Panasonic

パーソナルコンピュータ

MSX 2

使用説明書

Panasonic

MSX 2 パーソナルコンピュータ

BASIC説明書

はじめに

このたびは パナソニックパーソナルコンピュータFSシリーズをお買上げいただきまして，まことにありがとうございました。

本書はＭＳＸパーソナルコンピュータに搭載されているプログラム言語，MSX$_2$ ＢＡＳICの文法書です．本書では初心者の方にも活用していただけるように，親しみやすい使用例を随所に入れてあり，また，第３章では簡単なサンプルプログラムとその解説を幾つか用意しました．

なお，機械の操作，MSX　BASICの基礎などについては「取扱い説明書」でわかりやすく解説していますので，そちらをお読みください．

MSX はアスキーの商標です．

本書の使い方

本書は次のような構成になっています.

第1章　MSX$_2$ BASICの概要

BASICを使ってプログラムを作るときに必要となる基礎的な知識，および画面やファイルの入力についてを説明しています．これらは，第2章の命令の使い方についての説明を理解するために必要なことがらですが，BASICを初めて使う方で，第1章を読むのが重荷であるときは，とりあえず第2章や3章の文例やサンプルプログラムを試したり改造してみたりすることから始めるとよいでしょう.

第2章　コマンド・ステートメント・関数

MSX$_2$ BASICで用意している命令や関数の使い方をアルファベット順に説明しています。

第3章　サンプルプログラム

第2章で説明した命令をいくつか取り上げ，実践的なサンプルプログラムを紹介しながら解説しています.

資　料

キャラクタコード表，メモリマップ，エラーメッセージ表などプログラムの作成に役立つ資料をまとめてあります.

付　録

第2章で説明した命令のうち通常の使用では必要ないが，高度な使用をする場合に参考となる事項を補足して説明しています.

ご　注　意

○本書の内容の一部または全部を無断で転載することは禁止されています。

○本書の内容は将来予告なしに変更することがあります。

○DISK BASIC関係の命令は別冊の「DISK BASIC／MSX-DOS説明書」を参照してください。

目 次

第1章　文法編

1.1 MSX　BASICとは——1

1.2 動作モード——2

1.3 文——2

1.4 行——3

1.5 行番号——3

1.6 キャラクタセット——4

1.7 BASICで扱うデータ——4

1.8 定数——5

1.8.1 文字定数……5

1.8.2 整数型定数……5

1.8.3 単精度実数型定数……6

1.8.4 倍精度実数型定数……6

1.9 変数——6

1.9.1 変数名……7

1.9.2 型宣言文字……7

1.9.3 配列変数……7

1.9.4 変数と使用メモリ……8

1.9.5 システム変数……9

1.10 型変換——10

1.11 式と演算——11

1.11.1 算術演算……12

1.11.2 関係演算……13

1.11.3 論理演算……14

1.11.4 関数……16

1.12 文字列の連結——17

1.12.1 文字列の連結……17

1.12.2 文字列の比較……17

1.13 演算の優先順位——18

1.14 エラーメッセージ——18
1.15 画面モード——19
1.15.1 テキストモード……20
1.15.2 グラフィックモード……22
1.16 カラーコード——25
1.17 スプライト機能——26
1.18 ファイル——27
1.18.1 ファイル番号……28
1.18.2 ファイルスペック……28
1.18.3 デバイス名……28
1.18.4 ファイル名……29
1.19 割込み——29

第2章　リファレンス編

第3章　サンプルプログラム

数あてゲーム
（IF文、RND関数）……214
グラフィック画面への文字の表示
（OPEN文、PRINT＃文）……216
メモリ内容の表示
（INPUT文、PEEK文）……217
指定した色の表示
（INPUT文、PUT　SPRITE文）……218
画面の切換
（SET　PAGE文、RND関数）……219

時計を作ろう!
（GET文，MID関数，LINE文）……220
簡単なゲーム
（STICK関数，STRIG関数）……222
展覧会の絵を演奏する
（PUT KANJI文，PLAY文）……224

第4章　資料編

予約語表——227
キャラクターコード表——228
漢字コード表——229
コントロールコード表——238
メモリマップ——239
エラーメッセージ表——241

第5章　付　録

カラーコードについて——247
COLOR SPRITE文補足説明——248
VRAMについて——250
スーパーインポーズとディジタイズ——252
索引——254

INDEX

1 文法編

2 リファレンス編

3 サンプルプログラム

4 資料編

5 付録

第1章

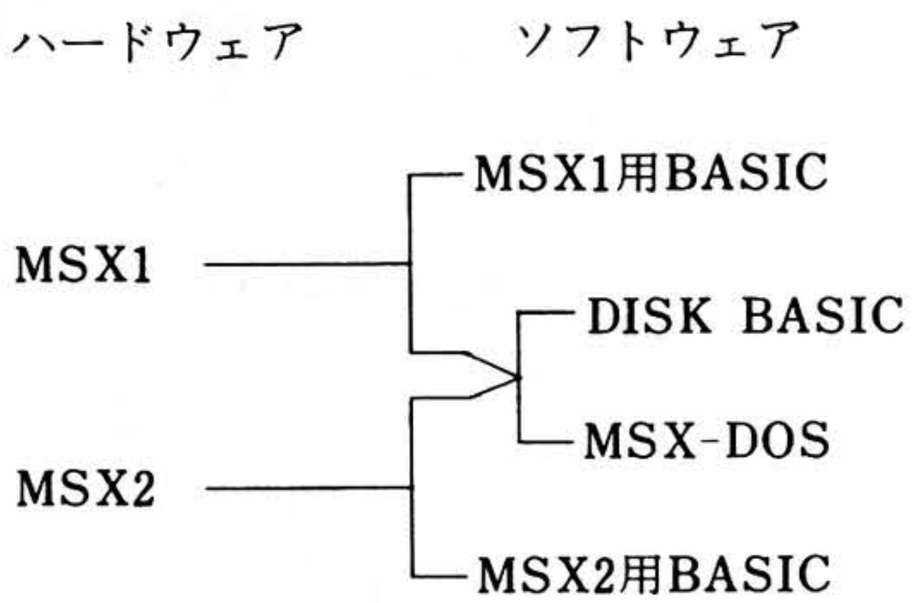
文法編

1

1.1 MSX BASICとは

人間どうしが会話するのに日本語，英語などの言語が必要なように，人間とコンピュータが会話するのにも言語が必要です．BASICは，この人間とコンピュータが会語するための言語の1つで，初心者でも理解しやすく使いやすい言語であるため，ほとんどのパーソナルコンピュータに採用されています．日本語にも方言があるように，BASICにもパーソナルコンピュータの機種により微妙に異なる部分があり，多くの種類があります．MSX BASICはMSX用に作られたBASICで，標準BASICに対して，グラフィック機能，音楽機能などが強化されています．

MSXに関連するソフトウェア体系を下図に示します．



MSX2用BASICは，MSX1用BASICをベースとしてバージョンアップが計られたものです．ソフトウェアは，アップワードコンパチブルになっていますので，MSX1用のソフトはMSX2でも動作させることができます．ただし，MSX2用のソフトは，一般的にはMSX1では使用できません．

本書では，MSX2用BASICの各命令やサンプルプログラムについて説明しています．DISK BASICやMSX DOSにつきましては，別冊(またはフロッピーディスク装置に付属)の「DISK BASIC／MSX DOS説明書」を参照してください．

1.2 動作モード

コンピュータに命令を実行させるのに2つのモードがあります．1つは命令を入力して RETURN キー（以後↵キーと表わす）を押すとすぐ実行する方式でダイレクトモードと呼ばれます．もう1つは，命令を入力して，↵キーを押しても，そのときは実行されずに，メモリに格納され，あとからまとめて実行される方式でプログラムモードと呼ばれます．

●ダイレクトモード

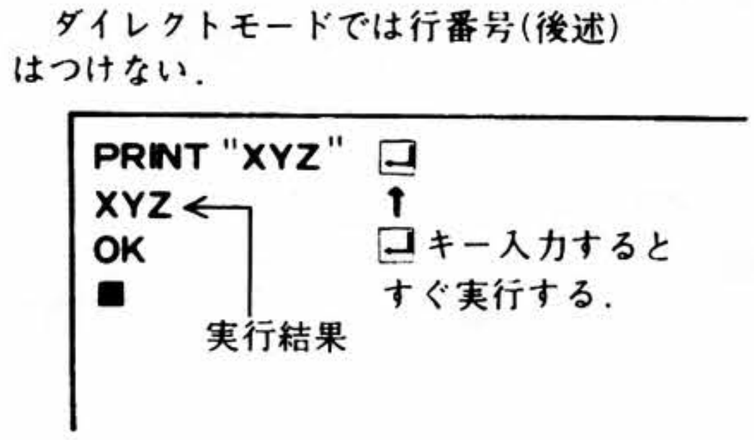
コンピュータに対する命令を↵キーを入力するとすぐに実行するモード

例）**RUN** ↵　メモリ上のプログラムの実行を開始せよ！

LIST ↵　メモリ上のプログラムリストを表示せよ！

CLOAD "〈ファイル名〉" ↵　カセットテープレコーダよりプログラムをロードせよ！

行番号（後述）をつけずに，いきなり命令を入力すればダイレクトモードでの実行となります．



●プログラムモード

まとまった仕事をさせるときには，複数の命令に行番号をつけて1つのプログラムとし，RUNコマンドなどで実行させることができます．これをプログラムモードでの実行といいます．

1.3 文

文はBASICに対する命令です．文には第2章で説明するようにコマンドかステートメントあるいは代入を行う式を書きます．コマンドとステートメントは明確に分かれている訳ではありませんが，ここでは次のように説明しておきます．

●コマンド

ダイレクトモードでの使用を主目的とした命令です．

例）**AUTO, CLOAD, CSAVE, CONT, LIST, RUN, DELETE・・・・・**

●ステートメント

ステートメントは，目的とする仕事つまり最終的にコンピュータにやってもらいたい仕事をします．

例) 計算をする，音楽を演奏する．画面に線を描く・・・・・・

1つのステートメントの機能はごく単純ですから，まとまった仕事をさせるときには，いくつか集めてプログラムを作り，プログラムモードで実行します．また，プログラムが正しく働いているかをチェックするためにダイレクトモードでも実行することができます．

1.4 行

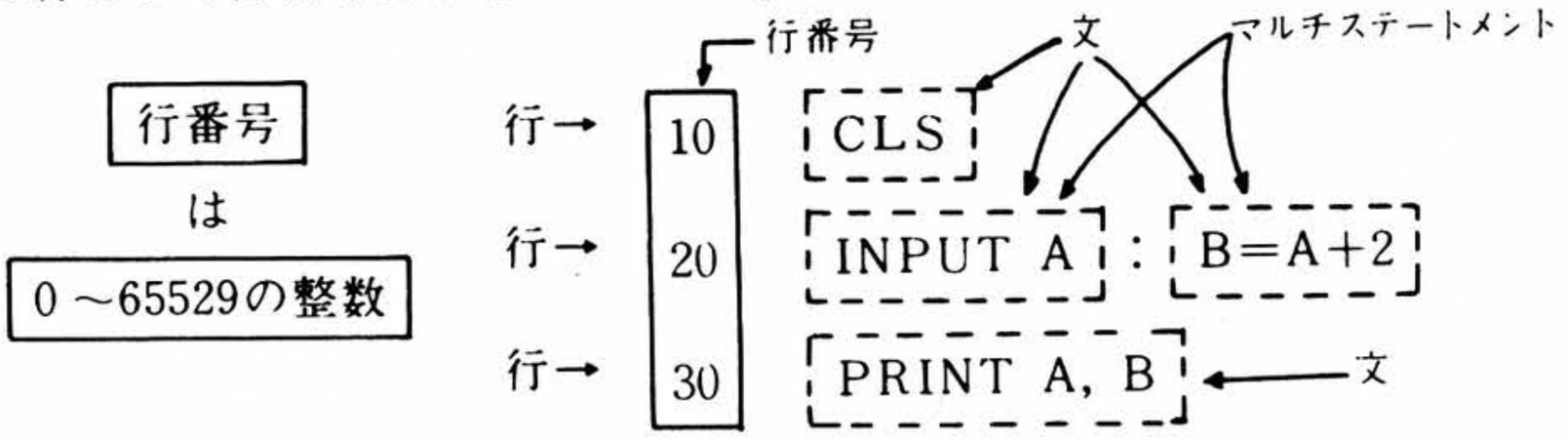
行は，BASICに対して命令を入力するときの単位で，キーインを開始して↵キーを押すまでを指します．

行の条件　　1行は最大255文字以内
　　1つ以上の文を含む

マルチステートメント(複文)　　1行に複数の文を含む場合をいいます．
この場合は文と文の間をコロン(:)で区切ります．

1.5 行番号

行の先頭につけた番号です．行番号をつけた行を入力すると，この行はプログラムを構成する行として行番号とともにメモリに格納されます．



行番号の役目

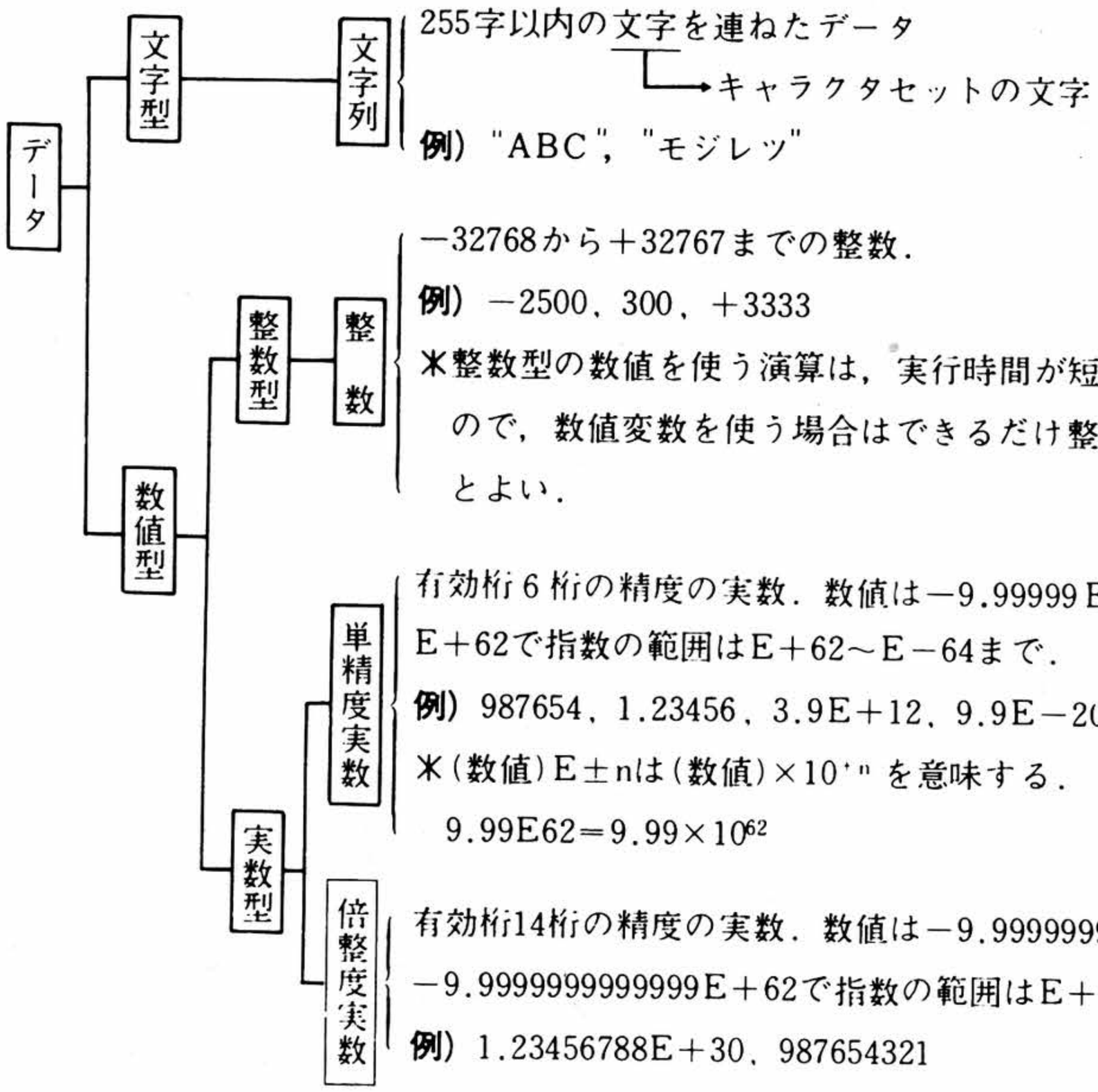
- メモリ上に格納される順序を決める．(通常は行番号の小さいものから順に実行されます．)
- 分岐命令 (GOTO命令など) で指定する分岐先
- プログラム編集コマンド(LIST, DELETEなど)における行の指定

1.6 キャラクタセット

MSX BASICで使用できる文字は英文字(大文字，小文字)，数字，カタカナ，ひらがな，特殊記号そしてグラフィックキャラクタから構成されておりこれらをキャラクタセットと呼んでいます．キャラクタセットの詳細は，資料「キャラクタコード表」を参照してください．

1.7 BASICで扱うデータ

BASICで扱うデータは次のように分類することができます．



＊MSX BASICでは，実数を使った演算はすべて倍精度実数に変換して行われ，結果も倍精度実数で得られますが，式中に単精度実数を入れたり，結果を単精度実数で受け取ることもできます．

1.8 定数

1.8.1 文字定数

文字定数は文字列を引用符（"）で囲んで表します．文字列は255字以内のキャラクタセットの文字で構成されます．

例) "THIS IS A STRING", "123456"

＊• 数字も引用符で囲むと文字列とみなされます．

• 文字列の中に引用符を入れたい場合は，CHR$関数を用います．

例) "インヨウフ"+CHR$(34)+"フクム"

引用符(")の文字コード

1.8.2 整数型定数

整数型定数は基数を10進，8進，16進，2進で表記することができます．

●10進形式

−32768～+32767までの整数．正の数値の場合，数値の前の＋記号はつけなくてもかまいません．

● 8進形式

先頭に&Oをつけた0～7までの数字の並び．範囲は&O0～&O177777です．

●16進形式

先頭に&Hをつけた0～9，A，B，C，D，E，Fまでの並び．範囲は&H0～&HFFFFです．

● 2進形式

先頭に&Bをつけた0と1の並び．範囲は&B0～&B1111111111111111です．

＜各形式間の対応＞（&O，&H，&Bは省略しています．）

10進形式	8進形式	16進形式	2進形式
0～7	0～7	0～7	0～111
8	10	8	1000
15	17	F	1111
16	20	10	10000
100	144	64	1100100
32767	77777	7FFF	111111111111111
−1	177777	FFFF	1111111111111111
−2	177776	FFFE	1111111111111110
−32768	100000	8000	1000000000000000

＊・変換例

2進形式	&B	1111111111111110
8進形式	&O	177776
16進形式	&H	FFFE
10進形式		−2

- 2進形式16桁で表わした数値の最上桁(16桁目)がサインビットに相当します．ここが1の時は負の数値を，0の時は正の数値を表わします．
- 8進形式，16進形式，2進形式で入力された数値でもPRINT文などで出力すると10進形式で表示されます．

1．8．3　単精度実数型定数

次のいずれかで表わします．

- 有効数字6桁以内の実数
- 有効数字6桁以内の実数とEを使った指数表示の組合せ．Eの範囲はE−64からE＋62までです．
- 定数の後に！をつけた数値．有効数字が6桁をこえていると，7桁目が四捨五入されます．

例) 7895.925！→ 7895.93！

1．8．4　倍精度実数型定数

次のいずれかで表わします．

- 有効数字14桁以内の実数
- 有効数字14桁以内の実数とEを使った指数表示の組合せ．Eの範囲はE−64からE＋62までです．
- 定数の後に＃をつけた数値．有効数字が14桁をこえていると，15桁目が四捨五入されます．

1.9　変数

変数はプログラム中で使う文字列や数値をしまっておくことができる入れ物（メモリ上のエリア)で，英数字から成る名前(変数名)がついています．
変数の値(内容)は代入文を実行することによって(例　A＝5）セットされます．変数に値を代入する前に参照すると，数値変数では0，文字変数はヌルストリング(空の文字列)となっています．

1．9．1　変数名

• 2文字までの英数字．ただし先頭の1文字は英字．

3文字以上の場合最初の2文字までしか判別しません．

また変数名の中に含まれるかな文字，グラフィック文字，空白は無視されます．

例) ABCD → 変数名はABとなる．

Aア1 → 変数名はA1となる．

ab → 変数名はABとなる．

＊変数名は小文字で入力しても大文字に変換されます．

• 変数名は予約語(BASICであらかじめ使っているキーワード)と同じであったり，予約語を含んではいけません．資料「予約語表」参照

例) CO, COT → 変数として使える．

COS, COST → 変数として使えない．(予約語COSを含んでいる)

1．9．2　型宣言文字

変数にも，格納するデータの型に応じた型があります．変数の型は，変数名の最後に型宣言文字を付けることによって区別します．型宣言文字の種類は次の通りです．

%……整数型　　**例)** A%＝123

!……単精度実数型　　A!＝1.23

#……倍精度実数型　　A#＝A＝1.2345678E−11

$……文字型　　A$＝"MSX"

なお，型宣言文字をつけていない変数は，倍精度実数型とみなされます．型宣言の方法として，プログラム中でDEFINT, DEFSNG, DEFDBL, DEFSTRのステートメント (第2章の各ステートメントの説明参照)で指定する方法もありますが，型宣言文字で指定した場合の方が優先します．

1．9．3　配列変数

配列変数とはいくつかの要素から構成される変数で，各要素は，添字により区別されています．したがって，プログラムが配列を調べて，特定の要素にデータを代入したり，データを取り出すことができます．

例) 表に示したようなデータを処理する場合

普及状況(単位%)〔(　)内は各要素の添数〕

品名＼年度	Y0	Y1	Y2	Y3
カラーテレビ	93.1 (0, 0)	95.0 (0, 1)	98.5 (0, 2)	99.2 (0, 3)
乗　用　車	48.5 (1, 0)	52.3 (1, 1)	56.1 (1, 2)	64.8 (1, 3)
オートバイ	21.6 (2, 0)	22.7 (2, 1)	28.3 (2, 2)	33.5 (2, 3)

• カラーテレビだけの処理

——1次元配列変数

```
AA(J)
  ↑  ↑
  |  └── データNo.(0～3)
  └── 配列名
```

• 表全体を処理

——2次元配列変数

```
HR(I, J)
  ↑   ↑  ↑
  |   |  └── データNo(0～3)
  |   └── データNo.(0～2)
  └── 配列名
```

HR(2, 1)の値は22.7

●配列変数の次元と要素の数

配列変数の次元は最大255次元まで(実際には，1行に記述できる文字数の関係で.もっと少なくなります.)，また要素の数はメモリが不足しない範囲内まで許されます.

●配列変数の宣言（DIM文）

配列変数は，配列の大きさをDIM文であらかじめ宣言しなければなりません．ただし，3次元までの配列変数については，DIM文を省略して配列変数を使おうとすると，自動的に各次元の添字の最大値を10として設定します.

添字は0から始まる整数なので，配列変数が含む各次元の要素数はDIM文で指定する各次元の添字の最大値＋1となります．プログラム中ではDIM文で宣言した添字の最大値を越えるような添字の値は使用することはできません.

例) DIM A(10)　……1次元配列変数Aを宣言
　　要素数は0～10までの11個

DIM HR(3,4)……2次元配列変数HRを宣言
　　要素数は(3＋1)×(4＋1)＝20個

1．9．4　変数と使用メモリ

変数は次に示すとおりのメモリを使用します．各変数の領域がメモリ上に確保されるのは変数に値を代入した時点，あるいはDIM文で配列変数を宣言した時です.

●変数

整数型	5バイト
単精度実数型	7バイト
倍精度実数型	11バイト
文字型	6＋(文字列の文字数)バイト

●配列変数

整数型	5＋2×(要素数)＋2×(次元数)＋1バイト
単精度実数型	5＋4×(要素数)＋2×(次元数)＋1バイト
倍精度実数型	5＋8×(要素数)＋2×(次元数)＋1バイト
文字型	5＋3×(要素数)＋2×(次元数)＋1＋(各要素に含まれる文字列の総文字数)バイト

例) DIM (3, 4, 5) で定義された配列変数の使用メモリ

整数型	5＋2×(4×5×6)＋2×3＋1＝252バイト
単精度実数型	5＋4×(4×5×6)＋2×3＋1＝492バイト
倍精度実数型	5＋8×(4×5×6)＋2×3＋1＝972バイト

1.9.5 システム変数

MSX BASICではBASIC自身で持っているシステム変数があります.

TIME インターバルタイマ. $\frac{1}{60}$秒ごとに値に1が加算されます. この変数には0～65535の値を代入することもできます.

ERL エラーが発生したとき, エラーが発生した行番号をもっています. この変数に値を代入することはできません.

ERR エラーが発生したとき, エラーの原因を表すエラーコードをもっています. この変数に値を代入することはできません.

BASE(n) 画面出力に関連するテーブルのアドレスを持っています.

SPRITE$(n) スプライトパターン.

VDP	VDP レジスタ.
CSRLIN	カーソルの垂直方向の位置を持っています.
POS	カーソル の水平方向の位置を持っています.
LPOS	プリンタヘッドの水平方向の位置を持っています.

1. 10　型変換

数値データは必要に応じてデータの型を他の型に変換することができます．ただし，文字型と数値型の間でこの交換を行うことはできません．型の変換は次の規則に従って行われます．

- **数値データを異なる型の数値変数に代入する場合，数値は代入先の変数の型に変換されます．**

  例）

```
10  ABC％＝1.234
20  PRINT ABC％
RUN
 1
```

- **精度の異なる数値を使った演算では，精度の低い方の数値が精度の高い方の型に変換されて演算が行われます．**

  例）

```
10  D＝6／7！          10  D！＝6／7
20  PRINT D            20  PRINT D！
RUN                    RUN
    857142857142 86        857143
```

- **論理演算の場合，扱われる数値はすべて整数型に変換され，結果は整数で得られます．**

  例）

```
10  A#＝12.34          整数に変換
20  B＝NOT  A#  ──────────→ 12＝&H000C
30  PRINT  B, A#                ↓反転
                          &HFFF3＝－13
－13            12.34
```

● 実数が整数に変換される場合は，小数点以下は切り捨てられます．このとき，切り捨てた結果が整数型で扱える範囲（32767～－32768）を越えていればエラーとなります．

例）

```
10　A％＝34.4                    10　A#＝1.234E＋07
20　B％＝34.5                    20　B％＝A#
30　PRINT　A％, B％              30　PRINT　B％, A#
RUN                              RUN
34　　34                         Overflow in 20
```

● 倍精度実数型変数を単精度実数型変数に代入すると，変数の値は有効数字６桁に丸めたものになります．単精度実数型の精度は６桁であり，もとの倍精度実数型の７桁目以降は四捨五入されます．

例）

```
10　A#＝1.23456789#
20　B！＝A#
30　PRINT　A#, B!
RUN
　1.23456789　　1.23457
```

1.11　式と演算

式とは定数や変数を演算子で結合した一般的な数式や，単一の定数や変数だけのものをいいます．

例）

```
"BASIC"
3.14
10＋３／５
A＋B／C－D
TAN（DO）
```

BASICの演算は次の５つに分類できます．

- 算術演算
- 関係演算
- 論理演算
- 関数
- 文字列演算

１．１１．１　算術演算

算術演算子には次のようなものがあります．

	演算子	演算	例
実行順	↑	指数演算	X↑Y
	−	負号	−X
	*，／	乗算，実数の除算	X＊Y，X／Y
	＋，−	加算，減算	X＋Y，X−Y

演算の順序を変更する場合，カッコ（　）を使用します．カッコの中の演算子は他の演算子より先に実行されます．また，カッコ内の式は通常の実行順序に従います．

次に式の例を示します．

代数表記	ＢＡＳＩＣでの表記
$2X+Y$	2＊X＋Y
$\frac{X}{Y}+2$	X／Y＋2
$\frac{X+Y}{2}$	(X＋Y) ／2
X^2+2X+1	X↑2＋2＊X＋1
X^{Y^2}	X↑(Y↑2)
$(X^Y)^2$	X↑Y↑2
Y（−X）	Y＊−X

- **整数の除算と剰余の計算**

整数の除算は／の代わりに¥によって行い，扱う数値が実数の場合は，演算が実行される前に小数点以下が切り捨てられます．商は小数点以下が切り捨てられた整数となります．

例）

```
PRINT 10¥3
 3                    ……（10／3＝3…余り1の演算が行われた）
PRINT 10¥2.5
 5                    ……（10／2＝5…余り0の演算が行われた）
```

剰余の計算はＭＯＤによって行います．扱う数値が実数の場合は，演算が実行さ

れる前に小数点以下が切り捨てられます．結果は整数の割り算の余りです．

例）

```
PRINT   13.3   MOD   4
   1                          …… (13／ 4 ＝ 3 … 余り1の演算が行われた)
PRINT   25.68   MOD   6.99
   1                          …… (25／ 6 ＝ 4 … 余り1の演算が行われた)
```

＊MOD，￥で扱える数値は，－32768～＋32767です．

- **0での除算**

  数値を0で除算した場合は，"Division by zero" というエラーメッセージが出力され，プログラムを中断してコマンドレベルに戻ります．

  また，0に対して負のべき乗を行った場合も同様になります．

- **桁あふれ（オーバーフロー）**

  代入や演算の結果がその変数の型内で表現することのできる範囲をこえた場合，桁あふれが発生します．桁あふれが起こった場合，"Overflow" というエラーメッセージが出力され，プログラムを中断してコマンドレベルに戻ります．

１．１１．２　関係演算

関係演算子は２つの数値あるいは２つの文字列を比較するときに用います．関係演算の結果は，真(－１)，偽(０)で得られ，条件判定文などプログラムの流れを変えるのに用いられます．

関係演算子	内容	例
＝	等しい	X＝Y
<>，><	等しくない	X<>Y，X><Y
<	小さい	X<Y
>	大きい	X>Y
<=，=<	小さいか等しい	X<=Y，X=<Y
>=，=>	大きいか等しい	X>=Y，X=>Y

詳しくは第２章のＩＦ文の項を参照してください．

ＩＦ文の中での使い方の例を次に示します．

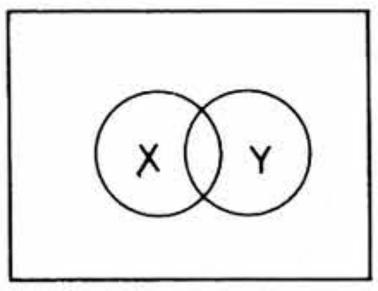
```
ＩＦ　Ｘ＝０　ＴＨＥＮ　１０００
ＩＦ　Ａ＋Ｂ<>０　ＴＨＥＮ　Ｘ＝Ｘ＋１：Ｙ＝Ｙ＋１
```

１．１１．３　論理演算

論理演算子は複数の条件を調べたり，ビット操作やブール演算を行ったりするのに用います．論理演算の結果は，ビットごとに０または１で得られます．

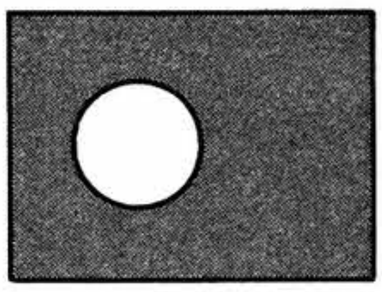
各論理演算の内容を次に示します．

入力条件	X	0	1	0	1
	Y	0	0	1	1
否　定　NOT X		1	0	1	0
論理積　X AND Y		0	0	0	1
論理和　X OR Y		0	1	1	1
排他的論理和　X XOR Y		0	1	1	0
包　含　X IMP Y		1	0	1	1
同　値　X EQV Y		1	0	0	1

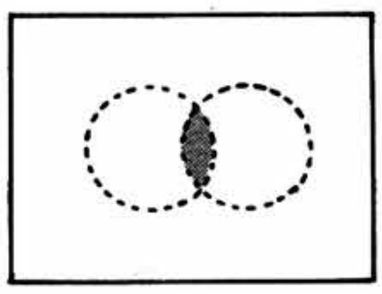


X．Yが右図の◯のような関係にある時，各論理演算との関係を下図に示します．

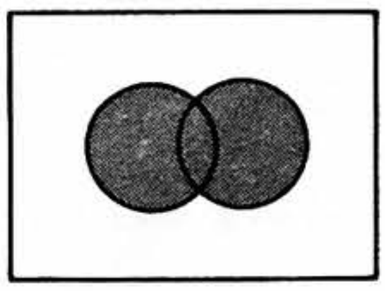
NOT X



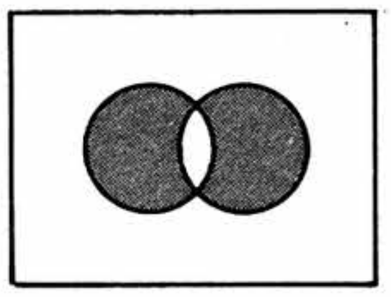
X AND Y



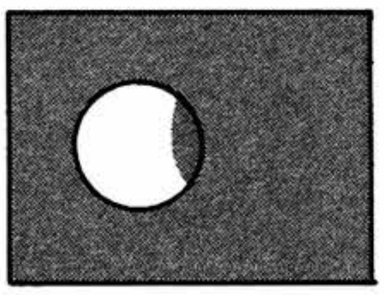
X OR Y



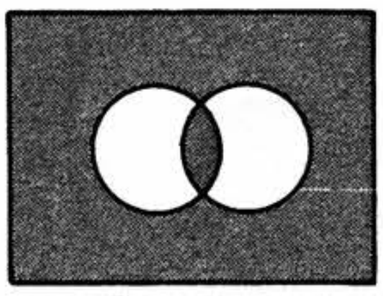
X XOR Y



X IMP Y



X EQV Y



条件判断文（ＩＦ文）では論理演算子を使って，複数の条件を判断することができます．次にその例を掲げますが，詳しくは第２章のＩＦ文を参照してください．

例）

IF X< 0 OR 99<X THEN 1000

Xが負または，99より大きければ行番号1000へジャンプする．

IF 0 <X AND X<100 THEN X= 0

Xが正でかつ，100より小さければXに０を代入する．

IF NOT（A＝0）THEN 20

Aが０でなければ行番号20へジャンプする．

注意：論理演算では演算の前に，扱う数値を－32768から＋32767までの２の補数表示の整数に変換します．もし，この範囲外となれば "Overflow" エラーとなります．論理演算では，この整数に対しビットごとに演算を行います．したがって，論理演算を使ってバイトデータをあるビットパターンに照らし合わせて調べることができます．

たとえば，ＡＮＤ演算子は機器のＩ／Ｏポートのステータスバイトの必要なビット以外のすべてのビットをマスクすることに使えます．またＯＲ演算子はある２つのビットパターンを混合させることができます．

以下の例は論理演算子がどのように働くかの例です．

＆Ｂ111001ＡＮＤ＆Ｂ1100＝８

＆Ｂ111001 ……0000000000111001

ＡＮＤ）＆Ｂ1100 ……0000000000001100

0000000000001000……結果は８

－１ＡＮＤ８＝８

－１を２進数で表すと ……1111111111111111

ＡＮＤ）８を２進数で表すと……0000000000001000

0000000000001000……結果は８

&B1100ＯＲ&B1011＝15

```
        &B1100  ……0000000000001100
ＯＲ  ) &B1011  ……0000000000001011
──────────────────────────────────
                0000000000001111……結果は15
```

32767ＯＲ －32768＝－1

```
        32767を2進数で表すと  ……0111111111111111
ＯＲ  ) －32768を2進数で表すと……1000000000000000
──────────────────────────────────
                1111111111111111……結果は－1
```

&B1100ＸＯＲ&B1011＝7

```
        &B1100  ……0000000000001100
ＸＯＲ  ) &B1011  ……0000000000001011
──────────────────────────────────
                0000000000000111……結果は7
```

10ＸＯＲ10＝0

```
        10を2進数で表すと……0000000000001010
ＸＯＲ  ) 10を2進数で表すと……0000000000001010
──────────────────────────────────
                0000000000000000…結果は0
```

１．１１．４　関数

関数とは，与えられた引数に対して，ある決まった演算を行うものです。

ＢＡＳＩＣでは"組込み関数"としてＳＩＮ（正弦），ＳＱＲ（平方根）などの数値関数やＣＨＲ＄，ＬＥＦＴ＄などの文字列関数が用意されています．これらの関数については第２章で詳しく説明します．

また，ＢＡＳＩＣは"ユーザー定義関数"としてユーザーが自由に定義できる関数機能ももっています．これは第２章の"ＤＥＦ　ＦＮ"の項で説明します．

数値関数のうち引数を実数で取り得るもの（ＳＩＮ関数など）は，引数を整数や単精度実数で指定することはできますが，演算は倍精度に変換して行われ，結果は倍精度で得られます．

一般に引数に整数しか取り得ないものは，小数部分を切り捨てて整数に丸めてから演算が行われます．

次に関数の使い方の例を示します．

例)

```
A=SIN (3.14)+COS(3.14)
PRINT 2, 2 * 2, SQR ( 2 )
```

1.12 文字列の連結

１．１２．１　文字列の連結

文字列は演算子＋を使って連結することができます。

例）

```
10  A$＝"モジレツ"：B$＝"ヲ"：C$＝"レンケツシマス"
20  D$＝A$＋"　"＋B$＋"　"＋C$
30  PRINT D$
RUN
モジレツ　ヲ　レンケツシマス
```

１．１２．２　文字列の比較

文字列も数値の比較に用いられるものと全く同じ関係演算子を用いて比較することができます.

＝, <, >, <>, ><, <＝, ＝<, >＝, ＝>

文字列の場合，それぞれの文字列の最初から１文字ずつ文字の比較を行います．もし相互に全く同じ文字列の場合は，その２つの文字列は等しくなりますが，１個所でも違った場合は，その文字のキャラクタコード（資料のキャラクタコード表参照）の大きい方の文字列が大きくなります．文字列の片方が短くて比較が途中で終わった場合は，短い文字列の方が小さくなります．

文字列の比較においては空白なども意味をもちますから注意してください.

例）

```
   "AA" < "AB"
"BASIC" = "BASIC"
   "X&" > "X#"
"PEN  " > "PEN "
   "cm" > "CM"
"DESK" < "DESKS"
```

このように，文字列の比較は文字列の内容を調べたり，文字をアルファベット順に並べたり（ソート）することに使うことができます．なお，文字列を使った比較については第２章のＩＦ文でも詳しく説明しています.

1.13　演算の優先順位

演算は次の順位によって行われます.

1. カッコで囲まれたもの
2. 関数
3. 指数（べき乗）
4. 負号（−）
5. ＊，／
6. ¥
7. MOD
8. ＋，−
9. 関係演算子（<，>，＝など）
10. NOT
11. AND
12. OR
13. XOR
14. IMP
15. EQV

例）

A＊B↑2/C+D OR E AND F MOD G＊H

は下記のような演算を実行します.

$\{[(A*B^2)/C]+D\}$ OR $\{E\ AND[F\ MOD(G*H)]\}$

1.14　エラーメッセージ

プログラムの実行を中断しなければならないようなエラーが実行時に発生した時，エラーメッセージが画面に表示され，コマンドレベルに戻ります.

ダイレクトモードのエラーメッセージの書式は，

XX……

プログラムモードでの書式は，

XX…… in　yyyyy

XX……はエラーメッセージで，yyyyyはエラーが検出された行番号です.エラーメッセージの内容については，資料の「エラーメッセージ表」を参照してください.

1.15 画面モード

MSX BASICで扱う画面には，大きく分けてテキスト画面とグラフィック画面とがあります．テキスト画面ではリストやメッセージなど文字を表示することができますが，点や線を描くことはできません．逆にグラフィック画面では，点や線を描くことができますが，リストやメッセージを表示することは得意ではありません．

MSX2 BASICには，9つの画面モードがありSCREEN文を使って選択することができます．各モードの表示仕様を下表に示します．

SCREEN No.	モード	表示桁数または解像度	文字サイズ	色数 ※1)	色指定	スプライト
0	テキストモード	80桁×24行	6×8	16パレット中の2色	前景色 背景色 周辺色	不可
1		32桁×24行	8×8			可 (4) ※3)
2	高解像度グラフィックモード	256×192ドット	——	16パレット	8ドットを1単位とする領域ごとに2色	
3	マルチカラーモード	※2) 64×48ブロック	——		ブロック毎に色指定	
4	高解像度グラフィックモード	256×192ドット	——		8ドットを1単位とする領域ごとに2色	可 (8) ※3)
5	ビットマップグラフィックモード	256×212ドット	——		1ドット毎に色指定	
6		512×212ドット	——	4パレット		
7		512×212ドット	——	16パレット		
8		256×212ドット	——	256色(固定)		

※1) パレットは512色中よりパレット毎に色指定することができる

※2) 1ブロック＝4×4ドット

※3) ( )内は横方向の同じラインで同時に表示可能なスプライトの数

一般のテレビをディスプレイとして使用した場合は，コンピュータの表示画面とテレビの表示領域との相性により，表示画面の左右上下がテレビの表示画面よりはみ出すことがあります．このような場合は，SET ADJUST文により表示画面が，テレビの表示領域の中央になるように設定しご使用ください．それでもなおはみ出す場合には，WIDTH文で表示桁数を1～2桁，小さい値に設定しなおしたり，グラフィック画面の画面のはみ出しに相当する部分には，表示されなくても構わないようなものの表示だけにするようプログラム作成時に考慮する，などの配慮をしてご使用ください．

1.15.1　テキストモード

文字を表示するための画面です.

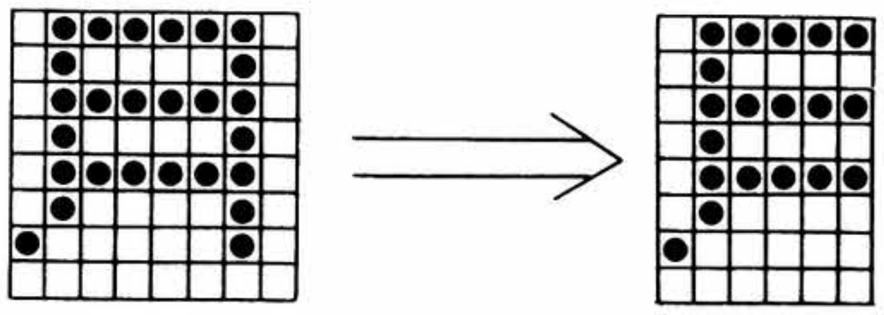
●SCREEN 0 (80桁×24行 表示モード)

画面に表示する文字の桁数はWIDTH文で指定することができますが，指定しなければ39桁になっています．ただし，SET SCREEN文により初期値39は，変更することも可能です．41～80桁を指定すると，表示画面の水平方向のドット密度が倍になるため，文字の幅が小さくなります.

ファンクションキーを表示した状態では画面の最下行を使うことはできません.（KEY OFF文を実行すると最下行も使用できます.）また，このモードではスプライト機能やグラフィック関係の命令を使うことはできません.

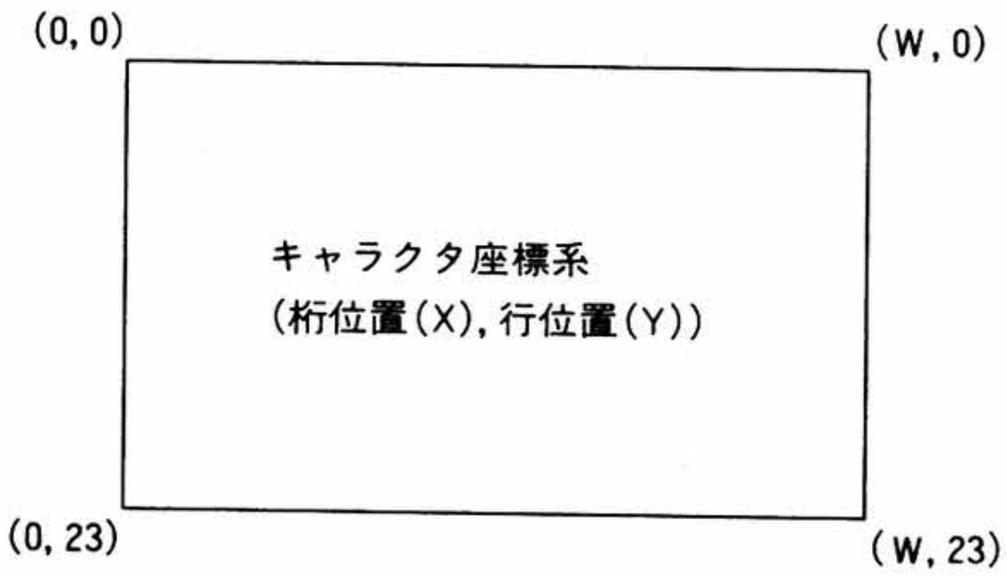
表示文字は，横8×縦8ドットで構成された文字パターンの右列2ドットをカットし，横6×縦8ドットで表示します．このため，ひらがなやグラフィックキャラクタは文字の右側が欠けて表示されるものがあります.

例



正規の文字パターン　　SCREENOでの表示パターン

文字の表示位置はLOCATE文により指定することができます．この場合のX, Yの座標と画面との関係を図に示します.
（WはWIDTH文で指定した表示桁数（最大79））



●SCREEN 1 （32桁×24行 表示モード）

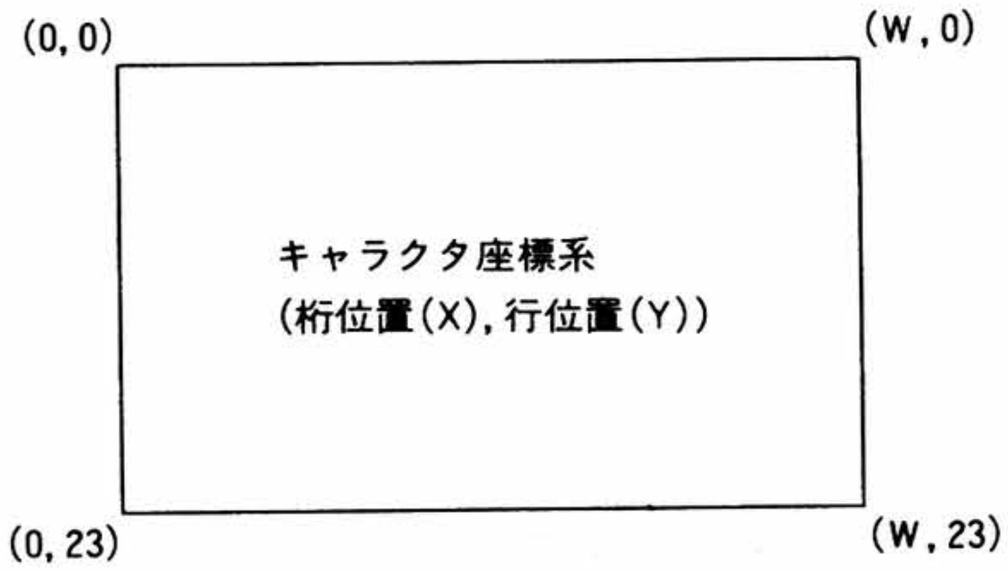
画面に表示する文字の桁数はWIDTH文で指定することができますが，指定しなければ29桁になっています．ただし，この値はSET SCREEN文により変更することも可能です．ファンクションキーを表示した状態では画面の最下行を使うことはできません．

このモードでは，スプライト機能は使えますが，グラフィック関係の命令は使うことはできません．

表示文字は横8×縦8ドットですので，SCREEN 0のモードのようにひらがなやグラフィックキャラクタの一部が欠けて表示されるようなことはありません．

文字の表示位置は，LOCATE文により指定することができます．この場合のX，Yの座標と画面との関係を図に示します．

（WはWIDTH文で指定した最大桁数（最大31））

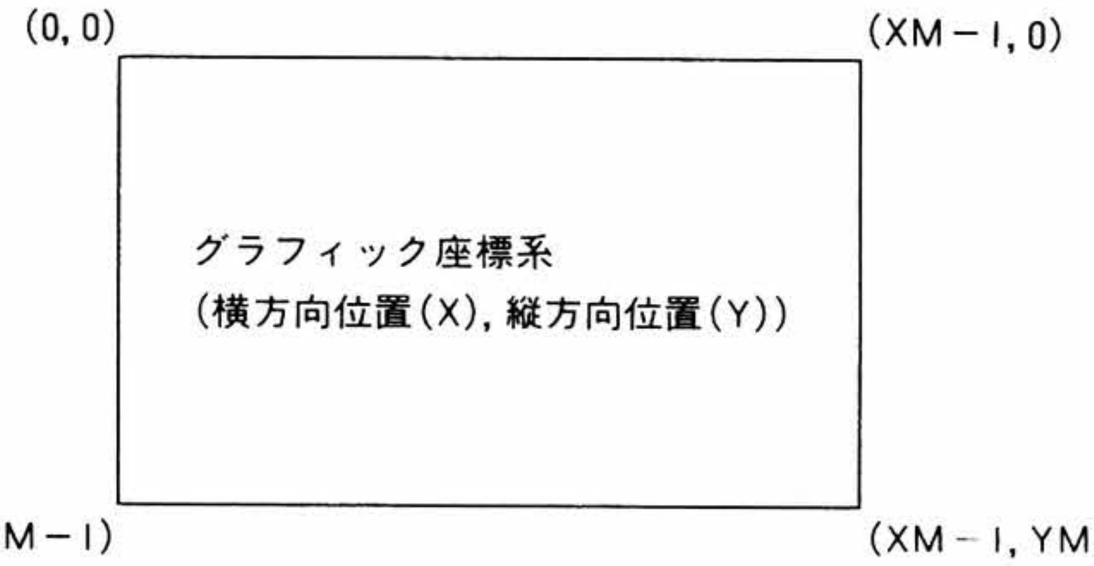


1.15.2　グラフィックモード

点や線などの絵を描くための画面です．解像度や機能により多くの種類がありますので，その目的に合ったモードを選んで使用してください．

グラフィック画面で文字を表示するには，OPEN文で〝GRP:〟のためのファイルをOPENした後，PRINT＃文で表示することができます．文字を描き始める位置はPRESET文でPRINT＃文実行前に指定します．グラフィックモードでの文字表示は，文字の表示ドットの部分だけがグラフィック画面に描かれます．このため，前に描いた文字の上に別の文字を描くような場合には，旧い文字と新しく描いた文字とが重なったような表示になりますので，文字を描く前に現在表示されている文字をLINE文などで消した後に新しい文字を描いてください．

グラフィックの座標系は横方向の表示ドット数をXM，縦方向の表示ドット数をYMとするといずれも図のようになります．



グラフィックモードでは，プログラムの実行が終了した場合などBASICがコマンドレベルにもどると，画面モードは自動的にテキストモードにもどります．したがって，描かせたグラフィックパターンを連続して表示させたい場合には，プログラムをループさせてコマンドレベルにもどらないようにしなければなりません．

例）5000　GOTO　5000

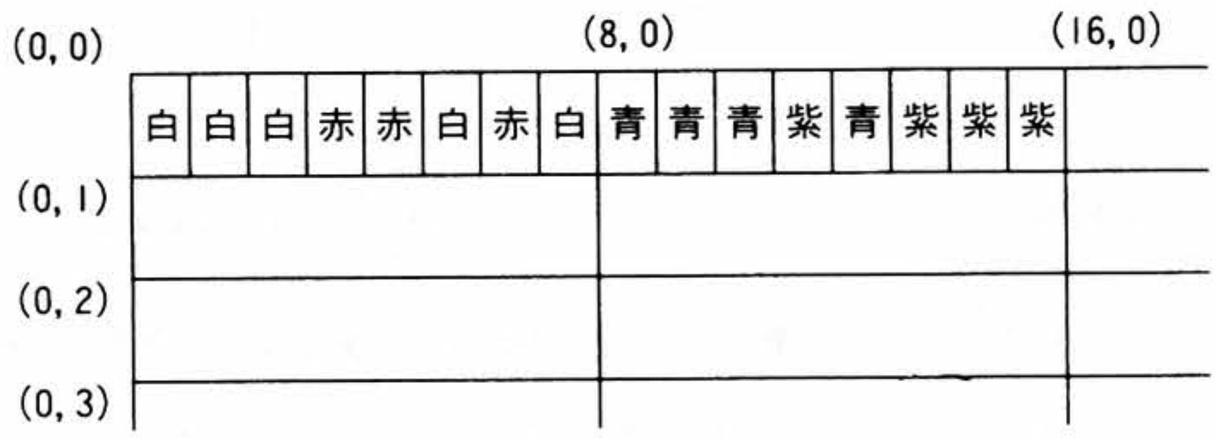
この場合，プログラムの実行を終了するには，CTRL＋STOPを押します．

●SCREEN 2 と 4 （256×192ドット高解像度グラフィックモード）

横256, 縦192ドットのグラフィック画面モードです. SCREEN 2 と 4 とでは, スプライトパターンを横方向の同じラインで同時に表示できる個数が異なります. SCREEN2は 4 個までですが, SCREEN4は 8 個まで表示することができます.

色は16個のパレットにより16色の使用が可能ですが, 下記のように 8 画素を単位とする領域ごとに 2 色という制約がありますので, MSXとの互換性を考慮される場合以外は, SCREEN5～8を使用するのがよいでしょう.

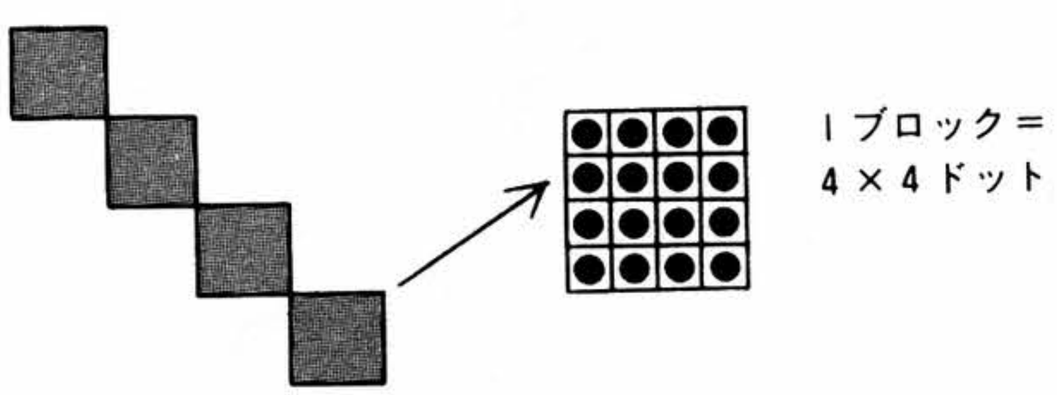
SCREEN 2 と 4 の画面モードでは, 画面を（0, 0）から右へ 8 画素ごとに分割して表示色を管理しています. 分割された 8 画素ごとに, 背景色を含めて 2 色まで使うことができます. たとえば, 次の例では（0, 0）～（7, 0）で赤と白を使い, （8, 0）～(15, 0）で青と紫を使っています. 分割された 8 画素の領域で, 3 色以上を指定すると, 表示が異常となるため, プログラム作成時, 3 色以上の指定にならないような配慮が必要です.



