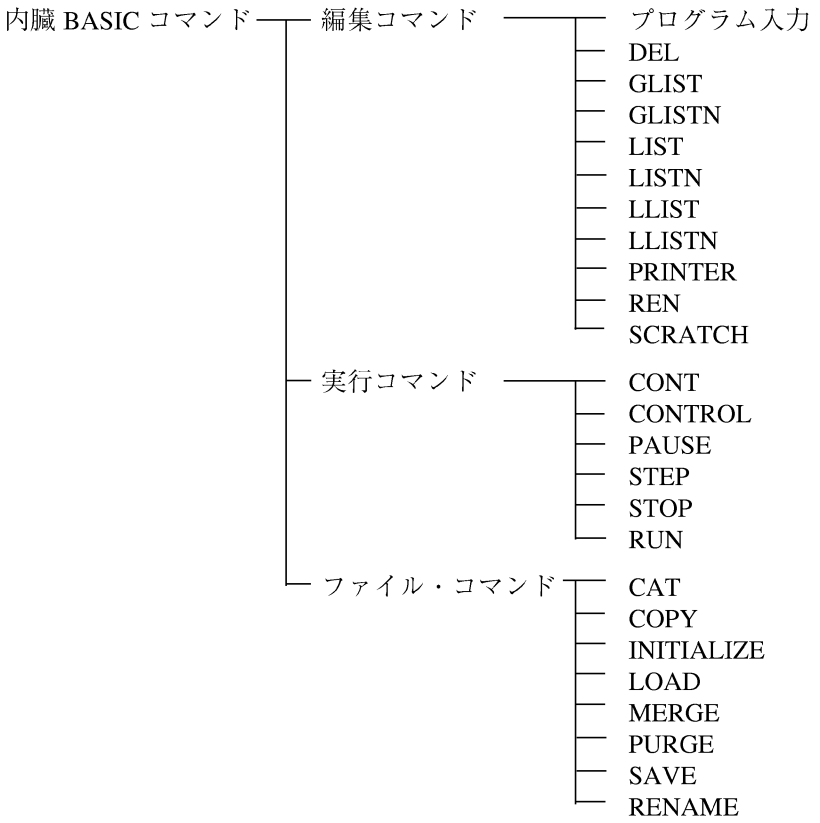
3. BASIC コマンド

3.1 各種コマンド

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



（注）これらのコマンドは、大文字で使用します。

3.1.1　コマンド機能一覧

コマンド		機能
編集コマンド	プログラム入力	ステートメントをプログラムとして格納
	DEL	指定行番号を削除
	GLIST	プログラム・リストを GPIB に出力
	GLISTN	プログラム・リストを GPIB に出力
	LIST	プログラム・リストをディスプレイ上に表示
	LISTN	プログラム・リストをディスプレイ上に表示
	LLIST	プログラム・リストをシリアル・ポートに出力
	LLISTN	プログラム・リストをシリアル・ポートに出力
	PRINTER	プリンタの GPIB アドレスを設定
	REN	行番号の変更
	SCRATCH	すでに入力されているプログラムを消去
実行コマンド	CONT	プログラム実行を再開
	CONTROL	BASIC のコントロール変数を設定（環境設定）
	PAUSE	プログラム実行を一時停止（CONT 可）
	STEP	プログラムを一行実行
	STOP	プログラム実行を停止（CONT 不可）
	RUN	プログラムを実行
ファイル・コマンド	CAT	フロッピー・ディスク内のファイル名をディスプレイ上に表示
	COPY	フロッピー・ディスク内のファイルを複写
	INITIALIZE	フロッピー・ディスクを初期化
	LOAD	プログラムをフロッピー・ディスクからロード（呼び出す）
	MERGE	すでに入力済のプログラムに追加する形で、フロッピー・ディスクからロード（呼び出す）
	PURGE	フロッピー・ディスク内のファイルを削除
	SAVE	プログラムをフロッピー・ディスクへセーブ（格納）
	RENAME	フロッピー・ディスク内のファイル名を変更

3.1.2　コマンド文法一覧

コマンド		文法
編集コマンド	プログラム入力	行番号ステートメント
	DEL	DEL 開始行〔,終了行〕
	GLIST	GLIST〔開始行〕〔,〔終了行〕〕
	GLISTN	GLISTN〔開始行〕〔,〔行数〕〕
	LIST	LIST〔開始行〕〔,〔終了行〕〕
	LISTN	LISTN〔開始行〕〔,〔行数〕〕
	LLIST	LLIST〔開始行〕〔,〔終了行〕〕
	LLISTN	LLISTN〔開始行〕〔,〔行数〕〕
	PRINTER	PRINTER 装置アドレス
	REN	REN〔〔旧行番号〕〔,＜新行番号＞〔,＜増分値＞〕〕〕
	SCRATCH	SCRATCH〔112〕
実行コマンド	CONT	CONT〔行番号〕
	CONTROL	CONTROL＜レジスタ番号＞;＜値＞
	PAUSE	PAUSE
	STEP	STEP〔行番号〕
	STOP	STOP
	RUN	RUN〔行番号〕
ファイル・コマンド	CAT	CAT〔"DATE"〕
	COPY	COPY "旧ファイル名","新ファイル名"
	INITIALIZE	INITIALIZE〔"ボリューム・ラベル"〕＜タイプ＞
	LOAD	LOAD "ファイル名"
	MERGE	MERGE "ファイル名"
	PURGE	PURGE "ファイル名"
	SAVE	SAVE "ファイル名"
	RENAME	RENAME "旧ファイル名","新ファイル名"

3.1.3 コマンドの共通注意事項

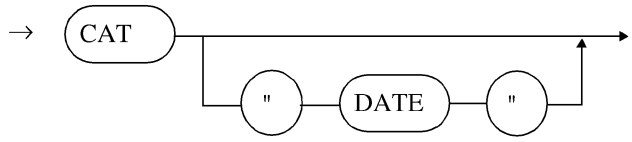
内部 BASIC コマンドには、以下に示す共通の注意事項があります。

1. パラメータ
   コマンドのパラメータには、文字列表現式または数値表現式が指定できます。つまり、BASIC で使用する変数が使えます。ここで、実数の場合は小数点以下が切り捨てられます。
   しかし各コマンドの解説では、見やすさのため整数、文字列などの表現を使っています。
2. 式の境界
   原則として BASIC コマンドは、式を複数個続けて指定する場合、その式の境界が構文上解釈できれば、カンマ (,) はスペースにしても構いません。
3. 行番号
   行番号の設定範囲は 1 ～ 65535 です。
   0 またはプログラムの先頭行よりも小さい値を指定した場合、プログラムの先頭行を指定したと解釈されます。
   65535 またはプログラムの最終行よりも大きい値を指定した場合、プログラムの最終行を指定したと解釈されます。

3.2　コマンド文法と活用

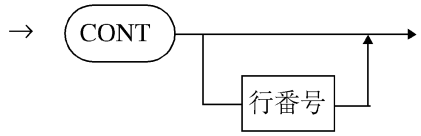
3.2 節のコマンド・インデックスを以下に示します。

1. プログラム入力
   3 章と 4 章に記載するコマンドとステートメントは、行番号を付けるとプログラムとして入力できます。
   入力済のプログラムに同一の行番号が存在する場合は、置き換えとなります。また、存在しない場合は、追加または挿入となります。
2. CAT
   - 概要　カレント・ドライブに記憶されたファイルを表示します。
   - 構文　(1)-1



   (1)-2
   CAT〔"DATE"〕
   - 解説　カレント・ドライブに記憶されたファイルとディレクトリをリスト表示します。
     CAT の場合　：左側から登録番号、ファイル名、使用バイト数、ファイルの属性が表示される。
     CAT "DATE" の場合　：左側から登録番号、ファイル名、作成日時が表示される。

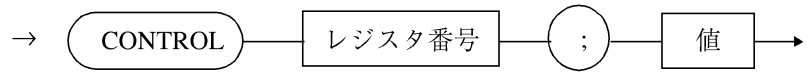
     ※ ファイルの取り扱いについては、2.4 ファイルの管理を参照して下さい。
3. CONT
   - 概要　BASIC プログラムの実行を再開させます。
   - 構文　(1)-1



   (1)-2
   CONT〔行番号〕
   - 解説　・PAUSE で停止した BASIC プログラムを、停止している行の次から再開できます。
     ・BASIC プログラムを指定した行から再開させます。
     変数の初期化はしません。
   - 例　CONT
     CONT 200

4. CONTROL

- 概要　BASIC のコントロールに関係する細かな部分の値を設定します（環境設定）。
- 構文　(1)-1



(1)-2

CONTROL レジスタ番号 ; 値

- 解説　・レジスタ番号で設定するコントロール対象を指定し、セミコロン（;）の後の値が実際の設定値です。
・レジスタ番号には、1 ～ 9 の値を設定します。各レジスタの内容を以下に示します。（ただし、レジスタ 4 は内部の都合上未登録です。）

< レジスタ 1> ... 初期値 79

シリアル I/O ポートの設定です。以下の値の和で指定します。

下線のある値は、電源投入時に設定されている値です。

1. ボーレート	:	0	; 1200 ボー	3. パリティ	:	0	; なし
		1	; 2400 ボー			16	; 奇数
		2	; 4800 ボー			48	; 偶数
		3	; 9600 ボー				
2. キャラクタ長	:	0	; 5 ビット	4. ストップ・ビット数	:	0	; なし
		4	; 6 ビット			64	; 1 ビット
		8	; 7 ビット			128	; 11/2 ビット
		12	; 8 ビット			192	; 2 ビット

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

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

< レジスタ 2> ... 初期値 0

LLIST または GLIST で左端からの印字位置をスペースの数で指定します。

（例）リスト出力を右に 5 文字分寄せる場合

CONTROL 2 ; 5 を実行してから LLIST または GLIST を実行すると、行番号の前にスペースが 5 つ入り、その後にリストが表示されます。

< レジスタ 3> ... 初期値 0

BASIC プログラムをフルネーム表示にするか、ショートネーム表示にするかを指定します。

0: フルネーム表示にする。

1: ショートネーム表示にする。

フルネームとショートの対応表は、表 4-2 を参照して下さい。

< レジスタ 5> ... 初期値 0

メンテナンス用コマンドの POKE を有効にするか、無効にするか指定します。

0: 無効

1: 有効

< レジスタ 6> ... 初期値 0

INPUT 文の終了条件を指定します。

0: **[ENT]** が押されたときのみ終了する。

1: **[ENT]** かファンクション・キーが押されたときに終了する。

< レジスタ 7> ... 初期値 0

GPIB 関係の設定です。以下の値は各々設定します。

0: GPIB モードを ADDRESSABLE モードに設定する。

1: GPIB モードを SYSTEM CONTROLLLER モードに設定する。

2: REQUEST CONTROL（コントローラ権の要求）を送信する。

4: BASIC 実行中に、外部コントローラからの GPIB コマンド設定を有効にする。

< レジスタ 8> ... 初期値 0

DMA 転送モードの ON/OFF を設定します。

0: OFF

1: ON

< レジスタ 9> ... 初期値 1

PRINT の出力先の指定です。以下に示す値の和で設定します。

1: デフォルト出力（各機種の正面パネル表示器）

2: メンテナンス用ポート（端末）への出力

4: 外部モニタおよび R3753 液晶ディスプレイへの出力

8: R3752 用の蛍光表示管への出力

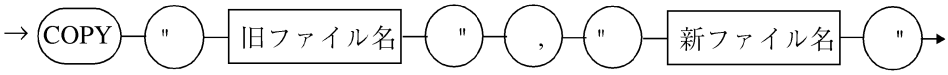
（例 1）デフォルト出力およびメンテナンス用ポートに出力したい場合

```
CONTROL 9 ; 3
```

（例 2）デフォルト出力、メンテナンス用ポートおよび外部モニタへ出力したい場合
CONTROL 9 ; 7

5. COPY

- 概要　ドライブに登録されたファイルを複写します。
- 構文　(1)-1



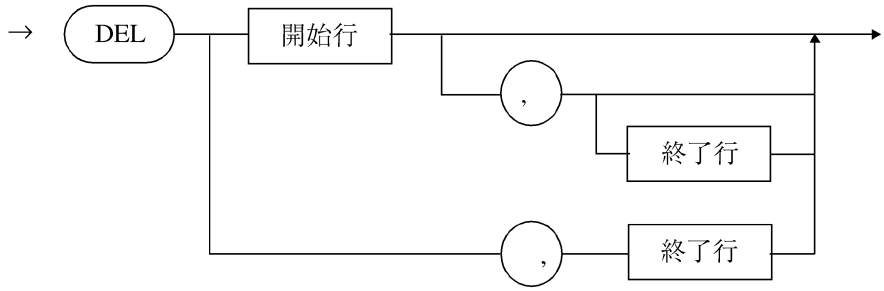
(1)-2
COPY " 旧ファイル名 ", " 新ファイル名 "

- 解説　・旧ファイル名の内容を新ファイル名に複写します。
・新ファイル名が既に存在する場合、旧ファイル名の内容が上書きされます。
・新ファイル名と旧ファイル名が同じ場合はエラーとなります。
・2 つのファイル名とも文字列表現式を使って指定できます。

※ ファイルの取り扱いについては、2.4 ファイルの管理を参照して下さい。

6. DEL

- 概要　プログラム中の行を削除します。
- 構文　(1)-1



(1)-2
DEL< 開始行〔,〔終了行〕> | <, 終了行 >

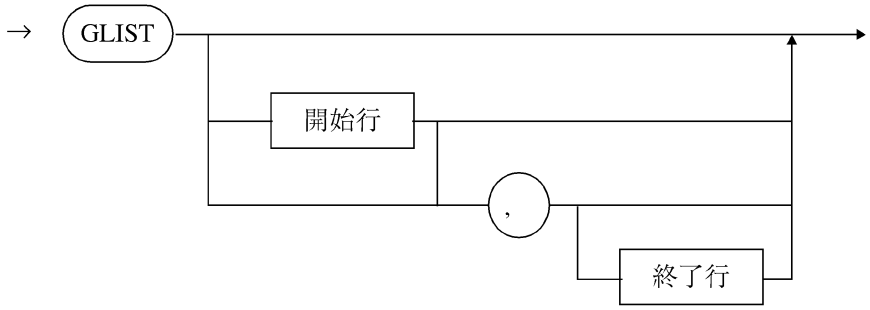
注　カンマ（,）をスペースにしても構いません。
行番号の設定範囲は 1 ～ 65535 です。

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

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

7. GLIST

- 概要　GPIB 経由でプリンタ等にプログラム・リストを出力します。
- 構文　(1)-1



(1)-2
GLIST〔開始行〔,〔終了行〕〕｜〔,〔終了行〕〕

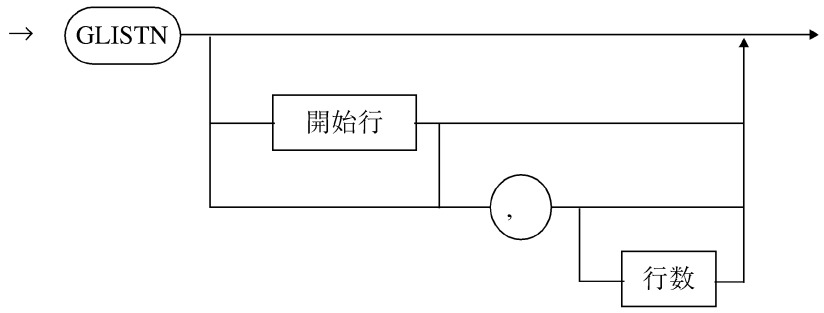
注　カンマ（,）をスペースにしても構いません。
行番号の設定範囲は 1 ～ 65535 です。

- 解説　・ GPIB に接続されたプリンタ等に、BASIC プログラム・リストを出力します。
・ プリンタの GPIB アドレスは、PRINTER 文で、または本器のパネル操作で設定します。
- 例

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

8. GLISTN

- 概要　GPIB 経由でプリンタ等にプログラム・リストを出力します。
- 構文　(1)-1



(1)-2

GLISTN〔開始行〔,〔行数〕〕｜〔,〔行数〕〕

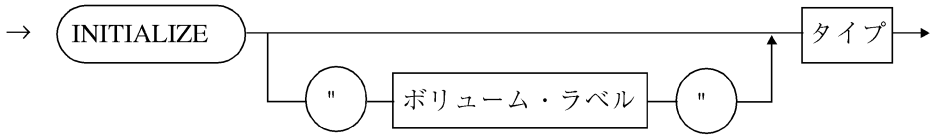
注　カンマ (,) はスペースにしても構いません。
行番号の設定範囲は 1 ～ 65535 です。

- 解説
  - ・ GPIB に接続されたプリンタ等に、BASIC プログラム・リストを出力します。
  - ・ プリンタの GPIB アドレスは、PRINTER 文で、または本器のパネル操作で設定します。
  - ・ 開始行で指定した行番号から行数で指定した行数分のプログラム・リストを出力します。
  - ・ 行数が負の数の場合、行番号の小さい方に向かって指定行数がとられます。
- 例

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

9. INITIALIZE (INIT)

- 概要　フロッピー・ディスクを初期化します。
- 構文　(1)-1



(1)-2

INITIALIZE〔" ボリューム・ラベル "〕タイプ

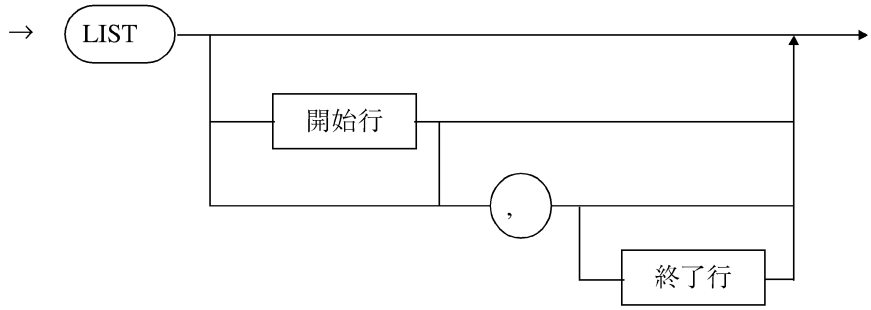
- 解説　・新品のフロッピー・ディスクや、転用したいフロッピー・ディスクを、タイプで指定したフォーマットで初期化します。
・初期化時には、ボリューム・ラベルを指定できます。
省略すると、ボリューム・ラベルなしになります。
・タイプは、以下のように指定して下さい。
タイプ指定:　0；720KB（512 バイト、9 セクタ）2DD
1；1.2MB（1024 バイト、8 セクタ）2HD
2；1.4MB（512 バイト、18 セクタ）2HD
3；1.2MB（512 バイト、15 セクタ）2HD

注意　本器は、2DD/2HD を自動認識します。よって、上記のタイプ指定が、挿入されているフロッピー・ディスクに適合しない場合は、以下のデフォルト設定で初期化します。
デフォルト設定：2DD の場合；720KB（タイプ 0）
2HD の場合；1.2MB（タイプ 1）

※ ファイルの取り扱いについては、2.4 ファイルの管理を参照して下さい。

10. LIST

- 概要　ディスプレイ上にプログラム・リストを表示します。
- 構文　(1)-1



(1)-2

LIST〔開始行〔,〔終了行〕〕〕｜〔,〔終了行〕〕〕

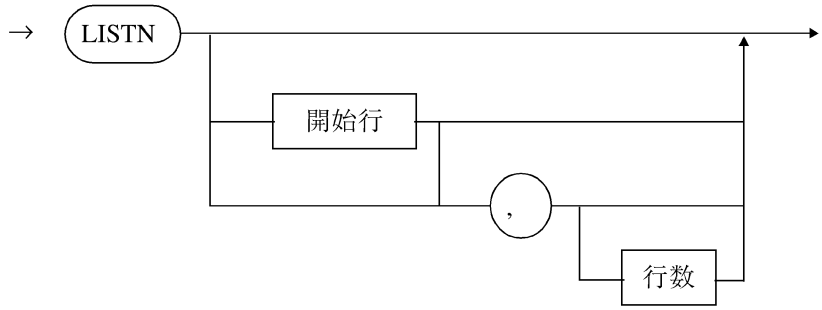
注	カンマ (,) をスペースにしても構いません。 行番号の設定範囲は 1 ～ 65535 です。

- 解説
  - ・ディスプレイ上にパラメータで指定した範囲の BASIC プログラム･リストを表示します。
  - ・STOP キーで表示を中止できます。しかし、プログラムの実行と異なるため、中断した行位置から再開できません。
- 例

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

11. LISTN

- 概要　ディスプレイ上にプログラム・リストを表示します。
- 構文　(1)-1



(1)-2

LISTN〔開始行〔,〔行数〕〕｜〔,〔行数〕〕

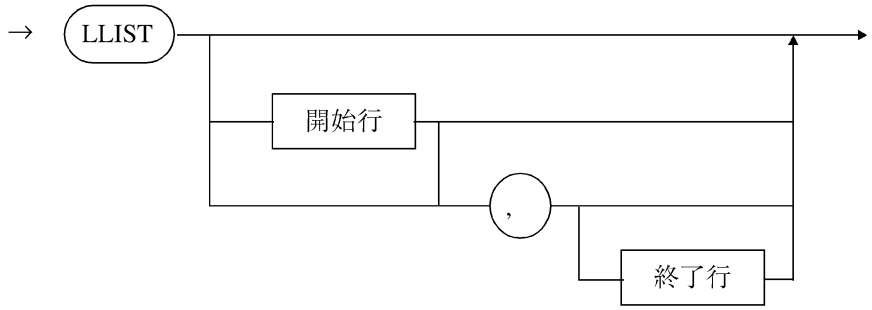
注　カンマ (,) はスペースにしても構いません。
行番号の設定範囲は 1 ～ 65535 です。

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

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

12. LLIST

- 概要　シリアル・ポート経由でプリンタ等にプログラム・リストを出力します。
- 構文　(1)-1



(1)-2
LLIST〔開始行〔,〔終了行〕〕〕｜〔,〔終了行〕〕

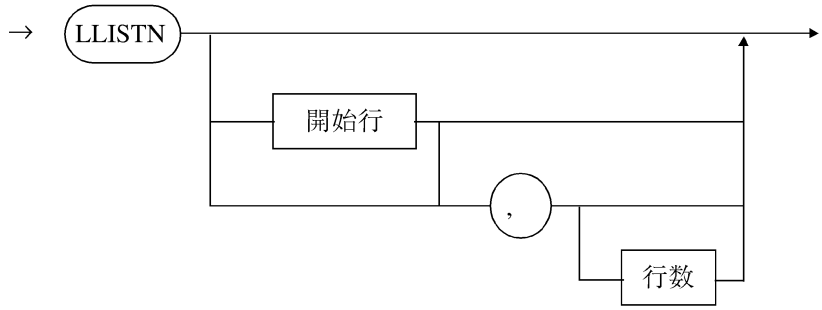
注	カンマ (,) をスペースにしても構いません。 行番号の設定範囲は 1 ～ 65535 です。

