2019年1月14日更新

■■KM-BASICについて■■

この文書はMachiKaniaに搭載のKM-BASICの使い方を解説するものです。
マニュアルのような全てを網羅するものではなく、入門書のように特に気を付けることや
知っておくと便利なことなど、思いつくままに追加していきたいと思います。

■基本ルール

・行番号はあってもなくてもよく、行番号の大小は実行順とは無関係です。
行番号はGOTOやGOSUBの飛び先として使われますが、LABEL命令を用いて6文字以内のラベル名を
指定することで行番号は不要となります。

・大文字と小文字の区別について
命令や関数名、変数名では大文字、小文字は区別されません。

・1行に複数の命令を記述する場合、「:」（コロン）でつなぎます。
（例）
```
A=10:B=20
```

・命令と引数の間には必ず1つ以上のスペースが必要です。
行頭には複数のスペースを入れることが可能です。
ループや条件文、サブルーチンなどで行頭スペースを適切に入れることで、
プログラムが見やすくなります。

・使用可能な数値
整数　32ビット符号付整数(-2147483648 以上 +2147483647 以下)が使用可能です。
10進数は単に数字で記述します。
16進数は先頭に$または0Xを付加します。
（例）
```
A=-123
B=$ABCD0123
C=0XABCD0123
```

実数　32ビットの単精度浮動小数点数が使用可能です。変数名には#を付加します。
（例）
```
A#=3.14159
B#=2.7E-2
```

・文字列について
文字列は" "で囲んで記述します。変数名には$を付加します。
（例）
```
S$="ABCDEFG"
```

・関数について
KM-BASICでは様々な戻り値を返す関数を用意しています。
関数はFUNC()のように関数名の後ろに括弧が付きます。
括弧内には必要に応じて引数が入ります。引数に使える型は厳密に定められており、異なる型を

指定した場合、コンパイル時に文法エラーとなります。
また、戻り値の型によって関数名の後ろに記号が付きます。

戻り値の型の種類
　整数型　：なし（例）K=KEYS()
　実数型　：#　　（例）Y#=SIN#(X#)
　文字列型：$　　（例）S$=DEC$(N)

・MachiKania でのプログラムの実行方法と終了方法
F4 キー（RUN）で実行します。
END 命令で終了します。
強制終了する場合、[Ctrl]＋[Break]キーを押します。
終了しない場合は、基板上の[UP][DOWN][START][FIRE]ボタンを同時に押します。

■KM-BASIC で使える変数

◎変数名について
特に指定しなければ、変数名は A～Z のアルファベット 1 文字のみとなります。
後述の USEVAR 命令で宣言することで 6 文字までの英数を使用することができます。

◎変数の型について
整数型、文字列型、実数型があります。変数名の後ろに付ける記号で区別されます。

・整数型
変数名の後ろに記号がない場合、整数型変数となります。
32 ビット符号付整数で-2147483648～+2147483647 を表すことができます。

＜使用例＞
A=123

・文字列型
変数名の後ろに$を付けると文字列型変数となります。

＜使用例＞
A$="XYZ"

実数型
変数名の後ろに#を付けると、浮動小数点の実数型変数となります。
32 ビットの単精度です。

＜使用例＞
A#=3.14159

◎配列の使い方
変数名の後ろに(n)を付けることで配列を使用することができます。nは0以上の整数です。
使用する前にDIM命令で配列の記憶領域を確保します。

＜使用例＞
```
DIM A(10)
A(0)=5
```

DIM A(10)とするとA(0)～A(10)の11個の変数が使用可能となります。
整数型以外にもA#(10)のように指定することで使用可能です。

また、多次元配列にも対応しています。
＜使用例＞
```
DIM A(3,5)
FOR I=0 TO 3
FOR J=0 TO 5
A(I,J)=I*J
NEXT
NEXT
```

◎ローカル変数について
特に指定しなければ、変数はプログラム全体で共通のものとなりますが、VAR指定することで
サブルーチン内でのみ使用される変数とすることができます。
VAR指定は型の種類、配列、長文字変数名のいずれも対応しています。

＜使用例＞
```
X=1
GOSUB SUB1
PRINT X
END
LABEL SUB1
VAR X
X=2
PRINT X
RETURN
```

＜実行結果＞
```
2
1
```

◎長文字変数名
特に指定しなければ変数名はA～Zのアルファベット1文字のみとなりますが、USEVARで宣言する
ことで最大6文字までの変数名を使用可能となります。
USEVAR命令のある行以降で使用可能となります。ここでは型の種類は指定しません。
配列、ローカル変数でも使用できます。
変数名に使用できる文字は、先頭がアルファベットで、2文字目からはアルファベットまたは数字

です。

＜使用例＞

```
USEVAR ABC          長文字変数名 ABC の使用を宣言
ABC=100             グローバルの整数型変数 ABC に 100 を代入
GOSUB SUBR1         サブルーチン SUBR1 を呼び出し
GOSUB SUBR2         サブルーチン SUBR2 を呼び出し
PRINT ABC           グローバルの整数型変数 ABC の値を表示（100）
END                 プログラム実行終了

LABEL SUBR1         サブルーチン呼び出し用ラベル
VAR ABC#            サブルーチン内ローカルの実数型変数 ABC#を宣言
ABC#=1.5            ローカル変数 ABC#に 1.5 を代入
PRINT ABC#          ローカル変数 ABC#の値を表示（1.5）
RETURN              サブルーチン終了

LABEL SUBR2         サブルーチン呼び出し用ラベル
VAR ABC$            サブルーチン内ローカルの文字列型変数 ABC$を宣言
ABC$="XYZ"          ローカル変数 ABC$に文字列"XYZ"を代入
PRINT ABC$          ローカル変数 ABC$の文字列を表示（XYZ）
RETURN              サブルーチン終了
```

＜実行結果＞

```
1.5
XYZ
100
```

■使用可能な演算子について

＜整数演算子＞

-x	符号を反転
x + y	整数加算
x - y	整数減算
x * y	整数乗算
x / y	整数除算
x % y	整数剰余
x >> y	xの値をｙビット右シフト
x << y	xの値をｙビット左シフト
x = y	２つの整数値が等しい場合に１、そうでないときに０

x != y　２つの整数値が等しい場合に０、そうでないときに１

x < y　xがyより小さい場合に１、そうでないときに０

x <= y　xがyより小さいか等しい場合に１、そうでないときに０

x > y　xがyより多きい場合に１、そうでないときに０

x >= y　xがyより多きいか等しい場合に１、そうでないときに０

x AND y　xとyの値のビットごとの AND（論理積でないことに注意）

x OR y　xとyの値のビットごとの OR

x XOR y　xとyの値のビットごとの XOR

なお、整数演算子の優先順位は、優先度が高いものから以下の順です。

+ -（単項演算子）
* / %
+ -（加算・減算）
<< >>
< <= > >=
= !=
XOR
AND
OR

＜文字列演算子＞

x$ + y$　文字列の連結

＜浮動小数点型演算子＞

-x#　符号を反転

x# + y#　実数加算

x# - y#　実数減算

x# * y#　実数乗算

x# / y#　実数除算

x# = y#　２つの実数値が等しい場合に１、そうでないときに０

x# != y#　２つの実数値が等しい場合に０、そうでないときに１

x# < y#　xがyより小さい場合に１、そうでないときに０

x# <= y#　xがyより小さいか等しい場合に１、そうでないときに０

x# > y#　　x が y より多きい場合に１、そうでないときに０

x# >= y#　x が y より多きいか等しい場合に１、そうでないときに０

x# AND y#　x と y の値の論理積（ビットごとの AND でないことに注意）

x# OR y#　x と y の値の論理和（ビットごとの OR でないことに注意）

なお、実数演算子の優先順位は、優先度が高いものから以下の順です。

+ -（単項演算子）
* /
+ -（加算・減算）
< <= > >=
= !=
AND
OR

■数値および文字列の型の厳密性について

KM-BASIC では整数値、実数値、文字列は厳密に区別されており、変数への代入や演算、
命令や関数の引数、関数の戻り値などで異なる型を指定するとコンパイルエラーとなります。
そのため、必要に応じて型の変換を行う関数が用意されています。

