

11

第 11 章

BS-BASICの総点検

BS-BASICのすべてをまとめてあるので、プログラムを作っていて、アレッと疑問を持った時とか、忘れてしまったときなどに利用してほしい。

一度全部の命令を確認して、間違って覚えていたり、見落していたりしたところがないか確かめておくことも大切なことだろう。この本は、読めば読むほど、新しい発見がある。

BS-BASIC命令のまとめ

コマンド(直接実行命令)

NEW	NEW	現在テキストエリア内にあるBASICプログラムを消去し、変数エリアをクリアします。LIMITコマンドによって設定した機械語プログラムエリアはクリアされません。
AUTO	AUTO	プログラム作成時に、行番号を10, 20, 30…と自動的に発生します。
	AUTO, 20	行番号を10, 30, 50…と20おきに自動的に発生します。
	AUTO 200, 20	行番号を、200から20おきに、200, 220, 240…と自動的に発生します。
RUN	RUN	AUTOコマンドは SHIFT キー + STOP キーによって解除されます。
	RUN 100	現在テキストエリアにあるBASICプログラムの先頭から実行します。 (プログラムの実行直前にすべての変数の内容を0または空(NULL)にします。)
LIST	LIST	行番号100からプログラムを実行します。(変数の内容のクリアは行いません。)
	LIST10-80	現在テキストエリア内にあるBASICプログラムの全リストを表示します。
	LIST500	行番号10-80までのリストを表示します。
CONT	CONT	行番号500までのリストを表示します。
		リスト表示中、スペースバーを押すと、その間リスト表示がいったん停止します。
SAVE	SAVE	プログラム実行を継続します。即ちプログラム中のEND文、STOP文あるいは SHIFT キー + STOP キーによって中断された箇所から実行を再開します。プログラム中断時に、BASICプログラムの修正を行なうとCONTコマンドは無効になります。
	SAVE " E "	現在BASICテキストエリアにあるプログラムをカセットテープに書き込みます。
LOAD	LOAD	現在BASICテキストエリアにあるプログラムを" E "というファイル名を付けてカセットテープに書き込みます。(ファイル名は16字以内のストリングです。)
	LOAD " C "	カセットテープ中の最初に見つかったBASICプログラムを読み出します。(それ以前にBASICテキストエリアにあったプログラムは無効になります。) " C "というファイル名のBASICプログラムをカセットテープから読み出します。

GET	機械語ファイル名 LOAD " D "	BASICプログラムにリンクされる" D "という名の機械語プログラムファイルをロードします。
PUT	PUT 文	OBJモードのファイルのローディングはLIMIT文によって確保した機械語プログラムエリアが既にあり、しかも該当OBJモードファイルのローディングアドレスがそのエリア内であれば、ローディングが実行されます。(このとき、BASICテキストエリアにあったプログラムは消去されません。) 安全入出力
VERIFY	VERIFY ファイル名 VERIFY " H "	現在BASICテキストエリア内にあるプログラムと、カセットテープ中の最初に見つかったBASICプログラムの内容を比較します。両者の内容が同一であれば" OK "が、異っている場合は" ERROR "が表示されます。
APPEND	APPEND ファイル名 APPEND " I "	現在BASICテキストエリア内にあるプログラムと、" H "という名のカセットテープにセーブしてあるプログラムの内容を比較します。
LIST/P	LIST/P	現在BASICテキストエリア内にあるプログラムと、カセットテープ中の最初に見つかったBASICプログラムを混ぜ合わせます。同じ行番号があるときは、カセットテープの方が優先されます。
LIST/P	LIST/P 10-80	現在BASICテキストエリア内にあるプログラムと、カセットテープにある" I "というファイル名のBASICプログラムを混ぜ合わせます。
MON	MON	リストをプリンタに出力します。
		行番号10から80までのリストをプリンタに出力します。
		システムのコントロールをBASICからモニタへ移します。(モニタからBASICへの復帰は、モニタコマンド" J "によって行うことができます。)

