NAKAMOZU TINY BASIC ユー

このＢＡＳＩＣは，電大版ＴＩＮＹＢＡＳＩＣを中心に，他の様々な言語の長所を取り入れて製作された整数型のインタープリターで，最高速を誇る電大版よりさらにＳＰＥＥＤＵＰされています．

なお名称のＮＡＫＡＭＯＺＵは，大阪府立大学のある中百舌鳥にちなんで付けられたものです．

変数及び配列

〔１〕変数

（１） 変数

- Ａ～Ｚと＠の27種．Base Page に，2 byte づつ割り当てられている．
- 変数のアドレスは，
  ＡＤＲＳ＝（ＡＳＣＩＩ－＄40）＊２
  とその次．

（２） システム変数

“［” ＳＯＵＲＣＥ ＰＲＯＧＲＡＭ 開始ＡＤＲＳ．
“￥” ＳＯＵＲＣＥ ＰＲＯＧＲＡＭ 最終ＡＤＲＳ．
“］” ＭＥＭＯＲＹ上限ＡＤＲＳ．
ＤＩＲＥＣＴ ＭＯＤＥでシステム変数の値を変更することにより，複数のＰＲＯＧＲＡＭをＭＥＭＯＲＹに置ける．

（３） ＩＸ変数

- ％数字：％０～％９の10種．
- 一般の変数と全く同等に使用できる．
- ＩＸ配列の実効ＡＤＲＳの基準になる．（対応するＩＸ配列のＤＩＭＥＮＳＩＯＮを宣言する．）

〔例〕
ＤＩＭＥＮＳＩＯＮ％０（10），％１（15），％８（20）
は次のＰＲＯＧＲＡＭと等価．
％０＝￥＋１：％１＝％０＋20：％８＝％１＋30

- 基準はＰＲＯＧＲＡＭ末（￥）からとらなくてもよい．

〔２〕配列（ＩＸ配列）

- ％数字（式）：％０（式）～％９（式）の10種．
- 使用前に必ず対応するＩＸ変数の値を決定しておく．
- 一般の配列と同様に，変数と同等の機能がある．
- 実効ＡＤＲＳに制限がないので，２バイト単位のＰＥＥＫ，ＰＯＫＥとしても使える．
- （実効ＡＤＲＳ）＝（ＯＦＳＥＴ式）＊２＋（対応するＩＸ変数）
- ＯＦＳＥＴ式の中にＩＸ配列が入ることは許されているので，多重間接指定が可能．

〔３〕定数

（１）10進定数

- －32768 ～＋32767 の整数．＋は省略可能．

（２）16進定数

- ＄０～＄ＦＦＦＦの16進数．（－＄nnnnの形で負の16進定数も使用できる）
- 前に＄をつけることで，プログラム中のどこにも使用できる．

〔例〕 ＶＲＡＭが＄Ｃ000 ～＄Ｃ１Ｆの時．
ＦＯＲＩ＝＄Ｃ000 ＴＯ＄Ｃ１ＦＦ
＃Ｉ＝＄20
ＮＥＸＴＩ
でＶＲＡＭをＣＬＲできる．
（ＰＲＩＮＴするには，ＦＯＲＭＡＴＴＥＲ，ＨＤＴ，ＨＤＦを用いる）

（３）リテラル定数（ＡＳＣＩＩ定数）

- “Ａ”や’Ａ’の形でＡＳＣＩＩ一文字のアスキーコードを値に持つ定数として使える．
- 注：ＰＲＩＮＴ文で用いるリテラルストリングスとは別のものである．

〔例〕
Ｘ＝“Ａ”＋“Ｃ”
ＰＲＩＮＴＣＨＲ＄（Ｘ／２）
結果“Ｂ”を出力する．
ＩＦ ＧＥＴ＄＝“Ｎ” ＥＮＤ

（４）リテラル・ストリングス

- ＰＲＩＮＴ文中でのみ有効な文字列定数．

・“ＡＢＣＤ”や’ＡＢＣＤ’の形で，ＰＲＩＮＴする文字列をそのまま書く．

（５）論理演算子

●〈 〉＝の自由な組合せ

すべて成立すれば１，しなければ０の値を持つ．演算子の優先順位は一般的なもの．

COMMAND

ＣＯＭＭＡＮＤはＤＩＲＥＣＴ　ＭＯＤＥのみで使用する．

LIST 〈・ｏｒＬ．〉
- ●ＬＩＳＴ　全ＴＥＸＴをリスト
- ●ＬＩＳＴｍ　ｍ行目をリスト
- ●ＬＩＳＴｍ，ｎ　ｍ行～ｎ行までリスト
- ●ＬＩＳＴｍ，ｏ　ｍ行～最後までリスト

