＜BASIC 言語の書式＞
BASIC プログラムの記述は、行番号式、ラベル式、その混合、いずれの方法でも構いません。以下、仕様について述べます。

＜利用可能な変数型＞
利用できる変数の型は、32 ビット符号付整数(-2147483648 以上 +2147483647 以下)と、文字列型の２種類です。文字列の末端部には 0x00 が付加されます。

A-Z の２６個の整数型変数が利用可能です。文字列として扱う場合は A$のように記述します。ただし、A（整数型）と A$（文字列型）を同時に使用することは出来ません。

USEVAR などのステートメントで指定すると、6 文字までの長い名前の変数を使用出来ます。名前の最初の一文字は英字(A-Z)とアンダースコアー(_)で、二文字目からはそれらに加えて数字(0-9)も使う事が出来ます。

整数型の定数は、１０進法で記述します。１６進法を使う場合、「$1200」のように、頭に「$」を付加するか、「0x1200」の様に表記して下さい。

文字列型の定数は、「″」で囲って記述してください。「″」を使用する場合は、「CHR$($22)」のように記述することが出来ます。

＜一般命令＞
以下、x, y, z 等は整数値を、x$, y$, z$は文字列を、x#, y#, z#は浮動小数点型実数値を指します。xxx, yyy, zzz, www 等は任意のステートメントを指します。[ ]は省略可能である事を示します。

命令同士を「:」で区切ることにより、一行で複数のコマンドを処理することが出来ます。

BGCOLOR r,g,b
　　背景色指定。
BREAK
　　FOR-NEXT, DO-LOOP, WHILE-WEND ループから抜け出す。
CDATA x[,y[,z[...]]]
　　データー列を８ビット整数値で指定する。
CLEAR
　　すべての文字列型変数と整数型配列を破棄し、整数値を０とする。また、PCG の使用をやめ、表示キャラクターをリセットする。
CLS
　　スクリーン消去。
COLOR x
　　テキスト色指定。
CONTINUE
　　FOR-NEXT, DO-LOOP, WHILE-WEND ループ中で、以降のコードをスキップする。
CURSOR x,y
　　カーソル位置指定。
CDATA xxx[,yyy[,zzz[...]]]
　　データー列を８ビット整数値で指定する。
DATA xxx[,yyy[,zzz[...]]]
　　データー列を整数値もしくは文字列で指定する。
DIM xxx [, yyy [, zzz [, ...]]]
　　整数型もしくは浮動小数点型の配列を割り当てる。
　　xxx,yyy,zzz は、例えば「A(10)」のように記述する。この場合、A(0)から A(10)までの１１個の整数型変数が確保される。浮動小数点型配列の場合は、「A#(10)」の様に記述する。多次元配列も、宣言することが出来る。
DO WHILE x
LOOP
　　x が 0 以外の場合、DO 文から LOOP 文までのステートメントを繰り返し実行する。
DO UNTIL x

LOOP
　　x が 0 の場合、DO 文から LOOP 文までのステートメントを繰り返し実行する。
DO
LOOP WHILE x
　　DO 文から LOOP 文までのステートメントを実行し、x が 0 以外の場合、繰り返す。
DO
LOOP UNTIL x
　　DO 文から LOOP 文までのステートメントを実行し、x が 0 の場合、繰り返す。
DRAWCOUNT
　　DRAWCOUNT 値を指定する。DRAWCOUNT 値に付いては、DRAWCOUNT()関数を
　　参照。
END
　　BASIC プログラムを停止する。
EXEC x[,y[,z[...]]]
　　機械語を実行する。ただし、x,y,z は 32 ビット整数値。
FOR x=yyy TO zzz [STEP www]
NEXT
　　yyy で示された計算結果を x に代入し、x の値が zzz になるまで次の NEXT 文
　　までのステートメントを、繰り返し実行する。繰り返しのたび、x の値は
　　www ずつ増加する（省略された場合は１ずつ）。「NEXT」の次に何も記述
　　しないことに注意。
GOSUB xxx [, yyy [, zzz [, ...]]]
　　現在の実行位置を記憶し、xxx 行目(もしくはラベル)に移動する。yyy, zzz
　　等は、サブルーチンに引き継がれる引数（ARGS()関数を参照）。
GOTO xxx
　　xxx 行目(もしくはラベル)に移動する。
IF x または x# THEN yyy [ELSE zzz]
　　x が 0 以外のとき、yyy を、0 のとき zzz を実行。yyy および zzz は、複数のステート
　　メントを「:」で挟んで記述可能。
IF x または x# THEN
xxx
[ELSEIF y または y# THEN
yyy]
[ELSE
zzz]
ENDIF
　　x が 0 以外の時 xxx を、それ以外で且つ y が 0 以外の時（記述された場合）yyy を、
　　それ以外の場合に zzz を実行。ELSEIF ステートメントは、複数記述可。各行で、
　　THEN ステートメントの次には何も記入しないことに注意。
LABEL xxx
　　GOTO/GOSUB のジャンプ先を登録する。xxx は、英数字６文字以内の文字列。
[LET] x=yyy
　　y で示された計算結果を、x（整数型変数）に代入する。「LET」は省略可。
[LET] x#=yyy
　　y で示された計算結果を、x（浮動小数点型変数）に代入する。「LET」は省略可。
[LET] x$=yyy
　　yyy で示された文字列（もしくは連結結果;連結演算子は「+」）を、x$に
　　代入する。「LET」は省略可。
MUSIC x$[,y]
　　BGM を演奏する。詳細は、下記<MUSIC>の項を参照。Type M では、y=1 の時右側だけ、
　　y=2 の時左側だけ、y=3 もしくは省略した場合に両方から音が出る。
PLAYWAVE x$[,y]
　　音楽用の WAVE ファイル(ファイル名を x$で指定)を演奏する。WAVE のフォーマットは、
　　Type M の場合はステレオもしくはモノラル（Type Z ではモノラルのみ）、ビット長
　　は８、サンプリング周波数が 15700 Hz の物を指定する。16000 Hz でも再生出来るが、
　　音程と再生速度が少しずれる。y を指定した場合、指定の箇所から再生される（１秒
　　目から再生したい場合は、15700 を指定）。x$に長さ０の文字列を指定すると、現在
　　再生中の音楽が停止する。
PALETTE n,r,g,b
　　パレット指定。