ステートメント

①入出力文

PRINT	10 PRINT A 20 PRINT, "A" ? A \$ 100 PRINT A; A \$, B; B \$ 110 PRINT "COST="; CS 120 PRINT	数値変数Aの内容を画面に表示します。 6文字右の位置に"A"を表示します。 文字変数A \$の内容を画面に表示します。 数値変数と文字変数を混合して使用できます。また区切りでセミコロンが使われると、スペースなしで続けて表示され、コンマが使われると次の表示位置（6文字ごとの区切り）から表示されます。 "(ダブルフォーテーション)で囲まれたストリングはその内容がそのまま表示されます。
INPUT	10 INPUT A 20 INPUT A \$ 30 INPUT "VALUE ?"; D 40 INPUT X, X \$, Y, Y \$	PRINTだけの場合は、行替えになります。 キーボードから数値変数Aに対する数値を入力します。 キーボードから文字変数A \$に対する文字を入力します。文字は""を用いずに入力できますが、文字の前後にスペースやコンマを与える場合は、全体を""で囲みます。 キーボードから入力する前に、ストリングのVALUE ?を表示させます。ストリングと変数の区切りは; (セミコロン)を使います。
READ～DATA	10 READ A, B, C 1000 DATA 25, -0.5, 500 10 READ H \$, H, S \$, S 30 DATA HEART, 3 35 DATA SPADE, 11	数値変数や文字変数は、(コンマ)で区切れば混合して使用できますが、入力する際には変数の型に合わせる必要があります。 DATA文に置かれた定数またはストリングをREAD文の中に示された変数に代入する命令です。READ文中の変数と、それに対応するDATA文中の各データは、数値変数なら数値データ、文字変数ならストリングデータと変数とデータの形が一致しなくてはなりません。 READ文でデータ並びの途中のデータまで読んだあと、次のREAD文を実行すると、その次のデータからひき続いて読み出されます。 DATA文の後のマルチステートメントはできません。
RESTORE	10 READ A, B, C 20 RESTORE 30 READ D, E 100 DATA 3, 6, 9, 12, 15 700 RESTORE 200	左のREAD～DATA文の実行によって数値変数A、B、Cのそれぞれに数値データ25、-0.5、500が代入されます。 READ文の最初の変数、すなわち文字変数H \$にDATA文の最初のデータ、すなわちストリングデータ"HEART"が代入されます。次に2番目の変数Hには、数値データ3が代入され、次々に読み込まれて行きます。 READ～DATA命令では、READ命令につれてDATA文中から読み込むデータが移って行きますが、RESTORE文を使うことによって読み込むデータをDATA文の最初に戻すことができます。 左の例では、行番号10のREAD文によって、変数A、B、Cにそれぞれ、3、6、9の値が代入されますが、次にRESTORE文が置かれているので、行番号30のREADによって変数D、Eに代入される値は、12、15とはならずに、それぞれ3、6が代入されることになります。 READ～DATA文におけるデータ読み出し位置を、行番号200のDATA文の先頭へ移します。

GET	10 GET N 20 GET K\$	キーボードから数値変数Nに対して、1文字の数値を入力します。 キーボードから文字変数K\$に対して、1個のストリングを入力します。 そのときキーが押されていないと、K\$は空になります。
------------	------------------------	---

②ループ文

FOR~TO	10 FOR A=1 TO 10 20 PRINT A 30 NEXT A	行番号10は変数Aを1から10まで変化させよという命令で、最初Aの値は1となります。行番号20でAの値が画面に表示されるので数値1が表示されます。次に行番号30でAの値は2になってこのループを繰り返します。こうしてAの値が10になるまでこのループが繰り返されます。(ループを終了した時点でAには11の値が入っています。)
	10 FOR B=2 TO 8 STEP 3 20 PRINT B*2 30 NEXT	行番号10は変数Bを2から8まで、3ずつ大きくして変化させよという命令です。STEPの値を負にして変数の値を小さくして行くこともできます。
	10 FOR A=1 TO 3 20 FOR B=10 TO 30 30 PRINT A, B 40 NEXT B 50 NEXT A	変数AとBについてFOR……NEXTループを重ねた例です。BループはAループの内部に置かれているところに注目して下さい。ループは2重、3重……とネスティングすることができますが、内側のループは必ず外側のループ内に閉じていなくてはなりません。 FOR……NEXTのネスティングは最大10まで重ねることができます。
	60 NEXT B, A 70 NEXT A, B	前の2重ループで、行番号40と50を、左の行番号60のように1つのNEXT文にまとめるすることができます。しかし行番号70に示したような書き方ではエラーになります。

