT (算術式)	算術式の値に対する常用対数(10を底とした対数)を返します。 例) LGT (10) → 1

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 = TRACE (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 (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)

## 1.7 演算式

オブジェクトを操作するのが演算子です。演算子とオブジェクトの組合せで式を構成します。

演算子	—	(1) 代入演算子	(1.7.1項参照)
	—	(2) 単項算術演算子	(1.7.2項参照)
	—	(3) 2項算術演算子	(1.7.3項参照)
	—	(4) 論理演算子	(1.7.4項参照)
	—	(5) 比較演算子	(1.7.5項参照)
	—	(6) サブ・ストリング演算子	(1.7.6項参照)
	—	(7) ビット演算子	(1.7.7項参照)

### 1.7.1 代入演算子

代入式は式自体に値を持ちます。

例)

```
A = 1 2 3
PRINT A
PRINT B $ = " ADVANTEST "
PRINT (A = 2) + A
```

【実行結果】

```
1 2 3. 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 = 1 2 3	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 " 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5"
S$ => " TEST" : PRINT S$
S$ =< " TEST" : PRINT S$
```

【実行結果】

```
5. 0
15. 0
12. 0
144. 0
72. 0
2. 0
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
                        TEST
TEST
```

### 1.7.2 単項算術演算子

-	マイナス符号
+	プラス符号
++	<p>前置／後置 インクリメント (変数に1を加える)</p> <ul style="list-style-type: none"> <li>・前置インクリメント 例)  <math>A = 2</math>      Aに2を入力  <math>B = ++A</math>    Aに1を加えてからBを入力</li> <li>・後置インクリメント 例)  <math>A = 2</math>      Aに2を入力  <math>B = A++</math>    BにAを入力してからAに1を加える</li> </ul>
--	<p>前置／後置 デクリメント (変数から1を引く)</p> <ul style="list-style-type: none"> <li>・前置デクリメント 例)  <math>A = 2</math>      Aに2を入力  <math>B = --A</math>    Aから1を引いてからBを入力</li> <li>・後置デクリメント 例)  <math>A = 2</math>      Aに2を入力  <math>B = A--</math>    BにAを入力してからAから1を引く</li> </ul>

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
ABCDEFGHI
```

#### 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$="ABCDEFGG"  
PRINT A$[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$="ABCDEFGG"  
PRINT A$[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) 1 2 3. 4  
4 5 6. 7  
1 2 3  
4 5 6. 7



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

機能	コマンドとステートメント	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
算 術 関 数	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
ビ ット 演 算	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
割 り 込 み 制 御	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
文 字 列 操 作	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
	CHRS	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	コマンドとステートメント	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 #	ファイルを開く。	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
	SCREEN	基本画面を設定する。	2-113
	KEY ON	ユーザ定義メニューを表示する。	2- 60
	KEY OFF	ユーザ定義メニューを消去する。	2- 59
	PANEL	パネルキーの制御を許可/禁止する。	2- 93
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	コマンドとステートメント	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
	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
	INTERFACE CLEAR	IFCを送出する。	2- 56
LOCAL	LOCAL	指定装置をローカル状態にする。	2- 64
	LOCAL LOCKOUT	指定装置をローカル・ロックアウト状態にする。	2- 65
コ	OUTPUT(OUT)	データをGPIBに出力する。 (パラレルI/O含む)	2- 89
	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	SROLL	指定装置のシリアル・ボールを行う。	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。

No	コマンドとステートメント	ページ	No	コマンドとステートメント	ページ
1	BUZZER	2- 7	41	OFF SRQ/ISRQ	2- 76
2	CALL	2- 9	42	ON END ~	
3	CAT	2- 11		GOTO/GOSUB	2- 77
4	CLEAR	2- 13	43	ON ERROR ~	
5	CLS	2- 14		GOTO/GOSUB	2- 79
6	CLOSE #	2- 16	44	ON KEY ~	
7	CONT	2- 18		GOTO/GOSUB	2- 81
8	CONTROL	2- 19	45	ON SRQ/ISRQ ~	
9	CURSOR (CSRと省略可)	2- 21		GOTO/GOSUB	2- 83
10	DELIMITER	2- 23	46	OPEN #	2- 86
11	DIM	2- 25	47	OUTPUT #	2- 88
12	DISABLE INTR	2- 27	48	OUTPUT (OUTと省略可)	2- 91
13	ENABLE INTR	2- 29	49	PANEL	2- 93
14	ENTER #	2- 30	50	PAUSE	2- 94
15	ENTER (ENTと省略可)	2- 34	51	PRINTF (PRFと省略可)	2- 95
16	FOR.....TO.....STEP		52	PRINT (?と省略可)	2- 98
	BREAK/CONTINUE~NEXT	2- 36	53	PRINT USING (USEと省略可)	2-100
17	GLIST	2- 39	54	PRINTER	2-102
18	GLISTN	2- 41	55	READ DATA/RESTORE	2-104
19	GOSUB~RETURN	2- 43	56	REM (!と省略可)	2-106
20	GOTO	2- 45	57	REMOTE	2-107
21	GPRINT	2- 46	58	RENAME	2-108
22	GPRINT USING (USEと省略可)	2- 48	59	REQUEST	2-109
23	IF THEN ELSE/ ELSE IF/END IF	2- 50	60	RUN	2-110
24	INITIALIZE (INITと省略可)	2- 53	61	PURGE	2-111
25	INTEGER	2- 54	62	SCRATCH	2-112
26	INTERFACE CLEAR	2- 56	63	SCREEN	2-113
27	INPUT (INPと省略可)	2- 57	64	SELECT CASE/ CASE ELSE/END SELECT	2-115
28	KEY OFF	2- 59	65	SEND	2-117
29	KEY ON	2- 60	66	SROLL	2-119
30	LIST	2- 61	67	SPRINTF	2-121
31	LISTN	2- 62	68	STEP	2-124
32	LLIST	2- 63	69	STOP	2-125
33	LLISTN	2- 64	70	TRIGGER	2-126
34	LOCAL	2- 66	71	WAIT	2-127
35	LOCAL LOCKOUT	2- 67			
36	LPRINT	2- 68			
37	LPRINT USING (USEと省略可)	2- 70			
38	OFF END	2- 72			
39	OFF ERROR	2- 73			
40	OFF KEY	2- 74			



## (1) B U Z Z E R

### 【概要】

本器内臓のブザーを鳴らします。

### 【書式】

B U Z Z E R <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】

B U Z Z E R 440, 1000 : ラの音を1秒間鳴らす。

### 【注意】

B U Z Z E R 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B U Z Z E R 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、W A I T 命令をB U Z Z E R 命令の次にB U Z Z E R 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。

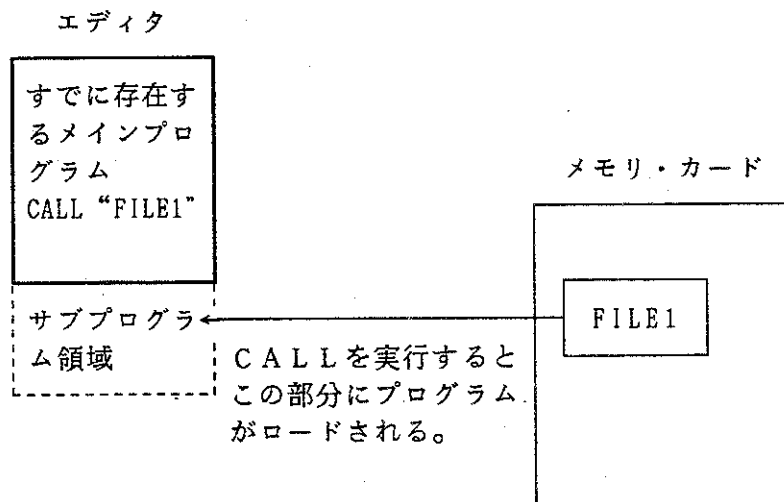
### 【書式】

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) C A T

#### 【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】

C A T

#### 【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

Total 25600 Bytes

Used 24576 Bytes (96 %)

—— 登録ファイル数 / 登録可能ファイル数

—— 全カード容量

—— 使用容量

(占有率%)

【例】

C A T

【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) C L S

### 【概要】

本体画面をクリアします。

### 【書式】

CLS [1 | 2]

### 【解説】

- ・ 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) C L O S E    #

### 【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】

C L O S E    <#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。

C L O S E    \*

このコマンドですべてのファイルをクローズします。

### 【例】

C L O S E    # F D

C L O S E    \*

### 【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で行ってください。(その他の実行は、本体CRTのCONTキーを押してください。)
- ・ 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 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をします。

・レジスタ 2  
LISTの出力を指定数分だけスペースをあげ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をします。

0 : フル・ネーム出力  
1 : ショート・ネーム出力

・レジスタ 4  
GPIBアドレスを設定をします。  
0から30までの数値を設定をします。  
\*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

で設定をします。

```
_____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

GP I B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】

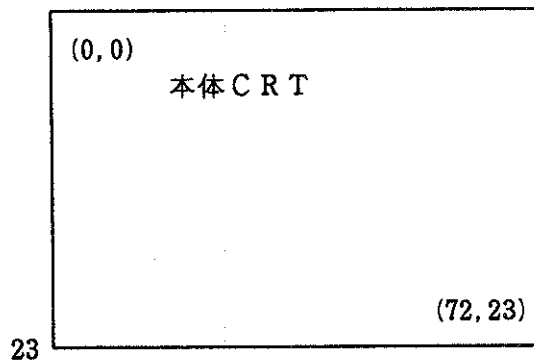
CURSOR <X:カラム>, <Y:行>

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。  
CRTでは、左上が(0, 0)になります。

72



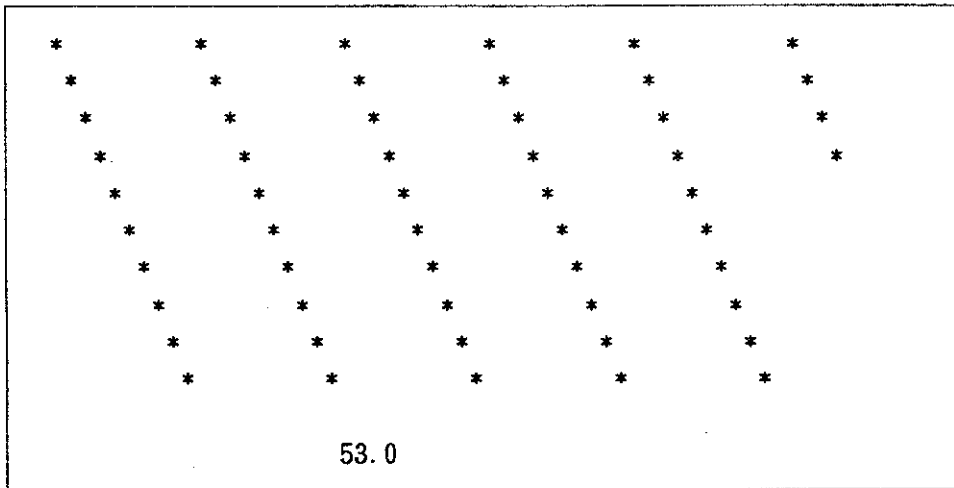
### 【例】

```
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
```

### 【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) D I M

### 【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】

D I M <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)
- D I M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りなくなり、エラーとしてプログラムの実行を中止します。  
(memory space full)
- 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】

D I M A ( 2 0 ) : 実数型配列変数Aを20個確保する。  
D I M B ( 3 0 ) , C ( 2 5 , 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。  
D I M S \$ [ 5 0 ] : 文字列変数S\$を50文字分確保する。  
D I M D ( 1 5 ) , T \$ [ 5 0 ] : 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。  
D I M 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わります。
- ・ 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),SS[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を代入
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を代入
NEXT Y	: NEXT Y
NEXT X	: NEXT X
SS="ABCDEFGHIJKLMNQRSTUWXYZ"	: S\$に文字列26文字を代入
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 SS	: 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
A B C D E F G I H J K L M N O P Q R S T
```

## (12) D I S A B L E I N T R

### 【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】

D I S A B L E I N T R

### 【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 D I S A B L E I N T R 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 SWISC 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を代入
  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と表示します。

(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。

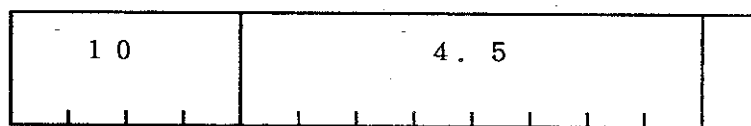
BINARY タイプ

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)

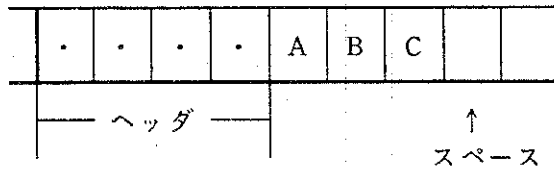
```
INTEGER I  
OPEN "FILE" FOR INPUT AS #FD  
ENTER #FD; I, R, S$
```



変数が実数型のときは、8バイトのデータを読み込み、変数に入力する。

変数が整数型のときは4バイトのデータを読み込み、変数に代入する。





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**

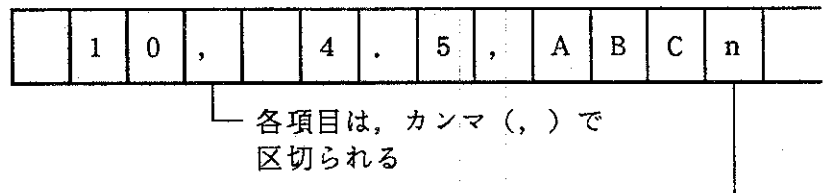
入力項目の数にかかわらずライン・フィールドまで読み込みます。カンマ ( , ) までが1つのデータとなり、入力項目の型に変換して入力されます。

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・フィールド ( 0 x 0 a ) が付きます。

例)