- 解説　シリアル・ポートに接続されたプリンタ等に、BASIC プログラム・リストを出力します。
- 例

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

13. LLISTN

- 概要　シリアル・ポート経由でプリンタ等にプログラム・リストを出力します。
- 構文　(1)-1



(1)-2

LLISTN〔開始行〔,〔行数〕〕〕｜〔,〔行数〕〕

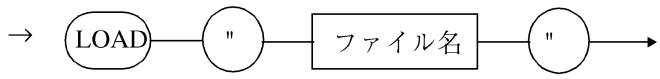
注　　行番号の設定範囲は１～ 65535 です。

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

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

14. LOAD

- 概要　ドライブに登録されたファイルを呼び出します。
- 構文　(1)-1



(1)-2

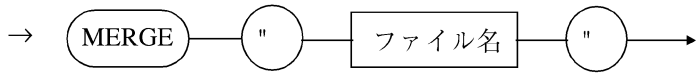
LOAD " ファイル名 "

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

※ ファイルの取り扱いについては、2.4 ファイルの管理を参照して下さい。

15. MERGE

- 概要　ドライブに登録されたファイルを呼び出します。
- 構文　(1)-1



(1)-2

MERGE " ファイル名 "

- 解説　・ LOAD と異なり、ロードする前に BASIC バッファの初期化は行いません。
  ・ すでにBASICバッファに存在するプログラムは行番号が一致しない限り削除されません。
  ・ SCRATCH と MERGE の組み合わせが LOAD の機能と同様になります。

※ ファイルの取り扱いについては、2.4 ファイルの管理を参照して下さい。

16. PAUSE

- 概要　プログラムの実行を一時停止させます。
- 構文　(1)-1

(1)-2
PAUSE

- 解説　・BASIC プログラムの実行を一時的に停止、または BASIC プログラム自身で一時的に停止させます。
・停止した次の行から、CONT によりプログラムを継続できます。
- 例

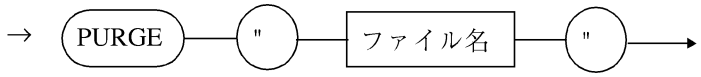
```
10    FOR I=1 TO 9
20        GOTO 60
30        GOTO *PRT
40    NEXT I
50    PAUSE
60    !
70    X = I * I
80    GOTO 30
90    *PRT
100   PRINT I;"*";I;"=";X
110   GOTO 40
```

17. PRINTER

4.3 節の 44. PRINTER を参照して下さい。

18. PURGE

- 概要　ドライブに登録されているファイルを消去します。
- 構文　(1)-1



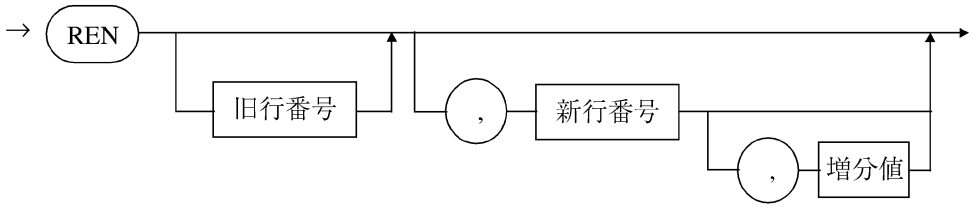
(1)-2
PURGE " ファイル名 "

- 解説　既存のファイルで、不要なものを消去します。

※ ファイルの取り扱いについては、2.4 ファイルの管理を参照して下さい。

19. REN

- 概要　プログラムの各行の行番号を付け直します。
- 構文　(1)-1



(1)-2

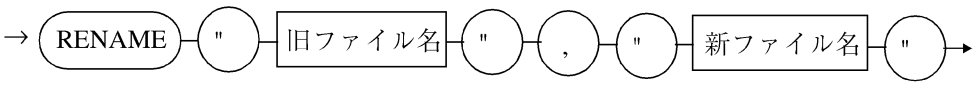
REN〔〔旧行番号〕〔, 新行番号〔, 増分値〕〕〕

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

- 解説
  - ・旧行番号は、番号の付け替えが始められる現在のプログラム中の行番号です。
  - ・新行番号は、新しく付ける先頭の行番号です。
  - ・増分値は、新しく付ける行番号の増分量を示します。
  - ・REN は、GOTO, GOSUB などで使っている行番号にも、新しい行番号に対応して変更します。
  - ・REN は、65535 を超える行番号を使えません。また、プログラム行の順序を変えるような指定をしてはいけません。
- 例
  - REN　：プログラムの先頭行を 10 にして、10 ステップで最後まで行番号を付け直す
  - REN 30, 50, 3　：行番号30を50にして3ステップで最後まで行番号を付け直す

20. RENAME

- 概要　ドライブに登録されているファイル名を変更します。
- 構文　(1)-1



(1)-2

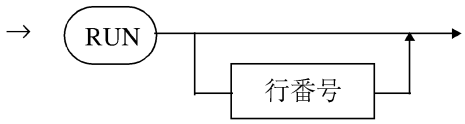
RENAME " 旧ファイル名 "," 新ファイル名 "

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

※ ファイルの取り扱いについては、2.4 ファイルの管理を参照して下さい。

21. RUN

- 概要　BASIC プログラムを実行させます。
- 構文　(1)-1



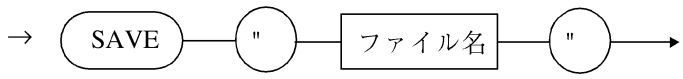
(1)-2

RUN〔行番号〕

- 解説　・ BASIC プログラムを指定した行から実行させます。
  ・ 行番号を指定しないときは、先頭行から実行します。
  ・ RUN を実行すると、プログラムを実行する前にすべての変数をクリアし、配列宣言なども強制的に無設定状態になります。
- 例　RUN
  RUN 200

22. SAVE

- 概要　ドライブにファイルを登録します。
- 構文　(1)-1

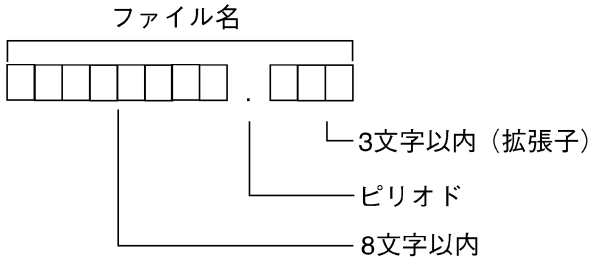


(1)-2

SAVE " ファイル名 "

- 解説　・編集したプログラム（行番号の付いている文の先頭から最後まで）をファイル名で指定した名前でファイルに登録します。
・既存のファイル名を指定すると、同一ファイルの更新とみなされ、その内容が更新されます。

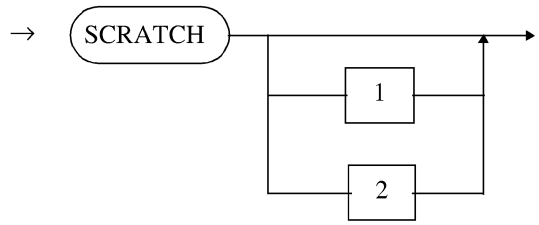
注意　ファイル名は、数字、英字および記号（ダブル・クォーテーション (") は除く）を使い、以下のように指定します。



※ ファイルの取り扱いについては、2.4 ファイルの管理を参照して下さい。

23. SCRATCH

- 概要　メモリに蓄えられたBASICプログラムを消去します。
- 構文　(1)-1



(1)-2

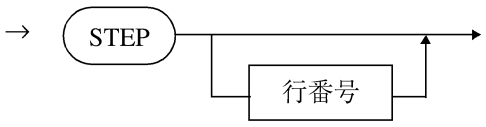
SCRATCH〔1|2〕

- 例

SCRATCH	BASICバッファ内のプログラムをすべて消去する。（データ・プロシージャ共に）
SCRATCH 1	BASICバッファ内のプログラムのデータのみ初期化する。
SCRATCH 2	BASIC バッファ内のプログラムのプロシージャのみ初期化する。

24. STEP

- 概要　BASICプログラムを一行のみ実行させます。
- 構文　(1)-1



(1)-2

STEP〔行番号〕

- 解説　・BASICプログラムを指定した一行のみ実行しますが、FOR文の中では実行しません。
  ・行番号の指定がない場合は、直前に終了した次の行を実行します。
- 例　STEP
  STEP 100

25. STOP

- 概要　BASIC プログラムの実行を停止させます。
- 構文　(1)-1

(1)-2
STOP

- 解説　BASIC プログラムの実行を停止、または BASIC プログラム自身で停止させます。

4. BASIC ステートメント

4.1 プログラミングのきまり

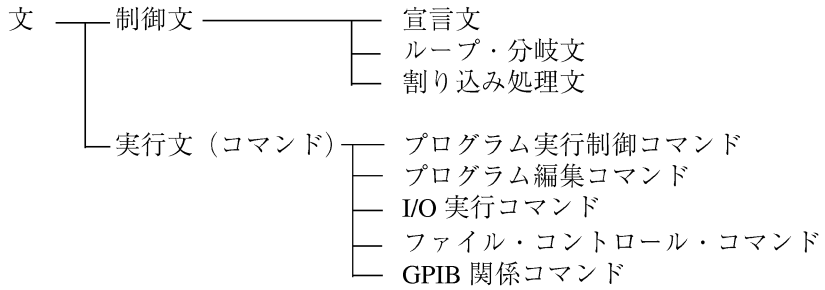
4.1.1 プログラム構造

1. 文

BASIC プログラムは各種の文の集まりです。

文は大きく分けて、制御文と実行文（コマンド）によって構成されています。

各文は、キー・ワードと式から構成されます。この構成を取り決めたものが文法の構文規則です。



2. キー・ワード

BASIC で、あらかじめ意味と用途を定めている言葉を「キー・ワード」と呼びます。

キー・ワードと同じ名前を、その他の目的に使用できません。

キー・ワードの名前（フルネーム）で、使用頻度が高く、かつ長い名前のものにはショート・ネームがあります。

表示をフルネームからショート･ネームに変更するには、CONTROL コマンドでコントロール･レジスタ 3 を 1 にします。フルネームに戻すには、コントロール･レジスタ 3 を 0 にします。

キー・ワード一覧表は、表 4-1 を参照して下さい。

フルネームとショート・ネームの対応表は、表 4-2 を参照して下さい。

表 4-1　キー・ワード一覧表

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

注：キー・ワードは、大文字で使用します。

表 4-2　フルネームとショートネームの対応表

フルネーム	ショートネーム
CURSOR	CSR
ENTER	ENT
INITIALIZE	INIT
INPUT	INP
OUTPUT	OUT
PRINTF	PRF
USING	USE
PRINT	?

3. 式

式は、オブジェクトと演算子からなり、文法上、式を指定できるところならば、どこにでも置くことができます。（ただし、従来の BASIC との互換性を考えて、IF 文の条件式では = を等号と解釈するため、代入式を書くことはできません。）

式には、演算結果の最終値がどのデータ型を取るかによって、以下の 4 通りあります。

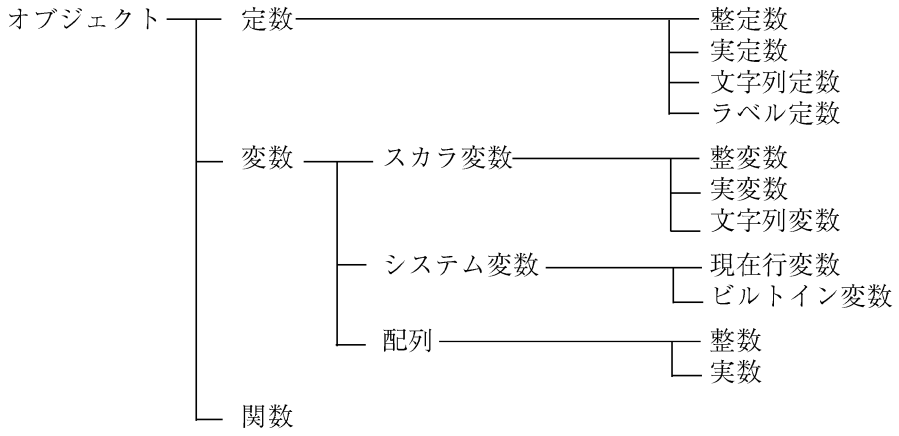
< 算術式 > < 文字列式 > < 論理式 > < ラベル >

算術式 : 整数値または実数値になる。

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

4.1.2 オブジェクト

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



1. 定数
   - 整定数

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

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

     文字列を表現するには、その文字列をダブル・クォーテーション (") で囲みます。

     文字列は " " の空文字列から最大 128 文字まで指定できます。構成する文字の単位は 8 ビットで、0 ～ 255 までの 256 種類の文字単位を表現できます。

     文字コードは ASCII を使用し、128 ～ 255 までは特殊なシンボルが登録されています。

     コードがキー・ボードに割り当てられていないものをプログラムで表現するために、または INPUT 文に対する入力のために、\ を使用して \f（フォーム・フィールド）という方法があります。同様にして、ダブル・クォーテーション (") を文字列に含めるために \" と書くことができます。

     ASCII の制御文字を表現するために、以下のエスケープ・シーケンスが用意されています。

表 4-3　エスケープ・シーケンス

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

- ラベル定数

  文番号に変わるものです。プログラムの先頭でアスタリスク (*) を付けて宣言します。

  使用できる文字は変数と同じですが、変数ではないので代入できません。またラベルが書ける位置は構文的に限られていて、4.3 ステートメント文法と活用で説明するラベル行番号、または分岐先と書かれている部分です。

2. 変数

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

変数名の最後が $ の場合　　：文字列変数になる。

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

表 4-4　アルファニューメリック

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

（例）変数の型

value, v123　　：実変数

string$, s123$　　：文字列変数

array (3)　　：配列型実変数

INTEGER code　　：整変数

INTEGER week(7)　　：配列型整変数

- スカラ変数
  - 整変数
  - 実変数
  - 文字列変数

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

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

    宣言をせずに参照すると、18 文字の文字列であるとみなされます。
    文字列はサブ・ストリング演算子 ([ ]) を使用して、文字列の一部を扱うことができます。
    4.1.3 項の 7. サブ・ストリング演算子を参照して下さい。

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

    結果

    ```
    ADVANTEST CORP.
    ```

- 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。値は代入できません。

    LIST@ : 現在実行している行を表示

  - 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、特定の値を代入することによって変更できます。起動時の値に戻すには明示的にその値を代入するか、SCRATCH 1, SCRACH で初期化します。

    PI　　：3.14159.....

    EXP　：2.71828.....

- 配列

  配列の宣言は、DIM, INTEGER 文を使用します。

  ・ 数値型配列

  宣言せずに参照すると、その配列の大きさ、つまり要素数は 10 になります。以下のように宣言したものと同じになります。添字は必ず 1 から割当られます。

```
DIM       array (10)
INTEGER   array (10)
```

実数型配列	DIM	real (20)
整数型配列	INTEGER	int (30, 40)

3. 関数

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

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

結果

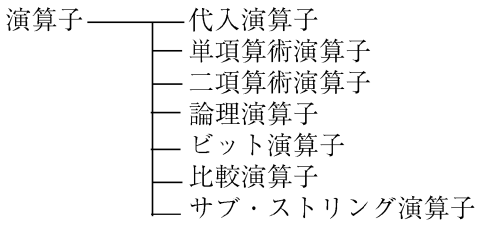
```
ADVANTEST
advantest
```

- 組み込み関数

関数	機能
SIN（算術式） COS（算術式） TAN（算術式） ATN（算術式）	正弦 (sin) 余弦 (cos) 正接 (tan) 逆正接 ($\tan^{-1}$)　　角度の単位 = ラジアン
LOG（算術式）	自然対数
SQR（算術式）	平方根
ABS（算術式）	絶対値
NUM（文字列式）	文字列式の先頭の 1 文字の ASCII コードを返す。 （例）NUM("A") ---> 65
CHR$（算術式）	算術式の値に対応する ASCII 文字 1 文字の文字列式を返す。 （例）CHR$(65) ---> "A"
LEN（文字列）	文字列式の長さを返す。 （例）LEN("ADVANTEST") ---> 9
POS （文字列式 1, 文字列式 2）	文字列式 1 の中から、文字列式 2 がある位置の先頭の位置を返す。 （例）POS("ADVANTEST", "AN") ---> 4
ビルトイン関数	測定値を扱うための関数 詳細は、4.4 ビルトイン関数を参照して下さい。

4.1.3 演算子

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



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

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

   代入演算子を以下に示します。

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

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

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

   結果
   10.0
   11.0
   10.0
   9.0
   9.0

3. 二項算術演算子
   + : 加算
   - : 減算
   * : 積算
   / : 割算
   % : モジュロ（余り）
   ^ : 累乗
   & : 文字列の連結

4. 論理演算子
   NOT
   AND
   OR
   XOR

5. ビット演算子
   数式は整数型のみ有効で、実数型ではエラーになります。
   BNOT
   BAND
   BOR
   BXOR

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

=　　: イコール
<>　　: ノットイコール（または！ =）
<
>
<=
>=

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

7. サブ・ストリング演算子
文字列式の一部を文字列として指定できます。
文字列式［算術式 1, 算術式 2］　: 文字列式の先頭から、算術式 1 が表現する値だけ進んだ所から算術式 2 が表わす値の所までをサブ・ストリングとする。
"ADVANTEST"［1,5］--->"ADVAN"

文字列式［算術式 1; 算術式 2］　: 文字列式の先頭から、算術式 1 が表現する値だけ進んだ所から算術式 2 が表わす値の文字数分をサブ・ストリングとする。
"ADVANTEST"［6;4］--->"TEST"

4.2　各種ステートメント

4.2.1　　ステートメント機能一覧

1.　基本ステートメント

(1/2)

ステートメント	機能
BUZZER	ブザーを鳴らす
CLS	ディスプレイ表示をクリア
CONSOLE	スクロール範囲を指定
CURSOR	カーソルを移動
DATA	READ 文で読み込むための数値、文字列を定義
DATE$	本器に内蔵している時計 (RTC) の日付の値を読み出す
DIM	配列変数または文字列変数を定義
DISABLE INTR	割り込みの受付を禁止
ENABLE INTR	割り込みの受付を許可
ERRM$	エラー・メッセージを返す
ERRN	エラー番号を返す
FOR-TO-STEP, NEXT, BREAK, CONTINUE	ループ処理を行う
FRE	BASIC プログラム・メモリの残量を返す
GOSUB, RETURN	サブルーチンへの分岐、復帰
GOTO	指定行への分岐
GPRINT	数値または文字列を GPIB へ出力
IF-THEN, ELSE, END IF	条件付分岐
INPUT	キーからの入力
INTEGER	変数が整数型であることを定義
KEY$	本器のパネル・キーのコードを返す
LPRINT	数値または文字列をシリアル・ポートへ出力
LET	変数への代入
OFF ERROR	BASIC エラーを検出したときの分岐の解除
OFF ISRQ	ISRQ による割り込み分岐を解除
OFF KEY	キー入力による割り込み分岐を解除
OFF SRQ	SRQ による割り込み分岐を解除
ON DELAY	指定時間経過後に分岐

(2/2)

ステートメント	機能
ON ERROR	BASIC エラーを検出したときの分岐の定義
ON ISRQ	本器内部要因による割り込み分岐を定義
ON KEY	キー入力による割り込み分岐を定義
ON SRQ	GPIB 外部 SRQ 信号による割り込み分岐を定義
PRINT [USING]	数値または文字列を表示
PRINTER	プリンタの GPIB アドレスを設定
PRINTF	数値または文字列を表示
READ	DATA 文の定数を変数に代入
REM	注釈
RESTORE	次の READ 文で読み込む DATA 行を指定
SELECT, CASE, END SELECT	式の値を条件として複数の分岐を行う
SPRINTF	PRINTF の書式に従った結果を文字列へ代入
TIME$	本器に内蔵されている時計 (RTC) の時刻の値を返す
TIMER	内蔵システム時間の読み出しおよびリセット
WAIT	指定時間だけ待つ
WAIT EVENT	指定したイベントが発生するまで待つ

2. GPIB 制御用ステートメント

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

3. ファイル制御用ステートメント

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

4.2.2 ステートメント文法一覧

1. 基本ステートメント

(1/3)

ステートメント	文法
BUZZER	BUZZER < 音程 > < 時間 >
CLS	CLS
CONSOLE	CONSOLE < 開始行 >,< 終了行 >
CURSOR	CURSOR <X 軸 >,<Y 軸 >
DATA	DATA 数値定数 I 文字列定数 {, 数値定数 I 文字列定数 }
DATE$	(1) DATE$ (2) DATE$ ="YY/MM/DD"
DIM	DIM <B> I <C>{, <B> I <C>}
DISABLE INTR	DISABLE INTR
ENABLE INTR	ENABLE INTR
ERRM$	ERRM$（エラー番号）
ERRN	ERRN
FOR-TO-STEP, NEXT, BREAK, CONTINUE	FOR 数値変数 = 数値表現式 TO 数値表現式 [STEP 数値表現式] [BREAK] [CONTINUE] NEXT [数値変数]
FRE	FRE（数値）
GOSUB, RETURN	GOSUB 整数 I ラベル式 RETURN
GOTO	GOTO 整数 I ラベル式

(2/3)

ステートメント	文法
GPRINT	GPRINT [A {, ｜;A}]
IF-THEN, ELSE, END IF	(1) IF < 条件式 > THEN < 文 > (2) IF < 条件式 > THEN [ELSE IF < 条件式 > THEN] [複文] [ELSE] [複文] END IF
INPUT	INPUT ["< 文字列 >",] A{,A}
INTEGER	INTEGER <B> {,<B>}
KEY$	KEY$
LPRINT	LPRINT [A {, ｜;A}]
LET	LET <D> ｜<E> {:<D>｜<E> }
OFF ERROR	OFF ERROR
OFF ISRQ	OFF ISRQ
OFF KEY	OFF KEY [キーコード]
OFF SRQ	OFF SRQ
ON DELAY	ON DELAY 時間 GOTO｜GOSUB 整数｜ラベル式
ON ERROR	ON ERROR GOTO ｜GOSUB 整数｜ラベル式
ON ISRQ	ON ISRQ GOTO｜GOSUB 整数｜ラベル式
ON KEY	ON KEY キーコード GOTO ｜GOSUB 整数｜ラベル式
ON SRQ	ON SRQ GOTO ｜GOSUB 整数｜ラベル式
PRINT [USING]	(1) PRINT [A {, ｜;A}] (2) PRINT USING 書式設定式 ;{,A}
PRINTER	PRINTER 数値表現式
PRINTF	PRINTF 書式表現式 {,A}
READ	READ 入力項目 {, 入力項目 }
REM	REM [文字列] または ![文字列]
RESTORE	RESTORE 整数｜ラベル式
SELECT, CASE, END SELECT	SELECT < 数式｜文字列式 > CASE < 数式｜文字列式 > 複文 [CASE ELSE] [複文] END SELECT

(3/3)

ステートメント	文法
SPRINTF	SPRINTF 文字列変数 書式指定 {,A}
TIME$	TIMER(0 | 1)
TIMER	(1) TIME$ (2) TIME$ ="HH:MM:SS"
WAIT	WAIT 時間
WAIT EVENT	WAIT EVENT < イベント番号 >