●SCREEN 3 （64×48 マルチカラーモード）

点を描く最小が 4 × 4 ドットのブロックを単位として横64×縦48ブロックの画面モードです. 1 ブロック毎に色指定が可能ですが, 細かい点や線を描くことはできませんので, 特殊な場合以外は使用しないでしょう.

座標系は, 横256, 縦192ドットとして指定します. 各々 4 で割り, 小数点以下を切捨てたものが, ブロックでの座標となります.



SCREEN 3 での表示例

●SCREEN 5 (256×212ドット ビットマップグラフィックモード)

横256，縦212ドットの画面モードです．1ドットの絵素ごとに色指定することができます．(16ケのパレットのうち1ケを指定) さらにVRAM上に複数の画面領域を持ち，必要に応じて画面領域を切換えて使用することができます．(SET PAGE文参照)

●SCREEN 6 (512×212ドット ビットマップグラフィックモード)

横512，縦212ドットの画面モードです．1ドットの絵素ごとに色指定（4ケのパレットのうち1ケを指定）することができますが，1つの画面で使用できる色の数は4色までです．さらに，VRAM上に複数の画面領域を持ち，必要に応じて画面領域を切換えて使用することができます．(SET PAGE文参照)

●SCREEN 7 (512×212ドット ビットマップグラフィックモード)

このモードはVRAMが64KBの装置では使用することはできません．

横512，縦212ドットの画面モードです．1ドットの絵素ごとに色指定（16ケのパレットのうち1ケを指定）することができます．1つの画面で使用できる色の数は16色です．さらに，VRAM上に2ケの画面領域を持ち，必要に応じて画面領域を切換えて使用することができます．(SET PAGE文参照)

●SCREEN 8 (256×212ドット ビットマップグラフィックモード)

このモードはVRAMが64KBの装置では使用することはできません．

横256，縦212ドットの画面モードです．1ドットの絵素ごとに256色中の1色を選んで色指定することができます．(パレット機能はありません.) さらにVRAM上に2ケの画面領域を持ち，必要に応じて画面領域を切換えて使用することができます．(SET PAGE文参照)

1.16 カラーコード

MSX2 BASICでは，パレット機能を使用することによりカラーコードへの色の割り当てを変えることができます．パレットとは，水彩画などに使用するパレットを思い浮かべてください．パレットには，他の絵のぐと混ざらないように仕切りを付けた複数の領域があります．この領域の1つに，例えば緑の絵の具を入れると，そこから取り出した色は緑色となり，赤の絵の具を入れた場合には，そこから取り出した色は赤色になります．MSX2の場合もこれと同様です．パレットの各領域に相当するものが，カラーコード0～15で16ケあります．各々のカラーコードには，COLOR＝命令によって，512色中の1色を割当てることができます．電源オン時やSCREEN命令を実行した場合には，カラーコードの色指定は初期化され次のようになっています．

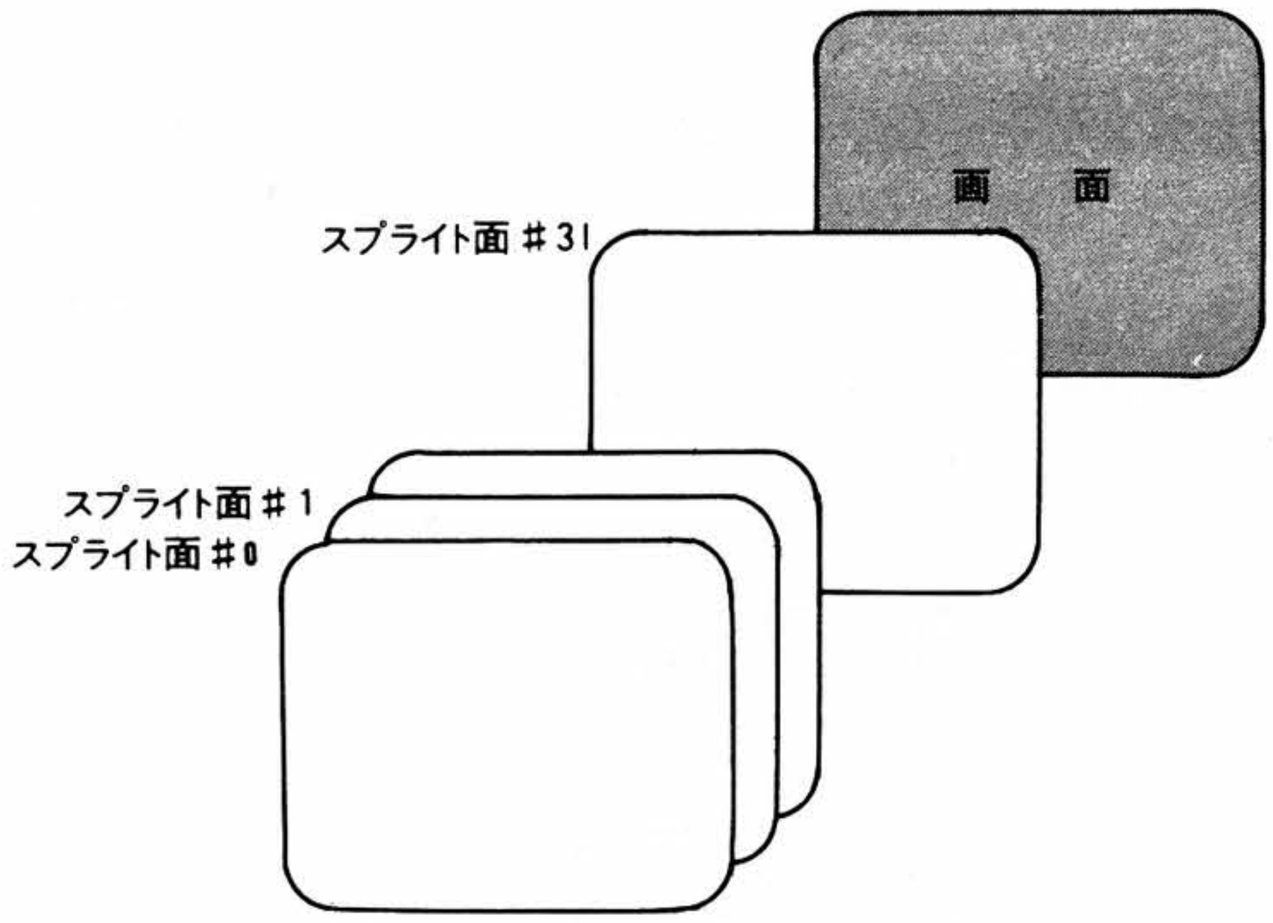
0：透明(周辺色と同じ色)　1：黒　2：緑
3：明るい緑　4：暗い青　5：明るい青
6：暗い赤　7：水色　8：赤
9：明るい赤　10：黄　11：明るい黄
12：暗い緑　13：紫　14：灰
15：白

SCREEN 6では，カラーコードは0～3の4つだけが使用可能です．

SCREEN 8だけはパレット機能はありません。0～255のカラーコードにより256色の色を個々に指定することができます．(COLOR文参照)

1.17　スプライト機能

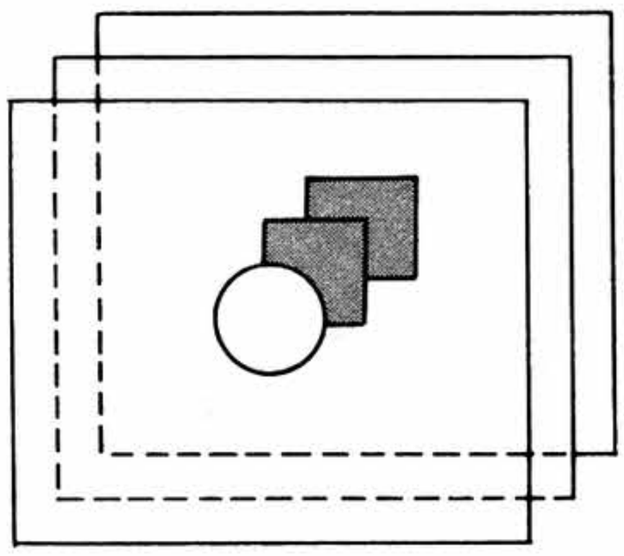
MSX BASICでは，１．15で説明している画面とは別にスプライト面というものがあります．スプライト面とは画面の前に重ねてあると考えることができる面のことで，０番から31番までの合計32枚が用意されています．このスプライト面にはプログラムで定義しておいたスプライトパターンをそれぞれ１つずつ表示することができます．



スプライトパターンの大きさは８×８画素，または16×16画素ですが，SCREEN 文で拡大モードを指定しておけば，縦横それぞれ２倍に拡大して表示することができます．

スプライトパターンは８×８画素ならば256個，16×16画素ならば64個まで定義しておくことができます．

スプライト機能の特徴は，スプライトパターンをスプライト面に表示すると，スプライトパターンが画面に描いてあった絵や文字の上に重なって見え，また，スプライトパターンを２つ以上表示したときには，小さい番号のスプライト面に表示したスプライトパターンが手前に見えることです．しかも，スプライトパターンが重なったことによりそれぞれの表示内容を損なうことはありませんので，表示画面を立体構造を持っているかのように扱うことができます．



なお,スプライトパターンは同時に32個まで表示することができますが，SCREEN 1～3では，水平方向に5つ以上並ぶと，前から5つ目以降のスプライトは表示されません．SCREEN 4～8では,水平方向に8つまでは表示可能です．

スプライト機能に関する命令は次のとおりです．詳しくは第2章を参照してください．

- スプライトパターンのサイズを定義する（SCREEN 文）
- スプライトパターンを定義する（SPRITE$関数）
- スプライト面にスプライトパターンを表示する（PUT SPRITE 文）
- スプライトパターンの衝突による割込み処理ルーチンを定義する（ON SPRITE GOSUB 文）
- スプライトパターンの衝突による割込みの可否を設定する（SPRITE ON/OFF/STOP 文）

1.18 ファイル

入力されたプログラムやデータはフィイルとして各種の記憶装置に保存し，必要に応じて読み出して再度実行することができます．また，MSXではグラフィック画面やプリンタへの出力もファイルという概念で扱うことができます．これらに関する命令は次の通りです．

入出力デバイス	入 出 力	入出力に使用できる命令
カセットテープ	出　　力	CSAVE,SAVE,BSAVE,PRINT#,PRINT#USIING
	入力※4）	CLOAD,LOAD,BLOAD,RUN,INPUT#
フロッピーディスク ※1）	出　　力	SAVE,BSAVE,PRINT#,PRINT#USING
	入力※4）	LOAD,BLOAD,RUN,INPUT#
RAMディスク ※2）	出　　力	SAVE,PRINT#
	入力※4）	LOAD,RUN,INPUT#
グラフィック画面　※3）	出　　力	PRINT#,PRINT#USING

※1）詳細は「DISK BASIC/MSX DOS説明書」を参照してください．

※2）RAMディスクとは&H7FFF以下の番地のメインメモリを利用したファイルのことです．BASICプログラムやデータを一時的に保存することができます．サポートする命令も用意されています.（CALL参照）

※3）グラフィック画面と同様にテキスト画面やプリンタにもPRINT#，PRINT# USING命令により出力可能です.（1.18.3デバイス名参照）

※4）この他にも,MERGE,LINE INPUT#,INPUT$も使用できます．

１．１８．１　ファイル番号

プログラム中でファィルへの入出力を行うときは，入出力に先立って，OPEN文でファイルの使用開始を宣言し，ファイルに対して番号を割当て，また，番号と対応する入出力用の領域を確保します．この番号をファイル番号と呼びます．ファイル番号はMAXFILES文で定義した数の範囲内の数を使用します．MAXFILES文での定義がなければファイル番号は１です．

例）
```
100  OPEN  "CAS : FILE" FOR INPUT AS # 1
  ⋮
110  INPUT  # 1, A$
  ⋮
150  CLOSE  # 1
```

１．１８．２　ファイルスペック

ファイルスペックは，ファイルの入出力の際に対象となる入出力機器やファイル名を指定します．ファイルスペックは次のような形式の文字列で表します．

"**〈デバイス名〉：＜ファイル名＞**"

ファイルスペックは，必ず引用符で囲みます．〈デバイス名〉，〈ファイル名〉は場合によって省略することができます．

１．１８．３　デバイス名

〈デバイス名〉は入出力機器を表すもので次のようなものがあります．

デバイス名	入出力デバイス	使用できるモート	
		INPUT	OUT PUT
CAS：	カセットテープ	○	○
A：～H：	フロッピーディスク	○	○
MEM：	RAMディスク	○	○
CRT：	テキスト画面	×	○
GRP：	グラフィック画面	×	○
LPT：	プリンダ	×	○

1.18.4 ファイル名

〈ファイル名〉とはプログラムファイルやデータファイルに付ける名前です．ファイル名はテキスト画面，グラフィック画面，プリンタへの出力の場合は省略することができます．

〈ファイル名〉はフロッピーディスクドライブとRAMディスクでは8文字，他は6文字以下の文字例です．〈ファイル名〉がこれより小さい場合は空いた部分に空白がうめられます．

〈ファイル名〉は，英数字，カタカナ，ひらがなが使用できます．

1.19 割込み

プログラムの実行中に，特別な事象が起きた時，現在実行中の処理を一時停止して，特別な事象に対応した処理を先に実行させることを割込みといいます．

たとえば，MSX BASICではファンクションキーが押された時に対応する割込み処理を実行するようにあらかじめ指定しておくと，プログラムの実行中にファンクションキーが押された時点で割込みがかかり，現在実行中の処理を停止して指定された処理を優先的に実行した後，停止していた処理が再開されます．

割込み処理とGOSUB文によるサブルーチンとの違いは，GOSUB文によるサブルーチンの実行はあらかじめプログラムで決めておいた手順通りに行われるのに対し，割込み処理が行われるのは特別な事象が発生した時点で，あらかじめ手順を決めておくことができないことです．つまり，プログラムがどの行を実行しているときに割込みがかかるのかは予測できません．

MSX BASICが用意している割込みは次のとおりです．また割込みの優先順序も次のとおりです．

ファンクションキー--割込み	(ON KEY GOSUB)
ストップキー割込み	(ON STOP GOSUB)
スプライトの衝突割込み	(ON SPRITE GOSUB)
ジョイスティックのトリガ割込み	(ON STRIG GOSUB)
インターバルタイマ割込み	(ON INTERVAL GOSUB)

第２章

リファレンス編

第2章の見方

●初心者の方は，命令名の右上に"米"を付した部分は，読みとばしてください．"米"の付いていない命令を理解された後で読まれた方が理解が容易です．

●機　能　命令の機能を簡単に示します．グラフィックモード専用またはテキストモード専用の命令は（　）で記しています．

●書　式　命令の書き方を示します．実際の入力時には次のような決まりに従ってください．

●英字の大文字で示された項目は，そのとおり入力します．入力する場合は，小文字でも大文字でもかまいません．ただし，引用符(")で囲まれた文字列(ファイル名など)は，大文字と小文字を区別してください．

●カギカッコ〈　〉で囲まれた項目は，ユーザーが指定します．

●角カッコ〔　〕で囲まれた項目は，オプションであり，省略することができます．省略した場合，省略値(BASICによって設定される値)または以前に指定した値が適用されます．

●カンマ(,)で区切られる複数のパラメータがあり省略可能な場合，次の例のような書式で示しています．

COLOR〔〈前景色〉〕〔,〈背景色〉〕〔,〈周辺色〉〕

この場合，角カッコで囲まれたパラメータは省略することが可能です．ここで，以後のパラメータをすべて省略する場合は，カンマも含めて省略しますが，後に続くパラメータを指定する場合は，それ以前のカンマはすべて指定する必要があります．

たとえば，上記の例で〈周辺色〉のみ指定する場合，次のように指定します．

COLOR ,,〈周辺色〉

なお，下記のようにパラメータの後にセミコロン（;）を指定するような場合は，セミコロンも含めて省略します．

INPUT〔〈"プロンプト文"〉;〕〈変数〉〔,〈変数〉・・・〕

ただし，この約束と異なっている場合もあります．その場合には，注意書きでその都度説明します．

● 上記のカギカッコ，角カッコ以外の記号でカッコ( )，カンマ(,)，セミコロン(;)，ハイフン(—)，等号(＝)などの記号は示された位置に正しく入力します．

● 省略記号 "・・・" の続く項目は，1行の許す長さ（255文字）の範囲内で任意の回数だけ繰り返すことができます．

例：　DATA〈定数〉〔〈定数〉……〕の場合

DATA　0, 10, 15, 100

● **文　例**　実際の入力の仕方の見本として簡単な例を示します．

● **解　説**　命令の使用法や詳しい機能とそれに関して注意しなければならない点などを説明します．

● **参　照**　関連する他の命令や項目を記しています．

● **サンプルプログラム**　いくつかの文を交えたプログラム例とその実行結果を示します．

ABS

アブソリュート(関数)

● **機　能**　**絶対値を得る．**

● **書　式**　**ABS（〈数式〉）**

● **文　例**　**B＝ABS（−2）**

実行結果　B＝2

● **解　説**　〈数式〉の絶対値を得ます．

〈数式〉はどのような型の数値でもかまいませんが，得られる値は倍精度です．

ASC

アスキー(関数)

● 機　能　文字のキャラクタコードを得る.

● 書　式　ASC (〈文字式〉)

● 文　例　A＝ASC ("B")

実行結果　A＝66 (66＝&H42)

● 解　説　〈文字式〉の最初の文字のキャラクタコードを得ます.〈文字式〉がヌルストリングであると "Illegal function call" エラーとなります. 引用符(")をデータとして扱うことはできません.

文字とキャラクタコードの対応については, キャラクタコード表を参照してください.

グラフィックキャラクタはすべて&H01となります.

● 参　照　CHR$, 資料「キャラクタコード表」

● サンプルプログラム

```
10 '*** アスキーコード ***
20 PRINT "キーを おしてください"
30 B$=INKEY$:IF B$="" THEN 30
40 PRINT B$;" の アスキーコードは";ASC(B$);"
=&h";HEX$(ASC(B$))
50 GOTO 30
```

ATN

アークタンジェット(関数)

● 機　能　逆正接（アークタンジェント）を得る.

● 書　式　ATN (〈数式〉)

● 文　例　A＝ATN (1)

● 解　説　〈数式〉の逆正接（アークタンジェント）を得ます. 結果の値の単位はラジアンです. 得られる値は, $-\pi/2$から$\pi/2$までの範囲です.

● 参　照　TAN, SIN, COS

● サンプルプログラム

```
10 '*** アークタンジェントの けいさん ***
20 PRINT "    X          ATN(X)"
30 FOR K=0 TO 5
40 F=.1*10^K
50 PRINT F;TAB(7);ATN(F)
60 NEXT K
```

AUTO

オート(コマンド)

● 書　式　AUTO〔〈先頭行番号〉〕〔,〈増分〉〕

● 文　例　AUTO 100, 5

● 解　説　AUTOコマンドは，プログラムを入力するのに便利なように指定された〈先頭行番号〉を画面に表示し，行の入力を待ちます．以後，リターンキーを押すごとに，〈増分〉ずつ増加した行番号が自動的に発生します．

〈先頭行番号〉を省略すると0，〈増分〉を省略すると10が省略値として選択され，また，〈先頭行番号〉と〈増分〉の両方を省略すると，それぞれ10が省略値として選択されます．

CTRL + C または，CTRL + STOP を押すと，AUTOコマンドはその動作を中止し，BASICのコマンドレベルに戻ります．このとき，最後に表示された行番号の行は格納されません．

プログラム中に既に存在する行と同じ行番号が発生した場合には，行番号の直後にアスタリスク（＊）が表示され，注意を促します．このとき，文字を入力してリターンキーを押すと，その行は新しい内容と入れ代わります．また，何も入力しないでリターンキーを押すと，その行は古い行の内容のままです．

AUTOのモード中でも，スクリーンエディット機能を利用することができます．カーソルを既に入力した行，あるいは表示されている行へ移動し，リターンキーを押した場合には，その行番号に増分を加えた行番号が次に発生される行番号となります．

BASE*

ベース(システム変数)

● 機　能　画面出力に関連するテーブルのアドレスを得る.

● 書　式　BASE (〈数式〉)

● 文　例　BASE (10)

● 解　説　画面出力に関連するテーブルのアドレスを得ます. 〈数式〉で指定できる数値は0～44です.
通常は使用することはありませんので, 詳細は省略します.

BEEP

ビープ(ステートメント)

● 機　能　スピーカを鳴らす.

● 書　式　BEEP

● 文　例　BEEP

● 解　説　BEEPはスピーカを約0.04秒鳴らします.
BEEP は CHR$（7）を実行したのと同じです.

● 参　照　SET BEEP

● サンプルプログラム

```
10 '*** BEEP おん ***
20 FOR A=1 TO 50
30 BEEP
40 FOR B=0 TO 1000/A
50 NEXT B,A
```

BIN$

バイナリーダラー(関数)

● 機　能　数値を2進表記の文字列に変換し，その結果を得る．

● 書　式　BIN$（〈数式〉）

● 文　例　PRINT BIN$ (A)

実行結果　A＝254の場合　　11111110が表示される

● 解　説　〈数式〉の値を2進表記の文字列に変換し，その結果を得ます．〈数式〉の値の範囲は－32768から65535までです．負数の場合は65536にその値を加えた値と同じです．

変換結果の文字例はゼロサプス（先頭の不用な0を取り除く）されています．

● 参　照　OCT$, HEX$

● サンプルプログラム

```
10 A=123:B=234
20 PRINT " A=";BIN$(A);
30 PRINT " B=";BIN$(B):PRINT
40 C=A AND B
50 PRINT "A AND B=";C;BIN$(C)
60 C=A OR  B
70 PRINT "A OR  B=";C;BIN$(C)
80 C=A XOR B
90 PRINT "A XOR B=";C;BIN$(C)
100 C=A EQV B
110 PRINT "A EQV B=";C;BIN$(C)
120 C=A IMP B
130 PRINT "A IMP B=";C;BIN$(C)
140 END
```

BLOAD*

ビーロード(コマンド)

● 機　能　機械語プログラムをロードする.

● 書　式　BLOAD　"〈ファイルスペック〉"〔, R|S〕〔, 〈オフセット〉〕

● 文　例　BLOAD　"CAS : DEMO", R

● 解　説　〈ファイルスペック〉で指定した機械語プログラムファイルをメモリ上にロードします. ファィル名を省略すると, カセットテープの現在の位置から最も近い機械語プログラムファィルがロードされます.

〈オフセット〉を省略すると機械語プログラムはBSAVEでセーブするときに指定した〈開始アドレス〉からロードされますが, 〈オフセット〉を指定した場合には, セーブの際に指定された開始アドレスに, 〈オフセット〉を加えた番地からロードします. したがって, 〈オフセット〉を付けてロードするプログラムは, セーブされたときと異なる番地にロードされても, 実行可能（リロケータブル）な性質をもっていなければなりません.

Rオプションを指定すると, プログラムをロード後, BSAVE で指定しておいた実行開始アドレスから, 直ちに実行されます. このとき, 既に開かれているファイルがあればファイルは開かれたまま機械語プログラムが実行されます.

Sオプションは, フロッピーディスクからロードする場合のみ有効です. Sオプションを指定するとビデオ RAM にロードします. SCREEN5から8のときは, ビデオ RAM のアクティブページにロードします.

〈ファイルスペック〉は第一章のファイルスペックの説明を参照してください.

〈デバイス名〉はカセットレコーダを使う場合"CAS:"です. RAMディスク("MEM:")には使用できません.

● 参　照　BSAVE, SET PAGE

BSAVE*

ビーセーブ(コマンド)

● 機　能　機械語プログラムをセーブする.

● 書　式　BSAVE "〈ファイルスペック〉",〈開始アドレス〉,〈終了アドレス〉〔,〈実行開始アドレス〉|S〕

● 文　例　BSAVE　"CAS : DEMO", &HA100, &HA2FF

● 解　説　〈開始アドレス〉から〈終了アドレス〉までのメモリ上に置かれている機械語プログラムを,〈ファイルスペック〉で指定したファイルにセーブします.
〈ファイルスペック〉は第一章のファイルスペックの説明を参照してください.
〈デバイス名〉はカセットレコーダを使う場合,"CAS:"を指定します.
"MEM:"は使用できません.

〈実行開始アドレス〉を省略すると,実行開始アドレス部分には,〈開始アドレス〉と同じ値が指定されます.

Sオプションを指定すると,ビデオRAMの内容をフロッピーディスクのファイルにセーブします.指定する番地はSET PAGE文で設定したページの先頭からのオフセットで0～65535の値です.

● 参　照　BLOAD, 付録「VRAMについて」

CALL

コール(ステートメント)

● 機　能　拡張ステートメントを呼び出す.

● 書　式　CALL 〈拡張ステートメント名〉〔(〈引数〉〔,〈引数〉……)〕

● 文　例　CALL TALK ("A","BC","DEF")

　　_ TALK ("A","BC","DEF")

● 解　説　拡張ROMカートリッジでサポートされる拡張ステートメントを呼び出します. "CALL"の代りに"_"を使うことができます.

MSX2では以下の拡張ステートメントが組込まれていますので利用することができます.

CALL MEMINI〔(〈RAMディスクの上限〉)〕

RAMディスクの上限アドレスを指定します. 範囲は0～&H7FFFです. 初期状態では, &H7FFFとなっています. RAMディスクの上限の指定を省略した場合は, RAMディスク中のフリー領域の大きさを返します.
RAMディスクを使用する場合な, 電源オン後必ず一度はこのステートメントを実行してください.

CALL MFILES

RAMディスクに保存されている全てのファイル名を表示します.

CALL MKILL ("〈ファイル名〉")

RAMディスク中の指定したファイル名のファイルを消去します.

CALL MNAME ("〈旧ファイル名〉" AS "〈新ファイル名〉")

RAMディスク中の旧ファイル名で指定したファイル名を新ファイル名で指定したファイル名に変えます.

例) CALL MNAME ("OLDNM"AS"ENWNM")

注意：RAM ディスクに保存できるファイルは32個までです.

● 参　照　第1章1.18「ファイル」

CDBL

シーダブル(関数)

● 機　能　整数，単精度実数を倍精度実数に変換し，その結果を得る．

● 書　式　CDBL（〈数式〉）

● 文　例　A#＝CDBL（B!/2）

● 解　説　〈数式〉の値を倍精度実数に変換します．ただし型変換が行われるだけで有効桁数は変化しませんので，得られる値の精度は変換する前の型と同じ（整数型なら整数部のみ，単精度実数型なら有効数字6桁）になります．

CDBLの演算結果は倍精度実数型変数へ代入をしたのと同じです．

CHR$

キャラクタダラー(関数)

● 機　能　指定したキャラクタコードを持つ文字を得る．

● 書　式　CHR$（〈数式〉）

● 文　例　A$＝CHR$（65）
実行結果　A$＝"A"

● 解　説　〈数式〉の値のキャラクタコードを持つ文字を得ます．

〈数式〉の値が32～255の場合，そのキャラクタコードに対応する文字が得られます．

値が0～255の範囲でない場合は，"Illegal function call"エラーとなります．

● 参　照　ASC，資料「キャラクタコード表」

● サンプルプログラム

```
10 PRINT "CHR$(32) カラ CHR$(255)ノ ヒョウシ゛"
20 PRINT
30 FOR I=32 TO 255
40 PRINT CHR$(I);
50 NEXT:PRINT:PRINT
60 PRINT "CHR$(1)+CHR$(65) カラ"
70 PRINT "CHR$(1)+CHR$(95) マデ゛ノ ヒョウシ゛"
80 PRINT
90 FOR I=65 TO 95
100 PRINT CHR$(1)+CHR$(I);
110 NEXT
```

CHR$

CINT

シーインテジャー(関数)

● **機　能**　単精度実数値，倍精度実数値を整数値に変換し，その値を得る．

● **書　式**　CINT（＜数式＞）

● **文　例**　A%＝CINT (B#＊ 2)

● **解　説**　〈数式〉の値の小数点以下を切り捨てて整数に変換します．結果の値が－32768～32767の範囲にない場合は，"Overflow" エラーとなります．CINT での変換結果は，整数型変数への代入結果と同じです．

● **参　照**　CSNG, CDBL

CIRCLE

サークル(ステートメント)

● 機　能　　円または楕円を描く．(グラフィックモード専用)

● 書　式　　CIRCLE | (X, Y) / STEP(X, Y) |, 〈半径〉〔, 〈カラーコード〉〕〔, 〈開始角度〉〕

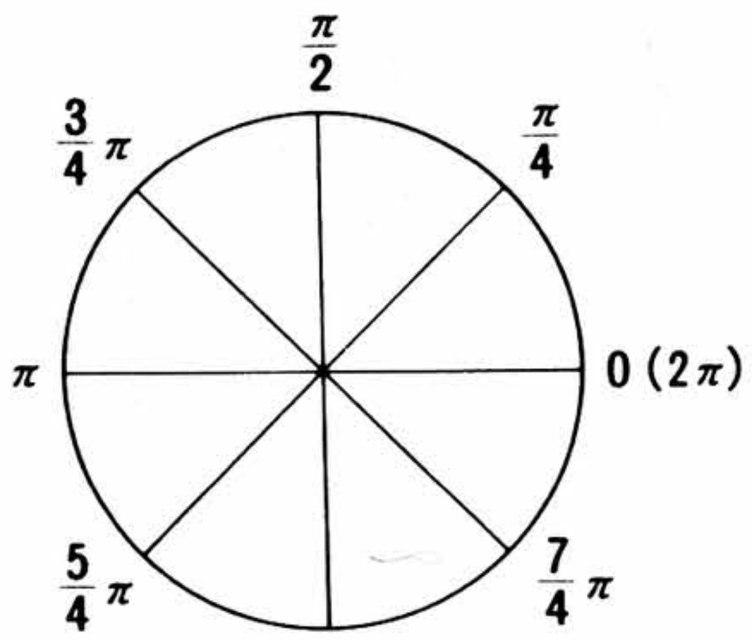
〔, 〈終了角度〉〕〔, 〈比率〉〕

● 文　例　　CIRCLE (80, 80), 40, 15, 0, 5, 0.5

● 解　説　　(X, Y) を中心とし，<半径>で指定する大きさの円を描きます．

〈カラーコード〉を指定すると，指定した色で円が描かれます．省略した場合には，COLOR 文で指定している前景色が用いられます．

〈開始角度〉，〈終了角度〉を指定すると，指定した角度の範囲のみ円弧が描かれます．角度の指定の単位はラジアンで，省略値はそれぞれ，0 と 2 π です．〈開始角度〉，〈終了角度〉が負であった場合には，その絶対値をとり，正にした角度が用いられますが，そのとき中心から半径が描かれますので扇形を描くことができます．〈開始角度〉，〈終了角度〉は，－ 2 π から＋ 2 π の範囲内にないと "Illegal function call" エラーとなります．



π＝3.1415926535898

〈比率〉は，(垂直方向の半径) ／ (水平方向の半径) で指定します．〈比率〉の省略値は1.0です．〈比率〉 が 1 以下の場合は，垂直方向の半径が短かくなり，逆に 1 以上の場合は，水平方向の半径が短かくなります．つまり，〈半径〉 が同じでも〈比率〉 が 1 以下ならば平べったい楕円となり，1 以上ならばたてに細長い楕円となります．

CIRCLE

中心座標は，STEPを付けてＬＰ（最終参照点）からの相対座標で指定することもできます．

CIRCLEを実行すると，LPは円の中心座標（X，Y）に設定されます．

● **参　照**　COLOR, SCREEN

● **サンプルプログラム**

```
10 '*** ミカン ***
20 COLOR 10,15:SCREEN 2
30 CIRCLE(128,96),76,10,,,1.15
40 CIRCLE(128,96),60,10,,,1.15
50 PAINT(128,96),10
60 FOR I=-6.28# TO  0 STEP .3
70 CIRCLE(128,96),60,15,I,I+.3,1.15
80 NEXT
90 FOR I=0 TO 2000:NEXT
100 COLOR 15,4,7
```

CLEAR

クリア(ステートメント)

● 機　能　　変数の初期化およびメモリ領域の大きさを設定をする.

● 書　式　　CLEAR〔〈文字列領域の大きさ〉〔,〈メモリの上限〉〕〕

● 文　例　　CLEAR 300, &HE000

●解　説　　すべての数値変数を0に，文字変数を""（ヌルストリング）に初期化します．また，DEF文（DEF FN, DEFSNG, DEFDBL, DEFSTR, DEF USR, DEFINT等)で定義した情報は無効となり，現在オープンしているファイルがあれば，すべてクローズされます.

〈文字列領域の大きさ〉は文字変数の内容をしまっておく文字列領域の大きさを示します．電源オン時は200にセットされています.

〈メモリの上限〉を指定すると，指定された番地の直前までをBASICが使用するメモリの上限値としますので，その番地以降から&HF380までに置かれたデータや，　機械語プログラムは，BASICによって破壊されることがありません．電源オン時は&HF380にセットされています.〈メモリの上限〉で指定できる値は&H831F～&HF380です.

メモリの上限を指定するときは，〈文字列領域の大きさ〉を省略することはできません.

〈文字列領域の大きさ〉や〈メモリの上限〉の値が，実際には取れないような値の場合は"Illegal function call"エラーとなります.

注）フロッピーディスクドライブを使用しているシステムでは，BASICが使用するメモリの上限値は変わります．この値はHIMEM(&HFC4Aと&HFC4B番地)に格納されていますので，下記命令を実行して，確認してください.

```
PRINT HEX$（PEEK(&HFC4B)＊&H100＋PEEK(&HFC4A)）
```

● 参　照　　資料「メモリマップ」，FRE

CLOAD/CLOAD?

シーロード／シーロードベリファイ(コマンド)

● 機　能　カセットテープからメモリへプログラムをロードする.

● 書　式　CLOAD〔"<ファイル名>"〕
　　CLOAD ?〔"<ファイル名>"〕

● 文　例　CLOAD "DEMO"
　　CLOAD? "DEMO"

● 解　説　CLOADはカセットテープから〈ファイル名〉で指定したプログラムを探し出してロードする命令です．探している最中に別のファイル名のプログラムがみつかればSkip :〈ファイル名〉が画面に表示され，また，目的のプログラムが見つかればFound :〈ファイル名〉が表示され，プログラムがロードされます.

〈ファイル名〉は6文字以内で指定してください．7文字以上の〈ファイル名〉を指定すると先頭の6文字だけが有効です．また，〈ファイル名〉を省略すると，カセットテープから最初に読み込まれたプログラムがロードされます.

CLOAD?は現在メモリ上にあるプログラムと〈ファイル名〉で指定されたテープ上のプログラムを比較(ベリファイ)します．このときすべての内容が等しければOkが表示され，もしそうでなければVerify errorとなります.

この比較はメモリ上のプログラムが間違いなくテープ上にセーブされているかを確認するためのもので，一般にCSAVEの直後に使います.

注意：CLOADを実行するとNEW文と同じ働きもしますので，CLOADの実行前にメモリ上にあったプログラムや変数は消去され，開いていたファイルがあれば，閉じられます.

● 参　照　CSAVE

CLOSE

クローズ(コマンド)

● 機　能　ファイルを閉じる.

● 書　式　CLOSE〔〔#〕<ファイル番号>〔,〔#〕<ファイル番号>〕・・・〕

● 文　例　CLOSE #1

● 解　説　〈ファイル番号〉に対応するファイルを閉じます.

CLOSE 文は,ファイルが出力用にオープンされていた場合には,バッファに残っていたデータの掃き出しを行いますので,ファイルへの出力処理を正しく終了するためには,必ずCLOSE 文を実行しなくてはなりません.END 文,NEW 文を実行すると,自動的にすべてのファイルを閉じますが,STOP 文はファイルを閉じませんので注意してください.

指定した〈ファイル番号〉は,以後の OPEN 文で自由に指定することができるようになります.

CLOSE では,〈ファイル番号〉を複数指定することにより,一度に複数のファイルを閉じることができます.また〈ファイル番号〉を省略すると,そのとき開いているファイルをすべて閉じます.

閉じたファイルに対しては,再びOPEN 文で開くまで入出力を行うことはできません.

● 参　照　第1章「ファイル」,OPEN, PRINT #, INPUT #, END, NEW

CLS

クリアスクリーン(ステートメント)

● 機　能　画面を消去する.

● 書　式　CLS

● 文　例　CLS

● 解　説　画面上の文字や絵をすべて消去します．ただし,KEY ON状態のとき,画面下のファンクションキー表示およびスプライトパターン表示だけは残ります.

テキスト画面が消去されたときは，カーソルの位置は（０，０）へ移動します．またグラフィック画面を消去した場合には，LP(最終参照点)はCLS実行前のままです.

● サンプルプログラム

```
10 CLS
20 FOR J=0 TO 50
30 X=INT(RND(1)*27):Y=INT(RND(1)*22)
40 LOCATE X, Y:PRINT "*"
50 NEXT J
60 LOCATE 0, 0:PRINT "CLSシマス"
70 FOR K=0 TO 1000:NEXT K
80 CLS
```

COLOR

カラー(ステートメント)

● 機　能　画面の各部の色を指定する.

● 解　説　色はカラーコードにより指定します. カラーコードは0～15までの16ヶあり, 各々512色の中から選んで色を割り当てることができます. (1.16「カラーコード」参照).電源オン時の初期状態では以下のような割り当てとなっています.( )内はそれぞれ赤, 緑, 青のレベルを示しています.(COLOR＝文参照).

0：透明(0, 0, 0)　1：黒(0,0,0)　2：緑(1,6,1)
3：明るい緑(3,7,3)　4：暗い青(1,1,7)　5：明るい青(2,3,7)
6：暗い赤(5,1,1)　7：水色(2,6,7)　8：赤(7,1,1)
9：明るい赤(7,3,3)　10：黄(6,6,1)　11：明るい黄(6,6,4)
12：暗い緑(1,4,1)　13：紫(6,2,5)　14：灰(5,5,5)
15：白(7,7,7)

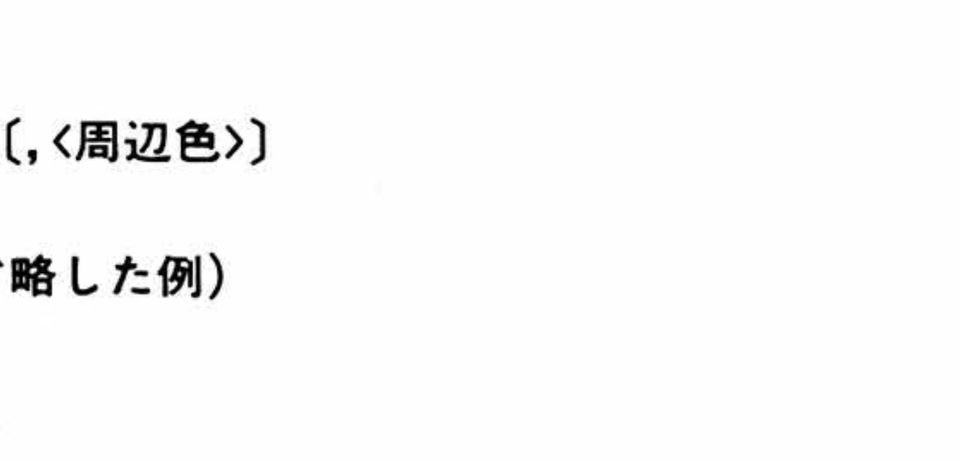
◆前景色,背景色,周辺色を指定する.

● 書　式　COLOR〔〈前景色〉〕〔,〈背景色〉,〕〔,〈周辺色〉〕

● 文　例　COLOR , 7, 4 (前景色の指定を省略した例)

● 解　説　各々カラーコードにより指定します.
カラーコードは, SCREEN 6 では 0 ～ 3, SCREEN 8 では 0 ～255, 他は 0 ～15の数値です. 指定を省略すると, それぞれCOLOR 文を実行する前の状態を保ちますが, 3つの指定を全部省略することはできません.
起動時には15(白), 4(暗い青), 7(水色)がセットされています.

〈前景色〉とは, テキスト画面に文字を表示したり, グラフィック画面に点や線を描くときに使われる色のことです. PSET, LINE, CIRCLE などのグラフィック画面では, 特別な色指定をしなかった場合, この前景色が使用されます.



COLOR

〈背景色〉とは，画面の背景となる色のことで，テキストモードであればこの命令実行後すぐに，また**グラフィックモードではこの命令の後のCLS文またはSCREEN文を実行時に指定したカラーコードに変わります．**

〈周辺色〉とは，表示領域外の周辺部の色のことです．SCREEN 0 のモードだけは，周辺色の指定は無視されます．このモードでは周辺色は背景色と同じになります．前景色および背景色に透明色が指定された場合には，この周辺色が表示されます．

◆カラーコードの色を変更します．

● **書　式**　**COLOR＝(〈カラーコード〉，〈Rコード〉，〈Gコード〉，〈Bコード〉)**

● **解　説**　各カラーコードの色は，512色中より任意の1色を割当てることができます．カラーコードは0～15です．（SCREEN 6 は 0 ～ 3 ）．カラーコード 0 は透明色に固定です．また，この指定はSCREEN 0 ～ 7 のモードで有効です．

〈R コード〉は赤色の成分を 0 ～ 7 の数値で指定します．

〈B コード〉は緑色の成分を 0 ～ 7 の数値で指定します．

〈G コード〉は青色の成分を 0 ～ 7 の数値で指定します．

例　COLOR＝(3 , 7 , 0 , 0)　カラーコード 3 を赤色にします．

　　COLOR＝(10, 7 , 7 , 7)　カラーコード10を白色にします．

　　COLOR＝(5 , 4 , 0 , 4)　カラーコード 5 を暗い紫にします．

COLOR＝文により割当てた色はCOLOR NEW文，SCREEN文の実行またはSCREENのモードが変わると無効となり，初期状態に復帰します．

す．

◆カラーコードの初期化

● **書　式**　**COLOR〔＝NEW〕**

● **解　説**　COLOR＝文により変更した色指定を無効とし，初期状態の割当てにもどします．

COLOR

◆VRAM上のルックアップテーブルの通りにカラーコードの色を設定する.

● **書　式**　**COLOR＝RESTORE**

● **解　説**　フロッピーディスクに保存されている画像データ（VRAM上のカラーテーブルの内容も含めて保存されているデータ）を読み込んだときに実行します．画像データを読み込んだだけではカラーコードの色は変更されません．COLOR＝RESTORE命令を実行することによって，保存したときと同じ色で表示することができます.

◆SCREEN 8 のカラーコード

SCREEN 8 のモードでは,他のモードと異なり，パレット機能はありません．0～255のカラーコードにより256色を使用することができます．今，使用するカラーコードの2進表現を$b_7 b_6 \cdots\cdots b_0$とするとき，このカラーコードに対応する色は次のようになります.

カラーコード　　$\underbrace{b_7\ b_6\ b_5}$　$\underbrace{b_4\ b_3\ b_2}$　$\underbrace{b_1\ b_0}$

緑色の成分（G＝0～7）　赤色の成分（R＝0～7）　青色の成分（B＝0～3）

数式で現わすと……$G \times 32 + R \times 4 + B$

例．カラーコード	2進表現	色
0	00000000	黒
255	11111111	白　（G＝7，R＝7，B＝3）
227	11100011	水色（G＝7，R＝0，B＝3）
28	00011100	赤　（G＝0，R＝7，B＝0）

◆SCREEN 6 のカラーコード

SCREEN 6 のモードでは，0～3までのカラーコードだけしか使用できません．4～15までのカラーコードを指定した場合には，それを4で割った時の余りがカラーコードとなります．前景色，背景色，周辺色は他のモードで指定されていたものをそのまま使用すると，多少変則的な所がありま

COLOR

すので，SCREEN6のモードに入った所で，設定しなおすとよいでしょう．(設定後CLS命令を実行することを忘れないこと．)また，SCREEN6のモードで指定した前景色，背景色，周辺色のカラーコードは，他のSCREENモードに移った場合，以下のように変わりますのでご注意ください．

SCREEN6で設定したカラーコード	他のモードに移った時のカラーコード
0	0
1	5
2	10
3	15

● 参　照　第1章「カラーコード」，COLOR SPRITE，CLS，付録「カラーコードについて」

● サンプルプログラム　・初期状態での前景色，背景色の全組合せを表示

```
10 CLS
20 FOR B=0 TO 15
30 FOR C=0 TO 15
40 COLOR C,B
50 LOCATE 5,5:PRINT"ぜんけいしょく=";C
60 LOCATE 5,7:PRINT"はいけいしょく =";B
70 FOR T=0 TO 300:NEXT T
80 NEXT C,B
90 COLOR 15,4,7
```

・SCREEN8での256色を表示

```
10 SCREEN 8
20 FOR C=0 TO 255
30 LINE(C,0)-(C,211),C
40 NEXT C
50 GOTO 50
```

C

COLOR SPRITE ✳

カラースプライト(システム変数)

● 機　能　スプライトパターンの1行毎の色を指定する．(SCREEN 4 ～ 8 専用)

● 書　式　COLOR SPRITE$(<スプライト面番号>)＝<カラーコード列>

COLOR SPRITE(<スプライト面番号>)＝<カラーコード>

● 文　例　COLOR SPRITE$(0)＝CHR$(10)＋CHR$(3)＋…….

結果：スプライトパターンの1行目をカラーコード10に，2行目をカラーコード3……に指定．

● 解　説　この命令が使用できるのはSCREEN 4 ～ 8 だけです．

COLOR SPRITE$文はスプライトパターンの1行毎に色を指定します．COLOR SPRITE$により色指定したスプライトパターンはPUT SPRITE命令により画面に表示すると，行毎に指定した色で表示されます．この時，PUT SPRITEでは色指定は省略してください．指定すると，そのスプライト面は指定した色で初期化されCOLOR SPRITE命令による色指定は無効となります．

COLOR SPRITE$命令の最初のデータより順次1行目，2行目……のカラーコードをCHR$関数で接いで指定します．色指定しない部分は前景色となります．

COLOR SPRITE文は，指定したスプライトパターンの全体の色を指定します．すなわち，スプライトサイズが8×8画素の場合，次の2つの命令は全く同じ働きをします．

```
COLOR SPRITE$(0)＝STRING$(8，CHR$(15))
COLOR SPRITE(0)＝15
```

COLOR SPRITE文には，特殊なカラーコードを指定することによりスプライトが重なった時の割り込みを禁止したり，重なった部分の色を変えるなどの機能があります．詳細は付録の「COLOR SPRITE補足説明」を参照してください．

● 参　照　PUT SPRITE，SPRITE$，付録「COLOR SPRITE補足説明」

COLOR SPRITE

● サンプルプログラム

```
10 SCREEN 5,1
20 SPRITE$(0)=STRING$(8,CHR$(&HFF))
30 COLOR SPRITE$(0)=CHR$(1)+CHR$(2)+CHR$(3)+
CHR$(4)+CHR$(5)+CHR$(6)+CHR$(7)+CHR$(8)
40 PUT SPRITE0,(100,100),,0
50 GOTO 50
```

1 行ずつ色を変えてスプライトを表示しています.

CONT *

コンティニュー(コマンド)

● **機　能**　コントロールストップキー入力，または STOP 文によって停止したプログラムの実行を再開する.

● **書　式**　CONT

● **文　例**　CONT

● **解　説**　CONT は通常デバッグのために STOP 文と共に用いられます．STOP 文やコントロールストップキー（CTRL + STOP）のキーインによりプログラムの実行を停止して，ダイレクトモードで変数の値等を調べたり変更した後に，CONT 文によって実行を再開することができます.

実行停止中にLISTや，PRINTなどのコマンドを実行することはできますが，実行停止中にプログラム内容の変更を行った場合にはCONT文による実行の継続はできません．また，表示画面の回復や，表示モードの再設定などは，ユーザーの対処が必要です.

● サンプルプログラム

```
10 PRINT"TEST PROGRAM"
20 STOP
30 PRINT"CONT ﾆﾖﾘ ｼﾞｯｺｳ ｻﾚﾏｽ。"
```

COPY ✳

コピー(ステートメント)

● **機　能**　**VRAMや配列などの内容を転送しコピーする.(SCREEN 5～8 専用)**

COPYには複写元と複写先の種類により各種の書式がありますので，類似の書式ごとに分けて説明します．この機能は画面モードが5～8のとき有効です.

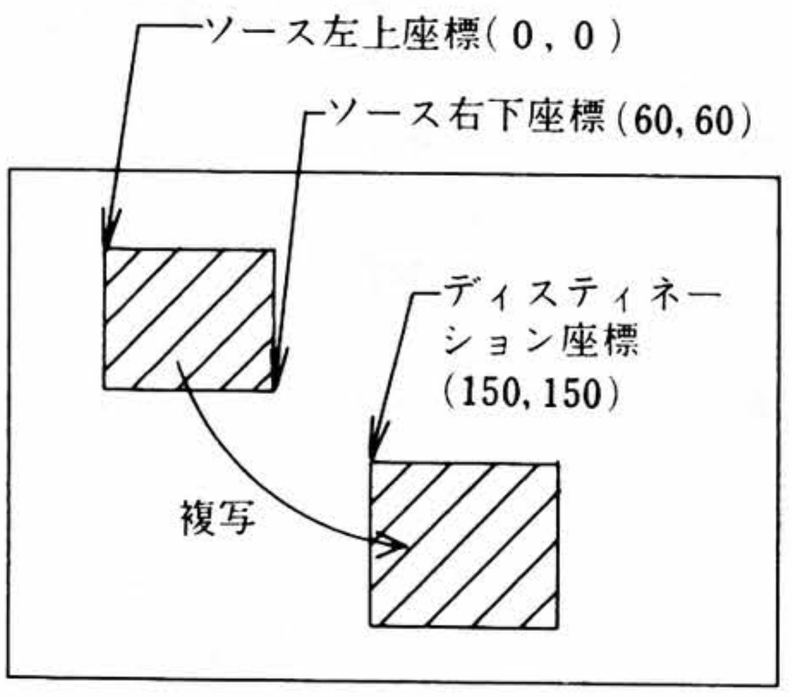
注）DISK BASICのCOPY命令は（フロッピーディスク内のファイルの複写)ここでは説明していません．必要な場合は,「DISK BASIC/MSX DOS説明書」を参照してください.

◆ VRAM内でのコピー

● **書　式**　**COPY（〈ソース左上座標〉)－(〈ソース右下座標〉)〔,〈ソースページ〉〕TO (〈ディスティネーション座標〉)〔,〈ディスティネーションページ〉〕〔,〈論理演算子〉〕**

● **文　例**　**COPY(0,0)–(60,60), 0 TO (150,150)**

● **解　説**　VRAM内の画像データの一部をVRAMの他の部分に複写します.
(図は文例の場合を示す.)
本モードはSCREEN 5～8 でのみ使用可能です.



〈ソースページ〉は現在使用中の〈アクティブページ〉と同じ場合は省略できます．ただし,〈ソースページ〉を省略する場合は，",〃も一緒に省略します.
また，〈ディスティネーションページ〉は，〈ソースページ〉と同じ場合には省略できます.（SET PAGE文参照）

〈論理演算子〉については56頁の説明とLINE文を参照してください.

COPY*

C

◆ VRAMの画像データと配列変数またはファイルとの間のコピー

● 書　式　COPY (〈ソース左上座標〉)－(〈ソース右下座標〉)〔,〈ソースページ〉〕TO
〈配列名またはファイル名〉

COPY〈配列名またはファイル名〉〔,〈方向〉〕TO(〈ディスティネーション
座標〉)〔,〈ディスティネーションページ〉〕〔,〈論理演算子〉〕

● 文　例　COPY(0,0)－(100,100) TO　XM

COPY"A：SAMPLE", 1　TO　(100,50)

● 解　説　VRAM 関係の指定方法は「◆VRAM内でのコピー」と同様です.

〈ソースページ〉,〈ディスティネーションページ〉は,現在使用中の〈アクティブページ〉と同じ場合は省略できます. ただし, ソースページを省略する場合は, ",″も一緒に省略します.

配列にコピーする場合は, 事前に DIM 命令にて,次の大きさ以上の領域の確保が必要です.

領域W＝(X＊Y／C)＋ 4(演算結果の端数は切上げ)

ここでXは横方向のドット数, Yは縦方向のドット数

Cは定数でSCREEN5と7では2, SCREEN6では4

SCREEN8では1

したがって, DIM 命令で確保すべき配列の大きさは次のようになります. DIM 命令の添字の値S＝(W／A)－1 (端数は切上げ)

Aは配列の型によって定まる定数で整数型

は 2, 単精度型は 4, 倍精度型は 8 です.

例) 文例の場合(SCREEN5, XM は単精度型と仮定)

W＝(101＊101／2)＋ 4 ＝5104.5⟶5105(端数切上げ)

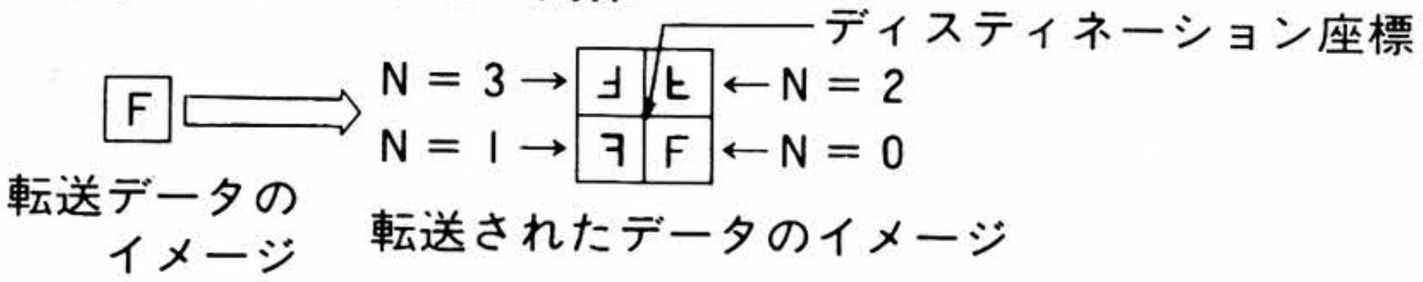
S ＝(5105／4)－ 1 ＝1275.25⟶1276(端数切上げ)

〈ファイル〉はフロッピーディスクのファイルに対してのみ可能です. RAMディスク,やカセットテープのファイルには使用できません.