PCG x,y,z
　　ASCII コードが x の文字の表示キャラクターを変更する。y,z は、キャラク
　　ターデーター。詳細は、下記<PCG>の項を参照。
POKE x,y
　　x で示される物理的アドレスに、y で示される値(１バイト値)を書き込む。
PEEK16 x,y
　　x で示される物理アドレスに、y で示される値(16 ビット値)を書き込む。x が奇数値の
　　場合、例外停止するので注意。
PEEK32 x,y
　　x で示される物理アドレスに、y で示される値(32 ビット値)を書き込む。x が４の倍数で
　　無い場合、例外停止するので注意。
PRINT [x または x$または x# [,または; [y または y$または y# [...]]]]
　　ディスプレイに、整数値または文字列を表示する。「;」を使用した場
　　合、次の表示が続けて行われる。「,」を使用した場合、１０文字ずつ
　　に区切って表示される。どちらも使用しない場合、次の表示は行を変え
　　て行われる。
REM xxx
　　何も実行しない
RESTORE xxx
　　DATA 読み出し開始位置を指定。xxx は行番号もしくはラベル。
RETURN
　　最後に実行された GOSUB 文の次のステートメントに移動する。戻り値を指
　　定することががができる。この場合の戻り値は GOSUB()関数にて取得が可能。
SCROLL x,y
　　画面を横方向、もしくは縦方向(斜めも可)に動かす。動かす方向と大きさ
　　は、x, y でそれぞれ、横方向の移動度、縦方向の移動度として指定する。
SOUND xxx[,y]
　　効果音を再生する。詳細は、下記<SOUND>の項を参照。xxx は行番号もしく
　　はラベル。Type M では、y=1 の時右側だけ、y=2 の時左側だけ、y=3 もしくは
　　省略した場合に両方から音が出る。
USEPCG [x]
　　PCG を使用、もしくは使用停止する。x=0 で使用停止、x=1 で使用、x=2 で
　　キャラクターをリセットして使用。x を省略した場合は、x=1 と同じ。
USEVAR xxx [, yyy [, zzz [, ...]]]
　　英数字とアンダースコアー(_)で最大６文字までの変数名を使用できるようにする。
　　このステートメント以降で xxx, yyy 等の長い変数名が使用可能となる。
VAR xxx [, yyy [, zzz [, ...]]]
　　サブルーチン内で使う、ローカル変数を指定する。xxx, yyy 等は、A-Z の
　　アルファベットで指定する。
WAIT x
　　x で示された時間、プログラムの実行を停止する。x が 60 の場合、約１秒間
　　停止。
WHILE x
WEND
　　x が 0 以外の場合、WHILE 文から WEND 文までのステートメントを繰り返し実行する。
WIDTH x
　　キャラクターディスプレイの横幅を文字数で指定。x は 30、36、40、48、
　　もしくは 80。

＜整数型関数＞
以下、x, y, z は整数値を、x$, y$, z$は文字列を指します。[ ]は省略可能である事
を示します。

ABS(x)
　　x の絶対値を返す。
ARGS(x)
　　サブルーチン中で、GOSUB もしくはメソッドに渡された x 番目の引数を整数値として取
　　り出す。但し、
　　　　ARGS(0)は、引数の数を返す。
　　　　ARGS(-1)は、一つ前に使われた引数を格納する配列へのポインターを返す。