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

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

  イメージ仕様

  D ： D の数で出力桁数を指定する。指定フィールドの余った部分にはスペースが入る。
  Z ： Z の数で出力桁数を指定する。指定フィールドの余った部分には 0 が入る。
  K ： 式の値をそのまま表示する。
  S ： S の位置に + または - のサイン・フラグを付けて表示する。
  M ： M の位置に、負のときは -、正のときはスペースを付けて表示する。
  . ： . の位置に小数点がくるように位置を合わせて表示する。
  E ： 指数形式（e, 符号 , 指数）で表示する。
  H ： K と同じ。ただし、小数点にカンマ (,) を使う。
  R ： . と同じ。ただし、小数点にカンマ (,) を使う。
  * ： * の数で出力桁数を指定する。指定フィールドの余った部分には * が入る。
  A ： 1 文字を表示する。
  k ： 文字列式をそのまま表示する。
  X ： スペース 1 文字を表示する。
  リテラル ： 書式指定式にリテラルを書くときは \" で囲む。
  B ： 式の値を ASCII コードとして表示する。
  @ ： 改ページする。
  + ： 表示の位置を同じ行の先頭に移動させる。
  - ： 改行する。
  # ： 最後に改行しない。
  n ： 数字で各イメージ仕様の繰り返し回数を指定できる。

- PRINTF の書式指定は、以下に示す方法で % に続けて指定します。
  %〔-〕〔0〕〔m〕〔. n〕文字

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

2. GPIB ステートメント

ステートメント	文法
CLEAR	CLEAR [装置アドレス {, 装置アドレス }]
DELIMITER	DELIMITER 数値表現式
ENTER	ENTER 装置アドレス ; B{,B}
INTERFACE CLEAR	INTERFACE CLEAR
LOCAL	LOCAL [装置アドレス {, 装置アドレス }]
LOCAL LOCKOUT	LOCAL LOCKOUT
OUTPUT	OUTPUT アドレス {, 装置アドレス }; A {,A}
REMOTE	REMOTE [装置アドレス {, 装置アドレス }]
REQUEST	REQUEST 整数
SEND	SEND <C> I <D> {,<C> I <D> }
SPOLL	SPOLL（装置アドレス）
TRIGGER	TRIGER [装置アドレス {, 装置アドレス }]

A: 数値表現式 I 文字列表現式
B: 数値変数 [文字列変数]
C: <CMD I DATA I LISTEN I TALK> [数値表現式 {, 数値表現式 }]
D: UNL I UNT

3. ファイル制御用ステートメント

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

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

- ENTER USING のイメージ仕様

  イメージ仕様

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

- OUTPUT USING のイメージ仕様

  イメージ仕様

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

4.3 ステートメント文法と活用

ここで説明するステートメントのインデックスを以下に示します。

1. BUZZER

- 概要　ブザーを鳴らします。
- 構文　(1)-1

(1)-2

BUZZER 整数 1 , 整数 2

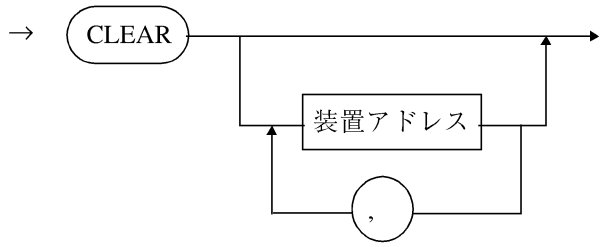
注	整数 1 は、高低音を 0（高音）～ 65535（低音）の範囲で指定します。 整数 2 は、時間（ms 単位）を示します。

- 解説　本器内蔵のブザーを指定にしたがって鳴らします。
- 例

```
10    FOR I=0 TO  255
20        BUZZER I, 10
30    NEXT I
40    STOP
```

2. CLEAR

- 概要　GPIB 上に接続されたすべての装置または選択された特定の装置を初期状態にします。
- 構文　(1)-1



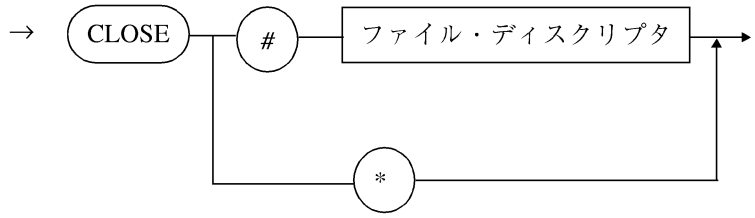
(1)-2

CLEAR〔装置アドレス {, 装置アドレス }〕

- 解説　・装置アドレスを指定せずに CLEAR だけを実行すると、GPIB 上にユニバーサル・コマンドのデバイス・クリア (DCL) を送ります。これによって、GPIB に接続されているすべての装置を初期状態にできます。
  ・CLEAR に続いて装置アドレスを指定すると、装置アドレスで指定された装置のみをアドレスし、アドレス・コマンドのセレクト・デバイス・クリ (SDC) を送ります。これによって、特定の装置のみを初期状態にします。
  なお、装置アドレスは複数指定できます。
  ・CLEAR で各装置に設定される初期状態の定義は、各装置に依存します。
- 例　10　CLEAR
  20　CLEAR 2
  30　CLEAR 1, 3, 5, 7
- 注意　ADDRESSABLE モードでは機能しません。

3. CLOSE

- 概要　ファイル・ディスクリプタに割り当てられているファイルをクローズします。
- 構文　(1)-1



(1)-2

CLOSE <# ファイル・ディスクリプタ | *>

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

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

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

CLOSE*

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

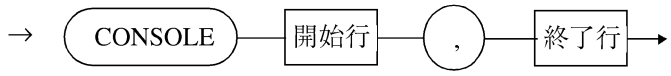
※　ファイルの取り扱いについては、2.4 ファイルの管理を参照して下さい。

4. CLS
   - 概要　ディスプレイの表示をクリアします。
   - 構文　(1)-1

(1)-2
CLS

   - 解説　・ディスプレイに表示されているキャラクタをクリアすると同時に、カーソルをホーム・ポジションに戻します。
     ・CONSOLE によって指定されたスクロール範囲内のみクリアします。
   - 例　10　CLS

5. CONSOLE
   - 概要　スクロール範囲を指定します。
   - 構文　(1)-1



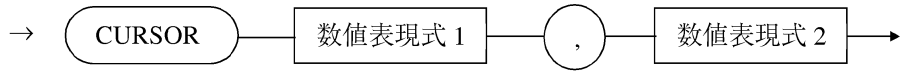
(1)-2
CONSOLE　開始行　,　終了行

注　開始行よりも小さい値を終了行に指定した場合、終了行 = 開始行として実行します。

   - 解説　・テキスト画面のスクロール範囲を設定します。
     ・開始行、終了行の指定範囲は、以下の通りです。
     R3752（蛍光表示管）　: 0 ～ 7
     R3752（外部モニタ）　: 0 ～ 29
     R3753　: 0 ～ 29
   - 例　10　CONSOLE 0,5
     20　PRINT "This is Network Analyzer"
     30　PRINT "....Sweep Check Program...."
     40　STOP

6. CURSOR

- 概要　指定された座標位置にカーソルを移動させます。
- 構文　(1)-1



(1)-2

CURSOR　数値表現式 1　,　数値表現式 2

注　　数値表現式 1:X 軸指定、カラム方向
　　数値表現式 2:Y 軸指定、行方向
　　カンマ (,) をスペースにしても構いません。

- 解説　・ディスプレイの指定された位置にカーソルを移動させます。
・数値表現式 1 が X 軸座標、数値表現式 2 が Y 軸座標を示します。
・X 軸座標、Y 軸座標の指定範囲を以下に示します。

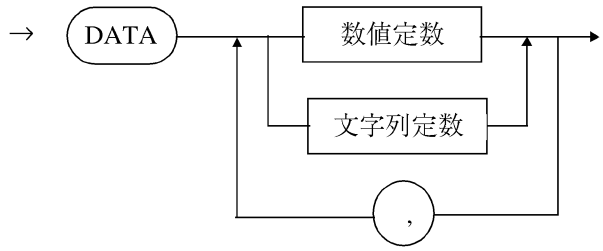
  R3752（蛍光表示管）　: 0 ≤ X ≤ 31　0 ≤ Y ≤ 7
  R3752（外部モニタ）　: 0 ≤ X ≤ 79　0 ≤ Y ≤ 29
  R3753　　: 0 ≤ X ≤ 66　0 ≤ Y ≤ 29

- 例

```
10    CLS
20    X=4:Y=4:X1=1:Y1=1
30    CURSOR X, Y:PRINT " " ;
40    X=X+X1:Y=Y+Y1
50    CURSOR X, Y:PRINT "*" ;
60    IF X<=0 OR 67<=X THEN X1*=-1
70    IF Y<=0 OR 29<=Y THEN Y1*=-1
80    GOTO 30
90    STOP
```

7. DATA

- 概要　READ 文で読み込むための数値、文字列を定義します。
- 構文　(1)-1



(1)-2

DATA < 数値定数 | 文字列定数 >{, < 数値定数 | 文字列定数 >}

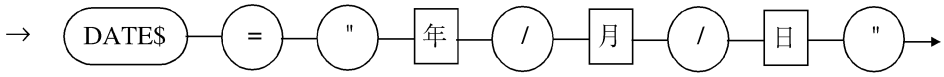
- 解説
  - DATA 文は実行の対象とはならないため、どの文番号にあっても構いませんが、原則として、READ 文で読み出す順序に従っている必要があります。
  - READ 文がプログラムの中から DATA 文を捜して、対象となるデータを読み込むことになります。
  - この順序を変更するには、RESTORE 文を使います。
  - DATA 文には、カンマ (,) かスペースで区切ることによって複数個の定数を指定できます。
    文字列は、文字列定数としてダブルクォーテーション (") で囲みます。
  - DATA の後にコロン (:) によるマルチ・ステートメントは、使用できませ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

8. DATE$

- 概要　日付の読み出しと、日付の変更をします。
- 構文　(1)-1

(1)-2
DATE$

(2)-1



(2)-2
DATE$=" 年／月／日 "

- 解説　本器内蔵の時計 (RTC) の日付を読み出します。
  読み出した日付は変更できます。
  以下のように入力して下さい。

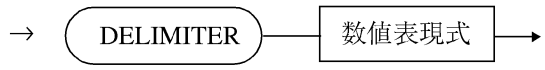
  DATE$="93/1/1"
  または
  DATE$="93/01/01"

- 例

```
10    DIM D$[10]
20    D$=DATE$
30    PRINT "Date is" ; D$
40    PRINT "Date Reset"
50    DATE$="93/1/1"
60    STOP
```

9. DELIMITER

- 概要　GPIB 出力時のデリミタを設定するステートメントです。
- 構文　(1)-1



(1)-2

DELIMITER 数値表現式

- 解説　・OUTPUT 文などで GPIB にデータ出力する際のデリミタ（終結子）を数値表現式で選択します。
選択番号および種類を下表に示します。

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

CR= キャリッジ・リターン（16 進 0D）
LF= ライン・フィード（16 進 0A）
EOI= 単線信号 (End OrI dentify)

・数値表現式の結果が 0 ～ 3 の範囲を超えた場合は、エラーとなります。
また、小数点以下の数値は無視し、整数として取り扱います。
・電源投入時は、DELIMITER=0 が自動的に設定されます。

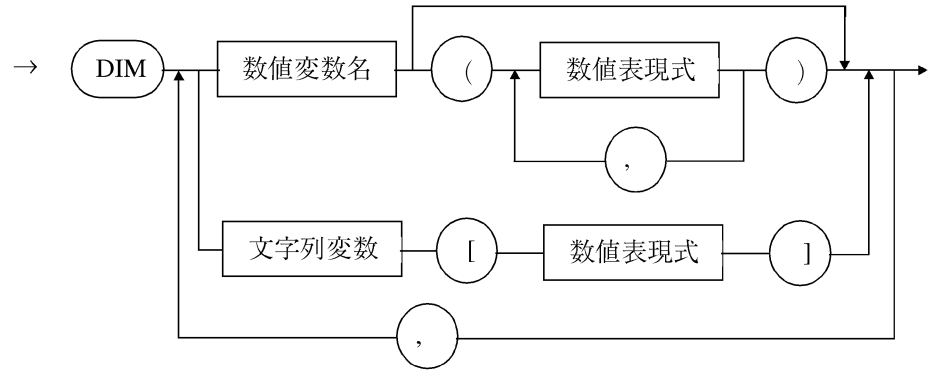
注　DELIMITER 文で指定されるのは、GPIB への出力時の終端コードです。
GPIB からのデータ入力の際には、常に LF または EOI で終端されます。

- 例

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

10. DIM

- 概要　配列変数または文字列変数の定義宣言を行います。
- 構文　(1)-1



(1)-2

DIM <A | B>{, <A | B>}

注　A: 数値変数名〔(数値表現式 {,　数値表現式 })〕
　　B: 文字列変数 [数値表現式]

- 解説
  - 配列変数および文字列変数を使用するときは、DIM で配列変数名と配列の大きさを定義しなくてはなりません。定義をしないで配列変数を使うと、配列は 1 次元で 10 個の要素数になり、文字列は 18 文字の長さになります。
  - DIM で配列宣言をすると、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、BASIC プログラムの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)
  - 配列変数の大きさを示す数値表現式は、演算の結果が実数表現となっても、小数点以下を切り捨て、整数として扱います。
    配列変数に 0 は使えません。
  - 文字列変数のとき、数値表現式は文字列の長さを宣言します。
- 例

```
10    DIM N(5)                    < 実行結果 >
20    FOR I = 1 TO 5                0.5
30        N(I) = I*I/2              2.0
40    NEXT I                        4.5
50    FOR I = 1 TO 5                8.0
60        PRINT N(I)                12.5
70    NEXT I
```

11. DISABLE INTR

- 概要　割り込みの受付を禁止します。
- 構文　(1)-1

(1)-2
DISABLE INTR

- 解説
  - DISABLE INTR は、ENABLE INTR で許可された割り込みを禁止します。
  - DISABLE INTR の実行後に、再び割り込みを許可する場合は、ENABLE INTR を実行します。このとき、ON XXX ステートメントで設定された分岐の条件は、以前の状態を保っています。
    ただし、割り込み分岐の条件を変更する場合は、ENABLE INTR を実行する前に ON XXX、または OFF XXX の各ステートメントを用いて設定して下さい。
  - プログラムを実行した直後は、ENABLE INTR を実行するまで割り込みは禁止状態になっています。
- 例

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

12. DSTAT

- 概要　ディレクトリの内容を BASIC 変数に取り込みます。
- 構文　(1)

  DSTAT <index> < 変数 >

  (2)

  DSTAT <index> <filename> <fileattribute> <size> <sectors>
  <year> <month> <day> <hour> <minutes> <start-sector>

  (3)

  DSTAT ; SELECT < 文字列 > COUNT < 変数 >
- 解説　・(1) の構文

  ファイル･システムのディレクトリに登録されているファイルの数を調べます。
  1 つ目のパラメータ <index> には 0 を指定します。
  2 つ目のパラメータには数値変数を指定します。ここに結果が代入されます。

  ・(2) の構文

  ファイル・システムのディレクトリ情報を BASIC 変数に取り込みます。
  1 つ目のパラメータ <index> で、ディレクトリ内のインデックスを指定します。指定可能な値は 1 から登録ファイル数までです。（登録ファイル数は、(1) の構文で得られる値です。）2 つ目のパラメータには文字列変数を指定します。ここに結果のファイル名が格納されます。
  3 つめのパラメータ以降には、すべて数値変数を指定します。ここに以下の内容が代入されます。

fileattribute	ファイル属性（複数の属性がある場合は、加算されて出力されます。） 1. READ ONLY　8. VOLUME LABEL 2. HIDDEN FILE　16. DIRECTORY 4. SYSTEM FILE　32. ARCHIVE FILE
size	ファイル・サイズ（バイト数）
sectors	セクタ数
year, month, day	ファイルの作成年月日
hour, minutes	ファイルの作成時刻
start-sector	ファイルの開始セクタ

・ (3) の構文

< 文字列 > で指定したファイルの数を < 変数 > に代入します。
この構文は、指定ファイルがディレクトリ内に存在するかなどのファイルサーチとして使用できます。
指定した < 文字列 > の中に以下の文字がある場合は、特別な意味になります。

? ： 1 文字と一致する。

* ： 1 文字以上と一致する。

[ ] ： [ ] で囲まれた文字列のいずれか 1 文字と一致する。

[文字 1 －文字 2] で指定すると、文字 1 から文字 2 の範囲にある文字と一致する。

13. ENABLE INTR

- 概要　割り込みの受け付けを許可します。
- 構文　(1)-1

(1)-2
ENABLE INTR

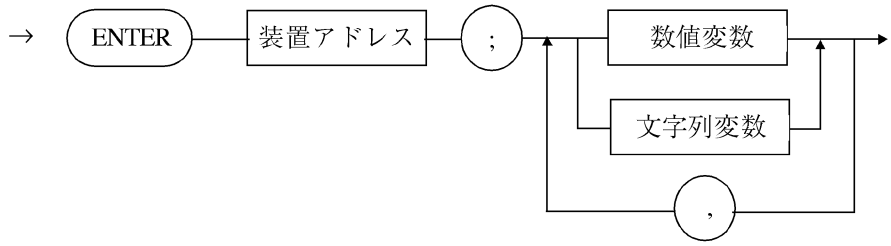
- 解説
  - ・ ENABLE INTR は、割り込みの受け付けを許可し、ON XXX ステートメントで定義された割り込み分岐を有効にします。
  - ・ DISABLE INTR の実行後に再び割り込みを許可する場合は、ENABLE INTR を実行します。
  - ・ プログラムを実行させた直後は、ENABLE INTR を実行するまで割り込みは禁止状態になっています。（DISABLE INTR 状態）
- 例

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

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

14. ENTER

- 概要　(1) GPIB およびパラレル I/O からデータを取り込みます。
  (2) ファイルからデータを読み込み、入力項目に代入します。
- 構文　(1)-1



(1)-2

ENTER 装置アドレス ;< 数値変数 | 文字列変数 >{,< 数値変数 | 文字列変数 >}

装置アドレス :

0 ～ 30 ; 外部 GPIB 接続機器のアドレス

31 ; 本器の測定部からのデータ入力

34 ; パラレル・ポートの Flip/Flop の状態の読み出し

35 ; パラレル・ポートの C ポートのデータ読み出し

36 ; パラレル・ポートの D ポートのデータ読み出し

37 ; パラレル・ポートの CD ポートのデータ読み出し

(2)-1

(2)-2

ENTER # ファイル・ディスクリプタ ; 入力項目 {, 入力項目 }

- 解説　(1) の構文
  - 装置アドレスによって指定された装置から GPIB を通してデータを入力し、数値または文字列として BASIC の変数内に蓄えます。ただし、装置アドレスによって指定された装置にトーカ機能がない場合、コントローラはハンドシェークを完了できずに停まってしまうので、注意して下さい。
    また、文字列変数を使用する場合は、あらかじめ DIM 文によって文字列変数を宣言しておかなければなりません。
  - 文字列で入力するときは、デスティネーションに使用する文字列変数の長さが十分でないと、入力データがオーバフローを起こし、文字列変数に入りきらないデータは無視されるので、注意して下さい。

・ GPIB からの入力は、LF または EOI によって終端されます。
4.3 節の 9. DELIMITER も参照して下さい。

・ 例

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

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

(2) の構文

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

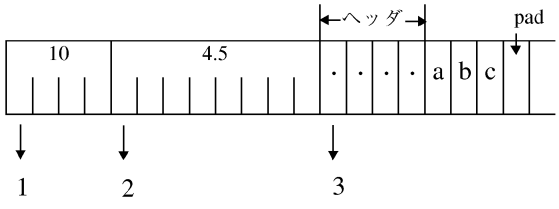
※ファイルの取り扱いについては、2.4 ファイルの管理を参照して下さい。

・ 例 1　BINARY ファイル

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

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

代入する変数のタイプによって読み込むバイト数が違ってきます。

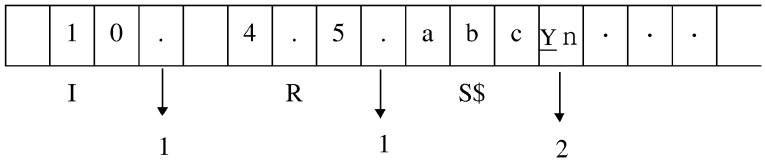


1 : 変数が整数のときは、4 バイトのデータを読み込み、そのままのデータを変数に代入します。
2 : 変数が実数のときは、8 バイトのデータを読み込み、そのままのデータを変数に代入します。
3 : 変数が文字列のときは、ヘッダ 4 バイトを読み込み、ヘッダが示す長さ分だけ読み込み、文字列変数に代入します。

・ 例 2　TEXT ファイル

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

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



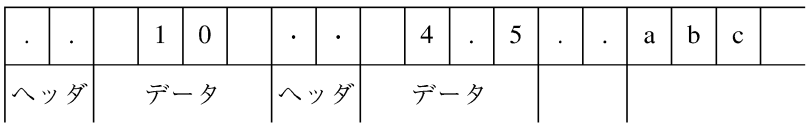
1: 各項目はカンマ (,) で区切られます。

2: 最後の項目の後にはライン・フィードがあります。

・ 例 3　ASCII ファイル

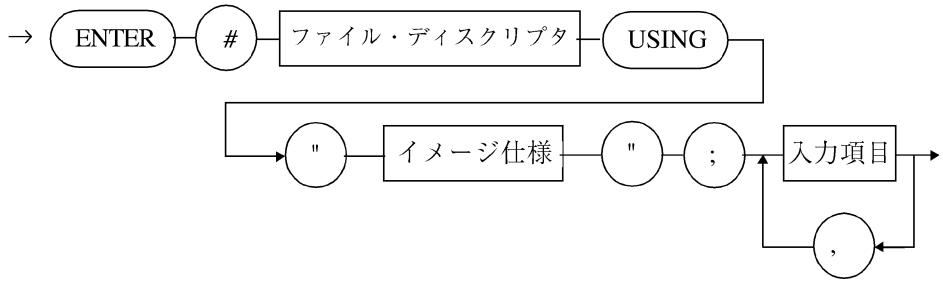
ヘッダ 2 バイトを読み込み、ヘッダが示す長さのデータを読み込みます。変数に型にデータを変換し代入します。

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



15. ENTER USING

- 概要　ファイルからイメージ仕様のフォーマットで入力項目に入力します。
- 構文　(1)-1



(1)-2

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

注　ENTER は ENT、USING は USE と省略できます。

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

イメージ仕様

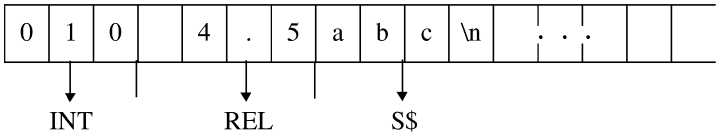
D　：D の数を数値の桁数と解釈して数値を読み込み、入力項目の変数に代入する。
Z　：D と同じ。
K　：1 行読み込み、数値データに変換し、入力項目の変数に代入する。
S　：D と同じ。
M　：D と同じ。
.　：D と同じ。
E　：K と同じ。
H　：K と同じ。ただし、小数点にカンマ (,) を使う。
*　：D と同じ。
A　：A の数分の文字を読み込み、文字列変数に代入する。
k　：1 行読み込み文字列変数に代入する。
X　：1 文字のデータを読み飛ばす。
リテラル：\" で囲まれた文字列の数のデータを読み飛ばす。
B　：1 文字読み込み、入力項目に ASCII コードとして代入する。
@　：1 バイトのデータを読み飛ばす。
+　：@ と同じ。
-　：@ と同じ。
#　：ENTER では無視される。

n　　：数字で各イメージ仕様の繰り返し回数を指定できる。
3D. 2D は DDD. DD と同じで、4A は AAAA と同じ。

※　ファイルの取り扱いについては、2.4 ファイルの管理を参照して下さい。

- 例

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

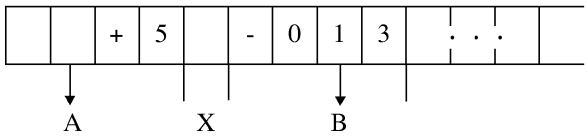


INT　：3 バイトのデータを読み込み、INT のデータタイプの整数型に変換し、INT に代入します。実行後の INT の値は 10 になります。

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

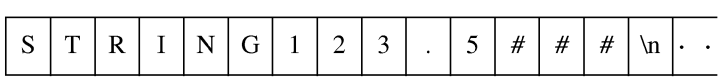
S$　：3 バイトのデータを読み込み、S$ に代入します。
実行後の S$ は abc です。