```

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



文字列の最後にはライン・フィールド ( 0 x 0 a ) がある

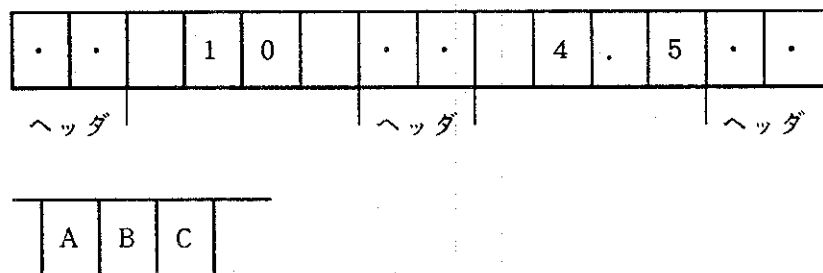
**ASCII タイプ**

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 BS[80]
OUTPUT 3;"CF?"
ENTER 3;A
OUTPUT 5;"SP?"
ENTER 5;BS
PRINT A
PRINT BS
```

例2

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

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。
- ・ 一对で使用するFOR~NEXT文の<数値変数>は、同じものでなくてはなりません。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。)
  - 増分値が正の値で、<初期値>が<最終値>より大きい場合
  - 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) G L I S T

### 【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(COMPILE 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、(54) 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(COMPILE 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 **COMPILE 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、**COMPILE 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

**COMPILE 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、(54) 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) G O S U B ~ R E T U R N

### 【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】

```
G O S U B <ラベル>
```

```
R E T U R N
```

### 【解説】

- ・ G O S U B <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G O S U B文の次の命令へと処理の制御を戻します。
- ・ G O S U B ~ R E T U R N文は、入れ子 ( n e s t ) 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 ( n e s t ) が多過ぎるとメモリ容量が足りなくなり、エラーになることがあります。

( G O S U B n e s t o v e r f l o w )

### 【例】

```
G O S U B * S 1
```

```
* S 1
```

```
A = A * 2
```

```
R E T U R N
```

### 【注意】

- ・ G O S U B <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。  
( U n d e f i n e d L A B E L )

【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 \*L A

### 【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】

- 0
- 0
- 0
- 0
- 0
- 0
- 0
- 0
- 0

## (21) G P R I N T

### 【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】

- ・ (54) 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、(54) 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) G P R I N T   U S I N G  
( U S E 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。
- ・ (54)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 プリントに出力する前には、必ず、(5.4) PRINTER 命令で GPIB プリントのアドレスを設定して下さい。(GPIB プリントとのアドレスが合わないとプリントに出力されません。)
- ・ 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"  
AS="K,5X,B,DDDRDD,k"  
GPRINT USE AS; 2.34,65,1.2, "\r"
```

【実行結果】

GPIB アドレス 5 のプリント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) I F        T H E N    E L S E / E L S E    I F / E N D    I F
---

【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】

- I F <論理式> THEN <文>
- I F <論理式> THEN  
    <複文>  
  E N D I F
- I F <論理式> THEN  
    <複文>  
  E L S E  
    <複文>  
  E N D I F
- I F <論理式> THEN  
    <複文>  
  E L S E I F <条件式> THEN  
    <複文>  
  E L S E I F <条件式> THEN  
    <複文>  
    .  
    .  
  E L S E  
    <複文>  
  E N D I F

【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 : ① 10 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 : ② I = 50、または I = 60 ならば以下の処理
    IF I=50 THEN : ③ I = 50 ならば以下の処理
      PRINT "I=50" : 文字列出力
    ELSE : ③ I = 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 : ② I = 70 ならば以下の処理
    PRINT "I=70" : 文字列出力
  ELSE IF I=90 THEN : ② I = 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
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) I N T E G E R

### 【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】

```
I N T E G E R < X > | < X > ( 数 値 表 現 式 ) { , < X > |  
                                     < X > ( 数 値 表 現 式 ) }
```

X : 数値変数

### 【解説】

- I N T E G E R 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、整数で扱える範囲と同等です。  
- 2, 1 4 7, 4 8 3, 6 4 8 ~ + 2, 1 4 7, 4 8 3, 6 4 7
- 整数しか扱わない変数では、I N T E G E R 文で宣言した方が、高速に処理できます。
- I N T E G E R 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、メモリ領域が足りなくなり、エラーとしてプログラムの実行を中止します。  
( m e m o r y s p a c e f u l l )
- 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】

```
I N T E G E R A  
I N T E G E R B, C  
I N T E G E R D ( 2 0 ), E ( 3 0 )
```



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

## (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100 $\mu$ s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、トーカー、またはリスナーの状態を解除します。

### 【例】

INTERFACE CLEAR

### 【注意】

- ・ スレーブ・モードは機能しません。

### 【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いりません。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, " MHz"
OUTPUT 31;" SP", SF, " MHz"
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) KEY OFF

### 【概要】

ユーザ定義メニューの表示を消去します。

### 【書式】

KEY OFF

### 【解説】

- ・ ユーザ定義メニューの表示を消去します。
- ・ KEY OFF実行後は、ソフトキーからの割り込みは発生しません。

### 【例】

KEY OFF

### 【注意】

- ・ KEY OFFは、ユーザ定義メニューの表示を消去するだけで割り込みの許可/禁止はできません。(13)ENABLE INTR、(12)DISABLE INTRを参照して下さい。

### 【プログラム例】

```
ON KEY 1 LABEL "KEY-OFF" GOSUB *KOF
KEY ON
*LOOP
  ENABLE INTR
  GOTO *LOOP
*KOF
  PRINT "KEY-OFF"
  KEY OFF
  RETURN
```

## (29) KEY ON

### 【概要】

ユーザ定義メニューをソフトキー・メニューに表示します。

### 【書式】

KEY ON

### 【解説】

- ・ ユーザ定義メニューを表示し、ON KEY命令によるソフトキーからの割り込みを可能にします。また「ON KEY LABEL」で定義した文字も表示します。

### 【例】

KEY ON

### 【注意】

- ・ KEY ONは、ユーザ定義メニューの表示するだけで割り込みの許可/禁止はできません。(13) ENABLE INTR、(12) DISABLE INTRを参照して下さい。

### 【プログラム例】

ON KEY 1 LABEL "PROGRAM \n A" GOSUB *PRG_A	: KEY 1 の定義
ON KEY 2 LABEL "PROGRAM \n B" GOSUB *PRG_B	: KEY 2 の定義
KEY ON	: LABELをソフトメニューに表示
ENABLE INTR	: 割り込みを許可する
*LOOP	: ループ
GOTO *LOOP	:
*PRG_A	: KEY 1 が押されたら
CALL "PRG_A"	PRG_AをCALLする
RETURN	
*PRG_B	: KEY 2 が押されたら
CALL "PRG_B"	PRG_BをCALLする
RETURN	

## (30) LIST

### 【概要】

本体の画面にプログラム・リストを出力します。

### 【書式】

```
LIST [<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(COMPILE 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】

```
LIST
```

```
LIST *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。

```
LIST [<出力開始行>] [, [<出力終了行>]]
```

例)

```
LIST 100
```

```
LIST 100,
```

```
LIST , 200
```

```
LIST 100, 200
```

### 【プログラム例】

エディタのCOMPILE 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。

## (31) LISTN

### 【概要】

本体の画面にプログラム・リストを出力します。

### 【書式】

LISTN [

### 【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 (30) 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
```



## (32) L L I S T

### 【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 (COMPILE 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COMPILE 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。

### (33) L L I S T N

#### 【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 [

#### 【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ず。
- ・ 出力方法は(32) L L I S T 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。

## (34) 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 (true) にするには(57)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)。

## (35) 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う。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。

## (36) LPRINT

### 【概要】

RS-232Cで接続されている装置（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・コロンの(;)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】

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

- ① 文字列変数S\$に" DEF"を入力
- ② " ABC"を改行しないで出力
- ③ 文字列変数S\$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】

- ・ LPRINT文の最後にセミ・コロンの(;)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】

```
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
```

(37) L P R I N T    U S I N G  
 ( U S E 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"
```

```
AS="K,5X,B,DDDRDD,k"
```

```
LPRINT USE AS; 2.34,65,1.2,"\r"
```

【実行結果】

```
1.20
```

```
001.20
```

```
**1.20
```

```
2.34          A    1,20
```

## (38) 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(42) ON END命令を使います。

## (39) 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(43) ON ERROR命令を使います。

## (40) 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>

	1～7	: ソフトキー	
	10	: ステップキー	↑
キー番号 (数値表現式)	11	: ステップキー	↓
	12	: ロータリエンコーダ	⊙ (右回し)
	13	: ロータリエンコーダ	⊙ (左回し)

### 【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 10
OFF KEY 11
OFF KEY KK
OFF KEY BB
```

### 【注意】

- この命令によって解除したあとは、ソフトキー、ステップキーまたはロータリエンコーダを入力しても分岐しません。
- 分岐先の定義は(44) ON KEY命令で行います。

【プログラム例】

```
ON KEY 1 GOSUB *L1
ON KEY 2 GOSUB *L2
ON KEY 3 GOSUB *L3
ENABLE INTR
*LOOP
GOTO *LOOP
*L1
PRINT "OFF KEY 1"
OFF KEY 1
RETURN
*L2
PRINT "OFF KEY 2"
OFF KEY 2
RETURN
*L3
PRINT "OFF KEY 3"
OFF KEY 3
RETURN
```

【実行結果】

1度、割り込みが発生したキーの2度目の割り込みは発生しない。

## (41) 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(45) ON SRQ命令、またはON ISRQ命令を使います。

(42) 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(38)OFF END命令で行います。
- ・ データの書き込み、読み込みは、(47)OUTPUT #、(14)ENTER #を参照して下さい。

### 【プログラム例】

OPEN "FFF" FOR OUTPUT AS #FD	: " F F F" 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	: " F F F" 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
```



(43) 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、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(39) 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"	: " S T A R T" を画面出力
PRINT 1/0	: 1 ÷ 0 を実行
PRINT "END"	: " E N D" を出力
STOP	: プログラム終了
*ERR	
PRINT "<<ERROR>>"	: " << E R R O R >>" を出力
PRINT " >> ",ERRMS(0)	: エラー・メッセージを出力

【実行結果】

```
S T A R T
<< E R R O R >>
>>      4 0 :  0  d i v i d e
```

## (44) ON KEY ~ GOTO / GOSUB

### 【概要】

本体パネルのソフト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>
ON KEY <キー番号> LABEL <文字列表現式> GOTO<ラベル>
ON KEY <キー番号> LABEL <文字列表現式> GOSUB<ラベル>
```

	1~7	: ソフトキー	
	10	: ステップキー	↑
キー番号 (数値表現式)	11	: ステップキー	↓
	12	: ロータリエンコーダ	⊙ (右回し)
	13	: ロータリエンコーダ	⊙ (左回し)

### 【解説】

- キー番号 (1~7) は、本体パネルのソフトキーに対応する SF1~SF7 の割り込み分岐先を定義します。ソフトキーからの割り込みは、ソフトメニューのユーザ定義メニューが表示されているときのみ有効です。(KEY ON参照)
- キー番号 (1~7) までのとき、LABEL を使用してソフトキー・メニュー内に指定した文字を表示することができます。
- キー番号 (10~13) は、本体パネルのロータリエンコーダ、ステップキーによる割り込みの分岐先を定義します。
- 割り込みの受信の許可、禁止は `ENABLE INTR / DISABLE INTR` を参照して下さい。

### 【例】

```
ON KEY 1 GOTO *K3
ON KEY 2 GOSUB *K4
ON KEY XX LABEL "ON" GOTO *K1
ON KEY BB LABEL "OFF" GOSUB *K2
ON KEY XX LABEL ST$ GOTO *K1
ON KEY BB LABEL ST$&"ON" GOSUB *K2

ON KEY 10 GOSUB *STUP
ON KEY 11 GOTO *STDWN
ON KEY 12 GOSUB *ENR
ON KEY 13 GOTO *ENL
```

【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の実行になります。
- ・ 分岐先定義の解除はOFF KEY命令で行ないます。(OFF KEY 参照)
- ・ ソフトキー (SF1～SF7) による割り込みは、ユーザ定義メニューが表示されていないときには使用できません。ユーザ定義キーを表示するには、KEY ON命令を実行するか、もしくはプログラム実行中にCONTRLR ONキーを押すことにより表示します。

【プログラム例】

```
ON KEY 1 LABEL "PROGRAM \n A" GOSUB *PRG_A      : KEY 1 の定義
ON KEY 2 LABEL "PROGRAM \n B" GOSUB *PRG_B      : KEY 2 の定義
KEY ON                                           : LABEL をソフトメニューに表示
ENABLE INTR                                     : 割り込みを許可する
*LOOP                                           : ループ
GOTO *LOOP                                     :
*PRG_A                                           : KEY 1 が押されたら
CALL "PRG_A"                                    PRG_AをCALLする
RETURN
*PRG_B                                           : KEY 2 が押されたら
CALL "PRG_B"                                    PRG_BをCALLする
RETURN
```

(45) 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 による分岐先定義)
```

【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。OUTPUT 31 命令にて実行して下さい。

GPIBコード	内 容
S 0	計測部からコントローラに対し SRQ 割り込みを送信する
S 1	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
S 2	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(41)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。

## (46) 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。)

(47) 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
```

## (47) 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。

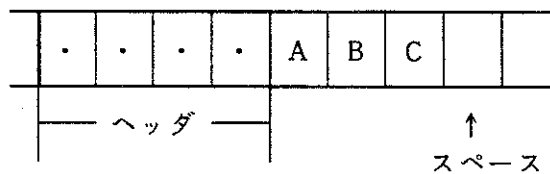
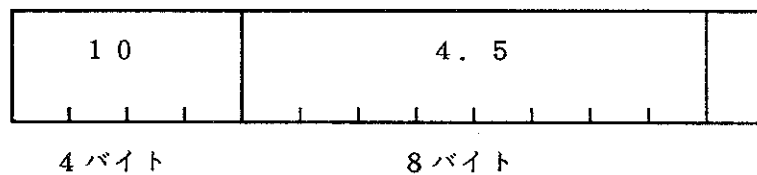
#### BINARY タイプ

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"
```

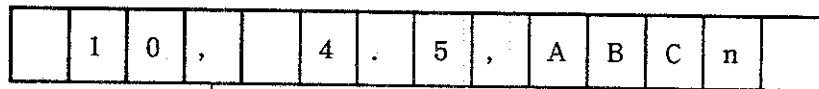


## TEXT タイプ

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"
```



各項目は、カンマ(,)で区切られる

文字列の最後にはライン・フィード(0x0a)が出力される

## ASCII タイプ

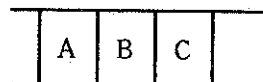
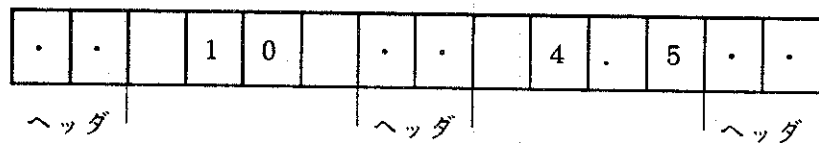
データをASCIIコードに変換して出力します。数値データは、スペースか符号がデータの先頭に付きます。

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（46）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
```