③分岐文

GOTO	100 GOTO 200	行番号200へジャンプ(プログラムの実行を移すこと)します。
IF~THEN	10 IF A>20 THEN 200	変数Aの値が20より大なら、行番号200へジャンプします。Aが20以下なら次の行を実行します。
	50 IF B<3 THEN B=B+3	変数Bの値が3より小なら変数BにB+3の値を代入します。Bが3以上なら次の行を実行します。 THEN以下には、続けて別のステートメントを書くことが、可能で、IF THEN IF~とやることもできます。
IF~GOTO	100 IF A>=B GOTO 10	変数Aの値が変数Bの値以上なら行番号10へジャンプします。AがBより小なら次の行を実行します。
ON~GOTO	50 ON A GOTO 70, 80, 90	変数Aの値が1なら行番号70へ、2なら行番号80へ、3なら行番号90へジャンプします。Aが0または4以上なら次の行を実行します.ONにはINTの機能が含まれており、Aが2.7ならば2の場合と同様に行番号80へジャンプします。(変数の値とGOTOの後の行番号の順番が対応します。)
GOSUB~RETURN	100 GOSUB 700	行番号700サブルーチンへ分岐(サブルーチンをコールすること)します。RETURN文でサブルーチンの実行を終了し、メインプログラムでGOSUB命令をした次のステートメントへ戻ります。
IF~GOSUB GOTO	30 IF A=B*2 GOSUB 90	変数Aの値が変数Bの値の2倍に等しいなら行番号90のサブルーチンへ分岐します。等しくないなら次の行を実行します。

ON～GOSUB	90 ON A GOSUB 700, 800	(条件文のあとにマルチステートメントが来る場合、ON文は条件が成り立たないとき実行されますが、IF文は条件が成り立たないとき次の行番号へ移りマルチステートメントは無視されます。) 変数Aの値が1なら行番号700、2なら行番号800のサブルーチンへ分岐します。Aが0または3以上なら次の文を実行します。 GOSUBによるネスティングは最大15まで重ねることができます。基本的な動作はON～GOTOと同じです。
-----------------	------------------------	---

④注釈文

REM 200 REM PROGRAM1 | REMは注釈文であり、プログラム実行の際無視されます。

⑤コントロール文

END	999 END	プログラムの終りを表わします。プログラムの実行をやめますがCONT命令を与えると、さらに先のプログラムを実行します。
CT	? CT	カウンタの表示を行ないます。 1カウントは133.721mSです。
CURSOR	10 CURSOR 15, 10 20 PRINT " A "	画面の左端から15桁目、上端から10行目にカーソルを移動させ、11桁目に"A"を表示します。 X軸方向は0～29、Y軸方向は0～22の数値または変数で位置を指定します。
CSRX		現在のカーソル位置のX座標(水平位置)を示すシステム数値変数です。
CSRY		現在のカーソル位置のY座標(垂直位置)を示すシステム数値変数です。
STOP	850 STOP	プログラムの実行をやめて、命令待ちとなります。ここでCONT命令を与えると、プログラムを続行します。
CLEAR	300 CLEAR	数値をとる変数および配列要素の内容をすべて0、ストリングをとる変数および配列要素の内容をすべて空(null)とします。 FOR～TO NEXTの中、およびサブルーチンの中にCLEAR文は書けません。
SIZE	? SIZE	BASICテキストエリアの未使用バイトサイズを表示させます。

⑥定義文

DEF FN	100 DEF FNA(X)=X↑2-X 110 DEF FNB(X)=LOG(X) +1 120 DEF FNZ(Y)=LN(Y)	DEF FNで関数の定義をします。行番号100は $X^2 - X$ をFNA(X)に、行番号110は $\log_{10}X + 1$ をFNB(X)に、行番号120は $\log_e Y$ をFNZ(Y)に定義します。関数は1変数に限ります。 関数定義のネスティングは最大15まで重ねることができます。 関数名はFNに続けてアルファベット1文字で指定し、()内に変数名を書きます。同時に利用できる関数は最大26で、また、1つのDEF文では1つの関数しか定義できません。
DIM	10 DIM A(20)	配列を使う場合には、このDIM (dimensionの略) 文で、配列要素の最大を宣言しておかなければなりません。配列要素は0から、最大255まで使えます。 1次元数値配列A()について、配列要素をA(0)からA(20)まで21個用意します。