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



A, B　：4 バイトのデータを読み込み、実数型に変換して A, B に代入します。
実行結果、A=5. 0, B=-13. 0
イメージ仕様の X は 1 バイトを読み込みますが、変数へ代入しません。
SDDD のフォーマットで入力するデータを実数型に変換して A に代入します。
X は変数を必要とせず、1 文字分が読み飛ばされます。
MZZZZ が B に対応し、4 バイトを入力し実数に変換して B に代入されます。

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



実行結果　　A=123.5

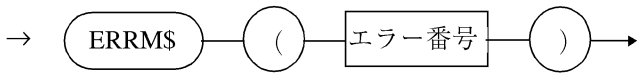
STRING123. 5## を読み込んで入力変数 A の実数型に変換します。

入力項目が実数型の場合、先行する数値、符号 (+, -)、指数の E, e 以外の文字は無視され、数値のみを取り込みます。数値を検出してから、数値でない文字になったところで数値への変換をやめます。

イメージ仕様の K, E, k, H は、ライン・フィードがターミネータになるので、現在のファイル・ポインタからライン・フィードまでを 1 つのデータとして変数に代入します。

16. ERRM$

- 概要　指定する番号のエラーメッセージを返すシステム関数です。
- 構文　(1)-1



(1)-2

ERRM$（エラー番号）

- 解説
  - パラメータで指定されたエラー・メッセージを返します。
    特にパラメータとして 0 を指定すると、直前に表示されたエラー・メッセージを返します。
  - エラー番号は、以下のような構造になっています。
    エラークラス *256+ エラー・メッセージ番号
    エラークラス：　1；データ入出力関係
    2；データ演算処理関係
    3；ビルトイン関数関係
    4；BASIC 構文関係
    5；その他
  - エラークラスを含む番号を指定しても、内部ではエラーメッセージ番号だけを参照します。したがって、エラー番号に ERRN を指定できます。

17. ERRN

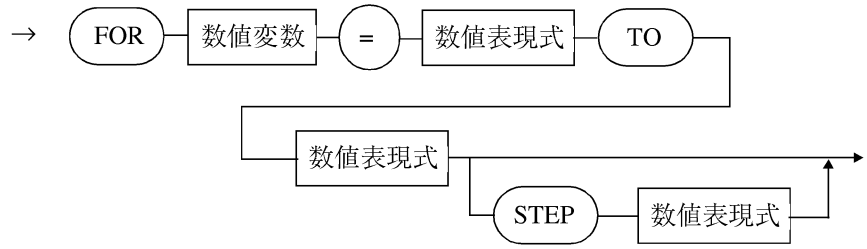
- 概要　エラー番号を保持しているシステム変数です。
- 構文　(1)-1

(1)-2
ERRN

- 解説
  - BASICプログラムの実行の際に発生したエラーの番号を保持するシステム変数です。
  - BASIC プログラムの開始時に 0 に初期化され、エラーが発生するとその値が代入されます。この値は、明示的に 0 を代入するか、BASIC プログラムを再び実行させると 0 に初期化します。
  - エラー番号は、以下のような構造になっています。
    エラークラス *256+ エラー・メッセージ番号
    エラークラス：　1；データ入出力関係
    2；データ演算処理関係
    3；ビルトイン関数関係
    4；BASIC 構文関係
    5；その他

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

- 概要　FOR 文と NEXT 文の一対でプログラムのループ（繰り返し処理）を構成します。
- 構文　(1)-1



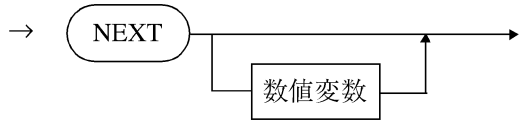
（1)-2
FOR 数値変数 = 数値表現式　TO　数値表現式〔STEP 数値表現式〕
(2)-1

(2)-2
BREAK

(3)-1

(3)-2
CONTINUE

(4)-1



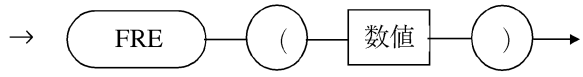
(4)-2
NEXT〔数値変数〕

- 解説
  - 指定された数値変数をループ（繰り返し）のカウンタとして用い、初期値から最終値まで増加分ずつ変化させていきます。カウンタの値が最終値より大きくなったとき、ループは終了します。カウンタの増減は NEXT 文で行います。したがって、FOR 文～ NEXT 文までの間に組まれたプログラムを繰り返し処理します。
  - 初期値、最終値、増加分は、以下のように指示します。
    FOR A=（初期値）TO（最終値）STEP（増加分）
  - STEP（増加分）を省略した場合、自動的に +1 となります。
  - FOR 文～ NEXT 文は入れ子 (Nest) にできます。
  - 一対の FOR 文と NEXT 文で使用するループ・カウンタの数値変数名は、同じにして下さい。数値変数名が異なっているとエラーになります。
  - FOR 文～ NEXT 文で繰り返し処理をしているときに、ループ・カウンタに使っている数値変数の値を変えると、正常な繰り返し処理をしなくなりますので注意して下さい。
  - NEXT 文の後の数値変数を省略した場合、自動的に直前の FOR 文と対応します。
  - FOR-NEXT のループは、BREAK 文で抜け出すことができます。
  - CONTINUE 文では、FOR-NEXT のループ内で次のステップ値のループに分岐します。
- 例

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

19. FRE

- 概要　BASIC バッファのメモリ残量を返すシステム関数です。
- 構文　(1)-1



(1)-2

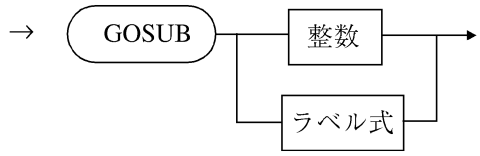
FRE（数値）

注　　数値に指定する値は何でも構いません。ただし省略はできません。

- 解説　・ BASIC が使用できるメモリのおおよその残りの容量を、文字数で返すシステム関数です。
  ・ これはおおよその容量を判断するためのもので、厳密にメモリ内の再構成はしません。したがって、一度セーブしてロードし直すと、より大きな容量を得ることがあります。
- 例　　PRINT FRE(0)

20. GOSUB, RETURN

- 概要　指定されたサブルーチンへの分岐、復帰を行います。
- 構文　(1)-1



(1)-2

GOSUB < 整数 | ラベル式 >

(2)-1

(2)-2

RETURN

- 解説
  - 整数またはラベル式によって指示された行番号から始まるサブルーチンへ処理の制御を移し、RETURN 文で GOSUB 文の次の文へ戻ります。
  - サブルーチンの最後には必ず RETURN 文を入れて、処理をメイン・プログラムへ戻して下さい。
  - サブルーチンの分岐をせずに RETURN 文を実行するとエラーになります。
  - GOSUB 文～ RETURN 文は入れ子 (Nest) にできるので、サブルーチンの中から別のサブルーチンへ分岐できます。ただし、あまり入れ子を大きくするとメモリ容量がなくなり、エラーになることがあります。
  - GOTO、GOSUB などでラベル式を書いたとき、その対象となるラベル式の行がない場合、または誤ってそのラベル式の行を削除した場合、GOTO や GOSUB のラベル式の値が 0 になってしまいます。このままで実行しようとすると、以下のメッセージが表示されます。

    Undefined line

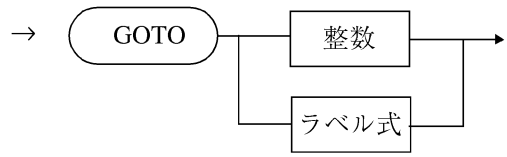
    このままでは実行できません。GOTO、GOSUB の行を正しいラベル式に直して下さい。

- 例

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

21. GOTO

- 概要　指定された行番号へ分岐します。
- 構文　(1)-1



(1)-2

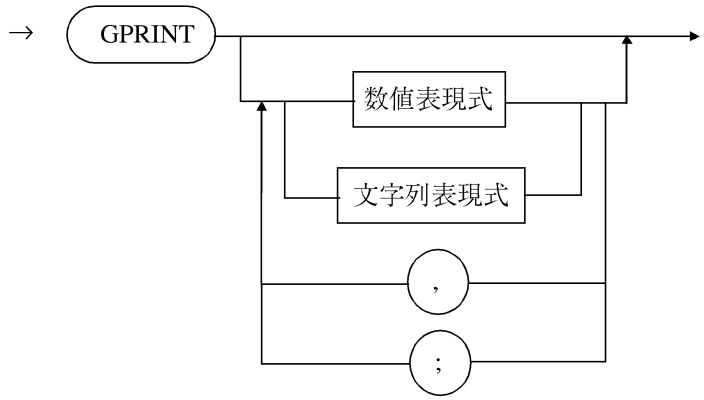
GOTO < 整数 | ラベル式 >

- 解説　指定された行番号へ無条件の分岐をします。
- 例

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

22. GPRINT, LPRINT

- 概要　数値または文字列を出力します。
  GPRINT ： GPIB 出力
  LPRINT ： シリアル出力
- 構文　(1)-1



(1)-2
GPRINT〔< 数値表現式 I 文字列表現式 >{, I ;< 数値表現式 I 文字列表現式 >}

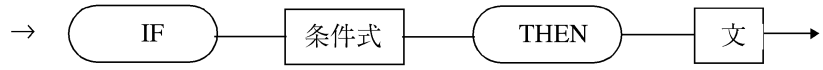
(2)
LPRINT は GPRINT と同様です。

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

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

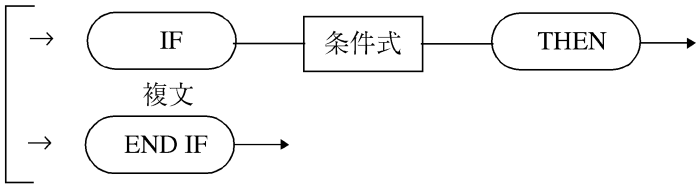
23. IF-THEN, ELSE, END IF

- 概要　条件判断による分岐、指定された文の実行をします。
- 構文　(1)-1



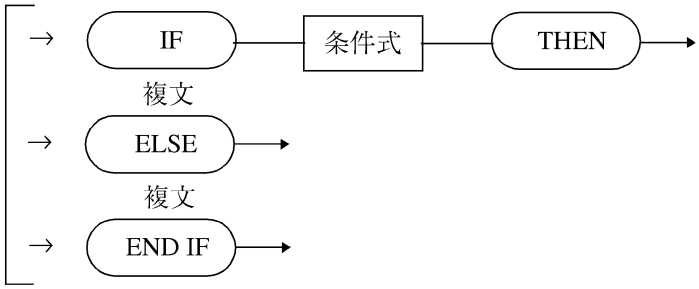
(1)-2
IF 条件式 THEN 文

(2)-1



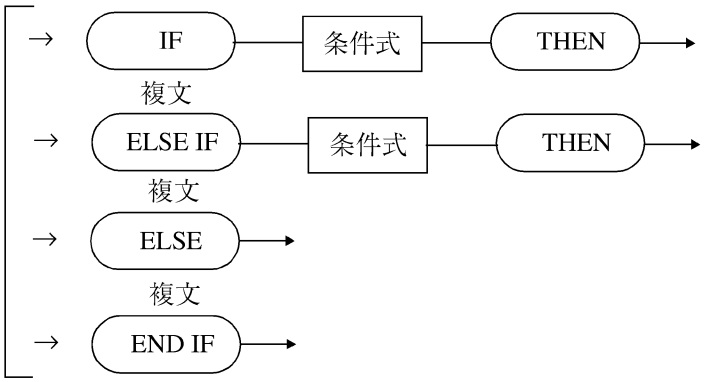
(2)-2
IF 条件式 THEN
　複文
END IF

(3)-1



(3)-2
IF 条件式 THEN
　複文
ELSE
　複文
END IF

(4)-1



(4)-2

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

- 解説
  - ・条件式は論理式ですが、ここには比較演算子を用いた論理式以外に数値表現式を書くこともできます。この場合、演算結果が 0 になったときのみ (false) とし、それ以外の値は、すべて真 (true) と解釈します。
  - ・論理式の条件によってプログラムの分岐、処理などを行います。
  - ・論理式の関係が成立すると、THEN 文を実行します。THEN 文には文を続けることができ、次文を実行します。
  - ・論理式の関係が不成立の場合は、そのまま次の行に進みます。
  - ・比較演算子には、以下の 6 種類があります。

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

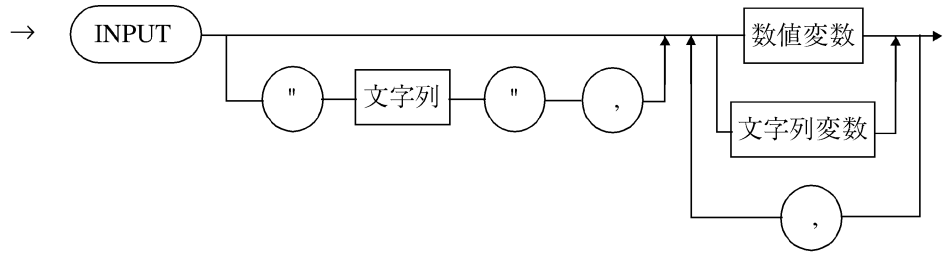
上の論理式で A, B はともに数値表現式で構成できます。
ただし、数値表現式と文字列表現式を比較することもできます。

- 例

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

24. INPUT

- 概要　キー入力したデータを数値変数に代入します。
- 構文　(1)-1



(1)-2

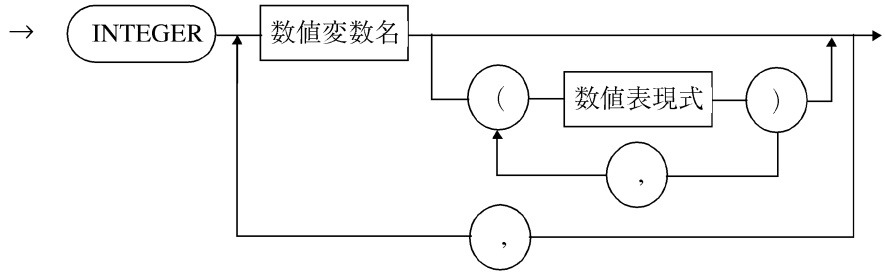
INPUT〔" 文字列 ",〕< 数値変数 | 文字列変数 >{,< 数値変数 | 文字列変数 >}

- 解説
  - INPUT を実行すると、プログラムは一時停止して、キー入力待ちとなります。キー入力待ちは ENTER キーが押されるまで続きます。データをキー入力後 ENTER キーを押すと、データが変数に代入されます。
  - INPUT では、数値変数、文字列変数のいずれも扱えるようになっていますが、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）を入力させると数字以外の文字は無視し、もし数字が一字もないときは 0 が変数に入力されます。また、ENTER キーのみが押されたときには変数への代入は行いません。つまり、INPUT 前の値がそのまま残ります。
  - 文字定数を入力するときは、引用符で囲む必要はありません。
    CONTROL コマンドのレジスタ 6 を 1 にすると、入力待ちのときにもファンクション・キーの受けができます。（R3752 は正面パネル・キーの構成上不可能）
- 例

```
10    OUTPUT 31;"OLDC OFF"
20    OUTPUT 31;"INIT: CONT OFF"
30    INPUT    "CENTER FREQUENCY(MHz)?", CF
40    INPUT    "SPAN FREQUENCY(KHz)?", SF
50    OUTPUT 31;"FREQ:CENT", CF, "MHz"
60    OUTPUT 31;"FREQ:SPAN", SF, "KHz"
70    OUTPUT 31;"INIT"
80    PRINT    "MAX =", MAX(0, 1200, 0)
90    STOP
```

25. INTEGER

- 概要　変数または配列変数が整数型であることを宣言します。
- 構文　(1)-1



(1)-2

INTEGERA〔B〕{, A〔B〕}

A: 数値変数名

B:（数値表現式 {, 数値表現式 }）

- 解説
  - INTEGER　文で、数値変数、配列変数を指定すると、以後その変数は、整数型となります。
  - 整数型変数で扱える数値は、整定数で扱える範囲と同じです。
    -2147483648 ～ +2147483647
  - 整数しか扱わない変数では、INTEGER　文で宣言した方が処理時間が短くなります。
  - INTEGER　命令で配列宣言すると、指定された大きさの配列変数をメモリ上に確保します。したがって、大きすぎる配列宣言をするとメモリ領域が足りなくなり、エラーとしてプログラムの実行を中止します。
    (memory space full)
  - 添字を複数個指定すると個数分の次元を持つ配列変数の指定となります。
    （次元数は、メモリ容量が許す限り）

- 例

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

< 実行結果 >

```
 J/I   1   2   3

  1   11  12  13
  2   21  22  23
```

注意

1. INTEGER 文で一度整数型に指定された変数は DEL やコメント文で命令を削除しても、整数型のままです。
2. 実数型に再指定したい場合は、DIM 命令を追加するか、SAVE/LOAD を一度行ってから RUN させます。

26. INTERFACE CLEAR

- 概要　本器に接続されているすべての GPIB インタフェースを初期化します。
- 構文　(1)-1

(1)-2
INTERFACE CLEAR

- 解説　INTERFACE CLEAR を実行すると、GPIB の単線信号 IFC を約 100 µs の間出力します。
  本器の GPIB に接続されている装置のすべての GPIB インタフェースは、IFC 信号を受け取ると、トーカまたはリスナの状態が解除されます。
- 例　10　INTERFACE CLEAR
- 注意　ADDRESSABLE モードでは機能しません。

27. KEY$

- 概要　パネル・キーのコードを返します。
- 構文　(1)-1

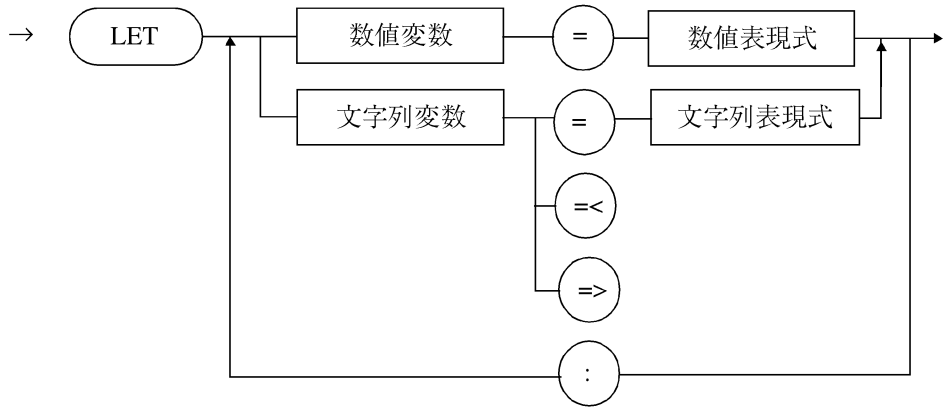
(1)-2
KEY$

- 解説　本器のパネル・キーのコードで、一番最後に押されたコードを返します。
一度参照すると、この変数の内容はクリアされます。
- 例

```
10    A$=KEY$
20    IF A$="1" THEN
30        GOSUB *TEST1
40    ELSE IF A$="2" THEN
50        GOSUB *TEST2
60    END IF
70    GOTO 10
80    STOP
100   *TEST1
110       PRINT "Check1 Start !!"
120       ......
130       RETURN
200   *TEST2
210       PRINT "Check2 Start !!"
220       ......
230       RETURN
```

28. LET

- 概要　（プログラム上では LET は使用せず、直接代入文を記述します。）
  変数に代入を行います。
- 構文　(1)-1



(1)-2

LET<A | B>{ : <A | B>}

A: 数値変数 = 数値表現式

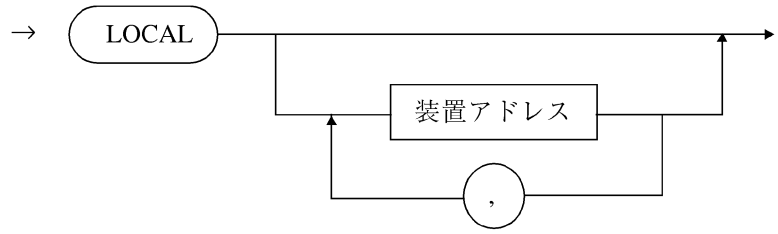
B: 文字列変数 = | = < | = > 文字列表現式

- 解説　・ ここで用いる等号 (=) は、代入を意味するもので、数学的な等号とは意味が異なります。
  ・ 等号の左辺が数値ならば文字列も数値の部分を変換して代入します。
  特に文字列を代入する場合
  = のとき、高々右辺の長さ分だけ代入されます。
  => のとき、左辺の文字列に比べ右辺の文字列が短いと、頭にスペースをつめて左辺に長さ分だけ代入します。
  =< のとき、後ろにスペースをつめます。
  したがって、=> と =< は文字列にのみ有効な代入演算子です。
- 例

```
10   DIM STR$                          < 実行結果 >
20   PRINT "123456789012345678"          123456789012345678
30   STR$ = "ABC" :PRINT STR$            ABC
40   STR$ = < "OPQ" :PRINT STR$          OPQ
50   STR$ = > "XYZ" :PRINT STR$XYZ
```

29. LOCAL

- 概要　指定した装置をリモート状態から解除するか、またはリモート・イネーブル (REN) ラインを偽にします。
- 構文　(1)-1



(1)-2

LOCAL〔装置アドレス {, 装置アドレス }〕

- 解説　・装置アドレスを指定せずに LOCAL だけを実行した場合、GPIB リモート・イネーブル (REN) ラインが偽 (High level) となり、GPIB 上のすべての装置がローカル状態となります。
  REN が偽のときは、OUTPUT 命令での GPIB 機器の設定はできなくなる（GPIB でコントロールできない）ので注意が必要です。
  ・再び REN を真 (Low level) にするためには、REMOTE を実行して下さい。
  ・LOCAL に続いて装置アドレスを指定した場合、装置アドレスで指定された装置のみをアドレスし、リモート状態を解除します。
- 例　10　LOCAL
  20　LOCAL 1
  30　LOCAL 1, 2, 3
- 注意　ADDRESSABLE モードでは機能しません。

30. LOCAL LOCKOUT

- 概要　GPIB に接続されている装置を、パネル面からローカル状態にする機能を禁止します。
- 構文　(1)-1

(1)-2
LOCAL LOCKOUT

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

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

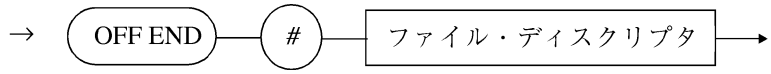
  この場合に LOCAL LOCKOUT を実行すると、GPIB 上の全装置のローカル・キーをロックして、完全に各装置のパネル面からの設定を禁止します。

