

## ■■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、

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。

MachiKania type P ではファイル選択画面で実行するファイルを選択し、[FIRE] ボタンを押して実行します。[START] ボタンを押すとファイルの中身が表示されます。

---

## ■ 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にも対応しています。

<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。(type P ではさらに長い変数名が使用可能)

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 変数 x の格納アドレス (ポインタ) [KM-1302 以降、および KM-1207 以降で使用可能]

-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

$x \neq y$  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

$x \leq y$   $x$ 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 $y$ より大きい場合に1、そうでないときに0

$x \geq y$   $x$ が $y$ より大きいか等しい場合に1、そうでないときに0

$x \text{ AND } y$   $x$ と $y$ の値のビットごとの AND (論理積でないことに注意)

$x \text{ OR } y$   $x$ と $y$ の値のビットごとの OR

$x \text{ XOR } y$   $x$ 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

$x\# \neq y\#$  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より大きい場合に1、そうでないときに0  
x# >= y# xがyより大きい等しい場合に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。(type M/type ZではCPU停止、type Pでは空ループ)  
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)

※MachiKania type ZのKM-1207以前では、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 Pではテレビへの信号出力はありませんが、同等の動作するようシステムタイマーが60分の1秒刻みとなるまでプログラム停止します。

◎IDLE [KM-1500以降(type P)、KM-1303以降(type M)、KM-1208以降(type Z)で使用可能]

CPU をアイドル状態にして一時停止します。システム内部でのビデオ出力やタイマーなどの割り込み発生によりアイドル状態から抜けます。

◎DELAYMS x [KM-1500 以降(type P)]

x ミリ秒間プログラムを停止します。

◎DELAYUS x [KM-1500 以降(type P)]

x マイクロ秒間プログラムを停止します。

◎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

IF I=7 THEN BREAK

PRINT I;

NEXT

I が 4 の場合、NEXT 命令にジャンプしループを続ける

I が 7 の場合、NEXT 命令の次にジャンプしループを抜ける

<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 ARG\$ () 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。

- ◎ARG\$ (x)
- ◎ARG\$# (x)
- ◎ARG\$ (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=ARG$ (1)                1 番目のパラメータを取得 (10)
Y=ARG$ (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となります。

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になっています。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。

[type P]

液晶横置きの場合、横 40 文字、縦 30 文字表示可能です。縦置きの場合、横 30 文字、縦 40 文字表示可能です。実際にはグラフィック画面に描画しています。横置き、縦置きの指定は WIDTH 命令の第 2 引数で行います。

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。

[type P]

グラフィック画面も同時にクリアします。

◎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] [type P]

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。

[type P]

x は 1～最大 40 まで指定可能。液晶横幅より小さい値を指定した場合、画面右側にはテキスト表示されません。

また、第 2 引数に 1 を指定すると画面が縦長方向、2 を指定すると横長方向となります。

---

## ■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。

MachiKania type P では標準でグラフィック対応しており、テキストもグラフィック画面に描画されます。USEGRAPHIC 命令は無視されます。

<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

[type P]

解像度 320 × 240 ドット（縦横の向きは WIDTH 命令で指定）

同時表示色数 256 色

使用メモリ なし

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、type P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>

G=GCOLOR(10, 20)

※type P では同じ色のパレットが複数ある場合、最も小さいパレット番号を返します。

◎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、type P 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。（type P ではテレビ出力がないため、擬似的に生成）

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

◎POKE16 x, y [KM-1500以降(type P)、KM-1302以降(type M)、KM-1207以降(type Z)で使用可能]

◎POKE32 x, y [KM-1500以降(type P)、KM-1302以降(type M)、KM-1207以降(type Z)で使用可能]

物理アドレスxにそれぞれ8ビット値または16ビット値、32ビット値yを書き込みます。

xはPOKE16の場合、16ビット境界、POKE32の場合32ビット境界である必要があります。

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できます。

type Pにはキーボード接続がありませんので、キーボードに関する関数は使用できません。

### ◎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$=INPUT$( )
```

### ◎READKEY( ) [KM-1303 以降 (type M)、KM-1208 以降 (type Z) で使用可能]

キーボードバッファから一文字読み込み返します。バッファが空の時は 0 を返します。  
戻り値は以下の形式の 24 ビット整数です。