ARGS(-2)は、クラスで用いられた場合、オブジェクトへのポインターを返す。

ASC(x$)

文字列の最初の一文字の、アスキーコードを返す。

CREAD()

DATA 文の後から、一つずつデーター（8 ビット整数値）を読み出す。「READ()」関数も参照。

DRAWCOUNT()

DRAWCOUNT 値を得る。DRAWCOUNT は１６ビット整数値で、1/60 秒ごとに１ずつ増える。

GOSUB(xxx [, y [, z [, ...]]])

GOSUB 命令と同じだが、戻り値(RETURN を参照)を得ることが出来る。xxx は、ラベルもしくは行番号。。yyy, zzz 等は、サブルーチンに引き継がれる引数（ARGS()関数を参照）。

INKEY([x])

x を指定しない場合、現在押されているキーの ASCII 値を返す。押されていない場合は、０。ASCII 値で x を指定した場合、そのキーが押されているかどうかを返す。

INT(x#)

実数値 x#を整数値に変換して返す。

KEYS([x])

キー入力を得る。x の値は以下の通り。x を指定しない場合は、x=63 と同じ。

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

Type Z でこの機能を使うと、INKEY 割り込みや READKEY()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。Type M では、その限りではない。

LEN(x$)

文字列の長さを返す。

MUSIC()

BGM の演奏の残り数を返す。

NOT(x)

x=0 の場合に１を、そうでない場合に０を返す。

PEEK(x)

x で示される物理アドレスから１バイト読み取り、返す。

PEEK16(x)

x で示される物理アドレスから２バイト読み取り、16 ビット値で返す。x が奇数値の場合、例外停止するので注意。

PEEK32(x)

x で示される物理アドレスから４バイト読み取り、32 ビット値で返す。x が４の倍数で無い場合、例外停止するので注意。

PLAYWAVE([x])

x を指定しない場合、もしくは０を指定した場合、再生中の WAVE ファイルの残りサンプリング数を返す。１を指定した場合、現在再生中のサンプリング番号を、２を指定した場合、WAVE ファイルの総サンプリング数を返す。

READ()

DATA 文の後から、一つずつデーター（32 ビット整数値）を読み出す。「CREAD()」関数も参照。

READKEY()

キーボードバッファーから一文字読み込み、返す。バッファーが空の時は０を返す。戻り値は 24 ビット整数で、内容は以下の通り。

```
bits 0-7   : ASCII コード
bits 8-15  : 仮想キーコード
bits 16-23 : シフトキー押下状態。
             上位から
<0><CAPSLK><NUMLK><SCRLK><Win><ALT><CTRL><SHIFT>。
```

RND()

0 から 32767 までの擬似乱数を返す。

SGN(x)
　　x の符号(-1, 0, または 1)を返す。
STRNCMP(x$,y$,z)
　　２つの文字列のうち z 文字分を比較し、結果を返す。同じ文字列の場合は０。
TVRAM([x])
　　ビデオ RAM の x 番目の内容を、バイト値で返す。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す。
VAL(x$)
　　１０進数もしくは１６進数文字列としての x$の値を、整数値で返す。

＜浮動小数点型関数＞
ACOS#(x#)
　　x# の逆余弦を実数値で返す。
ARGS#(x)
　　サブルーチン中で、GOSUB もしくはメソッドに渡された x 番目の引数を実数値として取
　　り出す。
ASIN#(x#)
　　x# の逆正弦を実数値で返す。
ATAN#(x#)
　　x# の逆正接を実数値で返す。
ATAN2#(y#,x#)
　　y#/x# の逆正接を実数値で返す。
CEIL#(x#)
　　x# 以上の最小の整数を実数値で返す。
COS#(x#)
　　x# の余弦を実数値で返す。
COSH#(x#)
　　x# の双曲線余弦を実数値で返す。
EXP#(x#)
　　e を底とする x# の指数関数値を実数値で返す。
FABS#(x#)
　　x# の絶対値を実数値で返す。
FLOAT#(x)
　　整数値 x を浮動小数点型実数値に変換して返す。
FLOOR#(x#)
　　x# 以下の最大の整数を実数値で返す。
FMOD#(x#,y#)
　　x# を y# で割った剰余を実数値で返す。
GOSUB#(xxx [, y [, z [, ...]]])
　　GOSUB 命令と同じだが、戻り値(RETURN を参照)を得ることが出来る。xxx は、
　　ラベルもしくは行番号。。yyy, zzz 等は、サブルーチンに引き継がれる引数（
　　ARGS()関数を参照）。
LOG#(x#)
　　x# の自然対数を実数値で返す。
LOG10#(x#)
　　x# の常用対数を実数値で返す。
MODF#(x#)
　　x# の小数部を実数値で返す。
PI#
　　3.141593 を返す。
POW#(x#,y#)
　　x# の y# 乗を実数値で返す。
SIN#(x#)
　　x# の正弦を実数値で返す。
SINH#(x#)
　　x# の双曲線正弦を実数値で返す。
SQRT#(x#)
　　x# の平方根を実数値で返す。
TAN#(x#)
　　x# の正接を実数値で返す。