COLOR	20 DIM B(6, 12)	2次元数値配列B()について、配列要素をB(0, 0)からB(6, 12)まで91個用意します。
	30 DIM C1\$(10)	1次元ストリング配列C1\$()について、配列要素をC1\$(0)からC1\$(10)まで11個用意します。
	40 DIM K\$(7, 5)	2次元ストリング配列K\$()について、配列要素を、K\$(0, 0)からK\$(7, 5)まで48個用意します。

⑦プリントコントロール文

PRINT/P	10 PRINT/P A, A\$	PRINTと同様の機能を、プリンタに対して実行します。プリンタが接続されていない時はエラーになります。
	20 PRINT/P, "A"	数値変数Aの内容、続けて文字変数A\$の内容をプリンタにプリントします。

⑧タブレーションコントロール関数

TAB	10 PRINT TAB(X); A	画面の左端から数えてX+1字目に変数Aの値を表示します。 Xは0~255まで指定可能ですが、水平文字数（画面では30）以上の場合は、Xは水平文字数で割った余りとなります。
------------	--------------------	--

⑨プリントコントロール文

PRINT USING	10 A=88 20 ? USING "#.#.#.#" A	#記号の個数によって表示する数字の桁数を決められます。 #記号は最大8個まで使えます。 表示したい値が-(マイナス)の時は、#の前か後ろに-をつけます。 表示できない場合は、#がそのまま表示されます。
	50 ?/P USING "#.#.#.#" A	プリンタにも、画面同様プリントできます。

⑩ジョイスティックコントロール文

JOYS1	JOYS1 C	ジョイスティックから数値変数Cに対して、数値を連続して入力します。
JOYS2	JOYS2 C	ジョイスティックキーを押していないと、Cに10が入力されます。数値はジョイスティックキーの押す位置によって決まり、キー入力の優先順位は9、0、その他の順となります。

⑪エラー処理文

IF ERN	1F ERN=44 THEN 1050	エラー番号が44であれば行番号1050へジャンプせよという命令です。
IF ERL	IF ERL=350 THEN 1090	エラー発生行番号が350であれば行番号1090へジャンプせよという命令です。
	IF (ERN=53)*(ERL=700) THEN END	エラー番号が53で、かつエラー発生行番号が700であるならば、プログラムを終了せよという命令です。
ON ERROR GOTO	ON ERROR GOTO 1000	プログラム実行中にエラーが発生したら、行番号1000にジャンプせよという宣言文です。

RESUME		エラー処理後、emainプログラムへ復帰する命令です。復帰の仕方によって次のようなそれぞれの使い方ができます。
	650 RESUME	エラーが発生した命令へ再びコントロールを移します。
	700 RESUME NEXT	エラーが発生した命令の次の命令へコントロールを移します。
	750 RESUME 400	行番号400へコントロールを移します。
	800 RESUME 0	プログラムの先頭へコントロールを移します。

⑫I/O入出力文

INP@		I/Oポート番号を指定して、そのポート上にあるデータの読み出しを行います。
	10 INP @ 12, A 20 PRINT A	行番号10で、I/Oポート番号12(10進)にあるデータを、変数Aに読み出します。
OUT@		外部デバイスにデータを送るため、I/Oポート番号を指定してデータを出力します。
	30 B=A↑2+0.3 40 OUT @ 13, B	行番号40で、変数Bの値をI/Oポート番号13の出力ポートへ出力します。

⑬機械語プログラムコントロール文

PEEK	150 A=PEEK(49450) 160 B=PEEK(C)	10進番地49450にはいっているデータを10進数に直して変数Aに代入します。 変数Cで指定される10進番地にはいっているデータを10進数に直して変数Bに代入します。
POKE	120 POKE 49450, 175 130 POKE AD, DA	10進番地49450にデータ175(10進表現)をセットします。 変数ADで指定する番地に、変数DAで示される値(0~255の範囲)をセットします。
USR	500 USR(49152) 550 USR(AD) 570 USR(\$C000)	10進番地49152にプログラムのコントロールを移します。このコントロールの移動は、機械語のCALLコマンドと同じ機能を持っています。従って、機械語プログラムに、RETコマンド、(10進コードで201)があると、BASICプログラムへリターンします。 変数ADで指定される10進番地をCALLします。 16進番地C000をCALLします。
LIMIT	100 LIMIT 49151 100 LIMIT A 100 LIMIT \$C000 300 LIMIT MAX	BASICプログラムで使用するエリアを、49150番地(16進でBFFE)に制限します。 BASICプログラムで使用するエリアを、変数A-1の値の番地に制限します。 BASICプログラムで使用するエリアを、16進番地BFFEに制限します。 16進表現する場合は、このように" \$"マークを用います。 BASICプログラムで使用するエリアを、メモリの最大(\$FFFF)に戻します。