＜例＞画面にサインカーブを表示

```
USEGRAPHIC
FOR I=0 TO 255
X#=FLOAT#(I)/128*PI#
PSET I,INT(SIN#(X#)*100+100),2
NEXT
WHILE INKEY()=0:WEND
```

■型変換関数

以下の型変換関数が用意されています。

◎INT(x#)

実数値 x#の整数部分を整数値を返します。

◎FLOAT#(x)

整数値 x を実数値に変換して返します。

◎VAL(x$)

10 進数または 16 進数の文字列としての x$を整数値で返します。

◎VAL#(x$)

10 進実数表現の文字列としての x$を実数値で返します。

◎ASC(x$)

文字列の最初の１文字のアスキーコードを返します。

◎CHR$(x)
x をアスキーコードとする１文字を返します。

◎FLOAT$(x#)
実数値 x#を 10 進数の文字列として返します。

◎DEC$(x)
整数値 x を 10 進数の文字列として返します。

◎HEX$(x,y)
y は省略可能。整数値 x を 16 進数の文字列として返します。
y が指定された場合、y バイト長の文字列とします。

◎SPRINTF$(x$,y#)
x$で示される書式に従って、実数 y#の内容を文字列として返します。

＜使用例＞
```
X#=3.14159
PRINT SPRINTF$("%.2f",x#)
```

＜実行結果＞
```
3.14
```

■プログラムの流れの制御について

プログラムの流れを制御する命令について説明します。

◎GOTO
指定された行番号または LABEL 命令で指定された行までジャンプします。
行番号は行頭に付けることができる数字です。
LABEL 命令では 6 文字までのラベル名を指定することができます。

◎WAIT x
60 分の x 秒間、プログラムを停止します。
正確にはテレビへの信号出力が画面最下行を表示し終えた時にカウントアップし、x 回となったところで実行再開します。（テレビ信号は 1 フレーム約 60 分の 1 秒）
SYSTEM 命令を使用して画面表示を停止した状態では、WAIT 命令は機能しません。

※MachiKania type M では画面表示停止中も WAIT 命令は機能します。MachiKania type Z でも今後改善の可能性があります。

◎END
実行中の BASIC プログラムを停止します。

■ループ関連

◎FOR～NEXT
使用例のように、FOR の後ろに整数変数で繰り返したい数を記述します。
NEXT 命令で変数を更新し、最後の数になるまで実行します。

＜使用例＞
```
FOR I=1 TO 10:PRINT I;:NEXT
```

＜実行結果＞
```
12345678910
```

◎WHILE～WEND
WHILE の後ろに条件式を書き、条件が満たされている間、WEND との間を繰り返します。
最初から条件が満たされない場合は、一度も実行せずに WEND の次にジャンプします。

＜使用例＞
```
WHILE INKEY()=0:WEND
```
INKEY()関数はキーボードの押されているキーコードを返す関数です。
何も押されていない場合 0 を返すので、キーボードが押されるまでループします。

◎DO～LOOP
条件付き繰り返し。書き方に 4 パターンあります。

１．最初に繰り返す条件をチェックする。一度も実行しない場合あり。
```
DO WHILE ＜条件式＞
  ＜実行したい内容＞
LOOP
```

２．最初に終了する条件をチェックする。一度も実行しない場合あり。
```
DO UNTIL ＜条件式＞
  ＜実行したい内容＞
LOOP
```

３．最初に実行し、繰り返す条件をチェックする。（一度は必ず実行する。）
```
DO
  ＜実行したい内容＞
LOOP WHILE ＜条件式＞
```

４．最初に実行し、終了する条件をチェックする。（一度は必ず実行する。）
```
DO
  ＜実行したい内容＞
LOOP UNTIL ＜条件式＞
```

＜使用例＞
```
A=1
DO UNTIL A=10
```

```
  PRINT A;
  A=A+1
 LOOP
```

＜実行結果＞
```
123456789
```

◎BREAK、CONTINUE
BREAK、CONTINUE ともFOR～NEXT、WHILE～WEND、DO～LOOP のループ内で実行します。
BREAK はループから無条件に抜け出します。
CONTINUE はNEXT、WEND、LOOP 命令にジャンプし、条件判断してループを続けます。

＜使用例＞
```
FOR I=1 TO 10
  IF I=4 THEN CONTINUE        Ｉが４の場合、NEXT 命令にジャンプしループを続ける
  IF I=7 THEN BREAK           Ｉが７の場合、NEXT 命令の次にジャンプしループを抜ける
   PRINT I;
NEXT
```

＜実行結果＞
```
12356
```

■サブルーチン関連
まとまった機能をサブルーチンとして定義することで、プログラムを読みやすくし、
また同じ機能を複数個所で記述する手間を防ぐことができます。
サブルーチンの最初に行番号を記述または、LABEL 命令で６文字までのサブルーチン名を
定義します。

◎GOSUB～RETURN
指定された行番号またはLABEL 命令で指定されたサブルーチンを呼び出し、
RETURN 命令で戻ってきます。

◎GOSUB(NAME,a,b,・・・)～RETURN x
◎GOSUB#(NAME,a,b,・・・)～RETURN x#
◎GOSUB$(NAME,a,b,・・・)～RETURN x$
サブルーチンを呼び出し、戻り値を得る関数です。戻り値の型により、#や$を付加します。
NAME はラベル名または行番号で指定します。a,b,・・・は引数で、整数、実数、文字列を
指定することができます。引数はサブルーチン内のARGS()関数で取り出します。
戻り値はRETURN 命令の引数で指定します。

◎ARGS(x)
◎ARGS#(x)
◎ARGS$(x)
サブルーチン中で、GOSUB(NAME,a,b,・・・)に渡されたｘ番目の引数を取り出す関数です。
引数の種類により#や$を付加します。

＜使用例＞

```
A=GOSUB(NAME,10,20)      括弧内に飛び先と引数を記述。NAMEは飛び先ラベル名
PRINT A
END
LABEL NAME               飛び先ラベル名
VAR X,Y                  ローカル変数指定
X=ARGS(1)                1番目のパラメータを取得（10）
Y=ARGS(2)                2番目のパラメータを取得（20）
RETURN X+Y               戻り値としてX+Yを指定
```

＜実行結果＞

```
30
```

■条件判断

◎IF ＜条件＞ THEN～ELSE

条件により実行するかしないかを指定します。
THENの後ろに条件を満たした場合に実行する内容を記述します。
ELSEの後ろに条件を満たさなかった場合に実行する内容を記述します。

＜使用例＞

```
IF A=10 THEN A=0                Aが10の場合、Aを0にする。

IF A=10 THEN A=0 ELSE A=A+1     Aが10の場合、Aを0にする。それ以外はAを1足す。

IF A=10 THEN GOTO CASE10        Aが10の場合、ラベル名CASE10の位置にジャンプ
```