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

  ・ローカル・ロックアウト状態を解除するには、LOCAL コマンドを用いて REN ラインを偽 (High level) にして下さい。
- 例　10　LOCAL LOCKOUT
- 注意　ADDRESSABLE モードでは機能しません。

31. OFF END

- 概要　ON END 文で指定したエンド・オブ・ファイル時の処理を解除します。
- 構文　(1)-1



(1)-2
OFF END # ファイル・ディスクリプタ

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

end of "DATAFILE" file

※　ファイルの取り扱いについては、2.4 ファイルの管理を参照して下さい。

32. OFF ERROR

- 概要　エラーが発生したときの分岐の機能、定義を解除します。
- 構文　(1)-1

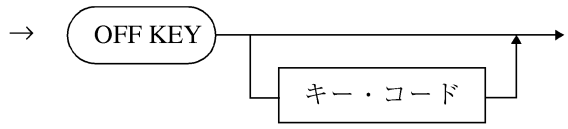
(1)-2
OFF ERROR

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

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

33. OFF KEY

- 概要　KEY 入力の割り込みによる分岐の機能、定義を解除します。
- 構文　(1)-1



(1)-2

OFF KEY〔キー・コード〕

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

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

34. OFF SRQ, OFF ISRQ

- 概要　SRQ または ISRQ の割り込みによる分岐の機能、定義を解除します。
（OFF SRQ はコントローラ・モード時のみ有効）
- 構文　(1)-1

(1)-2
OFF SRQ

(2)
OFF ISRQ は OFF SRQ と同様です。

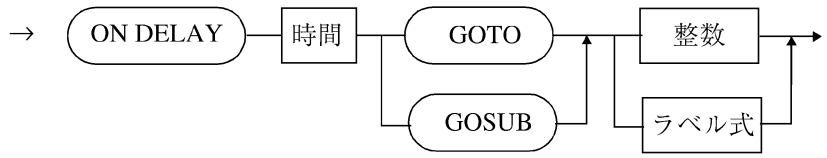
- 解説　・ OFF SRQ
ON SRQ によって許可された割り込みによる分岐を禁止します。
・ OFF ISRQ
ON ISRQ によって許可された割り込みによる分岐を禁止します。
- 例

```
100  OUTPUT 31;"OLDC OFF"
110  OUTPUT 31;"STAT:OPER:ENAB 8;*SRE 128":SPOLL(31)
120  ON ISRQ GOTO *MAX
130  OUTPUT 31;"INIT:CONT OFF;:ABOR;:INIT"
140  ENABLE INTR
150  ! LOOP
160  GOTO 140
170  *MAX
180  DISABLE INTR
190  OFF ISRQ
200  PRINT MAX(0, 1200, 0)
210  STOP
```

アドレス	内容
100	SRQ を ENABLE
110	内部 SRQ の割り込み分岐を設定
120	シングル掃引
130	割り込み受け付け
170	割り込み禁止
180	内部 SRQ の割り込み分岐を解除
190	最大レベルを表示

35. ON DELAY

- 概要　指定時間経過後に分岐します。
- 構文　(1)-1



(1)-2

ON DELAY 時間 <GOTO | GOSUB> < 整数 | ラベル式 >

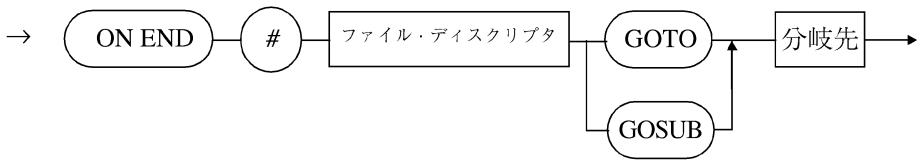
注　時間の単位は msec で、設定範囲は 0 ～ 65535 です。

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

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

36. ON END

- 概要　エンド・オブ・ファイル時の処理（分岐先）を定義します。
- 構文　(1)-1



(1)-2

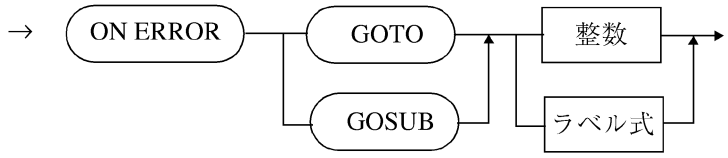
ON END # ファイル・ディスクリプタ <GOTO | GOSUB> 分岐先

- 解説
  - ・ ENTER でファイルからデータを読み込みますが、ファイルの終わりまで読み込んで入力するデータがない場合に、エンド・オブ・ファイルになります。
    ON END 文で処理を宣言しておかないと、ファイルをクローズした後に、エラー・メッセージを表示して実行を停止します。
  - ・ 分岐先を、数値変数、数値定数またはラベルで指定します。

※　ファイルの取り扱いについては、2.4 ファイルの管理を参照して下さい。

37. ON ERROR

- 概要　エラーが発生したときの分岐を許可します。
- 構文　(1)-1



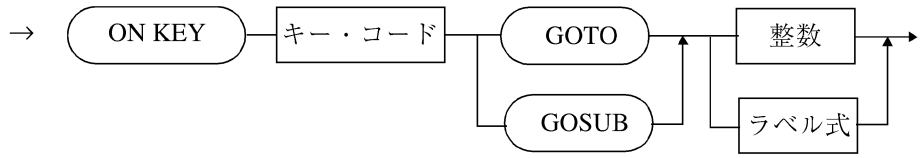
(1)-2

ON ERROR <GOTO | GOSUB> < 整数 | ラベル式 >

- 解説　・ BASIC プログラムの実行中にエラーが発生すると、そこの文番号とエラー・メッセージを表示してプログラムを停止します。
  特に、計測器のサービスを要求するビルトイン関数のエラーの際には、エラー・メッセージを表示するだけで実行し続けます。これらを検出して分岐する場合には、ON ERROR 文を使用します。
  ・ 分岐先は、数値定数、数値変数またはラベルで指定します。
  発生したエラーを分類するために、エラー番号を記憶した ERRN システム変数が用意されています。
  ・ エラーが発生した後に、そのエラー処理で確実に回復できないと永久ループになってしまいます。これを防ぐには、OFF ERROR 文を入れます。
- 例　ON ERROR GOTO 1000

38. ON KEY

- 概要　KEY 入力の割り込みによる分岐を許可します。
- 構文　(1)-1



(1)-2

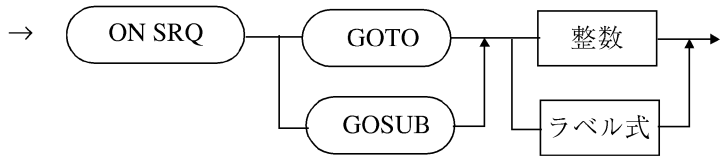
ON KEY キー・コード <GOTO | GOSUB> < 整数 | ラベル式 >

- 解説
  - ・ プログラム実行中に KEY 入力の割り込みで分岐します。
  - ・ 分岐は、割り込みが発生したときに、実行していたステートメントの処理が終了してから行われます。
  - ・ サブルーチンへ分岐した場合の戻り先は、割り込みが発生したときに実行していたステートメントの次のステートメントとなります。
  - ・ キー・コードは 1 ～ 6 までの数値で、正面パネル上のファンクション・キーとキー・ボード上の F1 ～ F6 に対応しています。
  - ・ ENABLE INTR で割り込みの受け付けを許可しておく必要があります。
- 例

```
10   CLS                          1010  GOTO *HERE
20   ON KEY 1 GOTO 1000           1100  PRINT "SECOND KEY"
30   ON KEY 2 GOTO 1100           1101  CNT = 10
40   ON KEY 3 GOTO 1200           1110  GOTO *HERE
50   ON KEY 4 GOTO 1300           1200  PRINT "THIRD KEY"
60   ON KEY 5 GOTO 1400           1201  CNT = 20
70   ON KEY 6 GOTO 1500           1210  GOTO *HERE
75   CNT=10                       1300  PRINT "FOURTH KEY"
80   *HERE:                       1301  CNT = 30
85   I =    0: PRINT " "          1310  GOTO *HERE
90   IF I=CNT THEN GOTO *HERE     1400  PRINT "FIFTH KEY"
100  ++I: PRINT ">" ;             1401  CNT = 40
110  ENABL EINTR                  1410  GOTO *HERE
120  GOTO 90                      1500  PRINT "SIXTH KEY"
1000 PRINT "FIRST KEY"            1501  CNT = 50
1001 CNT = 1                      1510  GOTO *HERE
```

39. ON SRQ, ON ISRQ

- 概要　ON SRQ は、GPIB 外部 SRQ 信号による割り込み分岐を許可します。
（ON SRQ コントローラ・モード時のみ有効）
ON ISRQ は、内部割り込み要因が発生したときの割り込み分岐を許可します。
- 構文　(1)-1



(1)-2

ON SRQ <GOTO | GOSUB> < 整数 | ラベル式 >

(2)

ON ISRQ は ON SRQ と同様です。

- 解説　・ プログラム実行中の割り込みで分岐します。
・ 分岐は割り込みが発生したときに実行していたステートメントの処理が終了してから行われます。
・ サブルーチンへ分岐した場合の戻り先は、割り込みが発生したときに実行していたステートメントの次のステートメントになります。
・ ONSRQ は、コントローラ・モードで実行時のみ GPIB 外部からの SRQ 信号で割り込み分岐します。
・ENABLE INTR で割り込みの受け付けを許可しておく必要があります。

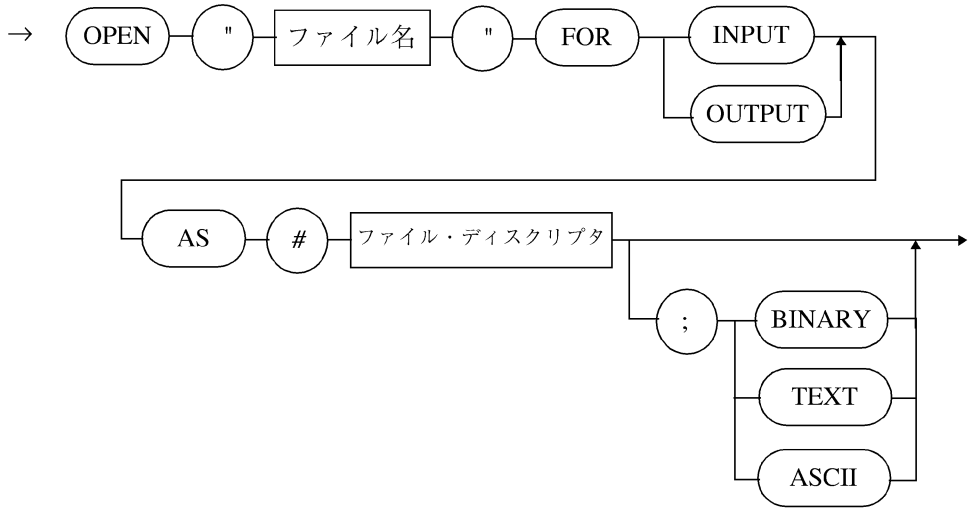
- 例　　シングル掃引ごとに、MAX をサーチするプログラム

```
100   OUTPUT 31; "OLDC OFF"
110   ON ISRQ GOTO *MAX
120   OUTPUT 31; "STAT:OPER:ENAB 8; *SRE 128" :SPOLL(31)
130   ENABLE INTR
135   OUTPUT 31; "INIT:CONT OFF; :ABOR; :INIT"
140   ! LOOP
150   GOTO 140
160   *MAX
170   DISABLE INTR:SPOLL(31)
180   PRINT MAX(0, 1200, 0)
190   GOTO 130
```

アドレス	内容
110	内部 SRQ の割り込み分岐を設定
120	SRQ を ENABLE
130	割り込み受け付け
135	シングル掃引
170	割り込み禁止
180	最大レベルを表示

40. OPEN

- 概要　ファイルに対しファイル・ディスクリプタを割り当て、指定した処理モードでオープンします。
- 構文　(1)-1



(1)-2

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

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

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

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

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

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

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

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

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

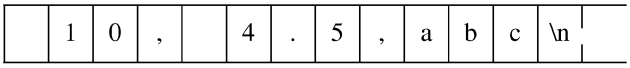
・ 既に他のファイルに割り当てられているファイル・ディスクリプタをオープンすると、前に割り当てられていたファイルをクローズして、指定されたファイルを新しくオープンします。

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

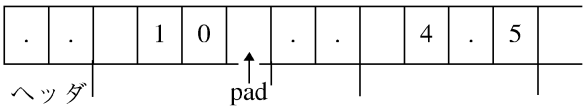
※ ファイルの取り扱いについては、2.4 ファイルの管理を参照して下さい。

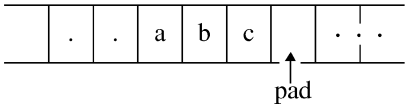
• 例

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



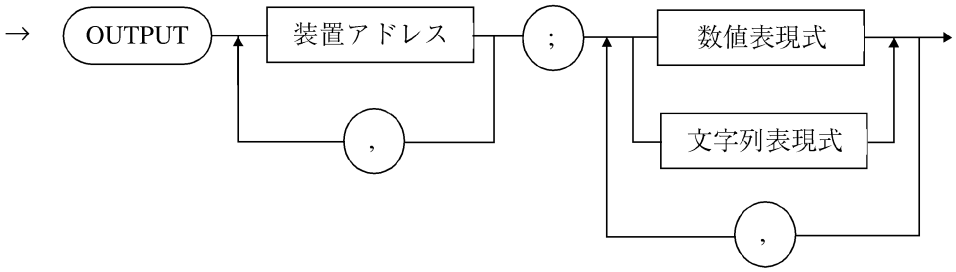

```
10  OPEN "DATA. BAS" FOR OUTPUT AS #FD ; ASCII
20  OUTPUT #FD;10, 4.5, "abc"
```





41. OUTPUT

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



(1)-2

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

装置アドレス :

0 ～ 30 ; 外部 GPIB 接続機器のアドレス

31 ; 本器の測定部への出力

33 ; パラレル・ポートの A ポートへの出力

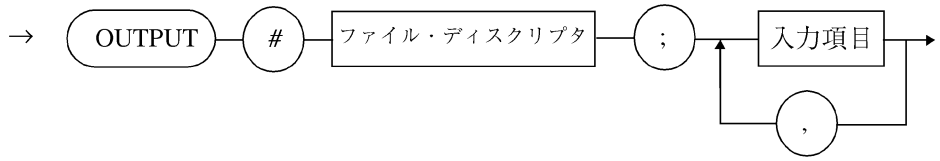
34 ; パラレル・ポートの B ポートへの出力

35 ; パラレル・ポートの C ポートへの出力および Flip/Flop のセット / リセット

36 ; パラレル・ポートの D ポートへの出力およびポートのモード設定

37 ; パラレル・ポートの CD ポートへの出力

(2)-1



(2)-2

OUTPUT # ファイル・ディスクリプタ ; 入力項目 {, 入力項目 }

- 解説　(1) の構文
  ・ 装置アドレスによって指定された装置へ、数値および文字列を ASCII データとして送ります。
  装置アドレスは、カンマ (,) で区切って複数を指定できます。
  また、数値表現式と文字列表現式もカンマで区切ると、混在して使えます。

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

・ GPIB 出力時の終端コードは DELIMITER 文で指定されます。
4.3 節の 9. DELIMITER も参照して下さい。

・ 例

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

・ 注意

SYSTEM CONTROLLER モード時は、指定アドレスの機器をリスナに指定し、データを出力します。
外部に指定したリスナがない場合、このコマンドは実行しません。

(2) の構文

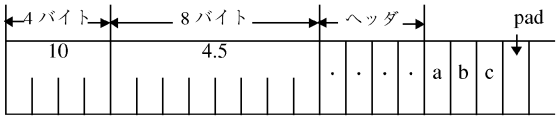
・ ファイル･ディスクリプタに割り当てられているファイルに､出力項目をBASICの標準の書式に変換してから出力します。
このデータタイプの形式で読み込んで、その入力項目に代入します。

・ 例 1　BINARY ファイル

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

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

※　ファイルの取り扱いについては、2.4 ファイルの管理を参照して下さい。

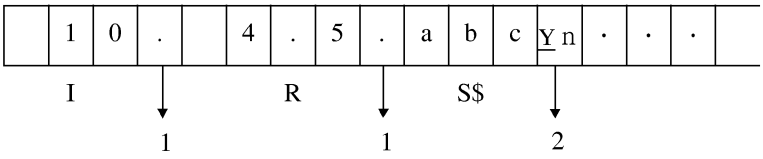


ヘッダはデータの長さを持っています。

・ 例 2　TEXT ファイル

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

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



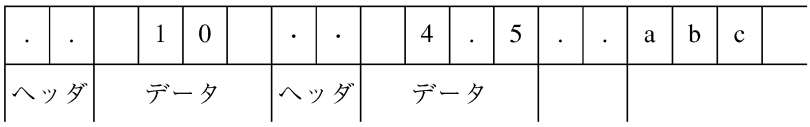
1: 各項目はカンマ (,) で区切られます。

2: 最後の項目の後にはライン・フィードが出力されます。

・ 例 3　ASCII ファイル

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

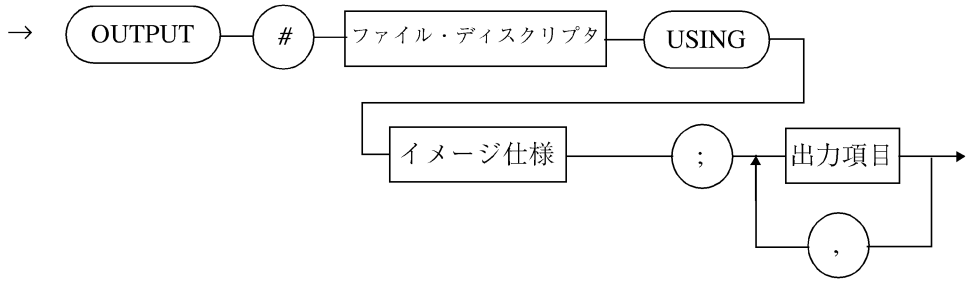
```
10  OPEN "FILE" FOR INPUT #FD;ASCII
20  OUTPUT #FD;10, 4.5, "abc"
```



ヘッダはデータの長さを持ちます。

42. OUTPUT USING

- 概要　#ファイル・ディスクリプタに割り当てられているファイルにデータを指定された形式で出力（書き込み）します。
- 構文　(1)-1



(1)-2

OUTPUT #ファイル･ディスクプリタ USING イメージ仕様 ; 出力項目 {, 出力項目 }

注　OUTPUT は OUT、USING は USE と省略できます。

- 解説　・USING とイメージ仕様を指定すると、自由に書式を変換して出力します。イメージ仕様は、文字列式で指定します。
・ファイル・ディスクリプタは、ファイル・オープン時に指定したものを使います。オープン時に、処理の対象になるファイルに対して、ファイル・ディスクリプタを割り当てます。以後、そのファイルに対する処理はすべてこのファイル・ディスクリプタを介して行います。

イメージ仕様

D　：D の数で数値を出力するときの桁数を指定する。
指定したフィールドで空いた部分にはスペースが入る。

Z　：Z の数で数値を出力するときの桁数を指定する。
指定したフィールドで空いた部分には 0 が入る。

K　：式の値を BASIC の標準形式（PRINT と同じ）で出力する。

S　：S の位置にプラス (+) かマイナス (-) を出力する。

M　：M の位置に、負のときはマイナス (-) を、正ならばスペースを出力する。

.　：. の位置に小数点がくるように位置を合わせる。

E　：e 符号指数という書式で出力する。

H　：K と同じ。ただし、小数点にカンマ (,) を使う。

R　：. と同じ。ただし、小数点にカンマ (,) を使う。

*　：* の数で数値の出力時の桁数を指定する。
指定したフィールドで空いた部分には * を出力する。

A　：A の部分に 1 文字出力する。

k　：文字列式の値をそのまま出力する。

リテラル:\" で囲まれた文字列を出力項目とは無関係にそのまま出力する。

X　：X の位置に 1 つスペースをとる。

B　：式の値を ASCII コードとして出力する。

@　：フォーム・フィードを出力する。

+　：キャリージ・リターンを出力する。

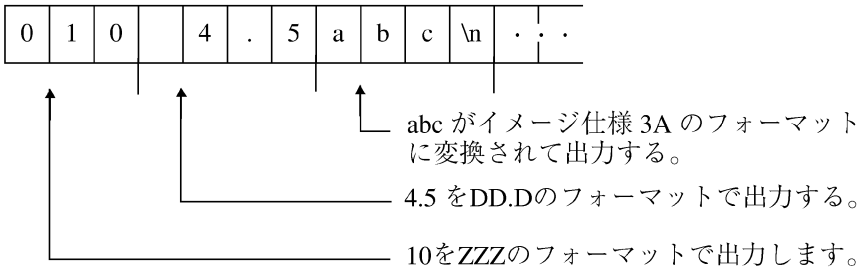
-　：ライン・フィードを出力する。

#　：最後の項目の後ろには、自動的にライン・フィードが付くが、このイメージ仕様を指定するとライン・フィードが付かない。

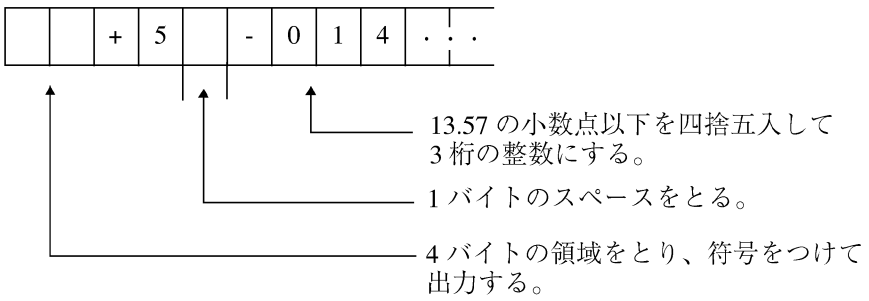
n　：数字で各イメージ仕様の繰り返し指定の回数を指定できる。
3D. 2D は DDD. DD と同じで、4A は AAAA と同じ。

※　ファイルの取り扱いについては、2.4 ファイルの管理を参照して下さい。

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

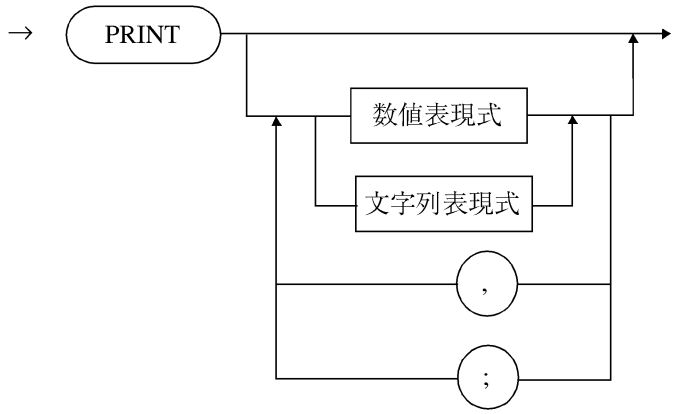


OUTPUT #FD USING "SDDD, X, MZZZ"; +5, -13.57



43. PRINT [USING]

- 概要　数値または文字列を表示します。
- 構文　(1)-1



(1)-2

PRINT〔数値表現式 I 文字列表現式 {, I ; 数値表現式 I 文字列表現式 }〕

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

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

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

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

- 例 1
```
10    PRINT USING "4Z, 2X, 5D, 2X, 5*" ;123, -444, 567
< 実行結果 >
0123          -444     **567
```