COPY*

〈方向〉はデータを画面に描き込む方向を0～3の数値で指定します．0の場合は省略できますが省略する場合は〝,〟も一緒に省略します．

数値(N)と転送データとの関係



〈論理演算子〉は転送されるデータと転送先の元のデータとの論理演算を指定します．下記10種類があり，省略値は〝PSET〟です．〝T〟が付いたものは，転送データが透明色の場合は，転送先の色は変わりません．

PSET,　PRESET,　XOR,　OR,　AND

TPSET,　TPRESET,　TXOR,　TOR,　TAND

詳細な動作内容については，LINE文を参照してください．

◆ 配列変数とファイル間のコピー

● 書　式　COPY〈配列名〉TO〈ファイル名〉

COPY〈ファイル名〉TO〈配列名〉

● 文　例　COPY　A　TO　"SAMPLE"

COPY　"A：FILEZ"　TO　HZ

● 解　説　フロッピーディスクのファイルと配列変数との間でデータ転送します．

RAMディスクやカセットテープのファイルは使用できません．

◆ 外部ビデオ信号をVRAMに取り込みます．

● 書　式　COPY SCREEN〔〈モード〉〕

● 文　例　COPY SCREEN 1

◆ 解　説　この機能はSCREEN 5 ～ 8 で有効で、外部からパソコンに入力されているテレビ画面やVTRの画面を静止画としてVRAMに取り込みます．このような機能をディジタイズといいます．(通常はSCREEN 8 で使用します．)

〈モード〉が 0 の場合は 1 フィールドの信号をディスプレイページに取り込みます．1 の場合は，連続する 2 フィールド(＝1 フレーム)の信号を〔デ

COPY*

C

ィスプレイページー1〕ページとディスプレイページとに取り込みます．この場合，ディスプレイページ番号は1または3である必要があります．
〈モード〉を省略した場合は0とみなされます．
実際の使用に当たっては，各種の手続や準備が必要です．詳細は付録「スーパーインポーズとディジタイズ」を参照してください．

COS

コサイン(関数)

● **機　能**　余弦（コサイン）を得る．

● **書　式**　COS（〈数式〉）

● **文　例**　C＝COS（3.1415926535898/2）

● **解　説**　〈数式〉の値に対する余弦（コサイン）を得ます．〈数式〉の単位はラジアンです．

〈数式〉はどの型でも構いませんが，結果は倍精度で得られます．

● **参　照**　SIN, TAN

● **サンプルプログラム**

```
10 '*** COS キョクセン ト エン ******
20 SCREEN 2
30 PAI=3.141592653#
40 LINE(0,96)-(255,96)
50 LINE(191,0)-(191,191)
60 FOR I=0 TO PAI*2 STEP PAI/100
70 X=I/PAI*63
80 Y=96-63*COS(PAI*X/64)
90 PSET(X,Y)
100 X=192+63*SIN(I)
110 Y=96-63*COS(I)
120 PSET(X,Y)
130 NEXT
140 GOTO 140
```

CSAVE

シーセーブ(コマンド)

● 機　能　メモリ上にあるプログラムをカセットテープへセーブする.

● 書　式　CSAVE "〈ファイル名〉"〔,〈ボーレート〉〕

● 文　例　CSAVE "DEMO1"

● 解　説　CSAVE文はメモリ上にあるプログラムをバイナリ形式でカセットテープへ記録（セーブ）します.

テープ上にセーブされたプログラムは，CLOAD文を使って再びメモリ上にロードすることができます.

〈ファイル名〉は最大6文字からなる文字列で指定します. 7文字以上の〈ファイル名〉を指定すると，先頭6文字までが有効で，以降は無視されます. プログラムが無事にセーブできたかどうかはCLOAD?を使って確認してください.

〈ボーレート〉にはカセットテープへ記録するときの転送速度を次の数値で指定します.〈ボーレート〉を省略すると，SCREEN文で指定したボーレートが採用されます. SCREEN文でも指定していない場合は1200ボーとなります.

1：　1200ボー

2：　2400ボー

CSAVEをプログラムモードで実行した場合，すぐにコマンドレベルに戻りますので，以降にプログラムが続いていても実行されません.

CSAVEの途中で実行を中止する場合は CTRL + STOP キーを押してください.

機械語プログラムをセーブ，ロードする場合にはBSAVE,BLOADを使ってください.

● 参　照　CLOAD, BSAVE

CSNG

シーシングル(関数)

● 機　能　整数値，倍精度実数値を単精度実数値に変換し，その値を得る．

● 書　式　CSNG（〈数式〉）

● 文　例　A!＝CSNG (B #)

● 解　説　〈数式〉の値を有効数字6桁の単精度実数型に変換します．その値が－9.99999E＋62～9.99999E＋62の範囲にない場合は＂Overflow＂エラーが起こります．

CSNGでの変換結果は，単精度実数型変数への代入結果と同じです．

● 参　照　CINT, CDBL

CSRLIN

カーソルライン(システム変数)

● 機　能　現在のカーソルの行位置を得る．(テキストモード専用)

● 書　式　CSRLIN

● 文　例　Y＝CSRLIN

● 解　説　現在のカーソルの行位置を調べます．値の範囲は0から23までです．なお，画面の最上行が0で，値が大きくなる程画面の下部に向います．

この変数に値を代入することはできません．

カーソルの桁位置（水平方向）を知るためにはPOS関数を使います．

● 参　照　POS, LOCATE

DATA

データ(ステートメント)

● 機　能　READ 文で読み出すことができる数値や文字定数を用意する.

● 書　式　DATA 〈定数〉〔,〈定数〉・・・〕

● 文　例　DATA 123, ABC, 456

● 解　説　DATA 文は READ 文で読み出す〈定数〉を用意します. 実行時には無視され, なにも動作しません.

〈定数〉は1行（255文字）に入る範囲内で, カンマで区切っていくつでも指定することができます.

また, 1つのプログラム中には複数の DATA 文を置くことができますが, DATA 文中の〈定数〉は, 1つのプログラム内で連続したデータの並びとみなされます.

〈定数〉は, 整数, 単精度実数, 倍精度実数, 文字定数のいずれでもかまいませんが, 式（2＋3など）は許されません. **またREAD文で読み出す場合, READ文で指定する変数の型は, 対応するDATA文の定数の型と一致しなければなりません.**

DATA文中のデータはカンマ（,）によって区切りますが, データが文字定数で, その文字列にカンマやセミコロン（:）, 意味のある空白を含む場合には, その文字定数の前後を引用符（"）で囲みます.

RESTORE文を使ってREAD文で読み出すDATA文を行単位で指定することができます.

● 参　照　READ, RESTORE

● サンプルプログラム

```
10 READ A$,B$:PRINT A$;B$
20 FOR I=1 TO 3
30 READ A$:PRINT A$
40 NEXT
50 READ A,B,C,D
60 PRINT A;B;C;D
70 DATA モシ゛テ゛ータ,ノ ヨマセカタ
80 DATA "     セントウノクウハクヤ,カンマ ハ"
90 DATA クォーテーションテ゛カコムコト
100 DATA スウチテ゛ータ ハ テイスウタ゛ケ !
110 DATA 3,&H20,&B1010,3.14
```

DEF FN*

デファイン ファンクション(ステートメント)

● 機　能　関数を定義する.

● 書　式　DEF FN〈名前〉〔(〈引数〉〔,〈引数〉・・・])〕＝〈関数の定義式〉

● 文　例　DEF FNA (X, Y)=X * 2+Y * 3

● 解　説　DEF FN 文を使ってプログラム中でたびたび使う演算と演算で使う変数とを関数としてあらかじめ定義しておくと，以降のプログラムでは，いちいち演算を記述しなくてもこの関数を呼び出すだけで演算結果を求めることができます.

"FN" とこれに続く〈名前〉は定義する関数の名前であり，〈名前〉は変数名の規則に従った1～2文字の英数字です.

〈引数〉は，〈関数の定義式〉中で使っている同じ名前の変数に対応しています．この変数名は，〈関数の定義式〉を評価する際にのみ使われますから，プログラム中に同一名の変数があっても影響はありません.〈引数〉は最高9ヶです.

〈関数の定義式〉は，その関数の演算内容を記述する式で，1行の範囲に限られます．DEF FN で定義する関数は，数値関数，文字関数，その混在のいずれでもかまいませんが，**この関数を呼び出す時に指定する〈引数〉の変数の型はDEF FN文で指定した〈引数〉の変数の型と同じでなければなりません.**

DEF FN 文は，これによって定義される関数がプログラム中で呼ばれる前に実行されていなければなりません．また，ダイレクトモードでの実行はできません.

● サンプルプログラム

```
10 DEF FNS(I, J)=I*J
20 DEF FNL(I, J)=2*(I+J)
30 PRINT "      *  BOX  *"
40 PRINT  "  タテ  ヨコ  メンセキ        ナガサ"
50 FOR I=1 TO 5
60 FOR J=1 TO 3
70 PRINT I;J;FNS(I, J), FNL(I, J)
80 NEXT J, I
```

DEFINT/SNG/DBL/STR

デファイン インテジャー／シングル／ダブル／ストリング（ステートメント）

● **機　能**　変数の型宣言をする．

● **書　式**　**DEFINT　〈文字の範囲〉〔,〈文字の範囲〉・・・〕**
DEFSNG　〈文字の範囲〉〔,〈文字の範囲〉・・・〕
DEFDBL　〈文字の範囲〉〔,〈文字の範囲〉・・・〕
DEFSTR　〈文字の範囲〉〔,〈文字の範囲〉・・・〕

● **文　例**　**DEFINT A, I−K**

● **解　説**　DEF 文を実行すると，〈文字の範囲〉で指定された文字で始まる変数名の変数を次のような型に宣言します．

DEFINT———整数型
DEFSNG———単精度実数型
DEFDBL———倍精度実数型
DEFSTR———文字型

〈文字の範囲〉は〈英字１文字〉または〈英字１文字－英字１文字〉で指定します．（文例の　DEFINT A, I-K　はAとI, J, K を指定）

ただし，**DEF文で行った型宣言より，型宣言文字による指定の方が優先されます**．また，型宣言が行われていない文字で始まる変数は，すべて倍精度変数とみなされます．なお，**DEF文で宣言した後にCLEAR文を実行すると，DEF文での宣言はすべて無効になります**．

● **参　照**　変数名の型宣言

● **サンプルプログラム**

```
10 DEFINT I
20 DEFSNG J
30 DEFDBL K
40 DEFSTR L
50 I=1.6:PRINT I
60 J=1!:PRINT J
70 K=1/3:PRINT K
80 L="ABCDEF":PRINT L
90 CLEAR
100 PRINT I;J;K;L
```

DEF USR*

デファイン ユーザー(ステートメント)

● 機　能　機械語で作られたユーザー関数の実行開始番地を定義する.

● 書　式　DEF USR〔<番号>〕＝<開始番地>

● 文　例　DEF USR3＝&HF000

● 解　説　USR 関数 (ユーザー関数) が呼び出す機械語ルーチンの実行開始番地を定義します. 事前に CLEAR 文のメモリの上限の定義で, 機械語ルーチンのための領域を確保しておくことが必要です.

<番号>は, 0 ～ 9 までの値で, 複数のユーザー関数を用いる場合の識別を行います, したがって最大10個までのユーザー関数を準備, 利用することができます. <番号>が省略された場合には 0 と解釈されます.

<開始番号>は機械語ルーチンの実行開始番地です.

● 参　照　USR, CLEAR

DELETE

デリート(コマンド)

● **機　能**　プログラムの一部分を削除する.

● **書　式**

DELETE | **〔〈始点行番号〉－〈終点行番号〉〕**
〈行番号〉
－〈終点行番号〉

● **文　例**　**DELETE 10－230**

● **解　説**　〈始点行番号〉から〈終点行番号〉までのプログラムを削除します.〈行番号〉だけを指定した場合はその行だけを削除し,－〈終点行番号〉を指定した場合には,プログラムの先頭から〈終点行番号〉までを削除します.行番号の指定ではBASICが現在着目している行の行番号の代わりにピリオド(.)を使うことができます.

プログラムモードでDELETE文を実行しますと,実行後にコマンドレベルに変ります.

例)

DELETE 10

(行番号10を削除する)

DELETE 50－100

(行番号50から行番号100までを削除する)

DELETE －100

(プログラムの先頭から行番号100までを削除する)

DIM

ディメンジョン(ステートメント)

● 機　能　配列変数を定義し，メモリ領域に割り当てる．

● 書　式　DIM <変数名>(<添字の最大値>〔，<添字の最大値>…〕)〔，<変数名>( )…〕

● 文　例　DIM A(15, 20)　　DIM A1(30), C$(5, 15), E(10, 5, 5)

● 解　説　配列変数の添字の最大値を設定し，同時にメモリ上にその配列の領域を割り当てます．DIM文で宣言せずに配列変数を用いた場合，3次元配列まではその添字の最大値を自動的に10とします．もし，設定された最大値より大きい値の添字が用いられた場合は，"Subscript out of range" エラーが起こります．

添字の最小値は，0です．

DIM 文が実行された時点で，その配列のすべての要素の値は0（文字型の場合はヌルストリング）に設定されます．
配列変数は第1章で説明したメモリを占有しますのでDIM文を使うときはメモリが不足しないことを確認して下さい．

不用になった配列変数はERASE文で削除することができます．いったんDIM文で定義した配列変数はERASE文，またはCLEAR文を実行しない限り再定義することはできません．

● 参　照　第1章「配列変数」，CLEAR，ERASE

● サンプルプログラム

```
10 PRINT "DIM ナシ デハイレツ ヲ ツカウト"
20 FOR I=0 TO 10
30 A(I)=I:PRINT A(I)
40 NEXT:PRINT
50 PRINT "(0)ー(10)　ノ ヨウソガヨウイサレマス"
60 N=15
70 PRINT "ヨウソスウ ヲ 16 コ ニスルナラ"
80 PRINT "DIM A(15)ト シマス"
90 ERASE A:DIM A(N)
100 FOR I=0 TO N:A(I)=I:PRINT A(I),:NEXT
110 PRINT"ERASE セズニ マタ DIM スルト ERROR ニ ナリマス!"
120 DIM A(16)
```

DRAW*

ドロー(ステートメント)

● 機　能　画面に図形を描く．(グラフィックモード専用)

● 書　式　DRAW〈文字式〉

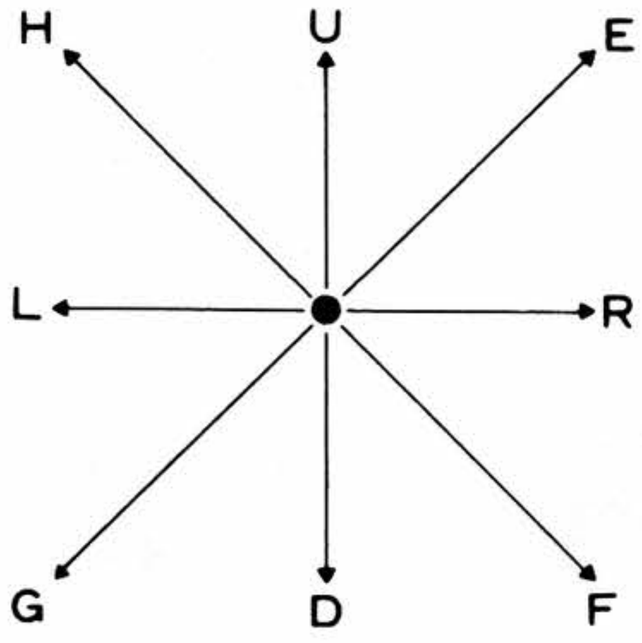
● 文　例　DRAW "BM10, 10F100U100G100U100"

● 解　説　画面上にある一点(参照点)を相対的に移動させながら線を描きます．

〈文字式〉には，参照点の位置設定，移動の方向と距離，角度などを指定するグラフィックマクロ命令を1つ以上連続して記述します．ここで使用するグラフィックマクロ命令には，大きく分けて，移動，回転，色指定，スケールファクタ，ローカルマクロ命令の5種類があります．

●移動マクロ命令

U〈距離〉　上へ移動
D〈距離〉　下へ移動
L〈距離〉　左へ移動
R〈距離〉　右へ移動
E〈距離〉　右斜め上へ移動
F〈距離〉　右斜め下へ移動
G〈距離〉　左斜め下へ移動
H〈距離〉　左斜め上へ移動



Mx, y　移動位置をx方向，y方向各々についての絶対座標あるいは相対座標で指定します．ここで，xの前にプラス（＋）あるいはマイナス（－）の符号が書かれていれば，参照点からの相対座標指定と解釈されます．

B　各々の方向へ移動しますが，描画は行いません．

N　各々の方向へ描画しながら移動しますが，描画後の参照点は始点となります．

上の各コマンド中の〈距離〉は，各方向への移動距離を画素単位で指定しますが，実際の移動距離は，これにSコマンドで設定されるスケール

DRAW*

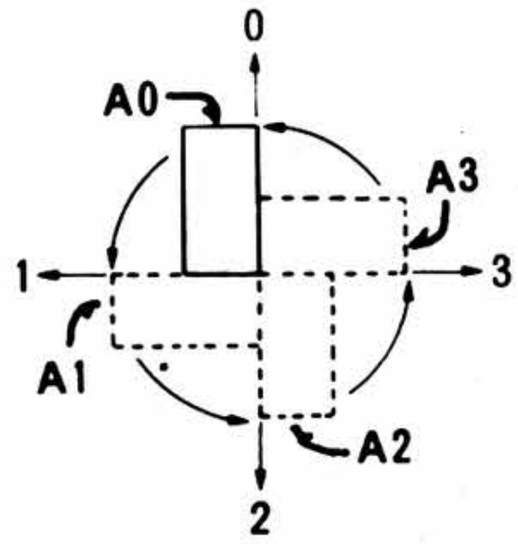
ファクタを乗じたものとなります．また，各コマンドに同じ〈距離〉を指定しても，画面上での実際の移動距離は，前頁の図のように各方向でそれぞれ異なっています．

また，B，Nのマクロ命令はU～Mのマクロ命令の直前に付加して使用します．

●回転マクロ命令

A〈角度〉 U, D, L, R, E, F, G, H および相対座標指定のときのMマクロで描画する図形を90°単位で回転して表示します．一度指定すると次のAマクロまで有効です．

〈角度〉は0～3の数字で表わし，0は0度，1は90度，2は180度，3は270度を表します．



●色指定マクロ命令

C〈カラーコード〉 〈カラーコード〉で指定した色で図形を描きます．〈カラーコード〉は0～15までで，詳細は第1章で説明しています．

●スケールファクタマクロ命令

S〈整数〉 スケールファクタを設定します．

〈整数〉に指定できる値は1～255の範囲で，この値の4分の1がスケールファクタとなります．たとえば〈整数〉に1を指定した場合，スケールファクタは4分の1となります．U, D, L, R,

DRAW*

E, F, G, H マクロ及びMマクロでの実際の移動距離は，距離にスケールファクタを乗じたものです．〈整数〉の指定を省略した場合は 4 が設定されます．Sマクロを設定すると，次のSマクロまで有効です．

●ローカルマクロ命令

X〈文字変数〉； 〈文字変数〉中の文字列に含まれているグラフィックマクロ命令を実行します．
〈文字変数〉には上記のグラフィックマクロをあらかじめ入れておきます．〈文字変数〉の後のセミコロン（；）は必ず入れてください．
ローカルマクロはグラフィックマクロの文字列が255を越える場合や繰り返し図形を描くときなどに便利です．

例)

```
A$= "U80R80D80" : DRAW "XA$;"
```

グラフィックマクロ命令では〈距離〉や〈角度〉などのパラメータに定数を指定しますが，
"＝〈変数名〉;" の形式にすれば数値変数を指定することができます．この場合〈変数名〉の後ろにはセミコロン（；）を必ず入れます．

例)

```
10 SCREEN2
20 PSET(100, 100)
30 X1=40:X2=50
40 DRAW "M+=X1;,-=X2;"
```

●サンプルプログラム

```
10 SCREEN 2
20 L=50:PSET(220.191).7
30 DRAW "U190"
40 FOR I=189 TO 1 STEP -4
50 A$="L"+STR$(I)+"D"+STR$(I-1)+"R"+STR
$(I-2)+"U"+STR$(I-3)
60 DRAW "XA$;"
70 NEXT I
80 GOTO 80
```

END

エンド(ステートメント)

● 機　能　プログラムを終了する.

● 書　式　END

● 文　例　END

● 解　説　プログラムの実行を終了し，すべてのファイルを閉じてコマンドレベルに戻ります.

END文はプログラムの実行を終了させたい所に置けばよく，また，いくつ置いてもかまいません．プログラムの最後のEND文は省略することができますが，この場合，ファイルのクローズは行われません.

● サンプルプログラム

```
10 PRINT "*** ﾌﾟﾛｸﾞﾗﾑ ヲ ｼｭｳﾘｮｳ ｼﾏｽ ***"
20 END
30 PRINT "ｺﾉ ｷﾞｮｳ ﾊ ｼﾞｯｺｳｻﾚﾏｾﾝ。"
```

EOF*

エンド オブ ファイル(関数)

● 機　能　ファイルが終了したかどうかを調べる.

● 書　式　EOF (〈ファイル番号〉)

● 文　例　IF EOF (1)　THEN CLOSE1 : END

● 解　説　〈ファイル番号〉で指定したファイルが終わりに達したかどうかを調べる関数です．関数の値として終りに達していれば真(－1)，そうでなければ偽(0)を返します.

〈ファイル番号〉で指定されたファイルはOPEN文のINPUTモードで開かれていなければなりません.

● 参　照　OPEN, INPUT #

ERASE *

イレース(ステートメント)

● 機　能　配列変数を削除する.

● 書　式　ERASE　〈配列変数名〉〔,〈配列変数名〉・・・〕

● 文　例　ERASE C, D

● 解　説　DIM 文で指定した配列変数を削除し，割り当てられていたメモリを解放します.

実行時にメモリが不足する場合，不要になった配列変数を削除してフリーメモリを増やし，メモリの不足を解消することができます．また，ERASE 文で削除した配列変数名は DIM 文を使って再宣言することができます.

ERASE 文を実行せずに同一変数名で宣言すると，"Redimensioned array" エラーが起こります.

● 参　照　DIM

ERL/ERR*

エラーライン／エラー(システム変数)

● 機　能　発生したエラーのエラーコードやエラーの発生した行番号を調べる.

● 書　式　ERL

ERR

● 文　例　L=ERL

E=ERR

● 解　説　通常はERL, ERR とも0です. エラーが発生した時点で, ERR はエラーコードを, ERL はエラーの発生した行番号を持っています. エラーがダイレクトモードの実行で発生した場合, ERL の値は65535 となります.

一般に ERR および ERL は ON ERROR GOTO 文によって指定したエラー処理ルーチンで, 処理の流れを制御するために使われます.

ERL を IF 文中で使用する場合, ERL は演算子の左辺におきます. これは, RENUM コマンドを実行したときに, 右辺の行番号もつけなおすことができるためです.

例)

```
IF  ERL=100  THEN 1000
```

● 参　照　ON ERROR GOTO, 資料「エラーメッセージ集」

ERROR*

エラー(ステートメント)

● 機　能　プログラムをエラー状態にする.

● 書　式　ERROR　〈エラーコード〉

● 文　例　ERROR 200

● 解　説　〈エラーコード〉の値は0から255までの範囲の数を指定します. 資料のエラーメッセージ表に記載している〈エラーコード〉はBASICで定義しているエラーで, これらの〈エラーコード〉を指定すると, 対応するエラーメッセージがプリントされ, コマンドレベルに戻ります.

また, BASICで, 用意していないユーザー独自のエラー処理を行いたい場合, ERROR文でエラーメッセージ表の"Unprintable error"となっているコードを指定してプログラムをエラー状態にし, ON ERROR GOTO文で指定するエラー処理ルーチン内で, 対応するエラー処理を行うことができます.

● 参　照　ON ERROR GOTO, ERL/ERR, 資料「エラーメッセージ表」

● サンプルプログラム

```
10 INPUT "ERROR No.=";A
20 ERROR A
```

EXP

エクスポーネンシャル(関数)

● 機　能　eに対する指数関数の結果を得る.

● 書　式　EXP (〈数式〉)

● 文　例　E=EXP (1)

● 解　説　自然対数の底eの〈数式〉乗を値とします.

〈数式〉の値は145.06286085862までです.

〈数式〉の値はどのような型でもかまいませんが, 得られる値は倍精度です.

● サンプルプログラム

```
10 PRINT " x     exp(x) log(exp(x))"
20 FOR I=.1 TO 5.5 STEP .1
30 PRINT USING"#.# ###.###";I;EXP(I);
40 PRINT LOG(EXP(I))
50 NEXT
```

FIX

フィックス(関数)

● 機　能　整数部を得る.

● 書　式　FIX (〈数式〉)

● 文　例　F=FIX (B/3)

● 解　説　〈数式〉の値の小数点以下を取り去った値を求めます.

INTとの違いはサンプルプログラムを見れば明らかです.

FIX (X) はSGN (X) * INT (ABS (X)) と同じ結果になります.

● 参　照　INT

● サンプルプログラム

```
10 FOR A=-1.2345 TO 2.2345 STEP .5
20 PRINT USING "##.####";A;
30 B=FIX(A)
40 PRINT USING "####";B;
50 B=INT(A)
60 PRINT USING "####" ;B
70 NEXT
```

FOR～NEXT

フォー～ネクスト(ステートメント)

● 機　能　FOR 文から NEXT 文までの一連の命令を指定回数だけ繰り返して実行する.

● 書　式　FOR〈変数名〉＝〈初期値〉　TO　〈終値〉〔STEP　〈増分〉〕

　　∫

　　NEXT〔〔〈変数名〉〕〔,〈変数名〉〕・・・・・〕

● 文　例　FOR J=0 TO 100 STEP 2

　　∫

　　NEXT J

● 解　説　〈変数〉はカウンタとして使われ，最初に初期値に設定されます．そして，FOR 文以降から NEXT 文までが実行され，カウンタの値は STEP によって指定された〈増分〉だけ増減されます．次にカウンタの値が〈終値〉と比べられ，カウンタの値が〈終値〉に達していなければ FOR 文の次の文へ戻り，同じ処理が繰り返されます．

〈初期値〉，〈終値〉，〈増分〉は数式で指定します．

STEP 文が省略された場合，〈増分〉は＋1とみなされます．また，〈増分〉は負の値をとることもできます．

次の場合には，FOR～NEXT 文は実行されずに，NEXT 文の次の文が実行されます．

●〈増分〉が正の値で〈初期値〉が〈終値〉より大きい場合．

●〈増分〉が負の値で〈初期値〉が〈終値〉より小さい場合．

●〈初期値〉と〈終値〉が同じ値で〈増分〉が0の場合．

ただし，〈変数〉には〈初期値〉が代入されています．

FOR～NEXT 文は入れ子構造にすることができます．これは1つの FOR～NEXT 文のなかに，さらに FOR～NEXT 文を置くことができるということで，この場合，それぞれの〈変数〉は別のものを使わなければなりません．また，**1つのFOR～NEXTは完全に他のFOR～NEXTの内部になければなりません.**

FOR～NEXT

例）

```
10  FOR I=0 TO ・・・
20  FOR J=1 TO ・・・
        ⋮
60  NEXT J
70  NEXT I
```

入れ子にした複数の FOR～NEXT 文が同一点で終わる場合は，NEXT 文を一つにまとめることができます．このときの NEXT 文には，一番内側の FOR 文で指定した〈変数名〉から，一番外側の FOR 文で指定した〈変数名〉までを順番に並べます．

例）

```
10  FOR I=0 TO ・・・
20  FOR J=1 TO ・・・
        ⋮
60  NEXT J, I
```

NEXT 文の〈変数名〉を省略すると，最も近くの FOR 文と応対しているものとみなされますが，入れ子構造の場合には混乱を防ぐために NEXT 文に〈変数名〉を指定した方がよいでしょう．

例）

```
10  FOR I=0 TO ・・・
20  FOR J=1 TO ・・・
        ⋮
60  NEXT
70  NEXT
```

FOR～NEXT

● サンプルプログラム

```
10 INPUT "バイスウ ヒョウジ(1－100)";A
20 FOR I=0 TO 100 STEP A
30 PRINT I,
40 NEXT I
50 PRINT
60 GOTO 10
```

FRE

フリー(関数)

● 機　能　メモリの未使用領域の大きさを調べる.

● 書　式　FRE (〈引数〉)

● 文　例　FRE (0)
　　FRE ("A")

● 解　説　FRE関数は利用可能なメモリ領域のうちの,未使用領域の大きさを関数値として持っています.〈引数〉はダミー引数ですので,定数,変数など,なにを指定してもかまいません.

FRE (〈数値〉) はテキスト (プログラム) 領域,変数領域などのユーザーエリアの未使用領域の大きさを値として持っています.

FRE (〈文字列〉) は文字列領域の未使用領域の大きさを値として持っています. また,FRE (〈文字列〉) では,文字列領域中の不用となった文字列を削除して,使用可能領域を広げる作業 (ガベージコレクション) を行います.

● 参　照　資料「メモリマップ」

● サンプルプログラム

```
20 PRINT "ｹﾞﾝｻﾞｲ";FRE(0);
30 PRINT "ﾊﾞｲﾄﾉ ﾌﾘｰ ﾒﾓﾘﾖ ﾍﾗｼﾃﾐﾏｽ"
40 FOR I=200 TO 1000 STEP 200
50 DIM A(I)
60 PRINT"DIM A(";I;")ﾃﾞ";FRE(0);"ﾊﾞｲﾄ ﾌﾘｰ"
70 ERASE A:NEXT
80 PRINT"ﾓｼﾞﾚﾂﾘｮｳｲｷﾉﾉｺﾘﾊ FRE(ﾓｼﾞﾚﾂ)ﾃﾞﾜｶﾘﾏｽ(
HIT ANY KEY !)"
90 I$=INPUT$(1):I$=""
100 FOR I=1 TO 5
110 A$=A$+"ABCD":PRINT A$;"・・";
120 PRINT FRE("");"ﾌﾘｰ"
130 NEXT
```

GET

ゲット(ステートメント)

● 機　能　　日付，時刻を変数に代入します

●書　式　　GET　DATE〈文字変数〉〔,A〕

　　GET　TIME 〈文字変数〉〔,A〕

● 文　例　　GET　DATE　DT$

　　GET　TIME　T$,A

● 解　説　　日付または時刻を文字変数に代入します.

日付は〝85／09／01〟のように〝／〟で区切られています.
　　年　月　日

時刻は〝13：27：06〟のように〝：〟で区切られています.
　　時　分　秒

オプションAを付けると，セットされているアラームの日付または時刻が得られます.

日付または，時刻のセットはSET文を参照してください.

● 参　照　　SET

● サンプルプログラム

```
10 CLS
20 GET TIME T$
30 LOCATE 5,5:PRINT T$
40 GOTO 20
```

現在の時刻を表示します．実行の終了は CTRL + STOP です.

〈参考〉・TIME文は，1/60秒毎の割り込み信号をカウントしているものであり，GET文とは直接的な関係はありません.

・GET文による日付，時刻は，装置の電源を切っていても，バックアップ電池により動作しています.

GOSUB

ゴーサブ(ステートメント)

● 機　能　サブルーチンを呼び出す.

● 書　式　GOSUB　〈行番号〉

　　⋮

　　RETURN〔〈行番号〉〕

● 文　例　GOSUB 100

● 解　説　〈行番号〉から始まるサブルーチンをGOSUB文によって呼び出して実行した後，サブルーチン内のRETURN文によって，GOSUB文の次の文へ戻ります.

サブルーチンとは他から独立した1つのプログラムで，GOSUBで指定する〈行番号〉から，RETURN文までの一連の命令のことをいいます．プログラムのあちこちで使う一連の命令をサブルーチンにしておくと，GOSUB文を使っていつでも何度でも呼び出す（コール）ことができますので便利です.

1つのサブルーチンの中から他のサブルーチンをコールすることもできます．これをサブルーチンの多重化と呼びます．この多重化はメモリのスタック領域の容量が許す限り行うことができ，もし，スタック領域が足りなくなった場合には，"Out of memory" エラーが起こります.

サブルーチンの中にCLEAR文があってはいけません.

サブルーチンは必ずGOSUB文によってコールしなければなりません．もし，GOTO文やRUN文で実行した場合，RETURN文実行時に，"RETURN without GOSUB" エラーが起こります.

1つのサブルーチン内には複数のRETURN文があってもかまいません.

RETURN文では〈行番号〉を指定して，条件に応じて特定の行へ強制的にリターンさせることができます.

● 参　照　RETURN, CLEAR

GOSUB

● サンプル プログラム

```
10 GOSUB 50
20 GOSUB 50
30 I=1:GOSUB 50
40 I=2:GOSUB 50
50 'ｻﾌﾞ ﾙｰﾁﾝ
60 PRINT"ｻﾌﾞ ﾙｰﾁﾝ ";
70 PRINT"I =";I
80 IF I=2 THEN RETURN 100
90 RETURN
100 END
```

GOTO

ゴートゥー(ステートメント)

● 機　能　指定した行へジャンプする.

● 書　式　1）GOTO 〈行番号〉

2）GO　TO　〈行番号〉

● 文　例　GOTO 500

● 解　説　〈行番号〉で指定した行へジャンプし，〈行番号〉以降の命令を実行します.

書式1），2）は全く同じ働きをしますが，"ＧＯ" と "ＴＯ" の間のスペースは1個のみです. 2個以上のスペースを入れた場合GOTO文とは解釈されません.

● サンプルプログラム

```
10 GOTO 40
20 PRINT"ｲｹﾏｾﾝ･":END
30 PRINT"ｺﾉﾖｳﾅ ";:GOTO 60
40 PRINT"ｺﾚﾊ";:GOTO 70
50 PRINT"ﾐﾎﾝﾃﾞｽ･":GOTO 80
60 PRINT"ﾌﾟﾛｸﾞﾗﾑ ﾖ":GOTO 90
70 PRINT"ﾜﾙｲ ";:GOTO 50
80 PRINT:GOTO 30
90 PRINT"ﾂｸｯﾃﾊ ";:GOTO 20
```

HEX$

ヘクサダラー(関数)

● 機　能　数値を16進表記の文字列に変換し，その結果を得る．

● 書　式　**HEX$ (〈数式〉)**

● 文　例　**PRINT HEX$ (13)**
実行結果　1101 が表示される．

● 解　説　〈数式〉の値を16進表記の文字列に変換し，その結果を得ます．〈数式〉の値の範囲は－32768から65535までです．負数の場合は65536にその値を加えた値と同じです．

変換結果の文字列はゼロサプレス(先頭の不用な0を取り除く)されています．

● 参　照　VAL, BIN$, OCT$,

● サンプルプログラム

```
10 PRINT "16ｼﾝﾃﾞｶｽﾞﾖ ｶｿﾞｴﾏｽ"
20 PRINT "16ｼﾝ     10ｼﾝ"
30 FOR I=0 TO 300
40 PRINT HEX$(I);"        ";I
50 FOR T=0 TO 300-I:NEXT
60 NEXT
70 FOR I=0 TO 10
80 PRINT " .            ."
90 NEXT
100 PRINT "FFFF       65535"
```

IF～THEN～ELSE/IF～GOTO～ELSE

イフ～ゼン～エルス／イフ～ゴートゥ～エルス(ステートメント)

● 機　能　条件判断を行う.

● 書　式

1）IF〈条件〉 THEN | 〈文〉 / 〈行番号〉 | 〔ELSE | 〈文〉 / 〈行番号〉 |〕

2）IF〈条件〉 GOTO 〈行番号〉 〔ELSE | 〈文〉 / 〈行番号〉 |〕

● 文　例　IF A$= "Y" THEN 200 ELSE 120

● 解　説　条件が成立するならばTHEN文またはGO TO文が実行され，条件が成立しないならば，ELSE文が実行されます．また，条件が成立しないときにELSE文がなければIF文の次の行が実行されます.

〈条件〉は1章の「式と演算」で説明した式で表現します．以下に式の使い方を紹介しましょう.

●数値や文字列の大小比較

数式や文字式で表した2つの数値や文字列を，比較演算子（=, <, >など）を使って比較します.

例)

```
IF A<236 THEN A=A+10
```

(変数Aの内容が236より小さかったらAの内容に10を加算します)

```
IF A$= "YES" THEN  100
```

(変数A$の内容が "YES" なら行番号100へジャンプ)

比較した結果，条件が成立しているときTHEN文かGOTO文が実行されます.

また，比較した結果，条件が成立しないときELSE文があればELSE文を実行し，ELSE文がなければ次の行が実行されます．比較演算子には次のような種類があります.

比較演算子	意　　味	例
〈式1〉　＝　〈式2〉	〈式1〉が〈式2〉と等しい.	A$＝B$
〈式1〉＜＞〈式2〉	〈式1〉が〈式2〉と等しくない.	A＋B＜＞C－1
〈式1〉　＜　〈式2〉	〈式1〉が〈式2〉より小さい.	A＜100
〈式1〉　＞　〈式2〉	〈式1〉が〈式2〉より大きい.	A＞B+C
〈式1〉＜＝〈式2〉 〈式1〉＝＜〈式2〉	〈式1〉が〈式2〉と等しいか, 〈式2〉より小さい.	A$＜＝B$＋" "
〈式1〉＞＝〈式2〉 〈式1〉＝＞〈式2〉	〈式1〉が〈式2〉と等しいか, 〈式2〉より大きい.	A$＝＞"A"

数値の大小比較は代数的に行われます．また文字列の大小比較は，それぞれの文字列の左端から順番に文字を比較し，右端の文字まで全部が同じなら2つの文字列は等しく，途中で等しくない文字があれば，その文字のキャラクタコードの大きい方の文字列が大きくなります．文字列の片方が短かくて比較が途中で終わる場合，短かい文字列の方が小さくなります．また，文字列中の空白には意味があります．

例）

```
"AA"        <     "AB"
"BASIC"     =     "BASIC"
"X&"        >     "X#"
"PEN "      >     "PEN"
"cm"        >     "CM"
"DESK"      <     "DESKS"
"ABC"       <     "AD"
```

●数値がノン0であるか否かの判断

数式の値がノン0であるか否かを判断します．

IF～THEN～ELSE/IF～GOTO～ELSE

例)

IF A THEN 150 ELSE 200

(変数Aの値が0ならば行番号200へジャンプ，0以外ならば150へ)

IF A+C THEN 200

(変数A，Cの加算の結果が0以外ならば行番号200へジャンプします)

式の値が0以外であれば，THEN文かGOTO文が実行されます．また，式の値が0であるとき，ELSE文があればELSE文を実行し，ELSE文がなければ次の行が実行されます．

●複合条件の判断

複数個の条件を論理的に演算子（NOT, AND, OR）を使って判断します．

例)

IF X<0 OR 99<X THEN 1000

(Xが負，または99より大きければ行番号1000へジャンプする)

IF 0<X AND X<100 THEN X=0

(Xが正で，かつ100より小さければXに0を代入する)

IF NOT (A=0) THEN 20

(Aが0でなければ，行番号20へジャンプする)

NOT, AND, ORを比較の優先順位が高い順に並べ，その意味を次頁の表に示します．

IF～THEN～ELSE/IF～GOTO～ELSE

論理演算子	意　　味	例
NOT〈条件〉	〈条件 〉が偽である	NOT（A＝0）
〈条件1〉 AND 〈条件2〉	〈条件1〉と〈条件2〉が共に真である	A＝0 AND B＝0
〈条件1〉 OR 〈条件2〉	〈条件1〉と〈条件2〉のどちらか，あるいは両方が真である	A＝0 OR B＝0

比較の優先順位を変えたいときはかっこ（　）を使ってください．

例）

```
IF A=0 AND (B=0 OR C=0) THEN 200
```

（A＝0かつB＝0，またはA＝0かつC＝0のときに行番号200へジャンプする）

●複文（マルチステートメント）の場合の実行

IF文を含む行が複文となっている場合は下表に示すように実行されます．

条件 / ELSE文	条件成立	条件不成立
あり	THEN文実行のみ	ELSE文実行に続いて後に続く文を次行
なし	THEN文実行に続いて後に続く文を実行	次の行を実行

例）IF X<0 THEN A=1 ELSE A=2:PRINT A

（PRINT Aは，X≧0のときのみELSE文に続いて実行される）

```
IF Y>0 THEN B=1:PRINT B
```

（PRINT Bは，Y＞0のときTHEN文に続いて実行される）

IF～THEN～ELSE/IF～GOTO～ELSE

● 入れ子構造のIF文

IF ～ THEN ～ ELSE 文では THEN 文や ELSE 文の中へさらに別の IF 文を置いて入れ子にすることができます．IF 文は何重にも入れ子にすることができますが，1行（255文字）に書ける範囲に限られます．

● 参　照　第1章「式と演算」「論理演算」

● サンプルプログラム

```
10 CLS:PRINT "*** サイノメ ヲ アテテクダ゛サイ ***"
20 DEFINT I,N,S
30 M=10000
40 PRINT " YOUR MONEY=";M
50 PRINT " $";M;"  カケキン イクラ ";:INPUT K
60 IF K<0 THEN PRINT"エッ!":GOTO 50
70 IF K>M THEN PRINT"アレ ソンナニ ナイヨ":GOTO
50
80 IF K=M THEN PRINT"スコ゛イ!" ELSE IF K>M
/2 THEN PRINT"イイノカナ-" ELSE IF K<M/100 T
HEN PRINT"シミッタレ !"
90 INPUT ," サイ ノ メ (1-6)";N
100 IF N<1 OR N>6 THEN PRINT"アキマヘン":GOT
O 90
110 S=RND(1)*6+1
120 PRINT"サイノ メハ";S;" テ゛シタ。"
130 IF N=S THEN M=M+K*3:PRINT" 、アタリ !!"
ELSE PRINT"ハズ゛レ":M=M-K
140 IF M<1 THEN PRINT"ハサンシマシタ。オツカレサマ。"
ELSE IF M>1000000! THEN PRINT"タ゛イ フコ゛ウ
ニナリマシタ。オメテ゛トウ" ELSE 50
```

INKEY$

インキーダラー(関数)

● 機　能　キーが押されていればその文字を，キーが押されなければヌルストリングを得る．

● 書　式　INKEY$

● 文　例　A$=INKEY$

● 解　説　キーが押されていない状態(キーボードバッファが空)であれば，INKEY$関数はヌルストリングを返します．
キーが押されている状態（キーボードバッファが空でない）であれば，バッファの先頭から1文字取り出し，その文字を返します．

なお，CTRL + C，CTRL + STOP はINKEY$関数によって拾うことはできません，押されたキーは画面上には表示されません．また，プログラム中，INKEY$の書かれた箇所でプログラムが停止することはありません．このため，キー入力待ちをするためには次のようにします．

```
100 A$=INKEY$ : IF A$= "" THEN 100
```

キー入力された文字を取り込む方法はINKEY$関数のほか，INPUT文，LINE INPUT文，INPUT$関数があります．

● 参　照　INPUT，LINE INPUT，INPUT$

● サンプルプログラム

```
10 SCREEN 2
20 IF INKEY$<>"" THEN 20 'ｷｰﾊﾞｯﾌｧ ｸﾘｱ
30 I$=INKEY$
40 IF I$="" THEN 30
50 A=ASC(I$)
60 IF A<28 OR A>31 THEN 30
70 ON A-27 GOTO 80,90,100,110
80 X=X+1:IF X>255  THEN X=1   :GOTO 120
ELSE 120
90 X=X-1:IF X<1    THEN X=255:GOTO 120
ELSE 120
100 Y=Y-1:IF Y<1   THEN Y=191:GOTO 120
ELSE 120
110 Y=Y+1:IF Y>191 THEN Y=1
120 PSET (X,Y)
130 GOTO 20
```

INP*

インポート(ステートメント)

● 機　能　入力ポートから値を得る.

● 書　式　INP (〈ポート番号〉)

● 文　例　A=INP (15)

● 解　説　〈ポート番号〉で指定された入力ポートから1バイト（8ビット）のデータを読み取り，それを関数値とします.〈ポート番号〉として指定できる値の範囲は0から255までです.

● 参　照　OUT

● サンプルプログラム

```
10 CLS
20 PRINT "PUSH SPACE.HOME.INS.DEL.CURSO
R KEYS !"
30 OUT 170,(INP(170)AND&HF0)OR8
40 LOCATE 10,10:PRINT RIGHT$("00000000"
+BIN$(INP(169)),8):GOTO 30
```

INPUT

インプット(ステートメント)

● **機　能**　キーボードからデータを入力し，指定した変数へ代入する．

● **書　式**　**INPUT** 〔"〈プロンプト文〉" ；〕〈変数名〉〔，〈変数名〉・・・〕

● **文　例**　**INPUT "Date";A$, B$**

● **解　説**　INPUT 文を実行すると，疑問符（？）と１桁のスペースが画面に表示され，プログラムはキーボードからの入力待ちの状態となります．以降，キーボードから入力されたデータが変数へ代入されます．

"〈プロンプト文〉"を指定していれば，疑問符の前にプロンプト文が表示されます．**〈変数名〉で指定した変数にはプロンプトが表示された後，リターンキーを押すまでの間にキーインされたデータが代入されます．**

〈変数名〉をカンマ（，）で区切って複数個指定した場合には，入力するデータもカンマで区切って〈変数名〉の数だけ入力します．また，対応する〈変数名〉とデータの型とは一致していなければなりません．もし，型が違っていた場合には "?.Redo from start" を表示して再び入力待ちとなります．入力するデータの個数が足りなければ"？？"を表示して入力待ちとなり，また，入力するデータの個数が多ければ，"？Extra ignored" が表示され余分に入力されたデータが無視されます．

何もせずにリターンキーを押した場合には変数の値はINPUT文を実行する前と同じです．この場合も〈変数〉が複数の場合，必要数のカンマだけは入力しなければなりません．

文字変数に文字列を代入する場合，カンマや文字列の前後の意味のある空白を入力したいときは，引用符（"）で文字列を囲む必要があります．この場合は文字として引用符を入力することはできません．これ以外の場合には，文字列を引用符で囲む必要はありません．

グラフィックモードでINPUT文を使用すると自動的にテキストモードになります．

INPUT

● サンプルプログラム

```
10 PRINT "INPUT ハ キーボ゛ート゛ カラ"
20 PRINT "モシ゛ヤ スウチヲ ニュウリョク スル"
30 PRINT "タメノ メイレイ テ゛ス"
40 INPUT "ワカリマシタカ(Y/N)";Y$
50 Y$=LEFT$(Y$,1)
60 IF Y$="Y" OR Y$="y" THEN 110
70 PRINT "テ゛ハマァ イロイロ ニュウリョクシテミテ クダ゛サイ"
80 PRINT "RETURN キーヲ ワスレズ゛ニ"
90 INPUT "A$,B";A$,B
100 PRINT"A$=";A$,"B=";B:GOTO 40
110 PRINT "ヨカッタ !"
```


INPUT#*

インプット シャープ(ステートメント)

● **機　能**　ファイルからデータを読み込む.

● **書　式**　INPUT #〈ファイル番号〉,〈変数〉〔,〈変数〉・・・〕

● **文　例**　INPUT #1, A, B

● **解　説**　あらかじめOPEN文でINPUTモードを指定して開いておいたファイルからデータを読み込み，変数にセットします．INPUT #文はデータを読み込む対象がファイルであることを除けば，INPUT文とほぼ同じです．

〈ファイル番号〉は，OPEN文で指定した番号です．

INPUT #文で読み込むデータはPRINT #文で書き出した各データと対応しています．

〈変数〉の型はファイルから入力するデータの型と対応しなくてはなりません．

● **参　照**　OPEN，PRINT #，第1章「ファイル」

INPUT$ *

インプット ダラー(関数)

● 機　能　指定されたファイルから指定された長さの文字を得る.

● 書　式　INPUT$ (〈文字数〉〔,〔#〕〈ファイル番号〉〕)

● 文　例　WORD$=INPUT$ (6, #2)

● 解　説　〈ファイル番号〉で指定されたファイルから〈文字数〉分の文字列を読み出します.〈ファイル番号〉が省略された場合には，キーボードから入力が行われますが，INPUT 文と異なり入力された文字は画面に表示されません.

INPUT$は指定された〈文字数〉の文字が入力されるのを待ち続けますが，既にバッファに入力済みのデータがある場合にはバッファ中から文字を拾ってきます. またINPUT$は，CTRL + STOP，CTRL + Cを除くすべての文字をそのまま読み出しますので，INPUT 文や LINE INPUT 文では入力することのできないラインフィードコードやリターンコードを入力することができます.

● 参　照　INPUT, LINE INPUT

● サンプルプログラム

```
10 'PASSWORD IS "MSX"+'ESC'key
20 P$="MSX"+CHR$(27)
30 PRINT"Password ";
40 A$=INPUT$(4):PRINT
50 IF A$<>P$ THEN PRINT " You are not a
 user !!":GOTO 30
60 BEEP:PRINT" Welcome to the MSX world
"
70 END
```

INSTR

インストリング(関数)

● 機　能　文字列の中から指定する文字列を捜してその文字の位置を得る.

● 書　式　INSTR ([〈数式〉,] 〈文字列 1〉, 〈文字列 2〉)

● 文　例　J=INSTR (A$, "J")

● 解　説　〈文字列１〉の中から〈文字列２〉を捜し，見つかれば，発見した位置を，見つからなければ０を値として持っています.

〈数式〉は捜し始める位置を０～255の範囲で指定します．もし省略した場合は〈文字例１〉の左側から捜し始めます.〈数式〉を省略する場合は,(,)も含めて省略します.

〈文字列１〉,〈文字列２〉は文字式で指定します.

〈文字列２〉の長さが〈文字列１〉より大きかったり，〈文字列１〉がヌルストリングであれば０を値として持っています.

● 参　照　LEN

● サンプルプログラム

```
10 '*** ﾋﾟｱﾉ ***
20 PRINT "ｵｵﾓｼﾞﾆｼﾃ,ｽｷﾅ ｷｰﾖ ｵｼﾃｸﾀﾞｻｲ"
30 PRINT "C=ﾄﾞ,F=ﾄﾞ#,V=ﾚ,G=ﾚ#..."
40 TB$="AZSXCFVGBNJMK,L./:Q2WE4R5T6YU8I
9OP-@^[¥"
50 PLAY "L8"
60 IN$=INKEY$:IF IN$="" THEN GOTO 60
70 I=INSTR(TB$,IN$)
80 IF I>0 THEN I=I+32
90 PLAY "N=I;"
100 GOTO 60
```

INT

インテジャー(関数)

● 機　能　小数点以下を切り捨てた整数値を得る.

● 書　式　**INT (〈数式〉)**

● 文　例　**PRINT INT (1.123)**
実行結果　1 が表示される.

● 解　説　〈数式〉 の値を越えない最大の整数を求めます.

FIX 関数との違いは FIX 関数の項を参照してください.

● 参　照　FIX, CINT

● サンプルプログラム

```
10 PRINT" X","INT(X)"
20 PRINT
30 FOR I=-2.4 TO 2.4
40 PRINT I,INT(I)
50 NEXT
60 END
```

INTERVAL ON/OFF/STOP *

インターバル オン／オフ／ストップ(ステートメント)

● 機　能　タイマ割込みを許可，禁止，または保留する．

● 書　式　**INTERVAL ON**
INTERVAL OFF
INTERVAL STOP

● 文　例　**INTERVAL ON**

● 解　説　ON INTERVAL GOSUB 文で設定したタイマ割込みを許可するか(ON)，禁止するか，(OFF)，保留するか（STOP）を指定します．

INTERVAL　ON 文はタイマ割込みを許可します．この命令を実行しておくと，ON INTERVAL GOSUB 文で設定した時間ごとに割込みがかかり，実行中のプログラムを中断してタイマ割込み処理ルーチンが実行されます．

INTERVAL OFF 文はタイマ割込みを禁止します．この命令を実行しておくと，ON INTERVAL GOSUB 文で設定した時間が経過しても割込みはかかりません．

INTERVAL STOP 文はタイマ割込みを保留します．この命令を実行しておくと，以後にかかったタイマ割込みは INTERVAL　ON 文を実行するまで保留され，INTERVAL ON を実行すると，直ちにタイマ割込み処理ルーチンが実行されます．この命令は INTERVAL ON 状態で実行する場合のみ有効です．

● 参　照　ON INTERVAL GOSUB

KEY＊

キー(コマンド)

● 機　能　キーボードの上部にあるプログラマブル・ファンクションキーの内容を定義する.

● 書　式　KEY 〈キー番号〉,〈文字列〉

● 文　例　KEY 1, "LOAD"

● 解　説　〈キー番号〉で指定するファンクションキーの内容を〈文字列〉で定義します．なお，定義した内容はKEY LIST文で確認することができます．ファンクションキーは全部で10あり，10個の〈文字列〉を記憶させておくことができます.

〈キー番号〉は1から10までで，それぞれ F1 ～ F10 に対応しています.

〈文字列〉は最大15文字までの文字およびコントロール文字です．キーボードから入力できない文字は，CHR$関数をプラス記号でつないで使います.

注意：一度定義したファンクションキーの内容は，あらたに定義し直すか，システムをリセットするまで変わりません.

KEY LIST

キーリスト(コマンド)

● 機　能　ファンクションキーの内容を画面に表示する.

● 書　式　KEY LIST

● 文　例　KEY LIST

● 解　説　ファンクションキーの内容の一部はテキスト画面の最終行に表示させておくことができますが，KEY LISTでは F1 ～ F10 のすべての内容を画面にリストします.

KEY(n)ON/OFF/STOP*

キー(エヌ)オン／オフ／ストップ(ステートメント)

● 機　能　ファンクションキーによる割り込みを許可，禁止，保留する．

● 書　式　KEY （〈キー番号〉）ON
KEY （〈キー番号〉）OFF
KEY （〈キー番号〉）STOP

● 文　例　KEY (1)　ON

● 解　説　ファンクションキーが押されたときにかかる割込みを許可するか (ON)，禁止するか (OFF)，保留するか (STOP) を設定します．

〈キー番号〉は，F1 から F10 までのプログラマブル・ファンクションキーの番号を表します．

KEY (n) ON は割込み動作を受付けます．この命令を実行すると，指定した〈キー番号〉のファンクションキーを押すごとに割込みがかかり，ON KEY GOSUB 文で定義した処理ルーチンにジャンプします．KEY (n) ON 状態ではファンクションキーを押してもキーの内容は画面には表示されません．

KEY (n) OFF は割込み動作を禁止します．この命令を実行すると，ファンクションキーを押しても処理ルーチンへはジャンプしません．

KEY (n) STOP は割込み動作を保留します．この命令を実行すると，ファンクションキーを押しても割込みは KEY (n) ON 文を実行するまで保留され，KEY (n) ON 文を実行すると，直ちに割込み処理ルーチンへジャンプします．この命令は KEY (n) ON 状態で実行する場合のみ有効です．

● 参　照　ON KEY GOSUB

KEY ON/OFF

キーオン／オフ(コマンド)

● 機　能　ファンクションキーの内容を画面最下行に表示するか，しないかを指定する．

● 書　式　KEY ON
　　KEY OFF

● 文　例　KEY ON
　　KEY OFF

●解　説　KEY ON はファンクションキーの内容を画面最下行に表示します．通常は F1 ～ F5 まで，シフトキーを押しているときは F6 ～ F10 までが表示されます．ファンクションキーを表示している間は画面に表示可能な行数が1行分減ります．

KEY OFF はファンクションキーの表示をとりやめます．

● 参　照　KEY

LEFT$

レフト ダラー(関数)

● 機　能　文字列の左側から指定した長さの文字列をとり出す．

● 書　式　LEFT$ (〈文字列〉, 〈数式〉)

● 文　例　B$＝LEFT$ (A$, 4)

● 解　説　〈文字列〉は文字式で表します．〈数式〉の値の範囲は0～255です．
〈数式〉が，〈文字列〉の総文字数以上の場合は，結果は〈文字列〉のすべてです．〈数式〉が0ならば，結果はヌルストリングです．
グラフィックキャラクタは1字あたり，2文字としてカウントします．

● サンプルプログラム

```
10 INPUT A$
20 FOR I=1 TO LEN(A$)
30 PRINT LEFT$(A$,I)
40 NEXT
50 END
```

LEN

レングス(関数)

● 機　能　文字列の総文字数を得る.

● 書　式　LEN (〈文字列〉)

● 文　例　PRINT LEN (A$)

● 解　説　LEN は表示されない文字や空白も含めた文字列の総文字数を数えます. 表示されない文字とは, キャラクタコードで1から31までの通常コントロールコードと呼ばれるものをいいます.
グラフィックキャラクタは1字あたり, 2文字としてカウントします.