```
bits 0-7   : ASCII コード (文字キー以外では 0)
bits 8-15  : 仮想キーコード (INKEY() 関数の説明参照)
```

bits 16-23 : シフトキー押下状態。押下状態で1  
上位から<0><CAPSLK><NUMLK><SCRLK><Win><ALT><CTRL><SHIFT>

---

## ■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。

※MachiKania type Pではテレビ出力がないため、単に60分の1秒ごとに増えるカウンタとして利用可能です。

### ◎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)

◎PEEK16(x) [KM-1500以降(type P)、KM-1302以降(type M)、KM-1207以降(type Z)で使用可能]

◎PEEK32(x) [KM-1500以降(type P)、KM-1302以降(type M)、KM-1207以降(type Z)で使用可能]

xで示される物理アドレスからそれぞれ8ビット値、16ビット値、32ビット値を読み取り、返します。xはPEEK16の場合16ビット境界、PEEK32の場合32ビット境界である必要があります。

### ◎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$="ABCDEFGH"
```

```
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”等を返す。
- 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(4) 現在実行中の CPU のクロック周波数を返す。
- 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) へのポインタを返す。  
SYSTEM (101) 乱数シードへのポインタを返す。  
SYSTEM (102) キャラクターディスプレイ領域 (TVRAM) へのポインタを返す。  
SYSTEM (103) フォント領域へのポインタを返す。  
SYSTEM (104) PCG フォント領域へのポインタを返す。  
SYSTEM (105) グラフィックディスプレイ領域へのポインタを返す。type P では無効。

[type M] [type Z]

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。 ※MachiKania type Z でも KM-1208 以降は機能するよう、仕様変更しました。

---

## ■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、成功すれば 0 を、失敗すれば -1 を返します。

◎SETDIR x\$ [KM-1500 以降 (type P)、KM-1302 以降 (type M)、KM-1207 以降 (type Z) で使用可能]  
カレントディレクトリを x\$ に移動します。

関数として呼ばれた場合、成功すれば 0 を、エラーがあれば 0 以外を返します。

◎GETDIR\$ () [KM-1500 以降 (type P)、KM-1302 以降 (type M)、KM-1207 以降 (type Z) で使用可能]  
カレントディレクトリを文字列として返します。

<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 P、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      ブブー
LABEL SOUND2
DATA $60557, $606BA, 2              ピンポンピンポン
```

[type P][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せん。

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 (type Pはモノラルのみ)

※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>

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。

type M、type Zの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 P] [type M]

MachiKania type Mおよびtype P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。

[type M]

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。

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。

[type P]

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。  
また、ピンごとにKM-BASICで割り当てられる機能を記載します。

|      |                             |                                   |
|------|-----------------------------|-----------------------------------|
| GP0  | デジタル入力 (内部プルアップ)            | I/O bit0 / PWM3                   |
| GP1  | デジタル入力 (内部プルアップ)            | I/O bit1 / PWM2                   |
| GP2  | デジタル入力 (内部プルアップ)            | I/O bit2 / PWM1                   |
| GP3  | デジタル入力 (内部プルアップ)            | I/O bit3 / SPI CS                 |
| GP4  | デジタル入力 (内部プルアップ)            | I/O bit4 / UART TX                |
| GP5  | デジタル入力 (内部プルアップ)            | I/O bit5 / UART RX                |
| GP6  | デジタル入力 (内部プルアップ)            | I/O bit6 / I2C SDA                |
| GP7  | デジタル入力 (内部プルアップ)            | I/O bit7 / I2C SCL                |
| GP8  | デジタル入力 (内部プルアップ)            | I/O bit8 / button1 (UP)           |
| GP9  | デジタル入力 (内部プルアップ)            | I/O bit9 / button2 (LEFT)         |
| GP10 | デジタル出力                      | LCD-DC                            |
| GP11 | デジタル出力                      | LCD-RESET                         |
| GP12 | デジタル入力                      | LCD-MISO                          |
| GP13 | デジタル出力                      | LCD-CS                            |
| GP14 | デジタル出力                      | LCD-SCK                           |
| GP15 | デジタル出力                      | LCD-MOSI                          |
| GP16 | デジタル入力 (10K $\Omega$ プルアップ) | SD-D0 (MISO) / SPI RX             |
| GP17 | デジタル出力                      | SD-CS                             |
| GP18 | デジタル出力                      | SD-SCLK / SCK                     |
| GP19 | デジタル出力                      | SD-DI (MOSI) / SPI TX             |
| GP20 | デジタル入力 (内部プルアップ)            | I/O bit10 / button3 (RIGHT)       |
| GP21 | デジタル入力 (内部プルアップ)            | I/O bit11 / button4 (DOWN)        |
| GP22 | デジタル入力 (内部プルアップ)            | I/O bit12 / button5 (START)       |
| GP26 | デジタル入力 (内部プルアップ)            | I/O bit13 / button6 (FIRE) / ADC0 |
| GP27 | デジタル入力 (内部プルアップ)            | I/O bit14 / ADC1                  |
| GP28 | デジタル入力 (内部プルアップ)            | I/O bit15 / SOUND OUT / ADC2      |
| GP29 | デジタル入力 (内部プルアップ)            | ADC3                              |