◎IF THEN～ELSEIF～ELSE～ENDIF

通常のIF命令と違いTHENの後ろで改行すると、ENDIFとなるまで複数の命令を記述できます。
ELSEIF節やELSE節を指定することで、別の条件や条件を満たさなかった場合の命令を複数記述できます。
最後に必ずENDIFを記述します。

＜使用例＞

```
IF A=10 THEN       Aが10の場合、次の2行を実行
  A=0
  B=0
ELSEIF A=20 THEN   Aが20の場合、次の2行を実行
  A=1
  B=1
ELSE               その他の場合、次の2行を実行
  A=A+1
  B=B+1
ENDIF              必ず最後にENDIFを記述する
```

■条件判断について

WHILE、UNTIL、IF、ELSEIFの後ろに付く条件式は、0以外の数の場合条件を満たし、
0の場合条件を満たさなかったと判断します。

例えば、IF X=5 THEN・・・とあるとき、「X=5」という比較演算は、Xが5の場合1を返し、
Xが5以外の場合0を返します。

IF X=5 AND Y=3 THEN・・・の場合、まず「X=5」の部分はXが5の場合1を返し、
「Y=3」の部分はYが3の場合1を返します。次に1と1のAND演算（ビット演算）で結果が1となり、
0ではないので条件を満たすことになります。

条件式は必ずしも比較演算を記述する必要はありません。以下のような記述が可能です。

```
WHILE 1                 条件式を1に固定（永久ループ）
 IF INKEY() BREAK       何かキーが押されていれば0以外となりループを抜ける
WEND
```

■テキスト表示関連

MachiKania type Zにはテキストモードとグラフィックモードがあります。
実行初期状態は横30文字、縦27文字のテキストモードなっています。
WIDTH命令を使うことで横40文字モードにも対応します。

[type M]
さらに、横36文字、48文字、80文字（モノクロ）モードにも対応しています。

色は256色同時表示が可能で、カラーパレット番号で指定します。実行初期状態のカラーパレット
設定は以下のようになります。

0：黒、1：青、2：赤、3：マゼンタ、4：緑、5：シアン、6：黄、7：白
8：黒、9～15：1～7の明るさを半分にしたもの
16～255：白

テキストモードでは以下の命令が使用できます。

◎CURSOR x,y
表示する位置を指定します。

◎BGCOLOR r,g,b
バックグラウンドカラーを指定します。r,g,bはそれぞれ赤、緑、青の強さを0～255で表します。

◎COLOR x
表示する文字の色を指定します。xはパレット番号です。

[type M]
モノクロモードでは、パレット番号128以上が反転文字となります。

◎CLS
テキスト画面をクリアします。

◎PALETTE n,r,g,b
カラーパレットの設定を行います。

n：カラーパレット番号
r,g,bはそれぞれ赤、緑、青の強さを0～255で表します。

[type M]
テキストとグラフィックでカラーパレットが共通のため、グラフィック画面にも反映されます。

◎PRINT
数値や文字列を表示します。
＜使用例＞
PRINT X;Y;　　整数変数XとYを続けて表示。最後に「;」がある場合改行しない
PRINT A$　　文字列変数A$を表示。最後は改行する
PRINT F#+G#　　実数変数F#とG#の和を表示、最後は改行する
PRINT "A=";A　　文字列と変数を続けて表示、最後は改行する

◎SCROLL x,y
テキスト画面をスクロールします。
xは横方向のスクロールする列数で正の数は右、負の数は左にスクロールします。
yは縦方向のスクロールする行数で正の数は下、負の数は上にスクロールします。

◎WIDTH x
30文字モードと40文字モードを切り替えます。xは30または40を指定。
初期状態は30文字モード。

[type M]
xに36、48、80も指定可能。

■グラフィック関連

MachiKania type Z、type Mでは、テキストモードのほかグラフィックモードが使えます。
プログラム実行初期はテキストモードのため、グラフィック画面を使用する場合、
グラフィックモードに変更する必要があります。以下の命令でモード変更します。

USEGRAPHIC x　（引数xでモード指定）

グラフィックモードに入るとグラフィック用にメモリを大量に確保します。
テキストモードに戻るにはUSEGRAPHIC命令の引数に0を指定します。ただし、
メモリの解放は行われません。

＜グラフィックの仕様＞
[type Z][type M]
USEGRAPHIC 1　（引数の1は省略可）
解像度　　横256×縦224ドット
同時表示色数　16色
使用メモリ　　28Kバイト

[type M]
標準グラフィックモード（横288ドット）、ワイドグラフィックモード（横384ドット）が追加されました。

これらのモードでは同時発色数が256色になり、またテキスト画面が重ね合わせて表示されるようになっています。縦のドット数が216ドットに下がっていることにご注意ください。
カラーパレットはテキストとグラフィックで共通のものとなりました。

USEGRAPHIC 5 （標準グラフィックモード）
縦横比4:3のディスプレイで1画素がほぼ正方形となるモードです。
解像度　　横288×縦216ドット
同時表示色数　256色
使用メモリ　　61Kバイト

USEGRAPHIC 9 （ワイドグラフィックモード）
縦横比16:9のディスプレイで1画素がほぼ正方形となるモードです。
解像度　　横384×縦216ドット
同時表示色数　256色
使用メモリ　　81Kバイト

色の指定はカラーパレットで行います。type Zではテキストモード用のカラーパレットとは別に持っており、type Mでは共通となっています。
カラーパレットの設定はGPALETTE命令で行います。初期時のカラーパレット設定は以下のようになります。

0：黒、1：青、2：赤、3：マゼンタ、4：緑、5：シアン、6：黄、7：白
8：黒、9～15：1～7の明るさを半分にしたもの
16～255：白[type M]

グラフィックモードでは以下の命令が使用できます。
なお、ほとんどの命令で座標とパレット番号が省略可能です。その場合、グラフィックカーソル位置およびGCOLOR命令で指定されたパレット番号が使用されます。

◎BOXFILL x1,y1,x2,y2,c
座標(x1,y1),(x2,y2)を対角線とするパレット番号cで塗られた長方形を描画します。
グラフィックカーソルは(x2,y2)に移動します。

＜使用例＞
BOXFILL 50,50,100,100,4

BOXFILL ,X,Y　　現在のグラフィックカーソル位置と座標(X,Y)を対角線とする長方形をGCOLOR命令で指定されたパレット番号で描画

◎CIRCLE x,y,r,c
座標(x,y)を中心に、半径r、パレット番号cの円を描画します。
グラフィックカーソルは(x,y)に移動します。

＜使用例＞
CIRCLE 100,100,50,5
CIRCLE ,50　　現在のグラフィックカーソル位置を中心に、GCOLOR命令で指定されたパレット番号で半径50の円を描画

◎CIRCLEFILL x,y,r,c

座標(x,y)を中心に、半径r、パレット番号cで塗られた円を描画します。
グラフィックカーソルは(x,y)に移動します。

＜使用例＞

CIRCLEFILL 100,100,50,5
CIRCLEFILL ,R　　現在のグラフィックカーソル位置を中心に、GCOLOR命令で指定された
パレット番号で塗られた半径Rの円を描画

◎GCLS
グラフィック画面を消去します。（パレット番号0で塗りつぶし）

◎GCOLOR c
各グラフィック関連命令でパレット番号を省略した場合に使用されるパレット番号を指定します。

◎GCOLOR(x,y)
グラフィック座標(x,y)の現在のパレット番号を返す関数です。

＜使用例＞
C=GCOLOR(10,20)