TANH#(x#)
　　x# の双曲線正接を実数値で返す。
VAL#(x$)
　　１０進数文字列としての x$の値を、実数値で返す。

＜文字列型関数＞
A$(x [,y])など
　　x の値が０の場合、文字列全体を返す。
　　x の値が正の場合、x で示される位置より右側の文字列を返す。
　　x の値が負のとき、文字列の右側 x 文字を返す。
　　y が指定された場合、y 文字分の文字列を返す。
ARGS$(x)
　　サブルーチン中で、GOSUB もしくはメソッドに渡された x 番目の引数を文字列として取り出す。
CHR$(x)
　　x をアスキーコードとする文字を返す。
DEC$(x)
　　x の値を、１０進数の文字列として返す。
FLOAT$(x#)
　　実数値 x#を、１０進数の文字列として返す。
GOSUB$(xxx [, y [, z [, ...]]])
　　GOSUB 命令と同じだが、戻り値(RETURN を参照)を文字列として得ることが出来る。
　　xxx は、ラベルもしくは行番号。。yyy, zzz 等は、サブルーチンに引き継がれる引数（ARGS()関数を参照）。
HEX$(x [,y])
　　x の値を、１６進数の文字列として返す。y が指定された場合、y バイト長の
　　文字列になる。
INPUT$()
　　文字列入力状態になり、入力が終了すると(Enter が押されると)文字列を返す。
SPRINTF$(x$,y#)
　　x$で示される書式に従って、実数 y#の内容を文字列として返す。
READ$()
　　DATA 文の後から、一つずつ文字列データーを読み出す。

＜整数演算子＞
&x
　　変数 x の格納アドレス(ポインター)
-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

＜特殊命令・関数＞
IDLE
　　アイドルモード(″wait″アセンブリー)に入る。ビデオ信号作製やタイマーなどの割り込みがかかるまで、CPU が停止する。
OPTION x[,y[,z ...]]]
　　各種オプションを指定する。オプションについては、下記＜オプション＞の項を参照。
SYSTEM x , y
　　様々なシステム値の設定を行なう。＜SYSTEM＞の項を参照。
SYSTEM(x)
　　様々なシステム値を、整数値で返す。＜システム変数＞の項を参照。
SYSTEM$(x)
　　様々なシステム値を、文字列で返す。＜システム変数＞の項を参照。

＜グラフィック関連命令と関数＞

BOXFILL [x1,y1],x2,y2[,c]
　　座標(x1,y1),(x2,y2)を対角線とするカラーｃで塗られた長方形を描画。
CIRCLE [x,y],r[,c]
　　座標(x,y)を中心に、半径ｒ、カラーｃの円を描画。
CIRCLEFILL [x,y],r[,c]
　　座標(x,y)を中心に、半径ｒ、カラーｃで塗られた円を描画。
GCLS
　　画面クリアー。
GCOLOR c
　　それぞれの命令で、c を省略した場合の色を指定。
GPALETTE n,r,g,b
　　パレット指定。
GPRINT [x,y],c,bc,s$
　　座標(x,y)にカラーｃで文字列 s$を表示、bc:背景色（負数の場合背景色指定なし）。
LINE [x1,y1],x2,y2[,c]
　　座標(x1,y1)から(x2,y2)にカラーｃで線分を描画。
POINT x,y
　　グラフィック現在位置を、設定する。
PSET [x,y][,c]
　　座標(x,y)の位置にカラーｃで点を描画。
PUTBMP [x,y],m,n,bbb
　　横 m*縦 n ドットのキャラクター(bbb で指定)を座標(x,y)に表示。
　　サイズ m*n の配列 bmp に、単純にカラー番号を並べる。
　　ただし、カラーが 0 の部分は透明色として扱う。ただし、bbb はラベル名もしくは配列へのポインター。
USEGRAPHIC [x]
Type M の場合
　　グラフィックディスプレイを使用、もしくは使用停止する。x=0 で使用停止、x=1, 5, 9 で使用、x=2, 6, 10 で画面とパレットをクリアーして使用、x=3,7,11 でグラフィック領域を確保するが表示はキャラクターディスプレイのまま。ただし、グラフィックディスプレイ未使用の状態で x=0, 4, 8 の場合は、領域を確保する。x を省略した場合は、x=1 と同じ。
　　ただし、x の値が 0-3 の場合は Type-Z 互換グラフィック、4-7 の場合は標準グラフィック、8-11 の場合はワイドグラフィック。
Type Z の場合
　　グラフィックディスプレイを使用、もしくは使用停止する。x=0 で使用停止、x=1 で使用、x=2 で画面とパレットをクリアーして使用、x=3 でグラフィック領域を確保するが表示はキャラクターディスプレイのまま。ただし、グラフィックディスプレイ未使用の状態で x=0 の場合は、領域を確保する。x を省略した場合は、x=1 と同じ。