- 例 2
```
10    PRINT USING "S3D, X, S3D" ; -4.5, 465
20    PRINT USING "M3Z. Z, X, M3ZR3Z" ; 1 .26, -5. 452
< 実行結果 >
  -5 +465
001. 3   -005, 452
```

- 例 3
```
10    PRINT USING "K, X, H" ; 5. 03884e+22, 4. 5563
< 実行結果 >
5. 03884e+22 4, 5563
```

- 例 4

```
10    PRINT USING "k, #" ; "character:"
20    PRINT USING "B" ; 69
< 実行結果 >
character:E
```

- 例 5

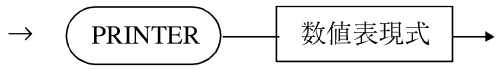
```
10    PRINT USING "\".....\", +, A" ; "*"
20    PRINT USING "k, -, \".END. \" "; "string"
< 実行結果 >
*.....
string
.END.
```

- 例 6

```
100   PRINT USING "DDD. DD" ; 1. 2
110   PRINT USING "ZZZ. ZZ" ; 1. 2
120   PRINT USING "K" ; 1. 2
130   PRINT USING "SDDD. DD" ; 1. 2
140   PRINT USING "MDDD. DD" ; 1. 2
150   PRINT USING "MDDD. DD" ; -1. 2
160   PRINT USING "H" ; 1. 2
170   PRINT USING "DDDRDD" ; 1. 2
180   PRINT USING "***. **" ; 1. 2
190   PRINT USING "A" ; "a"
200   PRINT USING "k" ; "string"
210   PRINT USING "B" ; 42
220   PRINT USING "3D.2D" ; 1. 2
< 実行結果 >
  1.20
001.20
1.2
 +1.20
  1.20
 -1.20
1, 2
  1, 20
**1.20
a
string
*
  1.20
```

44. PRINTER

- 概要　プリンタに送る装置アドレスを指定します。
- 構文　(1)-1



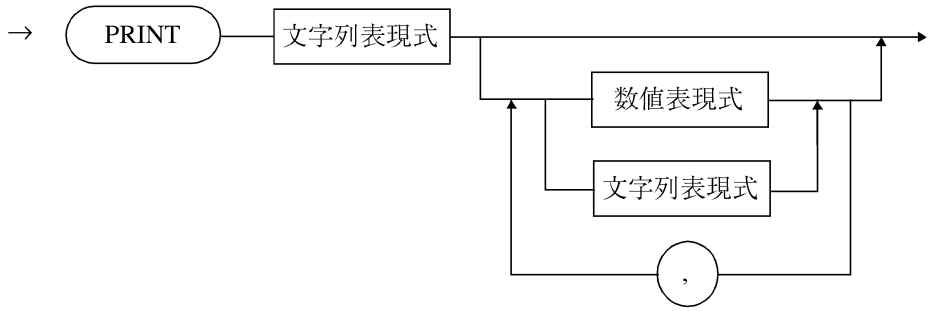
(1)-2

PRINTER 数値表現式

- 解説　・GPIB に接続されるプリンタの装置アドレスを設定します。
  ・PRINT を実行する前に、必ず PRINTER でプリンタの装置アドレスを本器に指示して下さい。
  ・装置アドレスは、0 ～ 30 までの整数です。
- 例　　10　PRINTER 1

45. PRINTF

- 概要　数値または文字列を表示します。
- 構文　(1)-1



(1)-2

PRINTF 文字列表現式〔数値表現式 | 文字列表現式
{, 数値表現式 | 文字列表現式 }〕

- 解説　・指定された数値、文字列を表示します。
  ・数値、文字列をカンマ (,) で区切って複数を指定すると、改行せずに数値、文字列を次々に出力します。改行する場合は \n を書式指定式の中で指定します。
  ・第 1 パラメータの文字列表現式が、その後のパラメータの書式を指定するために使われます。
  ・書式指定の方法は以下の通りです。

- PRINTF 書式指定式〔〔式〔式〔...]]〕

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

  %〔-〕〔0〕〔m〕〔.n〕文字

  - ..... 指定されたフィールド内で左詰めにする。この指定がなければ右詰めにする。

  0 ..... 指定フィールドの余った部分に詰める文字を、スペースでなく 0 にする。

  m ..... m 文字分のフィールドを取る。

  .n ..... n 桁の精度で出力する。文字列に対して指定すると、この値が実際の文字列の長さになる。

  文字 ..... d ; 符号付 10 進数　s ; 文字列
  o ; 8 進数　e ; 浮動小数点表現 (指数形式)
  x ; 16 進数　f ; 浮動小数点表現

- 例

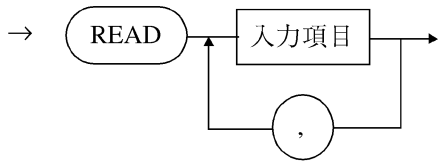
```
10    N = 500000
20    U = LOG(1+1/N)
30    V = U - 1 / N
40    PRINTF "%7d %16. 5e %16.5e\n", N, U, V
50    PRINTF "%s\n", "end"
```

< 実行結果 >

```
  500000 2. 00000e-06 -1.99994e-12
end
```

46. READ

- 概要　DATA 文の定数を、変数に代入します。
- 構文　(1)-1



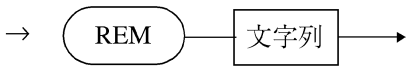
(1)-2

READ 入力項目 {, 入力項目 }

- 解説
  - DATA 文で定義されている数値、文字列を、引数で指定してある変数に読み込みます。
  - READ 文が現れたところで、プログラムの中から DATA 文を捜します。
  - 最初の READ では、原則として（RESTORE 文で変更されていなければ）、プログラムの先頭行から行番号順に捜して、最初に発見した値を引き数並びの変数に代入します。
    その後、順に対応する DATA 文の定数を捜して代入します。
  - READ の変数に対して、DATA で指定する定数の数の方が少ない場合には、エラーとなります。
  - 対象は、READ で読み出そうとする変数の数と、それに対する DATA 文の定数の数で、DATA 文や READ 文の行数は関係ありません。

47. REM

- 概要　プログラムの注釈です。
- 構文　(1)-1



(1)-2

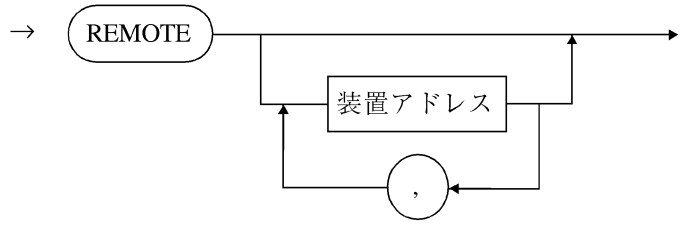
REM 文字列

- 解説
  - ・プログラム中に注釈をつけたいときに使用します。
  - ・REM は非実行ステートメントですから、REM に続く文字列はいかなるものでも構いません。すべての文字、数字、記号が使用できます。
  - ・REM は、感嘆符 (!) で代用できます。
  - ・REM の後にコロン (:) によるマルチ・ステートメントは使用できません。すべて注釈文として見なされます。
- 例

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

48. REMOTE

- 概要　指定した装置をリモート状態にするか、またはリモート・イネーブル (REN) ラインを真にします。
- 構文　(1)-1



(1)-2

REMOTE〔装置アドレス {, 装置アドレス }〕

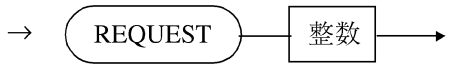
- 解説　・装置アドレスを指定せずに REMOTE だけを実行した場合、GPIB のリモート・イネーブル (REN) ラインが真 (Low level) となり、GPIB 上に接続された装置をリモート・コントロール可能な状態にします。
  REN ラインを偽 (High level) にするためには、LOCAL を実行して下さい。
  ・REMOTE に続いて装置アドレスを指定した場合、装置アドレスで指定された装置のみをリモート状態にします（ただし、REN ラインが真のときのみ)。
  装置アドレスは複数を指定できます。
  またリモート状態を解除するためには、LOCAL を実行して下さい。
  ・REMOTE は、選択した装置をリモート状態にするものですが、以下に示すステートメントを実行したときは、REMOTE を実行しなくても自動的に指定した装置をリモート状態にします（ただし、REN ラインが真のときのみ)。

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

- 例　10　REMOTE 1
  20　REMOTE 5
  30　REMOTE 1 2 3
- 注意　ADDRESSABLE モードでは機能しません。

49. REQUEST

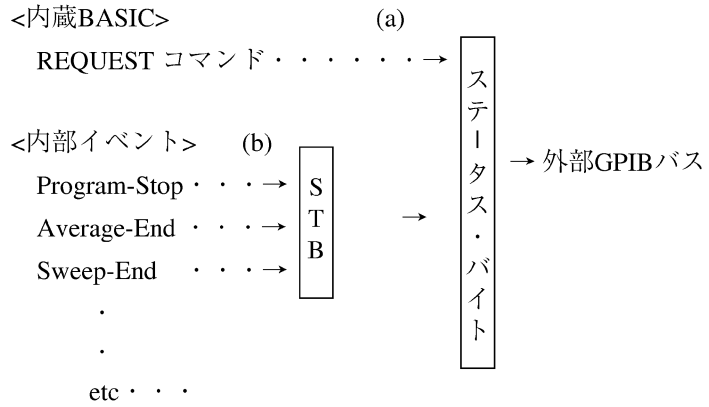
- 概要　ADDRESSABLE モード時に外部 GPIB コントローラへ送信するステータス・バイトを設定します。
- 構文　(1)-1



(1)-2
REQUEST 整数

注　　整数の設定範囲は 0 ～ 255 です。

- 解説　・ ADDRESSABLE モード時に外部 GPIB コントローラへ送信するステータス・バイトを設定します。
  ・ サービス・リクエストを発信する場合は、64 ～ 127 または 192 ～ 255（ビット 6 が 1）の値を設定します。
- 例　10　REQUEST 65
- 注意　・ SYSTEM CONTROLLER モードでは機能しません。
  ・ 外部コントローラからはシリアルポールを用いて読んで下さい。GPIB コマンドの *STB? では読めません。
  ・ GPIB コマンドの SRQD が実行されている場合は、ステータス・バイトのビット 6 は常に 0 で送信されます。
  したがって、サービス・リクエストも発信されません。
  ・ ステータス・バイト注意事項
  ステータス・バイトの出力経路は、下図のように 2 系統あります。



(a)　外部 GPIB バス上に出力されるステータス・バイトです。
シリアルポールで読み出し可能です。（読み出されると bit6 (RQS) は 0 となります）

(b)　内部イベント用のステータス・バイト・レジスタに相当します。
*STB?実行で読み出し可能です。(読み出されても bit6 (MSS)は変化しません)

注　(a) に出力されるのは < 内蔵 BASIC> または < 内部イベント > どちらか最後に発生した方となります。

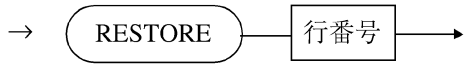
< 内蔵 BASIC> の REQUEST コマンド実行の場合は、指定した値が直接 (a) に出力されます。
< 内部イベント > 発生の場合は、(b) のレジスタにビットの変化があった場合のみ (b) 内容が (a) にコピー（出力）されます。

(b) のステータス・バイト・レジスタは、前段の各レジスタのイネーブル・レジスタ設定によりビット変化をマスクできます。しかし MAV ビット (bit4) だけはマスクできません。この MAV ビットは、クエリ・コマンドを受信した時点で 1 となり、クエリ・データ出力後に 0 となります。（内蔵 BASIC での実行も含む）要するにクエリ処理実行時は、必ずビットの変化が発生します。そのため REQUEST コマンドで (a) にステータス・バイトを出力してもシリアル・ポール実行前にクエリ・コマンドが実行されると、(b) の内容が (a) にコピー（出力）されてしまいます。

REQUEST コマンドを併用する場合のステータス・バイトの明示的なクリアは、*CLS 実行に加え、"REQUEST O" も実行して下さい。
*CLS の実行でクリアされるのは上記 (b) 以前のレジスタまでなので、(b) がすでに O である場合は、ビットの変化が発生しないため (a) には反映（出力）されません。

50. RESTORE

- 概要　次の READ 文で読み込む DATA 行を指定します。
- 構文　(1)-1

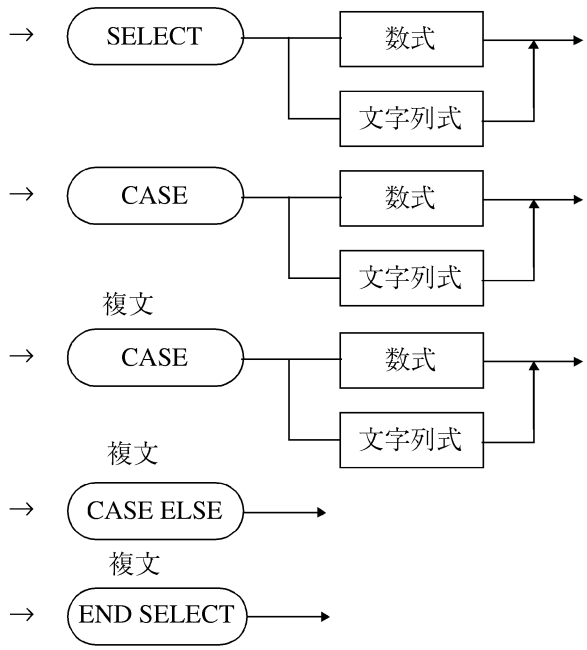


(1)-2
RESTORE 行番号

- 解説
  - 行番号は、数式またはラベルで指定します。
    特に指定がなければ、プログラムの先頭行から順番に DATA 文の定数が読み込まれ、RESTORE で次の READ の対象となる DATA 文を指定できます。
  - 引数の行番号が DATA 文を捜し始める先頭行という判断をするので、その行以降の最初の DATA 文が指定するものであれば構いません。

51. SELECT, CASE, END SELECT

- 概要　1 つの式の値を条件として、複数の分岐を行います。
- 構文　(1)-1



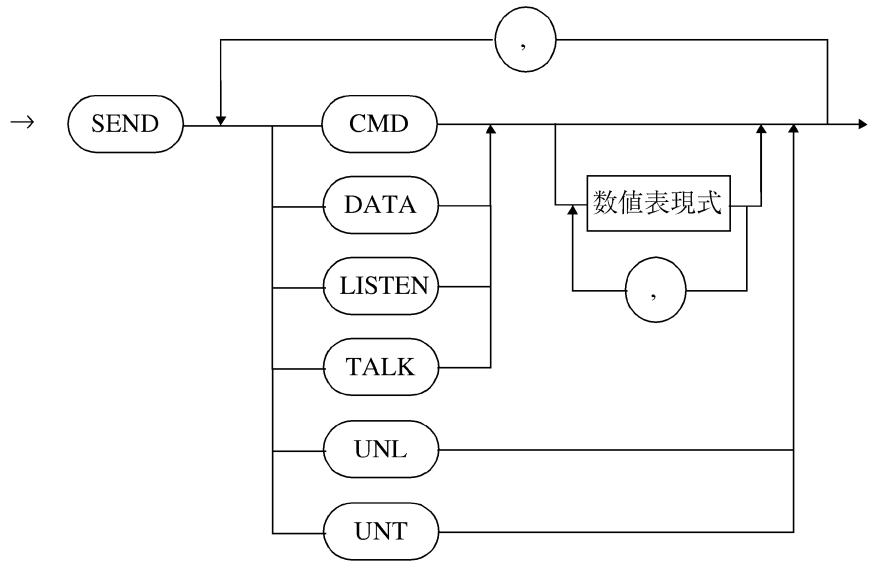
(1)-2

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

- 解説　・ SELECT で指定した式の値に、一致する CASE 以下の文（複文）を実行します。実行の対象は、次の CASE、CASE ELSE、または END SELECT までです。
・ SELECT 構文自体の入れ子ができます。この場合内部の SELECT は、完全に外部のものを含む形になります。

52. SEND

- 概要　GPIB にコマンドおよびデータを出力します。
- 構文　(1)-1



(1)-2

SEND <A | B>{, <A | B>}

注	A:<CMD | DATA | LISTEN | TALK>〔数値表現式 {, 数値表現式 }〕 B:UNL | UNT

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

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

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

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

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

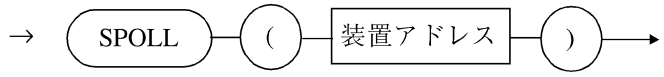
UNT　：アントーク (UNT) コマンドを GPIB に送ります。このコマンドを実行する前にトーカに指定されていた装置は、トーカを解除されます。

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

- 例　10　SEND UNT UNL LISTEN 1, 2, 3 TALK 4
  20　SEND UNT CMD 63, 33 DATA 30, 54
- 注意　ADDRESSABLE モードでは機能しません。

53. SPOLL

- 概要　指定した装置のシリアル・ポールを行い、ステータス・バイトを読み込みます。
- 構文　(1)-1

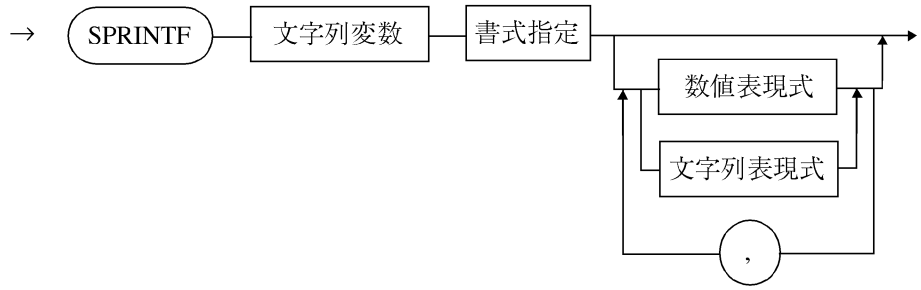


(1)-2
SPOLL（装置アドレス）

- 解説　・本器が SYSTEM CONTROLLER モードのとき、他の GPIB 装置に対してシリアル・ポールを行います。
  ・装置アドレスが 0 ～ 30 のときは、各アドレスに対応した装置のシリアル・ポールを行います。
  ・装置アドレスが 31 のとき、SYSTEM CONTROLLER モード、ADDRESSABLE モードに関係なく、本器に対してステータス・バイトを取り出します。
- 例　10　OUTPUT 31; "OLDC ON"
  20　ON ISRQ GOTO 70
  30　ENABLE INTR
  40　OUTPUT 31; "SRQE"
  50　OUTPUT 31; "SINGLE"
  60　GOTO 60
  70　PRINT SPOLL(31)
  80　STOP
- 注意　ADDRESSABLE モード時に装置アドレス 0 ～ 30 を指定し、SPOLL を行った場合は、0 が返ります。

54. SPRINTF

- 概要　PRINTF コマンドの書式変換仕様にしたがって書式を変換し、文字列変数に結果を代入します。
- 構文　(1)-1



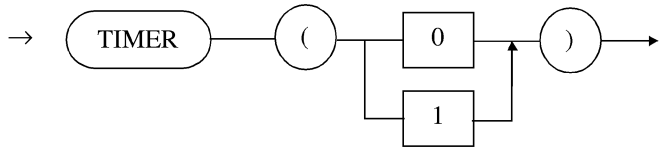
(1)-2

SPRINTF 文字列変数書式指定〔数値表現式 | 文字列表現式
{, 数値表現式 | 文字列表現式 }〕

- 解説　・ PRINTF の書式変換の方法で式の値を変換して、最初のパラメータの文字列変数に結果を代入します。
・ 書式指定の方法と式の数、それに結果を入れる文字列変数の大きさには十分な注意が必要です。
特に結果をいれる文字列が結果に対して十分な大きさがないと、BASIC バッファを破壊する恐れがあります。
書式の指定方法は、4.3 節の 45. PRINTF を参照して下さい。

55. TIMER

- 概要　内部システム時間の読み出しおよびリセットをします。
- 構文　(1)-1



(1)-2
TIMER (0 | 1)

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

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

・読み出した値は 10msec の分解能で、±10msec の誤差があります。

- 例

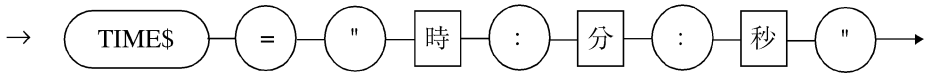
```
10   INTEGER I
20   TIMER(1)
30   FOR I=0 TO 10000
40   NEXT I
50   T1=TIMER(0)
60   !
70   TIMER(1)
80   FOR I=0 TO 10000
90       PRINT I
100  NEXT I
110  T2=TIMER(0)
120  !
130  PRINT "PRINT Command execute time is" ; T2-T1
140  STOP
```

56. TIME$

- 概要　時刻の読み出しおよび設定をします。
- 構文　(1)-1

(1)-2
TIME$

(2)-1



(2)-2
TIME$=" 時 : 分 : 秒 "

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

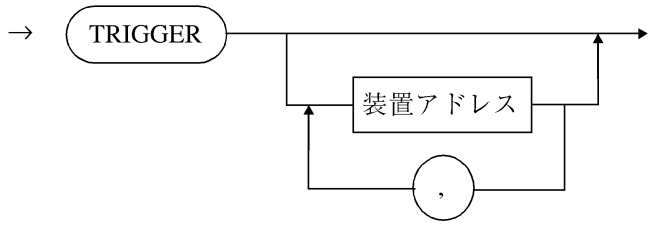
  TIME$= "23:43:12"
  TIME$= "11:5:6"

- 例

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

57. TRIGGER

- 概要　GPIB 上に接続されているすべての装置、または選択された特定の装置にアドレス・コマンド・グループ (ACG) のグループ・エグゼキュート・トリガ (GET) を送ります。
- 構文　(1)-1



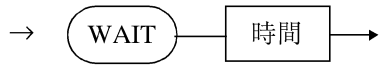
(1)-2

TRIGGER〔装置アドレス {, 装置アドレス }〕

- 解説　・装置アドレスを指定せずに TRIGGER だけを実行すると、GPIB にはアドレス・コマンドのグループ・エグゼキュート・トリガ (Group Execute Trigger-GET) のみが送られます。この場合、トリガをかけたい装置はあらかじめリスナに設定しておかなければなりません。
  ・TRIGGER に続いて装置アドレスを指定すると、装置アドレスで指定された装置のみに GET コマンドを送ります。
- 例　10　TRIGGER 1
  20　TRIGGER
- 注意　ADDRESSABLE モードでは機能しません。

58. WAIT

- 概要　指定時間だけ待ちます。
- 構文　(1)-1



(1)-2

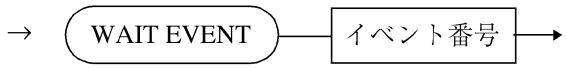
WAIT 時間

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

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

59. WAIT EVENT

- 概要　指定したイベントが発生するまで待ちます。
- 構文　(1)-1



(1)-2

WAIT EVENT イベント番号

- 解説　指定されたイベント番号のイベントが発生するまで待ちます。
  イベント番号 : 1; スイープエンド
- 例

```
10    INTEGER EV
20    EV=1
25    OUTPUT 31;"OLDCOFF"
30    OUTPUT 31;"INIT:CONT OFF;:ABOR;INIT"
40    WAIT EVENT EV
50    PRINT "SWEEP FINISHED"
60    STOP
```