## (48) 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(true)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  
(計測部へはトーカー/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 R3265A/3271A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
```

( 4 9 ) P A N E L

【概要】

本体パネル・キーによる操作を許可／禁止します。

【書式】

PANEL <数値表現式>  
数値表現式：0～1

【解説】

パネ ル 状 態	
0	パネル・アンロック：本体パネルによる通常操作を許可します。 (センタ、スパン等の操作の可能)
1	パネル・ロック：本体パネルによる操作を禁止します。 (操作不可)

【例】

```
PANEL 0  
PANEL 1  
PANEL X
```

【注意】

- ・ パネル・アンロック時での操作でユーザ定義メニューが消された場合、  
CONT L R L ONキーを押すことでユーザ定義メニューを表示します。

【プログラム例】

```
ON KEY 1 LABEL "PANEL \n UNLOCK" GOSUB *PU  
ON KEY 2 LABEL "PANEL \n LOCK" GOSUB *PL  
KEY ON  
ENABLE INTR  
SCREEN 4  
*LL  
GOTO *LL  
*PU  
PANEL 0  
PRINT "PLEASE KEY OPERATION"  
RETURN  
*PL  
PANEL 1  
PRINT "PANEL LOCK"  
RETURN
```

## (50) 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命令または本器パネルのCONT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。



## (51) PRINTF (PRFと省略可)

### 【概要】

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】

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・サブレスを行う。

#### 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
AS="ABCD"
PRINTF ":%8s:%-8s:%8.2s:%-8.2s:", AS, AS, AS, AS
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
```

## (52) PRINT ( ? と省略可 )

### 【概要】

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】

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

- ① 文字列変数S\$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S\$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

【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
```

(53) PRINT  
USING (USE と省略可)

【概要】

数値、文字列などを編集し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】

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・タイプ的小数点になる
*	-----	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"
```

```
AS="K,5X,B,DDDRDD,k"
```

```
PRINT USE AS; 2.34,65,1.2, "\r"
```

【実行結果】

```
1.20
```

```
001.20
```

```
**1.20
```

```
2.34      A      1,20
```

## (54) P R I N T E R

### 【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】

P R I N T E R <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アドレスを毎回設定して下さい。)

### 【例】

P R I N T E R 1

P R I N T E R 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】

P R I N T E R 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
P R I N T E R 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  
A B C D

アドレス12のプリンタには  
p r i n t e r = 1 2 . 0

と出力します。

## (55) 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

## (56) 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   <<< r e m a r k >>>
!
!   ****  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
```

## (57) R E M O T E

### 【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 (true) 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 (true) 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、(34) 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 ..... (48) 参照  
SEND LISTEN ..... (65) 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。

## (58) R E N A M E

### 【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認して下さい。
- ・ ファイル名は10文字以内です。11文字以上の場合には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。

## (59) REQUEST

### 【注意】

- ・ 外部コントローラを接続した場合、またはスレーブ・モードで使用する場合は [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。

## (60) 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、`COMPILE 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`で行ってください。（プログラムの実行の開始は、本体CRTのRUNキーを押してください。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。）



## (61) 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認して下さい。
- ・ 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。

## (62) 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。

## (63) SCREEN

### 【概要】

本体管面の表示状態を指定します。

### 【書式】

SCREEN<数値表現式>

数値表現式：0～4

### 【解説】

- ・プログラム実行時における本体管面の表示状態を指定します。
- ・0～4は、本器GPIBコードのVS0～VS4に対応しています。

	管 面 状 態	
0	SPA波形画面	(測定)
1	SPA波形+BASICの合成画面	(測定)
2	BASIC画面	(測定)
3	BASIC画面	(測定なし)
4	SPA波形+BASICの2画面	(測定)

#### 0：SPA波形画面

通常の波形画面のみでPRINT命令、グラフィック命令での表示は行いません。

#### 1：SPA波形画面+BASIC画面の合成画面

波形画面上にPRINT命令、グラフィック命令合成し表示します。

#### 2：BASIC画面

BASIC画面上で行いながらPRINT命令、グラフィック命令を表示します。

#### 3：BASIC画面

測定を停止させた状態でのBASIC画面にPRINT命令、グラフィック命令を高速に表示します。

#### 4：SPA波形画面+BASIC画面の2画面

画面上部にSPA波形を表示、下部にBASIC画面を表示します。

これらの画面設定は"IP"コマンドにて初期化されるので、"IP"コマンド実行後は再設定して下さい。

なお、画面設定はOUTPUT 31;"VS0"でも可能です。詳細は、3-2ページ(2)グラフィック機能の項を参照して下さい。

【例】

```
SCREEN 0  
SCREEN 1  
SCREEN M
```

【注意】

- ・ 管面状態1～3と4ではCURSOR命令による表示位置指定がとなります。

```
(64) S E L E C T   C A S E /
      C A S E   E L S E /
      E N D   S E L E C T
```

【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】

```
S E L E C T <数値表現式>
      C A S E <数値表現式>
      E N D S E L E C T
```

【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（n e s t）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

```

## (65) SEND

### 【概要】

GP I B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】

- GP I B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(true)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GP I B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(true)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GP I B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GP I B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GP I B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GP I B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GP I B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。



## (66) 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” S W E E P E N D ” をプリントします。

## (67) S P R I N T F

### 【概要】

数値、文字列を編集し文字列変数に入力します。

### 【書式】

S P R I N T F <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. n n n n n n E [±] x x 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、[-] m m m . n n n n 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
```

## (68) S T E P

### 【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、STEP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。

## (69) S T O P

### 【概要】

プログラムの実行を中止します。

### 【書式】

S T O P

### 【解説】

- プログラムの実行を中止します。S T O P 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】

S T O P

### 【注意】

- S T O P 命令でプログラムの実行を中止させたとき、C O N T 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.
```

## (70) 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。



## (71) 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。



### 3. ビルトイン関数

#### 3.1 概要

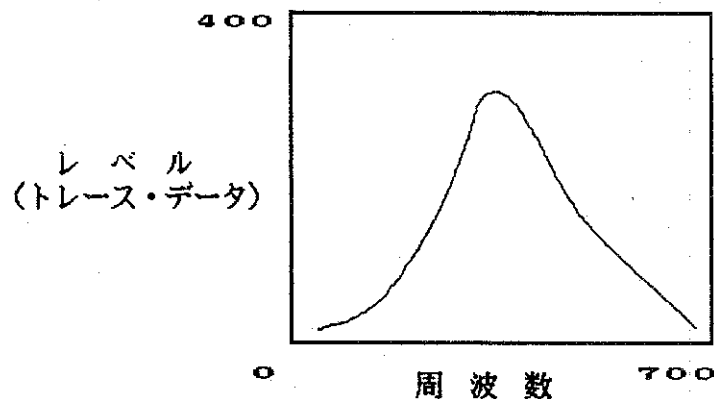
ビルトイン関数を使用することにより簡単な操作で解析処理ができ、プログラムの開発時間の短縮および高スループットを可能にします。

#### 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。  
(SCREEN命令でも指定可能。詳細は、2-113ページを参照。)

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"
波形画面+文字画面 (2画面表示)	OUTPUT 31;"VS4"

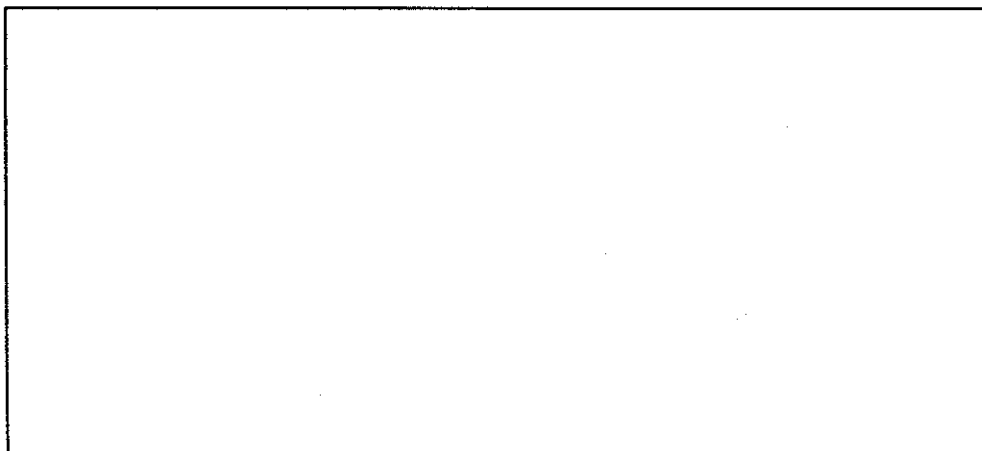
グラフィック関数を使用するときには、関数を実行する前にOUTPUT 31;"VS3"を実行して下さい。

なお、画面設定は"IP"コマンドにて初期化されるので、設定は"IP"コマンド実行後に行ってください。

グラフィック画面は以下の仕様になっています。

(0,0)

(1023,0)



(0,479)

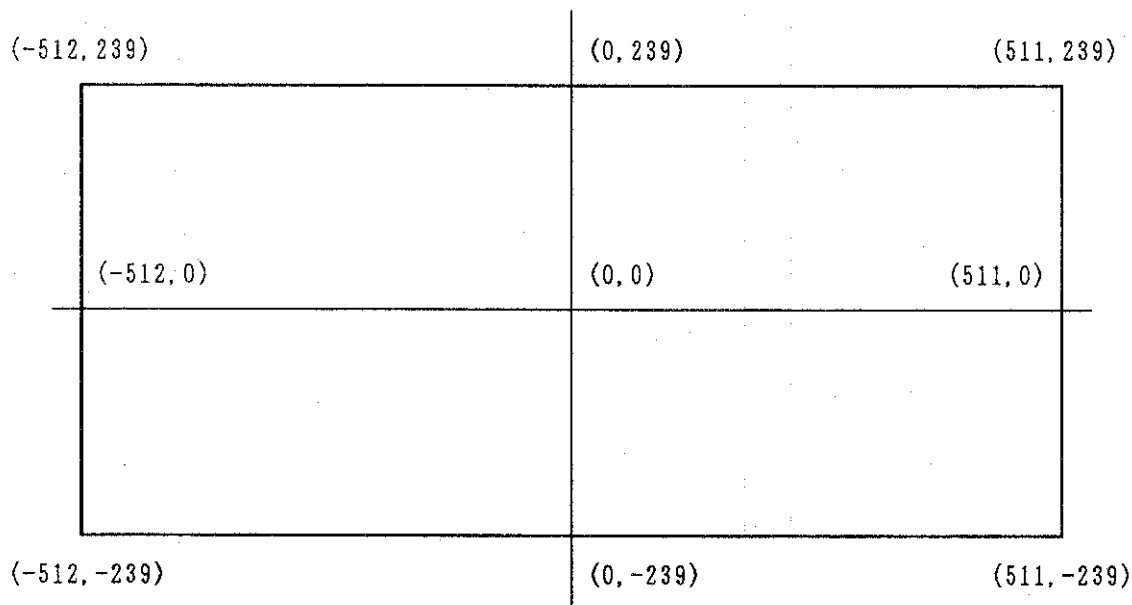
(1023,479)

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	PRPLL N	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	FRPLL N	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	VRPLL N	左からN番目の極小点のレベルを求める
33	RPL 1	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	LMTMD 1	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	LMTMD 2	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	LMTUL 1	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	LMTUL 2	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

## &lt;ビルトイン関数&gt;

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=PRPLL(N, M)
	29	F=FRPLHN(N, M)
	30	F=FRPLL(N, M)
	31	L=VRPLHN(N, M)
	32	L=VRPLL(N, M)
	33	L=RPL1(P1, P2, Dx, Dy, M)
	上下限の判定 を行う	34
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	Dx : 横軸微分係数
P : ポイント (0 ~ 700)	Dy : 縦軸微分係数
L : レベル	S : 基準値
T : トレース・データ (0 ~ 400)	Ds : 基準値幅
M : トレースA / B (0 / 1)	Dd : 被検データ
X : lossレベル	Lo : 下限値
C : チェック値 (0, 1, 2, -1)	Up : 上限値
N : リップル数	
W : ワット (電力)	



### 3.3 各種ビルトイン関数の説明

以下の順に説明します。

No	関数	ページ
1	FREQ (周波数)	3-12
2	DFREQ (周波数)	3-13
3	POINT (ポイント:横軸 [0~700])	3-14
4	DPOINT (ポイント:横軸 [0~700])	3-15
5	LEVEL (レベル)	3-16
6	DLEVEL (レベル)	3-17
7	LVPOINT (トレース・データ: [0~400])	3-18
8	LVDPOINT (トレース・データ: [0~400])	3-19
9	VALUE (レベル)	3-20
10	DVALUE (レベル)	3-21
11	CVALUE (レベル)	3-22
12	DCVALUE (レベル)	3-23
13	FMAX (周波数)	3-24
14	FMIN (周波数)	3-25
15	PMAX (ポイント:横軸 [0~700])	3-26
16	PMIN (ポイント:横軸 [0~700])	3-27
17	MAX (レベル)	3-28
18	MIN (レベル)	3-29
19	BND (周波数)	3-30
20	BNDL (周波数)	3-31
21	BNDH (周波数)	3-32
22	CBND (周波数)	3-33
23	CBNDL (周波数)	3-34
24	CBNDH (周波数)	3-35
25	NRPLH (極大点の数)	3-36
26	NRPLL (極小点の数)	3-38
27	PRPLHN (ポイント:横軸 [0~700])	3-40
28	PRPLLN (ポイント:横軸 [0~700])	3-41
29	FRPLHN (周波数)	3-42
30	FRPLLN (周波数)	3-43
31	VRPLHN (レベル)	3-44
32	VRPLLN (レベル)	3-45
33	RPL1 (レベル)	3-46
34	LMTMD1 (チェック値 [0, 1, 2])	3-47
35	LMTMD2 (チェック値 [0, 1, 2])	3-48
36	LMTUL1 (チェック値 [0, 1, 2])	3-49
37	LMTUL2 (チェック値 [0, 1, 2])	3-50
38	POWER (総電力)	3-51
39	RTRACE (トレース・データ: [0~400])	3-52
40	WTRACE	3-54

## (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 ~ 7 0 0 )

### 【戻り値】

正常終了時 : ポイント値を周波数に換算した値 ( H z ) 。ゼロスパン時は、時間 ( S e c )

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 ( 3 5 0 )