SAVE 〈Ｓ．〉

出力装置をカセットに設定し，ＴＥＸＴを出力，一行ごとに0.5秒ＤＥＬＡＹを置く．使用モードはＬＩＳＴと同様．

ＦＯＲＭＡＴ

イニシャライズ．　＄12　1秒
行先頭コード　＄02
行番号，ＴＥＸＴ
ＣＲ
ディレイ　0.5秒
・
・
＄03，＄13，＄14

LOAD 〈ＬＯ．〉

ＮＥＷ処理後ＴＡＰＥからＴＥＸＴを入力，イニシャライズ＄11を出力．解除のコントロールコードはテープから入力する．

APPEND 〈ＡＰ．〉

カセットからＴＥＸＴを編集しつつ入力する．他はＬＯＡＤと同様．

AUTO 〈Ａ．〉

●ＡＵＴＯｍ（ｍ≧１）　ｍから10おきに自動的に行番号とスペースを出力する．解除はＣＯＮＴＲＯＬ－Ｃ．

NEW 〈Ｎ．〉

ＴＥＸＴをクリアし，各種ポインタ・スタックをイニシャライズする．

RUN

各種ポインタ・スタックをイニシャライズし，一番最初の行から実行する．

DEL

●ＤＥＬｍ　ｍ行目から後をクリアし，ポインタ・スタックをイニシャライズする．〈Ｄ．〉

EXIT 〈Ｅ．〉

モニタ・プログラムにもどる．

STATEMENT

ＰＲＯＧＲＡＭ中で使用できるＣＯＭＭＡＮＤ．
※のものはＤＩＲＥＣＴ　ＭＯＤＥで使用できる．
このＢＡＳＩＣではＬＥＴ文は存在せず，全面省略して直接代入文を書く．

※FOR TO STEP 〈Ｆ.ＴＯＳ.〉

●ＦＯＲ（代入文）ＴＯ（式）ＳＴＥＰ（式）ＳＴＥＰ１の時，ＳＴＥＰは省略できる．

ＦＯＲ－ＮＥＸＴループはＦＯＲ・ＮＥＸＴ専用のスタックを使用しており，ＤＯ－ＵＮＴＩＬループとの併用も問題ない．

ＦＯＲの次の代入文の制御変数には，変数，ＩＸ変数，ＩＸ配列が使用でき，多重ループの時，同じ制御変数は使えない．

ＴＯ及びＳＴＥＰの次の式の値は，最初に記憶されるのでループ内で変更しても影響はない．

ネスティングは７レベルまで許される．

※NEXT 〈.or Ｎ.〉

●ＮＥＸＴ変数

ＮＥＸＴの次に書く制御変数は，ＦＯＲのものと一致させなければならない．

このＢＡＳＩＣでは制御変数は省略でき，自動的に判断させることができる．

※DO

ＵＮＴＩＬ文とペアになり，ループを構成する．ＩＦ文とＧＯＴＯ文によるループに変わるもので，行番号ジャンプでないだけスピードアップされる．

スタックはＧＯＳＵＢ－ＲＥＴと同じものを使用しており，ネスティングは27レベルまで許される．

※UNTIL 〈Ｕ.〉

●ＵＮＴＩＬ（式）

式の値が０であれば前のＤＯに戻る．

ＤＯ－ＵＮＴＩＬループの途中からの脱出は，ＵＮＴＩＬ１，でスタックを元に戻してから行なう．

※GOTO 〈Ｇ.〉

●ＧＯＴＯ（式）

式の値の行番号へＪＵＭＰする．

式の値が0の時は，次の文へ行く．

※GOSUB
〈GOS.〉

●GOSUB（式）

式の値の行番号からのサブルーチンへJUMPする．戻りのADRSは，DO－UNTILと同じスタックにPUSHされている．

式の値が0の時は，次の文へ行く．

※RET
〈R.〉
DIRECT MODEの時
〈RET〉

サブルーチンの終わりに使用し，スタックから戻りADRSを引き出し，そのADRSへJUMPする．

UNTIL0と動作は似ているが，UNTIL0はスタックポインタを変更せずに戻りADRSを取り出す点が異なる．

ダイレクトモードでは，STOP文で停止した続きを実行する．（CONTINUEの機能）

※IF

●IF（式）THEN（文）：（文）：（文）

IFの次の式の値が0以外ならTHENに続く文を実行し，0の時は次の行を実行する．

THENは省略可能で，THENGOTOmはTHENm，又はGOTOmのみでもよい．

※PRINT
〈P.〉

●PRINT（式）連結子“リテラル・ストリング”