【汎用デジタル入出力関連】

I/O bit 番号はKM-BASICで入出力指定する際のポート番号です。bit10以降はGP番号と異なりますのでご注意ください。

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。

また、bit8~bit13の6ポートは操作ボタンと共有、bit15は音声出力と共有しているため、組み込み用途などでボタンや音声出力をなくしたシステムの場合を除き、汎用的に使用可能なポートは9個となります。

以下にI/O関連命令および関数について記載します。B0~B15はtype PではI/O bit0~I/O bit15を表します。E5~E7はtype M専用です。

◎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】

type Mとtype Pで大きく異なります。

[type M]

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
```

[type P]

GP26～29 のアナログ入力に対応しています。ただし GP29 は外部接続されていません。  
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=0, 13, 26 で GP26、x=1, 14, 27 で GP27、x=2, 15, 28 で GP28、x=3, 16, 29 で GP29 が、  
指定される。

戻り値は 0～4095 で 0V～3.3V (VDD) を表します。

<使用例>

GP27 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(1)*322/4000-60;"[Deg]"      温度計算式 (v*1000*3.3/4096-600)/10
```

【PWM 出力関連】

[type M]

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。

[type P]

I/O bit0～I/O bit2 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

[type M] 有効値は 6～95454

[type P] 有効値は 1～125000

z : [type M] 1 で D10 ピン、2 で D11 ピンに出力。省略時は 1

[type P] 1 で I/O bit2、2 で I/O bit1、3 で I/O bit0 ピンに出力。省略時は 1

<使用例>

(1) D10 ピン (type P では I/O bit2) 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 (type P では I/O bit2、I/O bit1) に Hブリッジの 2 つの入力を接続した  
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シリアル通信 (UART) が可能です。

[type M] C13 が送信 (TX)、C14 が受信 (RX) となっています。

[type P] I/O bit4 が送信 (TX)、I/O bit5 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 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)

[type M]有効値は 93~47727

[type P]有効値は 2~31250

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

[type M]

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。省略時 D9 ピン。

[type P]