400ポイント目の周波数を求めます。

I = 2 0 0

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 ( P 1 , P 2 )

P 1 : 指定ポイント1 ( 0 ~ 7 0 0 )

P 2 : 指定ポイント2 ( 0 ~ 7 0 0 )

### 【戻り値】

正常終了時 : 指定ポイント間 P 1 , P 2 の周波数幅 ( H z ) 。ゼロスパン時は時間 ( S e c )  
エラー時 : 不定値

### 【エラー】

- ・ 指定ポイントが 0 ~ 7 0 0 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】

P 1 > P 2 の場合、P 1 と P 2 を入れ換えて処理します。

### 【例】

ポイント値 3 0 0 と 4 0 0 の間の周波数を求めます。

FS = DFREQ ( 3 0 0 , 4 0 0 )

I = 4 0 0

FS = DFREQ ( I , 3 0 0 )

### (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-10 dBmのトレース・データを求めます。

TD=LVPOINT (-10)

レベル値-20 dBm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 -5 dBm ~ -10 dBm 間のトレース・データを求めます。

TD=LVDPOINT (-5, -10)

レベル値 -5 dBm ~ -20 dBm 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 = 3 E 6		=		L = CVALUE ( 3 E 6, 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。)

(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 = 2 E 6			=			L = DCVALUE (2 E 6, 5 E 6, 1)
P1 = POINT (F)						
P2 = POINT (5 E 6)						
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 3.5 MHz のレベルを求めます。

```
L = VALUE ( 3.5 E 6, 0 )
```

トレース B の周波数 3 MHz のレベルを求める。

```
F = 3 E 6  
P = POINT ( F )  
L = VALUE ( P, 1 )
```

=
---

```
L = CVALUE ( 3 E 6, 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=DVALUE (P1, P2, 1) | = | L=DCVALUE (2E6, 5E6, 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) P MAX (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】**

P MAX (P 1, P 2, M )

P 1 : 指定ポイント1 (0~700)

P 2 : 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】**

P 1 > P 2 の場合、P 1 と P 2 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 = P MAX (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。

F 1 = 10 E 6

P 1 = P O I N T (F 1)

M P 1 = P M A X (P 1, P O I N T (20 E 6), 1)

M P 2 = P M A X (P O I N T (F 1), P O I N T (20 E 6), 1)

(M P 1 と M P 2 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 ( 3 E 6, 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 = 10 E 6

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(NRPLH)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。

PRPLL, FRPLL, VRPLL のビルトイン関数を使用する前には、必ずこの関数(NRPLL)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) FRPLL N (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】

FRPLL N (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=FRPLL N (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=FRPLL N (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 (D d, S, D s)

D d : 被検データ

S : 基準値

D s : 上下限幅

( (S - D s) ≤ D d ≤ (S + D s) ) 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】

上下限值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

( $Lo \leq Dd \leq Up$ ) 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"
```

上下限值を入力し、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】

上下限值とトレース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

(  $Lo \leq \text{レベル} \leq Up$  ) 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0 dBm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10 dB/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0 dBm以外に設定し、総電力を求める場合は、REFレベルが10 dB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=0 dBm時)

$$AW = \text{POWER}(300, 400, 0)$$

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=-20 dBm時)

$$AW = \text{POWER}(0, 700, 0) / 100$$

(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)

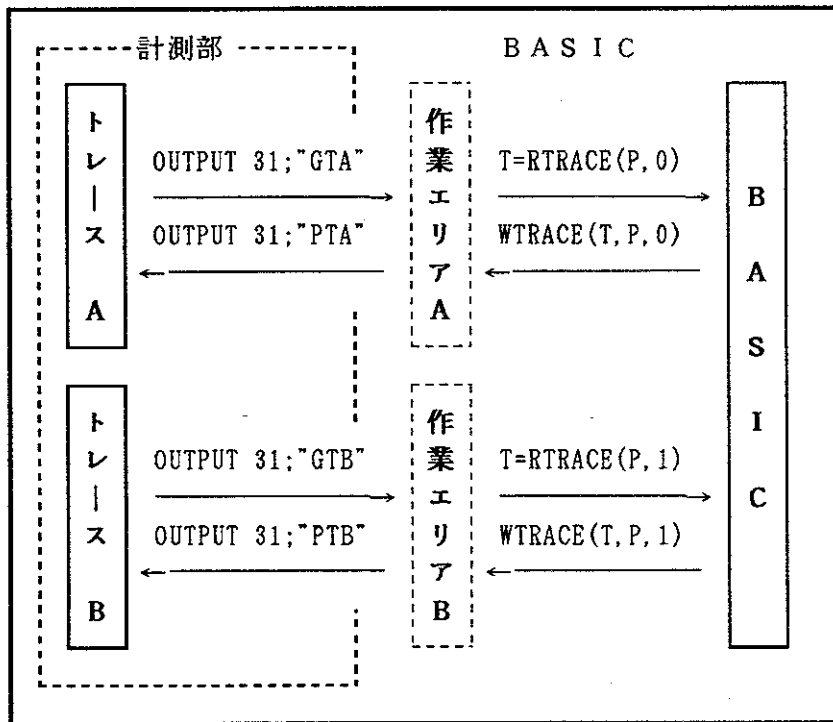
M: トレース 0: トレースA  
1: トレースB

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)

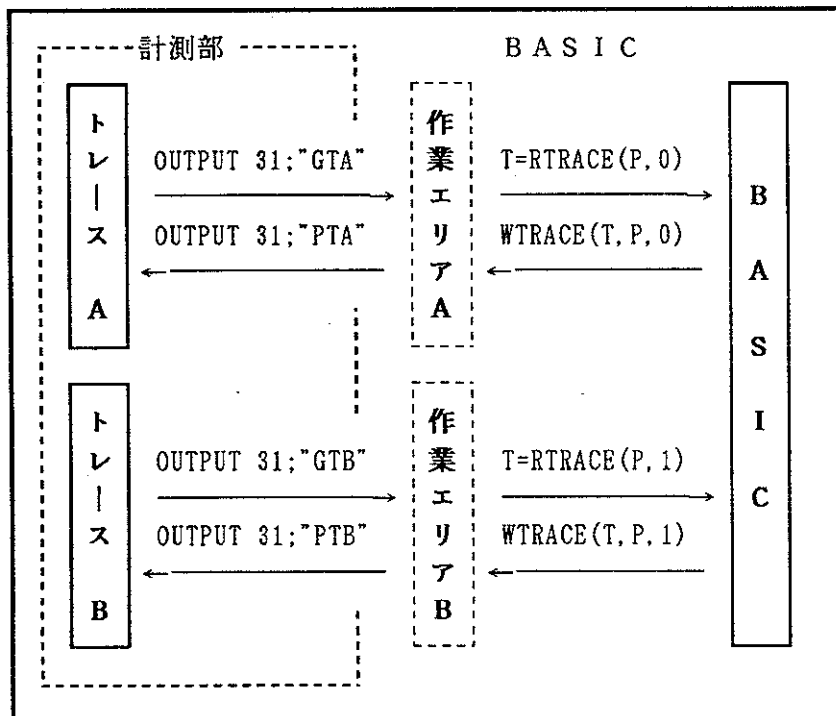
T : トレース・データ (0 ~ 400)

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、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 B 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。

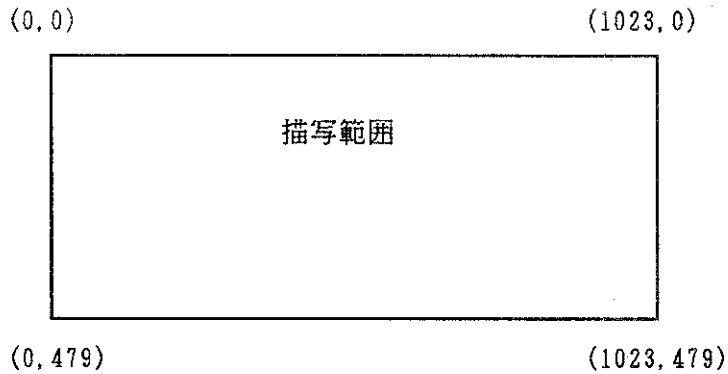
No	関 数	ページ
1	GADRS	3-57
2	GFLRECT	3-59
3	GLINE	3-61
4	GMKR	3-63
5	GPOINT	3-64
6	GRECT	3-65
7	GSTK	3-67
8	GSTYLE	3-68

## (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の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)

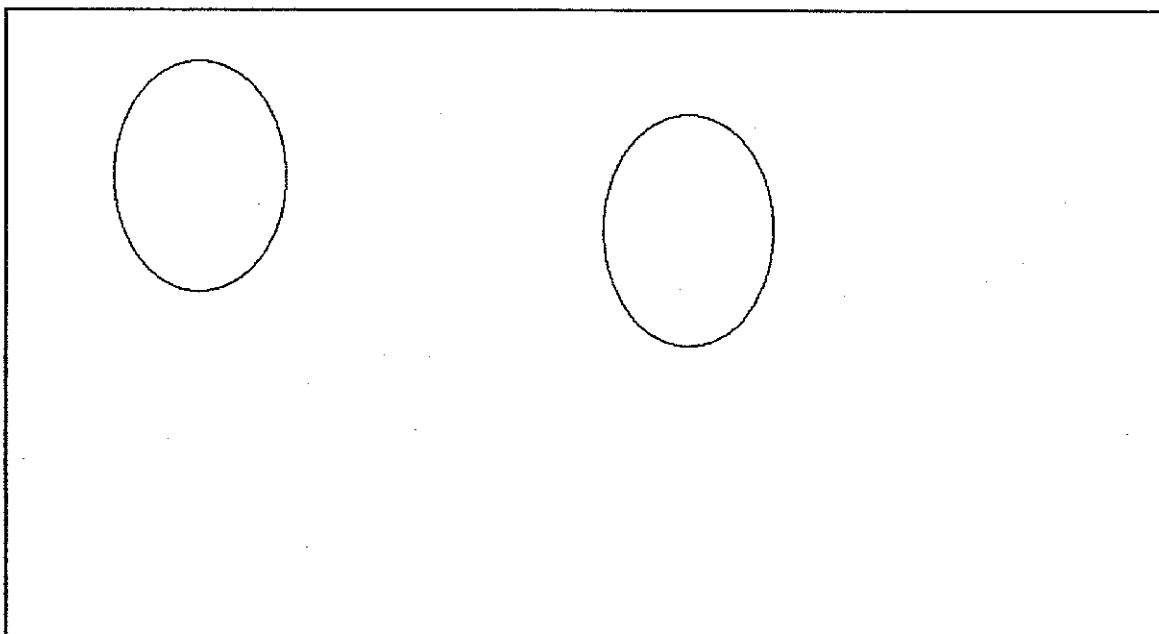
### 【エラー】

- ・ 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(I, 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。

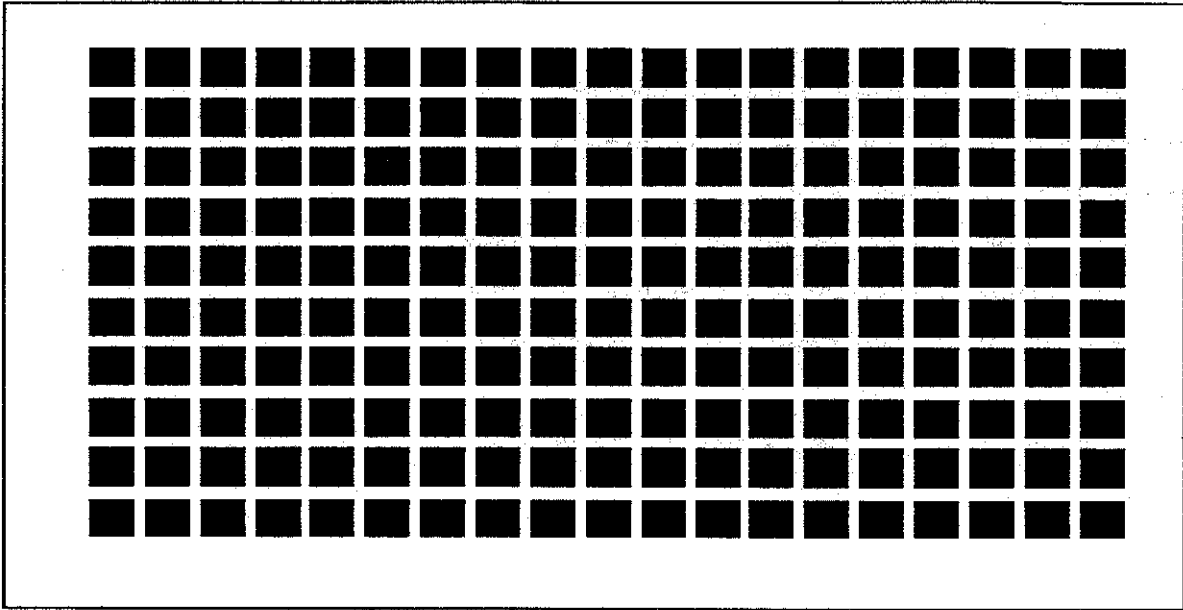
### 【注意】

- ・ 座標1は必ず左上、座標2は右下をなるべく指定して下さい。

### 【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) G L I N E

#### 【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】

G L I N E ( S , D , X 1 , Y 1 , X 2 , Y 2 )