GCOLOR(x,y)
　　グラフィック座標(x,y)の表示中パレット番号を返す。

＜ファイル関連命令と関数＞
ファイルは、最大２つまで同時に開く事が出来ます。

FCLOSE [x]
　　ファイルを閉じる。引数(x)がある場合は、そのファイルハンドルで指定されたファイルを閉じる。
FGET x,y
　　バッファー(x に配列として指定)に y バイト読み込む。関数として呼ばれた場合は、読み込みに成功したバイト数を返す。
FILE x
　　アクティブなファイルハンドル（１もしくは２）を x に指定する。
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く。同時に開けるファイルの数は、２つまで。関数として呼ばれた場合は、ファイルハンドルを返す。y$としては、次のものが有効。
　　″r″
：ファイルを読み込みモードで開く。
　　″r+″　：″r″と同じだが書き込みも可能。
　　″w″　：ファイルを書き込みモードで開く。同名のファイルが在る場合は、以前のファイルは消去される。
　　″w+″　：″w″と同じだが、読み込みも可能。
　　″a″　：ファイルを書き込みモードで開く。同名のファイルが在る場合は、ファイルは消去されず、ファイルの最後尾から書き込まれる。
　　″a+″　：″a″と同じだが、読み込みも可能。
　　z には、割り当てたいファイルハンドル（１もしくは２）を指定する。省略した場合、１が指定される。
FPRINT [x または x$または x# [,または; [y または y$または y# [...]]]]
　　PRINT 命令と同じだが、画面ではなくファイルに情報が書き込まれる。
FPUT x,y
　　バッファー(x に配列として指定)の y バイト分を書き込む。関数として呼ばれた場合は、書き込みに成功したバイト数を返す。
FPUTC x
　　x で示される１バイトのデーターをファイルに書き込む。関数として呼ばれた場合は、書き込みに成功したバイト数（１もしくは０）を返す。
FREMOVE x$
　　x$で示される名前のファイルを、SD カードから削除する。関数として呼ばれた場合は、削除に成功したか(0)、失敗したか(-1)を返す。
FSEEK x
　　x で示されるファイル位置に移動する。
SETDIR x$
　　カレントディレクトリーを x$に移動する。関数として呼ばれた場合、成功すれば 0 を、エラーが有れば 0 以外を返す。
FEOF()
　　FOPEN で開いたファイルの現在のファイル位置が、末端に到達しているかどうかを返す。１で末端に到達、０で未到達。
FGETC()
　　FOPEN で開いたファイルから１バイト読み込み、整数値として返す。ファイル末端に到達しているなどで読み込みに失敗した場合、-1 を返す。
FLEN()
　　FOPEN で開いたファイルのファイル長を、バイト数で返す。
FSEEK()
　　FOPEN で開いたファイルの、現在のファイル位置を返す。
FINPUT$([x])
　　FOPEN で開いたファイルから、x で示された長さの文字列を読み込む。x が省略された場合は、行の最後まで読み込む(改行コードが含まれる)。
GETDIR$()

カレントディレクトリーを文字列として返す。

＜タイマー関連命令と関数＞
タイマーは、通常タイマーとコアタイマーの２つがあります。通常タイマーは速度の設定や値の変更など出来る、汎用タイマーです。コアタイマーは、CPU クロックと同期した特殊なタイマーで、値を読む事は出来るが設定する事は出来ません。

CORETIMER
　コアタイマーを用いた割り込みの時期を設定する。コアタイマーの値を変更するわけではない事に注意。
USETIMER x
　タイマーを開始する。x はタイマーの速度を、μ 秒で指定（175769 以下の値）。
TIMER x
　現在のタイマー値を 32 ビット整数(x)で設定する。
CORETIMER()
　現在のコアタイマーの値を、32 ビット整数値として返す。
TIMER()
　タイマーの現在値を、32 ビット整数値として返す。