I/O bit 番号で指定。省略時は I/O bit3 ピン。

#### ◎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)。省略時は 100KHz。

[type M] 有効値は 12~3409

[type P] 有効値は 2~6250

◎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。

[type M] データは 0 個以上の任意の数を指定可能です。

[type P] データは 1 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。

◎I2CWRTEDATA 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
I2CWRTEDATA $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
```

※MachiKania type PではRaspberry Pi Picoのハードウェア仕様上、I2CWRITE 命令にはアドレスのほか、1つ以上のデータを出力するする必要があります。そのためWAITEN サブルーチン中の「I2CWRITE \$50」の代わりに「I2CWRITE \$50, 0」のようにダミーバイト出力を指定することになります。この時の動作は、アドレス\$50を出力しNACKが返った場合はダミーバイト0は出力されず、ACKとなったときにダミーバイト0が出力されます。ダミーバイトが出力されることが問題となる場合には、次回出力するデータをダミーバイト代わりにするなどの工夫が必要です。

---

## ■クラスライブラリについて

KM-1500以降[type P]、KM-1302以降[type M]、およびKM-1207以降[type Z]では、簡易的なオブジェクト指向化プログラミングに対応したクラスライブラリの利用をサポートしています。クラスライブラリでは、まとまった機能をBASICで書かれたクラスファイルとして用意し、実行時にメインプログラムから読み込んで実行することができます。

クラスファイルのファイル名は「クラス名.BAS」で保存します。クラス名は2文字以上6文字以下（type Pでは8文字以下）の文字列です。

（例） クラス名：CLASS1 → ファイル名：CLASS1.BAS

クラスファイルはメインプログラムと同じディレクトリ（カレントディレクトリ）または、SDカードの「¥LIB¥クラス名¥」ディレクトリに保存します。

（例） ¥LIB¥CLASS1¥CLASS1.BAS

実行時（コンパイル時）にクラス使用命令（USECLASS 命令）があると、まずカレントディレクトリを探し、なければ¥LIBを探しにいきます。よく使われる便利な機能は、共通ライブラリとして¥LIBディレクトリで提供することができます。

### 【クラスファイルの書き方】

クラスファイルでは、メソッド（機能として公開するサブルーチン）とフィールド（変数）を定義することができます。

フィールドは、公開／非公開／スタティックを設定することができます。クラスファイル中では以下の宣言文でメソッドとフィールドを定義します。

#### ◎FIELD PUBLIC f1, f2, f3...

公開されたフィールド変数を宣言します。「PUBLIC」は省略できます。

f1, f2, f3等はフィールド名で、2文字以上、6文字以内の英数字で指定します。

#### ◎FIELD PRIVATE f1, f2, f3...

非公開のフィールド変数を宣言します。

f1, f2, f3等はフィールド名で、2文字以上、6文字以内の英数字で指定します。

#### ◎STATIC PUBLIC f1, f2, f3...

公開されたスタティック変数を宣言します。「PUBLIC」は省略できます。

f1, f2, f3等はフィールド名で、2文字以上、6文字以内の英数字で指定します。

#### ◎STATIC PRIVATE f1, f2, f3...

非公開のスタティック変数を宣言します。USEVARと同じです。

f1, f2, f3等はフィールド名で、2文字以上、6文字以内の英数字で指定します。

#### ◎METHOD x

メソッドを宣言します。xはメソッド名で、2文字以上、6文字以内の英数字で指定します。

次の文からメソッドの内容を記述します。最後にRETURN文で呼び出し元に戻ります。

メソッド名をINITと指定すると、NEW文で新たにオブジェクトが作成されたときに、必ずこのINITメソッドを最初に呼び出します。（コンストラクタ）

#### [type P]

type Pでは上記変数やメソッド名は7文字以上も指定可能です。

### 【クラスの使用方法】

メインプログラム側では、クラスの利用宣言の後、オブジェクトを生成します。生成したオブジェクトに対してメソッド呼び出しや、フィールド値の設定などを行い、ライブラリとしてのクラス機能を使用します。

クラスの使用に関しては以下の命令があります。

#### ◎USECLASS c1, c2, c3...

クラスの利用を宣言します。c1, c2, c3等はクラス名。

#### ◎NEW(c, x, y...)

クラスオブジェクトを作成し、オブジェクトへのポインタを返します。

cはクラス名

x, y等はINITメソッド（コンストラクタ）がある際に渡される引数です。

#### <使用例>

```
P=NEW(CLASS1, 3, 0)
```

#### ◎DELETE x, y, z...

作成されたオブジェクトを破棄します。x, y, z等はオブジェクトを格納する変数名を指定します。

◎CALL x.m(a, b, c...)

xで指定されたオブジェクトのメソッドmを呼び出します。a, b, c等はメソッドに渡す引数です。戻り値を利用する場合、「CALL」は省略できます。

<使用例>

オブジェクトPのメソッドTEST1、TEST2を呼び出す。TEST2には引数Xを渡します。

```
CALL P.TEST1()  
A=P.TEST2(X)
```

◎x.f (フィールド変数へのアクセス)

xで指定されたオブジェクトの公開されたフィールド変数fを表します。

<使用例>

```
P.X1=100  
PRINT P.X1
```

◎c::s (スタティック変数へのアクセス)

クラスcの公開されたスタティック変数sを表します。

<使用例>