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

X 1 , Y 1 : 座標 1

X 2 , Y 2 : 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。

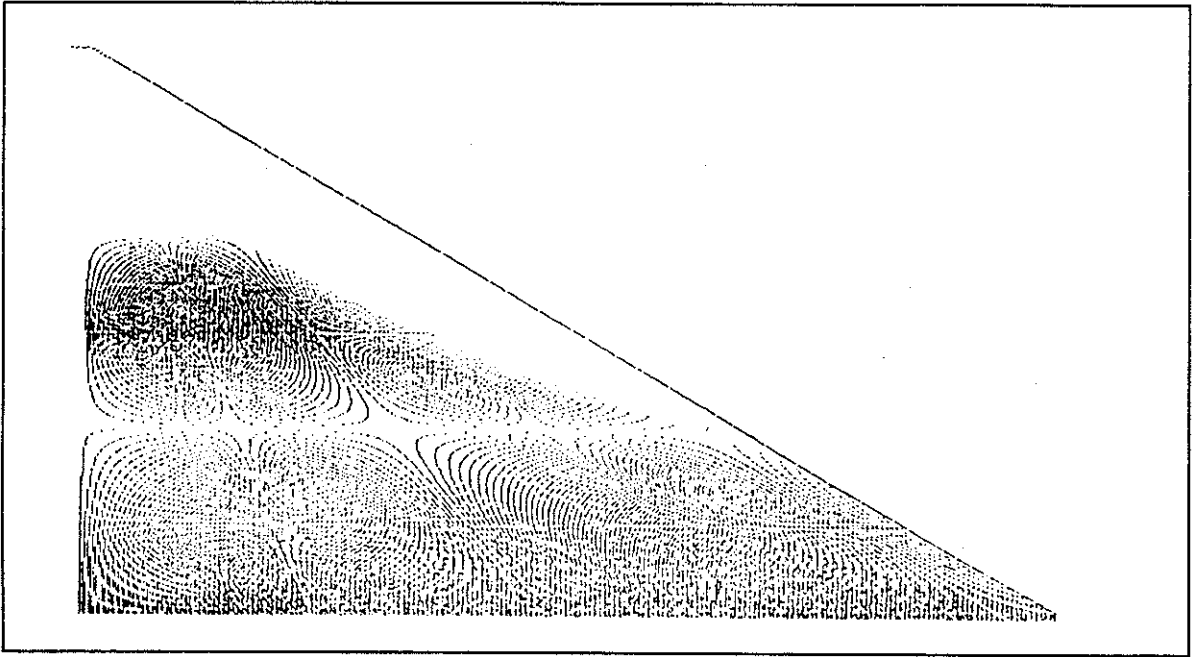
#### 【注意】

- ・ 実線を書く以外は、ライン・スタイル((8) G S T Y L E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、 $\Delta$ ）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 :  $\Delta$  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。

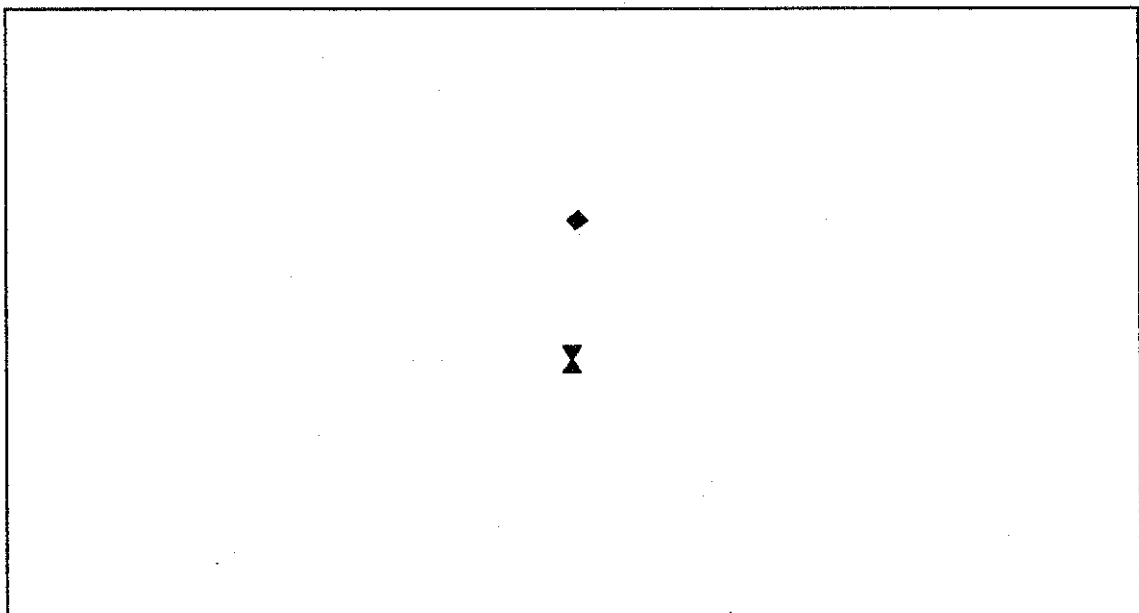
##### 【注意】

- ・ 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

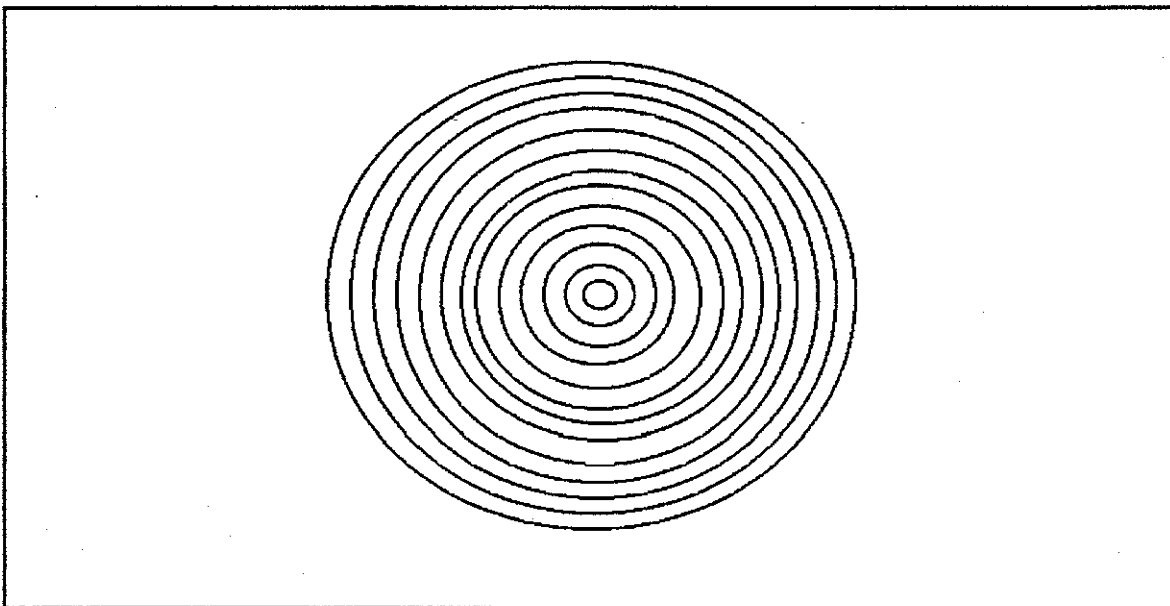
### 【エラー】

- ・ 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。

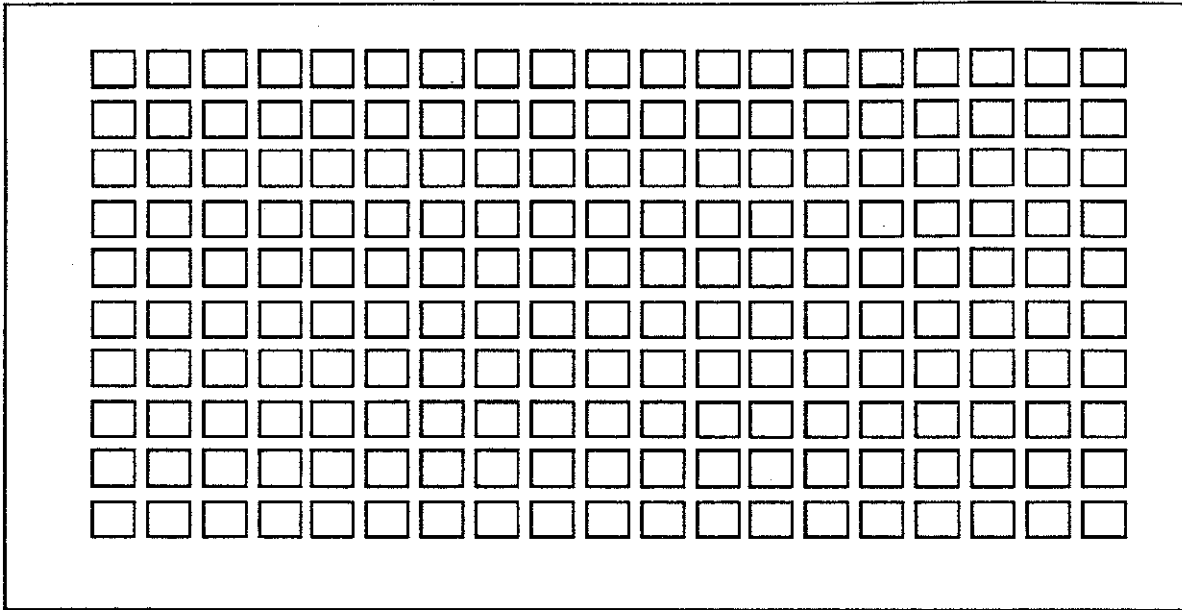
### 【注意】

- ・ 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"  
SS="ADVANTEST"  
FOR I=0 TO 450 STEP 25  
  GSTR(I%2, I, I, SS)  
NEXT I  
STOP
```

### 【実行結果】

```
ADVANTEST  
  ADVANTEST  
    ADVANTEST  
      ADVANTEST  
        ADVANTEST  
          ADVANTEST  
            ADVANTEST  
              ADVANTEST  
                ADVANTEST  
                  ADVANTEST  
                    ADVANTEST  
                      ADVANTEST  
                        ADVANTEST  
                          ADVANTEST  
                            ADVANTEST
```

## (8) G S T Y L E

### 【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。

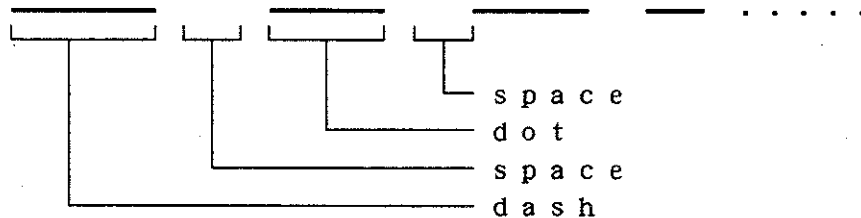
### 【書式】

G S T Y L E ( d a s h ,   s p a c e ,   d o t )

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           : d a s h + s p a c e  
点線           : d o t + s p a c e  
一点鎖線       : d a s h + s p a c e + d o t + s p a c e

例) 一点鎖線を指定のときは以下のようにになります。



### 【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
```







## 4. マスタ / スレーブ・モード

### 4.1 概要

ここでは R 3 2 6 5 A / 3 2 7 1 A とホスト・コンピュータ（外部コントローラ）との接続時の操作方法について説明します。

R 3 2 6 5 A / 3 2 7 1 A とホスト・コンピュータとの接続方法は、以下の 2 通りがあります。

#### CONNECTED HOST

R 3 2 6 5 A / 3 2 7 1 A 背面のスペクトラム・アナライザ側 GPIB ポート（上側）にホスト・コンピュータを接続した場合、このモードを選択します。

このとき、R 3 2 6 5 A / 3 2 7 1 A は本器 BASIC プログラムを実行しながらホスト・コンピュータとデータの送受信を行います。

また、このモードでは R 3 2 6 5 A / 3 2 7 1 A 背面のコントローラ側 GPIB ポート（下側）は、自動的にシステム・コントローラとなり、接続されている GPIB 機器を制御することができます。

#### MASTER/ SLAVE

R 3 2 6 5 A / 3 2 7 1 A 背面のコントローラ側 GPIB ポート（下側）にホスト・コンピュータを接続した場合、このモードを選択します。

このとき、R 3 2 6 5 A / 3 2 7 1 A コントローラ側 GPIB ポート（下側）は、マスタ（コントローラ） / スレーブ（被コントローラ）・モードとなります。

マスタ・モードは、外部に接続している GPIB 機器（ホスト・コンピュータを除く）を本器 BASIC プログラムにより、制御することができます。

スレーブ・モードは、ホスト・コンピュータと本器 BASIC プログラムのデータの送受信を行います。

（ただし、マスタ・モードではホスト・コンピュータ、スレーブ・モードではホスト・コンピュータ以外の GPIB 機器を接続することができないので、本器 BASIC の CONTROL 命令でそれぞれモードを切り換える必要があります。）

それぞれのソフト・メニューは

SHIFT

+

6

+

NEXT  
MENU

と押すと、表示されます。

## 4.2 CONNECTED HOSTモード

### 4.2.1 概要

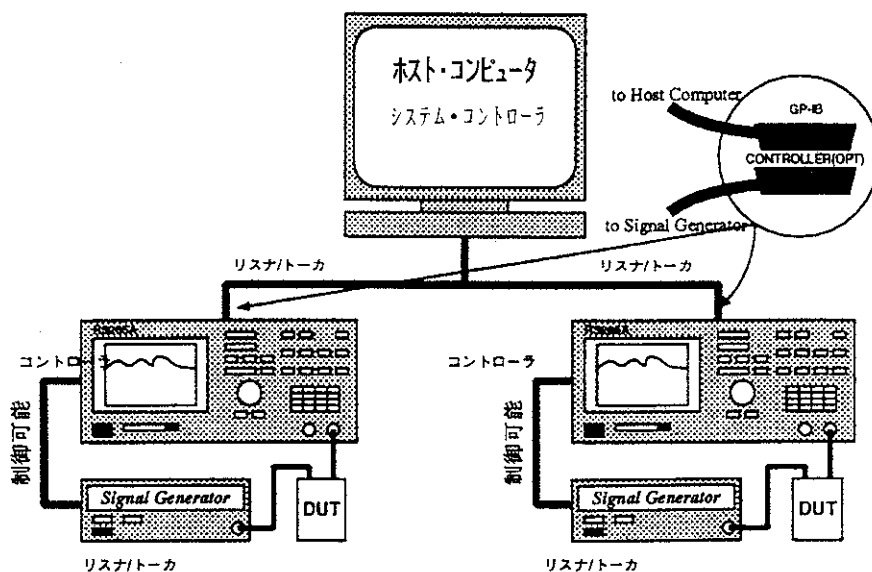


図4-1 外部機器との接続例

上図は

**CONNECTED  
HOST**

モード時の R 3 2 6 5 A / 3 2 7 1 A とホスト・コンピュータ

との接続の様子を示しています。

ホスト・コンピュータを R 3 2 6 5 A / 3 2 7 1 A のスペクトラム・アナライザ側 GPIBポート（上側）と接続することによって、ホスト・コンピュータがシステム・コントローラ、R 3 2 6 5 A / 3 2 7 1 A がリスナ／トーカーとなり、ホスト・コンピュータは R 3 2 6 5 A / 3 2 7 1 A を制御することができます。

また、外部機器（ホスト・コンピュータを除く）をコントローラ側 GPIBポート（下側）に接続することによって、R 3 2 6 5 A / 3 2 7 1 A がコントローラ、信号源（SG）やプロッタ等の外部機器がリスナ／トーカーになり、R 3 2 6 5 A / 3 2 7 1 A コントローラは、それらを制御することができます。