◎GPALETTE n,r,g,b
カラーパレットの設定を行います。
n：カラーパレット番号（type Z:0～15、type M:0～255）
r,g,bはそれぞれ赤、緑、青の強さを0～255で表します。

＜使用例＞
GPALETTE 1,255,128,0

※type Mではテキスト用カラーパレット設定命令PALETTEと同じ。

◎GPRINT x,y,c,bc,s$
座標(x,y)にパレット番号cで文字列s$を描画します。
文字は8×8ドットで座標はグラフィック座標です。
bcは文字の背景色のパレット番号を表します。背景色を使用しない場合、負数（-1など）を
指定します。
フォントは通常はシステムフォント、PCGを使用している場合はPCGで設定されたフォントで描画
します。c、bcは省略不可です。
グラフィックカーソルは(x+文字数*8,y)に移動します。

＜使用例＞
GPRINT 100,10,7,4,"ABC"　座標(100,10)にパレット番号7でABCと描画。背景色パレット番号4
GPRINT ,7,-1,"DEF"　　ABCの続きにDEFを表示。背景色なし

◎LINE x1,y1,x2,y2,c
座標(x1,y1)から(x2,y2)にパレット番号cで線分を描画します。
グラフィックカーソルは(x2,y2)に移動します。
連結した線を描画する場合に(x1,y1)を省略すると便利です。

＜使用例＞

```
GCOLOR 4
LINE 0,0,100,100,2      座標(0,0)から(100,100)にパレット番号2で線分を描画
LINE ,200,200           座標(100,100)から(200,200)にパレット番号4で線分を描画
```

◎POINT x,y

各命令で座標を省略した場合に使用される座標（グラフィックカーソル位置）を指定します。
グラフィックカーソル位置は、命令によっては実行後に自動的に更新されます。

◎PSET x,y,c

座標(x,y)の位置にパレット番号cで点を描画します。
グラフィックカーソルは(x,y)に移動します。

◎PUTBMP x,y,m,n,bmp

横m×縦nドットのビットマップ画像を座標(x,y)に描画します。x,yは画像の左上位置を指定します。
bmpはビットマップデータを格納する配列変数名または、CDATAデータ列を指定するラベル名です。
ビットマップデータは8ビットデータ列で、パレット番号を一次元的に羅列したものとなります。
ただしパレット番号0は透明色を表し、そのドットは描画されません。
グラフィックカーソルは(x,y)に移動します。

（注意）

KM-BASICの整数変数は32ビットとなるため、配列で指定する場合、配列1要素で4ドットとなり、
最下位8ビットが左端、最上位8ビットが右端になります。ビットマップデータは一次元的に
羅列されるため、横サイズが4の倍数でない場合は配列1要素の中で行の境界が入ります。

＜使用例1＞

ビットマップをラベルで指定の場合
CDATA命令で直接ビットマップを表現します。この例では5×5ドットの「×」印をパレット番号5で描画。

```
USEGRAPHIC              グラフィックモード設定
PUTBMP 0,0,5,5,BMP1     座標(0,0)に5×5ドットのビットマップ描画
WHILE INKEY()=0:WEND    何かキーが押されるまで待つ
                        （プログラム終了でテキストモードに戻り画面が消えるため）
END
LABEL BMP1              指定したラベル名以降に続くCDATAのデータ列をPUTBMP命令に渡す
CDATA 5,0,0,0,5,        CDATA命令は8ビットデータを羅列。行末にカンマを置くことで改行可能
      0,5,0,5,0,
      0,0,5,0,0,
      0,5,0,5,0,
      5,0,0,0,5
```

＜使用例2＞

ビットマップを配列で指定の場合（使用例1と同じ画像を描画）

```
USEGRAPHIC              グラフィックモード設定
DIM B(6)                5×5ドットのビットマップでは25バイト必要なため配列は7要素必要
B(0)=$00000005          (0,0)～(3,0)のデータ列。(0,0)が最下位8ビット、(3,0)が最上位8ビット
```

```
B(1)=$00050005          (4,0)～(2,1)のデータ列。(4,0)が最下位8ビット、(2,1)が最上位8ビット
B(2)=$00000005          (3,1)～(1,2)のデータ列。(3,1)が最下位8ビット、(1,2)が最上位8ビット
B(3)=$00000005          (2,2)～(0,3)のデータ列。(2,2)が最下位8ビット、(0,3)が最上位8ビット
B(4)=$00050005          (1,3)～(4,3)のデータ列。(1,3)が最下位8ビット、(4,3)が最上位8ビット
B(5)=$00000005          (0,4)～(3,4)のデータ列。(0,4)が最下位8ビット、(3,4)が最上位8ビット
B(6)=$00000005          (4,4)のデータが最下位8ビット。その他は無視
PUTBMP 0,0,5,5,B        座標(0,0)に5×5ドットのビットマップ描画
WHILE INKEY()=0:WEND    何かキーが押されるまで待つ
END
```

■その他の命令

◎CLEAR

全ての配列および文字列型変数を破棄し、配列でない整数型変数、実数型変数の値は0とします。
また、PCGの使用を停止し、フォントパターンを全て初期化します。

◎DATA xxx,yyy,zzz, ...

32ビットデータ列をメモリ上に展開します。
RESTORE命令で読み出し開始位置を指定し、READ()関数で読み出しを行います。

◎CDATA xxx,yyy,zzz, ...

8ビットデータ列をメモリ上に展開します。
RESTORE命令で読み出し開始位置を指定し、CREAD()関数で読み出しを行います。

◎RESTORE xxx

READ()関数、CREAD()関数で読み出すデータの開始位置を指定します。
xxxは行番号もしくはラベルを指定します。

◎DRAWCOUNT

DRAWCOUNT値を変更します。
DRAWCOUNT値はテレビ出力が画面最下行に到達するごと（約60分の1秒ごと）に自動的に1増える16ビット整数です。
DRAWCOUNT値の読み出しはDRAWCOUNT()関数で行います。また、WAIT命令も参照してください。

＜使用例＞

```
DRAWCOUNT 0
WHILE DRAWCOUNT()=0:WEND          テレビ出力が最下行に到達するまで待つ
```

◎EXEC x,y,z, ...

機械語プログラムを直接記述し実行します。x,y,zは機械語命令（32ビット整数値）です。

◎POKE x,y

物理アドレスxに8ビット値yを書き込みます。
ビデオメモリへの直接書き込みや、マイクロコントローラのI/Oに出力することが可能です。

◎REM xxx

コメントを記述します。xxxはコメントです。

■ボタン入力、キー入力について

MachiKaniaの基板上のボタンおよび外部接続されたPS/2キーボードのキー入力状態は、
以下の関数を使って読み出すことが出来ます。

◎KEYS(x)

本体上の指定したボタンが押されているかをチェックします。
各ボタンは押されている時に以下の数を返します。複数のボタンが押されている場合、
合計値を返します。
また、xが指定されている場合、xと各ボタンの合計値の論理積を返します。

```
KEYUP:    1
KEYDOWN:  2
KEYLEFT:  4
KEYRIGHT: 8
KEYSTART: 16
KEYFIRE:  32
```

＜使用例＞

```
WHILE KEYS(16)=0:WEND       STARTボタンが押されるまで待つ
WHILE KEYS()=0:WEND         いずれかのボタンが押されるまで待つ
IF KEYS(1) THEN X=X+1       UPボタンが押されている場合Xを1増加する
```