⑭グラフィックコントロール文

COLOR		<table border="1"> <thead> <tr> <th>カラーコード</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th></tr> </thead> <tbody> <tr> <td>色</td><td>黒</td><td>青</td><td>赤</td><td>紫</td><td>緑</td><td>水色</td><td>黄</td><td>白</td></tr> </tbody> </table>	カラーコード	0	1	2	3	4	5	6	7	色	黒	青	赤	紫	緑	水色	黄	白
カラーコード	0	1	2	3	4	5	6	7												
色	黒	青	赤	紫	緑	水色	黄	白												
COLOR 1, 2		カラーコードを青色にし、バックカラーを赤とします。																		
COLOR 3		バックカラーは前のまま、カラーコードを紫とします。																		
COLOR, 4		カラーコードは前のまま、バックカラーを緑とします。																		
SET	SET 100, 100	COLOR文の色で (100, 100) にドットをセットします。																		
SET [2] 100, 100		赤色のドットを (100, 100) にセットします。																		
SET[2, W0]100, 100		$\begin{cases} W1 & \dots \text{色を重ね合わせます。} \\ W0 & \dots \text{その色で表示します。} \end{cases}$																		
SET[2, W1]100, 100		(100, 100) のドットに赤色を重ねて表示します。 X座標は0~191、Y座標は0~183まで指定できます。 色の指定はCOLOR文と同じです。																		
POSX		現在のグラフィックポジション（ドット位置）のX座標を示すシステム数値変数です。																		
POSY		現在のグラフィックポジションのY座標を示すシステム数値変数です。																		
LINE	LINE 0, 0, 10, 10, 20, 20	COLOR文の色で、(0, 0) から (10, 10)、(10, 10) から (20, 20) まで線を引きます。																		
LINE[4, W0]30, 50, 70, 50		緑色の線を (30, 50) から (70, 50) まで引きます。																		
CLOSED		色の指定はCOLOR文、色の重ね合わせはSET文と同じです。 X座標は0~191、Y座標は0~183まで指定できます。																		
BOX	BOX 0, 0, 100, 100	COLOR文の色で、(0, 0)、(0, 100)、(100, 0)、(100, 100) を頂点とする四角形を描きます。																		
BOX[5, W1]10, 30, 50, 80		(10, 30)、(10, 80)、(50, 30)、(50, 80) を頂点とする四角形を、現在の色に水色を重ね合わせて描きます。																		
CIRCLE	CIRCLE 100, 100, 30	色の指定はCOLOR文、色の重ね合わせはSET文と同じです。 X座標は0~191、Y座標は0~183まで指定できます。																		
CIRCLE[1, W0]50, 70, 40		COLOR文の色で (100, 100) を中心とした、半径30の円を描きます。																		
CIRCLE[2]50, 60, 70, 1.3		(50, 70) を中心とした、半径40の円を青色で描きます。																		
CIRCLE[1]50, 90, 60, $\pi/4$, $\pi/2$, O		(50, 60) を中心とした横長の楕円を赤で描きます。																		
		(50, 90) を中心とした半径60、角度 $\frac{\pi}{4}$ の扇形を青で描きます。																		
		色の指定はCOLOR文、色の重ね合わせはSET文と同じです。 中心のX、Y座標は -16383 ~ 16383、半径は 0 ~ 16383 まで指定できます。 楕円の縦、横比率は 0 ~ 65535 まで指定でき、1.5 のとき円になります。 円弧または扇形を描くとき、終了角度の次に O を書くと扇形、省略すると円弧になります。																		