＜割り込み命令＞
INTERRUPT xxx,yyy[,z1[,z2 ...]]
　割り込みを設定する。xxx は割り込みの種類、yyy は割り込み時のサブルーチンをラベルで指定。z1, z2 等を指定すると、割り込み用サブルーチンの引数となる。使用可能な割り込みの種類は以下の通り。
　　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。Type M では、その限りではない。
　　INKEY
　　キーボード押下時。READKEY()関数と組み合わせて使う。
　　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。
INTERRUPT STOP xxx
　割り込みを停止する。xxx は割り込みの種類。

＜MUSIC＞
MUSIC 命令では、BGM 用のデーターを文字列で指定します。文字列の書式は、ABC notation 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

BGM 演奏時に一度に設定できる音の数は、31 迄です。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。

添付の music.bas に、使い方に関するサンプルがありますので、参考にして下さい。

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

DATA列では、３２ビット整数値として、交換音を表現します。この整数値の下位１６ビットは周波数の指定です。2048が440Hz(ラの音)に対応します。値が大きくなるほど、より低い音が出ます。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以下の値で、効果音の繰り返し回数を指定します。これらのデーターの数は、32を超えないようにして下さい。

添付のsound.basに、使い方に関するサンプルがありますので、参考にして下さい。

＜PCG＞
PCG(Programmable Character Generator)を用いると、ASCIIコードごとにフォントを指定して、疑似グラフィックスとして表示させることが出来ます。使用する場合は、まず

USEPCG

とします。フォントの変更は、PCGステートメントを用いて、

PCG 0x80,0x80402010,0x08040201

の様に設定します。この例では、ASCIIコード0x80の文字のフォントを設定していて、バックスラッシュの様な記号(左上から右下に向かう斜め線)が表示されるようになります。PCGの利用を停止し、オリジナルのフォントに戻す場合は、

USEPCG 0

とします。再度PCGを使用したい場合は、

USEPCG

として下さい。先に設定したフォントデーターが、復活します。なお、先に設定したフォントデーターを破棄してPCGの使用を始めたい場合は、

USEPCG 2

として下さい。

＜システム変数＞
SYSTEM関数及びSYSTEMステートメントを用いて、各種システム情報をやりとりすることが出来ます。

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、"1.2"等を返す。
SYSTEM$(2)
　　BASIC バージョン文字列、"KM-1208"等を返す。
SYSTEM$(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。

＜入出力命令・関数＞
入出力機能は、Type M でのみ使えます。

ANALOG(x)
　　PORTB の下位から x ビット目のアナログ入力値(１０ビット値;0-1023 の値)を返す。但し、x=16, 17, 18 の場合は、PORTE5, 6, 7 がそれぞれ指定される。
IN(x)
　　PORTB の下位から x ビット目の入力値(１ビット値;１か０)を返す。但し、x=16, 17, 18 の場合は、PORTE5, 6, 7 がそれぞれ指定される。入力は PIC 内部でプルアップされる。
IN8H()
　　PORTB の上位８ビットの入力値(８ビット値)を返す。入力は PIC 内部でプルアップされる。
IN8L()
　　PORTB の下位８ビットの入力値(８ビット値)を返す。入力は PIC 内部でプルアップされる。
IN16()
　　PORTB の入力値(１６ビット値)を返す。入力は PIC 内部でプルアップされる。
PWM x[,y[,z]]
　　PWM 出力を行なう。z=1 の場合 PORTD10 に、z=2 の場合 PORTD11 に出力される。z を省略した場合は、PORTD10。x にはデューティー比を、0-1000 の値で指定する。y はパルスの周波数を、Hz で指定する（省略した場合は、1000；有効値は 6-95454）。
SERIAL x[,y[,z]]
　　シリアル通信を開始する。x にはボーレートを指定する(シリアル通信の使用を終了する場合は、x=0 を指定)。y=0 の場合パリティ無し、y=1 の場合偶数パリティ、y=2 の場

合奇数パリティ、y=3 の場合９ビットパリティ無し。y を省略した場合は、y=0 と同じ。
z には受信バッファーの文字数を指定する。z を省略した場合、1/60 秒の連続受信が保証される大きさのバッファーを確保する。

SERIALIN([x])
シリアル通信で、一文字受信する。受信が無い場合は、-1 を返す。x=1 を指定すると、受信バッファーの文字数を返す。パリティ有り８ビットの受信の場合は、パリティエラーが起きた場合、0x100 以上の値を返す。

SERIALOUT x
シリアル通信で、一文字送信する。

OUT x,y
PORTB の下位から x ビット目に、y で示された値(１ビット値;１か０)を出力する。但し、x=16, 17, 18 の場合は、PORTE5, 6, 7 がそれぞれ指定される。PORTE5, 6, 7 は、オープンドレイン出力であることに注意。