◎INKEY(x)

xで指定したキーが押されているかを返します。xはキーコードを表します。
押されている場合は1、押されていない場合は0を返します。
xを省略した場合、現在押されているキーのキーコードを返します。
何も押されていない場合は0を返します。

＜使用例＞

```
WHILE INKEY(13)=0:WEND     Enterキーが押されるまで待つ
WHILE INKEY()=0:WEND       何かキーが押されるまで待つ
```

キーコードは、下記サイト掲載の仮想キーコードと互換性があります。
https://msdn.microsoft.com/ja-jp/library/windows/desktop/dd375731
また、以下のプログラムでキーを押すことにより、キーコードを調べることができます。

```
WHILE 1
  PRINT INKEY()
WEND
```

右シフトキーと左シフトキーなども区別可能です。

◎INPUT$()

キーボードから文字列入力状態になり、EnterまたはEscで入力終了します。
Enterが押されると入力した文字列を返します。Escが押された場合ヌル文字列を返します。

＜使用例＞

A$=INPUT$()

■一般関数

◎CREAD()
DATA 文のデータ（8 ビット整数値）を読み出して返します。「READ()」関数も参照。

◎DRAWCOUNT()
DRAWCOUNT 値を取得します。DRAWCOUNT 値はテレビ出力が画面最下行に到達するごと（約 60 分の 1 秒ごと）に自動的に 1 増える 16 ビット整数です。DRAWCOUNT 値は DRAWCOUNT 命令で初期化します。
DRAWCOUNT 値を利用することで、実行中の画面のチラつきを防止することができます。
WAIT 命令により、DRAWCOUNT 値は意識する必要がなくなります。

◎LEN(x$)
文字列の長さを整数で返します。

◎MUSIC()
MUSIC 命令で演奏中の BGM の残りの音の数を返します。

◎NOT(x)
x が 0 の場合に 1 を、そうでない場合に 0 を返します。

◎PEEK(x)
x で示される物理アドレスから 1 バイト読み取り、返します。

◎READ()
◎READ$()
DATA 文のデータ（32 ビット整数値または文字列）を読み出して返します。
「CREAD()」関数も参照。

◎RND()
0 から 32767 までの擬似乱数を返します。

◎SGN(x)
x の符号(-1, 0, または 1)を返します。

◎TVRAM(x)
ビデオ RAM の x 番目の内容をバイト値で返します。
x を省略した場合、ビデオ RAM の開始位置の物理アドレスを返します。

◎＜文字列変数名＞$(x,y)
y は省略可能。
x の値が 0 の場合、文字列全体を返します。
x の値が正の場合、x で示される位置より右側の文字列を返します。
x の値が負のとき、文字列の右側 x 文字を返します。
y が指定された場合、y 文字分の文字列を返します。

＜使用例＞

```
A$="ABCDEFG"
PRINT A$(2,3)
PRINT A$(-3)

＜実行結果＞
CDE
EFG
```

◎STRNCMP(x$,y$,z)
2つの文字列のうちz文字分を比較し、結果を返します。同じ文字列の場合は0。

■数学関数

◎ABS(x)
整数xの絶対値を整数値で返します。

◎FABS#(x#)
実数x#の絶対値を実数値で返します。

◎ACOS#(x#)
x#の逆余弦を実数値で返します。

◎ASIN#(x#)
x#の逆正弦を実数値で返します。

◎ATAN#(x#)
x#の逆正接を実数値で返します。

◎ATAN2#(x#,y#)
y#/x#の逆正接を実数値で返します。

◎CEIL#(x#)
x#以上の最小の整数を実数値で返します。

◎COS#(x#)
x#の余弦を実数値で返します。

◎COSH#(x#)
x#の双曲線余弦を実数値で返します。

◎EXP#(x#)
eを底とするx#の指数関数値を実数値で返します。

◎FLOOR#(x#)
x#以下の最大の整数を実数値で返します。

◎FMOD#(x#,y#)
x#をy#で割った剰余を実数値で返します。

◎LOG#(x#)
　x#の自然対数を実数値で返します。

◎LOG10#(x#)
　x#の常用対数を実数値で返します。

◎MODF#(x#)
　x#の小数部を実数値で返します。

◎PI#
　円周率（3.141593）を返します。（括弧は不要）

◎POW#(x#,y#)
　x#の y#乗を実数値で返します。

◎SIN#(x#)
　x#の正弦を実数値で返します。

◎SINH#(x#)
　x#の双曲線正弦を実数値で返します。

◎SQRT#(x#)
　x#の平方根を実数値で返します。

◎TAN#(x#)
　x#の正接を実数値で返します。

◎TANH#(x#)
　x#の双曲線正接を実数値で返します。

■SYSTEM 関数、SYSTEM 命令

以下の SYSTEM 関数、SYSTEM 命令が用意されています。

SYSTEM$(0)　MachiKania バージョン文字列、"Zoea"を返す。
SYSTEM$(1)　MachiKania バージョン文字列、"1.0"等を返す。
SYSTEM$(2)　BASIC バージョン文字列、"KM-1200"等を返す。
SYSTEM$(3)　現在実行中の HEX ファイル名、"ZOEA.HEX"等を返す。
SYSTEM(20)　キャラクターディスプレイ横幅を返す。
SYSTEM(21)　キャラクターディスプレイ縦幅を返す。
SYSTEM(22)　グラフィックディスプレイ横幅を返す。
SYSTEM(23)　グラフィックディスプレイ横幅を返す。
SYSTEM(24)　キャラクターディスプレイ用の指定色を返す。
SYSTEM(25)　グラフィックディスプレイ用の指定色を返す。
SYSTEM(26)　キャラクターディスプレイの、現在の X 位置を返す。
SYSTEM(27)　キャラクターディスプレイの、現在の Y 位置を返す。
SYSTEM(28)　グラフィックディスプレイの、現在の X 位置を返す。
SYSTEM(29)　グラフィックディスプレイの、現在の Y 位置を返す。

SYSTEM(40)　PS/2 キーボードを使用中かどうかを返す。
SYSTEM(41)　PS/2 キーボード情報、vkey を返す。
SYSTEM(42)　PS/2 キーボード情報、lockkey を返す。
SYSTEM(43)　PS/2 キーボード情報、keytype を返す。
SYSTEM(100)　変数格納領域(g_var_mem)へのポインターを返す。
SYSTEM(101)　乱数シードへのポインターを返す。
SYSTEM(102)　キャラクターディスプレイ領域(TVRAM)へのポインターを返す。
SYSTEM(103)　フォント領域へのポインターを返す。
SYSTEM(104)　PCG フォント領域へのポインターを返す。
SYSTEM(105)　グラフィックディスプレイ領域へのポインターを返す。
SYSTEM 200,x　ディスプレイの表示を停止(x が 0 のとき)、もしくは開始(x が 0 以外の時)する。

SYSTEM 200,0 とすることで画面表示を停止し、実行速度を大幅にアップさせることができます。
その際、MachiKania type M 以外では DRAWCOUNT 値および WAIT 命令が機能しなくなることに注意が必要です。

■ファイル操作関連

MachiKania では、FAT または FAT32 フォーマットされた SD カードに対して、以下のファイル関連命令、関数が使用可能です。
なお、ファイル名は 8.3 形式（8 文字以下+「.」+3 文字以下の拡張子）のみ使用可能で、長いファイル名は使用できません。
アクセス可能なディレクトリは、動作中の BASIC プログラムと同じディレクトリのみで、パスを指定することはできません。