そのため、同一バス上に複数のコントローラが存在することがなく、それぞれ単独でコントローラとして動作することが可能になります。R 3 2 6 5 A / 3 2 7 1 A コントローラで BASIC プログラムを実行させ、そのプログラム内で使用している変数の内容をホスト・コンピュータへ出力したり、ホスト・コンピュータから R 3 2 6 5 A / 3 2 7 1 A コントローラを制御することもでき、拡張性のあるシステムを構築することができます。

さらに、上図のように R 3 2 6 5 A / 3 2 7 1 A を接続しても全く問題なく動作させることができ、それぞれの R 3 2 6 5 A / 3 2 7 1 A で測定した結果をホスト・コンピュータで一括管理することも可能です。

#### 4.2.2 R3265A/3271Aコントローラの制御および変数の出力

表4-1 R3265A/3271Aコントローラ制御コマンド

コマンド名	機能	備考
@LOAD△ファイル名	プログラムのロード	
@RUN	実行	
@STOP	プログラム停止	
@CONT	プログラム再開	
@EXIT	BASIC (エディタ) モードを抜ける	
@CR	改行	INPUT文への入力等に使用
@数値	数値入力	INPUT文への入力等に使用
@CNTRLR	ユーザ定義メニュー表示	
@SF数値	ユーザ定義メニュー実行	
@INFO?	R3265Aコントローラ状態を返す	SRQと併用を推奨
@INFO△CLR	R3265Aコントローラ状態をクリア	
@変数名?	変数値をAscii型でホストへ出力	
@変数名△BIN?	変数値をBinary型でホストへ出力	IEEE64ビット浮動小数点準拠
@変数名△MANY?	配列変数を複数行に渡ってホストへ出力	

△： スペースを表わします。

上表は、ホスト・コンピュータからR3265A/3271Aコントローラを制御する場合のコマンドです。先頭に"@"を付けることによってR3265A/3271Aコントローラ制御用コマンドを認識します。R3265A/3271Aは、"@"付きのコマンドを受けると、その内容を解釈し、該当する処理をR3265A/3271Aコントローラに対して依頼します。

@変数名?コマンドに関しては変数値をホスト・コンピュータに対して送信する必要がありますので、@変数名?コマンドを受けた時点でR3265A/3271Aコントローラに対してデータ要求を起こします(下図では(1),(2))。獲得した変数値は専用バッファへ格納され、その後、ホスト・コンピュータから変数値の要求が発生したら、専用バッファの値をホスト・コンピュータに対して送信します(下図では(3),(4),(5),(6))。

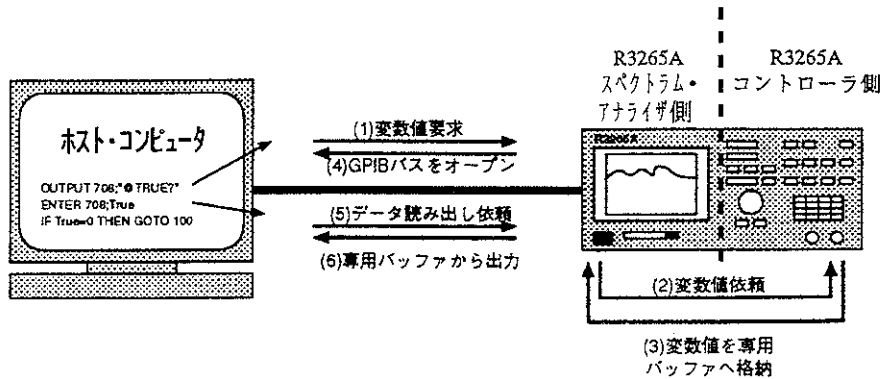


図4-2 R3265A/3271Aコントローラ変数の出力

変数は、Ascii型とBinary型の2通りの形式で読み出すことができます。R3265A/3271Aは、Ascii型の場合には数値をAscii文字に変換し、最後にターミネータを付加して1バイトずつ出力します。このとき、仮数部の有効桁数は15桁です。

また、Binary型の場合には整数値は4バイト、実数値は8バイト(IEEE浮動小数点準拠)でEOI信号をターミネータとして出力します。

読み出すことができる変数は以下の通りで、変数名は最大50文字までです。また、配列変数はすべての値を一括して出力するので、個別に出力することはできません。

1. 文字列変数
2. 整数型変数
3. 実数型変数
4. 整数型変数配列(最大10次元までで1次元あたり1~32767個)
5. 実数型変数配列(最大10次元までで1次元あたり1~32767個)
6. ビルトイン変数

変数値の読み出しに関しては、ホスト・コンピュータとR3265A/3271Aコントローラとの間で同期を取る必要があります。(詳細は、[4.2.3 R3265A/3271Aとホスト・コンピュータとの同期]を参照)

#### 4.2.3 R3265A/3271Aコントローラとホスト・コンピュータとの同期

R3265A/3271Aコントローラとホスト・コンピュータとは、同一バス上に存在しないため、GPIBアドレスによるアクセスができません。

そのため、双方ともR3265A/3271Aのスペクトラム・アナライザ側を經由してデータの送受信を行います。(詳細は[4.2.2 R3265A/3271Aコントローラの制御および変数の出力]を参照)

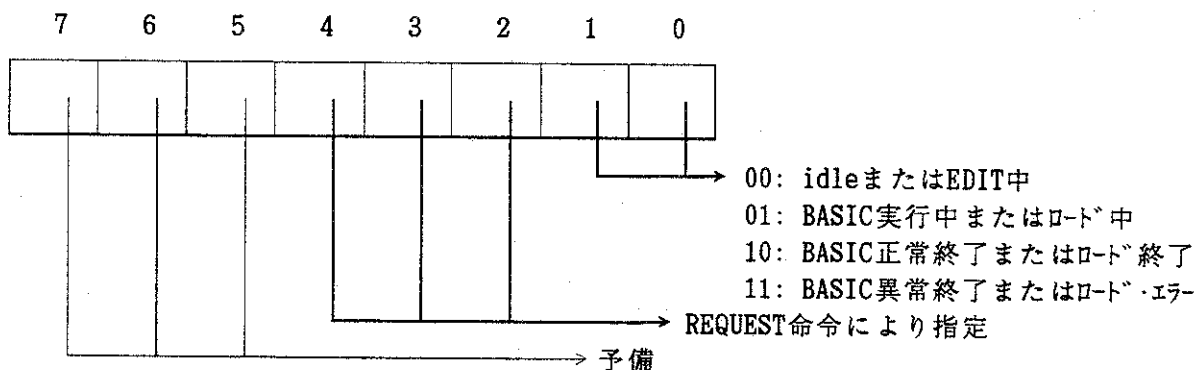


図4-3 R3265A/3271Aコントローラ状態ビット一覧

上図は、R3265A/3271Aコントローラ状態のビット情報一覧です。

双方の同期を取るためにR3265A/3271Aコントローラの状態をビット情報としてホスト・コンピュータに提供します(BASICプログラムではREQUEST命令を使用)。これにより、ホスト・コンピュータはビット情報をポーリングしながらデータ取得の同期を取ることができます。

しかし、単にポーリングだけで情報の収集を行うと、R3265A/3271Aのスペクトラム・アナライザ側の負担が大きくなってしまいますので、通常はSRQ割り込みと併用することを推奨します。

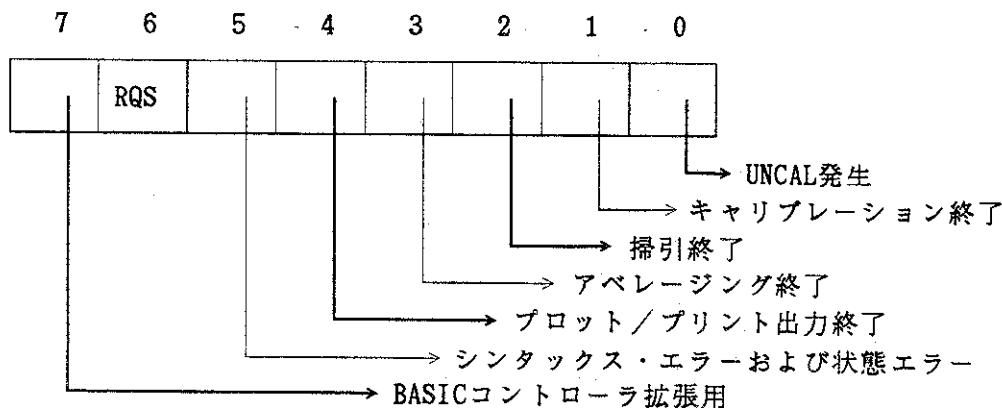


図4-4 R3265A/3271Aスペクトラム・アナライザ側SRQビット一覧

上図は、R3265A/3271Aのスペクトラム・アナライザ側のSRQビット一覧です。このうち、MSBをR3265A/3271Aコントローラ用ビットと定義し、R3265A/3271Aコントローラの状態によって、このビットがON/OFFします。対象となるコントローラ状態は、次の通りです。

1. BASIC正常終了および異常終了  
(BASIC実行中およびidleは対象外)
2. REQUEST命令により指定された場合  
(イベントの終了等BASICプログラムで任意に指定できます)

R3265A/3271Aコントローラ状態のビット情報はホスト・コンピュータ側で  
PRINT @8;"@INFO CLR"と発行することによってクリアできます。

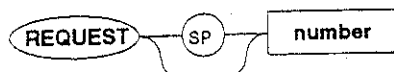
なお、変数を読み出す際には、PRINT @8;"@変数名?"を発行した時点で  
R3265A/3271AがGPIBバスをホールドしてしまい、変数値を確保した時点で  
GPIBバスをリリースするため、変数読み出し許可/禁止情報で改めてポーリングする必  
要はありません。

#### 4.2.4 R3265A/3271Aコントローラ制御コマンド

文中のBASICプログラムは、一部を除いてN-88BASICで記述してあります。

##### (1) REQUEST (Write Information Code of Controller)

###### 【Syntax】



###### 【解説】

REQUESTコマンドは、ユーザ任意のイベントが発生または終了したときに、ホスト・コンピュータへ通知するための手段として@INFO?コマンドと併用して使用します。

なお、このコマンドはR3265A/3271Aコントローラ上で実行するコマンドです。

以下に簡単な同期手段を示します。

1. R3265A/3271Aコントローラは、REQUESTコマンドを実行すると、R3265A/3271Aにイベント発生を通知します。
2. これを受けてR3265A/3271Aは、ホスト・コンピュータに対してSRQ (Service Request) を送出します。
3. ホスト・コンピュータは@INFO?コマンドを発行して、R3265A/3271Aに対してコントローラ状態のビット情報を要求します。
4. R3265A/3271Aは、現在のコントローラ状態 (つまりREQUEST命令) によって指示された値をホスト・コンピュータに返します。

###### 【パラメータ解説】

**number** ユーザ任意のイベントを0~7の範囲で指定します。  
この値は、[図4-3]のR3265A/3271Aコントローラ状態ビットの#2~#4ビットに反映され、@INFO?コマンドによってホスト・コンピュータに読み出されます。

【サンプル・プログラム】

●ホスト側

```

1000  ISET IFC:ISET REN
1010  SPA=8
1020  PRINT @SPA;"S0;RQS128"      ' コントローラからのSRQ使用を宣言
1030  PRINT @SPA;"S2"
1040  POLL SPA,S
1050  ON SRQ GOSUB *READSRQ
1060  SRQ ON                      ' SRQ割り込みを許可
1070  PRINT @SPA;"@RUN"          ' R3265A/3271Aコントローラを起動
1080  EVENT=0
1090  *READXMAX
1100  IF EVENT=0 THEN GOTO *READXMAX ' R3265A/3271Aコントローラの処理終了待ち
1110  PRINT @SPA;"@XMAX?"        ' 変数XMAX読み出しを依頼
1120  INPUT @SPA;XMAX            ' 変数XMAXの読み出し
1130  PRINT XMAX
1140  PRINT @SPA;"@INFO CLR"     ' R3265A/3271Aコントローラ情報をクリア
1150  PRINT @SPA;"@CR"          ' R3265A/3271Aコントローラに読み出し終了を
                                通知

1160  EVENT=0
1170  GOTO *READXMAX
1180  '
1190  *READSRQ
1200  POLL SPA,S
1210  PRINT @SPA;"@INFO?"        ' R3265A/3271Aコントローラ情報を読み出し、
                                処理終了を確認

1220  INPUT @SPA;INFO
1230  IF (INFO AND 4)=4 THEN EVENT=1
1240  SRQ ON:RETURN
1250  END

```

●R 3 2 6 5 A コントローラ側

```

SCREEN 1
*MEAS_START
  OUTPUT 31;'TS'                ! take sweep
  XMAX=MAX(0,700,0)             ! 最大レベル位置を検索
  REQUEST 1                     ! MAX関数実行終了を通知
  INPUT TIMING                  ! ホストからの読み出し終了待ち
  CURSOR 0,0
  GOTO *MEAS_START

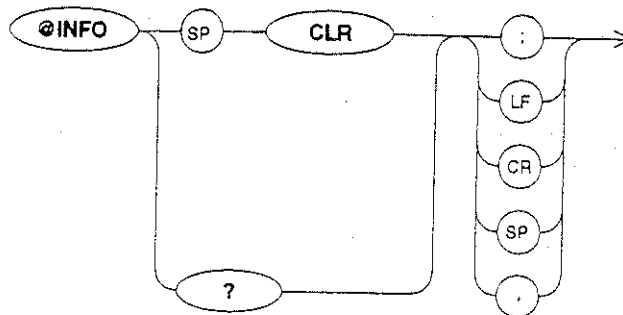
```



(2) @INFO? (@INFO CLR)

(Output Controller Information)