OUT8H x
PORTB の上位８ビットに、x で示された値(８ビット値)を出力する。

OUT8L x
PORTB の下位８ビットに、x で示された値(８ビット値)を出力する。

OUT16 x
PORTB に、x で示された値(１６ビット値)を出力する。

I2C [x]
I2C 利用をマスターモードで開始する。x は、クロック数を kHz 単位で指定(有効値は、12-3409)。省略した場合は、X=100。

I2CWRITE x[,y[,z[, ...]]
I2C 固定長送信を行なう。x は７ビットのスレーブアドレス。y, z 等はオプションの送信コードで、バイト値で指定。

I2CREAD(x[,y[,z[, ...]])
I2C 固定長送信の後、１バイトの受信を行なう。x は７ビットのスレーブアドレス。y, z 等はオプションの送信コードで、バイト値で指定。成功した場合に８ビット値を返す。エラーの場合、-1 を返す。

I2CWRITEDATA x,y,z1[,z2[,z3...]]
I2C 複数バイト送信を行なう。x は７ビットのスレーブアドレス。y は送信する内容を含むバッファーへのポインター。z1 はバッファーのバイト数。z2, z3 等はオプションの送信コードで、これらのバイト値がまず送信され、続けてバッファー y の内容が z1 バイトに渡って送信される。

I2CREADDATA x,y,z1[,z2[,z3...]]
I2C 複数バイト受信を行なう。x は７ビットのスレーブアドレス。y は受信する内容を格納するバッファーへのポインター。z1 は受信するバイト数。z2, z3 等はオプションの送信コードで、これらのバイト値がまず送信され、続けて z1 バイトのデーターを受信してバッファー y に格納する。

I2CERROR()
直前の I2C 送受信でエラーが在った場合に０以外を、なければ０を返す。

SPI x[,y[,z1[,z2]]]
SPI 利用をマスターモードで開始する。x は、クロック数を kHz 単位で指定(有効値は、93-47727)。y は、１ワードのビット数を 8/16/32 で指定(省略した場合は、8)。z1 は、SPI クロックの取り扱い方を指定(省略した場合は、0)。詳細は、下記に。z2 は、CS ラインにどのポートを使用するかを指定する。省略した場合は、0x39(PORTD9)。他のポートを使う場合、例えば PORTB5 なら 0x15、PORTB3 なら 0x13 とする。

z1=0:アイドル時に L、データー変更は L に変化する時(CKP=0,CKE=1)
z1=1:アイドル時に L、データー変更は H に変化する時(CKP=0,CKE=0)
z1=2:アイドル時に H、データー変更は H に変化する時(CKP=1,CKE=1)
z1=3:アイドル時に H、データー変更は L に変化する時(CKP=1,CKE=0)

SPIWRITE x[,y[,z[, ...]]
SPI 固定長送信を行なう。x, y, z 等は送信コード。

SPIREAD([x[,y[,z[, ...]]])
SPI 固定長送信(オプション)の後、１ワードの受信を行ない、返す。x,y,z 等は、受信前に送信するコード。

SPIWRITEDATA x,y[,z1[,z2[,z3...]]]
SPI 複数ワード送信を行なう。x は送信する内容を含むバッファーへのポインター。y はバッファーのワード数。z1,z2,z3 等はオプションの送信コードで、これらがまず送信され、続けてバッファー x の内容が y ワードに渡って送信される。

SPIREADDATA x,y[,z1[,z2[,z3...]]]
　　SPI 複数ワード受信を行なう。x は受信する内容を格納するバッファーへのポインター。y は受信するワード数。z1,z2,z3 等はオプションの送信コードで、これらがまず送信され、続けて y ワードのデーターを受信してバッファー x に格納する。
SPISWAPDATA x,y[,z1[,z2[,z3...]]]
　　SPI 複数ワード送受信を行なう。x は送受信する内容を格納するバッファーへのポインター。y は送受信するワード数。z1,z2,z3 等はオプションの送信コードで、これらがまず送信される。続けて、バッファー x の内容を送信した後にデーターを受信してバッファー x に格納しなおす動作を、y ワードに渡って繰り返す。

＜オプション＞
OPTION ステートメントを使って、コンパイル時もしくは実行時に色々なオプションを指定する事が可能です。次のオプションが有ります。

OPTION NOLINENUM
　　コンパイル時に、行番号を指定する命令を挿入しない。このオプションにより、プログラムサイズを小さくして実行速度を増加する効果が見込める。ただし、エラーが発生した場合に、どの行でのエラーかは分からなくなる。
OPTION FASTFIELD
　　クラスを２つ以上使っている時、パブリックフィールド名の重複がない場合に、フィールドへのアクセスを高速化する。ただし、存在しないオブジェクトのフィールドにアクセスしてもエラーにならない場合があるため、予め BASIC コードにエラーがない事を確認してから用いる事が望ましい。なお、クラスを一つだけしか使っていない場合は、指定しなくてもこのオプションは有効になっている。
OPTION CLASSCODE
　　このオプションの後に、クラスを記述するコードを書く事が出来る。但し、ファイル名を「クラス名.BAS」として保存する事。この機能を用いれば、クラスを一つだけ使うコードなら、１ファイルに収める事が出来る。クラスの開発用に用いると、便利。