◎FOPEN x$,y$,z
◎FOPEN(x$,y$,z)
x$で示される名前のファイルを、y$で示されたモードで開きます。同時に開けるファイルの数は、2 つまでです。
y$は、次のものを指定します。

"r"　：ファイルを読み込みモードで開く
"r+"　："r"と同じだが書き込みも可能
"w"　：ファイルを書き込みモードで開く。同名のファイルがある場合は、以前のファイルは消去される
"w+"　："w"と同じだが、読み込みも可能
"a"　：ファイルを書き込みモードで開く。同名のファイルが在る場合は、ファイルは消去されず、ファイルの最後尾から書き込まれる
"a+"　："a"と同じだが、読み込みも可能

z には、割り当てたいファイルハンドル番号（1 もしくは 2）を指定します。省略した場合、1 となります。
関数として呼ばれた場合、戻り値として開いたファイルハンドル番号を返します。

＜使用例＞
FOPEN "FILENAME.TXT","w",1　　ファイル名 FILENAME.TXT を書き込みモードで開く

◎FILE x
　x にはアクティブなファイルハンドル番号（1 または 2）を指定します。
　以降使用するファイル関連命令、関数はアクティブなファイルハンドルを対象とします。

◎FCLOSE x
　開いているファイルを閉じます。x はファイルハンドル番号で、省略した場合アクティブな
　ファイルハンドルのファイルを閉じます。

◎FGET x,y
◎FGET(x,y)
　配列 x に現在のファイルの位置から y バイト読み込みます。
　関数として呼ばれた場合、読み込みに成功したバイト数を返します。
　確保が必要な配列の要素数は、読み込む最大バイト数÷4（余りは切り上げ）です。

＜使用例＞
　DIM B(63)　　32 ビット整数 64 個分の領域確保（256 バイト）
　FGET B,256

◎FGETC()
　現在のファイルの位置から 1 バイト読み込み、整数値として返します。
　ファイル末端に到達しているなどで読み込みに失敗した場合、-1 を返します。

◎FINPUT$(x)
　現在のファイルの位置から長さ x の文字列を読み込み、戻り値として返します。
　x が省略された場合は、行の最後まで読み込みます（改行コードが含まれる）。

◎FPUT x,y
◎FPUT(x,y)
　配列 x の先頭から y バイト分をファイルの現在の位置に書き込みます。
　関数として呼ばれた場合は、書き込みに成功したバイト数を返します。

◎FPUTC x
◎FPUTC(x)
　1 バイトをファイルに書き込みます。
　関数として呼ばれた場合は、書き込みに成功したバイト数（1 または 0）を返します。

◎FPRINT
　PRINT 命令と同じですが、画面ではなくファイルの現在の位置に情報が書き込まれます。

◎FSEEK x
　ファイルの現在の位置を先頭から x バイトの位置に移動します。

◎FSEEK()
　現在のファイルの先頭からの位置を返します。

◎FEOF()
　現在のファイルの位置が、ファイル末端に到達しているかどうかを返します。
　1 で末端に到達、0 で未到達を表します。

◎FLEN()

現在のファイルのファイル長を、バイト数で返します。

◎FREMOVE x$
◎FREMOVE(x$)
ファイル名 x$のファイルを SD カードから削除します。
関数として呼ばれた場合は、削除に成功したか(0)、失敗したか(-1)を返します。

＜ファイル操作プログラム例＞
（1）FILE1.TXT を FILE2.TXT にコピーする

```
DIM P(127)                      128×4＝512 バイトの領域確保
FOPEN "FILE1.TXT","r",1         コピー元ファイルを読み込みモード、ハンドル 1 番で開く
FOPEN "FILE2.TXT","w",2         コピー先ファイルを書き込みモード、ハンドル 2 番で開く
WHILE 1
  FILE 1                        ハンドル番号 1 をアクティブに設定
  D=FGET(P,512)                 コピー元から最大 512 バイト読み込み
  IF D=0 THEN BREAK             読み込みがなかった場合ループを抜ける
  FILE 2                        ハンドル番号 2 をアクティブに設定
  FPUT P,D                      コピー先に読み込んだバイト数を書き込み
WEND
FCLOSE 1
FCLOSE 2
```

（2）解像度 256x192 ドット、4 ビット色の BMP 形式画像ファイルの画像を表示する

```
USEGRAPHIC
FOPEN "BITMAP.BMP","r"          画像ファイルを読み込みモードで開く
FSEEK 54                        ファイルの現在位置をパレット情報の位置に移動
FOR C=0 TO 15
  B=FGETC():G=FGETC()           カラーパレットデータを読み込み、パレット設定する
  R=FGETC():D=FGETC()           （D はダミーデータ）
  GPALETTE C,R,G,B
NEXT
FOR Y=191 TO 0 STEP -1          BMP ファイル形式は下から上に書かれている
  FOR X=0 TO 255 STEP 2
    D=FGETC()                   画像データを 1 バイト（2 ドット分）読み込み、描画する
    PSET X,Y,(D>>4) AND 15
    PSET X+1,Y,D AND 15
  NEXT
NEXT
FCLOSE                          ファイルを閉じる
WHILE INKEY()=0:WEND            何かキーを押すまで待つ
END
```

■サウンド関連

MachiKania には音を鳴らす命令として、音階を文字列で表現する MUSIC 命令と、効果音を数値で表現する SOUND 命令があります。MUSIC 命令、SOUND 命令とも実行は止めず、BGM として音を鳴らします。
また、type M では SD カードに保存された WAV 形式の音楽再生機能があります。

◎MUSIC x$

x$で表現される BGM を演奏します。

◎MUSIC()

BGM の演奏の残り数を返す関数です。

MUSIC 命令では、BGM 用のデータを文字列で指定します。文字列の書式は、ABC 記譜法に準拠しています。ただし、すべての記法が使えるわけではありません。
なお、キーや速度などのデフォルト設定値は以下の通りです。

```
Q: 1/4=90
L: 1/8
K: C
```

ABC 記譜法については、下記等を参照してください。
https://ja.wikipedia.org/wiki/ABC%E8%A8%98%E8%AD%9C%E6%B3%95

BGM 演奏時に一度に設定できる音の数は、31 までです。これを超えて音楽を再生したい場合は、MUSIC()関数の戻り値を調べ、その値が十分小さくなってから、次の MUSIC 命令を実行するようにします。

＜使用例＞

```
REM Star Trek season 1
MUSIC "Q:1/4=90"
MUSIC "L:1/24"
MUSIC "K:C"
MUSIC "G,6F9E3D2C2B,zB,_5zB,_6B,_12"
GOSUB WAITM
MUSIC "G,6G9F3E2D2C2B,5zB,6B,9B,_3"
GOSUB WAITM
MUSIC "A,9B,3C3D3E2F2E2G12B_9A3"
GOSUB WAITM
MUSIC "G6F,6z3A,4F4A4c12"
END

LABEL WAITM
WHILE MUSIC()>1
WEND
RETURN
```

◎SOUND xxx

効果音を再生します。xxx は行番号もしくはラベルを指定します。

SOUND 命令では、DATA 列のデータを行番号もしくはラベルで指定します。

DATA 列では、32 ビット整数値として効果音を表現します。
この整数値の下位 16 ビットは周波数の指定で、2048（$0800）が 440Hz（A＝ハ長調のラの音）に対応します。値が小さくなるほどより高い音となり、半分になると 1 オクターブ上がります。
無音は 0 で表します。
上位 16 ビットは音の長さです。1 が 1/60 秒に相当します。
最後に 65535 以下の値（上位 16 ビットが 0）で、効果音の繰り返し回数を指定します。
これらのデータの数は、32 個を超えないようにして下さい。