● サンプルプログラム

```
10 INPUT "モジ゛レツ";A$
20 PRINT A$;" ハ";LEN(A$);"モジ゛テ゛ス。"
30 L=LEN(A$)
40 PRINT:PRINT STRING$(L+4,42)
50 PRINT"* ";A$:" *"
60 FOR I=1 TO L+4
70 PRINT"*";
80 NEXT
90 END
```

LET

レット(ステートメント)

● 機　能　変数に値を代入する.

● 書　式　〔LET〕〈変数名〉＝〈式〉

● 文　例　LET　A=10
A$="¥"+"10"

● 解　説　キーワード LET は, なくてもかまいません. つまり〈式〉の値を〈変数名〉に代入するときは, =記号だけでもよいのです.

〈式〉の内容は, 数値, 文字列のいかなる型であってもかまいません. ただし, 文字型から数値型への代入またはその逆は許されません. 型の異なる数値型変数への代入では左辺の型に変換されます. 型の異なる数値型変数への代入については第1章の型変換を参照してください.

LINE

ライン(ステートメント)

● 機　能　画面上に直線または四角形を描く．(グラフィックモード専用)

● 書　式　LINE〔 | (x1,y1) / STEP(x1, y1) |〕− | (x2, y2) / STEP(x2, y2) |〔, <カラーコード>〕〔, | B / BF |〕

〔,<論理演算子>〕

● 文　例　LINE (100, 100)−(135,100), 8

● 解　説　指定した2点,（ｘ1，ｙ1）と（ｘ2，ｙ2）を結ぶ直線を描きます．
（ｘ1，ｙ1）を省略した場合，ＬＰ（最終参照点）の値が採用されます．座標の指定にSTEPを付けるとＬＰからの相対座標による指定になります．
LINE文を実行した場合，ＬＰは（ｘ2，ｙ2）に移動します．

〈カラーコード〉は直線や四角形の色を指定します.〈カラーコード〉を省略すると，現在COLOR文によって設定されている前景色が採用されます．

Ｂは Boxの意味で，Ｂを指定すると（ｘ1，ｙ1）と（ｘ2，ｙ2）の2点を対角とする四角形を描きます．
ＢＦは Box Fillの意味で，ＢＦを指定すると（ｘ1，ｙ1）と（ｘ2，ｙ2）の2点を対角とする四角形を描き，その内側を塗りつぶします．

〈論理演算子〉を指定すると，描かれる線や塗りつぶしの色を変えることができます．ただし，この指定ができるのはSCREEN 5 ~8だけです．
指定色のカラーコードをC.描かれる部分の元の色のカラーコードをA.とすると,〈論理演算子〉を付加することにより，描かれる色は次のように変わります.(この論理演算子による操作は，COPY, PSET, PUT KANJI 命令などにおいても同様です.)

論理演算子	描かれる色	例C＝6 A＝3の時)
AND	C AND A	0110 AND 0011＝0010→2
OR	C OR A	0110 OR 0011＝0111 ⟶7
XOR	C XOR A	0110 XOR 0011 ＝0101→5
PSET	C	0110 ⟶6
PRESET	$\overline{C}$	0110＝1001 ⟶9

LINE

SCREEN 8では，前記演算が0～255のカラーコードに相当する8ビットに対して，実行されます．

● サンプルプログラム

```
10 '*** □を おく ***
20 '***  ␣      ***
30 SCREEN 5
40 FOR J=0 TO 93 STEP 3
50 LINE(128-J,95-J)-(128+J,95+J),15,B
60 NEXT J
70 LINE(35,2)-(221,188),15
80 LINE(35,188)-(221,2),15
90 GOTO 90
```

〈論理演算子〉による操作の様子を表示します．上の帯から論理演算子の指定なし，AND，OR，XOR，PSET，PRESETの順に表示されます．60行のCのカラーコードを変えて実行してみてください．

```
10 '*** ろんりえんざんし ***
20 SCREEN 5
30 FOR A=0 TO 15
40 LINE(A*16,0)-(A*16+15,211),A,BF
50 NEXT A
60 C=6
70 LINE(0,30)-(255,59),C,BF
80 LINE(0,60)-(255,89),C,BF,AND
90 LINE(0,90)-(255,119),C,BF,OR
100 LINE(0,120)-(255,149),C,BF,XOR
110 LINE(0,150)-(255,179),C,BF,PSET
120 LINE(0,180)-(255,211),C,BF,PRESET
130 GOTO 130
```

LINE INPUT*

ライン インプット(ステートメント)

● **機　能**　キーボードから1行分全体のデータを区切ることなく入力し，文字型変数へ代入する．

● **書　式**　**LINE INPUT〔"〈プロント文〉";〕〈文字型変数名〉**

●**文　例**　**LINE INPUT "NAME";NA$**

● **解　説**　LINE INPUT 文を実行すると，プログラムはキーボードからの入力待ちの状態となります．**疑問符（?）は表示されません．以降，キャリッジリターンを押すまでに入力された文字列が〈文字型変数〉に代入されます．**

LINE INPUT文はINPUT文と異なり，カンマ（,）によってデータを区切り，複数の変数に文字列を代入することはできません．

"〈プロンプト文〉"を指定していれば入力待ちの状態となる前にプロンプト文が画面に表示されます．

キーボードから入力できる文字列の長さは254文字以内です．

● **参　照**　LINE INPUT #, INPUT

● **サンプルプログラム**

```
10 PRINT "LINE INPUT ﾊ ";CHR$(34);" ﾔ "
;CHR$(44);"ﾖ ﾆｭｳﾘｮｸﾃﾞｷﾏｽ。"
20 PRINT "ﾓｼﾞﾚﾂ ﾖ ﾆｭｳﾘｮｸｼﾃｸﾀﾞｻｲ。";
30 LINE INPUT A$
40 PRINT "ｺﾞﾗﾝﾉ ﾄｳﾘﾃﾞｽ。"
50 PRINT "A$=";A$
```

LINE INPUT #*

ライン インプット シャープ(ステートメント)

● 機　能　１行単位のデータをファイルから文字変数へ読み込む.

● 書　式　LINE INPUT　#〈ファイル番号〉,〈文字型変数名〉

● 文　例　LINE INPUT　#1, A$

● 解　説　あらかじめOPENでINPUTモードを指定して開いておいたファイルからキャリッジリターンコードまたは，キャリッジリターンコード＋ラインフィードコードで区切られている１行単位（254文字以内）のデータを指定した文字変数へ読み込みます.

注）ラインフィードコード＋キャリッジリターンコードの組み合わせは区切りコードとみなされません.

SAVE文で作成したアスキー形式のプログラムファイルには，一行一行がキャリッジリターンコード（CHR$(13)）とラインフィードコード（CHR$(10)）とで区切られて書き出されています．また，PRINT#文で作成したファイルにはPRINT#文を実行するごとに数値や文字列の並びがキャリッジリターンコード＋ラインフィードコードで区切られて書き出されています．LINE INPUT#文ではこれらのファイルからキャリッジリターン＋ラインフィードコードにはさまれているデータを読み込むときなどに使います.

〈ファイル番号〉はOPEN文で指定した番号です.

なお，キャリッジリターン＋ラインフィードではさまれているデータが254文字以上であれば，254文字までが文字変数に代入されます.

● 参　照　OPEN, CLOSE, PRINT#, 第１章「ファイル」

LIST/LLIST

リスト／エル リスト(コマンド)

● 機　能　メモリ上にあるプログラムの全部または一部を画面やプリンタにリストアウトする.

● 書　式　**LIST〔〈行番号〉〕〔−〈行番号〉〕**
LLIST〔〈行番号〉〕〔−〈行番号〉〕

● 文　例　**LIST 100−200**
LIST

● 解　説　LIST は画面にプログラムを表示し，LLIST はプログラムをプリンタにリストアウトします．このコマンドを実行し終わると，コマンドレベルにもどります.

〈行番号〉を省略した場合は，プログラムの先頭から最後まで全部がリストアウトされます.

〈行番号〉を指定すると，その行だけがリストアウトされます.

〈行番号〉−を指定すると，指定した行番号以降のプログラムがすべてリストアウトされます.

〈行番号〉−〈行番号〉を指定すると，1番目に指定した行番号から2番目に指定した行番号までのプログラムがリストアウトされます.

−〈行番号〉を指定すると，プログラムの先頭から指定した行番号までのプログラムがリストアウトされます.

〈行番号〉の代わりに "." (ピリオド) を指定すると BASIC が現在注目している行（スクリーンエディタや LIST 文などで最後に参照した行）を指します.

リストアウトの途中で STOP キーを押すとリストアウトは中断し，再び STOP キーを押すと，リストアウトを再開します．また，リストアウトの途中で CTRL ＋ STOP キーを押すと，リストアウトは終了します.

LOAD

ロード(コマンド)

● 機　能　アスキーファイルとして作成してあるプログラムをメモリ上にロードする.

● 書　式　LOAD "〈ファイルスペック〉"〔,R〕

● 文　例　LOAD "CAS:TEST"

● 解　説　〈ファイルスペック〉で指定したプログラムをメモリ上にロードします. ロードするプログラムはSAVE文を使ってアスキー形式のファイルとして作成されたものです. LOAD文を実行すると, LOAD文実行前にメモリ上にあったプログラムは消去されます. また, すべての開いているファイルが閉じられ, 変数の値は初期化されます.

〈ファイルスペック〉は"〈デバイス名〉:〈ファイル名〉"です. デバイス名には"CAS:"を指定します. また, ファイル名を省略すると, カセットテープの位置している場所以降にある最初のアスキー形式のファイルがロードされます.

Rオプションをつけると, ファイルは開いたままでプログラムをロード後, だだちに実行を開始します.

LOADは, 指定されたファイルを見つけてプログラムのロードを開始するまでは, メモリ上のプログラムを保存します.

● 参　照　SAVE, MERGE, CLOAD

LOCATE

ロケート(ステートメント)

● 機　能　テキスト画面のカーソルの位置を指定する．(テキストモード専用)

● 書　式　**LOCATE　〔〈X座標〉〕〔，〈Y座標〉〕〔，〈カーソルスイッチ〉〕**

● 文　例　**LOCATE 10, 10**

●解　説　カーソルをテキスト画面のキャラクタ座標（X，Y）の位置へ移動します．

この命令は，PRINT 文で表示する文字や INPUT 文で表示するプロンプトの開始位置を指定するときなどに用います．

〈X座標〉は画面の左端を0とする水平座標を表す数式で，省略した場合は0とみなされます．
〈Y座標〉は画面の上端を0とする垂直座標を表す数式で，省略した場合は現在カーソルがセットされている行とみなされます．

〈カーソルスイッチ〉は画面上のカーソルの表示状態を決めるスイッチです．

0：カーソルはキー入力待ちのとき以外表示されない

1：カーソルは常に表示される

● サンプルプログラム

```
10 CLS
20 FOR I=0 TO 30
30 X=INT(RND(1)*27)
40 Y=INT(RND(1)*21)
50 LOCATE X, Y:PRINT "ココ";
60 LOCATE 10, 22:PRINT USING "LOCATE ##. ##";X;Y;
70 FOR J=0 TO 700:NEXT
80 LOCATE X, Y:PRINT SPACE$(12);
90 NEXT
100 END
```

LOG

ログ(関数)

● **機　能**　自然対数を得る.

● **書　式**　**LOG (〈数式〉)**

● **文　例**　**PRINT LOG (35/9)**

● **解　説**　〈数式〉で指定した値の**自然対数 (e を底とした対数)** を求めます.〈数式〉は0より大きくなければなりません.

〈数式〉の型はどのような型でもかまいませんが，結果は倍精度実数で得られます.

● **参　照**　EXP

● **サンプルプログラム**

```
10 '*** LOG キョクセン **********
20 SCREEN 2
30 FOR I=.1 TO 25 STEP .1
40 Y=LOG(I)/3.2*100
50 LINE(0,115)-(255,115),7
60 PSET(20+X,115-Y),7
70 X=X+1
80 NEXT
90 GOTO 90
```

LPOS*

エルポジション(システム変数)

● **機　能**　プリンタのヘッド位置を調べます.

● **書　式**　**LPOS (〈式〉)**

● **文　例**　**LPOS (0)**

● **解　説**　得られる値はプリンタバッファ上のプリンタヘッドの位置で，必ずしも物理的なプリンタヘッドの位置とは限りません.

〈式〉はダミーの引数で，変数か定数であれば何であってもかまいません.

● **参　照**　WIDTH, LPRINT

LPRINT/LPRINT USING

エル プリント／エルプリント ユージング(ステートメント)

● 機　能　文字列や数値を指定した書式でプリンタに出力する.

● 書　式　LPRINT〔〈式〉・・・〕

LPRINT USING〈書式〉; 〈式〉・・・

● 文　例　LPRINT A$, B

LPRINT USING "@####. ##";A$;B

● 解　説　〈式〉で表す文字列や数値をプリンタに出力します. 複数個の〈式〉を指定する場合は〈式〉の間をカンマ, またはセミコロンで区切ります. LPRINT 文, LPRINT USING 文は, 出力先がプリンタであることを除けば, その機能は PRINT 文や PRINT USING 文とまったく同じですので, そちらを参照してください.

● 参　照　PRINT, PRINT USING

MAXFILES

マックス ファイルズ(ステートメント)

● 機　能　使用するファイルの数を定義する.

● 書　式　MAXFILES＝〈ファイル数〉

● 文　例　MAXFILES＝3

● 解　説　OPEN 文で開いて使用するファイルの数を定義します. 以降, 指定した数のファイルを同時に開いて使用することができます.

〈ファイル数〉は0～15の範囲の値を持つ式です.
ファイル1個について267バイトのファイルコントロールブロック領域が確保されますので, 必要なファイル数だけをプログラムの先頭で指定して下さい. プログラムの途中で指定するとそれまでに定義したデータが消えることがあります.
電源オン時はMAXFILES＝1となっています.

MERGE*

マージ(コマンド)

● 機　能　アスキーファイル上のプログラムをメモリ上のプログラムにマージ（混合）する.

● 書　式　MERGE "〈ファイルスペック〉"

● 文　例　MERGE "CAS:TEST"

● 解　説　メモリ上のプログラムに〈ファイルスペック〉で指定したプログラムファイルを混合して1つのプログラムにし，メモリ上に置きます.

〈ファイルスペック〉で指定するファイルはSAVE文を使ってアスキー形式でセーブされていなければなりません．そうでない場合には，エラーとなります.

〈ファイルスペック〉は"CAS:〈ファイル名〉"です．また，〈ファイルスペック〉のファイル名を省略すると，カセットテープの位置している場所以降にある最初のアスキーファイルがロードされます.

ファイル中のプログラムと，メモリ中のプログラムに同一の行番号があった場合には，ファイル中の行でメモリ中の行を置き換えます.

MERGEコマンドは実行を終了すると，コマンドレベルに戻ります.

● 参　照　SAVE, LOAD

MID$

ミドル ダラー(関数)

● 機　能　文字列の中から指定した長さの文字列をとり出す.

● 書　式　MID$ (〈文字列〉, 〈式 1〉〔, 〈式 2〉〕)

● 文　例　PRINT　MID$ (A$, 2, 3)

● 解　説　〈文字列〉の左から〈式 1〉番目の文字から〈式 2〉文字の文字列をとり出します. 〈式 2〉の値は 0 から255の範囲, また, 〈式 1〉の値は 1 から255の範囲です.

〈式 2〉を省略した場合や, 〈文字列〉の〈式 1〉番目の文字から右の文字数が〈式 2〉より小さい場合は, 〈文字列〉の〈式 1〉番目より右すべての文字列が結果となります.

〈文字列〉の文字数が〈式 1〉より小さければ, とり出す文字列はヌルストリングとなります.

グラフィックキャラクタを含む場合は, 文字コードが2バイトになっている点を考慮して〈式1〉, 〈式2〉の値を指定してください.

● 参　照　RIGHT$, LEFT$, MID$（ステートメント), LEN

● サンプルプログラム

```
10 '*** ﾀﾝｺﾞ ﾜｹ **************
20 A$=" ﾊｼｶﾞ ｺﾛｹﾞﾃ ﾀﾇｷｶﾞ ﾜﾗｳ"
30 L=LEN(A$):N=1
40 FOR I=1 TO L
50 S=INSTR(N+1,A$," ")
60 IF S=0 THEN S=L+1
70 PRINT MID$(A$,N,S-N)
80 I=S:N=S:NEXT
90 INPUT "ｵｽｷﾅ ﾌﾞﾝｼｮｳﾖ ﾄﾞｳｿﾞ";A$
100 A$=" "+A$:GOTO 30
```

MID$*

ミドル ダラー(ステートメント)

● 機　能　文字列の一部を置き換える.

● 書　式　MID$(〈文字変数1〉, 〈式1〉〔,〈式2〉〕)＝〈文字列2〉

● 文　例　MID$ (A$, 3)＝"ABC"

実行結果

A$＝"アイウエオカ"の場合　A$＝"アイABCカ"となる.

● 解　説　〈文字変数1〉の〈式1〉番目の文字から〈式2〉個の文字を,〈文字列2〉の最初から〈式2〉個の文字列で置き換えます.

〈文字変数1〉,〈文字列2〉は文字式で表します. また,〈式1〉,〈式2〉は数式で表します.

〈式2〉を省略した場合, または〈文字列2〉の文字数より多く指定した場合は,〈文字列2〉のすべての文字と置き換わります.

〈式1〉の値は1以上で,〈文字変数1〉の文字数以下の値です.

グラフィックキャラクタは, 文字コードが2バイトになっている点を配慮して〈式1〉,〈式2〉の値を指定してください.

● 参　照　MID$関数, LEN

● サンプルプログラム

```
10 A$="ABCDEFG"
20 B$="HIJK"
30 PRINT A$, B$
40 PRINT
50 FOR I=1 TO 7
60 C$=A$
70 MID$ (C$, I, 1)=B$
80 PRINT C$, I
90 NEXT
```

MOTOR

モータ(ステートメント)

● 機　能　　カセットテープレコーダのモータ ON/OFF を制御する.

● 書　式　　MOTOR 〔 ON / OFF 〕

● 文　例　　MOTOR ON

● 解　説　　ON を指定すればモータは ON となり，OFF を指定すればモータは OFF となります．また，ON/OFF を指定しない場合，現在モータが ON の状態ならばOFF に，OFF の状態ならば ON にします．
MOTOR コマンドはリモート端子付のカセットレコーダでカセットテープを巻きもどす時などに使用します.

● サンプルプログラム

```
10   'ｺﾚﾊ ｱﾏﾘ ﾖｲｺﾄﾃﾞﾊ ｱﾘﾏｾﾝ
20 FOR I=1 TO 100
30 MOTOR
40 FOR J=0 TO 10
50 NEXT J,I
60 END
```

NEW

ニュー(コマンド)

● 機　能　メモリ上にあるプログラムを削除し，すべての変数をクリアする．

● 書　式　NEW

● 文　例　NEW

● 解　説　NEWコマンドはすべての変数をクリア(初期状態にする)し，開かれているファイルがあれば，それも自動的にすべて閉じます．また，DEF FN文による定義も解消されます．
NEWコマンドは，コマンドレベルにあるとき，新しいプログラムを入力する前に実行します．NEWコマンドを実行するとコマンドレベルに戻ります．

● 参　照　DELETE, ERASE

● サンプルプログラム

```
10 'ｺﾉ ﾌﾟﾛｸﾞﾗﾑﾊ RUN ｽﾙﾄ ｼﾞﾄﾞｳｼｮｳﾒﾂ ｼﾏｽ
20 A$="サ ヨ ウ ナ ラ!"
30 FOR I=1 TO 10
40 BEEP
50 PRINT MID$(A$,I,1);
60 FOR J=0 TO 100:NEXT
70 NEXT
80 NEW
```

OCT$

オクタルダラー(関数)

● 機　能　数値を8進表記の文字列に変換し，その結果を得る．

● 書　式　OCT$　(〈数式〉)

● 文　例　LPRINT OCT$ (10)

実行結果　12をプリンタに出力します．

● 解　説　〈数式〉の値を8進表記の文字列に変換し，その結果を得ます．〈数式〉の値の範囲は−32768から65535までです．負数の場合は65536にその値を加えた値と同じです．

● サンプルプログラム

```
10 I=100
20 PRINT "   X   OCT$(X) HEX$(X)"
30 PRINT I;"  ";OCT$(I);"       ";HEX$(I)
40 END
```

ON ERROR GOTO*

オン エラー ゴートゥー(ステートメント)

● 機　能　エラー処理ルーチンの開始行を定義する.

● 書　式　ON ERROR GOTO〈行番号〉

● 文　例　ON ERROR GOTO 1000

● 解　説　この命令を実行しておくと，エラーが発生したときに割り込みがかかり，指定したエラー処理ルーチンへジャンプします．もし〈行番号〉の行がプログラム上に存在しなければ" Undefined line" エラーが起こります.

通常，エラーが発生すると，BASIC で用意しているエラーメッセージが表示され，プログラムの実行を終了し，コマンドレベルに戻ります．しかし，ON ERROR GOTO 文を実行してあると，エラーが発生したときに BASIC で用意しているエラー処理は行われず，〈行番号〉で指定したエラー処理ルーチンへジャンプします.

〈行番号〉で始まるエラー処理ルーチンでは任意のエラー処理を行い，RESUME 文でエラー回復処理をして終了します．このエラー処理ルーチンでは ERR 関数や ERL 関数を使ってエラーコードやエラーの発生した行番号を知ることができます.

ON ERROR GOTO 0 を実行すると，ON ERROR GOTO〈行番号〉での定義を無効にし，以降，エラーが発生すれば，BASIC で用意しているエラー処理が行われます.
エラー処理ルーチンに ON ERROR GOTO 0 の文がある場合には，BASIC はエラーメッセージを表示し，プログラムの実行を停止します.

ON ERROR GOTO 文の効果は，以降，別の ON ERROR GOTO 文，RUN 文か CLEAR 文を実行するまでです．このため，プログラム中で，エラー処理ルーチンの開始行を定義している場合，プログラムの終了前に ON ERROR GOTO 0 文を実行してください.
エラー処理ルーチンでは，独自のエラー処理を行う場合以外，ON ERROR GOTO 0 を実行することをお勧めします.

ON ERROR GOTO*

注意：エラー処理ルーチンを実行中にエラーが起きた場合には，それに対するエラーメッセージが表示され，実行が停止されます．エラー処理サブルーチンの中ではエラー割り込みは起こりません．

● **参　照**　RESUME, ERROR, ERR, ERL,第1章「割込み」

● **サンプルプログラム**

```
10 '*** エラー ショリ ノ レイ *******
20 ON ERROR GOTO 80
30 INPUT "1カラ 999ノ カズヲ ニュウリョクシテクダサイ
";A
40 PRINT "ニュウリョクサレタ カズハ";A
50 IF A<1 OR A>1000 THEN ERROR 26
60 GOTO 30
70 '*** エラーショリ ルーチン ********
80 IF ERR<>26 THEN 120
90 PRINT "タダシイカズヲ ニュウリョクシテクダサイ"
100 PRINT "タダイマ エラーカ オキタキョウハ ";ERL
110 RESUME NEXT
120 ON ERROR GOTO 0
```

ON GOTO/ON GOSUB

オン ゴートゥー／オン ゴーサブ(ステートメント)

● **機　能**　指定されたいずれかの行にジャンプする.

● **書　式**　**ON 〈式〉 GOSUB 〈行番号〉〔, 〈行番号〉・・・〕**

ON 〈式〉 GOTO 〈行番号〉〔, 〈行番号〉・・・〕

● **文　例**　**ON A GOSUB　100, 200, 300, 400**

ON A GOTO　100, 200, 300, 400

● **解　説**　〈式〉の値に応じて，指定した〈行番号〉のいずれかへジャンプします．〈行番号〉の並びは左側から順番に1から始まる数に対応します．たとえば，〈式〉の値が3であったなら，左から3番目の〈行番号〉へジャンプします．

ON　GOSUB文では，〈行番号〉はジャンプするサブルーチンの最初の行の行番号です．

〈式〉の値が負になった場合には，"Illegal function call" エラーが起こりますが，値が0または〈行番号〉の個数より大きくなった場合には，ON GOSUB/ON　GOTO文の次の行へプログラムの制御が移り，エラーは起こりません．

● **参　照**　GOSUB, GOTO

● **サンプルプログラム**

```
10 PRINT "123 ノ キーヲ オシテクダ゛サイ"
20 A=VAL(INKEY$)
30 ON A GOTO 50,60,70
40 GOTO 20
50 PLAY "O4C","E","G":GOTO 20
60 PLAY "O4C","F","A":GOTO 20
70 PLAY "O3B","D","G":GOTO 20
```

ON INTERVAL GOSUB*

オン インターバル ゴーサブ(ステートメント)

● 機　能　タイマ割込み条件を設定し，割込みルーチンの開始行を定義する．

● 書　式　ON INTERVAL＝〈時間〉 GOSUB 〈行番号〉

● 文　例　ON INTERVAL＝60 GOSUB 100

● 解　説　この命令は，割込みがかかる時間間隔を設定し，割込みがかかったときにジャンプする処理ルーチンの開始行番号を定義します．以後，INTERVAL ON状態にしておくと，指定した〈時間〉が経過するごとに割込みがかかり，割込み処理ルーチンが実行されます．

〈時間〉は1／60秒単位で，1～65535までの数値です．

〈行番号〉は割込み処理ルーチンの開始行番号です．

割込み処理ルーチンからの復帰は，GOSUB文で呼び出すサブルーチンと同じようにRETURN文を使います．

割込み処理ルーチンを実行している間，自動的にINTERVAL STOP状態になり，RETURN文実行とともにINTERVAL ON状態に戻ります．

割込みはプログラムモードでのみ有効です．また，ON ERROR文で定義するエラー割込み処理中，あるいは他の割込み処理中（STRIG, STOP, SPRITE, INTERVAL, KEY）では，これらの処理ルーチンが終了するまで割込みは保留されます．

注意：この命令は割込みの定義をするだけで，実行した時に割込みがかかるわけではありません．

● 参　照　INTERVAL ON/OFF/STOP，第1章「割込み」

ON INTERVAL GOSUB*

● サンプルプログラム

```
10 ON INTERVAL=100 GOSUB 80
20 N=20:CLS:COLOR 2
30 LOCATE 5,5:PRINT"TIME"
40 INTERVAL ON
50 LOCATE 5,7:PRINT SPACE$(128)
60 LOCATE 5,7:PRINT STRING$(N,"*")
70 GOTO 60
80 '
90 N=N-1
100 IF N<0 THEN END
110 IF N<10 THEN COLOR 10
120 IF N<5  THEN COLOR  8
130 RETURN 50
```

ON KEY GOSUB*

オン キー ゴーサブ(ステートメント)

● 機　能　ファンクションキーによる割込み処理ルーチンの開始行を定義する.

● 書　式　ON KEY GOSUB〔〈行番号〉〕〔,〈行番号〉・・・〕

● 文　例　ON KEY GOSUB 100, 200, 300, 400

● 解　説　ファンクションキーが押されたときにジャンプする処理ルーチンの開始行を定義します.

この命令は，ファンクションキーが押されて割込みがかかったときに，どの割込み処理ルーチンにジャンプするかを定義します．以後，KEY (n) ON 状態にしておけば，ファンクションキーが押されるたびに割込み処理ルーチンが実行されます.

〈行番号〉は，割込みがかかったときにジャンプする処理ルーチンの開始行番号で，この並びの順序は，ファンクションキーのキー番号と1対1に対応しています．MSX のファンクションキーは F1 から F10 までありますから，最大10個の〈行番号〉を並べて書くことができます.

割込み処理ルーチンからの復帰は GOSUB 文で呼び出すサブルーチンの場合と同じように RETURN 文を使います.

注意：この命令は割込みの定義をするだけで，実行した時に割込みがかかるわけではありません.

● 参　照　KEY ON/OFF/STOP，第1章「割込み」

ON KEY GOSUB*

● サンプルプログラム

```
10 ON KEY GOSUB 80,90,100,110,120,130,1
40,150,160,170
20 KEY(1) ON:KEY(2) ON
30 KEY(3) ON:KEY(4) ON
40 KEY(5) ON:KEY(6) ON
50 KEY(7) ON:KEY(8) ON
60 KEY(9) ON:KEY(10) ON
65 PRINT "ファンクション　キー　ヲ　オシテクダサイ"
70 GOTO 70
80 N=1:GOTO 180
90 N=2:GOTO 180
100 N=3:GOTO 180
110 N=4:GOTO 180
120 N=5:GOTO 180
130 N=6:GOTO 180
140 N=7:GOTO 180
150 N=8:GOTO 180
160 N=9:GOTO 180
170 N=10
180 PRINT USING" F･## キー ヲ オシマシタネ";N
190 RETURN
```

ON SPRITE GOSUB

オン スプライト ゴーサブ(ステートメント)

● **機　能**　スプライトが重なったときにジャンプする割込み処理ルーチンの開始行を定義する.

● **書　式**　**ON SPRITE GOSUB 〈行番号〉**

● **文　例**　**ON SPRITE GOSUB 100**

● **解　説**　この命令を実行すると，以後，SPRITE　ON 状態にしておけば，PUT SPRITE で画面に描いたスプライトが他のスプライトと重なったときに割込みがかかり，プログラムの実行を中断して〈行番号〉から始まる割込み処理ルーチンが実行されます.

割込み処理ルーチンからの復帰は，GOSUB 文で呼び出すサブルーチンと同じように RETURN 文を使います.

割込み処理ルーチンを実行している間，自動的に SPRITE STOP 状態になり，RETURN 文実行とともに SPRITE ON 状態になります.

この命令はプログラムモードでのみ有効です．また，ON ERROR GOTO 文，ON KEY GOSUB，または ON STOP GOSUB 文で定義する割込み処理を実行中は，これらの処理ルーチンが終了するまで割込みは保留されます.

● **参　照**　SPRITE ON/OFF/STOP，PUT SPRITE，第1章「割込み」

ON SPRITE GOSUB

● サンプルプログラム

```
10 '*** ショウトツ ********************
20 'A$,B$ ヲ セイカクニ ニュウリョク シテクダサイ
30 A$="♥~くくく~ズく"
40 B$="くズ~くくく~♥"
50 SCREEN 2,1
60 ON SPRITE GOSUB 160
70 SPRITE$(0)=A$
80 SPRITE$(1)=B$
90 SPRITE ON
100 XA=RND(1)*256
110 XB=RND(1)*256
120 FOR Y=0 TO 191
130 PUT SPRITE 0,(XA,Y),6
140 PUT SPRITE 1,(XB,191-Y),3
150 NEXT:GOTO 90
160 SPRITE OFF
170 PLAY "L4CDEFEDCREFGAGFER"
180 IF PLAY(0) THEN 180
190 PUT SPRITE 0,(0,208)
210 Y=191:RETURN
```

ON STOP GOSUB*

オン ストップ ゴーサブ(ステートメント)

● 機　能　CTRL + STOP キーによる割込み処理ルーチンの開始行を定義する。

● 書　式　ON STOP GOSUB〈行番号〉

● 文　例　GO STOP GOSUB 100

● 解　説　この命令を実行すると，以後，STOP ON 状態にしておけば，CTRL + STOP キーが押されたときに割込みがかかり，プログラムの実行を中断して〈行番号〉から始まる割込み処理ルーチンが実行されます．

割込み処理ルーチンからの復帰は GOSUB 文で呼び出すサブルーチンと同じように RETURN 文を使います．

割込み処理ルーチンを実行している間，自動的に STOP STOP 状態になり，RETURN 文実行とともに　STOP ON 状態になります．

この命令はプログラムモードでのみ有効です．また，ON ERROR GOTO 文，ON KEY GOSUB 文で定義する割込み処理を実行中は，これらの処理ルーチンが終了するまで割込みは保留されます．

この命令はプログラムの実行中に，誤ってストップキーを押してしまい，プログラムの実行が中断されてしまうのを防止するためのものです．したがって不用意にこの命令を実行してしまうと本来の CTRL + STOP キーの機能が失われ，システムリセットを行わない限りプログラムを中断することができなくなってしまいます．ON STOP GOSUB 文は動作ミスのないプログラムにしてから使用してください．

● 参　照　ON KEY GOSUB, STOP ON/OFF/STOP, 第1章「割込み」

● サンプルプログラム

```
10 ON STOP GOSUB 70
20 PRINT "CTRL+STOPｷｰ ｦ ｵｼﾃｸﾀﾞｻｲ。"
30 STOP ON
40 INPUT A$:PRINT A$
50 IF A$="END" THEN STOP OFF:END
60 GOTO 40
70 PRINT "ﾄﾏﾗﾅｲ!! END ﾄ ﾆｭｳﾘｮｸｽﾙﾄ ﾄﾏﾘﾏｽ。"
80 RETURN
```

ON STRIG GOSUB

オン エストリガ ゴーサブ(ステートメント)

● 機　能　ジョイスティックのトリガボタンが押されたときにジャンプする割込み処理ルーチンの開始行を定義する.

● 書　式　**ON STRIG GOSUB**〔〈行番号〉〕〔,〈行番号〉…〕

● 文　例　**ON STRIG GOSUB 100, 200**

● 解　説　この命令を実行すると,以後,STRIG ON 状態にしておけばスペースキーあるいはジョイスティックのトリガボタンが押されたときに割込みがかかり,プログラムの実行を中断して〈行番号〉から始まる割込み処理ルーチンが実行されます.

〈行番号〉は最大5個まで指定することができ,左側から順番に次のトリガボタンと対応しています.

1番目：キーボードのスペースキー
2番目：ポート1につながっているジョイスティックのトリーグーボタン1
3番目：ポート2につながっているジョイスティックのトリーグーボタン1
4番目：ポート1につながっているジョイスティックのトリーグーボタン2
5番目：ポート2につながっているジョイスティックのトリーグーボタン2

割込み処理からの復帰は,GOSUB 文で呼び出すサブルーチンと同じようにRETURN 文を使います.
割込み処理ルーチンを実行している間,自動的にSTRIG STOP 状態になり,RETURN 文実行とともに,STRIG ON 状態になります.

この命令はプログラムモードでのみ有効です.また,ON ERROR GOTO 文, ON KEY GOSUB 文, ON STOP GOSUB 文, ON SPRITE GOSUB 文で定義する割込み処理を実行中はこれらの処理ルーチンが終了するまで割込みは保留されます.

● 参　照　STRIG ON/OFF/STOP,第1章「割込み」

ON STRIG GOSUB

● サンプルプログラム

```
10 CLS
20 ON STRIG GOSUB 80
30 STRIG(0) ON
40 PRINT "ｽﾍﾟｰｽｷｰ ヲ ｵｼﾃｸﾀﾞｻｲ"
50 PRINT "ｺﾉ ﾌﾟﾛｸﾞﾗﾑ ヲ ﾄﾒﾙﾄｷﾊ"
60 PRINT "CTRL+STOPｷｰ ヲ ｵｼﾏｽ"
70 GOTO 70
80 COLOR 15,1,1
90 PLAY "T255CEG"
100 LOCATE 8,12:PRINT "PUSHED!!!!"
110 FOR I=0 TO 300:NEXT I
120 LOCATE 8,12:PRINT SPC(10)
130 COLOR 15,4,7
140 RETURN
```

OPEN

オープン(ステートメント)

● 機　能　ファイルを開く.

● 書　式　**OPEN　"〈ファイルスペック〉"〔FOR〈モード〉〕AS #〈ファイル番号〉**

● 文　例　**OPEN　"CAS :TEST" FOR INPUT AS #1**

● 解　説　〈ファイルスペック〉で指定するファイルを開き，INPUT #文やPRINT #文でデータを入出力できるようにします．オープンするファイルには〈ファイル番号〉を割り当て，以後のファイルへの入出力では，〈ファイル番号〉を指定します.

〈ファイルスペック〉の形式は"〈デバイス名〉:〈ファイル名〉"です.
〈デバイス名〉は次のような種類があります.

デバイス名	入出力デバイス	使用できるモード	
		INPUT	OUTPUT
CAS:	カセットテープ	○	○
A:～H:＊	フロッピーディスク	○	○
MEM:	RAMディスク	○	○
CRT:	テキスト画面	×	○
GRP:	グラフィック画面	×	○
LPT:	プリンタ	×	○

(＊「DISK BASIC／MSX DOS説明書」のOPEN文参照)

〈ファイル名〉は1～6文字の文字列で表しますが，デバイス名がCRT:，GRP:，LPT:の場合には指定する必要はありません.

〈モード〉は，ファイルへのアクセス方法を指定します．デバイス名がCRT:，GRP:，LPT:の場合はモードの指定は不要です．モードは次の2種類があります.

INPUT　既存のファイルから入力を行います.
OUTPUT　新しくファイルを作り，出力を行います.

〈ファイル番号〉は，1から15までの値を用いることができますが，MAXFILES文で指定した数を越えて指定することはできません．また，既にオープンしているファイルが利用しているファイル番号を用いることはできません.

OPEN

OPEN 文は，指定したファイルを入出力する際に使うバッファ領域を確保し，ファイルを CLOSE 文で閉じたときに解放します．〈ファイル番号〉は，このバッファ領域と対応しています．

● 参　照　CLOSE, INPUT #, PRINT #, MAXFILES, INPUT$, EOF

● サンプルプログラム

```
5 '*** グラフィックがめんに もじを かく ***
10 SCREEN 5
20 PRESET(32,32)
30 OPEN "GRP:" FOR OUTPUT AS#1
40 PRINT #1,"なにか キーを おしてください"
50 PRESET(32,48)
60 A$=INPUT$(1)
70 PRINT #1,A$;
80 GOTO 60
```

OUT*

アウトプット(ステートメント)

● 機　能　出力ポートに1バイトのデータを送る．

● 書　式　OUT 〈ポート番号〉,〈式〉

● 文　例　OUT &H22, &H80

● 解　説　〈ポート番号〉は出力ポートの番号，そして〈式〉は出力する1バイトのデータです．どちらのパラメータも 0～255(&H0～&HFF)までの整数で指定します．

OUT文は，入出力ポートの使い方について知識を得てから使用してください．

PAD*

パッド(関数)

● 機　能　タブレット，ライトペン，マウス，トラックボールの状態を調べます。

● 書　式　PAD（<式>）

● 文　例　PAD (1)

● 解　説　(　)内の式の値によって，命令実行後次のような値を返します．下表では，ベース値Bと最上段の数値dを加えたものが(　)内にセットすべき値です．

例）ポート2に接続されたタブレットのペンの状態（B+d=4)———→PAD(4)

装　置	ベース値	d＝0	d＝1	d＝2	d＝3
ポート1の タブレット	B＝0	ペンの状態 押されている：－1 押されていない：0	X座標 (絶対座標)	Y座標 (絶対座標)	スイッチの状態 押されている：－1 押されていない：0
ポート2の タブレット	B＝4				
ライトペン	B＝8	データの有効性 有効：－1 無効：0			
ポート1の マウスまたは トラックボール	B＝12	※1） 入力要求として使用 (－1を返す)	※2） X座標 (相対座標)	※2） Y座標 (相対座標)	※3） ———
ポート2の マウスまたは トラックボール	B＝16				

※1）PAD(12)またはPAD(16)が実施された時座標データを読みますので，この命令を実行した後で座標を読み取ってください．
FS-5500に内蔵されているボールマウスは，マウスに相当します．

※2）前回読取り座標からの相対座標です．内部カウンタの長さは，マウスが8ビット，トラックボールは4ビットです．読取りの時間々隔が長すぎると，内部カウンタがあふれて無意味な座標値となることがあります．

※3）トリガーボタンの状態は，ジョイスティックと同様，STRIG関数を使用します．

● サンプルプログラム

```
10 '*** おえかき ﾌﾟﾛｸﾞﾗﾑ ***
20 SCREEN 5
30 IF PAD(0)=0 THEN SW=0:GOTO 30
40 X=PAD(1):Y=PAD(2)
50 IF SW=0 THEN PSET(X,Y) ELSE LINE-(X,Y
)
60 SW=1
70 GOTO 30
```

PAINT

ペイント(ステートメント)

● 機　能　指定された境界色で囲まれた領域を，指定された色で塗る.(グラフィックモード専用)

● 書　式　PAINT | (X, Y) / STEP (X, Y) | 〔, 〈領域色〉〕〔, 〈境界色〉〕

● 文　例　PAINT (50, 120), 3, 4

● 解　説　(X, Y) で指定する座標を含む〈境界色〉で囲まれた領域を，指定された〈領域色〉で塗りつぶします．STEP を付けると LP (最終参照点) からの相対座標となります．

〈領域色〉，〈境界色〉はともにカラーコードで指定します．〈領域色〉を省略した場合には，COLOR 文で指定した前景色が用いられます．〈境界色〉を省略した場合には〈領域色〉の指定と同じカラーコードが〈境界色〉として用いられます．

〈境界色〉はSCREEN 3 および 5 ～ 8 のみ使用することが可能で，SCREEN 2，4 では使うことはできません．この場合，〈領域色〉と同じ色で囲まれた領域が塗りつぶされます．

(X，Y) は塗り始める座標を指定します．もしこの点が，既に指定された境界色と同じ色であった場合には，PAINTは塗りつぶしを行いません．

● 参　照　SCREEN, COLOR

● サンプルプログラム

```
10 SCREEN 2
20 FOR I=0 TO 19
30 X1=INT(RND(1)*255):X2=INT(RND(1)*255)
40 LINE(X1.0)-(X2.199),10
50 Y1=INT(RND(1)*190):Y2=INT(RND(1)*190)
60 LINE(0.Y1)-(255.Y2),10
70 NEXT
80 '
90 FOR I=0 TO 200
100 X=INT(RND(1)*255):Y=INT(RND(1)*190)
110 PAINT(X.Y),10
120 NEXT
```

PDL*

パドル(関数)

● 機　能　パドルの状態を得る.

● 書　式　PDL (〈パドルの番号〉)

● 文　例　PDL (2)

● 解　説　〈パドルの番号〉はパドルがポート1につながっていれば1，3，5，7，9または11を指定し，パドルがポート2につながっていれば2，4，6，8，10または12を指定します.パドルの状態は回転の度合によっては0～255の範囲の数値で得られます.

PEEK

ピーク(関数)

● 機　能　メモリ上の指定された番地の内容を読み出す.

● 書　式　PEEK (〈番地〉)

● 文　例　A=PEEK (&HA000)

● 解　説　〈番地〉で指定されたメモリの内容を読み出します．読み出すデータは1バイト分（0～255の値）です.

〈番地〉は，&H0～&HFFFFの値を持つ数式で指定します.

PEEK関数はPOKE文と対応している命令です.

● 参　照　POKE

● サンプルプログラム

```
10 '*** メモリ ﾀﾞﾝﾌﾟ ***
20 INPUT "ADDRESS =";A
30 IF A<0 THEN A=A+65536!
40 FOR I=A TO A+64 STEP 8
50 PRINT "  ";RIGHT$("000"+HEX$(I),4);
60 PRINT USING "(#####)";I;
70 FOR J=I TO I+7
80 PRINT RIGHT$("0"+HEX$(PEEK(J)),2);
90 NEXT J:PRINT
100 NEXT I:END
```

PLAY

プレイ(ステートメント)

● 機　能　音楽を演奏する.

● 書　式　PLAY　〈文字式 1〉〔,〈文字式 2〉〕〔,〈文字式 3〉〕

● 文　例　PLAY "CDEFGABO5C"

PLAY A$+B$, C$

● 解　説　〈文字式〉で定義された音楽を演奏します.

〈文字式1〉だけの記述の場合は，単音となり，〈文字式２〉,〈文字式３〉も合せて記述することにより，２重和音，３重和音も出すことができます.

〈文字式〉には，以下で述べる命令を１つ以上並べて置きます．また文字式は文字定数，文字変数，あるいはそれらを組合せた〈式〉でもかまいません.

命令は，大きく分けて音の高さ，音の長さ，音の速さ(テンポ)，音の大きさ，音色の５種類があります.

注) PLAY命令の場合，特にO(オー)と０(ゼロ)の区別にご注意ください.

●音の高さ

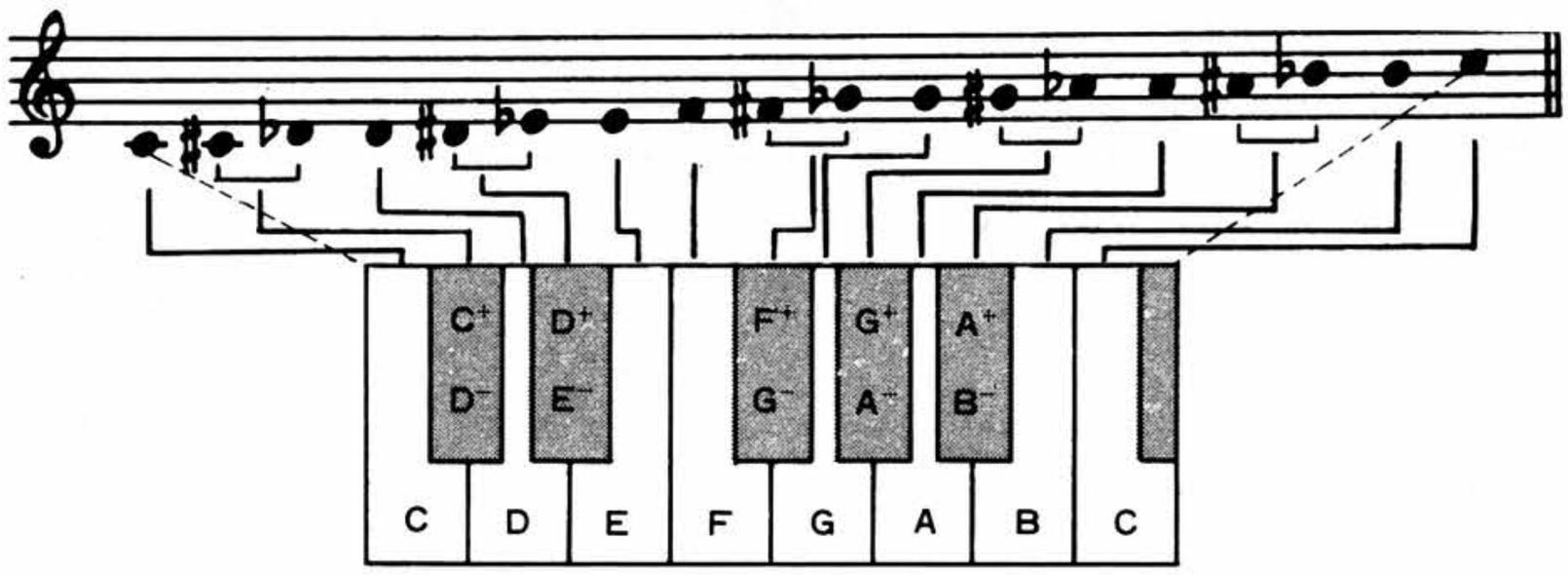
A～G　CDEFGがハ長調のド～ソ，AとBがラとシにそれぞれ対応しています．またA～Gの後に＃や＋記号を付けると半音上った音になり，逆に－記号を付けると半音下った音になります.

例)　ドレミファソラシの音を出す.

PLAY "CDEFGAB"

ド，ミ，ソ　３重和音を出す.

PLAY "C", "E", "G"



PLAY

O〈数値〉　前頁で説明した音階の音域を1～8の数値で指定します．初期値はO4で，1点ハ音から始まる1オクターブを指します．このようにして，O5はO4の1オクターブ上の音階を，また，O3はO4の1オクターブ下の音階を指します．なお，1度オクターブの指定をすると，そのチャンネルで次のオクターブ指定をするまで有効です．

例)

```
PLAY "O3CEGO4CEG O5CC"
```

N〈数値〉　音の高さを0～96の数値で表現します．N1はO1のC#と同じ高さで，N2，N3……と半音ずつ上がって行き，N95はO8のBと同じになります．また N0はR（休符）と同じ働きをします．

例)

```
PLAY "N36N37N38N39N40N41N42N43"
```

N〈数値〉と音階

高さ / オクターブ	C	C＋ D－	D	D＋ E－	E	F	F＋ G－	G	G＋ A－	A	A＋ B－	B
O 1	0	1	2	3	4	5	6	7	8	9	10	11
O 2	12	13	14	15	16	17	18	19	20	21	22	23
O 3	24	25	26	27	28	29	30	31	32	33	34	35
O 4	36	37	38	39	40	41	42	43	44	45	46	47
O 5	48	49	50	51	52	53	54	55	56	57	58	59
O 6	60	61	62	63	64	65	66	67	68	69	70	71
O 7	72	73	74	75	76	77	78	79	80	81	82	83
O 8	84	85	86	87	88	89	90	91	92	93	94	95

PLAY

●音の長さ

L〈数値〉

音の長さを$\frac{1}{\langle数値\rangle}$音にします．〈数値〉の範囲は１～64までです．Ｌ１を全音としますと，Ｌ２以降の長さは次のようになります．

Ｌ１　全音符　　（４拍）
Ｌ２　２分音符　　（２拍）
Ｌ３　１/３音符　　（４/３拍）
Ｌ４　４分音符　　（１拍）
：
Ｌ８　８分音符　　（１/２拍）
：
Ｌ64　64分音符　　（１/16拍）

初期値はＬ４です．１度音の長さを指定しますと，そのチャンネルで次に音の長さを指定するまで有効です．また，１音についてだけ長さを変えたいときは，音階名の後ろに音の長さを表す〈数値〉を付け加えます．たとえばL16CはC16と同じです．

例）

```
                                    ┌──┬──┬── 8分音符
PLAY "L4DCFG8O5C8O4AG8O5CO4AFGDC"
```

・（ピリオド）

ちょうど付点音符のように音階名や休符の後ろに付け加えて，本来の音の長さの$\frac{3}{2}$倍にします．また，２個並べて書きますと$\frac{9}{4}$倍，また，３個並べて書くと$\frac{27}{8}$倍となります．

例）

```
PLAY " T200  L4F. FF. GA. AA. GF. FF.
DC2"
```

R〔〈数値〉〕

休符を表します．〈数値〉はＬ命令での指定方法と同じです．〈数値〉を省略するとＲ４とみなされます．この命令はＬ命令で指定した長さに影響されません．

PLAY

●速さ（テンポ）

T〈数値〉　テンポを指定します．〈数値〉は1分間に演奏する4分音符の数を指定します．〈数値〉の範囲は32～255です．初期値はT120です．なお，1度速さの指定をすると，そのチャンネルで次に速さの指定をするまで有効です．

●音の大きさ

V〈数値〉　音量を指定します．〈数値〉の範囲は0～15で，数が大きいほど音も大きくなります．
なお，1度音量の指定をすると，次に音量の指定をするまで有効です．

例)

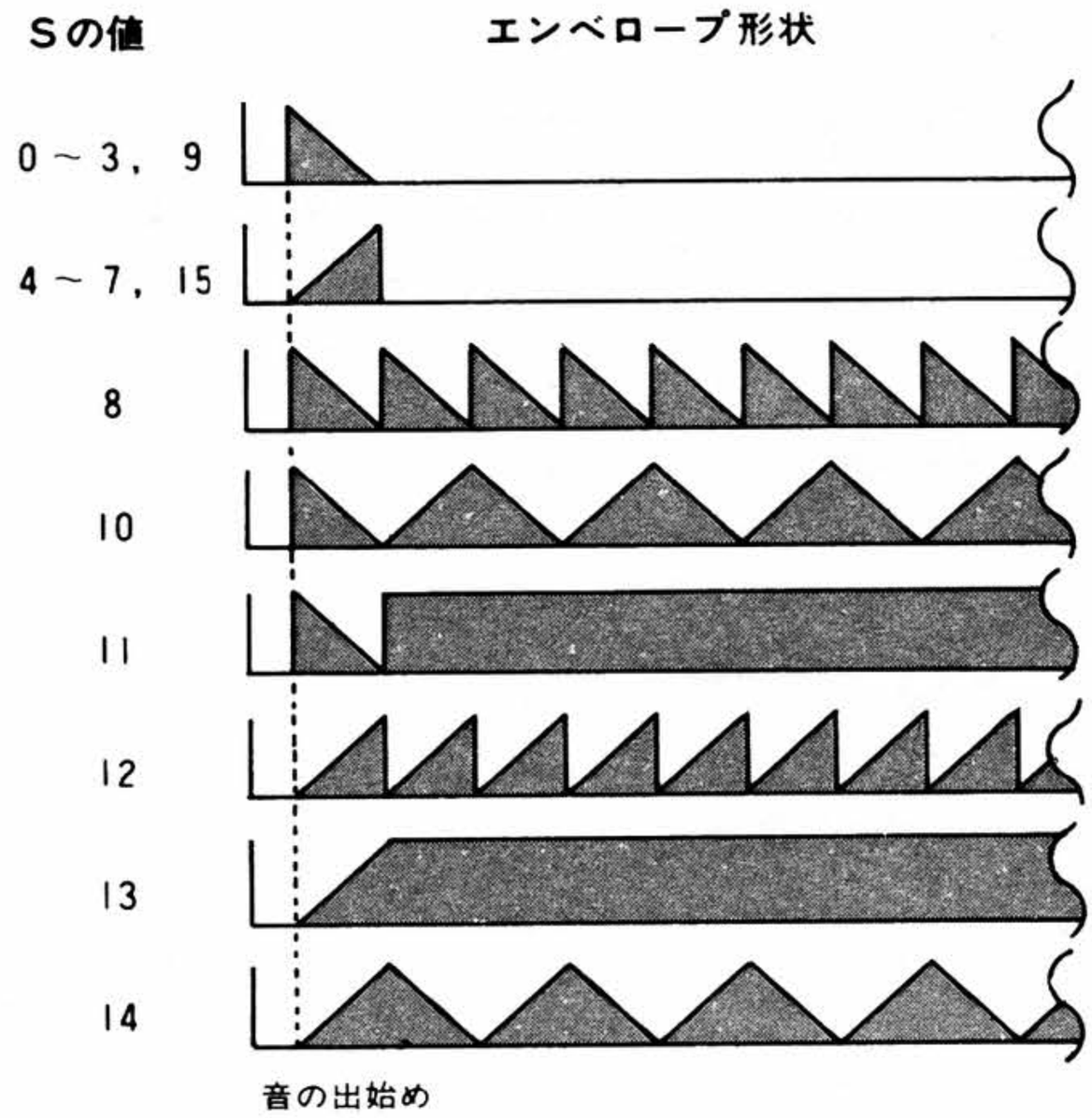
```
PLAY "V15CGV8CG"
```

●音色

音色はエンベロープ周期（M命令）とエンベロープ形状（S命令）とで決まります．

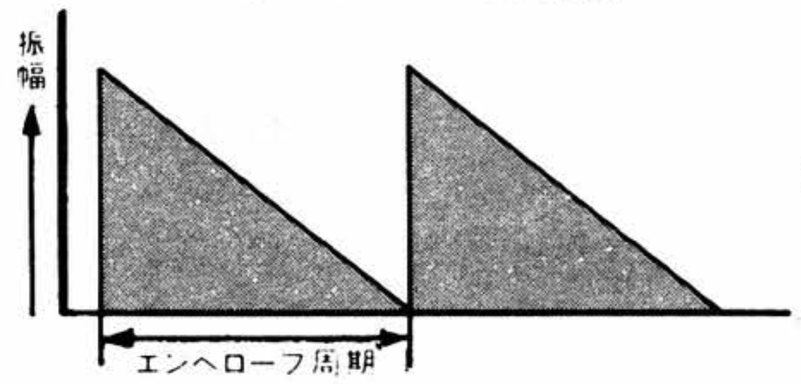
S〈数値〉　エンベロープ形状(音量変化の波形)を指定します．一音ごとに，〈数値〉で指定したエンベロープ形状で音の振幅が変化します．
〈数値〉と波形は次頁の図のように対応しています．
S命令使用時は，次頁のM命令の指定も必要です．
なお1度エンベロープ形状を指定すると，別のエンベロープ形状を指定するか，音量(V命令)指定するまで有効です．

PLAY



M〈数値〉

エンベロープ周期を指定します．エンベロープ周期は1～65535までの〈数値〉で指定しますが，あまり大きすぎるとエンベロープ形状の特徴が出ません．(一般的には100～10000)



エンベロープ周期

T＝1.43×10⁻⁴×(数値)
(単位：秒)

エンベロープ形状とエンベロープ周期の組合わせでいろいろな音を出すことができますので試してみるとよいでしょう．

例)

```
PLAY "V9CDE"
PLAY "S9M3000CDE"
PLAY "S4M2000CDE"
PLAY "S8M700CDE"
PLAY "S10M500CDE"
```

```
PLAY "S11M3000CDE"
PLAY "S12M1200CDE"
PLAY "S13M300CDE"
PLAY "S14M800CDE"
```

PLAY "S9M3000 T200 L4F. FF. GA. AA. GF. FF. DC2"（音の長さ(ヒリオド)の所の例と同じメロディーです．比較してみてください．）

●変数による指定

X〈文字変数〉； 〈文字変数〉中の文字列に含まれているミュージック命令を実行します．〈文字変数〉には上記のミュージック命令をあらかじめ入れておきます．〈文字変数〉の後にセミコロン(；)は必ず入れてください．文字変数による指定は，ミュージック命令の文字列が255を越える場合や音を少しずつ変化させながら繰り返し演奏するときなどに便利です．

例)

```
A$="CCGGAAG" : B$="FFEEDDC"
PLAY "XA$;XB$;"
```

〈数値変数〉 ミュージック命令では〈数値〉などのハラメータを定数で指定しますが，"=〈変数名〉;" の形式にすれば数値変数により指定することができます．この場合〈変数名〉の後ろにはセミコロンを必ず入れます．ただし，数値変数の値は32767以下です．

例)

```
10 FOR K=1 TO 15
20 PLAY "V=K;CEG"
30 NEXT K
```

〔ドミソ(CEG)を順次音量を変えて出力します.〕

PLAY

● サンプルプログラム

```
10 '*** ﾊﾞｯﾊ ***
20 PLAY "T240L6V12","T240L2V9"
30 PLAY "R8O6GABO7DCCED","O4GO5GE"
40 PLAY "DGF#GDO6BGAB","O4BO5EO4E"
50 PLAY "O7CDEDCO6BABG","O4ABO5C"
60 PLAY "F#GADF#AO7CO6BA","O5DF#D"
70 PLAY "BGABO7DCCED","GGC"
80 PLAY "DGF#GDO6BGAB","O4BO5ED"
90 PLAY "EO7DCO6BAGDGF#G2","CC#DG"
```

PLAY＊

プレイ(関数)

● 機　能　音楽を演奏中かどうかを調べる.