＜クラス・オブジェクト関連機能＞
クラスとオブジェクトの利用方法について、詳しくは class.txt を参照して下さい。
以下は、関連する命令と関数です。

USECLASS x[,y[,z[, ...]]]
　　クラスの利用を宣言する。x,y,z 等は、クラス名を 6 文字以内の英数字で指定。
FIELD [PUBLIC] x[,y[,z[, ...]]]
　　クラスファイル中で、パブリックフィールドを宣言する。"PUBLIC"は省略可。x,y,z 等はフィールド名を 6 文字以内の英数字で指定。
FIELD PRIVATE x[,y[,z[, ...]]]
　　クラスファイル中で、プライベートフィールドを宣言する。x,y,z 等はフィールド名を 6 文字以内の英数字で指定。
STATIC [PUBLIC] x[,y[,z[, ...]]]
　　クラスファイル中で、パブリックスなタティック変数を宣言する。"PUBLIC"は省略可。
STATIC PRIVATE x[,y[,z[, ...]]]
　　クラスファイル中で、プライベートなスタティック変数を宣言する。USEVAR と同じ。
METHOD x
　　クラスファイル中で、メソッドを宣言する。x は、メソッド名を 6 文字以内の英数字で指定。
NEW(x[,y[,z[, ...]]])
　　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ーを返す。x はクラス名を指定。y,z 等はコンストラクターがある際に利用される引数。
DELETE x[,y[,z[, ...]]]
　　作成されたオブジェクトを破棄する。x,y,z 等は、オブジェクトを格納する変数名。
CALL x
　　x で指定されたオブジェクトのメソッドを呼び出す。「CALL」は省略可。

＜ヒント＞
MachiKania ver 1.2 以降、FOR-NEXT ループ、WHILE-WEND ループ、DO-LOOP ループの途中で、RETURN 文が使えるようになりました。ただし、GOTO 文でループの外に飛ぶと、予期せぬ結果（機器のリセット等）を引き起こします。また、GOSUB 文でサブルーチンを呼んだり、別のループをネストして使う事は可能です。

ON GOTO 分や ON GOSUB 文はサポートしていません。ただし、例えば次のように記述することで、同様の動作をさせることは可能です。

```
 GOSUB 10000+A
 ....
 10000 PRINT "A=0" : RETURN
 10001 PRINT "A=1" : RETURN
 10002 PRINT "A=2" : RETURN
```

一行中で連続して文字列を扱うと、"String too complexed"というエラーがでて、停止することがあります。この場合は、文字列を扱う命令を独立した行するか、文字列関連の演算を幾つかのステップに分けて、それぞれ１行ずつの記述にして試してみて下さい。

割り込み関数中でグローバル変数にアクセスする際は、注意が必要です。同名の変数がいずれかのサブルーチン内で VAR 指定されてローカル変数としても使われている場合、いつ割り込みがトリガーされるかによって、変数値がグローバル変数の物になるかローカル変数の物になるか、不定になってしまいます。従って、割り込み関数中でグローバル変数を扱いたい場合は、同名の変数を VAR 指定しないようにして下さい。

＜バージョン履歴＞

・KM-1303 2019 年５月公開。
  ・タイマー機能(USETIMER,TIMER, CORETIMER ステートメントと TIMER(), CORETIMER()関数)を追加。
  ・割り込み機能(INTERRUPT ステートメント)を追加。
  ・オプション機能(OPTION ステートメント)を追加。
  ・アイドル機能(IDLE ステートメント)を追加。
  ・READKEY()関数を追加。
  ・EXEC()関数を追加。
  ・変数名などで、英数字に加えてアンダースコアーが使用可能に。
  ・PRINT でカンマを使った時の表示不具合を修正。
・KM-1302 2019 年３月公開。
  ・オブジェクト指向プログラミングに対応
  ・args(0)で引数の数を取得できるようにした
  ・POKE16, POKE32, PEEK16(), PEEK32()を追加
  ・&演算子を追加
  ・GETDIR$()関数と SETDIR ステートメントを追加
・KM-1301 2018 年 12 月公開。
  ・I2C 機能を搭載
  ・SPI 機能を搭載
  ・PUTBMP の第５引数に長い名前の変数が使えなかったバグの修正
・KM-1300 2018 年８月公開。