連結子には，“，”と；”があり，前者はスペースを含めて8文字後に次のものをPRINT，後者は改行することなく続いてPRINTするためのものである．

リテラルストリングには，“ストリング”’ストリング’の2つの形が使える．

TAB，CHR$，HDT，HDF，USING等のFORMATTERによりPRINTFORMATを決定できる．

REM

●REM（メッセージ）

REMARKS文は，PROGRAM中に注釈を書くのに使用する．

BASICはこの文を読みとばす．

＊(DATA)

●＊（式），（式），……

DATA文は，READ関数，RESTORE文と組み合わせて使用され，一般にはPROGRAM中にデータを置くのに使われる．

DATA文はマルチステートメントの2番目以降以外の任意の場所に置くことができ，PROGRAMのEND以降でもよい．

このBASIC固有の特長として，DATA内容として10進定数の他，16進定数，リテラル定数，変数（配列），さらに式を置くことも可能で，一般のDATA文の範囲を超えた使用法が考えられる．

たとえば，READが関数であることを利用し，関数の定義を行なうこともできる．

RESTORE

●RESTORE（式）

DATA文のポインタは，END，及びRUNコマンドの実行で，PROGRAMの先頭に設定され，READ関数で読み出されるたびに更新されるが，RESTORE文で任意の文の先頭に移すことができる．

※THEN
〈T.〉

●THEN（ステートメント）

●THEN（行番号）

IF文とペアで使用され，IF文の後の式が0以外の値を持った時，THENに制御が移る．

うしろにステートメントがある時，THENは全面的に省略でき，またそのうしろはマルチステートメントであってもよい．

うしろに行番号がある時は，GOTO文を使わずに指定の行にJUMPできる．

※COPY
〈CO.〉

オプションステートメント．

現在は，V－RAM1画面をプリンタに出力するステートメントとされている．

INPUT
〈IN.〉

●INPUT（リテラルストリングス），（変数），……，（変数）

INPUT文は，（リテラルストリングス）がある時はそれをPRINTした後，改行しないで，必要な数だけ“？”を出力しインプットを待つ．

変数には，IX変数，IX配列も使用でき，入力データとしては，数字，16進定数，変数，IX変数，IX配列，式，が可能である．

不適当なKEYINに対しては，RE－ENTERが表示され，再度入力待ちとなる．

END
〈E.〉

ＥＮＤ文の実行によりプログラムの実行を終了し，コマンドモードに移る．

ＥＮＤ文はＰＲＯＧＲＡＭ中いくつあってもよい．ＥＮＤ文，ＳＴＯＰ文がない場合は最後の行の実行後，停止する．

STOP
〈S.〉

●ＳＴＯＰ（リテラルストリングス）

ＳＴＯＰ文は，ＰＲＯＧＲＡＭのデバッグ用のブレークポイントとして使用するもので，リテラルストリングスがあればそれをＰＲＩＮＴしてから，実行ポインタをＩＸスタックにＰＵＳＨして実行を停止する．

そのため，ＤＩＲＥＣＴ　ＭＯＤＥでＲＥＴを実行することにより，中断された箇所から実行を再開できる．

機械語のＳＷＩと同等の機能と考えてよい．

※＃(POKE)