```
CLASS1::S1=10  
PRINT CLASS1::S1
```

<クラス利用の実例>

「移動する円」というオブジェクトを作成します。フィールド変数として、座標と移動方向（右または左）を持ちます。メソッドとしては、初期座標設定、表示、消去、移動の4つを作ります。

【クラスファイル】ファイル名 CLASS1.BAS でメインプログラムと同じディレクトリに保存する

FIELD PUBLIC Y1                    公開フィールド変数を定義。メインプログラムでも参照可能  
FIELD PRIVATE X1, V1                非公開フィールド変数を定義。クラスファイル内でのみ参照可能

METHOD INIT                        クラス生成時に呼び出されるメソッド  
  X1=ARGS(1):Y1=ARGS(2):V1=1        フィールド変数を初期設定  
RETURN

METHOD DISP  
  CIRCLEFILL X1, Y1, 2, ARGS(1)     半径2の塗りつぶし円を表示。色は引数で与える  
RETURN

METHOD ERASE  
  CIRCLEFILL X1, Y1, 2, 0            円を消去（黒で塗りつぶし）  
RETURN

METHOD MOVE  
  X1=X1+V1                            横方向に移動  
  IF X1<2 THEN X1=2:V1=1            左端で折り返し  
  IF X1>253 THEN X1=253:V1=-1      右端で折り返し  
RETURN

## 【メインプログラム】 ファイル名は任意

|                         |                                     |
|-------------------------|-------------------------------------|
| USECLASS CLASS1         | クラス利用宣言                             |
| USEGRAPHIC              |                                     |
| P=NEW(CLASS1, 120, 100) | オブジェクトの生成。120, 100 は INIT メソッドに渡す引数 |
| WHILE 1                 |                                     |
| CALL P.DISP(2)          | DISP メソッド呼び出し。引数 2 (赤色) を指定         |
| WAIT 1                  | 60 分の 1 秒待ち                         |
| CALL P.ERASE()          | ERASE メソッド呼び出し                      |
| CALL P.MOVE()           | MOVE メソッド呼び出し                       |
| K=INKEY()               |                                     |
| IF K=\$26 THEN          | 上矢印キーが押された場合                        |
| P.Y1=P.Y1-1             | 公開フィールド変数 Y1 を 1 引く                 |
| ELSEIF K=\$28 THEN      | 下矢印キーが押された場合                        |
| P.Y1=P.Y1+1             | 公開フィールド変数 Y1 を 1 足す                 |
| ENDIF                   |                                     |
| WEND                    |                                     |

このメインプログラムでは、オブジェクトは1つしか生成していませんが、複数生成することも可能です。例えば Q=NEW(CLASS1, 30, 50) として、P と Q の 2 つの移動する円を表示させることができます。つまりフィールド変数は生成したオブジェクトごとに保有されます。オブジェクトに共通の変数を持ちたい場合は、スタティック変数として宣言します。

---

## ■タイマー機能と割り込み

[KM-1500 以降 (type P)、KM-1303 以降 (type M)、KM-1208 以降 (type Z) で使用可能]

### 【タイマー機能】

一定時間のウェイトを作成する方法として、ビデオ出力の垂直同期と連携した 60 分の 1 秒単位の WAIT 命令、60 分の 1 秒ごとにカウントされる DRAWCOUNT () 関数のほか、2 種類のタイマーが追加されました。

#### ◎USETIMER x

x で指定したタイミングごとにカウントアップするタイマーを開始します。x の単位はマイクロ秒 (175769 以下の値)。タイマー値は 0 に初期化されます。

#### ◎TIMER x

タイマー値を 32 ビット整数 x で設定します。

#### ◎TIMER()

現在のタイマー値を、32 ビット整数値で返します。

#### <使用例 1>

指定引数のマイクロ秒スリープさせる

```
GOSUB SLEEP, 1000      1 ミリ秒 (1000 マイクロ秒) ウェイト
```

```
.....
```

```
LABEL SLEEP
```

```
  USETIMER ARGS(1)    GOSUB の引数でタイマーのカウントアップタイミングを指定
```

```

WHILE TIMER()=0      タイマー値が0の間ループする
  IDLE              CPUをアイドル状態に設定
WEND                繰り返し
RETURN

```

#### <使用例 2>