● 書　式　PLAY (〈ボイスチャンネル〉)

● 文　例　PLAY (0)

● 解　説　ミュージックキューの状態を調べ,各ボイスチャンネルがPLAY文で命じた音楽を演奏中かどうかを判断し，演奏中であれば－1，そうでなければ0の値を得られます.〈ボイスチャンネル〉は次のように指定します.

0：ボイスチャンネル1，2，3（いずれかが演奏中であれば－1の値を得られます)
1：ボイスチャンネル1
2：ボイスチャンネル2
3：ボイスチャンネル3

● サンプルプログラム

```
10 '*** キラキラホ゛シ ***
20 COLOR 15, 0, 0
30 CLS:KEY OFF
40 READ A$:IF A$=":¥" THEN 100
50 PLAY A$
60 LOCATE INT(RND(1)*29), INT(RND(1)*23)
70 PRINT "*";
80 IF PLAY(0) THEN 80
90 GOTO 40
100 COLOR 15, 4, 7:CLS
110 END
120 DATA C,C,G,G,A,A,G,R
130 DATA F,F,E,E,D,D,C,R
140 DATA G,G,F,F,E.E.D,R
150 DATA G,G,F,F.E,E,D,R
160 DATA C.C,G,G,A,A.G.R
170 DATA F.F,E.E.D,D.C,R
180 DATA ¥
```

POINT

ポイント(関数)

● 機　能　指定された座標の画素の色を調べる.(グラフィックモード専用)

● 書　式　**POINT (X, Y)**

● 文　例　**POINT (50, 50)**

● 解　説　(X, Y)で指定した座標上に表示されている画素の色を調べ，結果をカラーコードで得ます．指定した座標が表示領域外であった場合には，－1を返します．

指定された座標上にスプライトパターンが表示されていても，スプライトパターンの色を調べることはできません．この場合は背景の画面上の色が得られます．

● サンプルプログラム

```
10 SCREEN 2
20 OPEN "GRP:" FOR OUTPUT AS #1
30 FOR I=0 TO 20
40 FOR J=0 TO 8
50 PRESET(J*24,I*8)
60 C=INT(RND(1)*13)+2
70 COLOR C:PRINT #1,"♥"
80 NEXT J,I
90 COLOR 15
100 PRESET(100,170)
110 PRINT #1,"Color No."
120 FOR I=0 TO 20
130 FOR J=0 TO 8
140 K=POINT(J*24+4,I*8+4)
150 PRESET(J*24+8,I*8)
160 PRINT #1,USING"##";K
170 NEXT J,I
180 GOTO 180
```

POKE

ポーク(ステートメント)

● 機　能　メモリ上の指定した番地へデータを書き込む.

● 書　式　POKE 〈番地〉,〈データ〉

● 文　例　POKE &HF000 , &HFF

● 解　説　指定されたメモリ上の番地に1バイト（8ビット）のデータを書き込みます.

〈番地〉は&H0～&HFFFFの値を持つ式で指定します.

〈データ〉はメモリに書き込まれるデータで，0～255 (&H0～&HFF) の値を持つ数式です．もし小数点以下があれば整数化してから実行されます.

この命令は，現在のメモリの内容を書き換えてしまうため，不用意に使うとBASICが使っている作業領域を壊してしまい，誤動作の原因となることがあります．使う時にはメモリマップなどで使用可能な領域かどうかを確認してください。

POKE文の逆の働きをする関数としてPEEK関数が用意されています．これらPOKEとPEEKは，データの格納や機械語サブルーチンとの引数の受け渡しなどに役立ちます.

● 参　照　PEEK

● サンプルプログラム

```
10 CLEAR 256, &HDFFF
20 PRINT "    ＊POKE マエ"
30 GOSUB 100
40 FOR I=&HE000 TO &HE020
50 POKE I, I−&HE000
60 NEXT
70 PRINT "    ＊POKE アト"
80 GOSUB 100
90 END
100 ' PEEK スル
110 FOR I=&HE000 TO &HE020 STEP 8
120 FOR J=I TO I+7
130 PRINT USING "&&  ";HEX$(PEEK(J));
140 NEXT J:PRINT
150 NEXT I
160 RETURN
```

POS

ポジション(システム変数)

● **機　能**　テキスト画面上のカーソルの水平位置を調べる.(テキストモード専用)

● **書　式**　**POS (〈式〉)**

● **文　例**　**P=POS (0)**

● **解　説**　〈式〉はダミーの引数で，何を指定しても同じですが，通常0を使います．
得られる値はカーソルのキャラクタ座標の左からの水平位置です．

PRESET

ポイント リセット(ステートメント)

● **機　能**　**指定された座標上に描かれた点を消去する.(グラフィックモード専用)**

● **書　式**　**PRESET** | **(X, Y)** / **STEP (X, Y)** | **〔, 〈カラーコード〉〕〔, 〈論理演算子〉〕**

● **文　例**　**PRESET (100, 100)**

● **解　説**　(X, Y) で表される位置に描かれた点を消去します．STEP を付けた場合は LP（最終参照点）からの相対座標による指定となります．

〈カラーコード〉は点を消すときに用いる色を指定します．省略した場合には現在の背景色(COLOR文参照)が採用されます．

PRESET 文を実行した結果，LP は指定した座標へ移動します．

〈論理演算子〉については，LINE文を参照してください．

● **参　照**　PSET, COLOR

● **サンプルプログラム**

```
10 SCREEN 2
20 LINE(10,10)-(245,180),15,BF
30 FOR I=0 TO 700
40 X=INT(RND(1)*233)+11
50 Y=INT(RND(1)*168)+11
60 PRESET (X,Y)
70 NEXT
80 GOTO 80
```

PRINT

プリント(ステートメント)

● 機　能　テキスト画面に出力する.(テキストモード専用)

● 書　式　PRINT　〔〈式〉…〕

● 文　例　PRINT "ABC"
　　? 　"CDE"

● 解　説　〈式〉で指定する数値や文字列を画面に出力します.〈式〉が省略されている場合は改行のみを行います.

複数個の〈式〉を指定する場合は式の間をカンマ，またはセミコロンで区切ります．式の値や文字列を表示する領域は，あらかじめ各行が14文字毎に分割して定められており，**区切り記号にカンマを使うと次の領域の始めから表示され，頭ぞろえを行うことができます.**

区切り記号にセミコロンを使うと直前にプリントしたもののすぐ後ろに続いて次の値や文字列が表示されます．一番最後の〈式〉の後ろにセミコロンがあれば，次に実行されるPRINT文で出力する〈式〉の内容がすぐ後に表示されます.

数値を表示した場合，その後ろに必ず空白が1つ挿入されます．また，数値の前には符号を表示するための桁が確保され，数値が正の値ならば空白，数値が負の値ならばマイナス符号が表示されます.

単精度の数値は指数形式でなくても6桁以下の桁数で精度に影響をおよぼさず表示できるものは，実数形式で表示されます．同様に倍精度の数値は14桁以下の桁数で精度に影響を及ぼさず表示できるものは，実数形式で表示されます.

この命令のキーワード"PRINT"の簡略形として疑問符（?）を使用することができます..

表示しようとする数値が現在のカーソルの位置より後方に表示しきれない場合（それを表示すると次の行に渡ってしまう）改行してそれを表示します.

PRINT

● サンプル プログラム

```
10  '*** PRINT ﾌﾞﾝ ｻﾝﾌﾟﾙ ***
20  A=10:B=33.3:A$="TOUKYOU":B$="OOSAKA"
30  PRINT "A+B"
40  PRINT  A+B
50  PRINT  A;B
60  PRINT  A,B
70  PRINT"A$+B$"
80  PRINT A$+B$
90  PRINT A$;B$
100 PRINT A$,B$
110 PRINT A;
120 PRINT A
130 PRINT
140 PRINT B;
150 PRINT B
160 PRINT CHR$(65)+CHR$(66);CHR$(67)
```

```
 ｷﾞょう        ｼﾞっこう ｹっか
(30)        A+B
(40)         43.3
(50)         10   33.3
(60)         10                  33.3
(70)        A$+B$
(80)        TOUKYOUOOSAKA
(90)        TOUKYOUOOSAKA
(100)       TOUKYOU          OOSAKA
(110,120)   10   10
(130)
(140,150)   33.3   33.3
(160)       ABC
```

P

PRINT USING*

プリント ユージング(ステートメント)

● 機　能　文字列や数値を指定した書式でテキスト画面に表示する.(テキストモード専用)

● 書　式　PRINT USING 〈書式〉; 〈式〉…

● 文　例　PRINT USING "＃＃＃＃,." ;A,B

● 解　説　〈式〉で表す文字列や数値を〈書式〉に従って編集し，テキスト画面に表示します.

〈式〉はカンマ（,）やセミコロン（;）で区切って複数個指定することができます．この場合，左側の〈式〉から順番に画面へ表示されます.

〈書式〉は次に説明する書式制御文字を連ねた文字列で，文字列を編集するものと数値を編集するものとがあります.

●文字列を編集する書式制御文字〔グラフィックキャラクタを使用する場合は,1字あたり2文字分の書式制御文字が必要です.〕

!……文字列の左側1字を表示します.

例)

```
A$= "Japan"
Ok
PRINT USING "!" ; A$
J
Ok
```

& 〈n個の空白〉&

文字列の左側から（n＋2）文字を表示します．文字列が（n＋2）文字より長ければ余分な文字は表示されず，短ければ足りない文字数だけの空白を右側に補って表示されます.

例)

```
A$= "Japan"
Ok
PRINT USING "&    &" ; A$
Japa
Ok
```

PRINT USING*

@……〈書式〉の文字列中に含まれる"@"を〈式〉の文字列で置き換え，〈書式〉を画面に表示します．1つの"@"は左側の〈式〉から順番に対応しており，〈書式〉中の"@"の数が〈式〉の数より多い場合，残りの"@"は無視されます．

例）

```
A$= "I" : B$= "Japan"
Ok
PRINT USING "@  Love  @." ; A$, B$
I Love Japan.
Ok
```

●数値を編集する書式制御文字

#……数値を"#"で指定した桁数だけ表示します．数値が指定した桁数より小さければ右づめで表示され，左側に空白が補われます．

.……小数点の位置を指定します．小数部の桁数が書式で指定した桁数より小さければ，右側に0が補われます．

例）

```
PRINT USING "###.##" ; 10.2,  2 , 3.456,. 24
  10.20    2.00    3.46    0.24
Ok
```

+……〈書式〉の左端に"+"を付けると，数値の前に正負符号が表示され，〈書式〉の右端に"+"を付けると数値の後ろに正負符号が表示されます．また，"+"を2個以上並べた場合，数値の外側の"+"は制御文字とみなされません．

例）

```
PRINT USING "+####.##" ; 1.25, -1.25
  +1.25    -1.25
Ok
PRINT USING "####.##+" ; 1.25, -1.25
  1.25+    1.25-
Ok
```

PRINT USING*

－……〈書式〉の右端に"－"を付けると，数値が負の場合に数値の後ろに"－"が表示されます．〈書式〉の左端に付けたり，2個以上並べたときには制御文字とみなされません．

例）

```
PRINT USING "###.##－"; 1.25, －1.25
  1.25     1.25－
Ok
```

＊＊……〈書式〉の左端を"##"の代わりに"＊＊"にしておくと，数値の整数部の桁数が書式で指定した桁数より小さければ，左側に"＊"が補われます．

例）

```
PRINT USING "＊＊###.##"; 1.25, －1.25
＊＊＊1.25＊＊－1.25
Ok
```

￥￥……〈書式〉の左端に"￥￥"を付けておくと，数値の直前に"￥"が表示されます．"￥￥"は"##"を指定したのと同じように2桁分の表示桁を確保しますが，"￥"を表示するために，このうちの1桁が使われます．"￥￥"は指数形式の書式指定をしているときには使用できません．

例）

```
PRINT USING "￥￥###.##"; 12.35, －12.35
  ￥12.35   －￥12.35
Ok
PRINT USING "￥￥###.##－" ; 12.35, －12.35
  ￥12.35     ￥12.35－
Ok
```

PRINT USING*

＊＊¥……〈書式〉の左端に"＊＊¥"を付けておくと，数値の直前に"¥"が表示され，また，数値の整数部の桁数が書式で指定した桁数より小さければ，左側に"＊"が補われます．"＊＊¥"は"＊＊＊"を指定したのと同じように3桁分の表示桁を確保しますが，"¥"を表示するために，このうちの1桁が使われます．"＊＊¥"は指数形式の書式指定をしているときには使用できません．

例）

```
PRINT USING "＊＊¥#.##"; 12.35
＊¥12.35
Ok
```

,……","を〈書式〉の整数部の"#"の並びにおいた場合，数値の整数部が3桁ごとに","で区切られて表示されます．また，","を小数部（小数点より右側）に置いた場合は数値の右側に","が表示され，3桁ごとの区切りは行われません．","は指数形式の書式指定をしているときには使用できません．

例）

```
PRINT USING "####,.##"; 1234.5
1,234.50
O
PRINT USING "#####.##,"; 1234.5
1234.50,
Ok
```

^ ^ ^ ^…"^ ^ ^ ^"を桁数指定の"#"の並びの後に付けると，数値が指数形式で表示されます．また，〈書式〉の先頭に"＋"，または〈書式〉の後に"＋"か"－"を指定していない場合，数値が正ならば数値の直前に1桁の空白が表示され，数値が負ならば数値の直前に"－"が表示されます．

PRINT USING*

例)

```
PRINT USING "##.## ^^^^";234.56
 2.35E+02
Ok
PRINT USING "#.##^^^^-";-12.34
 1.23E+01-
Ok
PRINT USING "+#.##^^^^";12.34, -12.34
+1.23E+01-1.23E+01
Ok
```

%……表示しようとする編集済の数値が書式で指定した桁数より大きい場合，数値の直前に "%" が表示されます．また，表示しようとする数値を丸めたことによって書式で指定した桁数より大きくなった場合も，数値の直前に "%" が表示されます．

例)

```
PRINT USING ".##"; .999
%1.00
Ok
PRINT USING "##.##"; 123.45
%123.45
Ok
```

〈書式〉中に上記の書式制御文字以外の文字を置くと，文字の位置に応じて数値の前や後ろにそのキャラクタが表示されます．

例)

```
PRINT USING "TAX **¥###"; 123
TAX **¥123
Ok
```

PRINT USING*

● サンプルプログラム

```
10 PRINT USING"#######";123456!
20 PRINT USING"#.####";12.3456
30 PRINT USING"####.#";1234.56
40 PRINT USING"¥¥#####";12345!
50 PRINT USING"####,#";12345!
60 PRINT USING"**¥###";12!
70 PRINT USING"######+";12345
80 PRINT USING"######^^^^";1234567890#
```

PRINT

プリント シャープ(ステートメント)

● 機　能　ファイルにデータを書き出す.

● 書　式　PRINT　#〈ファイル番号〉,〔〈式〉・・・〕

● 文　例　PRINT　#2, A, B, C

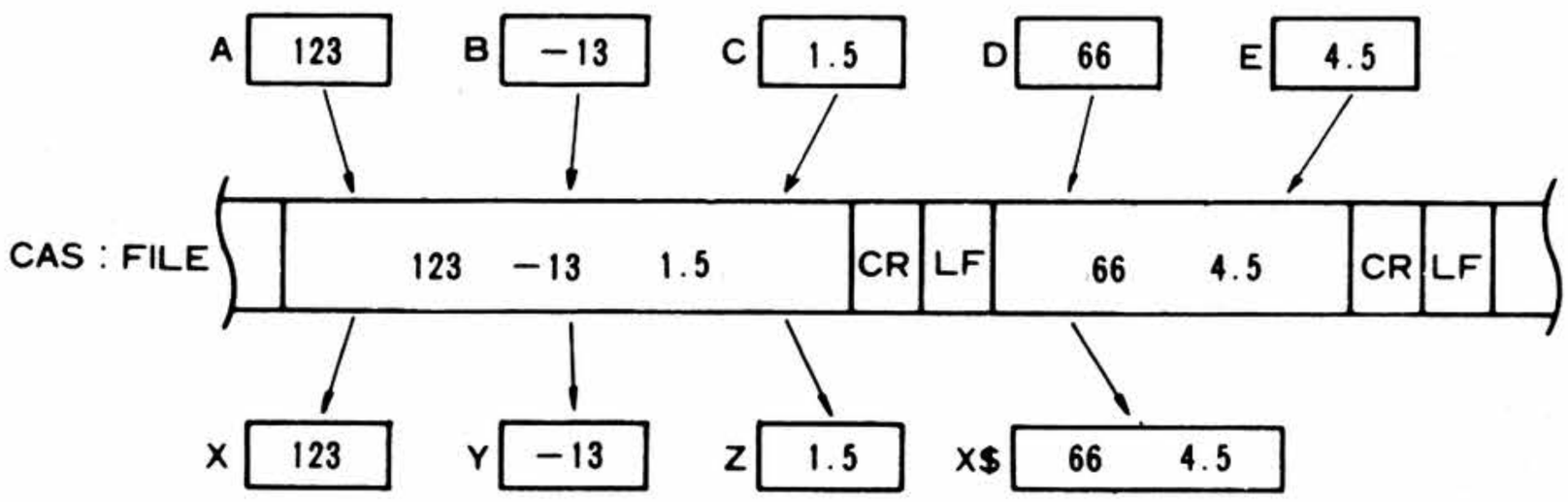
● 解　説　あらかじめOPEN 文でOUTPUT モードを指定して開いておいたファイルへ〈式〉で表す数値や文字列を書き出します. PRINT #文は書き出すファイルが"CRT:"（テキスト画面）や"GRP:"（グラフィック画面）であればPRINT 文と同じ働きをし,"LPT:"（ラインプリンタ)であればLPRINT 文と同じ働きをします. また,"CAS:"（カセットテープ）へ書き出したデータは, INPUT #文, LINE INPUT #文, INPUT$関数を使って読み込むことができます.

〈ファイル番号〉はOPEN 文で指定した番号です.

〈式〉はファイルに書き出す数値, または文字列です. 複数の〈式〉を書き出す場合は〈式〉と〈式〉の間をセミコロン（;）かカンマ（,）で区切ります.

〈式〉が数値であれば, PRINT #文を実行すると, 文字列に変換された数値がPRINT 文と同様に空白で区切って書き出され, 最後にキャリッジリターンコード（CHR$ (13)）とラインフィードコード（CHR$ (10)）とが書き出されます.

```
OPEN "CAS:FILE" FOR OUTPUT AS#1
PRINT #1, A;B;C
PRINT #1, D;E
CLOSE #1
```



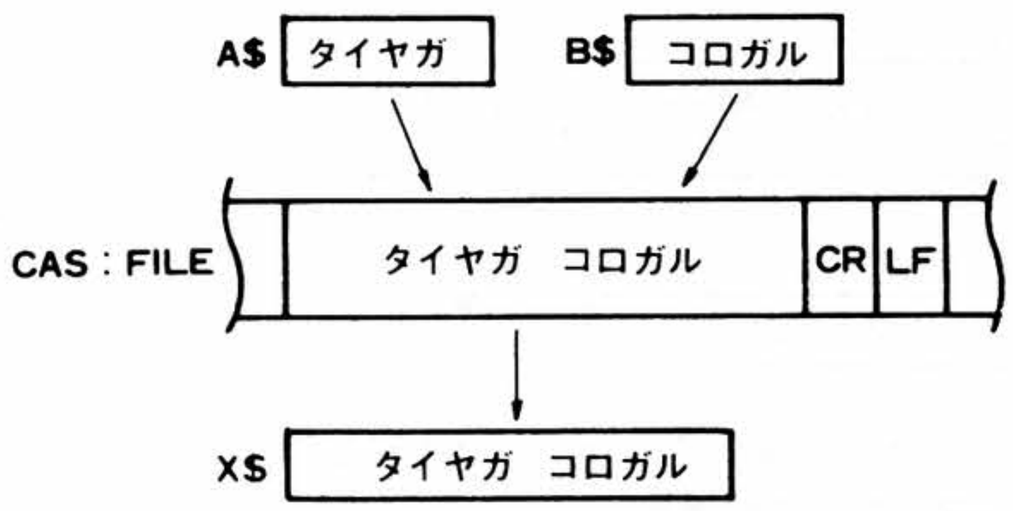

```
OPEN "CAS:FILE" FOR INPUT AS#1
INPUT #1, X, Y, Z
LINE INPUT #1, X$
CLOSE #1
```

PRINT＃*

このようにしてカセットテープ上にいったんファイルを作成しておくと，INPUT #文を使って空白で区切られたデータを読み込んだり，LINE INPUT #文を使ってキャリッジリターンコード／ラインフィードコードごと（1行単位）にデータを読み込んだり，また，INPUT$関数を使って指定した長さだけのデータを読み込んだりすることができます．
PRINT #文で書き出すデータは，PRINT 文で画面へ表示する内容と同じですから，カセットテープファイルへ書き出す場合，区切り記号にカンマを使うと余分な空白がファイルに書き出されてしまいます．

〈式〉の内容が文字列であれば，PRINT #文を実行すると〈式〉で指定した文字列が連続してファイルに書き出され，最後にキャリッジリターンコード（CHR$(13)）とラインフィードコード（CHR$(10)）とが書き出されます．ただし，式の並びのうち，最後の式の後ろにセミコロンをつけたときはこの2つのコードは書き出されません．
〈式〉と〈式〉とをセミコロンで区切ると，数値の場合と異なり，データは空白で区切られずにファイル上に書き出されます．このため INPUT #文ではこれらのデータを別々の文字列として読み込むことはできません．

```
OPEN "CAS:FILE" FOR OUTPUT AS#1
PRINT #1, A$;B$
CLOSE #1
```

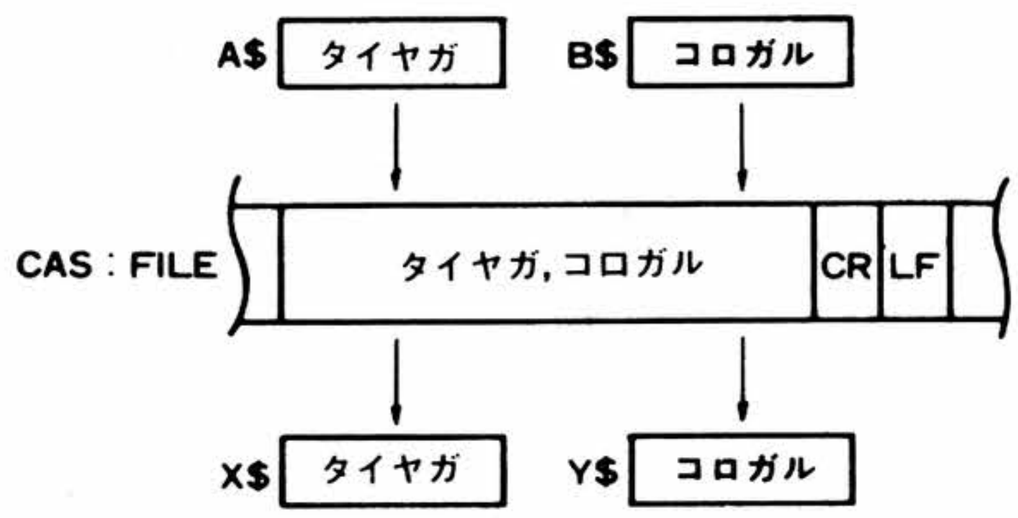



```
OPEN "CAS:FILE" FOR INPUT AS#1
INPUT #1, X$
CLOSE #1
```

INPUT＃文で2つの〈式〉を別々の文字列として読み込みたい場合はPRINT＃文中の〈式〉と〈式〉の間に区切り記号としてカンマを入れてください．

PRINT# *

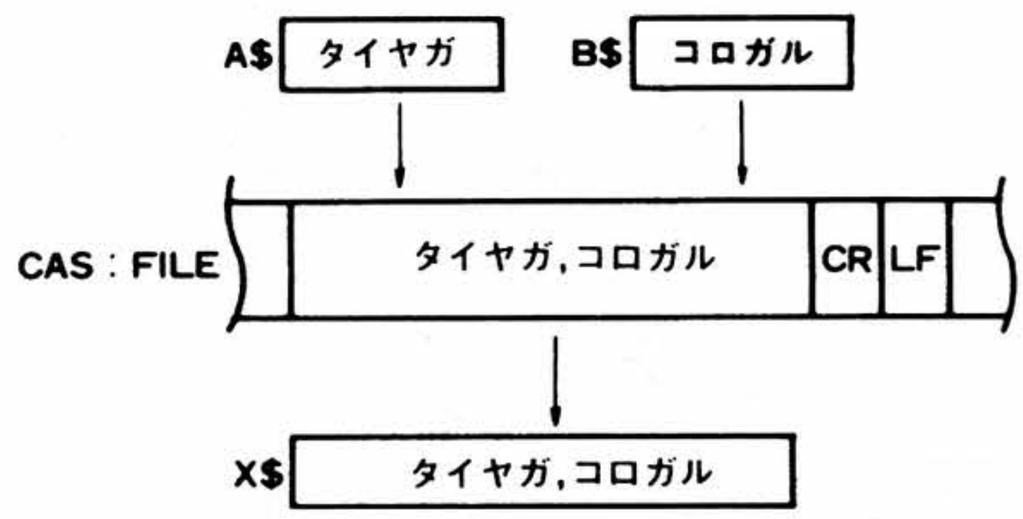
```
OPEN "CAS:FILE" FOR OUTPUT AS#1
PRINT #1, A$;",";B$
CLOSE #1
```




```
OPEN "CAS:FILE" FOR INPUT AS#1
INPUT #1, X$, Y$
CLOSE #1
```

LINE INPUT #文を使うとキャリッジリターンコード＋ラインフィードコードで囲まれたデータが読み込まれます.

```
OPEN "CAS:FILE" FOR OUTPUT AS#1
PRINT #1, A$;",";B$
CLOSE #1
```

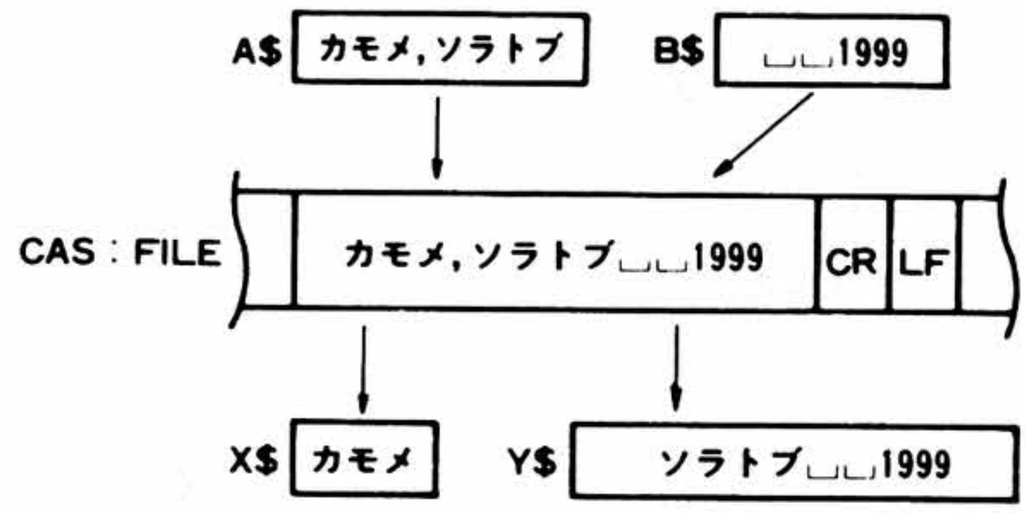



```
OPEN "CAS:FILE" FOR INPUT AS#1
LINE INPUT #1, X$
CLOSE #1
```

文字列中にカンマ，セミコロン，キャリッジリターンコード＋ラインフィードコードを含む場合，または文字列の先頭に意味のある空白がある場合はPRINT #文中の〈式〉の前後を独立した引用符（CHR$ (34)）で囲んで下さい.

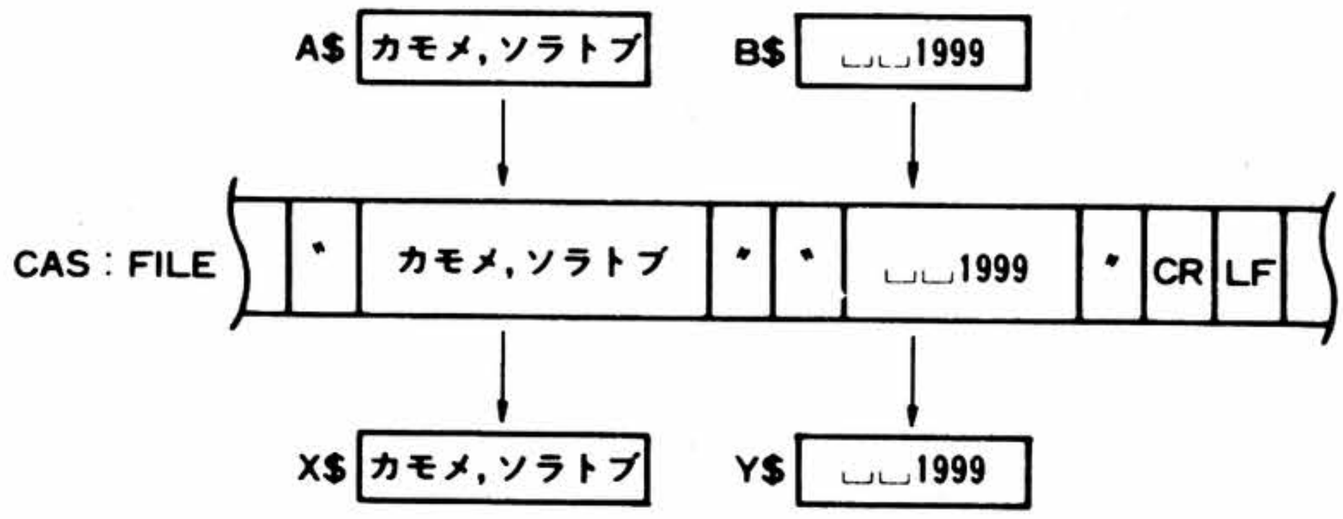
PRINT# *

```
OPEN  "CAS : FILE" FOR OUTPUT AS #1
PRINT   #1, A$ ; B$
CLOSE   #1
```




```
OPEN  "CAS : FILE" FOR INPUT AS #1
INPUT   #1, X$, Y$
CLOSE   #1
```

```
OPEN  "CAS : FILE" FOR OUTPUT AS  #1
PRINT   #1, CHR$(34) ; A$ ; CHR$(34) ; CHR$(34) ; B$ ; CHR$(34)
CLOSE   #1
```




```
OPEN  "CAS : FILE" FOR INPUT AS #1
INPUT   #1, X$, Y$
CLOSE   #1
```

PRINT USING 文と同じように，PRINT # USING を使用してファイルに書き出すデータの書式を指定することもできます．

● 参　照　OPEN，CLOSE，PRINT # USING，INPUT #，LINE INPUT #，第1章「ファイル」

PRINT＃＊

● サンプルプログラム

```
10 '*** ファイル ノ ヨミカキ ********
20 PRINT "コレカラ テﾞータヲ カキﾀﾞシマス。"
30 PRINT "カセットヲ ロクオンノ シﾞョウタイニシテ クﾀﾞサイ。"
40 PRINT "HIT ANY KEY"
50 IF INKEY$="" THEN GOTO 50
60 OPEN "CAS:FILE" FOR OUTPUT AS #1
70 FOR I=1 TO 5
80 READ A$,B$,C
90 PRINT #1,A$;",";B$;",";C
100 NEXT I
110 CLOSE #1
120 DATA エﾍﾞレスト,ネパ°ール,8848
130 DATA アンナプ°ルナ,ネパ°ール,8078
140 DATA アコンカクﾞア,アルセﾞンチン,6960
150 DATA キリマンシﾞャロ,タンサﾞニア,5895
160 DATA マッターホルン,スイス,4427
170 PRINT "カキﾀﾞシタ テﾞータヲ ヨミマス。"
180 PRINT "カセットヲ マキモトﾞシテ サイセイノ シﾞョウタイニ
シテクﾀﾞサイ。"
190 PRINT "HIT ANY KEY"
200 IF INKEY$="" THEN GOTO 200
210 OPEN "CAS:FILE" FOR INPUT AS #1
220 FOR I=1 TO 5
230 INPUT #1.A$.B$,C
240 PRINT A$;TAB(8);"(";B$;")";TAB(17);
C;"M"
250 NEXT I
260 CLOSE #1:END
```

PRINT #USING*

プリント シャープ ユージング(ステートメント)

● 機　能　文字列や数値を指定した書式でファイルに出力する.

● 書　式　PRINT #〈ファイル番号〉,USING〈書式〉;〈式〉〔;|,〕〈式〉・・・〔;|,〕

● 文　例　PRINT #1,USING "@####. ##";A$;B

● 解　説　〈式〉で表す文字列や数値を〈書式〉に従って編集し〈ファイル番号〉で指定したファイルへ出力します. PRINT # USING 文は文字列や数値を〈書式〉に従って出力することを除けば, その機能は PRINT #と同じです. また,〈書式〉を使った編集方法は PRINT USING 文と同じですので, そちらを参照してください.

● 参　照　PRINT #, PRINT USING

PSET

ポイント セット(ステートメント)

● **機　能**　指定した座標上に点を描く．(グラフィックモード専用)

● **書　式**　**PSET** | **(X, Y)** / **STEP (X, Y)** | **〔, 〈カラーコード〉〕〔, 〈論理演算子〉〕**

● **文　例**　**PSET (100, 100), 8**

● **解　説**　(X, Y) の位置に点を描きます．STEP を付けると LP (最終参照点) からの相対座標による指定となります．

〈カラーコード〉は点の色を指定し，もし省略された場合は，現在の前景色 (COLOR 文参照) が採用されます．

PSET 実行の結果，LP は指定された座標 (X, Y) に移動します．

〈論理演算子〉はSCREEN 5 ～ 8 でのみ使用することができます．詳細は LINE命令の説明を参照してください．

● **参　照**　COLOR, PRESET

● **サンプルプログラム**

```
10 '*** ﾎﾟﾁﾎﾟﾁ ***
20 SCREEN 3
30 FOR I=1 TO 200
40 X1=INT(RND(1)*256)
50 Y1=INT(RND(1)*192)
60 CL=INT(RND(1)*14+2)
70 PSET(X1,Y1),CL
80 NEXT I
```

PUT KANJI

プット カンジ(ステートメント)

● **機　能**　グラフィック画面に漢字を表示する．(SCREEN 5 ～ 8 専用)

● **書　式**　**PUT KANJI〔(X,Y)〕,〈漢字コード〉〔,〈カラーコード〉〕〔,〈論理演算子〉〕〔,〈モード〉〕**

● **文　例**　**PUT KANJI (20, 10), &H3441, 15**

● **解　説**　SCREEN5～8のグラフィック画面の指定した所（X，Y）に漢字を表示します．ただし，内蔵漢字 ROM を持っていない機種では，スロットに漢字 ROM カートリッジが必要です．漢字 ROM がない場合には，PUT KANJI を実行しても漢字は表示されません．

表示位置（X，Y）は文字の左上角です．文字は縦16，横16ドットのパターンで表示されます．省略した場合は，最終参照点より表示されます．最終参照点は，PUT KANJI 文の実行により16ドット左に移動します．

〈漢字コード〉は資料編の「漢字コード表」を参照してください．

〈カラーコード〉は，漢字を表示する色を指定します．省略すると前景色で表示されます．(文字の周辺(16×16ドットの領域)の色は背景色になります．)

〈論理演算子〉は漢字パターンと転送先データとの論理演算を指定します．
下記10種類があり，省略値は〝PSET〟です．〝T〟が付いたものは，転送データが透明色の部分の転送先の色は変わりません．

PSET，PRESET，XOR，OR，AND
TPSET，TPRESET，TXOR，TOR，TAND

〝TPSET〟などを指定して，文字部分だけを転送したいような場合は，事前に背景色を透明色にしておいてください．

LINE 文の〈論理演算〉の説明も参照してください．

PUT KANJI*

〈モード〉には，0.1または2のいずれかの数値をセットします．省略した場合は0となります．

0：16×16ドット表示モード

1：偶数行表示モード(文字パターンの偶数行だけを VRAM に描き込む．縦8×横16ドット)

2：奇数行表示モード(文字パターンの奇数行だけを VRAM に描き込む．縦8×横16ドット)

モード0は通常の表示モードです．モード1および2はインタレース表示での漢字表示に使用されます．すなわち SCREEN 文の第6の指定項目の表示指定で3の even/odd インタレースモードを設定すると，VRAM ページ0とページ1（またはページ2とページ3）が交互にしかも，走査線がお互いの間になるように走査されます．この時ページ0に PUT KANJI のモード1で漢字を描き，ページ1の同じ座標にモード2で同じ漢字を描くと，ページ0と1が交互に表示されるため，完全な漢字となって表示されます．(次頁のサンプルプログラム参照) ただし，インタレースモードによる漢字表示は，一般の家庭用テレビを使用した場合にはちらつきが大きく実用的ではありません．

● 参　照　SCREEN 文の表示モード，資料編「漢字コード表」

PUT KANJI

● サンプルプログラム ●〈漢字コード〉&H3021～&H307Eの文字を表示します.

```
10 SCREEN 5
20 CD=&H3021
30 FOR L=0 TO 7
40 FOR N=0 TO 12
50 PUT KANJI(10+17*N,10+18*L),CD
60 CD=CD+1
70 NEXT N,L
80 GOTO 80
```

● SCREEN7のインタレースモードで漢字を表示します.

```
10 SCREEN 7,,,,,3
20 SET PAGE 0,0:CLS:SET PAGE 1,1:CLS
30 CD=&H3021
40 FOR L=0 TO 3
50 FOR N=0 TO 24
60 SET PAGE 0,0:PUT KANJI(20+17*N,
10+10*L),CD,,,1
70 SET PAGE 1,1:PUT KANJI(20+17*N,
10+10*L),CD,,,2
80 CD=CD+1
90 NEXT N,L
100 GOTO 100
```

注) CTRL + STOP でテキスト画面にもどすと，文字が上下にゆれているような感じになります。これはインタレース表示のためです．通常の表示にもどすために次の命令を実行してください.

SCREEN,,,,,0

P

PUT SPRITE

プット スプライト(ステートメント)

● 機　能　スプライトパターンを表示する.

● 書　式　PUT SPRITE 〈スプライト面番号〉〔, | (X, Y) / STEP (X, Y) |〕〔, 〈カラーコード〉〕〔, 〈スプライトパターン番号〉〕

● 文　例　PUT SPRITE 1, (50, 50), 8, 2

● 解　説　指定するスプライト面にあらかじめ定義してあったスプライトパターンを表示します.

〈スプライト面番号〉は0～31の値を持つ数式で指定します.
1つのスプライト面には1つのスプライトパターンのみを表示することができます. このため, 以前に同じスプライト面へスプライトパターンを表示していてもそれは消去され, 新たに指定したスプライトパターンが表示されます.
スプライト面0～31に同時にスプライトパターンを表示することができますが, スプライトパターンが水平方向に5個以上並ぶと, 4個までは表示されますが, 5つ目以降は画面から消えます. ただし, SCREEN 4～8では, 水平方面に8つまで並べて表示することができます.
また, 番号の小さいスプライト面ほど優先順位が高いため, 同じ座標に表示される複数のスプライトパターンがあれば, 番号の小さいスプライト面に表示されているスプライトパターンが手前に重なって見えます.

(X, Y)はスプライトパターンを表示する座標を指定します. スプライトパターンの左上角をこの座標に合わせて表示します.
Xは－32～255の範囲の数式で指定します. また, Yは－32～191(SCREEN 5～8では－32～211)の範囲の数式で指定します.
注) SCREEN 6, 7では横方向Xのドット数は512に増加しますが, スプライトの座標系は255のままで変わりません. またスプライトパターンの横幅の大きさも同様です.

PUT SPRITE

Y座標に208(SCREEN 4 ～ 8 では216) を指定するとそのスプライト面以降のスプライトパターンはすべて画面から消えます.

Y座標に209(SCREEN 4 ～ 8 では217) を指定すると指定したスプライト面に表示されていたスプライトパターンが消えます.

STEP(X, Y)を付けると, LP(最終参照点)からの相対指定となります.

〈カラーコード〉はスプライトパターンの色を指定します. カラーコードと色との対応についてはCOLOR文を参照してください.
SCREEN 8 のモードでのスプライトは16色でパレット機能はありません.
カラーコードと色との対応は, 初期状態での割り当てと同じです. (1.16「カラーコード」参照)

〈スプライトパターン番号〉はSPRITE$ 文であらかじめ定義しておいた, 表示させたいスプライトパターンの番号です.

スプライトパターンのサイズは, 8×8, 16×16, または32×32ドットでSCREEN文で指定します.

PUT SPRITE文の各パラメータを省略すると, スプライト面の現在の値が用いられます. また〈カラーコード〉を省略すると前景色となります.

SCREEN 4 ～ 8 のモードではCOLOR SPRITE文により1ラインごとに色を付けることができます. この場合のPUT SPRITE文のカラーコードは省略(指定しない) してください. (指定するとすべてのラインが指定された色で表示されます.)

例) PUT SPRITE 2, (X, Y),, 3

● 参 照 SPRITE$, SCREEN, COLOR SPRITE, 第1章「スプライト機能」

PUT SPRITE

● サンプルプログラム

```
10 '*** チョウチョウ ****************
20 DEFINT A-Z
30 DIM X(10),Y(10)
40 SCREEN 2,3:COLOR ,1,1:CLS:I=RND(-TIM
E)
50 FOR I=1 TO 10:X(I)=96:Y(I)=I*15:NEXT
60 FOR I=0 TO 31:READ A$:B$=B$+CHR$(VAL
("&h"+A$)):NEXT
70 SPRITE$(0)=B$
80 FOR I=1 TO 10
90 PUT SPRITE I,(X(I),Y(I)),I+4,0
100 NEXT
110 FOR I=1 TO 10
120 X(I)=(X(I)+(RND(1)*21-10)) MOD 256
130 Y(I)=(Y(I)+(RND(1)*21-10)) MOD 192
140 NEXT
150 GOTO 80
160 DATA 0C,0C,62,F2,FA,DD,CF,C7
170 DATA FF,7F,3F,1B,37,3E,1C,00
180 DATA 30,30,46,4F,5F,BB,F3,E3
190 DATA FF,FE,FC,D8,EC,7C,38,00
```

READ

リード(ステートメント)

● 機　能　DATA 文中に含まれているデータを読み，変数に代入する．

● 書　式　READ 変数〔,〈変数〉…〕

● 文　例　READ K, M, S$

● 解　説　READ 文は DATA 文と組合わせて使います．1つのプログラム中に複数の DATA 文がある場合，これらに含まれるデータは一連のデータの並びとみなされます．READ 文は行番号の小さい DATA 文の中のデータから順番に1対1対応の方法で変数に代入していきます．

READ 文の変数は数値変数でも文字変数でもかまいません．**しかし読み出された値と変数の型は一致していなければなりません**．もし型が一致しない場合には，"Syntax error" が起こります．

〈変数〉を1つ以上指定する場合はカンマ（,）で区切って並べます．

1つの READ 文が1つまたはそれ以上の DATA 文を参照したり（複数の場合は順番に参照されます），またいくつかの READ 文が1つの DATA 文を参照することができます．

〈変数〉の数が DATA 文のデータの個数を越えてしまった場合は，"Out of DATA" のエラーメッセージが表示されます．指定された変数が DATA 文のデータの個数よりも少ない場合には，その次の READ 文が読まれなかったデータから読み始めます．もしそれ以上 READ 文がない場合には，余分のデータは無視されます．

始めから DATA 文を読みなおすには，RESTORE 文を使います．

● 参　照　RESTORE, DATA

READ

● サンプルプログラム

```
10 PRINT "   ****** ｾｲｾｷﾋｮｳ *********":PRINT
20 PRINT "   ﾅﾏｴ      ｺｸｺﾞ ｽｳｶﾞｸ ｺﾞｳｹｲ":PRINT
30 FOR I=1 TO 4
40 READ A$,B,C
50 PRINT TAB(2) A$;
60 D=B+C
70 PRINT TAB(6) USING "#######";B;C;D
80 NEXT I
90 DATA ﾀﾛｳ,90,73
100 DATA ﾊﾅｺ,85,80
110 DATA ｼﾞﾛｳ,82,65
120 DATA ﾖｼｺ,75,92
```

REM

リマーク(ステートメント)

● 機　能　プログラムに注釈を入れる.

● 書　式　REM〔〈注釈文〉〕

● 文　例　REM TEST　PROGRAM

　　TEST　PROGRAM

● 解　説　REM 文は非実行文であり，プログラムの実行に影響を与えません. REM 文はプログラムを見易すくするための覚え書きとして使います. プログラムのリストを取ると入力した内容がそのまま出力されます.

REM文はGOTO文，GOSUB文，などの飛先として使うことは可能です. この場合，REM文以降にある最初の実行文から実行されます.

REM 文ではキーワード "REM" の代わりにアポストロフィ(')を使うことができます. REM 文の後にはコロンで区切って複文にすることはできません.

DATA 文中に REM 文を置くことはできません.

● サンプルプログラム

```
10 '*** ﾅﾆﾓｼﾅｲ ﾌﾟﾛｸﾞﾗﾑ ******
20 REM ｺﾉｷﾞｮｳﾊ ｵﾎﾞｴｶﾞｷﾃﾞｽ
30 ' ｱﾎﾟｽﾄﾛﾌｨ(')ﾖ REMﾉ ｶﾜﾘﾆ ﾂｶｴﾏｽ
40 END
```

RENUM

リナンバー(ステートメント)

● 機　能　プログラムの行番号をつけ直す.

● 書　式　RENUM 〔〈新行番号〉〕〔,〈旧行番号〉〕〔,〈増分〉〕

● 文　例　RENUM

● 解　説　〈新行番号〉は，新しくつける行番号の最初の行番号で，省略値は10です.

〈旧行番号〉は行番号のつけ替えをはじめる現在のプログラムの行番号です.
省略値はそのプログラムの最初の行番号です.

〈増分〉は新しく付ける各行番号のあいだの増分で，省略値は10です.

RENUMコマンドは，GOTO, GOSUB, THEN, ON・・・GOTO, ON・・・GOSUBおよびERL文などで参照している行番号があれば新しい行番号に合わせて変更します．これらの文が参照している行番号の行が存在しない場合には，"Undefined line xxxxx in yyyyy" というエラーメッセージが表示されます．この場合，誤った行番号(xxxxx)は変更されません.

RENUMコマンドをプログラム行の順序を変えるのに使うことはできません．また65529以上の行番号を発生することもできません．このような場合には "Illegal function call" エラーが起ります.

RESTORE

リストア(ステートメント)

● 機　能　READ 文で読む DATA 文を指定する.

● 書　式　RESTORE 〔〈行番号〉〕

● 文　例　RESTORE　800

● 解　説　〈行番号〉を省略すると，次に実行するREAD文はプログラム中の最初のDATA文のデータから読み始めます.

● 参　照　READ, DATA

RESUME*

リシューム(ステートメント)

● 機　能　エラー回復処理を終了し，プログラムの実行を再開する．

● 書　式

RESUME	〔0〕
	NEXT
	〈行番号〉

● 文　例　RESUME 0

RESUME NEXT

RESUME 1000

● 解　説　プログラムの実行を再開する場所に応じて次のように3つの書式を選ぶことができます．

RESUME	エラーの原因となった文からプログラムの実行が再開されます．
RESUME 0	
RESUME NEXT	エラーの原因となった文のすぐ次の文から実行が再開されます．
RESUME〈行番号〉	〈行番号〉で指定した行から実行が再開されます．

● 参　照　ON ERROR GOTO

RETURN

リターン(ステートメント)

● 機　能　サブルーチンから復帰する.

● 書　式　RETURN〔〈行番号〉〕

● 文　例　RETURN

RETURN 300

● 解　説　RETURN 文はサブルーチンを終了し，サブルーチンを呼び出したGOSUB文の次の文へ復帰（実行を開始）します．また，割込み処理ルーチンで使うと，割込み処理を終了し，割込みがかかったときに中断されたプログラムが再開されます.

〈行番号〉を指定すると，サブルーチンや割込み処理ルーチン終了後，〈行番号〉へジャンプします.

サブルーチンや割込み処理ルーチンからの復帰には必ずRETURN文を使ってください．また，1つのサブルーチン中に複数個のサブルーチンがあってもかまいませんが，GOSUB文と正しく対応しなければなりません.

サブルーチン中でCLEAR文を実行すると，RETURN文による復帰ができなくなります.

● 参　照　GOSUB, ON GOSUB,第1章「割込み」

RIGHT$

ライトダラー(関数)

● 機　能　文字列の右側から指定した長さの文字列をとり出す.

● 書　式　RIGHT$ (〈文字列〉, 〈数式〉)

● 文　例　PRINT RIGHT$(A$, 4)

● 解　説　〈文字列〉は文字式で表します.〈数式〉の値の範囲は0から255です.
〈数式〉の値が0ならば, RIGHT$はヌルストリングを値とします.
〈数式〉が〈文字列〉の文字数より大きければ,〈文字列〉のすべてを値とします.

グラフィックキャラクタは1字あたり, 2文字としてカウントします.

● 参　照　LEFT$, MID$

● サンプルプログラム

```
10 CLS
20 INPUT"ﾓｼﾞﾚﾂ";A$
30 L=LEN(A$)
40 FOR I=1 TO L
50 B$=RIGHT$(A$,I)
60 B$=RIGHT$(SPACE$(50)+B$,L)
70 PRINT B$
80 NEXT
90 END
```

RND

ランダム(関数)

● 機　能　乱数を得る.

● 書　式　RND〔(〈数式〉)〕

● 文　例　R=RND (1)

● 解　説　0以上1未満の乱数を発生します.

発生される乱数は〈数式〉の値によって次のように異なります.

〈数式〉が正の場合――RUN文が実行されるごとに同系列の乱数を発生.

ただし，事前にRND(－〈数式1〉)関数を実行し，〈数式1〉の数値（数値＞0）を変化することによって，発生させる乱数の系列を変えることができます.

〈数式〉が0の場合――直前に発生した乱数の値をとります.

〈数式〉が負の場合――〈数式〉の値によって決まる値を発生します．乱数系列を初期化する働きもします.

● サンプルプログラム

```
10 '*** ラクガキ ***
20 SCREEN 2
30 R=RND(-TIME)
40 X=RND(1)*256:Y=RND(1)*192
50 L=RND(1)*16+16
60 XS=X:YS=Y
70 X=X+RND(1)*L-L/2
80 Y=Y+RND(1)*L-L/2
90 C=INT(RND(1)*16)
100 LINE(XS,YS)-(X,Y),C
110 IF X<0 OR X>255 THEN 40
120 IF Y<0 OR Y>191 THEN 40
130 GOTO 60
```

＊30行がない場合には，RUN文実行ごとに毎回同じ系列の乱数を発生.