●＃（数字or変数）＝（式）

●＃（式）＝（式）

式の左辺に表われる＃はＰＯＫＥと同等のはたらきをするステートメントである．

＃（ＡＤＲＳ）＝（ＤＡＴＡ）の型でメモリーにデータを書き込める．ＡＤＲＳが数字，16進定数，変数の場合，カッコは省略できる．

※CLR
〈C.〉
〈C.(X,Y)〉

●ＣＬＲ

●ＣＬＲ（式Ｘ，式Ｙ）０≦Ｘ≦Ｙ≦15

引数なしの場合，Ｖ－ＲＡＭの全面をブランクにする．引数がある時は，Ｘ行目～Ｙ行目までをクリアする．

※CURS
〈CU.(X,Y)〉

●ＣＵＲＳ（式，式）　０≦Ｘ≦31，
０≦Ｙ≦15

ＣＵＲＳＯＲを，座標（Ｘ，Ｙ）に移す．

※NEG

●ＮＥＧ

Ｖ－ＲＡＭ上のスペース，及びグラフィック部分を白黒反転する．

※！W

●！Ｗ（式，式）　０≦Ｘ≦63
０≦Ｙ≦47

グラフィックステートメントで，グラフィック座標（Ｘ，Ｙ）を白にする．

※！B

●！Ｂ（式，式）　０≦Ｘ≦63
０≦Ｙ≦47

座標（Ｘ，Ｙ）を黒にする．

※！R

●！Ｒ（式，式）　０≦Ｘ≦63
０≦Ｙ＝47

座標（Ｘ，Ｙ）を反転する．

FUNCTION

代入文の右辺及び式の中で用いられる命令．
引数にはすべて（式）を使用できる．

RND
〈R.(X)〉

●ＲＮＤ（Ｘ）

Ｘ≧１のとき，０～Ｘ－１の乱数，
Ｘ≦－１のとき，Ｘ＋１～０の乱数．

引数０は，０による割り算となりＥＲＲＯＲとなる．

MOD
〈M.〉
〈M.(X,Y)〉

●ＭＯＤ
●ＭＯＤ（Ｘ，Ｙ）　Ｙ≠０

引数なしの場合，直前に行なった割算の余りが入る．ＲＮＤ関数を使った時はその値が入る．

引数のある場合，Ｘ／Ｙの余りが入る．

ABS
〈A.(X)〉

●ＡＢＳ（Ｘ）

Ｘの絶対値を与える関数．

SGN
〈S.(X)〉

●ＳＧＮ（Ｘ）

Ｘ＞０のとき１
Ｘ＝０のとき０
Ｘ＜０のとき－１　を与える関数．

論理式（Ｘ＞０）－（Ｘ＜０）と等価であるが，ＳＧＮを用いる方が速い．

AND

●ＡＮＤ（Ｘ，Ｙ）

ＸとＹのbit ごと（16 bit）のＬＯＧＩＣＡＬＡＮＤを与える関数．

OR

●ＯＲ（Ｘ，Ｙ）

ＸとＹのbit ごとのＯＲを与える関数．

XOR
〈X.(X,Y)〉

●ＸＯＲ（Ｘ，Ｙ）

ＸとＹのbit ごとのＥＸＯＲを与える関数．

READ 〈RE.〉

●READ

このBASICの最もユニークな関数で，DATAのポインタが示すDATAの値を与える関数．DATA文，RESTORE文と併用する．

関数であるから一般の式の中に自由に使用できる．対応するDATAは式でもよい．

一般のBASICのREAD文と等価な使用法の例を挙げる．

一般のBASIC－－READA，B

NTB－－－A＝READ：B＝READ

GET$ 〈G.〉

●GET$

ASCII一文字入力関数．

この関数が実行されると，入力待ちになり，一文字キーインするとそのASCIIコードが値として与えられる．

CRは必要ない．また，コントロールコードも入力可能．改行はされない．

使用例：PRINT　GET$

IFGET$＝“E”THENE

ND

KEY 〈K.〉

●一般のフルキーボード使用の場合．

●KEY

この関数が実行された時KEYを押していれば，そのASCIIコードを与え，押していなければ，100 を与える．

これはリアルタイムKEY入力で，入力待ちをしない．

●H68TRコンソール使用の場合．

●KEY　　コンソールの0～Zまですべて

●KEY（X，Y）　0≦X≦Y≦5　コンソールのX列～Y列のみをスキャンする．

KEYを押していれば，そのKEYコード．押していなければ100 を与える．

#(PEEK)