【Syntax】



【Query Response】



【解説】

@INFO? コマンドは、R3265A/3271A コントローラの内部状態をホスト・コンピュータに返します。ビット情報の内容は、下図の通りです。

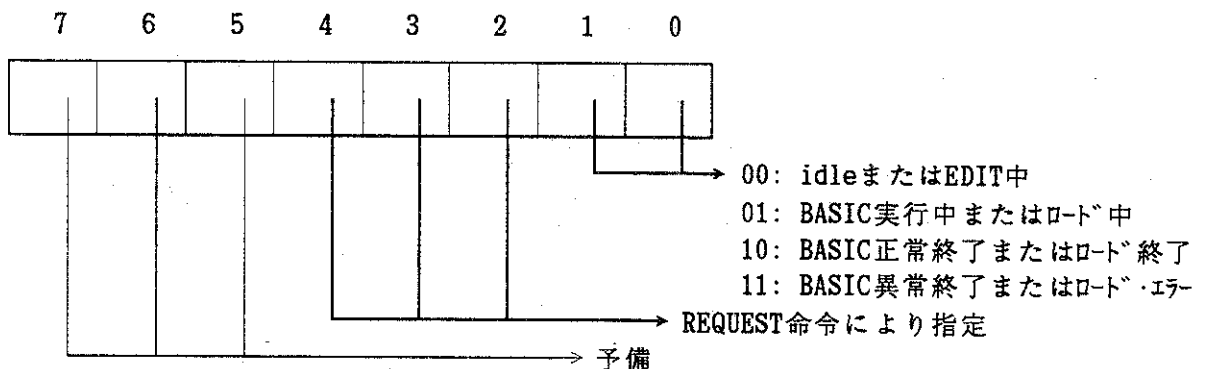


図4-5 R3265A/3271A コントローラ状態ビット一覧

また、@INFO CLR コマンドは、REQUEST 命令によって指定されるビットをクリアします。BASIC 実行状況を示すビット (#0, #1 Bit) は影響されません。

【パラメータ解説】

info R3265A/3271A コントローラの内部状態を返します。

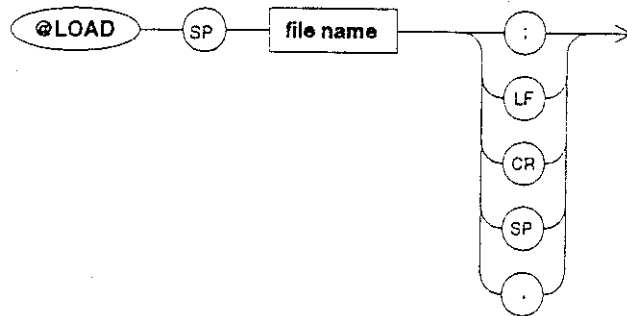
【サンプル・プログラム】

(REQUEST 命令を参照)

(3) @LOAD

(Load Basic Program from IC Card)

【Syntax】



【解説】

@LOADコマンドは、ICカードから指定したBASICプログラムをロードします。

【パラメータ解説】

file name          ロードするBASICプログラム名（ファイル名）を指定します。

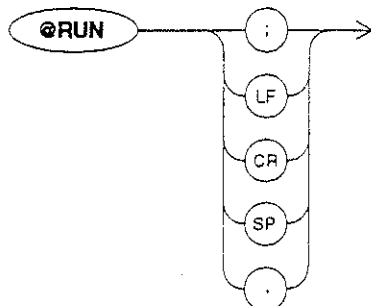
【サンプル・プログラム】

●ホスト側

```
1000 ISET IFC:ISET REN
1010 SPA=8:LOADEND=0
1020 PRINT @SPA;"S0;RQS128"          ' コントローラからのSRQ使用を宣言
1030 PRINT @SPA;"S2"
1040 PRINT @SPA;"@INFO CLR:@INFO?"
1050 PRINT @SPA;"@LOAD XMAX"          ' XMAXファイルをカードからロード依頼
1060 ON SRQ GOSUB *READSRQ
1070 SRQ ON
1080 *WAITEND
1090 IF LOADEND=0 THEN GOTO *WAITEND ' ロード終了待ち
1100 *READINFO
1110 INPUT @SPA;INFO                  ' ロード終了コードは2、エラーは3が返る
1120 IF (INFO AND 3)=3 THEN PRINT "Load Error !!":STOP
1130 IF (INFO AND 3)<>2 THEN GOTO *READINFO
1140 PRINT @SPA;"@RUN"                  ' ロード終了後、実行依頼
1150 STOP
1160 '
1170 *READSRQ
1180 POLL SPA,S
1190 LOADEND=1
1200 RETURN
1210 END
```

(4) @RUN (Execute BASIC Program)  
(@CNTRLR/@CONT/@CR/@EXIT/@STOP/@SF[1-7]/  
@numeric)

【Syntax】



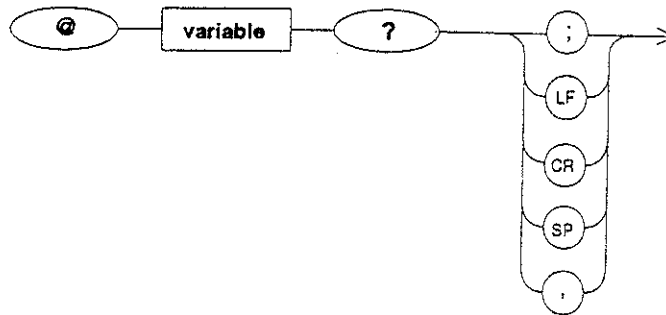
【解説】

- @RUNコマンドは、R3265A/3271Aコントローラ上にロードされたBASICプログラムを実行します。
- @CNTRLRコマンドは、BASIC実行中に限り、ユーザ定義メニューを表示します。
- @CONTコマンドは、停止中のBASICプログラムを継続実行します。
- @CRコマンドおよび@numericは、R3265A/3271Aコントローラ側がINPUT文等で入力待ち状態のときに使用します。
- @EXITコマンドは、コントローラ・モードからスペクトラム・モードに戻ります。
- @STOPコマンドは、実行中のBASICプログラムを停止します。
- @SF [1-7]コマンドは、BASIC実行中に限り、ユーザ定義メニューを実行します。

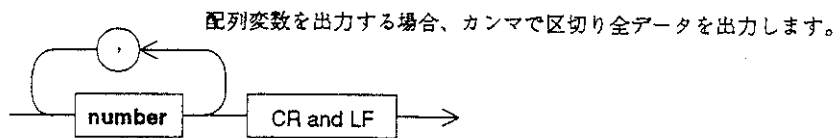
(5) @variable?

(Output Variable)

【Syntax】



【Query Response】



【解説】

@変数名?コマンドは、R3265A/3271Aコントローラで実行したBASICプログラム中の変数値を返します。整数型変数は11桁、実数型変数は23桁（内、仮数部15桁）で出力します。

配列変数はデータ間をカンマで区切り、全データを連続で出力します。

【パラメータ解説】

**variable**      読み出したい変数名を指定します。  
配列変数の場合は、変数名の後に(\*)を付加します。

【サンプル・プログラム】

● ホスト側

```

1000  ISET IFC:ISET REN
1010  DIM ABC(8)
1020  SPA=8
1030  PRINT @SPA;"S0;RQS128"
1040  PRINT @SPA;"S2"
1050  PRINT @SPA;"@RUN"
1060  POLL SPA,S
1070  ON SRQ GOSUB *READSRQ
1080  PRINT @SPA;"@INFO CLR"
1090  EVENT=0
1100  SRQ ON
1110  *READXMAX
1120  IF EVENT=0 THEN GOTO *READXMAX          ' 測定終了待ち
1130  PRINT @SPA;"@ABC(*)?"                  ' 配列変数ABCを読み出し依頼
                                                ' CSV形式データを読み出し
1140  INPUT @SPA;ABC(1),ABC(2),ABC(3),ABC(4),ABC(5),ABC(6),ABC(7),ABC(8)
1150  FOR I=1 TO 8
1160    PRINT ABC(I)
1170  NEXT I
1180  STOP
1190  '
1200  *READSRQ
1210  POLL SPA,S
1220  PRINT @SPA;"@INFO?"
1230  INPUT @SPA;INFO
1240  IF (INFO AND 8)=8 THEN EVENT=1
1250  SRQ ON:RETURN
1260  '
1270  END

```

● R 3 2 6 5 A コントローラ側

```

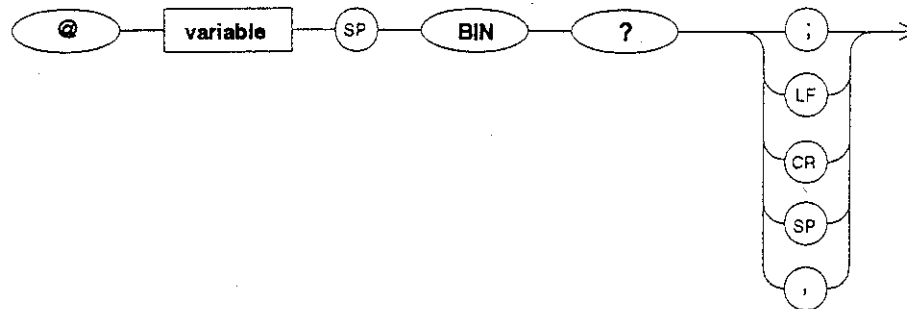
DIM ABC(8)
!
SCREEN 1
FOR I=1 TO 8
  OUTPUT 31;''TS''
  ABC(I)=MAX(0,700,0)
! take sweep
! 最大レベル位置を検索
NEXT I
REQUEST 2
! 実行終了を通知
END

```

(6) @variable BIN?

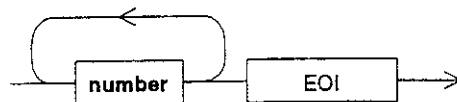
(Output Variable by Binary Format)

【Syntax】



【Query Response】

配列変数を出力する場合、全データを連続で出力します。



【解説】

@変数名BIN?コマンドは、R3265A/3271Aコントローラで実行したBASICプログラム中の変数値をBinary形式で返します。整数型変数は32ビット、実数型変数はIEEE浮動小数点フォーマットの64ビットで出力します。最終バイトを出力する際にEOI信号を付加します。

また、配列変数は全データを連続で出力します。

なお、IEEE64ビット浮動小数点フォーマットを認識できないホスト・コンピュータでは、そのまま使用することができないので、データ読み出し後、内部フォーマットに変換する必要があります。

【パラメータ解説】

variable           読み出したい変数名を指定します。  
配列変数の場合は、変数名の後に(\*)を付加します。

【サンプル・プログラム】

HP-BASICで記述してあります。N88-BASICでは、そのまま使用できません。

●ホスト側

```

1000  OPTION BASE 1
1010  Spa =708
1020  ASSIGN @Device TO Spa;FORMAT OFF    ! IEEE64ビット浮動小数点フォーマットを指定
1030  DIM Abc(8)
1040  !
1050  ON INTR 7 GOSUB Srq_intr
1060  OUTPUT Spa;"S0;S2;RQS128"
1070  S=SPOLL(Spa)
1080  OUTPUT Spa;"@RUN"
1090  Event=0
1100  OUTPUT Spa;"INFO CLR"
1110  ENABLE INTR 7;2
1120  !
1130  Wait_end:!
1140  IF Event=0 THEN GOTO Wait_end        ! 測定終了待ち
1150  OUTPUT Spa;"DL2"                    ! デリミタをEOI指示
1160  OUTPUT Spa;"@ABC(*) BIN?"         ! Binary形式で配列変数を読み出し依頼
1170  ENTER @Device;Abc(*)              ! 配列変数の一括読み出し
1180  PRINT Abc(*)
1190  STOP
1200  !
1210  Srq_intr:!
1220  S=SPOLL(Spa)
1230  OUTPUT Spa;"@INFO?"
1240  ENTER Spa;Info
1250  IF BINAND(Info,8)=8 THEN Event=1
1260  ENABLE INTR 7;2
1270  RETURN
1280  !!
1290  END

```

●R3265Aコントローラ側

```

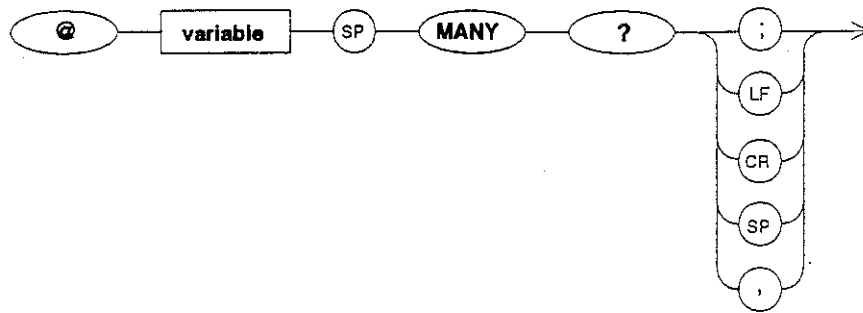
DIM ABC(8)
!
SCREEN 1
FOR I=1 TO 8
  OUTPUT 31;"TS"          ! take sweep
  ABC(I)=MAX(0,700,0)    ! 最大レベル位置を検索
NEXT I
REQUEST 2                ! 実行終了を通知
END

```

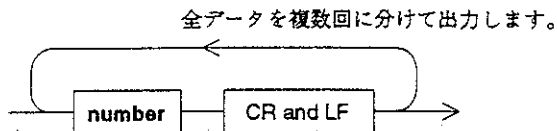
(7) @variable MANY?