RUN

ラン(コマンド)

● **機　能**　**プログラムを実行する**

● **書　式**　**RUN〔〈行号〉〕**　　**RUN "〈ファイルスペック〉"**

● **文　例**　**RUN 100**　　**RUN "CAS:PROG1"**

● **解　説**　〈行番号〉を指定すると，その行から実行がはじまります．指定のない場合には，最も若い行番号から実行がはじまります．

〈ファイルスペック〉は，指定されたBASICのプログラムをロード後すぐに実行を開始します。

LOAD"〈ファイルスペック〉"，Rと同じ働らきをします。

SAVE

セーブ(コマンド)

● **機　能**　**メモリのBASICプログラムをアスキー形式でファイルにセーブする．**

● **書　式**　**SAVE "〈ファイルスペック〉"**

● **文　例**　**SAVE "CAS:MYPROG"**

● **解　説**　〈ファイルスペック〉で指定するファイルにアスキー形式で プログラムを書き込みます。

アスキー形式でプログラムをセーブしておくと，プログラムはCSAVE文で作成するバイナリ形式ファイルのように圧縮されず，アスキーキャラクタ形式のままファイルに保存されます．また，プログラムの各行がキャリッジリターンコード＋ラインフィードコードで区切られてセーブされます．

アスキー形式のファイルはバイナリ形式のファイルよりも多くのファイルスペースを必要としますが，MERGEコマンド(プログラムの混合)を使うことができます．また，データファイルとして読み出すこともできます．

● **参　照**　**LOAD, MERGE**

SCREEN

スクリーン(ステートメント)

● 機　能　画面のモード，スプライトサイズ，キークリック，カセットボーレート，プリンタの種類，表示モードを指定する．

● 書　式　**SCREEN** 〔〈画面モード〉〕〔,〈スプライトサイズ〉〕〔,〈キークリック〉〕〔,〈カセットボーレート〉〕〔,〈プリンタの種類〉〕〔,〈表示モード〉〕

● 文　例　**SCREEN　7,S,,,1,0**（Sは0～3の数値を持つ変数）

● 解　説　〈画面モード〉

0～8の数値で画面モードを指定します．画面モードについては，第1章の「画面モード」で説明していますのでそちらを参照してください

SCREEN文を実行すると，新しい画面モードの0ページに移り0ページの画面とスプライトは消去されます．スプライトパターンはそのまま残っています．

VRAM容量が64KBの機種では画面モード7と8は使えません．

画面モード

0　：最大80桁×24行のテキストモード

1　：最大32桁×24行のテキストモード

2　：横256×縦192ドットの高解像度グラフィックモード

3　：横64×縦48ブロックのマルチカラーモード

4　：横256×縦192ドットの高解像度グラフィックモード

5　：横256×縦212ドットのビットマップグラフィックモード

6　：横512×縦212ドットのビットマップグラフィックモード

7　：横512×縦212ドットのビットマップグラフィックモード

8　：横256×縦212ドットのビットマップグラフィックモード

〈スプライトサイズ〉

0～3の値を持つ式または数値で指定します．スプライトサイズを指定するとそれまでに定義されていたスプライトパターンは全て消されてしまいます．

0：スプライトを8×8画素で構成する．

1：スプライトを8×8画素で構成し，縦横それぞれ2倍に拡大して表示する．

2：スプライトを16×16画素で構成する．

3：スプライトを16×16画素で構成し，縦横それぞれ2倍に拡大して表示する．

初期設定はモード0の状態です．

〈キークリック〉

キーを押したときクリック音を出すかどうかを指定します.

0：キークリック音を出さない.

1：キークリック音を出す.(初期はこの状態に設定されている)

〈カセットボーレート〉

カセットテープに書き込むときのボーレートを指定します. 以後CSAVE, BSAVE コマンドでボーレートを省略すると，ここで指定したボーレートで書き込みます.

1：1200ボー（初期は，この状態に設定されている）

2：2400ボー

カセットテープからの読み込み時は，ボーレートは自動的に選択されますので，読み込もうとするカセットテープがどのボーレートで書き込まれていたかを知っている必要はありません。

〈プリンタオプション〉

使用しているプリンタが，MSX仕様のプリンタかどうか指定します.

0：MSX仕様のプリンタ(初期はこの状態に設定されている)

0以外：MXS仕様でないプリンタ

0以外を指定した場合は，グラフィック文字は空白に，ひらがなはカタカナに変換してプリンタに出力されます.

〈表示モード〉

表示画面の表示モードを0～3の値を持つ式または数値で指定します.

指定値	走査モード	表 示 ペ ー ジ
0	ノンインタレース	ディスプレイページ
1	インタレース	
2	ノンインタレース	Even/odd (ディスプレイページ/ディスプレイページ－1)
3	インタレース	

0は通常の表示モードで，初期もこの状態に設定されています.

1は，ディスプレイページのみでインタレース表示を行ないます. 文字や絵が上下に細かくゆれるような感じの表示になります.

SCREEN

2と3は，ディスプレイページが1または3である必要があり，ディスプレイページとそれより1つ小さい番号のページを交互に表示します．モード3は，PUT KANJI文のモード1，2との組合せにより表示モード0での漢字より高さを½に圧縮した漢字を表示させることができます．(PUT KANJI文のサンプルプログラム参照)．しかし表示モード2，3による表示は一般の家庭用テレビをディスプレイとして使用した場合は，ちらつきが大きく実用的ではありません．

SET

セット(ステートメント)

● 機　能　時刻や表示画面位置など各種のセットを行ないます.

一度セットされた値は，再度セットしなおすまで有効です．（バックアップ電池により保持されていますので，電源をオフにしてもセット内容は保持されています.）

◆テレビ画面の表示位置を上下左右に調整する.

● 書　式　SET ADJUST(X,Y)

● 文　例　SET ADJUST(-3,5)

● 解　説　テレビ画面の表示位置を指定したドット位置に移動します.

（０，０）が標準位置で，Ｘ，Ｙはそれぞれ－７～＋８の値です.

数値を大きくすると右(Ｘの場合)または下(Ｙの場合)に移動します.

◆パスワードをセットする.

● 書　式　SET PASSWORD "〈パスワード〉"

● 文　例　SET PASSWORD "MSX2"

● 解　説　パスワードをセットします．パスワードをセットすると，電源オンのとき〈パスワード〉を入力しなければその装置が使えないようになります.

〈パスワード〉は255文字以内です．あなただけの秘密の言葉をセットしてください.

〈パスワード〉を忘れてしまったときは，GRAPHキーとSTOPキーを押しながら電源をオンまたはリセットすると起動することができます．どうしても思い出せない場合は，新しいパスワードをセットしてください.

注）・〈パスワード〉入力モードのときは，入力した文字列は表示されません.

・SET PROMPTまたはSET TITLEを実行すると，パスワードは無効となります.（セットされていない状態にもどる.）

SET

◆BASICのプロンプト〝OK〟を変更する.

● **書　式**　**SET PROMPT** "〈プロンプト〉"

● **文　例**　**SET PROMPT** "ヨロシイ"

● **解　説**　テキスト画面におけるコマンド受付モードになった時に表示されるプロンプトを変更します. 6 文字以内で定義してください.

注）・SET PASSWORDまたはSET TITLEを実行するとプロンプトは〝OK〟にもどります.

◆タイトルを指定する.

● **書　式**　**SET TITLE** "〈タイトル〉"〔,〈背景色〉〕

● **文　例**　**SET TITLE** "AKIRA"

● **解　説**　電源オン時に表示される〈タイトル〉とその背景色を指定します.

〈タイトル〉は初期画面でMSXマークの下に表示されます. 文字数は 6 文字以内ですが, ちょうど 6 文字の場合は何かキーを押すまで初期画面のままとなります. 背景色は 1 ～ 4 で, 1 は青, 2 は緑, 3 は赤, 4 はオレンジです. 背景色指定を省略した場合は現在設定されている色を保持します.

注）・SET PASSWORDまたはSET PROMPT を実行するとタイトルは無効となります.（設定されていない状態にもどる. 背景色は青になる.）

◆現在の表示パラメータを初期値として登録する.

● **書　式**　**SET SCREEN**

● **文　例**　**SET SCREEN**

● **解　説**　現在の表示文字幅（WIDTH）と SCREEN モード（0 または 1）などを初期値として登録します. 以後電源オン時は, 登録したモードでスタートします. 登録内容は次の通りです.

画面モード（0 または 1）, 表示文字幅, キークリック, カセットボーレート, プリンタの種類, 表示モード, 前景色, 背景色, 周辺色

● **参　照**　SCREEN, COLOR

SET

◆日付をセットする

● 書　式　**SET DATE "〈日付〉" 〔,A〕**

● 文　例　**SET DATE "85/11/20"**

● 解　説　〈日付〉を内蔵の時計にセットします．年，月，日を〝86/12/01〟のように〝/〟で区切って，各2桁ずつ入力してください．〝01〟のように1桁の部分は数字の前に0を付けてください．

オプションのAを付けると，この日付にアラーム出力端子が低レベルになります．（日付と時刻両方にアラームがセットされている場合は，SET TIMEを参照）

注）・アラーム出力端子のない機種ではオプションAを付けても無効です．

・アラーム出力端子への外部機器の接続条件などは，取扱説明書を参照してください．

● 参　照　GET，SET TIME

◆時刻をセットする

● 書　式　**SET TIME "〈時刻〉" 〔,A〕**

● 文　例　**SET TIME "21:05:16"**

● 解　説　〈時刻〉を内蔵の時計にセットします．時，分，秒を24時間時計で〝21：05：16〟のように〝：〟で区切って，各2桁ずつ入力してください．〝05〟のように1桁の部分は数字の前に0を付けてください．

オプションのAを付けると，セットされた時刻(アラームは時分のみ有効，秒はダミー）に1分間だけアラーム出力端子が低レベルになります．日付のアラームもセットされていると指定した日の指定した時刻の時だけアラームが動作します．この場合，時刻のアラームをセットすると，日付のアラームはリセットされますので，日付のアラームは時刻のアラームの後でセットしてください．このアラーム機能は電源オフの状態でも指定時刻になると動作します．

注）・アラーム出力端子のない機種ではオプションAを付けても無効です．

・アラーム出力端子への外部機器の接続条件などは，取扱説明書を参照してください．

● 参　照　GET，SET DATE

SET

◆VRAM上の使用するページを指定

● 書　式　**SET PAGE** 〈ディスプレイページ〉,〈アクティブページ〉

● 文　例　**SET PAGE 2, 1**

● 解　説　〈ディスプレイページ〉は表示されるページ，〈アクティブページ〉はVRAMへのデータの書込みや読出しの行なわれるページです．指定できるページはSCREENモードや，使用機種のVRAM容量によって以下のように異なります．

使用機種	VRAM64KBの機種	VRAM128KBの機種
SCREEN 5	0～1ページ	0～3ページ
SCREEN 6	0～1ページ	0～3ページ
SCREEN 7	使用不可	0～1ページ
SCREEN 8	使用不可	0～1ページ

〈参考〉PUT KANJIのサンプルプログラム参照

◆BEEP音をセットする

● 書　式　**SET BEEP 〈音色〉,〈大きさ〉**

● 文　例　**SET BEEP 1, 3**

● 解　説　BEEP命令によって出力される音の音色と大きさをそれぞれ1～4の数値でセットします．音色は4種類の音より1つを選びます．大きさは数値が大きい程大きくなります．

● 参　照　BEEP

SET

◆スーパーインポーズモードのセット

● 書　式　SET VIDEO 〈モード〉〔,〈YM〉〕〔,〈CB〉〕〔,〈同期〉〕〔,〈音声〉〕〔,〈ビデオ入力〉〕〔,〈AVコントロール〉〕

● 文　例　SET VIDEO 2,,,1

スーパーインポーズのモードを設定します.

● 解　説　〈モード〉は0～3です.(表中のYM, 同期は, 各々表に示したパラメータの設定ができます.)

モード	画　面	Y　M	同　期
0	コンピュータの画面	——— (関係なし)	内　部
1			内部, 外部
2	スーパーインポーズ	0, 1	外　部
3	テレビ		

〈YM〉は, テレビ画面の輝度を制御します. 0であれば通常の画面, 1にすると輝度を半分にします.

〈CB〉は, VDPのカラーバスの状態を制御します. 0であれば出力状態, 1とすると入力状態です.

〈同期〉は1であれば外部同期が、0であれば内部同期が選択されます.

〈音声〉は, 下表のように指定します.

〈音声〉	機　能
0	外部音声信号を混合しない
1	右チャンネルの外部音声信号を混合する
2	左チャンネルの外部音声信号を混合する
3	両チャンネルの外部音声信号を混合する

〈ビデオ入力〉は, 外部映像信号の入力を切換えます. 0であればRGBマルチコネクタ(21ピンリニアRGB)の入力が、1であれば, 外部映像信号端子からの入力が選択されます.

〈AVコントロール〉は, RGBマルチ(21ピンRGB)コネクタのAVコントロール信号の出力を指定します.0ならば出力がOFFに,1ならばONになります. ただし, AVコントロール信号線のないテレビに対しては無効です.

SET VIDEO文は, ご使用のパソコン, テレビおよびその接続のしかたなどによって動作が無効となるようなケースもありますのでご注意ください.

● 参　照　付録「スーパーインポーズとディジタイズ」

SGN

サイン(関数)

● 機　能　正負符号を調べる.

● 書　式　SGN（＜数式＞）

● 文　例　PRINT SGN(A)

● 解　説　〈数式〉が正の場合は1を，〈数式〉が0の場合は0を，〈数式〉が負の場合には－1を結果として得られます.

● サンプルプログラム

```
10 INPUT"ｽｳｼﾞ";I
20 J=SGN(I)
30 J=J+2
40 ON J GOSUB 90,100,110
50 PRINT A$;
60 PRINT " ﾃﾞｽ。"
70 PRINT
80 GOTO 10
90 A$=" ﾏｲﾅｽ":RETURN
100 PRINT "ｾﾞﾛ";:RETURN 60
110 A$="ﾌﾟﾗｽ" :RETURN
```

SIN

サイン(関数)

● 機　能　正弦（サイン）を得る.

● 書　式　SIN (〈数式〉)

● 文　例　PRINT SIN (1)

● 解　説　〈数式〉の正弦（サイン）を求めます.〈数式〉の単位はラジアンです.
〈数式〉は整数型，単精度実数型，倍精度実数型のいずれでもかまいませんが，結果は倍精度で得られます.

● 参　照　COS, TAN

● サンプルプログラム

```
10 '＊＊＊　ウズマキ　＊＊＊
20 K=75
30 PI=3.141592653#
40 SCREEN 2
50 PSET(128,177)
60 FOR I=0 TO PI*2 STEP PI/30
70 K=K-.25
80 IF K<0 THEN 140
90 X=128+K*SIN(I)
100 Y= 88+K*COS(I)*1.2
110 LINE -(X,Y)
120 NEXT
130 GOTO 60
140 GOTO 140
```

SOUND ＊

サウンド（ステートメント）

● 機　能　PSG（プログラマブルサウンドジェネレータ）のレジスタに値を書き込む．

● 書　式　SOUND　〈レジスタ番号〉,〈データ〉

● 文　例　SOUND 7, 254: SOUND 8, 8: SOUND 0,255

● 解　説　PSGのレジスタを直接書き換えて音を出します．PLAY文のように簡単に音を出すわけには行きませんが，細い指定をすることができるので，いろいろな音を出すことができます．

PSGの音の発生装置と音量の制御装置は3チャンネルに独立していますので，別々に指定することができます．ただし，ノイズの発生装置とエンベロープの発生装置は各チャンネルで共用します．

PSGに用意されているレジスタは次のとうりです．

レジスタ＼ビット		B7	B6	B5	B4	B3	B2	B1	B0
R 0	チャンネルA周波数	8BIT				FT A			
R 1						4BIT CT A			
R 2	チャンネルB周波数	8BIT				FT B			
R 3						4BIT CT B			
R 4	チャンネルC周波数	8BIT				FT C			
R 5						4BIT CT C			
R 6	ノイズ周波数					5BIT NP			
R 7	チャンネル設定	IN OUT		NOISE			TONE		
		IOB	IOA	C	B	A	C	B	A
R 8	チャンネルA音量				M	L3	L2	L1	L0
R 9	チャンネルB音量				M	L3	L2	L1	L0
R10	チャンネルC音量				M	L3	L2	L1	L0
R11	エンベロープ周期	8BIT FT							
R12		8BIT CT							
R13	エンベロープ形状					E3	E2	E1	E0

SOUND*

R0, R1	チャンネルAの周波数を2つのレジスタを使って指定する.
R2, R3	チャンネルBの周波数を2つのレジスタを使って指定する.
R4, R5	チャンネルCの周波数を2つのレジスタを使って指定する.
R6	ノイズ周波数を指定する.
R7	各チャンネルから音を出すかどうかを指定する.
R8, R9, R10	チャンネルA, B, Cのそれぞれの音量を指定する. L0～L3を0にすると無音に, 15にすると最大の音量になります. エンベロープの指定をするときは, Mを1にします.
R11, R12	エンベロープの周期を2つのレジスタを使って指定する.
R13	エンベロープ形状を指定する.

● **周波数の設定 (R0～R5)**

$$TP=\frac{fclock}{16\times fT} \qquad CT+\frac{FT}{256}=\frac{TP}{256}$$

fclock　：1.78977MHz
fT　：出力周波数
FT　：FTA (R0) の値
CT　：CTA (R1) の値

例)
440Hzを出力する

$$TP=\frac{1.78977\times10^{6}}{16\times440}\fallingdotseq254$$

$$CT+\frac{FT}{256}=\frac{254}{256}$$

R0には　FT=254
R1には　CT=0
を設定します.

● **ノイズ周波数の設定 (R6)**

$$NP=\frac{fclock}{16\times fN}$$

fclock　：1.78977MHz

SOUND*

fN　　：ノイズ周波数（Hz）

NP　　：R6のNPに設定する値

例）

30KHzのノイズ周波数を出力する．

$$NP=\frac{1.78977\times10^{6}}{16\times\ (30\times10^{3})}\fallingdotseq 4$$

R6には　NP＝4

を設定します．

●チャンネルの設定（R7）

使用したいチャンネルに対応するビットを0にする．

```
B7---------------B0
 1 0 0 0 0 0 0 0
 |                チャンネルAから音を出す
 |                チャンネルBから音を出す
 |                チャンネルCから音を出す
 |                チャンネルAからノイズを出す
 |                チャンネルBからノイズを出す
 |                チャンネルCからノイズを出す
```

例）

チャンネルA，Cから音を出す（他の動作はしない）

R7＝&B10111010

●音量設定（R8～R10）

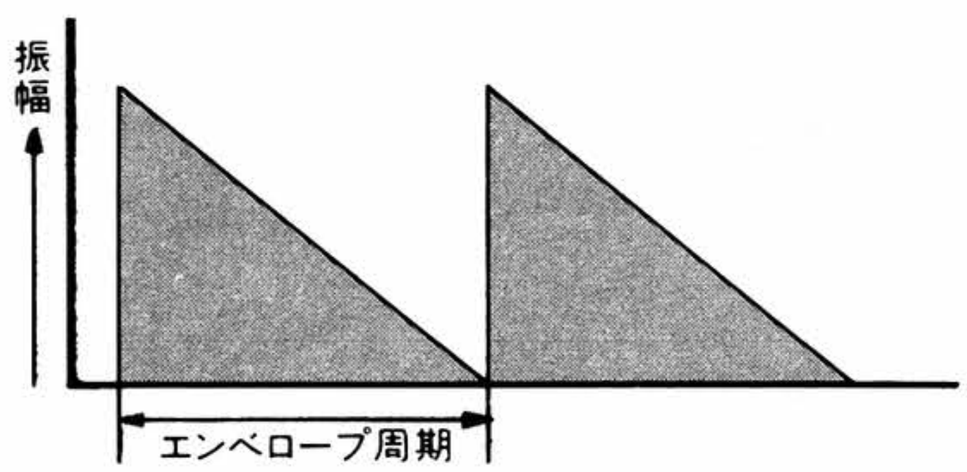
音量設定には大きさが一定のものと，時間的に変化するエンベロープモードとがあります．エンベロープモードを指定している場合はさらにエンベロープの形状とその周期を指定します．

SOUND*

B7---------- B0
×××1××××

音量を0～15の範囲(ビット列0000～1111)で指定する．0の場合は無音となります．

1であればエンベロープモード，0であればB3～B0で決まる音量となります．



●エンベロープ周期の設定 (R11, R12)

$$EP=\frac{fclock}{256fE} \qquad CT+\frac{FT}{256}=\frac{EP}{256}$$

fclock　：1.78977MHz
fE　：音が増加・減衰する周波数Hz
FT　：FT（R11）に設定する値
CT　：CT（R12）に設定する値

例）
音が減衰して0になるまでの時間を2秒にする

$$EP=\frac{1.78977\times10^{6}}{256\times0.5}=14.000$$

$$CT+\frac{FT}{256}=\frac{14.000}{256}=54+\frac{176}{256}$$

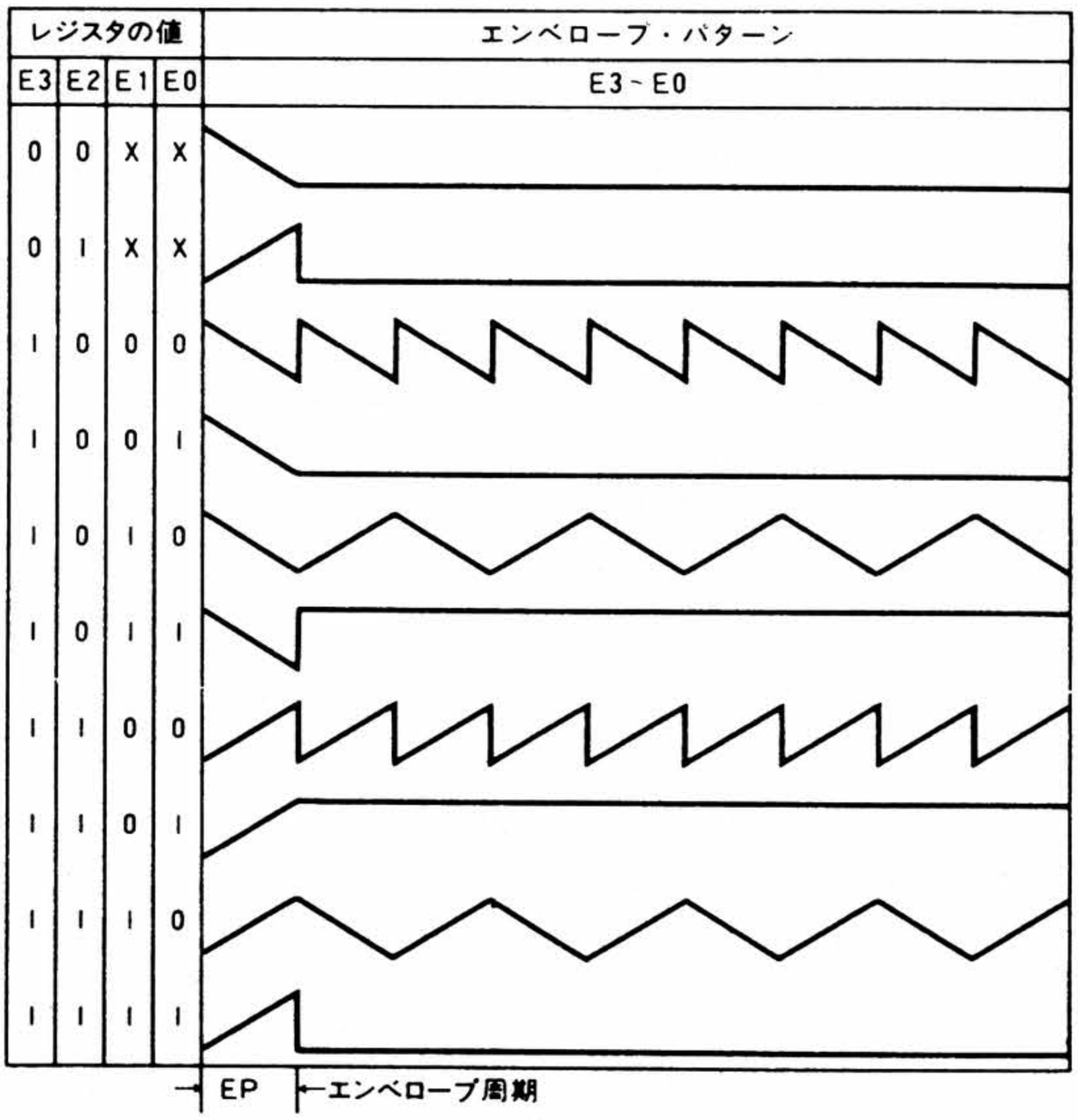
R11にはFT＝176（10進）を設定する
R12にはCT＝54（10進）を設定する

SOUND*

● エンベロープ形状の設定（R13）

エンベロープ形状はレジスタR13の下位4ビット（E0～E3）へ0～15（ビット列0000～1111）で指定します．

エンベロープ発生器出力



● サンプルプログラム

```
10 PRINT" ランダム ナ オト デス。"
20 FOR I=0 TO 13
30 SOUND I,0:NEXT
40 SOUND 7,254:SOUND 8,15
50 FOR I=1 TO 255 STEP .1
60 GOSUB 100
70 NEXT
80 GOSUB 100
90 GOTO 80
100 R0=INT(RND(1)*I)
110 SOUND 0,R0
120 RETURN
```

SPACE$*

スペース ダラー(関数)

● 機　能　指定した長さの空白の文字列を用意する.

● 書　式　SPACE$ (〈数式〉)

● 文　例　PRINT　SPACE$(10); "ABC"

● 解　説　〈数式〉の数だけの空白をもった文字列を得ます.

〈数式〉の値の範囲は0から255までです.

● 参　照　SPC

● サンプルプログラム

```
10 CLS
20 PRINT "  12345678901234567"
30 FOR J=0 TO 10
40 S=INT(RND(1)*17)
50 PRINT "*";
60 PRINT SPACE$(S); "*";
70 PRINT SPACE$(17-S); "スペース";S
80 NEXT J
```

SPC*

スペース(関数)

● 機　能　画面やプリンタに指定した数だけ空白を出力する.

● 書　式　SPC (〈数式〉)

● 文　例　PRINT SPC(10); A$

● 解　説　SPC 関数は PRINT 文や LPRINT 文など出力文中のみ使用することができます.

〈数式〉の値の範囲は 0 ～255です.

● 参　照　SPACE$, TAB

● サンプルプログラム

```
10 FOR J=1 TO 8
20 PRINT "*****";SPC(J);"0"
30 NEXT J
40 END
```

SPRITE ON/OFF/STOP

スプライト オン／オフ／ストップ（ステートメント）

● 機　能　スプライトパターンが重なったときに起る割込みを許可，禁止，または保留する．

● 書　式　SPRITE ON
SPRITE OFF
SPRITE STOP

● 文　例　SPRITE ON

● 解　説　SPRITE ON 文はスプライト割込みを許可します．この命令を実行しておくと，PUT SPRITE 文で画面にスプライトパターンを出したとき，他のスプライトパターンと重なっていれば割込みがかかり，実行中のプログラムを中断して ON SPRITE GOSUB 文で定義しておいた割込み処理ルーチンが実行されます．

SPRITE OFF 文はスプライト割込みを禁止します．この命令を実行しておくと，PUT SPRITE 文で画面にスプライトパターンを出したとき，他のスプライトパターンと重っていても割込みはかかりません．

SPRITE STOP 文はスプライト割込みを保留します．この命令を実行しておくと，以後にかかったスプライト割込みは SPRITE ON 文を実行するまで保留され，SPRITE ON 文を実行すると直ちに割込み処理ルーチンが実行されます．この命令は SPRITE ON 状態で実行する場合のみ有効です．

● 参　照　ON SPRITE GOSUB, PUT SPRITE

SPRITE$

スプライト ダラー(システム変数)

● 機　能　スプライトパターン

● 書　式　SPRITE$(〈スプライトパターン番号〉)

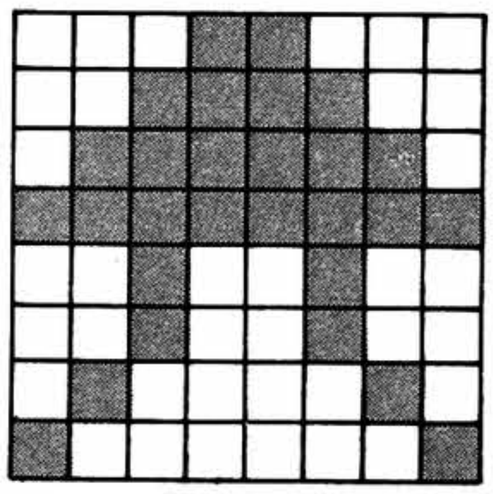
● 文　例　SPRITE$(0)=STRING$(32, CHR$(&HFF))

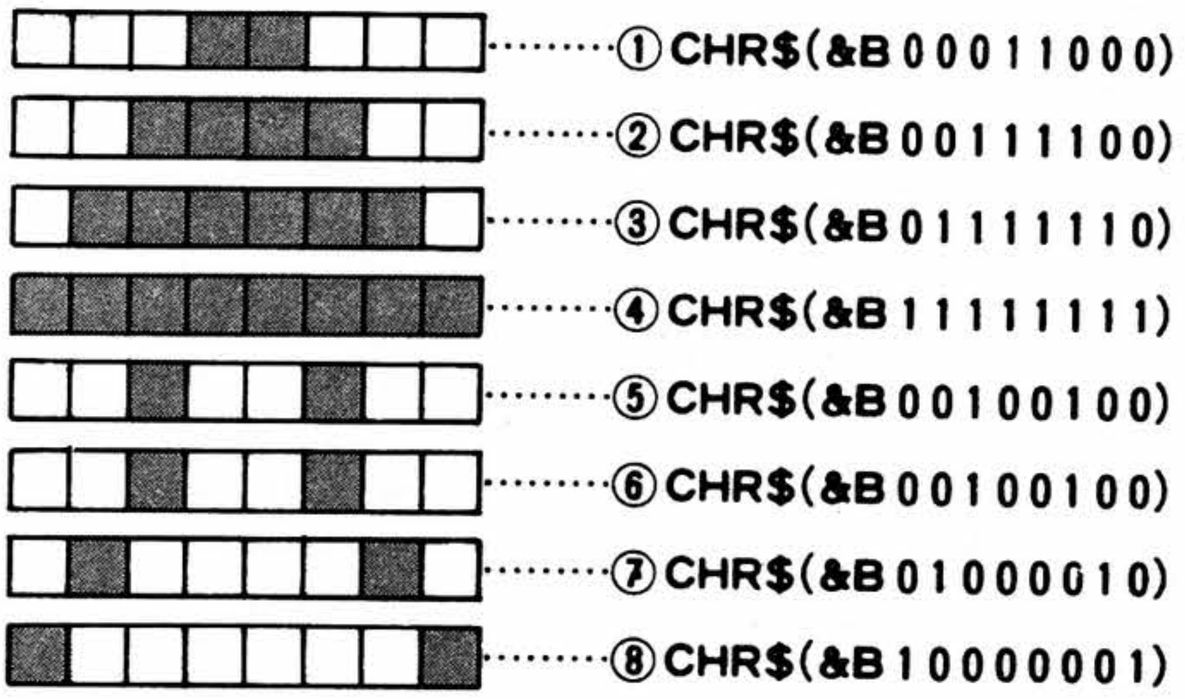
● 解　説　スプライト機能で使用するスプライトパターンを定義します.

〈スプライトパターン番号〉は，SCREEN文でスプライトサイズを0または1（8×8画素）と指定していれば0～255の範囲，スプライトサイズを2または3（16×16画素）と指定していれば0～63の範囲の数式で指定します.

スプライトパターンのデータは8×8画素の場合8文字分の文字列（8バイト），16×16画素の場合32文字分の文字列（32バイト）です.
データの指定の方法を次の図で説明します.

8×8画素のスプライトパターン





横8画素を1組として，それぞれを1バイトのビット列(0と1の列)で表します.
点を描く部分が1で，点を描かない部分は0です.

SPRITE$

このようにしてできた①～⑧の各1バイトを1文字分ずつスプライトパターンの文字列へ順番に並べます.

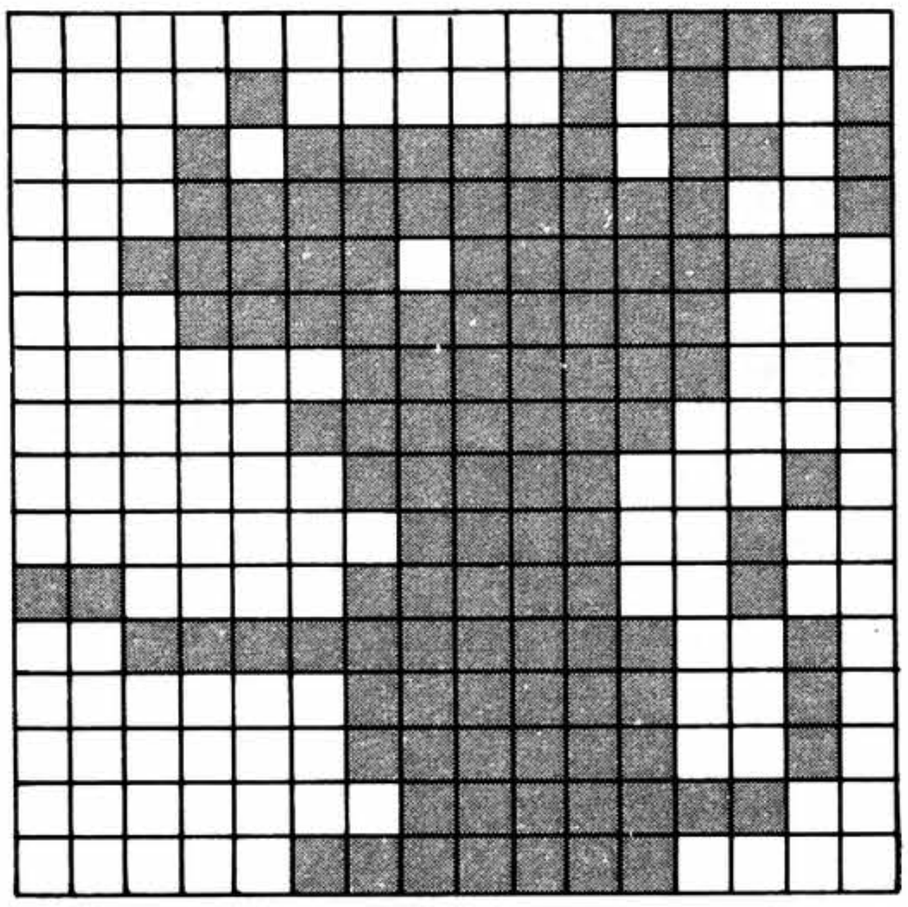
スプライトパターン ⇦

①	②	③	④	⑤	⑥	⑦	⑧

スプライトパターンの定義例

```
10  SCREEN 1, 0
20  FOR I=1 TO 8
30  READ A$:B$=B$ +CHR$(VAL("&B"+A$))
40  NEXT I : SPRITE$(0)=B$          ←── スプライトパターン0のパターンを定義.
50  PUT SPRITE 0, (100, 100), 8, 0  ←── スプライトパターンの表示.
60  DATA 00011000
70  DATA 00111100
80  DATA 01111110
90  DATA 11111111
100 DATA 00100100
110 DATA 00100100
120 DATA 01000010
130 DATA 10000001
```

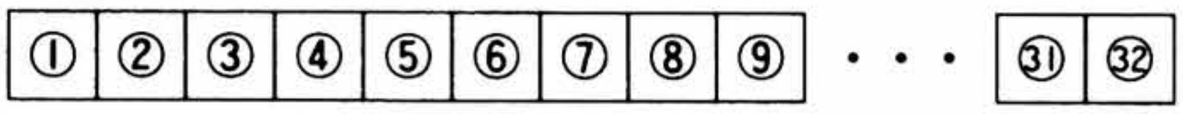
16×16画素のスプライトパターン



① CHR$(&B00000000)
② CHR$(&B00001000)
③ CHR$(&B00010111)
④ CHR$(&B00011111)
⑤ CHR$(&B00111110)
⑥ CHR$(&B00011111)
⑦ CHR$(&B00000011)
⑧ CHR$(&B00000111)
⑨ CHR$(&B00000011)
⑩ CHR$(&B00000001)
⑪ CHR$(&B11000011)
⑫ CHR$(&B00111111)
⑬ CHR$(&B00000011)
⑭ CHR$(&B00000011)
⑮ CHR$(&B00000001)
⑯ CHR$(&B00000111)
⑰ CHR$(&B00011110)
⑱ CHR$(&B00101001)
⑲ CHR$(&B11101101)
⑳ CHR$(&B11111001)
㉑ CHR$(&B11111110)
㉒ CHR$(&B11111000)
㉓ CHR$(&B11111000)
㉔ CHR$(&B11110000)
㉕ CHR$(&B11100010)
㉖ CHR$(&B11100100)
㉗ CHR$(&B11100100)
㉘ CHR$(&B11110010)
㉙ CHR$(&B11110010)
㉚ CHR$(&B11110010)
㉛ CHR$(&B11111100)
㉜ CHR$(&B11110000)

SPRITE$

8×8画素のときと同様に①～㉜の各バイトをスプライトパターンの文字列へ順番に並べます．最初に左半分のパターンを定義し，続けて右半分のパターンを定義します．



スプライトパターンの定義例を次のサンプルプログラムで説明します．

● サンプルプログラム

```
10 SCREEN 1,3
20 PRINT "*** ネズミ ***"
30 FOR I=1 TO 16
40 READ D$
50 A$=A$+CHR$(VAL("&B"+LEFT$(D$,8)))
60 B$=B$+CHR$(VAL("&B"+RIGHT$(D$,8)))
70 NEXT I
80 SPRITE$(0)=A$+B$
90 PUT SPRITE 0,(50,70),15,0
100 PUT SPRITE 1,(90,70),14,0
110 PUT SPRITE 2,(130,70),1,0
120 PUT SPRITE 3,(170,70),13,0
130 PRINT "コノ ネズミヲ ケストキハ"
140 PRINT "PUT SPRITE 0,(0,208)"
150 PRINT "ト キーインシマス。"
160 DATA 0000000000011110
170 DATA 0000100000101001
180 DATA 0001011111101101
190 DATA 0001111111111001
200 DATA 0011111011111111
210 DATA 0001111111111000
220 DATA 0000001111111000
230 DATA 0000011111110000
240 DATA 0000001111100010
250 DATA 0000000111100100
260 DATA 1100001111100100
270 DATA 0011111111110010
280 DATA 0000001111110010
290 DATA 0000001111110010
300 DATA 0000000111111100
310 DATA 0000011111110000
```

左半分のパターン
右半分のパターン
スプライトパターンの表示

SQR

スクエア ルート(関数)

● 機　能　平方根（スクエアルート）を得る.

● 書　式　SQR（〈数式〉）

● 文　例　A＝SQR(2)

● 解　説　〈数式〉で指定された値の平方根を求めます.

〈数式〉の値は0以上でなければなりません.〈数式〉はどの型の数値でもかまいませんが，結果は倍精度で得られます.

● サンプルプログラム

```
10 INPUT I
20 PRINT"SQR(I) =" ;
30 PRINT SQR(I)
40 PRINT"I^.5   =" ;
50 PRINT I^.5
60 PRINT
70 GOTO 10
```

STICK

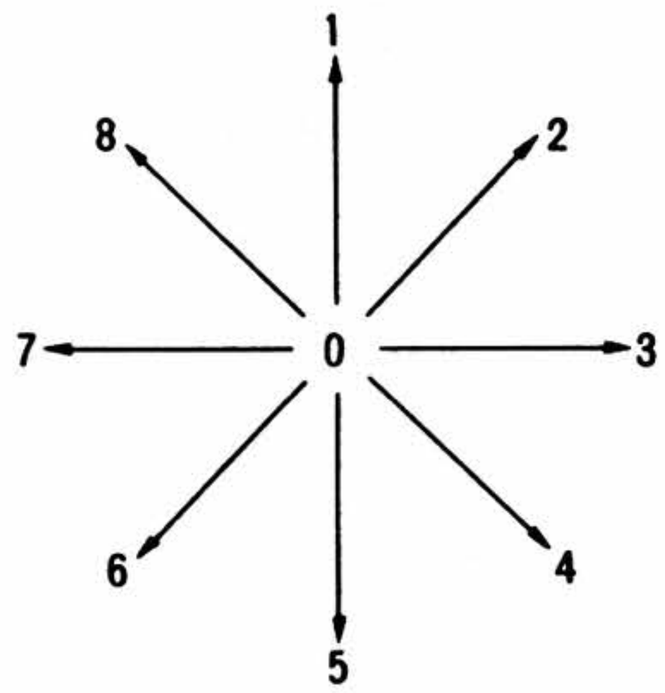
スティック(関数)

● **機　能**　ジョイスティックがどの方向へ押されているかを調べる.

● **書　式**　**STICK (〈ジョイスティックの番号〉)**

● **文　例**　**STICK (1)**

● **解　説**　指定したジョイスティックがどの方向へ押されでいるかを調べ，その結果を次の図に示す値で得ることができます．なお，ジョイスティックがどちらへも押されていなければ値は 0 です.



〈ジョイスティックの番号〉は次のように指定します.

0：ジョイスティックの代わりにキーボードのカーソルキーを使う.(↑が押されていれば 1，→が押されていれば 3，↑と→が同時に押されていれば 2，…という結果になります)

1：ポート 1 につながっているジョイスティック

2：ポート 2 につながっているジョイスティック

〈参考〉マウスのジョイスティックモード.

電源オン時，マウスのトリガボタン I をオンして起動すると，マウスをジョイスティックとして使用することができます.(マウスの状態を STICK 関数で調べることができます.)

注 トラックボールにはこのモードはありません.

STICK

● サンプルプログラム

```
10 SCREEN 1,1
20 SPRITE$(0)=STRING$(8,CHR$(255))
30 X=120:Y=90
40 PUT SPRITE 0,(X,Y),8,0
50 ON KEY GOSUB 230
60 KEY(1) ON
70 PRINT "カーソルキー ヲ オシテクダサイ"
80 PRINT "コノ プログラム ヲ トメルトキ ハ"
90 PRINT "F1キー ヲ オシマス"
100 A=STICK(0)
110 ON A GOTO 130,140,150,160,170,180,190,200
120 GOTO 100
130 Y=Y-1         :GOTO 210
140 X=X+1:Y=Y-1:GOTO 210
150 X=X+1         :GOTO 210
160 X=X+1:Y=Y+1:GOTO 210
170 Y=Y+1         :GOTO 210
180 X=X-1:Y=Y+1:GOTO 210
190 X=X-1         :GOTO 210
200 X=X-1:Y=Y-1:GOTO 210
210 PUT SPRITE 0,(X,Y),8,0
220 GOTO 100
230 PUT SPRITE 0,(0,209)
240 END
```

STOP

ストップ（ステートメント）

● 機　能　プログラムの実行を停止してコマンドレベルに戻る．

● 書　式　STOP

● 文　例　STOP

● 解　説　STOP 文はプログラムの実行を停止するためのもので，プログラムのどこで使用してもかまいません．STOP 文を実行すると，次のメッセージがプリントされてコマンドレベルに戻ります．

Break in yyyyy（yyyyy はストップした行番号）

END 文と異なり，STOP 文は使用していたファイルを閉じません．

停止したプログラムは CONT コマンドによりプログラムの実行を再開することができます．また，停止したプログラムを終了させたい場合は，CTRL ＋ STOP キーを押します．

● 参　照　CONT

STOP ON/OFF/STOP*

ストップ オン／オフ／ストップ（ステートメント）

● 機　能　CTRL + STOP キー割込みを許可，禁止，または保留する．

● 書　式　STOP ON
　　STOP OFF
　　STOP STOP

● 文　例　STOP ON

● 解　説　STOP ON 文は CTRL + STOP キー割込みを許可します．この命令を実行しておくと，プログラム実行時に CTRL + STOP キーが押されたときに割込みがかかり，実行中のプログラムを中断し，ON STOP GOSUB 文で定義しておいた割込み処理ルーチンが実行されます．

STOP OFF 文は CTRL + STOP キー割込みを禁止します．この命令を実行しておくと，プログラム実行中に CTRL + STOP キーが押されても割込みはかかりません．

STOP　STOP 文は CTRL + STOP キー割込みを保留します．この命令を実行しておくと，以後にかかった CTRL + STOP キー割込みは STOP ON 文を実行するまで保留され，STOP ON 文を実行すると直ちに割込み処理ルーチンが実行されます．この命令は STOP ON 状態で実行する場合のみ有効です．

● 参　照　ON STOP GOSUB，第1章「割込み」

STRIG

エストリガ(関数)

● **機　能**　ジョイスティックのトリガボタンが押されたかどうかを調べる.

● **書　式**　**STRIG (〈ジョイスティックの番号〉)**

● **文　例**　**STRIG (2)**

● **解　説**　指定したジョイスティックのボタンあるいはスペースキーが押されたかどうかを調べ，押されていれば－1，押されていなければ0の値が得られます.

〈ジョイスティックの番号〉は次のように指定します.

0　　：トリガボタンの代わりにキーボードのスペースキーを使う.

1または3：ポート1につながっているジョイスティック

2または4：ポート2につながっているジョイスティック

なお，トリガボタンが押されたときに，割込みがかかるように設定するON STRIG GOSUB命令もありますので，組合わせて使うと便利です.

● **参　照**　ON STRIG GOSUB, STRIG ON/OFF/STOP

● **サンプルプログラム**

```
10 PRINT"HIT SPACE KEY"
20 I=STRIG(0)
30 IF I THEN BEEP
40 GOTO 20
```

STRIG ON/OFF/STOP*

エストリガ オン／オフ／ストップ(ステートメント)

● 機　能　ジョイスティックのトリガボタンが押されたときに起る割込みを許可，禁止，または保留する．

● 書　式　**STRIG (〈ジョイスティックの番号〉) ON**
STRIG (〈ジョイスティックの番号〉) OFF
STRIG (〈ジョイスティックの番号〉) STOP

● 文　例　**STRIG (0) ON**

● 解　説　STRIG ON/OFF/STOP 文は〈数式〉で指定するジョイスティックのトリガボタンあるいはスペースキーが押されたときに起る割込みを許可，禁止，または保留します．

〈ジョイスティックの番号〉は次のように指定します．

0　　：トリガボタンの代わりにキーボードのスペースキーを使う．
1または3：ポート1につながっているジョイスティック
2または4：ポート2につながっているジョイスティック

STRIG ON 文はトリガボタン割込みを許可します．この命令を実行しておくと，プログラム実行中にジョイスティックのトリガボタンが押されたとき割込みがかかり，実行中のプログラムを中断して ON STRIG GOSUB 文で定義しておいた割込み処理ルーチンが実行されます．

STRIG OFF 文はトリガボタン割込みを禁止します．この命令を実行しておくと，プログラム実行中にトリガボタンが押されても割込みはかかりません．

STRIG STOP 文はトリガボタン割込みを保留します．この命令を実行しておくと，以後にかかったトリガボタン割込みは STRIG ON 文を実行するまで保留され，STRIG ON を実行すると直ちに割込み処理ルーチンが実行されます．この命令は STRIG ON 状態で実行する場合のみ有効です．

● 参　照　ON STRIG GOSUB，第1章「割込み」

STR$

ストリング ダラー(関数)

● 機　能　数値を表す文字列を得る.

● 書　式　**STR$ (〈数式〉)**

● 文　例　**A$＝STR$ (123)**

実行結果　A$＝″　123″

● 解　説　〈数式〉の値を10進表記の文字列に変換します. 〈数式〉の値は整数型または実数型(倍精度, 単精度)のいずれの数値型でも使うことができます.

〈数式〉が負の場合, 変換された文字列の1字目は″－″になります. 0または正の場合は, 1字目は空白(スペース)になります.

注意：STRING$ 関数と区別すること.

● 参　照　VAL, STRING$

● サンプルプログラム

```
10 INPUT"スウシ゛";A
20 A$=STR$(A)
30 FOR I=1 TO LEN(A$)
40 PRINT MID$(A$,I,1);" ";
50 NEXT
```

STRING$*

ストリング ダラー(関数)

● 機　能　指定する文字を指定する数だけ得る.

● 書　式　STRING$ (〈式〉, | 〈文字式〉 / 〈数式〉 |)

● 文　例　A$=STRING$ (10, "+")

● 解　説　〈式〉は文字数を指定します.

〈文字式〉または〈数式〉は文字を指定します.

〈文字式〉の場合は〈文字列〉の最初の一文字が有効です.

〈数式〉の場合は値が0から255の範囲に限られ，この値をキャラクタコードとみなし，対応する文字を採用します.

グラフィックキャラクタは使用できません.

● 参　照　STR$

● サンプルプログラム

```
10 CLS:CLEAR 500
20 INPUT "モジ(1ツ)=";A$
30 INPUT "カズ(セイスウ)=";A
40 PRINT STRING$(A, A$)
50 GOTO 20
```

SWAP

スワップ（ステートメント）

● **機　能**　2つの変数の値を交換する．

● **書　式**　**SWAP 〈変数〉, 〈変数〉**

● **文　例**　**SWAP A$, B$**

● **解　説**　交換する2つの変数の型が一致しているならば，どの型でもSWAP文でその値を交換することができます（整数，単精度，倍精度，文字，または配列変数）．2つの変数の型が一致していないと"Type mismatch"エラーが起こります．

● **サンプルプログラム**

```
10 FOR I=0 TO 5
20 X(I)=INT(RND(1)*99)
30 Y(I)=INT(RND(1)*99)
40 NEXT:GOSUB 110
50 PRINT
60 FOR I=0 TO 5
70 SWAP X(I),Y(I)
80 NEXT
90 GOSUB 110
100 END
110 PRINT " I   X(I)   Y(I)"
120 FOR I=0 TO 5
130 PRINT USING" #    ##    ##";I;X(I);Y(
I)
140 NEXT:RETURN
```

TAB*

タブ(関数)

● 機　能　指定された桁まで空白を出力する.

● 書　式　TAB (〈数式〉)

● 文　例　PRINT TAB(10); "BASIC"

● 解　説　画面の左端より〈数式〉で指定される位置まで空白を出力します. ただし, TAB実行時のカーソルまたはプリンタヘッド位置が, 〈数式〉で指定される位置より右側にあった場合は, TABは何もしません.
TABはPRINT文やLPRINT文など出力文中で使用します.

● 参　照　SPC

● サンプルプログラム

```
10 FOR K=1 TO 8
20 PRINT "*****";TAB(K);"0"
30 NEXT  K
```

TAN

タンジェント(関数)

● **機　能**　正接（タンジェント）を得る.

● **書　式**　**TAN (〈数式〉)**

● **文　例**　**A＝TAN (123)**

● **解　説**　〈数式〉の正接（タンジェント）を求めます.〈数式〉の単位はラジアンです.

〈数式〉は整数型，単精度実数型，倍精度実数型のいずれでもかまいませんが，結果は倍精度で得られます.

● **参　照**　SIN, COS, ATN

● **サンプルプログラム**

```
10 '*** TAN キョクセン ***
20 SCREEN 2
30 PI=3.141592653#
40 X=5:LINE(5,100)-(240,100),7
50 FOR I=-PI TO PI STEP PI/59
60 Y=100-TAN(I)*10
70 PRINT X
80 PSET(X,Y)
90 X=X+2
100 NEXT
110 GOTO 110
```

TIME

タイム(システム変数)

● 機　能　インターバルタイマ.

● 書　式　TIME

TIME＝0

● 文　例　PRINT TIME

● 解　説　TIMEは1／60秒ごとに値が1づつカウントアップされるシステム変数です．TIMEが持つ値は0から65535までで，この範囲の値をTIMEに代入することもできます.

プログラムの実行時間を計りたいとき，計りたい処理の一番始めにTIMEへ0を代入し，処理の最後でTIMEの内容を調べれば1／60秒単位の時間を得ることができます.

このタイマはVDPジェネレータが1／60秒ごとに割込みをかけるのをきっかけに値を更新しているため，カセットテープを操作している間など，割込みをかけることができない間はタイマの値は更新されません.

● 参　照　ON INTERVAL GOSUB

● サンプルプログラム

```
10 '*** タイマ ***
20 CLS
30 LOCATE 10,8
40 PRINT"START !"
50 TIME=0
60 LOCATE 10,10
70 T=INT(TIME/60)
80 H=INT(T/3600)
90 M=INT(T/60) : M=M MOD 60
100 S=T MOD 60
110 PRINT USING"## : ## ' ##";H;M;S
120 GOTO 60
```

TRON/TROFF

トレース オン／トレース オフ(コマンド)

● 機　能　プログラムの実行状態を追跡する.

● 書　式　**TRON**

TROFF

● 文　例　**TRON**

TROFF

● 解　説　TRON 文を実行すると,以後実行したプログラムの行番号が画面に表示されます.それぞれの行番号は角カッコに囲まれて表示されます.TROFF 文または NEW コマンドを実行するとリセットされます.

TRON 文はプログラムのデバッグを助けるために使います.

TRON 文の実行はダイレクトモードでもプログラムモードでも可能です.

USR*

ユーザー(関数)

● 機　能　機械語で作られたルーチンを呼び出す.

● 書　式　USR 〔〈番号〉〕(〈引数〉)

● 文　例　I=USR 3(J)

● 解　説　あらかじめメモリ中に準備しておいた機械語ルーチンを呼び出します．機械語ルーチンの実行開始番地はDEF USR文で前もって設定されていなければなりません.

〈番号〉は0から9までの値で，DEF USR文により定義された番号に対応しています.〈番号〉が省略された場合には0と解釈されます.

〈引数〉はBASICから機械語ルーチンへの値の受け渡しをします．なお，引数は省略することができませんので，呼び出すルーチンに引数を渡す必要がない場合でもダミーの引数を指定します.

USR関数を実行すると，レジスタAに引数の型がセットされ，またHLには引数のアドレス(文字列の場合にはDEにストリングディスクリプタのアドレス)がセットされ，機械語ルーチンからの参照ができるようになります。引数の型によるAレジスタ，HLレジスタおよびDEレジスタにセットされる内容を次頁に示しています.

機械語ルーチンでの処理の結果を引渡す場合は，引数の内容を書き換えます．このとき，呼び出すときに指定した〈引数〉と異る型に変更することはできません．また，文字列の場合は文字列長を変更することはできません.

機械語ルーチンからの戻りはRET命令を使います(スタックには戻り番地がセットされています).

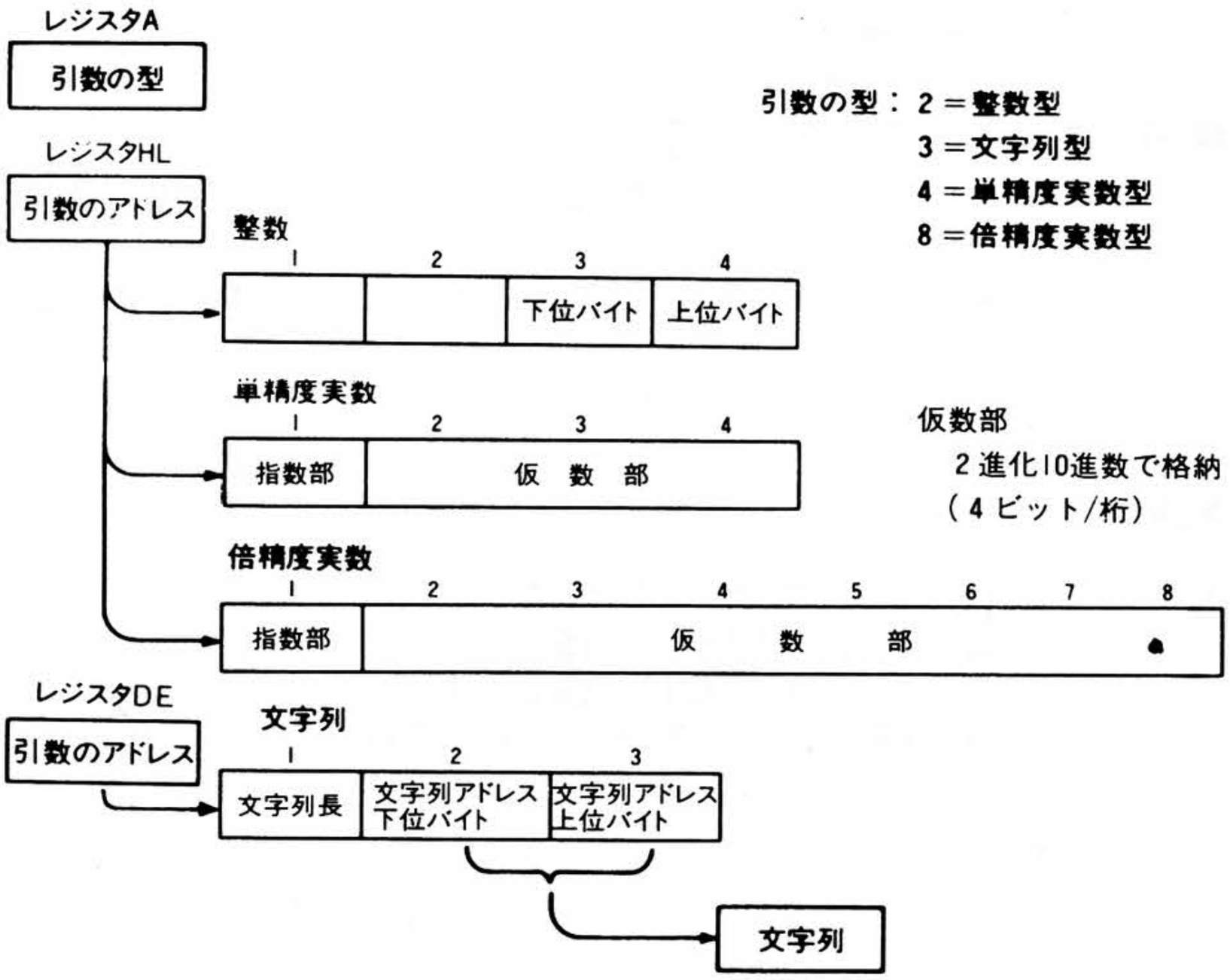
引数は，メモリの特定のアドレスを利用しても行なうこともできます.