SOUND 命令による効果音再生中は、BGM は再生されません。また、前の効果音が終わる前に次の SOUND 命令を実行すると、前の効果音の再生は停止し、新しい効果音がすぐに再生されます。

＜使用例＞

```
SOUND SOUND1
WAIT 60
SOUND SOUND2
END

LABEL SOUND1
DATA $64000,$20000,$1E4000,1          ブッブー
LABEL SOUND2
DATA $60557,$606BA,2                  ピンポンピンポン
```

[type M]

◎PLAYWAVE x$,y

WAVE 形式のファイルを再生します。x$はファイル名で、SD カードの実行プログラムと同じディレクトリ（未保存の場合カレントディレクトリ）に保存されている必要があります。
y を指定した場合、サンプリングの指定位置から再生開始します。
（1 秒目から再生する場合 15700）

演奏可能な WAVE ファイルは以下のものに限ります。
サンプリング周波数　15.7KHz
ビット長　8 ビット
ステレオまたはモノラル

※サンプリング周波数 16KHz のファイルも再生可能ですが、音程がずれます。

＜使用例＞

```
PLAYWAVE "MUSIC1.WAV"
```

◎PLAYWAVE(x)

関数として使用した場合、再生中のファイルのサンプリング数に関する数値を返します。
x を 0 または省略とした場合、現在再生中ファイルの残りサンプリング数、1 を指定した場合、現在のサンプリング番号、2 を指定した場合は総サンプリング数を返します。

■PCGについて

MachiKaniaではテキスト表示のフォントパターンを変えられるPCG（Programable Character Generator）機能が使用可能です。
PCG機能を使用するために、以下の命令があります。

◎USEPCG x

PCGを使用開始または使用停止します。

xの値の意味
- 0：PCG使用停止
- 1：PCG使用開始（最初に使用する場合、フォントパターンをシステムフォントで初期化）
- 2：フォントパターンをシステムフォントに初期化して使用開始

xを省略した場合、x=1を指定した場合と同じになります。

◎PCG x,y,z

キャラクター番号xのフォントパターンを定義します。xは0～255を指定可能です。
y、zは各32ビットのフォントパターンを表します。
フォントサイズは30文字モードでは横8ドット×縦8ドット、40文字モードでは横6ドット×縦8ドットとなります。

＜使用例＞
30文字モードで以下のフォントパターンをキャラクターコード$80に定義する場合、次のように記述します。

```
PCG $80,$80402010,$08040201
```

●○○○○○○○　10000000b＝$80
○●○○○○○○　01000000b＝$40
○○●○○○○○　00100000b＝$20
○○○●○○○○　00010000b＝$10
○○○○●○○○　00001000b＝$08
○○○○○●○○　00000100b＝$04
○○○○○○●○　00000010b＝$02
○○○○○○○●　00000001b＝$01

フォントパターンの1ワード目が上4行分、2ワード目が下4行分となり、1行8ビットで上位ビットが左側になります。

40文字モードの場合、横6ドットのパターンのため、上位6ビットで指定し、下位2ビットは無効となります。

■I/O関連について[type M]

MachiKania type Mは多数のI/Oポートを備えており、BASICにて各種入出力に対応しています。
汎用デジタル入出力、アナログ入力、PWM出力、シリアル通信、SPI通信（マスター機能）、I2C通信（マスター機能）をサポートしています。

MachiKania BASIC Systemのリセット時の各ピンの状態は以下のようになります。

B0～B15　デジタル入力、内部でＨにプルアップ（汎用入出力）
C12,C15　使用不可（オシレータ接続）
C13　デジタル出力、初期値不定（シリアル通信のTXピン）
C14　デジタル入力、内部でＨにプルアップ（シリアル通信のRXピン）
D0～D5　デジタル入力、内部でＨにプルアップ（操作用ボタンに接続）
D6～D8　デジタル出力、初期値不定（予約）
D9　デジタル出力、初期値不定（SPI通信の標準CSピン）
D10,D11　デジタル出力、初期値不定（PWM出力）
E0～E4　デジタル出力（ビデオ出力専用）
E5～E7　デジタル入力、内部でＨにプルアップ（汎用入出力、出力時はオープンドレイン）
F0,F1　デジタル入出力（PS/2専用、外部プルアップ、出力時はオープンドレイン）
F2　デジタル入力、内部でＨにプルアップ（SPIのMISOピン）
F3　デジタル出力（SDカードのCS専用）
F4,F5　デジタル出力（オーディオ出力専用）
F6　デジタル出力、初期値不定（SPI通信のSCKピン）
G2　デジタル出力、初期値Ｈ（I2C通信のSCLピン）
G3　デジタル出力、初期値Ｈ（I2C通信のSDAピン）
G6,G8　デジタル出力（SDカード専用）
G7　デジタル入力（SDカード専用）
G9　デジタル出力、初期値Ｌ（SPI通信のMOSI）

【汎用デジタル入出力関連】
外部接続ピンのB0～B15、E5～E7の入出力に対応しています。
E5～E7を出力として使用する場合、オープンドレイン出力となります。
以下の出力命令や入力関数を実行する際、入出力方向の設定は自動的に行われるため、
事前の設定は不要です。
入力関数を利用する場合は、マイコン内部でＨにプルアップされます。

◎OUT x,y
特定の１つのピンをＨまたはＬに設定します。
xが0～15はB0～B15、xが16～18はE5～E7を表します。
yは０または１で、０がＬ、１がＨを表します。

◎OUT8L x
B0～B7に8ビット値xを出力します。

◎OUT8H x
B8～B15に8ビット値xを出力します。

◎OUT16 x
B0～B15に16ビット値xを出力します。

◎IN(x)
特定の１つのピンの入力値を返します。
xが0～15はB0～B15、xが16～18はE5～E7を表します。
ピンがＨの場合1、Ｌの場合0を返します。

◎IN8L()

B0～B7の8ビット値を返します。

◎IN8H()
B8～B15の8ビット値を返します。

◎IN16()
B0～B15の16ビット値を返します。

＜使用例＞
B0に接続されたスイッチを読み、B1に接続されたLEDを点灯または消灯させる。

```
WHILE 1
 A=IN(0)
 OUT 1,A
WEND
```

【アナログ入力関連】
外部接続ピンのB0～B15、E5～E7からアナログ入力に対応しています。
以下の関数を使用する場合、自動的にアナログ入力に設定されます。

◎ANALOG(x)
xが0～15はB0～B15、xが16～18はE5～E7を表します。
戻り値は0～1023で0V～3.3V（VDD）を表します。

＜使用例＞
(1) E5ピンに接続されたアナログ信号を1秒ごとに読み込み、グラフ表示する。

```
USEGRAPHIC 9
FOR I=0 TO 383
 A=ANALOG(16)
 PSET I,A*216/1023,2
 WAIT 60
NEXT
```

(2) B9ピンに接続された温度センサーLM61BIZで温度を表示する。

```
PRINT ANALOG(9)*322/1000-60;"[Deg]"        温度計算式(v*1000*3.3/1024-600)/10
```

【PWM出力関連】
D10およびD11ピンにパルス幅を指定したPWM信号を出力します。

◎PWM x,y,z
x：パルスのデューティー比を0～1000で表し、0が常時L、1000が常時H。
y：パルスの周波数。単位はHz。省略時は1000。有効値は6～95454。
z：1でD10ピン、2でD11ピンに出力。省略時は1。