POSITION	POSITION 10, 10	PATTERN文のためのX、Y座標を(10, 10)にセットし、ここからPATTERN文を実行します。 X座標は0~191、Y座標は0~183まで指定できます。
PATTERN	PATTERN 4, "ABCD" 10 C \$ = "ABCDEF" 20 PATTERN[4, W1]4, C \$	POSITION文で指定された位置から、右方向4列に、A、B、C、Dの8ビット単位ドットパターンをこの順番で表示します。色はCOLOR文で指定した色になります。 POSITION文で指定された位置から、右方向4列にA、B、C、Dのドットパターンを、A、Bの下にそれぞれE、Fのドットパターンを表示します。色は現在の色に緑色を重ね合わせた色になります。
PAINT	PAINT 50, 100 PAINT[1]0, 0, 2, 4	色の指定はCOLOR文、色の重ね合わせはSET文と同じです。ドットパターンの表示方法(向き、列数)は1~24まで指定できます。 ドットパターン表示位置のX座標が192を越えるとX=0となります。 COLOR文で指定された色を赤とすると、(50, 100)を囲む赤色の境界線の範囲内を赤色でぬりつぶします。 (0, 0)を囲む赤と緑の境界線の範囲内を青色でぬりつぶします。
CLS	CLS CLS1	色の指定はCOLOR文と同じです。 X座標は0~191、Y座標は0~183まで指定できます。 境界線の色は7色まで指定できます。 COLOR文で指定した色を画面から消します。 青色を画面から消します。
(15) ミュージックコントロール文		
MUSIC	10 M \$ = "L3C-7D5E+3" 20 MUSIC M \$	2分音符で低音のド、4分音符でレ、8分音符で高音のミの音を音量3で出します。 音量はL0(無音)~L5(最大)の6段階で指定します。音階は、3オクターブで、低音域は-、高音域は+をつけます。ド、レ、ミ、ファ、ソ、ラ、シはそれぞれ、C、D、E、F、G、A、Bで表わし、C、D、F、G、Aには#(シャープ)がつけられます。また休符として、R、Pの2つが使えます。音長は32分音符(休符)~全音符(休符)までを0~9の数字で指定します。
MUSIC @	MUSIC@ "L333CEG7" 10 M \$ = "L345C # - EG # + 7" 20 MUSIC@ M \$	音量3で、2分音符のドミソの和音を出します。 音量3で低音のドの#、音量4でミ、音量5で高音のソの#を2分音符で出します。 音量、音階、音長の指定方法はMUSIC文と同じですが、音量、音階は3音それぞれ指定が必要で、音長は3音とも同じ長さになります。 休符としてR(3音すべて休み)、P(1音のみ休み)の2つが使えます。

TEMPO	TEMPO7	MUSIC文、MUSIC@文で指定した音を、テンポ7で演奏します。 テンポの指定は1~7の数字で行ない、数字が大きくなる程テンポは速くなります。
SOUND	10 SOUND 0, 1	周期ノイズと呼ばれる効果音を、周波数1、音量10で出します。
SOUND@	20 SOUND @ 10, 0	
LEN	40 SOUND 1, 3 50 SOUND @ 50, 500	ホワイトノイズと呼ばれる効果音を、周波数3(こま切れ音)、分周比500で音量を最大にして出します。
	SOUND0	効果音を止めます。
		効果音の種類は、0(周期ノイズ)、1(ホワイトノイズ)、2(TONE3)の3種類があります。
		周波数は0~3まで指定でき、この順に音が低くなります、ただし3はこま切れ音になります。
		音量は0(無音)~15(最大)まで指定できます。
		分周比は0~1023まで、数字が大きくなる程、音の出る間隔が長くなり、音の高さも低くなります。

⑯データファイル入出力文

WOPEN/T	WOPEN/T ↓ ファイル名 WOPEN/T "A"	カセットテープ中にシーケンシャルデータファイルを作成するために、カセットテープファイルをオープンします。
PRINT/T	PRINT/T A, A\$	カセットテープ中にシーケンシャルデータファイル"A"を作成するために、カセットテープファイルをオープンします。
CLOSE/T	CLOSE/T	WOPEN/T文によってオープンされたファイルをクローズして、カセットテープ上に1つのシーケンシャルデータファイルを作成します。
ROPEN/T	ROPEN/T ↓ ファイル名 ROPEN/T "B"	ROPE/T文を実行している場合は、オープンされたファイルをクローズして、新たに別のファイルのオープンができるようにします。
INPUT/T	INPUT/T B, B\$	カセットテープ中の最初に見つかったBASICシーケンシャルデータファイルに登録されているデータを読み出すためカセットテープファイルをオープンします。
		カセットテープ中のシーケンシャルデータファイル"B"に登録されているデータを読み出すためカセットテープファイルをオープンします。
		ROPE/Tによってオープンされているカセットシーケンシャルデータファイルに登録されているデータを順に読み出し、数値変数B、文字変数B\$に代入します。
		データファイル上のデータ並びと、INPUT/Tの後の変数並びのデータタイプは、それぞれ一致していなければなりません。