4.4 ビルトイン関数

4.4.1 概要

ビルトイン関数は、ネットワーク・アナライザで測定したデータを、高速処理できる組み込み関数で、測定データの解析や判定に使用します。

従来のネットワーク・アナライザ R3751 と基本的な関数は同じですが、一部追加や削除した関数がありますので注意して下さい。また、処理スピードも向上しています。

ビルトイン関数で扱う数値には単位を指定できません。すべて基準単位として処理されます。

例　10KHz のアドレス・ポイントを求める場合

```
P = POINT2 (10000, 0)
```

また、ビルトイン関数が返す応答データも同様に基準単位の数値となります。

1. 測定ポイントとアドレス・ポイント
   測定データの解析範囲の指定や、測定データの中の位置指定には、アドレス・ポイントを使用します。アドレス・ポイントは、0 ～ 1200 の値で測定データの指定を行います。以下のような対応づけになります。
   - 測定ポイント数が 1201 のとき
     データごとにアドレス・ポイントが 1 増加します。

最初のデータ	アドレス・ポイント 0
2 番目のデータ	アドレス・ポイント 1
3 番目のデータ	アドレス・ポイント 2
：	
：	
n 番目のデータ	アドレス・ポイント n-1
：	
：	
1201 番目のデータ	アドレス・ポイント 1200

   - 測定ポイント数が 801 のとき
     アドレス・ポイント 801 ～ 1201 のデータは無効となります。

- 測定ポイント数が 601 のとき
  データごとにアドレス・ポイントが 2 増加します。

  最初のデータ　　アドレス・ポイント 0
  2 番目のデータ　　アドレス・ポイント 2
  3 番目のデータ　　アドレス・ポイント 4
  　：
  　：
  n 番目のデータ　　アドレス・ポイント 2(n-1)
  　：
  　：
  601 番目のデータ　　アドレス・ポイント 1200
- 測定ポイント数が 301 のとき
  データごとにアドレス・ポイントが 4 増加します。

  最初のデータ　　アドレス・ポイント 0
  2 番目のデータ　　アドレス・ポイント 4
  3 番目のデータ　　アドレス・ポイント 8
  　：
  　：
  n 番目のデータ　　アドレス・ポイント 4(n-1)
  　：
  　：
  301 番目のデータ　　アドレス・ポイント 1200

  測定ポイント数とアドレス・ポイントの増加値の関係は、以下のようになります。

測定ポイント数	アドレス・ポイントの増加値	測定ポイント数	アドレス・ポイントの増加値
1201	1	101	12
801	1	51	24
601	2	21	60
401	3	11	120
301	4	6	240
201	6	3	600

ユーザ／プログラム掃引時も、この基準に準じます。測定ポイント数が 1201 でユーザ掃引にしたときは、アドレス・ポイントの増加値は必ず 1 になります。アドレス・ポイント 0 からつめてデータを配置します。測定データ・ポイント数を 601 にし、セグメントのポイント数の合計が 601 を超えない限り、アドレス・ポイントの増加値は 2 となり、測定データは 1 ポイントおきに配置されます。これによりアドレス・ポイントで指定するとき、測定ポイント数が変更されてもビルトイン関数の指定は変更不要です。

ユーザ／プログラム掃引時にセグメントのポイント数の合計が測定ポイント数より小さい場合には余った測定ポイントのデータは無効となります（測定ポイント数 801 の場合と同様）。逆にセグメントのポイント数の合計が測定ポイント数より大きくなった場合には測定ポイントは自動的に変更されます。例えば、測定ポイントが 201 のときにセグメントのポイント数の合計が 250 になった（201 を超えた）場合、自動的に測定ポイントは 301 となり、アドレス･ポイント 1000 ～ 1200 のデータは無効となります。

例のように測定ポイントが 301 に変更されてから、合計ポイント数を 201 以下に設定し直しても測定ポイント数は元に戻りません。

上記の測定ポイント数とアドレス・ポイントの関係は R3751 と同様です。

2. 解析チャンネル
解析のチャンネル指定は、以下のようになります。

チャンネル	内容
0	CH1 の表示データ
1	CH2 の表示データ
2	CH1 のメモリデータ
3	CH2 のメモリデータ
8	CH1 の表示データ（LOGMAG&PHASE の位相データなど）
9	CH2 の表示データ（LOGMAG&PHASE の位相データなど）
10	CH1 のメモリデータ（LOGMAG&PHASE の位相データなど）
11	CH2 のメモリデータ（LOGMAG&PHASE の位相データなど）
32	CH1 の LOGMAG データ
33	CH1 の位相データ
34	CH1 の実数部データ
35	CH1 の虚数部データ
36	CH2 の LOGMAG データ
37	CH2 の位相データ
38	CH2 の実数部データ
39	CH2 の虚数部データ
40	CH1 のメモリの LOGMAG データ
41	CH1 のメモリの位相データ
42	CH1 のメモリの実数部データ
43	CH1 のメモリの虚数部データ
44	CH2 のメモリの LOGMAG データ
45	CH2 のメモリの位相データ
46	CH2 のメモリの実数部データ
47	CH2 のメモリの虚数部データ

3. ビルトイン関数の応答形式
ビルトイン関数の応答形式には 3 種類あります。

- 測定ポイント　　：測定データのあるアドレス・ポイント。
（例）MAX 関数
- アドレス・ポイント：測定ポイント以外のときは、補間してアドレス・ポイントでの値にする。
（例）VALUE 関数
- コンペンセート　　：アドレス・ポイントの間でも、補間して値を求める。
（例）CVALUE 関数

4.4.2 ビルトイン関数一覧

- アドレス・ポイント関係

  POINT1(F, C) : meas point ; 指定周波数に最も近い測定ポイント
  POINT2(F, C) : address point; 指定周波数に最も近いアドレス・ポイント
  DPOINT(F0, F1, C) : address point; 指定周波数幅に対応するアドレス・ポイント幅
  POINT1L(F, C) : meas point ; 指定周波数を越えない最も大きい測定ポイント
  POINT1H(F, C) : meas point ; 指定周波数より大きな最も小さな測定ポイント
  POINT2L(F, C) : address point; 指定周波数を越えない最も大きいアドレス･ポイント
  POINT2H(F, C) : address point; 指定周波数より大きな最も小さなアドレス･ポイント
  SWPOINT(C) : meas point ; 最新の測定ポイント

- 周波数関係

  FREQ(P, C) : address point; 指定アドレス・ポイントに対応する周波数
  DFREQ(P0, P1, C) : address point; 指定アドレス・ポイント幅に対応する周波数幅
  SWFREQ(C) : meas point ; 最新の測定周波数

- レスポンス関係

  VALUE(P, C) : address point; 指定アドレス・ポイントでのレスポンス値
  DVALUE(P0, P1, C) : address point; 指定アドレス・ポイント間のレスポンス値差
  CVALUE(F, C) : compensate ; 指定周波数でのレスポンス値
  DCVALUE(F0, F1, C) : compensate ; 指定周波数間でレスポンス値差
  SWVALUE(C) : meas point ; 最新のレスポンス値

- 最大値・最小値関係

  MAX(P0, P1, C) : meas point ; 指定アドレス・ポイント間の最大レスポンス値
  FMAX(P0, P1, C) : meas point ; 指定アドレス･ポイント間の最大レスポンスの周波数
  PMAX(P0, P1, C) : meas point ; 指定アドレス･ポイント間の最大レスポンスの測定ポイント
  MIN(P0, P1, C) : meas point ; 指定アドレス・ポイント間の最小レスポンス値
  FMIN(P0, P1, C) : meas point ; 指定アドレス･ポイント間の最小レスポンスの周波数
  PMIN(P0, P1, C) : meas point ; 指定アドレス･ポイント間の最小レスポンスの測定ポイント

- 帯域幅関係

  BND(P, X, C) : compensate ; 指定アドレス・ポイントから指定データ減衰した帯域幅

  BNDL(P, X, C) : compensate ; 指定アドレス・ポイントから指定データ減衰した低周波数側の周波数

  BNDH(P, X, C) : compensate ; 指定アドレス・ポイントから指定データ減衰した高周波数側の周波数

  CBND(F, X, C) : compensate ; 指定周波数から指定データ減衰した帯域幅

  CBNDL(F, X, C) : compensate ; 指定周波数から指定データ減衰した低周波数側の波数

  CBNDH(F, X, C) : compensate ; 指定周波数から指定データ減衰した高周波数側の周波数

  MBNDI(P0, P1, P, N, La, Fa, C):compensate ; 指定アドレス・ポイント間で、指定アドレス・ポイントから指定データ減衰した低周波数側の周波数、高周波数側の周波数、中心周波数、帯域幅

  MBNDO(P0, P1, P, N, La, Fa, C):compensate ; 指定アドレス・ポイント間で、指定アドレス・ポイントから指定データ減衰した低周波数側の周波数、高周波数側の周波数、中心周波数、帯域幅

- リップル関係 -1

  RPL1(P0, P1, dX, dY, C) : meas point ; 指定アドレス・ポイント間の極大値と極小値の差

  RPL2(P0, P1, dX, dY, C) : meas point ; 指定アドレス・ポイント間の隣あう極大値と極小値の差の最大値

  RPL3(P0, P1, dX, dY, C) : meas point ; 指定アドレス・ポイント間の隣あう極大と極小の差を加算した中の最大値

  RPL4(P0, P1, dX, dY, C) : meas point ; 指定アドレス・ポイント間の隣あう極大値と極小値の差の最大値

  RPL5(P0, P1, dX, dY, C) : meas point ; 指定アドレス・ポイント間の極大値の最大値

  RPL6(P0, P1, dX, dY, C) : meas point ; 指定アドレス・ポイント間の極大値の最小値

  RPLF(P0, P1, dX, dY, C) : meas point ; 指定アドレス・ポイント間の最初の極大点と極小点の周波数差

  RPLR(P0, P1, dX, dY, C) : meas point ; 指定アドレス・ポイント間の最初の極大点と極小点のレスポンス差

  RPLH(P0, P1, dX, dY, C) : meas point ; 指定アドレス・ポイント間の最初の極大値のレスポンス値

  FRPLH(P0, P1, dX, dY, C) : meas point ; 指定アドレス・ポイント間の最初の極大値の周波数

  PRPLH(P0, P1, dX, dY, C) : meas point ; 指定アドレス・ポイント間の最初の極大値の測定ポイント

  RPLL(P0, P1, dX, dY, C) : meas point ; 指定アドレス・ポイント間の最初の極小値のレスポンス値

FRPLL(P0, P1, dX, dY, C) : meas point ; 指定アドレス・ポイント間の最初の極小値の周波数
PRPLL(P0, P1, dX, dY, C) : meas point ; 指定アドレス・ポイント間の最初の極小値の測定ポイント

- リップル関係 -2

NRPLH(P0, P1, dX, dY, C) : meas point ; 指定アドレス・ポイント間の極大点の数
NRPLL(P0, P1, dX, dY, C) : meas point ; 指定アドレス・ポイント間の極小点の数
PRPLHN(N, C) : meas point ; NRPLHで求めたN番目の極大値の測定ポイント
PRPLLN(N, C) : meas point ; NRPLLで求めたN番目の極小値の測定ポイント
FRPLHN(N, C) : meas point ; NRPLH で求めた N 番目の極大値の周波数
FRPLLN(N, C) : meas point ; NRPLL で求めた N 番目の極小値の周波数
VRPLHN(N, C) : meas point ; NRPLHで求めたN番目の極大値のレスポンス値
VRPLLN(N, C) : meas point ; NRPLLで求めたN番目の極小値のレスポンス値
PRPLHM(Pa, C) : meas point ; NRPLH で求めた極大値の測定ポイント配列
PRPLLM(Pa, C) : meas point ; NRPLL で求めた極小値の測定ポイント配列
FRPLHM(Xa, C) : meas point ; NRPLH で求めた極大値の周波数配列
FRPLLM(Xa, C) : meas point ; NRPLL で求めた極小値の周波数配列
VRPLHM(Xa, C) : meas point ; NRPLH で求めた極大値のレスポンス値配列
VRPLLM(Xa, C) : meas point ; NRPLL で求めた極小値のレスポンス値配列

- ダイレクトサーチ関係

DIRECT(P0, P1, X, C) : address point; 指定アドレス・ポイント間で最初に検出されたデータの最も近いアドレス・ポイント
DIRECTL(P0, P1, X, C) : meas point ; 指定アドレス・ポイント間で低周波数側からサーチして最初に検出されたデータの測定ポイント
DIRECTH(P0, P1, X, C) : meas point ; 指定アドレス・ポイント間で高周波数側からサーチして最初に検出されたデータの測定ポイント
CDIRECT(F0, F1, X, C) : compensate ; 指定周波数間で最初に検出されたデータの周波数
CDIRECTL(F0, F1, X, C) : compemsate ; 指定周波数間で低周波数側からサーチして最初に検出されたデータの周波数
CDIRECTH(F0, F1, X, C) : compemsate ; 指定周波数間で高周波数側からサーチして最初に検出されたデータの周波数
DDIRECT(P0, P1, X, C) : address point; 指定アドレス・ポイント間で指定データのアドレス・ポイント幅
CDDIRECT(F0, F1, X, C) : compensate ; 指定周波数間で指定データの帯域幅
ZEROPHS(P0, P1, C) : compensate ; 指定アドレス・ポイント間のゼロ位相の周波数

- データ転送関係

  TRANSR(P0, P1, Xa, C)　：meas point　；指定アドレス・ポイント間の測定データの配列への転送

  TRANSW(P0, P1, Xa, C)　：meas point　；配列から指定アドレス・ポイントへの転送

  P, P0, P1　：アドレス・ポイント指定
  F, F0, F1　：周波数指定
  C　：解析チャンネル指定
  dX　：傾きの横軸指定
  dY　：傾きの縦軸指定
  X　：レベル指定
  N　：個数や何番目の指定
  Xa, La, Fa　：配列の指定
  Pa　：整数配列の指定

4.4.3 アドレス・ポイントを求める関数

1. 測定ポイントを求める関数 POINT1, POINT1L, POINT1H

POINT1（周波数，解析チャンネル） POINT1L（周波数，解析チャンネル） POINT1H（周波数，解析チャンネル）

説明　指定された周波数の測定ポイントを求めます。

POINT1 関数　：指定周波数に最も近い測定ポイントを求めます。すなわち測定ポイントへの変換で四捨五入を行います。

POINT1L 関数　：指定周波数を越えない最も大きい測定ポイントを求めます。すなわち測定ポイントへの変換で切り捨てを行います。

POINT1H 関数　：指定周波数より大きな最も小さい測定ポイントを求めます。すなわち測定ポイントへの変換で切り上げを行います。

用途　多くのビルトイン関数は、アドレス・ポイントを引き数にしています。他のビルトイン関数を使用するために、周波数を測定ポイントに変換します。
解析範囲を指定するときに切り上げ、切り捨てた方が範囲の指定が正確になります。

例　P0=POINT1L(F0, 0)
P1=POINT1H(F1, 0)
X=MAX(P0, P1, 0)　周波数 F0, F1 を含む範囲で最大値を探します。

P=POINT1(F, 0)
Y=VALUE(P, 0)　周波数 F に最も近い測定データを読み出します。

2. アドレス・ポイントを求める関数 POINT2, POINT2L, POINT2H

POINT2（周波数，解析チャンネル） POINT2L（周波数，解析チャンネル） POINT2H（周波数，解析チャンネル）

説明　指定された周波数のアドレス・ポイントを求めます。

POINT2 関数　：指定周波数に最も近いアドレス・ポイントを求めます。すなわちアドレス・ポイントへの変換で四捨五入を行います。

POINT2L 関数　：指定周波数を越えない最も大きいアドレス・ポイントを求めます。すなわちアドレス・ポイントへの変換で切り捨てを行います。

POINT2H 関数　：指定周波数より大きな最も小さいアドレス・ポイントを求めます。すなわちアドレス・ポイントへの変換で切り上げを行います。

用途　多くのビルトイン関数は、アドレス・ポイントを引き数にしています。他のビルトイン関数を使用するために、周波数をアドレス・ポイントに変換します。

例　P=POINT2(F, 0)
Y=VALUE(P, 0)

周波数 F に最も近い測定データを、測定ポイントのときには測定データ、そうでないときは補間して読み出します。

3. アドレス・ポイント幅を求める関数 DPOINT

DPOINT（周波数1,　周波数2,　解析チャンネル）

説明　周波数幅に対応するアドレス・ポイント幅を求めます。

4. 最新の測定ポイントを求める関数 SWPOINT

SWPOINT（解析チャンネル）

説明　測定中に最新の測定ポイントを求めます。

用途　SWPOINT（解析チャンネル）を使用して掃引の状態を知ることができます。
以下の例のように、掃引中にすでに掃引されたデータの解析もできます。

例　*SWEEPING1
IF SWPOINT(0)<P1 THEN GOTO *SWEEPING1
X=MAX(P0, P1, 0)

注意　本器が高速に掃引しているとき、測定ポイントは間欠読み出しになります。

4.4.4 周波数を求める関数

1. 周波数を求める関数 FREQ

FREQ（アドレス・ポイント, 解析チャンネル）

説明　アドレス・ポイントを周波数に変換します。

用途　アドレス・ポイントを返す関数の値を、周波数に変換します。

例　P=PMAX(0, 1200, 0)
F=FREQ(P, 0)
X=VALUE(P, 0)

最大値の周波数とレスポンス値を求めます。MAX, FMAX を使用するより、サーチが一度で済むので、より高速に処理できます。

2. 周波数幅を求める関数 DFREQ

DFREQ（アドレス・ポイント1, アドレス・ポイント2, 解析チャンネル）

説明　指定したアドレス・ポイントから周波数幅へ変換します。

3. 最新の周波数を求める関数 SWFREQ

SWFREQ（周波数, 解析チャンネル）

説明　測定中に最新の測定周波数を求めます。

用途　SWFREQ（解析チャンネル）を使用して、掃引している周波数を知ることができます。

例　*SWEEPING1
IF SWFREQ(0)<F1 THEN GOTO *SWEEPING1
X=CVALUE(F1)

注意　本器が高速に掃引しているとき、測定周波数は間欠読み出しになります。

4.4.5 レスポンスを求める関数

1. レスポンスを求める関数 VALUE

VALUE（アドレス・ポイント, 解析チャンネル）

説明　指定アドレス・ポイントのレスポンスを読み出します。アドレス・ポイントが測定ポイントでないときは、補間して求めます。

用途　アドレス・ポイントを返す関数の値をレスポンス値に変換します。

例　P=PMAX(0, 1200, 0)
F=FREQ(P, 0)
X=VALUE(P, 0)

最大値の周波数とレスポンス値を求めます。MAX, FMAX を使用するより、サーチが一度で済むので、より高速に処理できます。

2. レスポンス差を求める関数 DVALUE

DVALUE（アドレス・ポイント1, アドレス・ポイント2, 解析チャンネル）

説明　指定アドレス・ポイントのそれぞれのレスポンス値の差を求めます。

3. レスポンス値を求める関数 CVALUE

CVALUE（周波数, 解析チャンネル）

説明　指定した周波数に対応するレスポンス値を求めます。

4. レスポンス差を求める関数 DCVALUE

DCVALUE（周波数1, 周波数2, 解析チャンネル）

説明　指定周波数のそれぞれのレスポンス値の差を求めます。

5. 最新のレスポンス値を求める関数 SWVALUE

SWVALUE（解析チャンネル）

説明　測定中に最新の測定レスポンス値を求めます。

用途　レスポンス値をモニタして、調整などに使用できます。

例　　*ADJUST
IF SWVALUE(33)<=PHASE1 THEN GOTO *ADJUST_END
OUTPUT 33;C
GOTO *ADJUST
*ADJUST_END
位相値がある値以下になるまでパラレル I/O へ出力します。

注意　本器が高速に掃引しているとき、測定レスポンス値は間欠読み出しになります。

4.4.6　最大値、最小値を求める関数

1. 最大レスポンス値を求める関数 MAX

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

説明　指定アドレス・ポイント間で、最大レスポンス値をサーチします。

用途　共振点のレスポンス値を求めるときなどに使用します。

例　　X=MAX(0, 1200, 0)

2. 最大レスポンスの周波数を求める関数 FMAX

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

説明　指定アドレス・ポイント間で、最大レスポンスの周波数を求めます。

用途　共振点の周波数を求めるときなどに使用します。

例　　F=FMAX(0, 1200, 0)

3. 最大レスポンスの測定ポイントを求める関数 PMAX

PMAX（開始アドレス・ポイント, 終了アドレス・ポイント, 解析チャンネル）

説明　指定アドレス・ポイント間で、最大レスポンスの測定ポイントを求めます。

用途　共振点の周波数、レスポンス値を求めるときなどに使用します。また、他の解析のアドレス・ポイントなどに使用します。

例 1　P=PMAX(0, 1200, 0)
F=FREQ(P, 0)
X=VALUE(P, 0)
最大値の測定ポイントから周波数、レスポンス値を求めます。FMAX, MAX 使用時に比べ、一度のサーチなので、高速になります。

例 2　P=PMAX(0, 1200, 0)
FB=BND(P, 3, 0)
ピークから 3dB ダウンの帯域幅を求めます。

4. 最小レスポンス値を求める関数 MIN

MIN（開始アドレス・ポイント, 終了アドレス・ポイント, 解析チャンネル）

説明　指定アドレス・ポイント間で、最小レスポンス値をサーチします。

用途　反共振点のレスポンス値を求める場合などに使用します。

例　X=MIN(0, 1200, 0)

5. 最小レスポンスの周波数を求める関数 FMIN

FMIN（開始アドレス・ポイント, 終了アドレス・ポイント, 解析チャンネル）

説明　指定アドレス・ポイント間で、最小レスポンスの周波数を求めます。

用途　反共振点の周波数を求めるときなどに使用します。

例　F=FMIN(0, 1200, 0)

6. 最小レスポンスの測定ポイントを求める関数 PMIN

PMIN（開始アドレス・ポイント, 終了アドレス・ポイント, 解析チャンネル）

説明　指定アドレス・ポイント間の最小レスポンスの測定ポイントを求めます。

用途　反共振点の周波数、レスポンス値を求めるときなどに使用します。

例　P=PMIN(0, 1200, 0)
F=FREQ(P, 0)
X=VALUE(P, 0)

最小値の測定ポイントから、周波数・レスポンス値を求めます。FMIN, MIN の使用時に比べ、一度のサーチなので高速になります。

4.4.7 帯域幅などを求める関数

1. 帯域幅を求める関数 BND

BND（アドレス・ポイント, 減衰レベル, 解析チャンネル）

説明　指定アドレス・ポイントから、指定された減衰レベルだけ減衰したポイントをサーチし、帯域幅を求めます。サーチは指定アドレス・ポイントから外側に行います。

用途　3dB ダウンの帯域幅などを求めます。

例　P=PMAX(0, 1200, 0)
F=BND(P, 3, 0)