●#（式）or#（変数or数字）

式，変数，数字，16進定数，リテラル定数の値のADRSの内容を与えるPEEK関数．ADRSが式で表現されていない場合，カッコは省略できる．

!P

●!P（式，式）　0≦X≦63　0≦X≦47

グラフィック座標（X，Y）が白なら1，黒なら0，キャラクタなら100 を与える関数．

USER 〈U.(X,Y,Z)〉

PRINT文中では省略形は使えない．

●マシン語サブルーチンとリンクする関数．

マシン語ルーチンはRTSにより終了する．

○USER（式1，式2，式3）

PC←式1

IX←式2

AccB，A←式3

○USER（式1，式2）

PC←式1

IX←式2

AccB，A←ゼロ

○USER（式）

PC←式

IX，AccA，B←ゼロ

例：K＝USR（$1000，A，B＋C）

ユーザールーチンからリターンすると，AccBが上位，AccAが下位となってKに代入される．

FORMATTER

PRINT文中で使用し，PRINTFORMATをコントロールする．

USING 〈U.〉

○USING文字（第1カラム）文字（第2カラム）（式）　カッコは省略可

式の値の数字を次のFORMATで右づめ出力する．

FORMATは，最初が符号部，次の5ケタが数値部となっている．

○式の値が正又は0の時．

符号部を第1カラムの文字，数字出力部(右づめ）の上位のスペースを第2カラムの文字が埋める．

○式の値が負の時．

符号部には“－”が出力され，数値部の左側スペースに第2カラムの文字が入る．

例：A＝12，B＝－345 のとき，

```
PRINT USING␣␣A:␣␣␣␣12
PRINT USING␣␣B:－␣␣345
PRINT USING＊＊A:＊＊＊＊12
PRINT USING＊＊B:－＊＊345
PRINT USING+␣A:+␣␣␣12
PRINT USING+␣B:－␣␣345
PRINT USINGABA:ABBB12
```

TAB ・TAB（式）．カッコは省略可．

〈T.〉

スペースを式の値だけ出力して，CURSORを進める．ただし，その行の左端からTABがとられるので，ひとつのPRINT文中に下の例のごとく複数のTABがある場合は，

PRINTTABA；“＊”；TABB；“#”

B＞Aの時のみ意味を持ち，B≦Aの時，TABBは無視される．

**CHR$** ・CHR$（式），カッコは省略可

〈C.〉

式の値のASCIIコードの文字を与える．

コントロールコードも使える．

例：

```
PRINT CHR$65    ……A
PRINT CHR$$42   ……B
PRINT CHR$ GET$+1
 …… キーインS……出力T，
```

N．T．Bのオペレーティングシステムでは，次のコントロールコードが定義されています．

CHR$4　0，5秒ディレイ．
CHR$5　画面をクリアし，カーソルを左上へ．
CHR$6　発振音LOW
CHR$7　発振音HIGH
CHR$17　入力をカセットに設定
CHR$18　出力をカセットに設定
CHR$19　カセット入力を解除
CHR$20　カセット出力を解除

HDF ○HDF（式），カッコは省略可

（Hexa Decimal・Four digits）

式の値を16進4ケタに変換する．

HDT ○HDT（式），カッコは省略可

（Hexa Decimal Two digits）

式の値を16進2ケタに変換する．

; セミコロン．PRINT文中で，数字，変数，式及びFORMATTERを連続してPRINTするための連結子．

改行せずに続けてPRINTする．

, カンマ，PRINT文中の連結子で，直前にPRINTしたものから右の8字ごとに区切られた位置にCURSORを移して次のPRINTを行なう．

これらの連結子でPRINT文を終わるとこの効果は次のPRINT文まで持続する．

ERROR MESSAGE

NO.	メッセージ	意　味
1	INPUT ERROR	1行に72文字以上入力された． INPUT文の変数及び入力ERROR．
2	MEMORY FULL	プログラム・サイズが設定したメモリー範囲を越えた．
3	INVALID LINE NUMBER	行番号負，0または32768以上になった．
4	PRINT ERROR	PRINT文，INPUT文，STOP文中のLITERAL STRINGS ERROR．
5	DIVIDE BY ZERO	0で割ろうとした． RND(0)，MOD(A，0)を実行した．
6	ARITHMETIC STACK ERROR	算術のスタック・オーバー，カッコのネスティングオーバー
7	ILLEGAL ARITHMETIC	算術式，及び関数ERROR．
8	MOD ERROR	MODの使い方がおかしい．
9	UNRECOGNIZED STAMENT	STATMENTがおかしい．
10	INDEX STACK ERROR	GOSUBなしにRETまたは，DOなしにUNTIL がある． GOSUBとDOのネスティング合計が27を越えた．
11	LINE NUMBER NOT FOUND	GOTO，GOSUBの飛び先がない． RESTORE，DEL，で行番号が見つからない．
12	FOR NEXT STACK ERROR	FOR，NEXTのネスティングが7を越えた．
13	NEXT WITHOUT FOR	対応するFORのないNEXTが存在する． FOR NEXT LOOPが混線している．
14	INDEX ARRAY ERROR	IX変数及びIX配列の使い方がおかしい．
15	GRAPHIC ERROR	GRAPHIC STATMENTの使い方がおかしい．
16	OVER SIZED PARAMETER	CURS，CLR，KEY，及びグラフィックでの引き数が範囲を越えた．
17	READ ERROR	READ関数で，読むべきDATAがない．