①引数のための絶対アドレスADを決める．このアドレスADは，CLEAR文によるメモリ上限とシステム用エリアとの間に設ける事(メモリマップ参照)

②POKE文により機械語ルーチンに渡す引数をアドレスADに書き込む.

USR*

③機械語ルーチンでは，アドレスADを参照して処理する．
④処理結果はアドレスAD'(ADと異なってもよい)に書き込む．
⑤BASICでは，PEEK文でAD'の内容を参照して処理する．



● 参　照　DEF USR

● サンプルプログラム

```
10 CLEAR 200,&HD000
20 AD=&HD000
30 FOR I=AD TO AD+9
40 READ A$:A=VAL("&H"+A$)
50 POKE I,A
60 NEXT I
70 DEF USR 1=&HD000
80 PRINT "-32768~32766 ノ カズヲ"
90 INPUT "ニュウリョク シテクダサイ";A%
100 PRINT "A%=";A%
110 PRINT "キカイゴルーチン ヲ ヨビダシマス"
120 I=USR 1(A%)
130 PRINT "I=";I
140 DATA 23,23,4E,23,46,03,70,2B,71,C9
```

VAL

バリュー(関数)

● 機　能　文字列の表す数値を得る.

● 書　式　VAL (〈文字式〉)

● 文　例　A＝VAL (〝2〞)

実行結果　A＝2

● 解　説　〈文字式〉で表す文字例を数値に変換してその結果を得ます. もし, 文字例の最初の文字が＋, －,&, または数字でなければ, VAL の値は 0 になります.

また途中に数値以外の文字があると, それ以降の文字列は無視します.

ただし, 文字列中のスペースは無視されます.

● 参　照　STR$

● サンプルプログラム

```
10 INPUT " &H";A$
20 A=VAL("&H"+A$)
30 PRINT "&H";A$;" ﾊ "
40 PRINT "ｼﾞｭｯｼﾝ ﾃﾞﾊ";A
```

VARPTR*

バーポインター(関数)

● 機　能　変数の格納されているメモリ番地，ファイルに割り当てられているファイルコントロールブロックの開始番地を求めます．

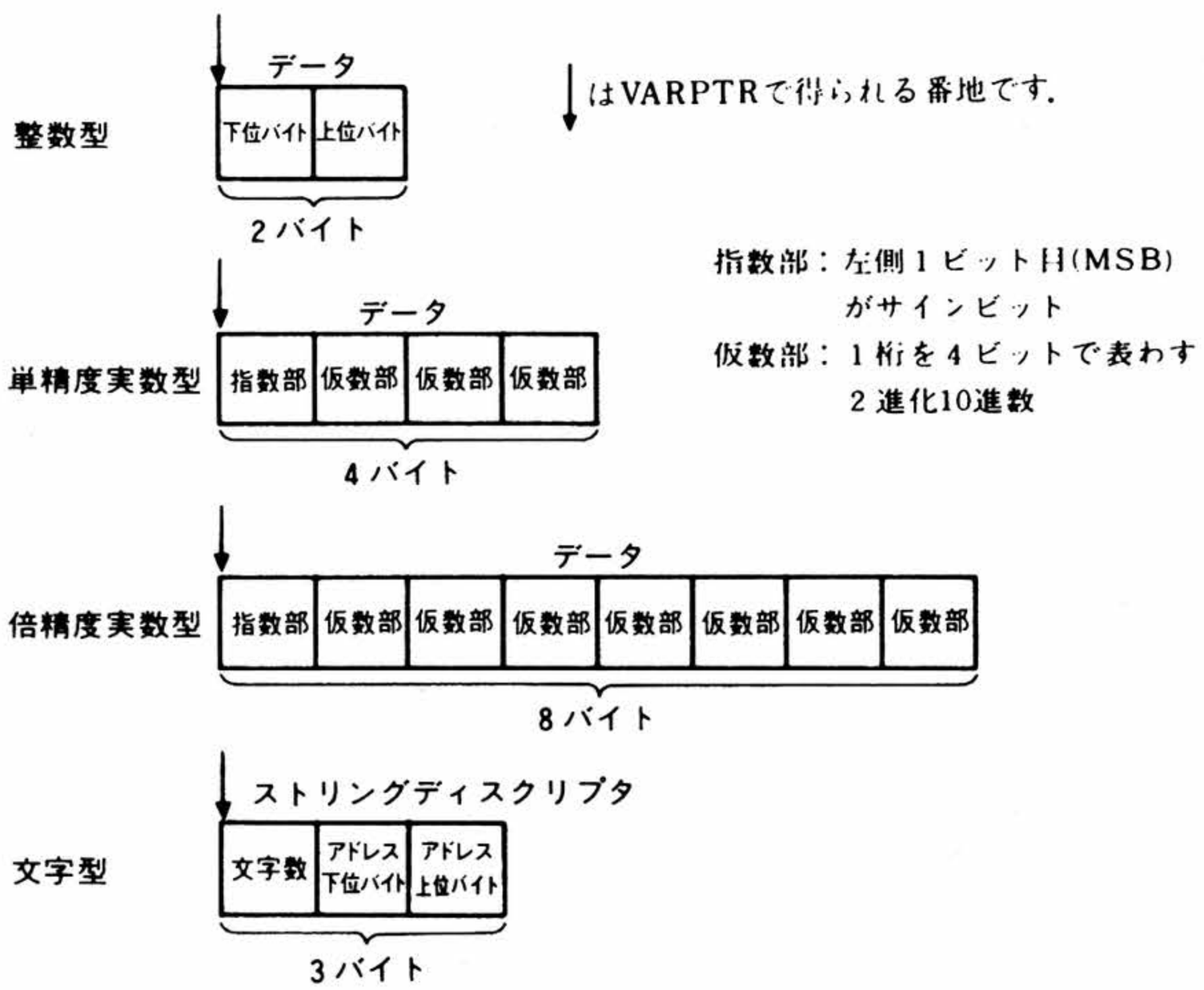
● 書　式　1) VARPTR（〈変数名〉）

2) VARPTR　（#〈ファイル番号〉）

● 文　例　1) PRINT HEX$(VARPTR(A))

2) BVAD＝VARPTR(#1)

● 解　説　1）〈変数名〉で指定した変数のデータが格納されている変数領域のメモリ番地を求めます．このとき指定される変数には値が代入されていなければなりません．



2）指定した〈ファイル番号〉に割当てられているファイルコントロールブロックの開始番地を得ます．

VDP*

ブイ ディー ピー(システム変数)

● 機　能　VDP(ビデオディスプレイプロセッサ)ジェネレータのレジスタへデータを直接書き込んだり参照したりする.

● 書　式　VDP (〈レジスタ番号〉)

● 文　例　A＝VDP(8)

● 解　説　VDP ジェネレータの機能は BASIC で用意しているステートメントや関数でほとんど使うことができるようになっていますが, このシステム変数は, 自分で VDP ジェネレータのレジスタを参照更新するときに用います.
一般には使用することはありませんので, 詳細は省略します.

VPEEK*

ブイ ピーク(関数)

● **機　能**　**VRAM 上の指定した番地の内容を読み出す.**

● **書　式**　**VPEEK (〈番地〉)**

● **文　例**　**A%＝VPEEK(1)**

VRAMのアクティブページの1番地の内容を変数A%に代入する.

● **解　説**　VRAM上にある画面データを直接参照します.

〈番地〉は, SET PAGE文により設定されているアクティブページの先頭からの値(オフセット値)を, &HO～&HFFFFの値を持つ数式で指定します.

● **参　照**　VPOKE, BASE, SET PAGE, 付録「VRAMについて」

VPOKE*

ブイ ポーク(ステートメント)

● **機　能**　**VRAM 上の指定した番地にデータを書込む.**

● **書　式**　**VPOKE 〈番地〉, 〈データ〉**

● **文　例**　**VPOKE &H8000, &HFF**

VRAMのアクティブページの&H8000番地に&HFFを書き込む.

● **解　説**　VRAM上にある画面データを直接書き換えます.

〈番地〉はSET PAGE命令により設定されているアクティブページの先頭からの値(オフセット値)を, &HO～&HFFFFの値を持つ数式で指定します.

〈データ〉は書き込むデータ1バイト分（0～255）の値を持つ数式です.

● **参　照**　VPEEK, BASE, SET PAGE, 付録「VRAMについて」

WAIT*

ウエイト(ステートメント)

● 機　能　コンピュータの入力ポートをモニタする間，プログラムの実行を中断する．

● 書　式　WAIT 〈ポート番号〉，〈式 1〉〔，〈式 2〉〕

● 文　例　WAIT　1, &H22, &H22

● 解　説　WAIT 文を実行すると，指定した入力ポートのビットパターンが指定した状態になるまでプログラムの実行が中断されます．

〈ポート番号〉で指定したポートから読み込んだデータと〈式 2〉との XOR の結果を得，さらに〈式 1〉との AND が取られます．もしその結果が 0 (偽) ならBASICはもう一度ポートの状態を読み込み，同じ操作を繰り返します．もし結果が 0 でなければ(真)，プログラムの実行は次の文にうつります．〈式 2〉が省略された場合は 0 とみなします．

注意：WAIT 文の実行により無限ループにはいってしまう場合があります．この場合にはシステムをリセットしなければなりません．

WIDTH

ウィドス(ステートメント)

● 機　能　画面に表示する桁数を指定する．(テキストモード専用)

● 書　式　WIDTH 〈桁数〉

● 書　式　WIDTH 29

● 解　説　画面に表示する 1 行の桁数を設定します．

〈桁数〉の範囲は，32×24のモード(SCREEN 1)では 1 ～32(初期値は29に設定)，80×24のモード(SCREEN 0)では 1 ～80(初期値は39に設定)です．なお，80×24のモードで桁数41～80を指定すると，幅の小さい文字で表示されます．

● 参　照　SCREEN

第３章

サンプルプログラム

3

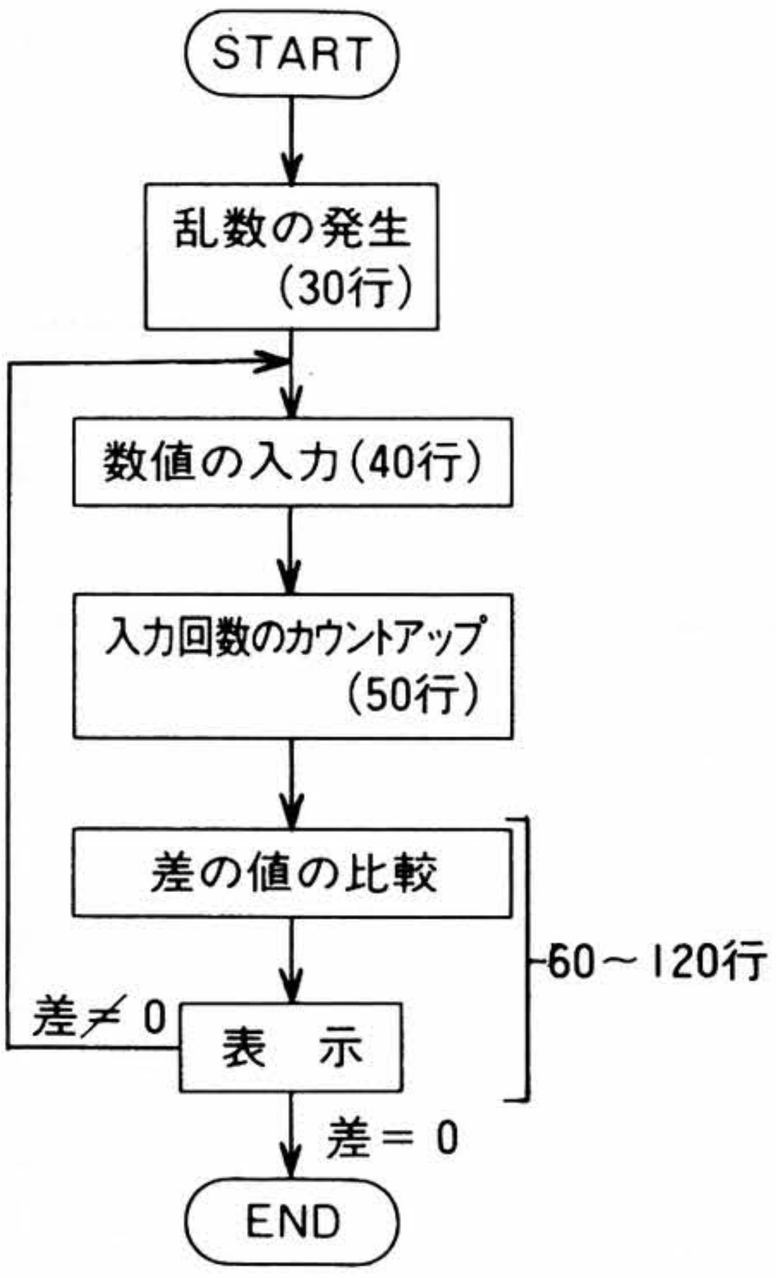
数あてゲーム

IF文，RND関数

RND関数で発生した乱数0～99の値を何回で当てるかというゲームです．入力された数値と正解数値との差により次のような表示をします．

差が20以上　　「ぜんぜんダメ」
差が10～19　　「ダメ！」
差が1～9　　「おしい！」
差が0　　「おめでとう！○かいてアタリ!!」

フローチャート




```
10 CLS
20 LOCATE 5,3:PRINT "おす゛あ
て ケ゛ーム"
30 A=INT(RND(-TIME)*100)
40 LOCATE 2,5:INPUT"よそう す
うち(0-99)を にゅうりょく";K
50 N=N+1:C=ABS(A-K)
60 IF C=0 THEN 120
70 IF C<10 THEN M$="おしい ！
     ":GOTO 100
80 IF C<20 THEN M$="タ゛メ ！
     ":GOTO 100
90 M$="せ゛んせ゛ん タ゛メ"
100 LOCATE 5,9:PRINT M$
110 GOTO 40
120 LOCATE 2,9:PRINT "おめて
゛とう ！";N;"おいて゛ アタリ ！！"
```

次は右図の枠において，コンピュータがRND関数により指定したポイントを何回で当てるかというゲームです．入力された数値のポイントと同じ行または列にコンピュータが指定したポイントがあると「おしい！」と表示され，指定したポイントが含まれていなければ「かんけいなし！」と表示します．

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

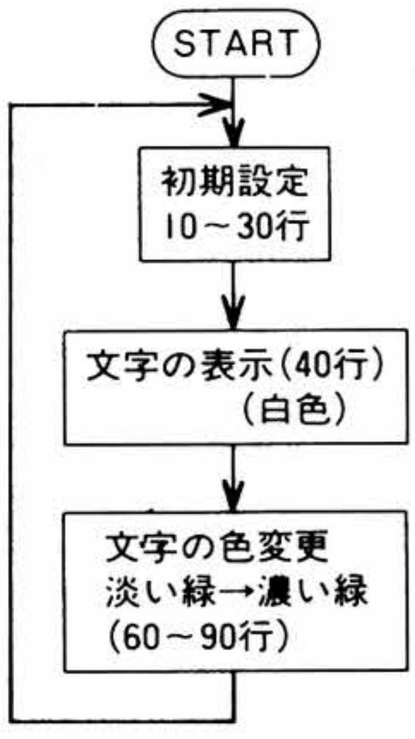
```
10 '*** おず あて ゲーム 2 ***
20 CLS:R=INT(RND(-TIME)*25)
30 X=R MOD 5:Y=R ¥ 5
40 INPUT"よそう すうち(0-24)を にゅうりょく";K
50 IF K<0 OR K>24 THEN 40
60 N=N+1:X1=K MOD 5:Y1=K ¥ 5
70 IF X=X1 AND Y=Y1 THEN 110
80 IF X=X1 OR Y=Y1 THEN 100
90 PRINT "かんけいなし !":GOTO 40
100 PRINT "おしい !":GOTO 40
110 PRINT "おめでとう";N;"かいで アタリ !!"
```

グラフィック画面への文字の表示

OPEN文，PRINT＃文

グラフィック画面に文字を表示させ，この文字の色を変えるプログラムです．

フローチャート




```
10 SCREEN 3:COLOR 15,1,1:CLS
20 OPEN"GRP:"FOR OUTPUT AS#1
30 PRESET(4,80)
40 PRINT #1,"NATIONAL"
50 CLOSE#1
60 FOR C=0 TO 7
70 FOR T=0 TO 500:NEXT T
80 COLOR=(15,0,C,0)
90 NEXT C
100 GOTO 10
```

グラフィック画面への文字の表示は，20行でファイルを開き，40行でPRINT＃命令により表示文字を転送します．50行でファイルを閉じます．閉じるのを忘れた場合は，次に20行を実行する時〝File already open〟エラーとなります．SCREEN 3以外のグラフィック画面でも，文字が小さくなるだけで表示の手続きとしては同じです．SCREEN 3以外でも実行してみてください．

70行は，色の変化の速さを遅くするためのソフトタイマーです．80行の（）の部分を(15，C，0，0)すると赤色に，(15，0，0，C)とすると青色になります．

このプログラムでは，〝NATIONAL〟という文字が画面の左右一ぱいに表示されていますが，文字と文字との間はもっと詰めることができます，文字間を詰めて，左右にも余裕を持って表示できるようにしてみましょう．

```
10 SCREEN 3
20 OPEN"GRP:"FOR OUTPUT AS#1
30 N$="NATIONAL"
40 FOR C=1 TO 8
50 P$=MID$(N$,C,1)
60 PRESET(12+24*C,80)
70 PRINT #1,P$
80 NEXT C
90 CLOSE #1
100 GOTO 100
```

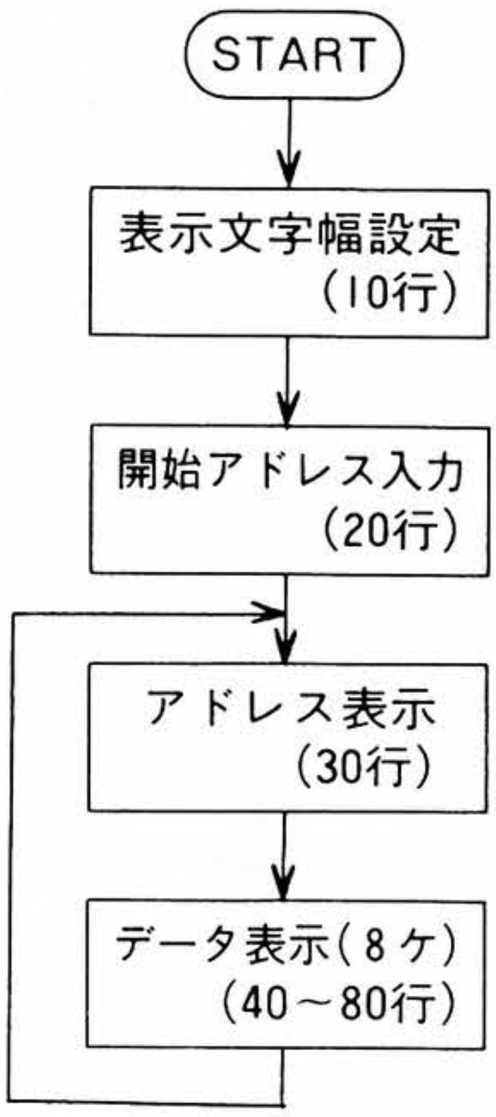
50行でN$(NATIONAL)のC番目の1文字を取出し，60行で表示位置をセットし，70行で表示しています．

メモリ内容の表示

INPUT文，PEEK文

メモリの内容を読み取り画面に16進数値で表示させるプログラムです．

フローチャート




```
10 WIDTH 30
20 INPUT"ADDRESS=";A$:AD=
VAL("&H"+A$)
30 PRINT HEX$(AD);"  ";
40 FOR A=1 TO 8
50 DT=PEEK(AD):IF DT<16 T
HEN PRINT " ";
60 PRINT HEX$(DT);"  ";
70 AD=AD+1
80 NEXT A
90 PRINT
100 GOTO 30
```

RUNすると〝ADDRESS＝〟とアドレスを聞いて来ますので，16進数4桁でアドレスを入力します．20行で入力された文字列を16進数値に変換しますので，入力アドレスには〝&H〟を付ける必要はありません．

アドレスを入力すると表示を開始します．画面を止めて見たい場合は[STOP]キーを押します．30行の命令によって各行の左端に表示される数値は，続いて表示されているデータ8バイトの先頭アドレスを示しています．

50行は，データの内容が1桁(&H10未満)の場合の表示位置の補正を行ないます，実行の中止は[CTRL]+[STOP]です．

上記プログラムで，16行分のデータを表示すると表示が停止し，何かのキーを押すと，再び続きのデータを表示するよう，プログラムを変更してみましょう．挑戦してみてください．

PEEK文のサンプルプログラム部にも同様のプログラムが紹介されています．参考にしてください．

指定した色の表示

COLOR文，PUT SPRITE文

MSX２で使用できる色を512色のうち８ヶずつ表示します．RUNすると色を聞いて来ますので，０～７まですべてのレベルを表示する色(RまたはGまたはB)と他の２色のレベル(０～７)を入力します．例えば赤のレベルは５，青のレベルは４で緑色のレベルを変えて８色表示する場合には，G，５，４↵と入力すると指定した色が表示されます．固定レベル色のレベルを入力する順序はR，G，Bの順です．色はスプライトを使用して表示しています．このプログラムは CTRL ＋ STOP キーを押してプログラムの実行を中止すると，スプライトパターンの表示が画面に残してしまいますので必ず〝Y/N〟を聞きに来た状態で〝N〟を入力して終わるようにしてください．

CTRL ＋ STOP を押してしまった場合は，〝SCREEN１，０〟を実行してください．

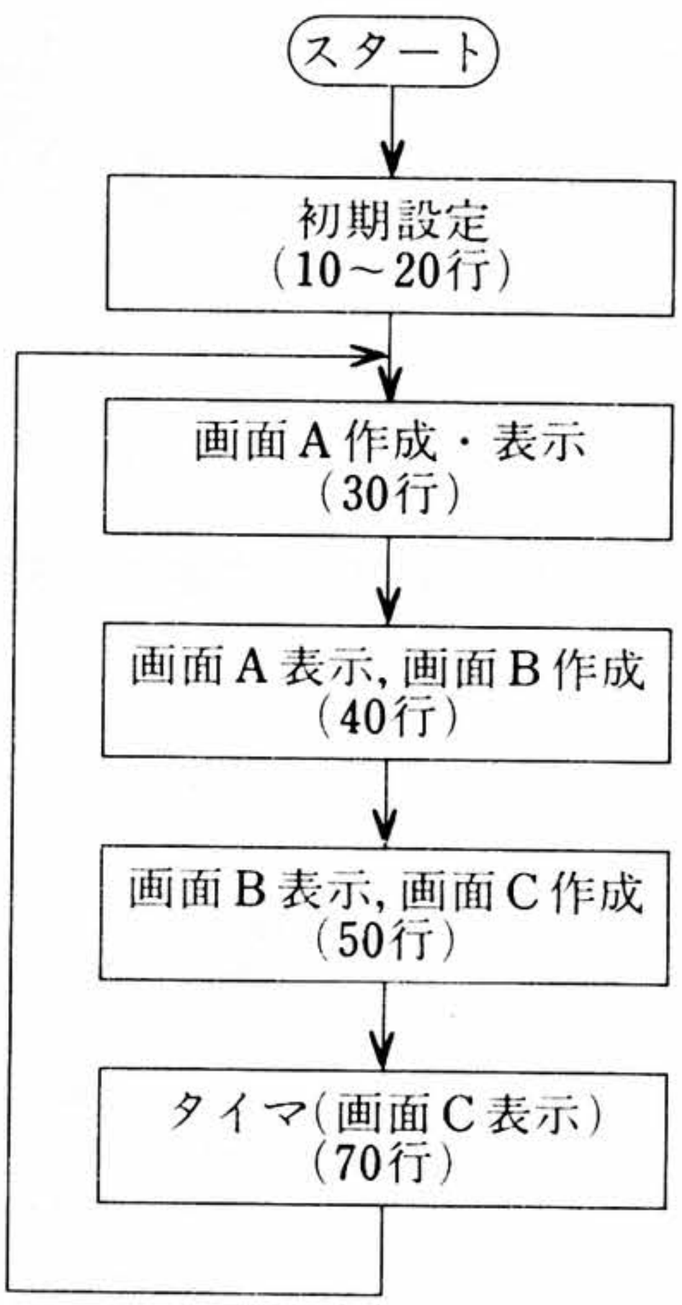
```
10 SCREEN 1,3:COLOR 15,13,13:CLS:KEY OFF
20 COLOR=(13,0,2,7)
30 SPRITE$(0)=STRING$(32,CHR$(255))
40 '*** いろの してい ***
50 LOCATE 4,2:PRINT "していした いろを ひょうじ゛します"
60 LOCATE 4,4:INPUT "RGB,0-7,0-7";C$,C1,C2
70 IF C1>7 OR C1<0 OR C2>7 OR C2<0 THEN 60
80 IF C$="R" OR C$="r" THEN R$="0-7":G$=STR$(
C1):B$=STR$(C2):GOTO 190
90 IF C$="G" OR C$="g" THEN R$=STR$(C1):G$="0
-7":B$=STR$(C2):GOTO 160
100 IF C$<>"B" AND C$<>"b" THEN 60
110 R$=STR$(C1):G$=STR$(C2):B$="0-7"
120 '*** いろの せって い ***
130 FOR CL=0 TO 7
140 COLOR=(CL+1,C1,C2,CL)
150 NEXT CL:GOTO 230
160 FOR CL=0 TO 7
170 COLOR=(CL+1,C1,CL,C2)
180 NEXT CL:GOTO 230
190 FOR CL=0 TO 7
200 COLOR=(CL+1,CL,C1,C2)
210 NEXT CL
220 '*** いろの ひょうじ゛ ***
230 FOR SP=1 TO 8
240 PUT SPRITE SP,(8+SP*24,48+SP*8),SP,0
250 NEXT SP
260 LOCATE 2,6:PRINT C$;"=0   1   2   3   4   5   6
  7"
270 LOCATE 2,17:PRINT"R=";R$;"   ";"G=";G$;"
";"B=";B$
280 LOCATE 4,21:PRINT "つき゛の いろを ひょうじ゛しますか ?"
290 LOCATE 4,22:INPUT"Y/N";Y$
300 IF Y$="Y" OR Y$="y" THEN 50
310 SCREEN 1,1:CLS:KEY ON
```

画面の切換

PAGE文，RND関数

ランダム関数を使用して，画面をジェネレートするプログラムです．表示の方法には，現在表示されている画面(ページ)にジェネレートする方法と，表示されている画面とは別のページにジェネレートし，画面が出来上った後に表示を切換える方法とがあります．後者の方法は画面の表示が瞬時に切り換わります．

フローチャート




```
10 SCREEN 7
20 A=RND(-TIME)
30 COLOR ,13:GOSUB 100
40 SET PAGE 0,1:COLOR ,5:GOSUB 100
50 SET PAGE 1,0:COLOR ,1:GOSUB 100
60 SET PAGE 0,0
70 FOR T1=0 TO 200:NEXT T1
80 GOTO 30
100'*** ｶﾞめん さくせい ｻﾌﾞﾙｰﾁﾝ ***
110 CLS
120 FOR N=0 TO 25
130 X=RND(1)*512
140 Y=RND(1)*212
150 R=RND(1)*50
160 C=RND(1)*16
170 CIRCLE(X,Y),R,C
180 NEXT N
190 RETURN
```

画面Aの背景色：紫

画面Bの背景色：青

画面Cの背景色：黒

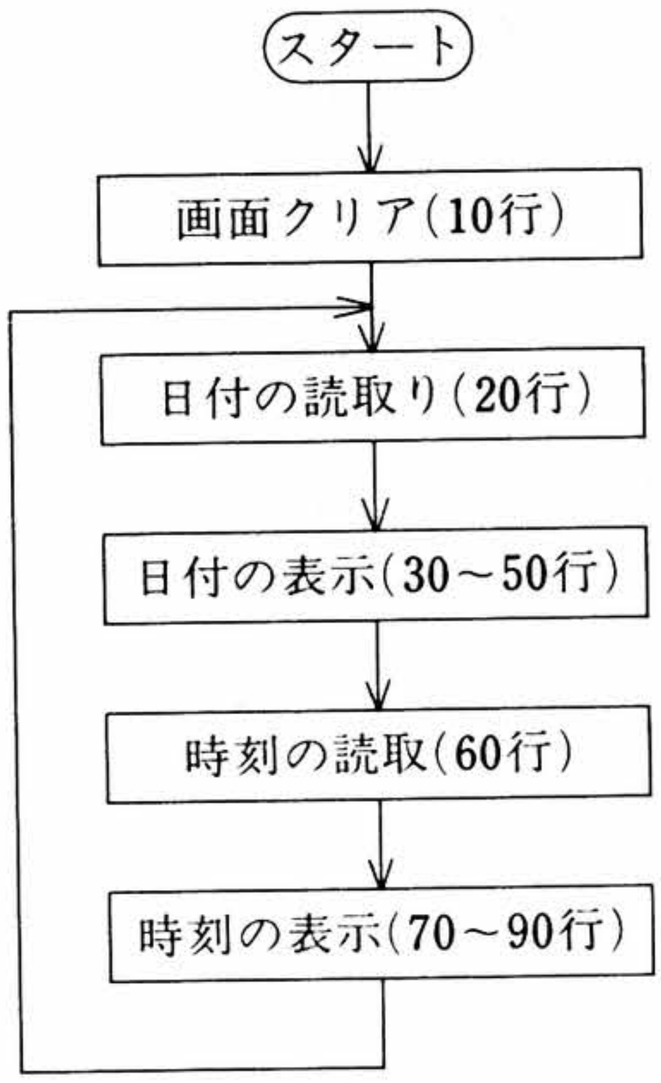
プログラムを実行して画面の切換わる様子と，SET PAGE命令との関係を対比してみてください．

100～190行はサブルーチンです．この例のように同じ処理をプログラムの各所で実行する必要がある場合には，サブルーチン形式にした方が，プログラムが簡単になります．

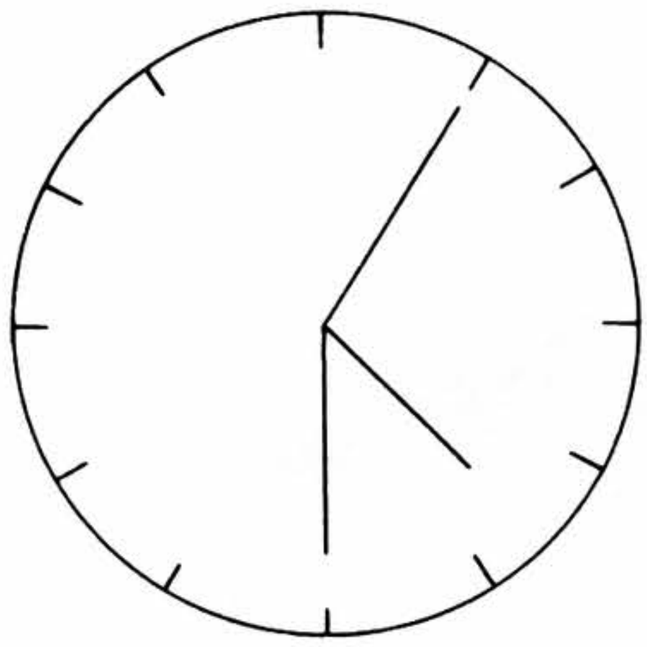
時計を作ろう！

GET文，MID関数，LINE文

PUT文でも簡単な時刻表示プログラムを紹介していますが，日付と時刻に年，月，日，時，分，秒を付けて表示してみましょう．




```
10 CLS
20 GET DATE DT$
30 LOCATE 5,5
40 Y$=MID$(DT$,1,2):M$=MI
D$(DT$,4,2):D$=MID$(DT$,7
,2)
50 PRINT Y$;"年";M$;"月";D$
;"日"
60 GET TIME TM$
70 LOCATE 5,7
80 H$=MID$(TM$,1,2):MN$=M
ID$(TM$,4,2):S$=MID$(TM$,
7,2)
90 PRINT H$;"時";MN$;"分";S
$;"秒"
100 GOTO 20
```



次は時刻を読取り図のような時計を作ってみましょう．まずは目盛板の作成です．図のように中心が(120,100)で半径60の円に目盛を打ちます．

```
10 COLOR ,4:SCREEN 5
20 CIRCLE(120,100),60,15
30 LINE(120,40)-(120,160),15
40 LINE(60,100)-(180,100),15
50 S3=60*SIN(3.14/6):C3=60*COS(3.14/6)
60 LINE(120+S3,100-C3)-(120-S3,100+C3),15
70 LINE(120+C3,100-S3)-(120-C3,100+S3),15
80 LINE(120+C3,100+S3)-(120-C3,100-S3),15
90 LINE(120+S3,100+C3)-(120-S3,100-C3),15
100 CIRCLE(120,100),55,7
110 PAINT(120,100),7
120 CIRCLE(120,100),55,4
130 PAINT(120,100),4
140 GOTO 140
```

うまく目盛板が描けましたか？

次は秒針の表示です．140行を消して，引き続き次のプログラムを入力してください．

```
140 S1=120:S2=100:SS=0
150 GET TIME TM$
160 S=VAL(MID$(TM$,7,2)):IF S=SS THEN 150
170 SX=120.2+50*SIN(S*3.14/30)
180 SY=100.2-50*COS(S*3.14/30)
190 LINE(120,100)-(S1,S2),4
200 LINE(120,100)-(SX,SY),15
210 S1=SX:S2=SY:SS=S
220 GOTO 150
```

170行と180行は秒針の先端の座標を求めています．190行は旧い秒針を消し，200行で新らしい秒針を描いています．

時針，分針も付け加えた時計を完成させましょう．140行以降から入力しなおします．
前のプログラムの秒針表示と同様の作業を分針・時針について追加しています．頭文字Hの変数が時針，Mが分針，Sが秒針に関連した変数です．

```
140 H1=HX:H2=HY:HH=H
150 M1=MX:M2=MY:MM=M
160 S1=SX:S2=SY:SS=S
170 GET TIME TM$
180 S=VAL(MID$(TM$,7,2)):IF S=SS THEN 170
190 M=VAL(MID$(TM$,4,2)):H=(VAL(MID$(TM$,1
,2))MOD 12+M/60)*5
200 SX=120.2+50*SIN(S*3.14/30):SY=100.2-50
*COS(S*3.14/30)
210 MX=120.2+40*SIN(M*3.14/30):MY=100.2-40
*COS(M*3.14/30)
220 HX=120.2+30*SIN(H*3.14/30):HY=100.2-30
*COS(H*3.14/30)
230 IF H1=0 THEN 270
240 IF M=MM THEN 260
250 LINE(120,100)-(M1,M2),4:LINE(120,100)-
(H1,H2),4
260 LINE(120,100)-(S1,S2),4
270 LINE(120,100)-(SX,SY),8
280 LINE(120,100)-(MX,MY),15
290 LINE(120,100)-(HX,HY),15
300 GOTO 140
```

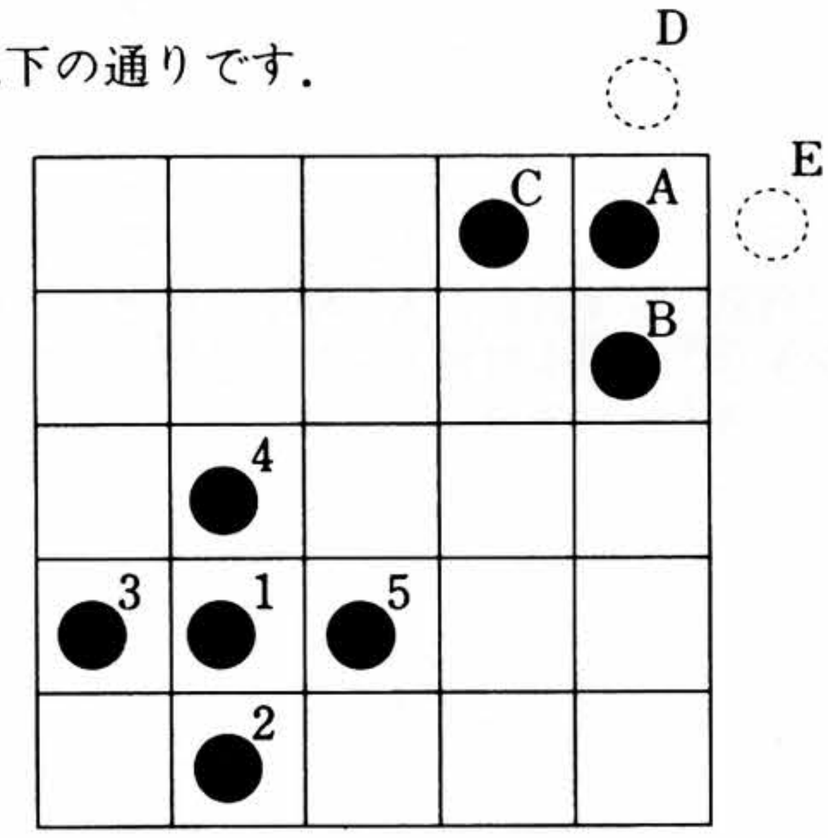
もう少し飾りをつけてみましょう．以下の行を追加してみてください．

```
24 CIRCLE(120,100),65,14
26 LINE(20,25)-(220,175),14,B
28 PAINT(30,30),14
295 BEEP
```

簡単なゲーム

STICK文，STRIG文，PUT SPRITE文

簡単なゲームを作成してみましょう．ルールは以下の通りです．

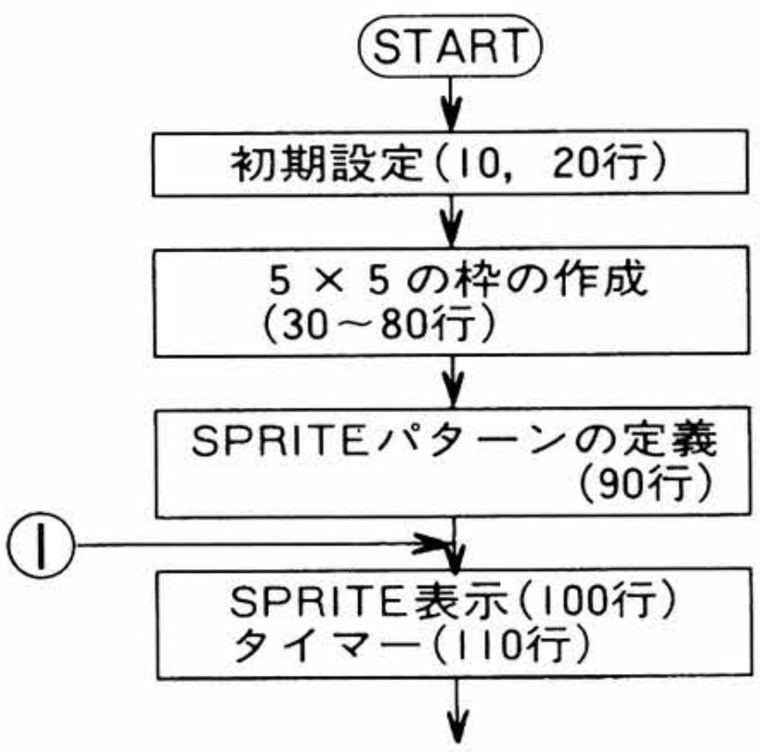


- 横５×縦５ケの枠を●で埋めます．
- 枠の１つを指定すると，その枠と上下左右の枠が反転し，◉となります．（右図１の指定に対して，2，3，4，5）．ただし，反転すべき枠がすでに◉であった場合には□（当初の状態）になります．
- 反転すべき枠が，５×５の配列よりはみ出す場合には，はみ出した部分は無視されます．（図Aの指定に対してD，E）
- 上記操作を何度か繰り返して５×５の枠全てが◉になればゲーム終了です．

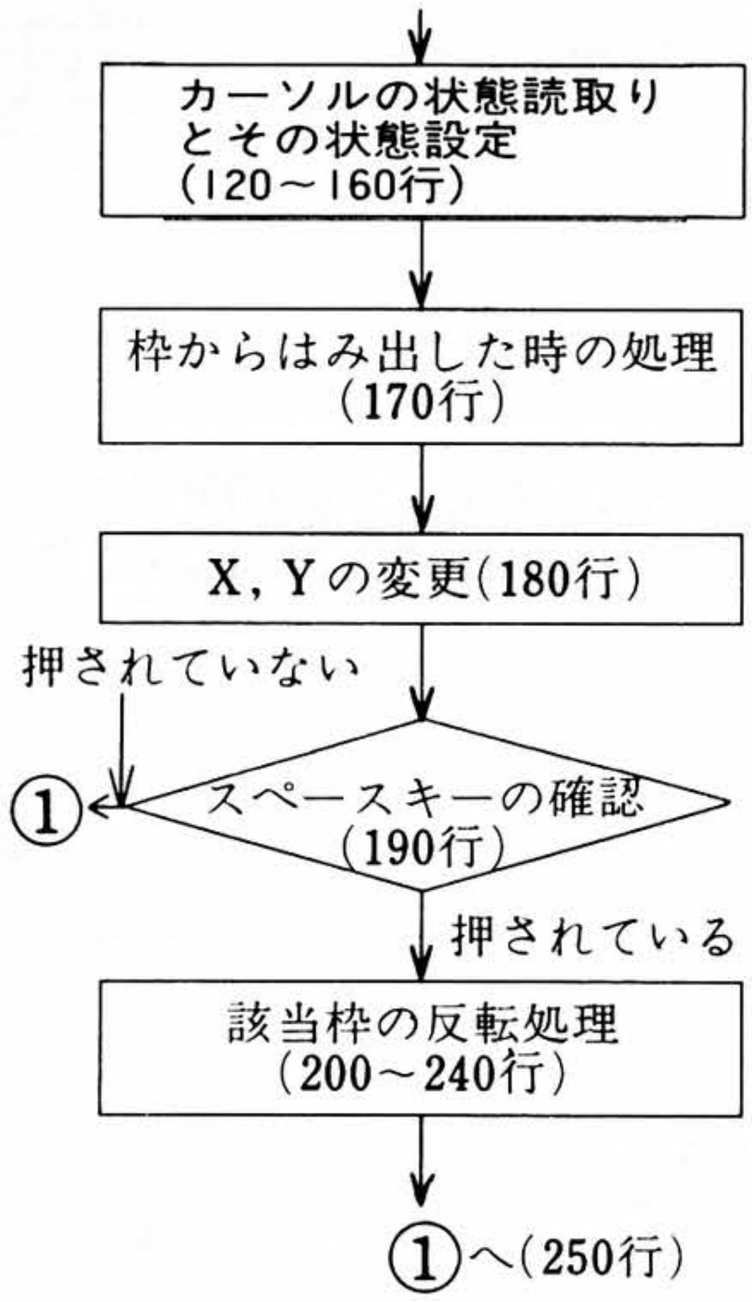
プログラム作成のための仮定および操作方法

- ５×５の各枠には，配列変数〔W（AD）〕の各要素を割り当てましょう．（ADは０～24）該当の枠が□の所はW(AD)＝０，◉になるとW(AD)＝１とします。
- 枠の指定は，カーソルキーによりポインタを移動させ，スペースキーにより指定します．ポインタは，スプライトパターン面０に□を表示させます．

フローチャート

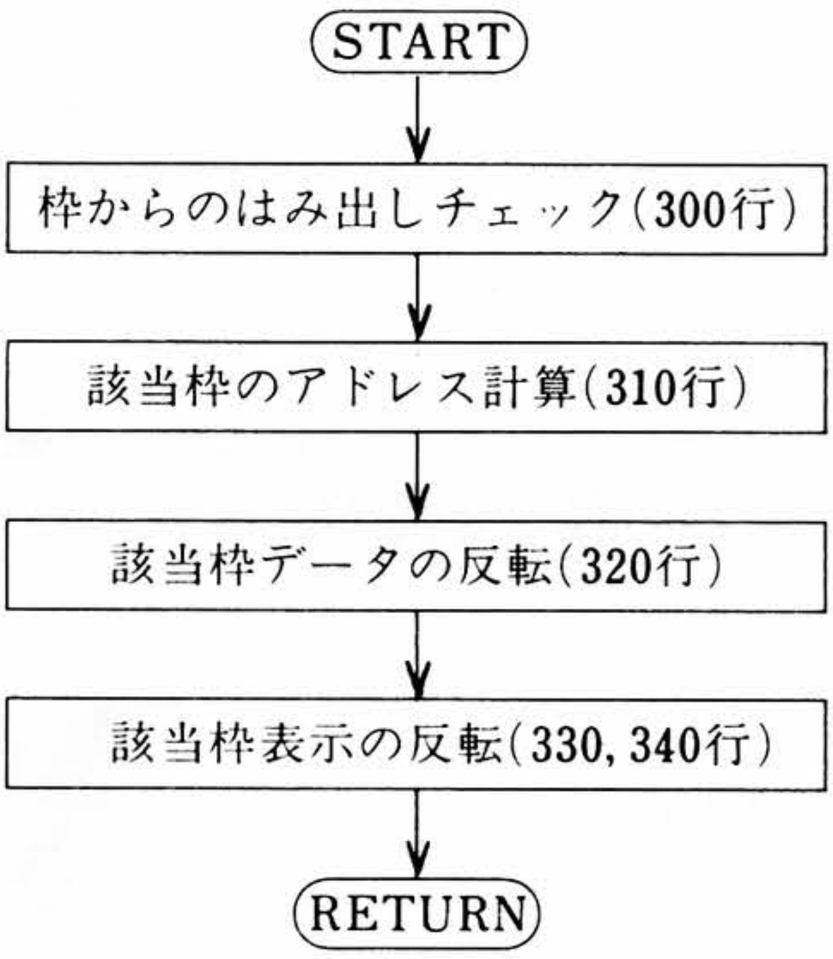



```
10 DIM W(24)
20 SCREEN 1,1
30 LOCATE 5,5      :PRINT"┌
┬┬┬┬┬"
40 FOR L=0 TO 4
50 LOCATE 5,6+2*L:PRINT"|
 |||||"
60 LOCATE 5,7+2*L:PRINT"├
┼┼┼┼┼"
70 NEXT L
80 LOCATE 5,15     :PRINT"└
┴┴┴┴┴"
90 SPRITE$(0)=CHR$(0)+CHR
$(126)+CHR$(66)+CHR$(66)+
CHR$(66)+CHR$(66)+CHR$(12
6)+CHR$(0)
100 PUT SPRITE 0,(59+X*16
,43+Y*16),3,0
110 FOR B=0 TO 30 :NEXT B
```




```
120 DIR=STICK(0):X1=X:Y1=
Y
130 IF DIR=1 THEN Y1=Y1-1
140 IF DIR=5 THEN Y1=Y1+1
150 IF DIR=3 THEN X1=X1+1
160 IF DIR=7 THEN X1=X1-1
170 IF X1<0 OR Y1<0 OR X1
>4 OR Y1>4 THEN 120
180 X=X1:Y=Y1
190 IF STRIG(0)=0 THEN 10
0
200 GOSUB 300
210 Y1=Y1+1:GOSUB 300
220 Y1=Y1-2:GOSUB 300
230 Y1=Y1+1:X1=X1+1:GOSUB
 300
240 X1=X1-2:GOSUB 300
250 GOTO 100
290 REM **** SUB ****
300 IF X1<0 OR Y1<0 OR X1
>4 ORY1>4 THEN RETURN
310 AD=X1+Y1*5
320 IF W(AD)=0 THEN W(AD)
=1 ELSE W(AD)=0
330 LOCATE 6+X1*2,6+Y1*2
340 IF W(AD)=0 THEN PRINT
" " ELSE PRINTCHR$(133)
350 RETURN
```

サブルーチン



ゲームの完成は，あなたがすべての枠が◉になった事を確認して終了とするわけですが，これをプログラムにより検出し終了するように改善してみてください．この時PLAY文により完成を祝福するメロディを流すのも面白いでしょう．

CTRL + STOP でゲームを終了した時、スプライトで表示している□が残ります．これを消すためには次の命令を実行してください．

```
PUT SPRITE  0,(0,208)
```

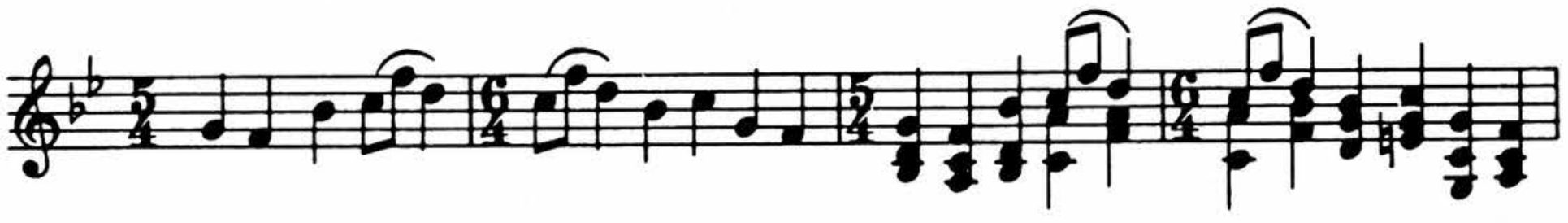
「展覧会の絵」を演奏する

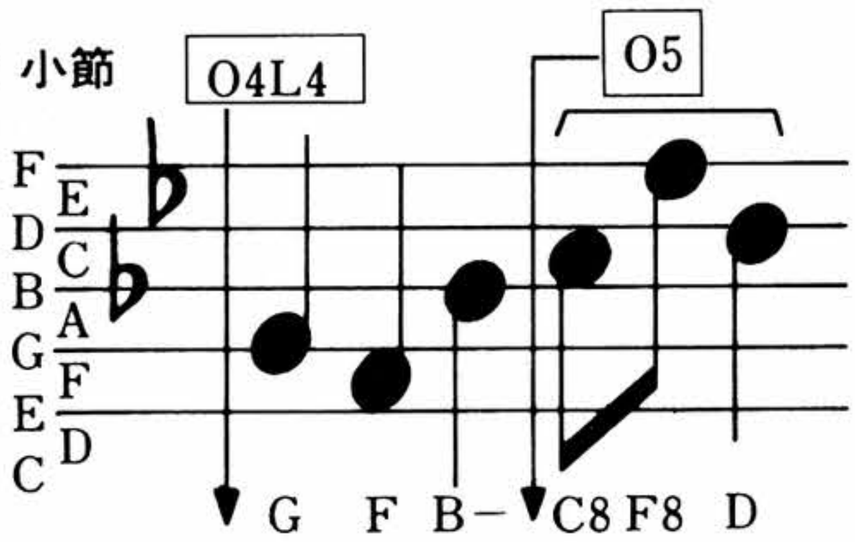
PUT KANJI文，PLAY文

ムソルグスキーの「展覧会の絵」の一部をPLAY文を使って演奏します．

40～150行はタイトル「展覧会の絵」〈演奏中〉を表示します．DATA文140と150行が，各漢字の区，点コード(奇数番目が区コード，偶数番目が点コード)です．漢字ROMを持っていない装置では，漢字は表示されませんので，160行以降だけを入力してください．

170～220行が演奏するためのルーチンです．ここでは240行以降のDATA文中に1小節ごとの音楽データを用意し，READ文で順次読み出して演奏します．

展　覧　会　の　絵　　ムソルグスキー

第1小節



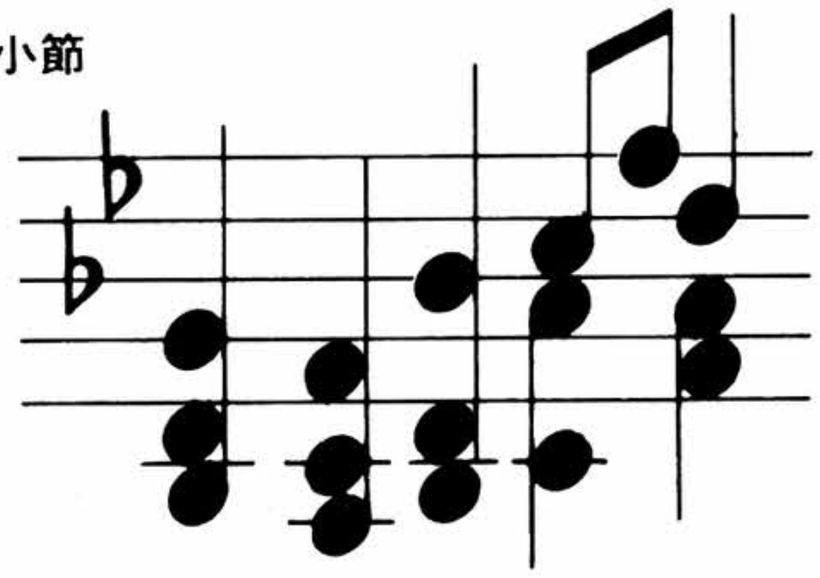
ボイスチャンネル1(250行)

O4L4GFB－O5C8F8D

ボイスチャンネル2，3(250行の・以降)

休符

第3小節



ボイスチャンネル1(270行) 上段の楽符

O4L4GFB－O5C8F8D

ボイスチャンネル2(280行) 中段の楽符

O4L4DCDAA

ボイスチャンネル3(290行) 下段の楽符

O3L4B－AB－O4CF

※BとEの部分には♭が付いていますので半音下がります．このためBはB－，EはE－となります．

```
10 '
20 '┌──────────────────────────────┐
   '│♥♥♥ てんらんかい の え ♥♥♥│
30 '└──────────────────────────────┘
40 '*** タイトル ひょうじ ***
50 SCREEN 5
60 FOR M=0 TO 6
70 READ K,T:C=&H2020+&H100*K+T
80 PUT KANJI(55+M*20,50),C
90 NEXT M
100 FOR M=0 TO 4
110 READ K,T:C=&H2020+&H100*K+T
120 PUT KANJI(75+M*20,80),C
130 NEXT M
140 DATA 1,56,37,24,45,87,18,81,4,46,19,8
,1,57
150 DATA 1,50,17,73,33,53,35,70,1,51
160 '*** えんそう ***
170 RESTORE 240
180 READ A$,B$,C$
190 IF A$="" THEN 220
200 PLAY A$,B$,C$
210 GOTO 180
220 IF PLAY(0)=-1 THEN 220 ELSE END
230 '**** ﾃﾞｰﾀ ****
240 DATA V13,V10,V10
250 DATA O4L4GFB-O5C8F8D,,
260 DATA C8F8DO4B-O5CO4GF,,
270 DATA O4L4GFB-O5C8F8D
280 DATA O4L4DCDAA
290 DATA O3L4B-AB-O4CF
300 DATA C8F8DO4B-O5CO4GF
310 DATA O4L4AB-GGCC
320 DATA O4L4CFDEO3GA
330 DATA O4L4B-O5C8F8DC8F8DO4B-
340 DATA O4FA8R8B-A8R8B-F
350 DATA O3DCO2B-AGO3G
360 DATA O5L4CO4GFGFB-
370 DATA O4GECDCD
380 DATA O3L4CEFO2B-AG
390 DATA O5L4C8F8DO4B-O5E-CO4B-
400 DATA O4A8R8B-GO5CO4AF
410 DATA O2L4FB-O3GCFB-
420 DATA ,,
```

第４章

資料編

4

予約語表

ABS
AND
ASC
ATN
ATTR$
AUTO
BASE
BEEP
BIN$
BLOAD
BSAVE
CALL
CDBL
CHR$
CINT
CIRCLE
CLEAR
CLOAD
CLOSE
CLS
CMD
COLOR
CONT
COPY
COS
CSAVE
CSNG
CSRLIN
CVD
CVI
CVS
DATA
DEF
DEFDBL
DEFINT
DEFSNG
DEFSTR
DELETE
DIM
DRAW
DSKF
DSKI$
DSKO$
ELSE
END
EOF
EQV
ERASE
ERL
ERR
ERROR
EXP
FIELD
FILES
FIX
FN
FOR
FPOS
FRE
GET
GO TO
GOSUB
GOTO
HEX$
IF
IMP
INKEY$
INP
INPUT
INSTR
INT
IPL
KEY
KILL
LEFT$
LEN
LET
LFILES
LINE
LIST
LLIST
LOAD
LOC
LOCATE
LOF
LOG
LPOS
LPRINT
LSET
MAX
MERGE
MID$
MKD$
MKI$
MKS$
MOD
MOTOR
NAME
NEW
NEXT
NOT
OCT$
OFF
ON
OPEN
OR
OUT
PAD
PAINT
PDL
PEEK
PLAY
POINT
POKE
POS
PRESET
PRINT
PSET
PUT
READ
REM
RENUM
RESTORE
RESUME
RETURN
RIGHT$
RND
RSET
RUN
SAVE
SCREEN
SET
SGN
SIN
SOUND
SPACE$
SPC(
SPRITE
SQR
STEP
STICK
STOP
STR$
STRIG
STRING$
SWAP
TAB(
TAN
THEN
TIME
TO
TROFF
TRON
USING
USR
VAL
VARPTR
VDP
VPEEK
VPOKE
WAIT
WIDTH
XOR

キャラクタコード表

上位4ビット(16進数)→

下位4ビット(16進数)↓		2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		0	@	P	`	p	♠			ー	タ	ミ	た	み
1	!	1	A	Q	a	q	♥	あ	。	ア	チ	ム	ち	む
2	"	2	B	R	b	r	♣	い	「	イ	ツ	メ	つ	め
3	#	3	C	S	c	s	◆	う	」	ウ	テ	モ	て	も
4	$	4	D	T	d	t	○	え	、	エ	ト	ヤ	と	や
5	%	5	E	U	e	u	●	お	・	オ	ナ	ユ	な	ゆ
6	&	6	F	V	f	v	を	か	ヲ	カ	ニ	ヨ	に	よ
7	´	7	G	W	g	w	ぁ	き	ァ	キ	ヌ	ラ	ぬ	ら
8	(	8	H	X	h	x	ぃ	く	ィ	ク	ネ	リ	ね	り
9	)	9	I	Y	i	y	ぅ	け	ゥ	ケ	ノ	ル	の	る
A	*	:	J	Z	j	z	ぇ	こ	ェ	コ	ハ	レ	は	れ
B	+	;	K	[	k	{	ぉ	さ	ォ	サ	ヒ	ロ	ひ	ろ
C	,	く	L	¥	l	¦	ゃ	し	ャ	シ	フ	ワ	ふ	わ
D	−	=	M	]	m	}	ゅ	す	ュ	ス	ヘ	ン	へ	ん
E	.	>	N	^	n	~	ょ	せ	ョ	セ	ホ	゛	ほ	
F	/	?	O	_	o		っ	そ	ッ	ソ	マ	゜	ま	■

グラフィックキャラクタ

	4	5
0		π
1	月	
2	火	
3	水	
4	木	
5	金	
6	土	
7	日	
8	年	
9	円	
A	時	
B	分	
C	秒	╳
D	百	大
E	千	中
F	万	小

↳カーソル

〈参考〉
特殊キーの
キーコード

キー		コード
BS		&H8
TAB		&H9
HOME		&HB
RETURN		&HD
INS		&H12
SELECT		&H18
ESC		&H1B
カーソルキー	→	&H1C
	←	&H1D
	↑	&H1E
	↓	&H1F
DEL		&H7F

例　Aのキャラクタコードは、&H41＝16×4＋1＝65(10進数)

　　¥のキャラクタコードは、&H5C＝16×5＋12＝92(10進数)

グラフィックキャラクタの入出力

グラフィックキャラクタは、グラフィクヘッダ(&H01)を付して入出力されます。例えば「時」の場合は以下のようになります。

キーボードからの入力：&H01, &H4Aの2バイトが入力されます。

テレビ、プリンタへの出力：&H01、&H4Aの2バイトを出力します。

例) テレビへの出力の場合　PRINT CHR$(1);CHR$(&H4A);

〈参考〉　10進数と16進数の対応

10進数	0～9	10	11	12	13	14	15	16	17	18	…	31	32	33	…	63	64	…	255
16進数	0～9	A	B	C	D	E	F	10	11	12	…	1F	20	21	…	3F	40	…	FF

すなわち、X_2X_1(16進数)＝X_2×16＋X_1(10進数) となります.