＜使用例＞
(1) D10ピンに接続されたLEDをぼんやり点滅させる。

```
WHILE 1
 FOR I=0 TO 99
  PWM INT(SIN#(FLOAT#(I)/50*PI#)*500+500),5000
  WAIT 1
 NEXT
WEND
```

(2) D10、D11 ピンを H ブリッジの 2 つの入力に接続したモーターを UP ボタン、DOWN ボタンで速度制御する。減速して停止後は反転する。

```
D=0:PWM 0,100,1:PWM 0,100,2
WHILE 1
 K=KEYS()
 IF K=2 THEN
  D=D+10:IF D>1000 THEN D=1000
 ELSEIF K=1 THEN
  D=D-10:IF D<-1000 THEN D=-1000
 ENDIF
 IF D>=0 THEN
  PWM 0,100,2:PWM D,100,1
 ELSE
  PWM 0,100,1:PWM -D,100,2
 ENDIF
 WAIT 3
WEND
```

【シリアル通信関連】

以下の命令および関数により C13、C14 ピンでシリアル通信（UART）が可能です。
C13 が送信（TX）、C14 が受信（RX）となっています。

◎SERIAL x,y,z

シリアル通信の設定を行い利用開始します。シリアル通信使用時、必ず設定を行う必要があります。
x：通信速度（ボーレート）を指定。0 でシリアル通信を終了。
y：0 パリティなし、1 偶数パリティ、2 奇数パリティ、3 9 ビットパリティなし
z：受信バッファとして確保する文字数。省略時は 1/60 秒受信分のバッファを確保。

◎SERIALIN()

受信バッファ先頭の 1 文字を返します。受信バッファが空の場合-1 を返します。
パリティエラーが発生した場合、256 以上の数字を返します。

◎SERIALIN(1)

受信バッファの現在の受信数を返します。

◎SERIALOUT x

x を送信します。

＜使用例＞

(1) 115.2Kbps で C14 ピン(RX)からシリアルデータを読み込み、SD カードに「TEST.TXT」というファイル名で保存する。ESC キーで終了する。

```
SERIAL 115200,0
FOPEN "TEST.TXT","W"
WHILE 1
 A=SERIALIN()
 IF A>=0 THEN FPUTC A
 IF INKEY()=27 THEN BREAK
WEND
FCLOSE
```

(2) SD カードの「TEST.TXT」というファイルの内容を C15 ピン(TX)から 9600bps でシリアル送信する。

```
DIM B(0)
SERIAL 9600,0
FOPEN "TEST.TXT","R"
WHILE FGET(B,1)>0
 SERIALOUT B(0)
WEND
FCLOSE
```

【SPI 通信関連】

SPI 通信のマスター機能に対応しています。8 ビット、16 ビットまたは 32 ビットを 1 ワードとするシリアル通信を上位ビットから行います。
通信は出力、入力、双方向が可能です。
以下の各通信命令、関数の開始時に CS ピンを L にし、終了時に CS ピンを H にします。

◎SPI s,i,m,p

SPI マスター機能を初期化します。SPI 通信使用時、必ず設定を行う必要があります。
s：通信速度（KHz 単位）、有効値は 93～47727。
i：1 ワードのビット数を 8、16、32 で指定。省略時は 8。
m：SPI モード 0～3 を指定、省略時は 0。
　0 アイドル時クロックは L、立ち下がり時にデータ変更、立ち上がりで読み込み
　1 アイドル時クロックは L、立ち上がり時にデータ変更、立ち下がりで読み込み
　2 アイドル時クロックは H、立ち上がり時にデータ変更、立ち下がりで読み込み
　3 アイドル時クロックは H、立ち下がり時にデータ変更、立ち上がりで読み込み
p：CS ピンを指定（省略時 D9 ピン）。
　上位 4 ビットがポートグループ名、下位 4 がポート番号を表す。
　例えば B5 ピンは$15、D6 ピンは$36。

◎SPIWRITE d1,d2,d3...

固定ワード数のデータを書き込みます。データは 1 個以上の任意の数を指定可能です。

◎SPIREAD(d1,d2,d3...)

1 ワードのデータを読み込んで返す関数です。
引数を指定した場合、指定データ全てを書き込み後に読み込みを行います。

◎SPIWRITEDATA b,n,d1,d2,d3...
バッファアドレスｂからｎワード分の書き込みを行います。
d1,d2,d3...を指定した場合、先に指定したデータの書き込みを行います。

◎SPIREADDATA b,n,d1,d2,d3...
バッファアドレスｂにｎワード分の読み込みを行います。
d1,d2,d3...を指定した場合、先に指定したデータの書き込みを行います。

◎SPISWAPDATA b,n,d1,d2,d3...
バッファアドレスｂからｎワード分のデータ交換を行います（送受信同時）。
d1,d2,d3...を指定した場合、先に指定したデータの書き込みを行います。

＜使用例＞
10bit DAコンバーターMCP4911を使用して、約1Hzの擬似ノコギリ波を出力する。
（B1ピンにLDACを接続。）

```
SPI 10000,16,0:REM 10MHz/16bit/Mode0/CS:D9pin
WHILE 1
 FOR I=0 TO 59
  OUT 1,1:REM LDAC=H
  SPIWRITE $7000+(I*17<<2)
  OUT 1,0:REM LDAC=L
  WAIT 1
 NEXT
WEND
```

【I2C通信関連】
I2Cのマスター機能に対応しています。

◎I2C s
I2Cマスター機能を初期化します。I2C通信使用時、必ず設定を行う必要があります。
s：通信速度（KHz単位）。有効値は12～3409。省略時は100KHz。

◎I2CWRITE a,d1,d2,d3...
スレーブアドレスａの装置に、固定バイト数のデータを書き込みます。データは0個以上の
任意の数を指定可能です。

◎I2CREAD (a,d1,d2,d3...)
スレーブアドレスａの装置から1バイトのデータを読み込んで返す関数です。
d1,d2,d3...を指定した場合、指定データ全てを書き込み後に読み込みを行います。

◎I2CWRITEDATA a,b,n,d1,d2,d3...
スレーブアドレスａの装置に、バッファアドレスｂからｎバイト分の書き込みを行います。
d1,d2,d3...を指定した場合、先に指定したデータの書き込みを行います。

◎I2CREADDATA a,b,n,d1,d2,d3...
スレーブアドレスａの装置からバッファアドレスｂにｎバイト分の読み込みを行います。
d1,d2,d3...を指定した場合、先に指定したデータの書き込みを行います。

◎I2CERROR()

直前に実行したI2C送受信で正常の場合0、エラー（NACK）の場合0以外を返す関数です。

＜使用例＞

スレーブアドレス$50のEEPROM（24LC256等）の先頭アドレスから文字列を書き込み後、
読み出して画面に表示する。

```
I2C 100
DIM D(4)                        20バイトの領域確保
T$="Hello MachiKania!"          18バイト（17文字＋$00）の文字列
I2CWRITEDATA $50,T,18,0,0       スレーブアドレス$50に0,0を出力後、Tから18バイト出力
GOSUB WAITEN
I2CREADDATA $50,D,18,0,0        スレーブアドレス$50に0,0を出力後、Dに18バイト読み込む
PRINT D$
END

LABEL WAITEN                    書き込み終了待ち
 DO
  I2CWRITE $50
 LOOP WHILE I2CERROR()          ACKが返るまで繰り返す
RETURN
```