3dB ダウンの帯域幅を求めます。

2. 帯域幅の低周波数側の周波数を求める関数 BNDL

BNDL（アドレス・ポイント, 減衰レベル, 解析チャンネル）

説明　指定アドレス・ポイントから、指定された減衰レベルだけ減衰したポイントを低周波数側にサーチし、その周波数を求めます。

用途　BNDH と組み合わせて中心周波数を求めることができます。

3. 帯域幅の高周波数側の周波数を求める関数 BNDH

BNDH（アドレス・ポイント, 減衰レベル, 解析チャンネル）

説明　指定アドレス・ポイントから、指定された減衰レベルだけ減衰したポイントを高周波数側にサーチし、その周波数を求めます。

用途　BNDL と組み合わせて中心周波数を求めることができます。

例　P=PMAX(0, 1200, 0)
FH=BNDH(P, 3, 0)
FL=BNDL(P, 3, 0)
FB=FH-FL
FC=(FL+FH)*0.5

4. 帯域幅を求める関数 CBND

CBND（周波数, 減衰レベル, 解析チャンネル）

説明　指定周波数から、指定された減衰レベルだけ減衰したポイントをサーチし、帯域幅を求めます。サーチは指定アドレス・ポイントから外側に行います。

用途　3dB ダウンの帯域幅などを求めます。

例　F=BND(F, 3, 0)

3dB ダウンの帯域幅を求めます。

5. 帯域幅の低周波数側の周波数を求める関数 CBNDL

CBNDL（周波数, 減衰レベル, 解析チャンネル）

説明　指定周波数から、指定された減衰レベルだけ減衰したポイントを低周波数側にサーチし、その周波数を求めます。

用途　CBNDH と組み合わせて中心周波数を求めることができます。

6. 帯域幅の高周波数側の周波数を求める関数 CBNDH

CBNDH（周波数, 減衰レベル, 解析チャンネル）

説明　指定周波数から、指定された減衰レベルだけ減衰したポイントを高周波数側にサーチし、その周波数を求めます。

用途　CBNDL と組み合わせて中心周波数を求めることができます。

例　FH=CBNDH(F, 3, 0)
FL=CBNDL(F, 3, 0)
FB=FH-FL
FC=(FL+FH)*0.5

7. 複数の減衰レベルの帯域幅解析を行う関数 MBNDI

MBNDI（開始アドレス・ポイント, 終了アドレス・ポイント, 基準アドレス・ポイント, 減衰レベルの数, 減衰レベルの配列, 帯域幅などの解析結果を格納する配列, 解析チャンネル）

説明　複数の減衰レベルの解析を一度に行います。1 つの減衰レベルに対し、低周波数側の周波数、高周波数側の周波数、中心周波数、帯域幅の 4 つを出力します。
減衰レベルは配列で指定し、解析結果も配列に格納されます。サーチは、指定アドレス・ポイントから外側に向かって行います。減衰レベルの配列は、レベルの小さい順になっている必要があります。

用途　複数の減衰レベルで解析を行うときに、高速処理できます。減衰レベルが 1 つのときでも、4 つの周波数が必要なときに便利です。

例
```
DIML(3), F(3, 4)
L(1)=1.0
L(2)=3.0
L(3)=10.0
P=PMAX(0, 1200, 0)
N=MBNDI(0, 1200, P, 3, L(1), F(1, 1), 0)
```

このとき配列 F には、以下のものが格納されます。

F(1, 1) 減衰レベル 1.0 のときの低周波数側周波数
F(1, 2) 減衰レベル 1.0 のときの高周波数側周波数
F(1, 3) 減衰レベル 1.0 のときの中心周波数
F(1, 4) 減衰レベル 1.0 のときの帯域幅
F(2, 1) 減衰レベル 3.0 のときの低周波数側周波数
F(2, 2) 減衰レベル 3.0 のときの高周波数側周波数
F(2, 3) 減衰レベル 3.0 のときの中心周波数
F(2, 4) 減衰レベル 3.0 のときの帯域幅
F(3, 1) 減衰レベル 10.0 のときの低周波数側周波数
F(3, 2) 減衰レベル 10.0 のときの高周波数側周波数
F(3, 3) 減衰レベル 10.0 のときの中心周波数
F(3, 4) 減衰レベル 10.0 のときの帯域幅

なお、サーチできなかった場合には 0.0 が入ります。N には、サーチできた減衰レベルの数が入ります。

8. 複数の減衰レベルの帯域幅解析を行う関数 MBNDO

MBNDO（開始アドレス・ポイント, 終了アドレス・ポイント, 基準アドレス・ポイント, 減衰レベルの数, 減衰レベルの配列, 帯域幅などの解析結果を格納する配列, 解析チャンネル ）

説明　機能は MBNDI と同様ですが、サーチは外側から内側へ行います。

用途　サーチを外側から内側に行うときに使用します。

例　DIML(3), F(3, 4)
L(1)=1.0
L(2)=3.0
L(3)=10.0
P=PMAX(0, 1200, 0)
N=MBNDO(0, 1200, P, 3, L(1), F(1, 1), 0)

このとき配列 F に格納されるのは、MBNDI のときと同じです。

4.4.8 リップル解析関数 -1

1. 最大の極大値と最小の極大値の差を求める関数 RPL1

RPL1　（開始アドレス・ポイント, 終了アドレス・ポイント, 横軸の傾き係数, 縦軸の傾き係数, 解析チャンネル ）

説明　横軸と縦軸の傾き係数に従って極大、極小を指定アドレス・ポイント間で検出し、最大の極大値と最小の極小値の差を求めます。

用途　測定対象のリップルを解析します。

例　X=RPL1(0, 1200, 1, 0.5, 0)
1 ポイントあたり 0.5dB 上がる／下がるものをリップルとし、その最大と最小の差を求めます。

2. 隣接する極大値と極小値の差の最大値を求める関数 RPL2

RPL2　（開始アドレス・ポイント, 終了アドレス・ポイント, 横軸の傾き係数, 縦軸の傾き係数, 解析チャンネル ）

説明　横軸と縦軸の傾き係数に従って極大、極小を指定アドレス・ポイント間で検出します。隣接する極大値と極小値の差を求め、さらにその最大値を求めます。隣接する極大、極小では、極大の方が低周波数側になります。

用途　測定対象のリップルを解析します。

例　P=PMAX(0, 1200, 0)
X=RPL2(0, P, 1, 0.5, 0)

1 ポイントあたり 0.5dB 上がる／下がるものをリップルとし、ピーク点の左側で隣接する極大、極小の差の最大値を求めます。

3. 隣接する極大値と極小値の差を加算した値の最大値を求める関数 RPL3

RPL3　（開始アドレス・ポイント, 終了アドレス・ポイント, 横軸の傾き係数, 縦軸の傾き係数, 解析チャンネル）

説明　横軸と縦軸の傾き係数に従って極大、極小を指定アドレス・ポイント間で検出します。隣接する極大値と極小値の差、極小値と極大値の差、を加算し、その最大値を求めます。

用途　測定対象のリップルを解析します。

例　X=RPL3(0, 1200, 1, 0.5, 0)

1 ポイントあたり 0.5dB 上がる／下がるものをリップルとし、解析します。

4. 隣接する極大値と極小値の差の最大値を求める関数 RPL4

RPL4　（開始アドレス・ポイント, 終了アドレス・ポイント, 横軸の傾き係数, 縦軸の傾き係数, 解析チャンネル）

説明　横軸と縦軸の傾き係数に従って極大、極小を指定アドレス・ポイント間で検出し、隣接する極大値と極小値の差を求め、その最大値を求めます。隣接する極大・極小では、極大の方が高周波数側になります。RPL2 とは、極大、極小のペアの取り方が逆になります。

用途　測定対象のリップルを解析します。

例　P=PMAX(0, 1200, 0)
X=RPL4(P, 1200, 1, 0.5, 0)

1 ポイントあたり 0.5dB 上がる／下がるものをリップルとし、ピーク点の右側で隣接する極大、極小の差の最大値を求めます。

5. 極大値の最大値を求める関数 RPL5

RPL5 （開始アドレス・ポイント, 終了アドレス・ポイント, 横軸の傾き係数, 縦軸の傾き係数, 解析チャンネル ）

説明　横軸と縦軸の傾き係数に従って極大を指定アドレス・ポイント間で検出し、最大の極大値を求めます。

用途　測定対象のリップル・スプリアスを解析します。

例　X=RPL5(P0, P1, 1, 0.5, 0)

1 ポイントあたり 0.5dB 上がる／下がるものをリップルとし、最大の極大値を求めます。

6. 極大値の最小値を求める関数 RPL6

RPL6 （開始アドレス・ポイント, 終了アドレス・ポイント, 横軸の傾き係数, 縦軸の傾き係数, 解析チャンネル ）

説明　横軸と縦軸の傾き係数にしたがって極大を指定アドレス・ポイント間で検出し、最小の極大値を求めます。

用途　測定対象のリップル・スプリアスを解析します。

例　X=RPL6(P0, P1, 1, 0.5, 0)

1 ポイントあたり 0.5dB 上がる／下がるものをリップルとし、最小の極大値を求めます。

7. 極大、極小の周波数差を求める関数 RPLF

RPLF （開始アドレス・ポイント, 終了アドレス・ポイント, 横軸の傾き係数, 縦軸の傾き係数, 解析チャンネル ）

説明　横軸と縦軸の傾き係数に従って極大を指定アドレス・ポイント間で検出し、最初に見つかった極大値と次の極小値の周波数差を求めます。

用途　測定対象のリップルを解析します。

例　X=RPLF(P0, P1, 1, 0.5, 0)

1 ポイントあたり 0.5dB 上がる／下がるものをリップルとし、極大、極小の周波数差を求めます。

8. 極大、極小のレスポンス差を求める関数 RPLR

RPLR （開始アドレス・ポイント, 終了アドレス・ポイント, 横軸の傾き係数, 縦軸の傾き係数, 解析チャンネル ）

説明　横軸と縦軸の傾き係数に従って極大を指定アドレス・ポイント間で検出し、最初に見つかった極大値と次の極小値のレスポンス差を求めます。

用途　測定対象のリップルを解析します。

例　X=RPLR(P0, P1, 1, 0.5, 0)

1 ポイントあたり 0.5dB 上がる／下がるものをリップルとし、極大、極小のレスポンス差を求めます。

9. 極大値のレスポンス値を求める関数 RPLH

RPLH （開始アドレス・ポイント, 終了アドレス・ポイント, 横軸の傾き係数, 縦軸の傾き係数, 解析チャンネル ）

説明　横軸と縦軸の傾き係数に従って極大を指定アドレス・ポイント間で検出し、最初に見つかった極大値のレスポンス値を求めます。

用途　測定対象のリップルを解析します。

例　X=RPLH(P0, P1, 1, 0.5, 0)

1 ポイントあたり 0.5dB 上がる／下がるものをリップルとし、極大のレスポンス値を求めます。

10. 極大値の周波数を求める関数 FRPLH

FRPLH （開始アドレス・ポイント, 終了アドレス・ポイント, 横軸の傾き係数, 縦軸の傾き係数, 解析チャンネル ）

説明　横軸と縦軸の傾き係数に従って極大を指定アドレス・ポイント間で検出し、最初に見つかった極大値の周波数を求めます。

用途　測定対象のリップルを解析します。

例　X=FRPLH(P0, P1, 1, 0.5, 0)

1 ポイントあたり 0.5dB 上がる／下がるものをリップルとし、極大の周波数を求めます。

11. 極大値の測定ポイントを求める関数 PRPLH

PRPLH （開始アドレス・ポイント, 終了アドレス・ポイント, 横軸の傾き係数, 縦軸の傾き係数, 解析チャンネル）

説明　横軸と縦軸の傾き係数に従って極大を指定アドレス・ポイント間で検出し、最初に見つかった極大値の測定ポイントを求めます。

用途　測定対象のリップルを解析します。

例　X=PRPLH(P0, P1, 1, 0.5, 0)

1 ポイントあたり 0.5dB 上がる／下がるものをリップルとし、極大の測定ポイントを求めます。

12. 極小値のレスポンス値を求める関数 RPLL

RPLL （開始アドレス・ポイント, 終了アドレス・ポイント, 横軸の傾き係数, 縦軸の傾き係数, 解析チャンネル）

説明　横軸と縦軸の傾き係数に従って極小を指定アドレス・ポイント間で検出し、最初に見つかった極小値のレスポンス値を求めます。

用途　測定対象のリップルを解析します。

例　X=RPLL(P0, P1, 1, 0.5, 0)

1 ポイントあたり 0.5dB 上がる／下がるものをリップルとし、極小のレスポンス値を求めます。

13. 極小値の周波数を求める関数 FRPLL

FRPLL （開始アドレス・ポイント, 終了アドレス・ポイント, 横軸の傾き係数, 縦軸の傾き係数, 解析チャンネル）

説明　横軸と縦軸の傾き係数に従って極小を指定アドレス・ポイント間で検出し、最初に見つかった極小値の周波数を求めます。

用途　測定対象のリップルを解析します。

例　X=FRPLL(P0, P1, 1, 0.5, 0)

1 ポイントあたり 0.5dB 上がる／下がるものをリップルとし、極小の周波数を求めます。

14. 極小値の測定ポイントを求める関数 PRPLL

PRPLL （開始アドレス・ポイント, 終了アドレス・ポイント, 横軸の傾き係数, 縦軸の傾き係数, 解析チャンネル ）

説明　横軸と縦軸の傾き係数に従って極小を指定アドレス・ポイント間で検出し、最初に見つかった極小値の測定ポイントを求めます。

用途　測定対象のリップルを解析します。

例　X=PRPLH(P0, P1, 1, 0.5, 0)

1 ポイントあたり 0.5dB 上がる／下がるものをリップルとし、極小の測定ポイントを求めます。

4.4.9 リップル解析関数 -2

1. 極大値の数を求める関数 NRPLH

NRPLH （開始アドレス・ポイント, 終了アドレス・ポイント, 横軸の傾き係数, 縦軸の傾き係数, 解析チャンネル ）

説明　横軸と縦軸の傾き係数に従って極大を指定アドレス・ポイント間で検出し、極大値の情報を内部に保存して極大値の数を求めます。

用途　測定対象のリップルを解析します。

例　NH=NRPLH(0, 1200, 1, 0.5, 0)

1 ポイントあたり 0.5dB 上がる／下がるものをリップルとし、極大値の個数を求めます。

2. 極小値の数を求める関数 NRPLL

NRPLL （開始アドレス・ポイント, 終了アドレス・ポイント, 横軸の傾き係数, 縦軸の傾き係数, 解析チャンネル ）

説明　横軸と縦軸の傾き係数に従って極小を指定アドレス・ポイント間で検出し、極小値の情報を内部に保存して極小値の数を求めます。

用途　測定対象のリップルを解析します。

例　　NL=NRPLL(0, 1200, 1, 0.5, 0)

1 ポイントあたり 0.5dB 上がる／下がるものをリップルとし、極小値の個数を求ます。

3. 極大値、極小値の測定ポイントを求める関数 PRPLHN, PRPLLN

PRPLHN（リップルの番号指定, 解析チャンネル） PRPLLN（リップルの番号指定, 解析チャンネル）

説明　　PRPLHN:NRPLH で求めた N 番目の極大値の測定ポイントを求めます。
PRPLLN:NRPLL で求めた N 番目の極小値の測定ポイントを求めます。

例　　NH=NRPLH(0, 1200, 1, 0.5, 0)
NL=NRPLL(0, 1200, 1, 0.5, 0)
PH2=PRPLHN(2, 0)
PL2=PRPLLN(2, 0)

NRPLH, NRPLL を実行し、2 番目の極大値・極小値の測定ポイントを求めます。

4. 極大値、極小値の周波数を求める関数 FRPLHN, FRPLLN

FRPLHN（リップルの番号指定, 解析チャンネル） FRPLLN（リップルの番号指定, 解析チャンネル）

説明　　FRPLHN:NRPLH で求めた N 番目の極大値の周波数を求めます。
FRPLLN:NRPLL で求めた N 番目の極小値の周波数を求めます。

用途　　測定対象のリップルを解析します。

例　　NH=NRPLH(0, 1200, 1, 0.5, 0)
NL=NRPLL(0, 1200, 1, 0.5, 0)
FH2=FRPLHN(2, 0)
FL2=FRPLLN(2, 0)

NRPLH, NRPLL を実行し、2 番目の極大値、極小値の周波数を求めます。

5. 極大値、極小値のレスポンス値を求める関数 VRPLHN, VRPLLN

VRPLHN（リップルの番号指定, 解析チャンネル） VRPLLN（リップルの番号指定, 解析チャンネル）

説明　VRPLHN:NRPLH で求めた N 番目の極大値のレスポンス値を求めます。
VRPLLN:NRPLL で求めた N 番目の極小値のレスポンス値を求めます。

用途　測定対象のリップルを解析します。

例　NH=NRPLH(0, 1200, 1, 0.5, 0)
NL=NRPLL(0, 1200, 1, 0.5, 0)
XH2=VRPLHN(2, 0)
XL2=VRPLLN(2, 0)

NRPLH, NRPLL を実行し、2 番目の極大値、極小値のレスポンス値を求めます。

6. 極大値、極小値の測定ポイントを一括で求める関数 PRPLHM, PRPLLM

PRPLHM（整数配列, 解析チャンネル） PRPLLM（整数配列, 解析チャンネル）

説明　PRPLHM:NRPLH で求めた極大値の測定ポイントを求めます。
PRPLLM:NRPLL で求めた極小値の測定ポイントを求めます。

用途　測定対象のリップルを解析します。

例　INTEGERPH(600), PL(600)
NH=NRPLH(0, 1200, 1, 0.5, 0)
NL=NRPLL(0, 1200, 1, 0.5, 0)
NH=PRPLHM(PH(1), 0)
NL=PRPLLM(PL(1), 0)

NRPLH, NRPLL を実行し、極大値、極小値の測定ポイントを配列に取り込みます。

7. 極大値、極小値の周波数を一括で求める関数 FRPLHM, FRPLLM

FRPLHM（実数配列, 解析チャンネル） FRPLLM（実数配列, 解析チャンネル）

説明　FRPLHM:NRPLH で求めた極大値の周波数を求めます。
FRPLLM:NRPLL で求めた極小値の周波数を求めます。

用途　測定対象のリップルを解析します。

例　DIMFH(600), FL(600)
NH=NRPLH(0, 1200, 1, 0.5, 0)
NL=NRPLL(0, 1200, 1, 0.5, 0)
NH=FRPLHM(FH(1), 0)
NL=FRPLLM(FH(1), 0)

NRPLH, NRPLL を実行し、極大値、極小値の周波数を配列に取り込みます。

8. 極大値、極小値のレスポンス値を一括で求める関数 VRPLHM, VRPLLM

VRPLHM（実数配列, 解析チャンネル） VRPLLM（実数配列, 解析チャンネル）

説明　VRPLHM:NRPLH で求めた極大値のレスポンス値を求めます。
VRPLLM:NRPLL で求めた極小値のレスポンス値を求めます。

用途　測定対象のリップルを解析します。

例　DIMXH(600), XL(600)
NH=NRPLH(0, 1200, 1, 0.5, 0)
NL=NRPLL(0, 1200, 1, 0.5, 0)
NH=VRPLHM(XH(1), 0)
NL=VRPLLM(XL(1), 0)

NRPLH, NRPLL を実行し、極大値、極小値のレスポンス値を配列に取り込みます。

4.4.10 ダイレクト・サーチ

1. 指定レスポンスに対応するアドレス・ポイントを求める関数 DIRECT

DIRECT（開始アドレス・ポイント, 終了アドレス・ポイント, レスポンス値, 解析チャンネル）

説明　指定アドレス・ポイント間で、指定レスポンス値をサーチし、対応するアドレス・ポイントを求めます。サーチの方向は低周波から高周波です。

例　P=DIRECT(0, 1200, -10.0, 0)

-10dB のデータの位置を探します。

2. 指定レスポンスに対応する測定ポイントを求める関数 DIRECTL, DIRECTH

DIRECTL（開始アドレス・ポイント, 終了アドレス・ポイント, レスポンス値, 解析チャンネル） DIRECTH（開始アドレス・ポイント, 終了アドレス・ポイント, レスポンス値, 解析チャンネル）

説明　指定アドレス・ポイント間で指定レスポンス値をサーチし、対応する測定ポイントを求めます。サーチの方向は低周波から高周波が DIRECTL、高周波から低周波が DIRECTH です。指定レスポンスに一致したレスポンスがあったときは、その測定ポイントを返します。一致しないときは指定レスポンス値を越えた測定ポイントを返します。そのため、連続サーチが行いやすくなっています。

例　P0=DIRECTL(0, 1200, -3.0, 0)
P1=DIRECTH(0, 1200, -3.0, 0)
F=DFREQ(P0, P1, 0)

外側からサーチして帯域幅を求めます。

3. 指定レスポンスに対応する周波数を求める関数 CDIRECT

CDIRECT（開始周波数, 終了周波数,レスポンス値, 解析チャンネル）

説明　指定周波数間で、指定レスポンス値をサーチし、対応する周波数を求めます。サーチの方向は低周波から高周波です。

例　F=CDIRECT(F0, F1, -10.0, 0)

-10dB のデータの周波数を求めます。

4. 指定レスポンスに対応する周波数を求める関数 CDIRECTL, CDIRECTH

CDIRECTL（開始周波数, 終了周波数,レスポンス値, 解析チャンネル） CDIRECTH（開始周波数, 終了周波数,レスポンス値, 解析チャンネル）

説明　指定周波数間で、指定レスポンス値をサーチし、対応する周波数を求めます。サーチの方向は低周波から高周波が CDIRECTL、高周波から低周波が CDIRECTH です。

例　F0=CDIRECTL(F0, F1, -3.0, 0)
F1=CDIRECTH(F0, F1, -3.0, 0)
F=F1-F0

外側からサーチして帯域幅を求めます。

5. 指定レスポンスのアドレス・ポイント幅を求める関数 DDIRECT

DDIRECT（開始アドレス・ポイント, 終了アドレス・ポイント, レスポンス値, 解析チャンネル）

説明　指定アドレス・ポイント間で、指定レスポンス値を高周波数側へサーチし、検出した 2 つの測定ポイントからアドレス・ポイント幅を求めます。

6. 指定レスポンスの帯域幅を求める関数 CDDIRECT

CDDIRECT（開始周波数, 終了周波数,レスポンス値, 解析チャンネル）

説明　指定周波数間で、指定レスポンス値を高周波数側へサーチし、検出した 2 つの測定ポイントから帯域幅を求めます。

7. ゼロ位相の周波数を求める関数 ZEROPHS

ZEROPHS（開始アドレス・ポイント, 終了アドレス・ポイント, レスポンス値, 解析チャンネル）

説明　指定アドレス・ポイント間で位相ゼロを検出し、その周波数を求めます。

4.4.11 データ転送

1. 指定解析チャンネルのデータを配列に読み込む関数 TRANSR

TRANSR（開始アドレス・ポイント, 終了アドレス・ポイント, 実数配列, 解析チャンネル）

説明　指定した解析チャンネルの測定データを、アドレス・ポイントを指定して BASIC の配列に読み込み、データ数を返します。

用途　測定データを 2 次処理するときに使用します。

例　DIMX(1201)
N=TRANSR(0, 1200, X(1), 0)

2. 指定解析チャンネルに配列の内容を書き込む関数 TRANSW

TRANSW（開始アドレス・ポイント, 終了アドレス・ポイント, 実数配列, 解析チャンネル）

説明　指定した解析チャンネルに、BASIC の配列の内容を書き込みます。

用途　測定データを 2 次処理するときに使用します。

例　DIM X(1201)
N=TRANSW(0, 1200, X(1), 0)