(Output Variable by Plural Read)

【Syntax】



【Query Response】



【解説】

@変数名MANY? コマンドは、R 3 2 6 5 A / 3 2 7 1 A コントローラで実行した BASIC プログラム中の配列変数値を複数回に分けて出力します。整数型変数は 1 1 桁、実数型変数は 2 3 桁 (内、仮数部 1 5 桁) で出力し、最後にターミネータを付加します。ホスト・コンピュータ側では FOR 文等を用いて読み出すことになります。

このコマンドは配列変数を出力する場合に限り有効です。

【パラメータ解説】

variable 読み出したい変数名を指定します。  
配列変数のみ有効で、変数名の後に (\*) を付加します。



【サンプル・プログラム】

● ホスト側

```
1000 ISET IFC:ISET REN
1010 DIM ABC(8)
1020 SPA=8
1030 PRINT @SPA;"S0;RQS128"
1040 PRINT @SPA;"S2"
1050 PRINT @SPA;"@RUN"
1060 POLL SPA,S
1070 ON SRQ GOSUB *READSRQ
1080 PRINT @SPA;"@INFO CLR"
1090 EVENT=0
1100 SRQ ON
1110 *READXMAX
1120 IF EVENT=0 THEN FOTO *READXMAX ' 測定終了待ち
1130 PRINT @SPA;"@ABC(*) MANY?" ' 配列変数ABCを複数回に渡って読み出し依頼
                                   ' 1データごとに読み出し
1140 FOR I=1 TO 8
1150   INPUT @SPA;ABC(I)
1160   PRINT ABC(I)
1170 NEXT I
1180 STOP
1190 '
1200 *READSRQ
1210 POLL SPA,S
1220 PRINT @SPA;"@INFO?"
1230 INPUT @SPA;INFO
1240 IF (INFO AND 8)=8 THAN EVENT=1
1250 SRQ ON:RETURN
1260 '
1270 END
```

● R 3 2 6 5 A コントローラ側

```
DIM ABC(8)
!
SCREEN 1
FOR I=1 TO 8
  OUTPUT 31;"TS" ' take sweep
  ABC(I)=MAX(0,700,0) ' 最大レベル位置を検索
NEXT I
REQUEST 2 ' 実行終了を通知
END
```

### 4.3 MASTER/SLAVEモード

#### 4.3.1 概要

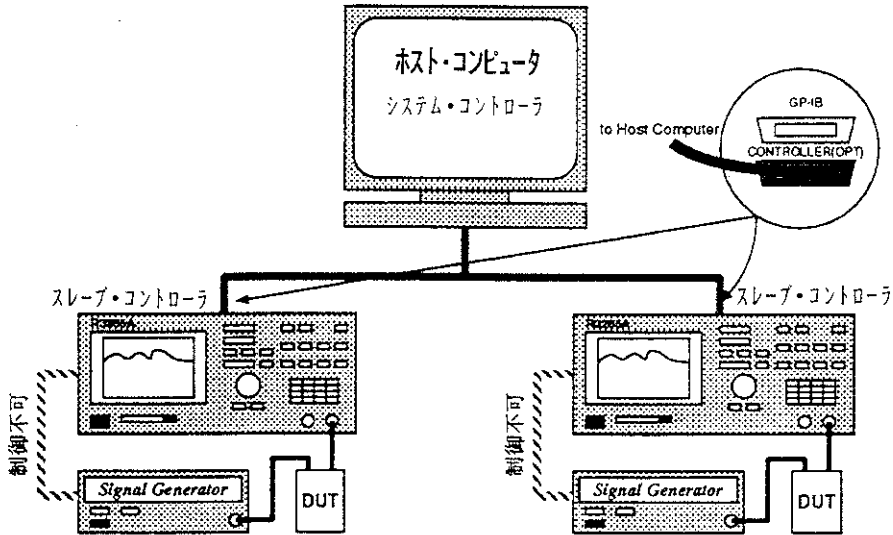


図4-6 外部機器との接続例

上図は **MASTER/SLAVE** モード時の R3265A/3271A とホスト・コンピュータとの

接続の様子を示しています。

ホスト・コンピュータと R3265A/3271A コントローラとの関係は、ホスト・コンピュータを R3265A/3271A のコントローラ側 GPIB ポート（下側）と接続することにより、ホスト・コンピュータがシステム・コントローラ、R3265A/3271A がスレーブ・コントローラとなっています。

（マスタ/スレーブの切り換えは **MASTER/SLAVE** ソフトキーまたは CONTROL 命令にて

行います。）

そのため、同一バス上に複数のコントローラが存在することになるので、R3265A/3271A コントローラは、被コントローラとして動作します。

したがって、R3265A/3271A コントローラは、BASIC プログラムで使用している変数値等は専用コマンドによってホスト・コンピュータと送受信できますが、ホスト・コンピュータ以外の GPIB 機器を制御することはできません。信号源 (SG) やプロッタ等を接続して動作させることができないので、これらの制御はホスト・コンピュータが行います。

また、上図のように複数の R3265A/3271A を接続してもコントローラとして動作できるのは、1台のみです。

#### 4.3.2 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コマンド	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	指定装置をトーカーに指定し、 GPIBポートからASCII I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 Iデータを出力します。	外部コントローラからトーカーの 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。

### 4.3.3 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ー指定を受けたとき、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

注)

CONNECTED  
HOST

モード選択時と使用方法が異なります。



#### 4.3.4 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 R 3 2 6 5 A / 3 2 7 1 A コントローラ機能の 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"	COMPILE & 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、R 3 2 6 5 A / 3 2 7 1 A 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、R 3 2 6 5 A / 3 2 7 1 A 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
```

R 3 2 6 5 A / 3 2 7 1 A (スレーブ・モード アドレス9)

```
ENTER 0;AS  
PRINT AS  
*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R 3 2 6 5 A / 3 2 7 1 A側のBASICプログラムが起動します。

R 3 2 6 5 A / 3 2 7 1 A側プログラムのENTER 0;AS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。

(AS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、R 3 2 6 5 A / 3 2 7 1 A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R 3 2 6 5 A / 3 2 7 1 Aをトーカーに指定するとR 3 2 6 5 A / 3 2 7 1 A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 SRQ ON
1130 WHILE F
1140 WEND
1150 GOTO *L
1160 *INTR
1170 POLL SPA,RSV
1180 IF RSV-64 THEN INPUT @SPA;LV:PRINT "FAIL LEVEL=" ;LV ELSE PRINT "PASS"
1190 F=0
1200 SRQ ON:RETURN
```

プログラムの説明

1000~1030	初期設定
1040	R 3 2 6 5 A / 3 2 7 1 A 側の BASIC プログラムを起動します。
1050	R 3 2 6 5 A / 3 2 7 1 A との同期をとるために通信します。
1070~1080	基準値、許容値を設定します。
1090~1100	基準値、許容値を R 3 2 6 5 A / 3 2 7 1 A へ送信します。
1110~1150	サービス・リクエスト割り込み待ちループ
1160	サービス・リクエスト割り込み処理
1170	シリアル・ポール
1180	シリアル・ポール値が 6 4 ならば PASS、6 4 以外ならば FAIL として レベル値を R 3 2 6 5 A / 3 2 7 1 A から受信し表示します。
1190	ループフラグ OFF
1200	割り込み許可:リターン



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=LMTMD1(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	



## 5. レジューム機能

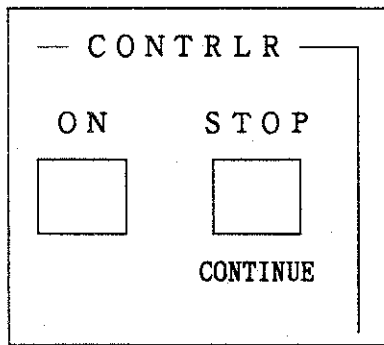
BASICプログラムの実行を行うと、コンパイル後のプログラムはバックアップメモリに蓄えられます。プログラム実行終了後、電源を切断してもコンパイル後のプログラムはメモリ内に保持され、再度電源を投入にしたときも即座にプログラムの実行が可能になっています。

### 注意

プログラム実行中に電源を切断しますと、電源投入後にプログラムの実行はできません。プログラム停止を確認の上、電源を切断して下さい。



## 6. コントローラ・キー



本体パネル右上部

ON



コントローラのBASICエディタをオープンします。  
また、BASIC実行中にはユーザ定義メニューを表示します。

STOP



BASICの実行を停止、または再実行させます。

CONTINUE



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 .....	A-1
A.2	機能別ビルトイン関数とグラフィック機能一覧 .....	A-4
A.3	パラメータ指定一覧 .....	A-6
A.4	エラー・メッセージ一覧 .....	A-9



機 能	コマンドとステートメント	内 容
①コマンド	CONT CONTROL LIST LLIST LISTN LLISTN RUN SCRATCH STEP	プログラム停止後の再実行を行う。 各制御に関する値を設定する。 プログラム・リストを出力する。 プログラム・リストを出力する。(RS-232C) プログラム・リストを出力する。 プログラム・リストを出力する。(RS-232C) プログラムを実行させる。 BASIC内のプログラムを消去する。 プログラムを1行実行する。
②算術関数	ABS ATN COS LOG SIN SQR TAN	与えられた値の絶対値を求める。 与えられた値の逆正接値を求める。 与えられた値の余弦値を求める。 与えられた値の自然対数を求める。 与えられた値の正弦値を求める。 与えられた値の平方根を求める。 与えられた値の正接値を求める。
③ビット演算	BAND BNOT BOR BXOR	ビットANDを求める。 ビットNOTを求める。 ビットORを求める。 ビットXORを求める。
④割り込み 制 御	ENABLE INTR DISABLE INTR ON END ON KEY ON ISRQ ON SRQ ON ERROR OFF END OFF KEY OFF ISRQ  OFF SRQ OFF ERROR	割り込み受信許可状態にする。 割り込み受信禁止状態にする。 EOF割り込みの分岐を定義する。 キー割り込みの分岐を定義する。 本体計測部SRQ割り込みの分岐を定義する。 GPIB、SRQ割り込みの分岐を定義する。 エラー発生割り込みの分岐を定義する。 EOF割り込みの分岐定義を解除する。 キー割り込みの分岐定義を解除する。 本体計測部SRQ割り込みの分岐定義を解除する。 GPIB、SRQ割り込みの分岐定義を解除する。 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  CHRS LEN POS SPRINTF	文字列の先頭の文字のASCIIコードを求める。 数値をASCII文字に変換する。 文字列の長さを求める。 文字列2の中から文字列1がある位置を求める。 書式を決め文字列変数に代入する。

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 CALL CLOSE # ENTER # OPEN # OUTPUT # INITIALIZE (INIT) PURGE RENAME	メモリ・カードの内容を表示する。 サブ・プログラムをロードする。 ファイルを閉じる。 ファイルからデータを読み込む。 ファイルに開く。 ファイルにデータを書き込む。 メモリ・カードを初期化する。 指定ファイルを消去する。 ファイル名を変更する。
⑦画面制御	CURSOR (CSR) CLS	指定位置にカーソルを移動する。 画面を消去する。
⑧ステートメント	BUZZER DIM FOR TO STEP NEXT BREAK CONTINUE GOSUB RETURN GOTO IF THEN ELSE END IF INPUT (INP) INTEGER LPRINT LPRINT USING (USE) PAUSE PRINT (?) PRINT USING (USE) PRINTER PRINTF (PRF) READ DATA  RESTORE REM (!) SELECT CASE END SELECT STOP WAIT	ブザーを鳴らす。 配列変数を宣言する。 繰り返し処理を設定する。 繰り返し処理から抜ける。 繰り返し処理を先頭に戻す。 サブ・ルーチン分岐する。 サブ・ルーチンからの復帰する。 指定位置へ分岐する。 条件判断をして処理する。 変数への入力を行う。 整数型変数の宣言する。 プリンタ (RS232C) 出力する。 プリンタ (RS232C) 出力する。(書式指定) 一時停止する。 文字を画面に出力する。 文字を画面に出力する。(書式指定) GPIBプリンタ装置のアドレス指定を行う。 文字を画面に出力する。(書式指定) DATA文からデータを読み取り、変数へ入力する。 READ文で読むDATA文を指定する。 注釈文 条件判断を行い処理する。 プログラム実行停止する。 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 DELIMITER ENTER (ENT) GLIST  GLISTN  GPRINT GPRINT USING (USE)  INTERFACE CLEAR LOCAL LOCAL LOCKOUT OUTPUT (OUT)  REMOTE REQUEST SEND SPOLL TRIGGER	DCL、SDCを送出する。 デリミタを設定する。 GPIBデータを入力する。(パラレルI/O含む) GPIBプリンタへプログラム・リストを出力する。 GPIBプリンタへプログラム・リストを出力する。 データをGPIBプリンタへ出力する。 データをGPIBプリンタへ出力する。 (書式指定) IFCを送出する。 指定装置をローカル状態にする。 指定装置をローカル・ロックアウト状態にする。 データをGPIBに出力する。 (パラレルI/O含む) 指定装置をリモート状態にする。 標準GPIBにSRQを出力する。 GPIBデータを1つずつ出力する。 指定装置のシリアル・ポールを行う。 GETを送出する。
⑩R3265A/ 3271A コントローラ制御 コマンド	@LOAD△ファイル名 @RUN @STOP @CONT @EXIT @CR @数値 @CNTRLR @SF数値 @INFO?  @INFO△CLR @変数名? @変数名△BIN?  @変数名△MANY?	プログラムのロード。 実行。 プログラム停止。 プログラム再開。 BASIC (エディタ) モードを抜ける。 改行。INPUT文への入力等に使用。 数値入力。INPUT文への入力等に使用。 ユーザ定義メニュー表示。 ユーザ定義メニュー実行。 R3265Aコントローラ状態を返す。 SRQと併用を推奨。 R3265Aコントローラ状態をクリア。 変数値をAscii型でホストへ出力。 変数値をBinary型でホストへ出力。 IEEE64ビット浮動小数点準拠。 配列変数を複数行に渡ってホストへ出力。

△： スペースを表わします。

## 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	PRPLL N	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	FRPLL N	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	VRPLL N	左からN番目の極小点のレベルを求める
33	RPL 1	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	LMTMD 1	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	LMTMD 2	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	LMTUL 1	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	LMTUL 2	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

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



F : 周波数	Dx : 横軸微分係数
P : ポイント (0 ~ 700)	Dy : 縦軸微分係数
L : レベル	S : 基準値
T : トレース・データ (0 ~ 400)	Ds : 基準値幅
M : トレースA / B (0 / 1)	Dd : 被検データ
X : lossレベル	Lo : 下限値
C : チェック値 (0, 1, 2, -1)	Up : 上限値
N : リップル数	
W : ワット (電力)	



#### 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 split 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

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文のカウント変数がない
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	x x x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	x x x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	x x x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間がない
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	P O K E 命令を実行しようとした
yy is invalid value in xxx	y y 行目にある x x x の値が無効である
yy: Undefined Control Register	C O N T R O L 命令のレジスタ番号が誤っている
yy: UNIT addr error in xxx	G P I B アドレス指定が誤っている





## 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)コールセンターにご連絡下さい。

## 製品修理サービス

- 製品修理期間  
製品の修理サービス期間は、製品の納入後10年間とさせていただきます。
- 製品修理活動  
当社の製品に故障が発生した場合、当社に送っていただく引取り修理、または当社技術員が現地に出張しての出張修理にて対応いたします。

## 製品校正サービス

- 校正サービス  
ご使用中の製品に対し、品質および信頼性の維持を図ることを目的に行うもので、校正後の製品には校正ラベルを貼付けし、品質を保証いたします。
- 校正サービス活動  
校正サービス活動は、株式会社アドバンテスト 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実施することで、製品の安定稼働を図り、不意の費用発生を防ぐため、年間保守契約による予防保守の実施をお勧めいたします。

なお、年間保守契約は、製品、使用状況および使用環境により内容が変わりますので、最寄りの弊社営業支店にお問い合わせ下さい。

# ADVANTEST®

<http://www.advantest.co.jp>

### 株式会社アドバンテスト

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)コールセンター ☎ TEL 0120-919-570  
FAX 0120-057-508  
E-mail: [icc@acs.advantest.co.jp](mailto:icc@acs.advantest.co.jp)