```

USETIMER 1000      タイマーを1ミリ秒（1000マイクロ秒）にセット
GOSUB SLEEP2, 3000  3000ミリ秒（3秒）ウェイト
. . . . .
LABEL SLEEP2
V=ARGS(1)
TIMER 0           タイマー値初期化
WHILE TIMER()<V  タイマー値が引き数より小さい間ループする
  IDLE           CPUをアイドル状態に設定
WEND            繰り返し
RETURN

```

※IDLE 命令で CPU はアイドル状態になりますが、各種割り込み発生でウェイクアップします。MachiKania システムでは常時各種の割り込みが発生していますが、このように IDLE 命令をループさせることで消費電力を抑えることができます。ただし、より正確なタイミングで実行したい場合は IDLE 命令を使用せず CPU を 100% で動作させてください。

#### ◎CORETIMER

[type M][type Z]

コアタイマーは CPU のシステムクロック 2 クロックごとにカウントアップされる 32 ビットのカウンタです。CPU がアイドル状態の場合でもカウントアップします。

[type P]

1 マイクロ秒ごとにカウントアップされる 32 ビットのカウンタです。

本命令はコアタイマー割り込みの発生タイミングを設定します。コアタイマー値自体は変更せず、CORETIMER() 関数と合わせて使用します。

#### ◎CORETIMER()

現在のコアタイマー値を、32 ビット整数値で返します。

#### 【割り込み機能】

BASIC プログラム実行中、各種割り込み発生により指定のサブルーチンを呼び出すことができます。

#### <割り込みプログラム作成時の注意事項>

- ・割り込みサブルーチンはなるべく短時間で処理終了することが望ましい。
- ・他の部分で VAR 指定で使用しているローカル変数名と同じ名前のグローバル変数を、割り込みサブルーチン内で使用しない。（ローカル変数を破壊する可能性があるため。）

#### ◎INTERRUPT xxx, yyy, z1, z2 . . .

割り込みの設定を行います。本命令実行以降、BASIC プログラム上で割り込みが有効となります。

xxx : 割り込みの種類名として以下を指定します。

TIMER        タイマー割り込み。タイマー値が 1 増えるごとに割り込みがかかる。

DRAWCOUNT   1/60 秒毎の割り込み。

KEYS         ボタンの押下状態が変化した時。Type Z でこの機能を使うと、INKEY 割り込みや READKEY()、INKEY() 等のキーボードを利用した機能が使えなくなる事に注意。

INKEY キーボード押下時。READKEY()関数と組み合わせて使う。type Pでは利用不可。  
MUSIC 音楽再生の時、最後の音の再生時に割り込み。  
WAVE WAVE ファイル再生終了時。  
CORETIMER コアタイマーの値がCORETIMER 命令で設定した値と一致した時。

yyy : 割り込みサブルーチンのラベル名を指定します。

z1, z2 ... : 割り込みサブルーチンに引き渡すパラメータ (省略可)。

#### ◎INTERRUPT STOP xxx

割り込みを停止します。xxx は停止する割り込みの種類を指定します。

#### <使用例>

MachiKania type M でキーボードを押した文字のフォントを8×8のマトリクスLEDにダイナミック点灯表示する。列側アノードでB0-B7、行側カソードでB8-B15に接続。

```
A=SYSTEM(103)           システムフォント先頭アドレス
B=0                     ダイナミック点灯させる行番号
USETIMER 1000          1ミリ秒ごとに割り込み
INTERRUPT TIMER, ISRTMR タイマー割り込みルーチン設定、タイマー割り込み開始
INTERRUPT INKEY, ISRKEY キーボード割り込みルーチン設定、キーボード割り込み開始
DO: IDLE: LOOP         割り込み待ち無限ループ

LABEL ISRKEY           キーボード割り込みサービスルーチン
VAR K                  割り込みルーチン内で使用する変数はローカル宣言が望ましい
K=READKEY() AND $FF   キーバッファから1つ読み出し
IF K>0 THEN A=SYSTEM(103)+K*8 文字キーの場合、フォントアドレスを計算
RETURN

LABEL ISRTMR          タイマー割り込みサービスルーチン
OUT8L 0               いったん全LED消灯
OUT8H (1<<B) XOR $FF マトリクスLEDの行番号出力 (B8-B15の該当行のみLにする)
OUT8L PEEK(A+B)       マトリクスLEDの行に対するフォントパターンをB0-B7に出力
B=(B+1) AND 7         次回の行番号設定
RETURN
```