⑰数値関数

RND	100 A=RND(1) 110 B=RND(10)	行番号100または110のようにカッコ内に正の整数を与えるとRND関数を使うたびに、順次0.00000001から0.99999999までの間の値をとる乱数値を発生します。(カッコ内に与える正の整数の値には無関係です。)
	200 A=RND(0)	行番号200または210のようにカッコ内に0または負の整数を与えると乱

	210 B=RND(-3)	数発生のイニシャライズが行なわれて、いつもある特定の数値を発生してAにもBにも同じ値が代入されます。(A=B=0.20298827)
SIN	100 A=SIN(X)	変数Xの値(ラジアン)について、sinXの値を求め変数Aに代入します。カッコ内は定数、式でもかまいません。ラジアンと度の関係は、 $1\text{ 度} = \frac{\pi}{180}\text{ ラジアン}$
	110 A=SIN(30*π/180)	ですから、たとえばsin30°の値を変数Aに代入するには行番号110のようにします。
COS	200 A=COS(X) 210 A=COS(200*π/180)	変数Xの値(ラジアン)について、cosXの値を求め変数Aに代入します。カッコ内は定数、式でもかまいません。度で計算するにはSIN関数と同様の方法を使います。行番号210はcos200°の値を変数Aに代入する命令文です。
TAN	300 A=TAN(X) 310 A=TAN(Y*π/180)	変数Xの値(ラジアン)について、tanXの値を求め変数Aに代入します。カッコ内は定数、式でもかまいません。度で計算するにはSIN関数と同様の方法を使います。行番号310はtanY°の値を変数Aに代入する命令文です。
ATN	400 X=ATN(A) 410 Y=180/π*ATN(A)	変数Aの値について、 $\tan^{-1}X$ の値(ラジアン)を求め変数Xに代入します。カッコ内は定数、式でもかまいません。計算結果は $-\frac{\pi}{2}$ と $\frac{\pi}{2}$ の間の値となります。行番号410は $\tan^{-1}X$ の値を度にして変数Yに代入する命令です。
ABS	100 A=ABS(X)	変数Xの値の絶対値 $ X $ を変数Aに代入します。カッコ内は定数、式でもかまいません。 (例) ABS (-3)=3 ABS (12)=12
INT	100 A=INT(X)	変数Xの値について、Xを越えない最大の整数を求めて変数Aに代入します。カッコ内は定数、式でもかまいません。 (例) INT (3. 87)=3 INT (0. 6)=0 INT (-3. 87)=-4
SGN	100 A=SGN(X)	変数Xの値について $X < 0$ のとき-1を、 $X = 0$ のとき0を、 $X > 0$ のとき1を変数Aに代入します。カッコ内は定数、式でもかまいません。 (例) SGN (0. 4)=1 SGN (0)=0 SGN (-400)=-1
SQR	100 A=SQR(X)	変数Xの値について、 \sqrt{X} の値を求めて変数Aに代入します。カッコ内は定数、式でもかまいませんが、正または0の値でなければなりません。
EXP	100 A=EXP(X)	変数Xの値について、 e^x の値を求めて変数Aに代入します。カッコ内は定数、式でもかまいません。
LOG	100 A=LOG(X)	変数Xの値について、常用対数 $\log_{10}X$ の値を求めて変数Aに代入します。カッコ内は定数、式でもかまいませんが、正の値でなければなりません。
LN	100 A=LN(X) 110 A=LOG(X)/LOG(Y) 120 A=LN(X)/LN(Y)	変数Xの値について、自然対数 $\log_e X$ の値を求めて変数Aに代入します。カッコ内は定数、式でもかまいませんが、正の値でなければなりません。 対数の底がYのときの対数 $\log_Y X$ を求めるには行番号110または行番号120によって求められます。

⑯ストリング関数

LEFT\$	10 A \$ = LEFT \$ (X \$, N)	文字変数X \$の最初からN文字目までを、文字変数A \$に代入します。Nは定数でも、変数、式でもかまいません。
RIGHT\$	30 C \$ = RIGHT \$ (X \$, N)	文字変数X \$の右からN文字を、文字変数C \$に代入します。
MID\$	20 B \$ = MID \$ (X \$, M, N)	文字変数X \$の第M文字目からN文字を、文字変数B \$に代入します。
LEN	100 LX=LEN(X \$)	文字変数X \$の文字の長さ（文字数）を、変数LXに代入します。
	110 LS=LEN(X \$ + Y \$)	文字変数X \$とY \$の文字の長さの和を、変数LSに代入します。
SPACE\$	40 D \$ = SPACE \$ (N)	N個のスペースを文字変数D \$に代入します。
STRING\$	50 E \$ = STRING \$ ("*", 10)	10個の連続した*を、文字変数E \$に代入します。
VAL	90 I=VAL(N \$)	文字変数N \$の数字ストリングを、そのまま数値として変数Iに代入します。
STR\$	80 N \$ = STR \$ (I)	VAL関数の逆で、変数Iの数値をそのままストリングとして、文字変数N \$に代入します。（正の数値の+を省略した場合、数字の前にスペースはとられません。）
CHR\$	60 F \$ = CHR \$ (A)	ASC関数の逆で、変数Aの値に等しいASCIIコードの文字（キャラクタ）を文字変数F \$に代入します。Aは定数、変数、式いずれでもかまいません。
ASC	70 A=ASC(X \$)	文字変数X \$の最初の文字のASCIIコード（10進数）の値を、変数Aに代入します。

⑯算術演算子

+	10 A=X+Y(加算)	変数AにXとYの数値の加算結果を代入します。
-	10 A=X-Y(減算)	変数AにXとYの数値の減算結果を代入します。
*	10 A=X*Y(乗算)	変数AにXとYの数値の乗算結果を代入します。
/	10 A=X/Y(除算)	変数AにXとYの数値の除算結果を代入します。
\uparrow	10 A=X \uparrow Y(べき乗)	変数AにX ^Y の計算結果を代入します。 (但しX \uparrow YでXが負数のとき、Yが整数でなければエラーとなります。)
-	10 A=-B(負号)	0-Bは減算ですが、-Bの「-」は負号であることに注意して下さい。

②比較・論理演算子

=	10 IF A=X THEN…	変数AとXの数値が等しいならば、THEN以降の命令を実行します。
20 IF A \$ = "XYZ" THEN…		文字変数A \$ の内容が文字XYZであれば、THEN以降の命令を実行します。
>=, =>	10 IF A>=X THEN…	変数AがXより大きいか等しいならば、THEN以降の命令を実行します。
<=, =<	10 IF A<=X THEN…	変数AがXより小さいか等しいならば、THEN以降の命令を実行します。
<>, ><	10 IF A<>X THEN…	変数AとXの数値が等しくないならば、THEN以降の命令を実行します。
*	40 IF(A>X)*(B>Y) THEN…	変数AがXより大きく、かつ変数BがYより大きいならば、THEN以降の命令を実行します。
+	50 IF(A>X)+(B>Y) THEN…	変数AがXより大きいか、または変数BがYより大きいならば、THEN以降の命令を実行します。

③他のシンボル

?	200 ? "A+B=" ; A+B 210 PRINT "A+B=" ; A+B	PRINTの代わりに用いることができます。したがって行番号200と210は同等です。
:	220 A=X : B=X↑2 : ?A, B	命令文の区切りを表わす記号で、マルチステートメントに使用します。行番号220のマルチステートメントには、3つの命令文が置かれています。
;	230 PRINT "AB" ; "CD" ; "EF"	PRINTを続けて実行します。行番号230では画面上に、「ABCDEF」とスペースを空けずに続けて表示されます。
	240 INPUT "X=" ; X \$	画面に「X=」と表示し、ストリング変数X \$ のデータキー入力を持ちます。
,	250 PRINT "AB", "CD", "E"	ダブルーションをつけてPRINTを実行します。行番号250の文では、画面上にまずABと表示し、次にAから6文字右の場所よりCDと表示し、次にCから6文字右の場所にEと表示されます。ただし、前の文字と重なる場合はさらに6文字右の場所に表示します。
300 DIM A(20), B \$(3, 6)		変数の区切りに用いた例です。
" "	320 A \$ = "BS-BASIC"	" " 内がストリングであることを示します。
\$	340 C \$ = "ABC" + CHR \$(3)	ストリング変数であることを示します。
	500 LIMIT \$BFFF	16進数であることを示します。
π	550 S = SIN(X * π / 180)	円周率の近似値3.1415927を π で表わします。