例) &H1F＝1×16＋15＝31(10進数)

　　&HFF＝15×16＋15＝255(10進数)

漢字コード表

漢字コードは縦軸の値(A)と横軸の(B)との和となります.

例) ●のコード＝2170＋C＝217C

(数値はいずれも16進数)

文字の配列はJIS読みで50音順に配置されています.

注) ・漢字コード表は旧字体で示していますが, 機種によっては新字体で表示されるものもあります.

・漢字コード222F～2327の範囲の記号は, JIS C 6226漢字コード表と異なっています.

また, この222F～2327の範囲の記号を印刷した場合, ご使用のプリンタによっては異なる記号が印刷されることがあります.

・画数の多い漢字については, 一部を簡略化して表示されるものがあります.

例) 鑒 → 鑑

記号 A＼B	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2120		(sp)	、	。	,	.	・	:	;	?	!	゛	゜	´	`	¨
2130	^	￣	_	ヽ	ヾ	ゝ	ゞ	〃	仝	々	〆	〇	―	―	‖	＼
2140	＼	～	∥	｜	…	‥	‘	’	“	”	(	)	〔	〕	[	]
2150	{	}	<	>	《	》	「	」	『	』	【	】	＋	－	±	×
2160	÷	＝	≠	＜	＞	≦	≧	∞	∴	♂	♀	°	′	″	℃	￥
2170	$	¢	£	%	#	&	*	@	§	☆	★	○	●	◎	◇	
2220		◆	□	■	△	▲	▽	▼	※	〒	→	←	↑	↓	＝	(株)
2230	(有)	Ⅰ	Ⅱ	Ⅲ	Ⅳ	Ⅴ	Ⅵ	Ⅶ	Ⅷ	Ⅸ	Ⅹ	g	m	cm	mm	km
2240	cm²	m²	mℓ	mg	kg	μs	(代)	K.K.	TEL	No.	インチ		グラム	メートル	センチ	ミリ
2250	ヘクタール	アール	ドル	トン	キロ	キログラム	キロワット	ワット	リットル	パーセント	ページ	ヤード	フィート	①	②	③
2260	④	⑤	⑥	⑦	⑧	⑨	⑩	〔	〕	㊮	☎	㊀	∫	√	―	￣
2270	｜	＿	∩	∪	⊂	⊃	∋	(	)	CR1	CR2	禁	均	↑	↓	
2320		◀	◀	▪▪	▪▪	▪	⁝	⇨								

	A＼B	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
英・数字	2330	0	1	2	3	4	5	6	7	8	9						
	2340		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	2350	P	Q	R	S	T	U	V	W	X	Y	Z					
	2360		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
	2370	p	q	r	s	t	u	v	w	x	y	z					
ひらがな	2420		ぁ	あ	ぃ	い	ぅ	う	ぇ	え	ぉ	お	か	が	き	ぎ	く
	2430	ぐ	け	げ	こ	ご	さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た
	2440	だ	ち	ぢ	っ	つ	づ	て	で	と	ど	な	に	ぬ	ね	の	は
	2450	ば	ぱ	ひ	び	ぴ	ふ	ぶ	ぷ	へ	べ	ぺ	ほ	ぼ	ぽ	ま	み
	2460	む	め	も	ゃ	や	ゅ	ゆ	ょ	よ	ら	り	る	れ	ろ	ゎ	わ
	2470	ゐ	ゑ	を	ん												
カタカナ	2520		ァ	ア	ィ	イ	ゥ	ウ	ェ	エ	ォ	オ	カ	ガ	キ	ギ	ク
	2530	グ	ケ	ゲ	コ	ゴ	サ	ザ	シ	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ
	2540	ダ	チ	ヂ	ッ	ツ	ヅ	テ	デ	ト	ド	ナ	ニ	ヌ	ネ	ノ	ハ
	2550	バ	パ	ヒ	ビ	ピ	フ	ブ	プ	ヘ	ベ	ペ	ホ	ボ	ポ	マ	ミ
	2560	ム	メ	モ	ャ	ヤ	ュ	ユ	ョ	ヨ	ラ	リ	ル	レ	ロ	ヮ	ワ
	2570	ヰ	ヱ	ヲ	ン	ヴ	ヵ	ヶ									
ギリシャ文字	2620		Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο
	2630	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω							
	2640		α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο
	2650	π	ρ	σ	τ	υ	φ	χ	ψ	ω							
ロシア文字	2720		А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н
	2730	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
	2740	Ю	Я														
	2750		а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н
	2760	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э
	2770	ю	я														

	A \ B	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
あ	3020		亜	唖	娃	阿	哀	愛	挨	姶	逢	葵	茜	穐	悪	握	渥
	3030	旭	葦	芦	鯵	梓	圧	斡	扱	宛	姐	虻	飴	絢	綾	鮎	或
	3040	粟	袷	安	庵	按	暗	案	闇	鞍	杏						
い	3040											以	伊	位	依	偉	囲
	3050	夷	委	威	尉	惟	意	慰	易	椅	為	畏	異	移	維	緯	胃
	3060	萎	衣	謂	違	遺	医	井	亥	域	育	郁	磯	一	壱	溢	逸
	3070	稲	茨	芋	鰯	允	印	咽	員	因	姻	引	飲	淫	胤	蔭	
	3120		院	陰	隠	韻	吋										
う	3120							右	宇	烏	羽	迂	雨	卯	鵜	窺	丑
	3130	碓	臼	渦	嘘	唄	欝	蔚	鰻	姥	厩	浦	瓜	閏	噂	云	運
	3140	雲															
え	3140		荏	餌	叡	営	嬰	影	映	曳	栄	永	泳	洩	瑛	盈	穎
	3150	頴	英	衛	詠	鋭	液	疫	益	駅	悦	謁	越	閲	榎	厭	円
	3160	園	堰	奄	宴	延	怨	掩	援	沿	演	炎	焔	煙	燕	猿	縁
	3170	艶	苑	薗	遠	鉛	鴛	塩									
お	3170								於	汚	甥	凹	央	奥	往	応	
	3220		押	旺	横	欧	殴	王	翁	襖	鴬	鴎	黄	岡	沖	荻	億
	3230	屋	憶	臆	桶	牡	乙	俺	卸	恩	温	穏	音				
か	3230													下	化	仮	何
	3240	伽	価	佳	加	可	嘉	夏	嫁	家	寡	科	暇	果	架	歌	河
	3250	火	珂	禍	禾	稼	箇	花	苛	茄	荷	華	菓	蝦	課	嘩	貨
	3260	迦	過	霞	蚊	俄	峨	我	牙	画	臥	芽	蛾	賀	雅	餓	駕
	3270	介	会	解	回	塊	壊	廻	快	怪	悔	恢	懐	戒	拐	改	
	3320		魁	晦	械	海	灰	界	皆	絵	芥	蟹	開	階	貝	凱	劾
	3330	外	咳	害	崖	慨	概	涯	碍	蓋	街	該	鎧	骸	浬	馨	蛙
	3340	垣	柿	蛎	鈎	劃	嚇	各	廓	拡	撹	格	核	殻	獲	確	穫
	3350	覚	角	赫	較	郭	閣	隔	革	学	岳	楽	額	顎	掛	笠	樫
	3360	橿	梶	鰍	潟	割	喝	恰	括	活	渇	滑	葛	褐	轄	且	鰹
	3370	叶	椛	樺	鞄	株	兜	竈	蒲	釜	鎌	噛	鴨	栢	茅	萱	
	3420		粥	刈	苅	瓦	乾	侃	冠	寒	刊	勘	勧	巻	喚	堪	姦
	3430	完	官	寛	干	幹	患	感	慣	憾	換	敢	柑	桓	棺	款	歓
	3440	汗	漢	澗	潅	環	甘	監	看	竿	管	簡	緩	缶	翰	肝	艦
	3450	莞	観	諌	貫	還	鑑	間	閑	関	陥	韓	館	舘	丸	含	岸
	3460	巌	玩	癌	眼	岩	翫	贋	雁	頑	顔	願					

	A \ B	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
き	3460												企	伎	危	喜	器
	3470	基	奇	嬉	寄	岐	希	幾	忌	揮	机	旗	既	期	棋	棄	
	3520		機	帰	毅	気	汽	畿	祈	季	稀	紀	徽	規	記	貴	起
	3530	軌	輝	飢	騎	鬼	亀	偽	儀	妓	宜	戯	技	擬	欺	犠	疑
	3540	祇	義	蟻	誼	議	掬	菊	鞠	吉	吃	喫	桔	橘	詰	砧	杵
	3550	黍	却	客	脚	虐	逆	丘	久	仇	休	及	吸	宮	弓	急	救
	3560	朽	求	汲	泣	灸	球	究	窮	笈	級	糾	給	旧	牛	去	居
	3570	巨	拒	拠	挙	渠	虚	許	距	鋸	漁	禦	魚	亨	享	京	
	3620		供	侠	僑	兇	競	共	凶	協	匡	卿	叫	喬	境	峡	強
	3630	彊	怯	恐	恭	挟	教	橋	況	狂	狭	矯	胸	脅	興	蕎	郷
	3640	鏡	響	饗	驚	仰	凝	尭	暁	業	局	曲	極	玉	桐	粁	僅
	3650	勤	均	巾	錦	斤	欣	欽	琴	禁	禽	筋	緊	芹	菌	衿	襟
	3660	謹	近	金	吟	銀											
く	3660						九	倶	句	区	狗	玖	矩	苦	躯	駆	駈
	3670	駒	具	愚	虞	喰	空	偶	寓	遇	隅	串	櫛	釧	屑	屈	
	3720		掘	窟	沓	靴	轡	窪	熊	隈	粂	栗	繰	桑	鍬	勲	君
	3730	薫	訓	群	軍	郡											
け	3730						卦	袈	祁	係	傾	刑	兄	啓	圭	珪	型
	3740	契	形	径	恵	慶	慧	憩	掲	携	敬	景	桂	渓	畦	稽	系
	3750	経	継	繋	罫	茎	荊	蛍	計	詣	警	軽	頚	鶏	芸	迎	鯨
	3760	劇	戟	撃	激	隙	桁	傑	欠	決	潔	穴	結	血	訣	月	件
	3770	倹	倦	健	兼	券	剣	喧	圏	堅	嫌	建	憲	懸	拳	捲	
	3820		検	権	牽	犬	献	研	硯	絹	県	肩	見	謙	賢	軒	遣
	3830	鍵	険	顕	験	鹸	元	原	厳	幻	弦	減	源	玄	現	絃	舷
	3840	言	諺	限													
こ	3840				乎	個	古	呼	固	姑	孤	己	庫	弧	戸	故	枯
	3850	湖	狐	糊	袴	股	胡	菰	虎	誇	跨	鈷	雇	顧	鼓	五	互
	3860	伍	午	呉	吾	娯	後	御	悟	梧	檎	瑚	碁	語	誤	護	醐
	3870	乞	鯉	交	佼	侯	候	倖	光	公	功	効	勾	厚	口	向	
	3920		后	喉	坑	垢	好	孔	孝	宏	工	巧	巷	幸	広	庚	康
	3930	弘	恒	慌	抗	拘	控	攻	昂	晃	更	杭	校	梗	構	江	洪
	3940	浩	港	溝	甲	皇	硬	稿	糠	紅	紘	絞	綱	耕	考	肯	肱
	3950	腔	膏	航	荒	行	衡	講	貢	購	郊	酵	鉱	砿	鋼	閤	降
	3960	項	香	高	鴻	剛	劫	号	合	壕	拷	濠	豪	轟	麹	克	刻
	3970	告	国	穀	酷	鵠	黒	獄	漉	腰	甑	忽	惚	骨	狛	込	
	3A20		此	頃	今	困	坤	墾	婚	恨	懇	昏	昆	根	梱	混	痕
	3A30	紺	艮	魂													

	A＼B	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
さ	3A30				些	佐	叉	唆	嵯	左	差	査	沙	瑳	砂	詐	鎖
	3A40	裟	坐	座	挫	債	催	再	最	哉	塞	妻	宰	彩	才	採	栽
	3A50	歳	済	災	采	犀	砕	砦	祭	斎	細	菜	裁	載	際	剤	在
	3A60	材	罪	財	冴	坂	阪	堺	榊	肴	咲	崎	埼	碕	鷺	作	削
	3A70	昨	搾	昨	朔	柵	窄	策	索	錯	桜	鮭	笹	匙	冊	刷	
	3B20		察	拶	撮	擦	札	殺	薩	雑	皐	鯖	捌	錆	鮫	皿	晒
	3B30	三	傘	参	山	惨	撒	散	桟	燦	珊	産	算	纂	蚕	讃	賛
	3B40	酸	餐	斬	暫	残											
し	3B40						仕	仔	伺	使	刺	司	史	嗣	四	士	始
	3B50	姉	姿	子	屍	市	師	志	思	指	支	孜	斯	施	旨	枝	止
	3B60	死	氏	獅	祉	私	糸	紙	紫	肢	脂	至	視	詞	詩	試	誌
	3B70	諮	資	賜	雌	飼	歯	事	似	侍	児	字	寺	慈	持	時	
	3C20		次	滋	治	爾	璽	痔	磁	示	而	耳	自	蒔	辞	汐	鹿
	3C30	式	識	鳴	竺	軸	宍	雫	七	叱	執	失	嫉	室	悉	湿	漆
	3C40	疾	質	実	蔀	篠	偲	柴	芝	屡	蕊	縞	舎	写	射	捨	赦
	3C50	斜	煮	社	紗	者	謝	車	遮	蛇	邪	借	勺	尺	杓	灼	爵
	3C60	酌	釈	錫	若	寂	弱	惹	主	取	守	手	朱	殊	狩	珠	種
	3C70	腫	趣	酒	首	儒	受	呪	寿	授	樹	綬	需	囚	収	周	
	3D20		宗	就	州	修	愁	拾	洲	秀	秋	終	繍	習	臭	舟	蒐
	3D30	衆	襲	讐	蹴	輯	週	酋	酬	集	醜	什	住	充	十	従	戎
	3D40	柔	汁	渋	獣	縦	重	銃	叔	夙	宿	淑	祝	縮	粛	塾	熟
	3D50	出	術	述	俊	峻	春	瞬	竣	舜	駿	准	循	旬	楯	殉	淳
	3D60	準	潤	盾	純	巡	遵	醇	順	処	初	所	暑	曙	渚	庶	緒
	3D70	署	書	薯	藷	諸	助	叙	女	序	徐	恕	鋤	除	傷	償	
	3E20		勝	匠	升	召	哨	商	唱	嘗	奨	妾	娼	宵	将	小	少
	3E30	尚	庄	床	廠	彰	承	抄	招	掌	捷	昇	昌	昭	晶	松	梢
	3E40	樟	樵	沼	消	渉	湘	焼	焦	照	症	省	硝	礁	祥	称	章
	3E50	笑	粧	紹	肖	菖	蒋	蕉	衝	裳	訟	証	詔	詳	象	賞	醤
	3E60	鉦	鍾	鐘	障	鞘	上	丈	丞	乗	冗	剰	城	場	壌	嬢	常
	3E70	情	擾	条	杖	浄	状	畳	穣	蒸	譲	醸	錠	嘱	埴	飾	
	3F20		拭	植	殖	燭	織	職	色	触	食	蝕	辱	尻	伸	信	侵
	3F30	唇	娠	寝	審	心	慎	振	新	晋	森	榛	浸	深	申	疹	真
	3F40	神	秦	紳	臣	芯	薪	親	診	身	辛	進	針	震	人	仁	刃
	3F50	塵	壬	尋	甚	尽	腎	訊	迅	陣	靭						

	A \ B	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
す	3F50											笥	諏	須	酢	図	厨
	3F60	逗	吹	垂	帥	推	水	炊	睡	粋	翠	衰	遂	酔	錐	錘	随
	3F70	瑞	髄	崇	嵩	数	枢	趨	雛	据	杉	椙	菅	頗	雀	裾	
	4020		澄	摺	寸												
せ	4020					世	瀬	畝	是	凄	制	勢	姓	征	性	成	政
	4030	整	星	晴	棲	栖	正	清	牲	生	盛	精	聖	声	製	西	誠
	4040	誓	請	逝	醒	青	静	斉	税	脆	隻	席	惜	戚	斥	昔	析
	4050	石	積	籍	績	脊	責	赤	跡	蹟	碩	切	拙	接	摂	折	設
	4060	窃	節	説	雪	絶	舌	蝉	仙	先	千	占	宣	専	尖	川	戦
	4070	扇	撰	栓	栴	泉	浅	洗	染	潜	煎	煽	旋	穿	箭	線	
	4120		繊	羨	腺	舛	船	薦	詮	賤	践	選	遷	銭	銑	閃	鮮
	4130	前	善	漸	然	全	禅	繕	膳	糎							
そ	4130										噌	塑	岨	措	曾	曽	楚
	4140	狙	疎	疏	礎	祖	租	粗	素	組	蘇	訴	阻	遡	鼠	僧	創
	4150	双	叢	倉	喪	壮	奏	爽	宋	層	匝	惣	想	捜	掃	挿	掻
	4160	操	早	曹	巣	槍	槽	漕	燥	争	痩	相	窓	糟	総	綜	聡
	4170	草	荘	葬	蒼	藻	装	走	送	遭	鎗	霜	騒	像	増	憎	
	4220		臓	蔵	贈	造	促	側	則	即	息	捉	束	測	足	速	俗
	4230	属	賊	族	続	卒	袖	其	揃	存	孫	尊	損	村	遜		
た	4230															他	多
	4240	太	汰	詑	唾	堕	妥	惰	打	柁	舵	楕	陀	駄	騨	体	堆
	4250	対	耐	岱	帯	待	怠	態	戴	替	泰	滞	胎	腿	苔	袋	貸
	4260	退	逮	隊	黛	鯛	代	台	大	第	醍	題	鷹	滝	瀧	卓	啄
	4270	宅	托	択	拓	沢	濯	琢	託	鐸	濁	諾	茸	凧	蛸	只	
	4320		叩	但	達	辰	奪	脱	巽	竪	辿	棚	谷	狸	鱈	樽	誰
	4330	丹	単	嘆	坦	担	探	旦	歎	淡	湛	炭	短	端	箪	綻	耽
	4340	胆	蛋	誕	鍛	団	壇	弾	断	暖	檀	段	男	談			
ち	4340														値	知	地
	4350	弛	恥	智	池	痴	稚	置	致	蜘	遅	馳	築	畜	竹	筑	蓄
	4360	逐	秩	窒	茶	嫡	着	中	仲	宙	忠	抽	昼	柱	注	虫	衷
	4370	註	酎	鋳	駐	樗	瀦	猪	苧	著	貯	丁	兆	凋	喋	寵	
	4420		帖	帳	庁	弔	張	彫	徴	懲	挑	暢	朝	潮	牒	町	眺
	4430	聴	脹	腸	蝶	調	諜	超	跳	銚	長	頂	鳥	勅	捗	直	朕
	4440	沈	珍	賃	鎮	陳											
つ	4440						津	墜	椎	槌	追	鎚	痛	通	塚	栂	掴
	4450	槻	佃	漬	柘	辻	蔦	綴	鍔	椿	潰	坪	壺	嬬	紬	爪	吊
	4460	釣	鶴														

	A \ B	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
て	4460			亭	低	停	偵	剃	貞	呈	提	定	帝	底	庭	廷	弟
	4470	悌	抵	挺	提	梯	汀	碇	禎	程	締	艇	訂	諦	蹄	逓	
	4520		邸	鄭	釘	鼎	泥	摘	擢	敵	滴	的	笛	適	鏑	溺	哲
	4530	徹	撤	轍	迭	鉄	典	填	天	展	店	添	纏	甜	貼	転	顛
	4540	点	伝	殿	澱	田	電										
と	4540							兎	吐	堵	塗	妬	屠	徒	斗	杜	渡
	4550	登	菟	賭	途	都	鍍	砥	砺	努	度	土	奴	怒	倒	党	冬
	4560	凍	刀	唐	塔	塘	套	宕	島	嶋	悼	投	搭	東	桃	梼	棟
	4570	盗	淘	湯	涛	灯	燈	当	痘	祷	等	答	筒	糖	統	到	
	4620		董	蕩	藤	討	謄	豆	踏	逃	透	鐙	陶	頭	騰	闘	働
	4630	動	同	堂	導	憧	撞	洞	瞳	童	胴	萄	道	銅	峠	鴇	匿
	4640	得	徳	涜	特	督	禿	篤	毒	独	読	栃	橡	凸	突	椴	届
	4650	鳶	苫	寅	酉	瀞	噸	屯	惇	敦	沌	豚	遁	頓	呑	曇	鈍
な	4660	奈	那	内	乍	凪	薙	謎	灘	捺	鍋	楢	馴	縄	畷	南	楠
	4670	軟	難	汝													
に	4670				二	尼	弐	迩	匂	賑	肉	虹	廿	日	乳	入	
	4720		如	尿	韮	任	妊	忍	認								
ぬ	4720									濡							
ね	4720										禰	称	寧	葱	猫	熱	年
	4730	念	捻	撚	燃	粘											
の	4730						乃	廼	之	埜	嚢	悩	濃	納	能	脳	膿
	4740	農	覗	蚤													
は	4740				巴	把	播	覇	杷	波	派	琶	破	婆	罵	芭	馬
	4750	俳	廃	拝	排	敗	杯	盃	牌	背	肺	輩	配	倍	培	媒	梅
	4760	楳	煤	狽	買	売	賠	陪	這	蝿	秤	矧	萩	伯	剝	博	拍
	4770	柏	泊	白	箔	粕	舶	薄	迫	曝	漠	爆	縛	莫	駁	麦	
	4820		函	箱	硲	箸	肇	筈	櫨	幡	肌	畑	畠	八	鉢	溌	発
	4830	醗	髪	伐	罰	抜	筏	閥	鳩	噺	塙	蛤	隼	伴	判	半	反
	4840	叛	帆	搬	斑	板	氾	汎	版	犯	班	畔	繁	般	藩	販	範
	4850	釆	煩	頒	飯	挽	晩	番	盤	磐	蕃	蛮					
ひ	4850												匪	卑	否	妃	庇
	4860	彼	悲	扉	批	披	斐	比	泌	疲	皮	碑	秘	緋	罷	肥	被
	4870	誹	費	避	非	飛	樋	簸	備	尾	微	枇	毘	琵	眉	美	
	4920		鼻	柊	稗	匹	疋	髭	彦	膝	菱	肘	弼	必	畢	筆	逼
	4930	桧	姫	媛	紐	百	謬	俵	彪	標	氷	漂	瓢	票	表	評	豹
	4940	廟	描	病	秒	苗	錨	鋲	蒜	蛭	鰭	品	彬	斌	浜	瀕	貧
	4950	賓	頻	敏	瓶												

	A \ B	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ふ	4950					不	付	埠	夫	婦	富	冨	布	府	怖	扶	敷
	4960	斧	普	浮	父	符	腐	膚	芙	譜	負	賦	赴	阜	附	侮	撫
	4970	武	舞	葡	蕪	部	封	楓	風	葺	蕗	伏	副	復	幅	服	
	4A20		福	腹	複	覆	淵	弗	払	沸	仏	物	鮒	分	吻	噴	墳
	4A30	憤	扮	焚	奮	粉	糞	紛	雰	文	聞						
へ	4A30											丙	併	兵	塀	幣	平
	4A40	弊	柄	並	蔽	閉	陛	米	頁	僻	壁	癖	碧	別	瞥	蔑	箆
	4A50	偏	変	片	篇	編	辺	返	遍	便	勉	娩	弁	鞭			
ほ	4A50														保	舗	鋪
	4A60	圃	捕	歩	甫	補	輔	穂	募	墓	慕	戊	暮	母	簿	菩	倣
	4A70	俸	包	呆	報	奉	宝	峰	峯	崩	庖	抱	捧	放	方	朋	
	4B20		法	泡	烹	砲	縫	胞	芳	萌	蓬	蜂	褒	訪	豊	邦	鋒
	4B30	飽	鳳	鵬	乏	亡	傍	剖	坊	妨	帽	忘	忙	房	暴	望	某
	4B40	棒	冒	紡	肪	膨	謀	貌	貿	鉾	防	吠	頬	北	僕	ト	墨
	4B50	撲	朴	牧	睦	穆	釦	勃	没	殆	堀	幌	奔	本	翻	凡	盆
ま	4B60	摩	磨	魔	麻	埋	妹	昧	枚	毎	哩	槇	幕	膜	枕	鮪	柾
	4B70	鱒	桝	亦	俣	又	抹	末	沫	迄	侭	繭	麿	万	慢	満	
	4C20		漫	蔓													
み	4C20				味	未	魅	巳	箕	岬	密	蜜	湊	蓑	稔	脈	妙
	4C30	耗	民	眠													
む	4C30				務	夢	無	牟	矛	霧	鵡	椋	婿	娘			
め	4C30														冥	名	命
	4C40	明	盟	迷	銘	鳴	姪	牝	滅	免	棉	綿	緬	面	麺		
も	4C40															摸	模
	4C50	茂	妄	孟	毛	猛	盲	網	耗	蒙	儲	木	黙	目	杢	勿	餅
	4C60	尤	戻	籾	貫	問	悶	紋	門	匁							
や	4C60										也	冶	夜	爺	耶	野	弥
	4C70	矢	厄	役	約	薬	訳	躍	靖	柳	薮	鑓					
ゆ	4C70												愉	愈	油	癒	
	4D20		諭	輸	唯	佑	優	勇	友	宥	幽	悠	憂	揖	有	柚	湧
	4D30	涌	猶	猷	由	祐	裕	誘	遊	邑	郵	雄	融	夕			
よ	4D30														予	余	与
	4D40	誉	輿	預	傭	幼	妖	容	庸	揚	揺	擁	曜	楊	様	洋	溶
	4D50	熔	用	窯	羊	耀	葉	蓉	要	謡	踊	遥	陽	養	慾	抑	欲
	4D60	沃	浴	翌	翼	淀											

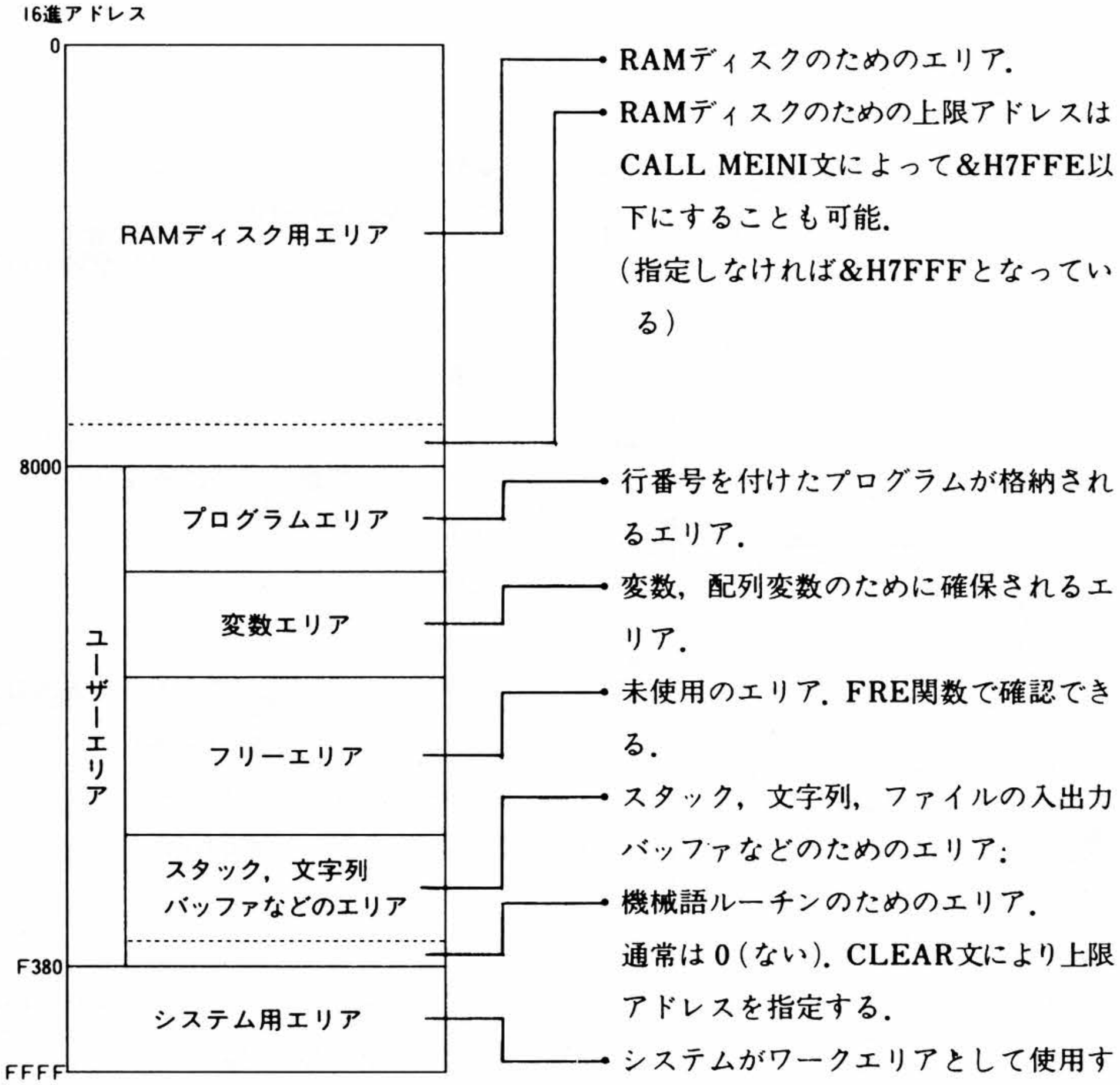
	A＼B	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ら	4D60						羅	螺	裸	来	莢	頼	雷	洛	絡	落	酪
	4D70	乱	卵	嵐	欄	濫	藍	蘭	覧								
り	4D70									利	吏	履	李	梨	理	璃	
	4E20		痢	裏	裡	里	離	陸	律	率	立	葎	掠	略	劉	流	溜
	4E30	琉	留	硫	粒	隆	竜	龍	侶	慮	旅	虜	了	亮	僚	両	凌
	4E40	寮	料	梁	涼	猟	療	瞭	稜	糧	良	諒	遼	量	陵	領	力
	4E50	緑	倫	厘	林	淋	燐	琳	臨	輪	隣	鱗	麟				
る	4E50													瑠	塁	涙	累
	4E60	類															
れ	4E60		令	伶	例	冷	励	嶺	怜	玲	礼	苓	鈴	隷	零	霊	麗
	4E70	齢	暦	歴	列	劣	烈	裂	廉	恋	憐	漣	煉	簾	練	聯	
	4F20		蓮	連	錬												
ろ	4F20					呂	魯	櫓	炉	賂	路	露	労	婁	廊	弄	朗
	4F30	楼	榔	浪	漏	牢	狼	篭	老	聾	蝋	郎	六	麓	禄	肋	録
	4F40	論															
わ	4F40		倭	和	話	歪	賄	脇	惑	枠	鷲	亙	亘	鰐	詫	藁	蕨
	4F50	椀	湾	碗	腕												

コントロールコード表

コード（10進）	コード（16進）	機　　能	対応キー
0	00		
1	01	グラフィックキャラクタの入出力時のヘッダ	CTRL ＋ A
2	02	カーソルを直前の語の先頭へ移動	CTRL ＋ B
3	03	入力待ち状態を終了する	CTRL ＋ C
4	04		CTRL ＋ D
5	05	カーソル以下を削除	CTRL ＋ E
6	06	カーソルを次の語の先頭へ移動	CTRL ＋ F
7	07	スピーカを鳴らす（BEEP文と同じ）	CTRL ＋ G，
8	08	カーソルの１つ前の文字を削除する	CTRL ＋ H，BS
9	09	次の水平タブ位置へ移動	CTRL ＋ I，TAB
10	0A	行送り（ラインフィード）	CTRL ＋ J
11	0B	カーソルをホームポジション（左上）に戻す	CTRL ＋ K
12	0C	画面をクリアし、カーソルをホームポジションに戻す	CTRL ＋ L，
13	0D	カーソルを左端に戻す（キャリッジターン）	CTRL ＋ M，RETURN
14	0E	カーソルを行末へ移動	CTRL ＋ N
15	0F		CTRL ＋ O
16	10		CTRL ＋ P
17	11		CTRL ＋ Q，
18	12	挿入モードのON/OFFスイッチ	CTRL ＋ R，INS
19	13		CTRL ＋ S
20	14		CTRL ＋ T
21	15	一行を画面から削除	CTRL ＋ U
22	16		CTRL ＋ V
23	17		CTRL ＋ W
24	18		CTRL ＋ X，SELECT
25	19		CTRL ＋ Y
26	1A		CTRL ＋ Z
27	1B		CTRL ＋ [，ESC
28	1C	カーソルを右へ移動	CTRL ＋ ¥，→
29	1D	カーソルを左へ移動	CTRL ＋]，←
30	1E	カーソルを上へ移動	CTRL ＋ ^，↑
31	1F	カーソルを下へ移動	CTRL ＋ _，↓
127	7F	カーソルの指す文字を削除	DEL

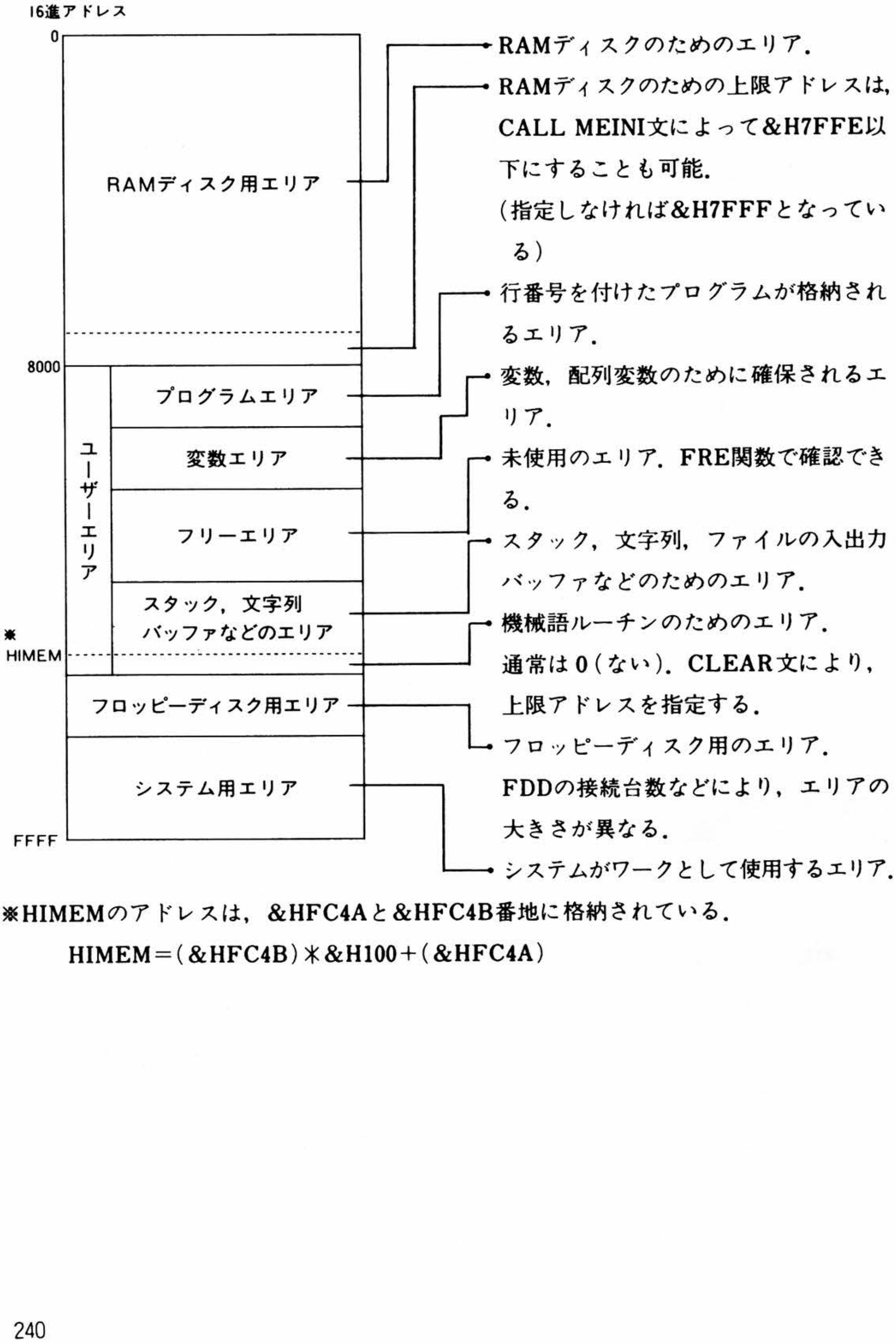
メモリマップ(BASIC使用時)

●フロッピーディスクを持っていない装置



※フロッピーディスクが接続されている装置でも，電源オン時 SHIFT キーを押しながら起動した場合は，このマップの状態になります.

●フロッピーディスクを持っている装置



※HIMEMのアドレスは，&HFC4Aと&HFC4B番地に格納されている．

HIMEM＝(&HFC4B)＊&H100＋(&HFC4A)

エラーメッセージ表

((D)はDISK BASIC関係のエラーメッセージ)

メッセージ	エラーコード	説明
Bad allocation table (D)	60	ディスクがイニシャライズされていない.
Bab drive name (D)	62	ドライブ名の指定が間違っている.
Bad file mode (D)	61	シーケンシャルファイルに対してPUT文,GET文またはLOF関数が使用されたり,ランダムファイルに対してLOAD文が使用された. また, OPEN文のファイルモードとしてI, O, A, R以外が設定されている.
Bad file name	56	ファイル名が適当でない.
Bad file number	52	オープンしていない(開いていない)ファイル番号を指定した. 指定したファイル番号がMAXFILES文で指定した数を越えている.
Can't continue	17	CONTコマンドでプログラムの実行を再開できない. エラーが発生して終了したプログラムは再開できません. また, プログラムの1部を書き換えたり, CLEAR文などを実行すると, 次に実行すべきポインタが壊れてプログラムを再開することはできなくなります.
Device I/O error	19	カセットレコーダ, プリンタ, CRTなどへの入出力中にエラーが発生した.
Direct statement	57	アスキー形式のプログラムファイルをロード中, プログラム以外のデータがあった.
Disk full (D)	66	ディスク上に空領域が無い.
Disk I/O error (D)	69	ディスク入出力操作において, オペレ--ティングシステムが回復できない入出力エラーが生じた.
Disk offline (D)	70	フロッピーディスクが装着されていない.

メッセージ	エラーコード	説　　明
Disk write protected (D)	68	書き込み禁止のディスクに書き込みを行った.
Division by zero	11	0による除算を行おうとした．または，0に対して負のべき乗を行った.
Field overflow (D)	50	FIELD文で定義したフィールドサイズの合計が256バイトまたは指定の値を越えた.
File already exists (D)	65	NAME文の新しいファイル名として指定された名前が，ディスク上に登録されているファイル名と重複した.
File already open	54	OPEN文で指定したファイル番号は，他のファイルで使用中です.
File not found	53	指定されたファイルが見つからない.
File not OPEN	59	OPEN文で開いていないファイルに対して入出力を行おうとした.
File still open (D)	64	ファイルのCLOSEを忘れている.
Illegal direct	12	ダイレクトモードでは実行できないステートメントをダイレクトモードで実行しようとした.
Illegal function call	5	ステートメントや関数の使い方が誤っている．引数などが指定可能な範囲を越えている.
Input past end	55	ファイル中のデータをすべて読み込んだ後，さらに入力文を実行した．ファイル中のデータをすべて読み込んだかどうかはEOF関数で調べてください.
Internal error	51	BASIC内部でエラーが発生した.
Line buffer overflow	25	入力されたデータが許されている文字数を越えた.

メッセージ	エラーコード	説　　明
Missing operand	24	文中に必要なパラメータが指定していない.
NEXT without FOR	1	NEXT文に対応するFOR文がない.
No RESUME	21	エラー処理ルーチンにRESUME文がない.
Out of DATA	4	READ文で読み出せるデータがない.
Out of memory	7	プログラムが長すぎたり，配列が大きすぎたりなどの原因でメモリが足りなくなった.
Out of string space	14	文字列変数用のメモリが足りなくなった.
Overflow	6	演算結果や，入力された数値が型に応じた許容範囲を超えた.
Redimensioned array	10	配列変数を2重に定義しようとした．または，配列変数に添字の最大値10が割当てられた後で配列を定義しようとした.
Rename across disk (D)	71	異なるドライブ間でNAMEを実行した.
RESUME without error	22	エラー処理ルーチン以外でRESUME文を使用した.
RETURN without GOSUB	3	GOSUB文とRETURN文が正しく対応していない.
Sequential I/O only	58	シーケンシャルファイルに対してランダム入出力を行おうとした.
String formula too complex	16	指定した文字列が複雑すぎる．文字式を2つ以上に分けてください.
String too long	15	文字列が255文字より大きい.

メッセージ	エラーコード	説　　明
Subscript out of range	9	指定した配列変数の添字の値が，DIM文で定義した配列変数の添字の最大値より大きい．
Syntax error	2	文の書き方(構文)を誤っている．(かっこが対応していない．コマンド・ステートメントの繰りを誤っている．区切り記号がないなど)
Too many files	67	ファイルの数が112を越えた．すなわち，ディレクトリの112個のエントリすべてが使用されている時に新しいファイルを生成しようとした． RAMディスクの場合は，ファイル数が32を越えるとこのエラーになります．
Type mismatch	13	数式や数値関数に文字列を使ったり，文字式や文字関数に数値を使うなど，データの形式が一致しない．
Undefined line number	8	GOTO文，GOSUB文，IF文などで指定した行番号がプログラム中にない．
Undefined user function	18	DEF FNで定義していないユーザーを定義関数を使っている．
Unprintable error	23 26~49 72~255	メッセージが定義されていないエラー．
Veri error	20	カセットにセーブされたプログラムはメモリ上のプログラムと内容が異なる． プログラムのセーブをやり直してください．

第５章

付　　録

5

カラーコードについて

1）カラーコード0の色を変える方法

カラーコード0は通常透明色に固定されていますが，次の命令を実行すると，他のカラーコードと同じように，好きな色を割り当てて使用することができます．

```
VDP(9)=VDP(9) OR &H20
```

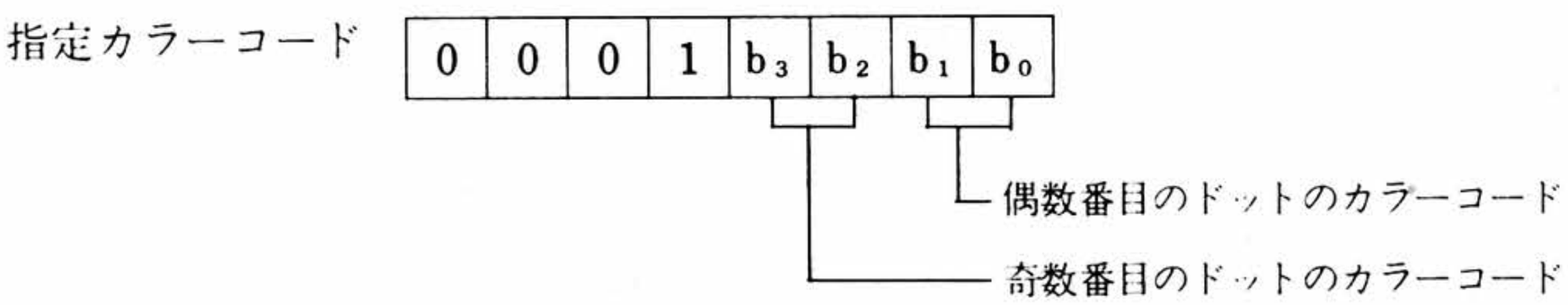
もとの状態にもどしたい場合は次の命令を実行します．

```
VDP(9)=VDP(9) AND &HDF
```

ただし，SET VIDEO 2を実行した状態では，上記にかかわらず，カラーコード0は透明に固定されます．

2）SCREEN 6での周辺色，スプライト色の特例

SCREEN 6では，0～3の4つのカラーコードしか使用することはできませんが，周辺色とスプライトについては，ハードウェアタイリング機能により10色の色を使用することができます．ハードウェアタイリング機能とは，X座標の偶数番目のドットと奇数番目のドットに別個にカラーコード(0～3)を指定できる機能です．
カラーコードとして16～31を指定するとタイリング機能による表示となります．



したがって，異なる色として使用できる色数は下記の10色です．(　)内は同系色．

16，17(20)，18(24)，19(28)，21，22(25)，23(29)，26，27(30)，31

カラーコード0～15の場合は，4で割った余りの値が実際のカラーコードとなります．この場合は，偶数・奇数ドットとも同じ色となります．

注）・前景色，背景色にはハードウェアタイリング機能はありません．
　　使用できる色はカラーコード0～3の4色だけです．(SCREEN 6の場合)

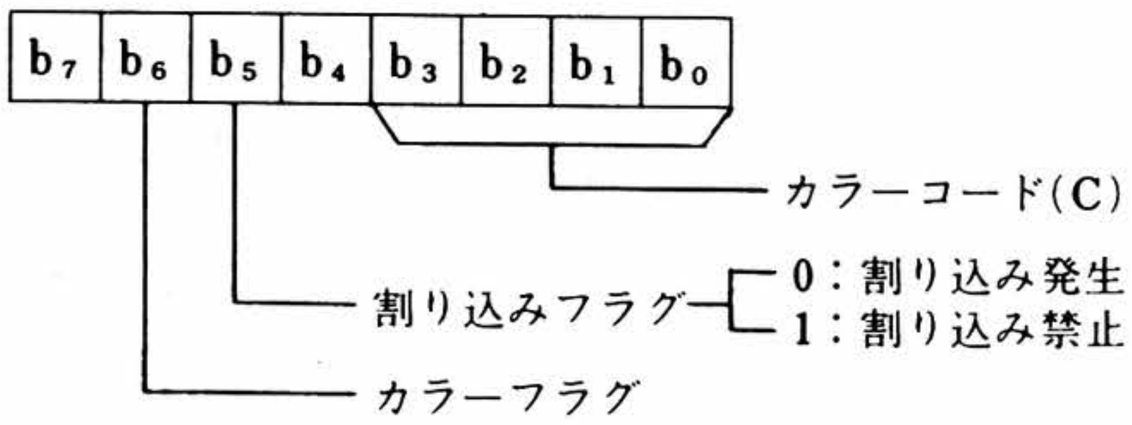
次頁に上記10色を表示するプログラムを示していますので参考にしてください。

・SCREEN 6 の10色を表示するプログラムです。

```
10 '*** SCREEN 6 COLOR SAMPLE ***
20 SCREEN 6,3:COLOR 3,2,3:CLS
30 VDP(9)=VDP(9) OR &H20
40 SPRITE$(0)=STRING$(32,CHR$(255))
50 COLOR=(0,7,0,0)
60 COLOR=(1,0,7,0)
70 COLOR=(2,0,0,7)
80 COLOR=(3,7,7,7)
90 OPEN "GRP:" FOR OUTPUT AS#1
100 PRESET(76,60):PRINT #1,"COLOR CODE"
110 PRESET(76,70)
120 PRINT #1,"16   17   18   19   21   2
2   23   26   27   31"
130 CLOSE #1
140 PUT SPRITE 0,(20,80),16,0
150 PUT SPRITE 1,(40,84),17,0
160 PUT SPRITE 2,(60,88),18,0
170 PUT SPRITE 3,(80,92),19,0
180 PUT SPRITE 4,(100,96),21,0
190 PUT SPRITE 5,(120,100),22,0
200 PUT SPRITE 6,(140,104),23,0
210 PUT SPRITE 7,(160,108),26,0
220 PUT SPRITE 8,(180,112),27,0
230 PUT SPRITE 9,(200,116),31,0
240 GOTO 240
```

COLOR SPRITE補足説明

COLOR SPRITE命令には，スプライトの行毎に，他のスプライトと重なった部分の色を変えたり，重なりによる割り込みを禁止するような機能もあります．
指定方法は，カラーコードのビット5と6によります．



割り込みフラグ(b_5)は，〝1〟にするとその部分が他のスプライトと重なっても割り込みを発生しなくなります．このフラグは，COLOR SPRITE$文により，スプライトの各行毎に設定することができます．COLOR SPRITE(N)＝〈カラーコード〉＋&H20とするとそのスプライトパターン全体に割り込み禁止フラグが設定されます．

カラーフラグ(b_6)は次のような動作をします.

- $b_6 = 1$の部分のパターンは，画面の同一走査行に，そのスプライト面番号より小さい面番号で，かつ$b_6 = 0$のパターンがある時のみ表示される.
- 上記$b_6 = 0$のパターンと$b_6 = 1$(複数個あっても可)のパターンの重なった部分は各々のカラーコード(C)のOR論理によって得られるカラーコードの色となる.
- 前記パターンの重なり部での割り込みは発生しない.

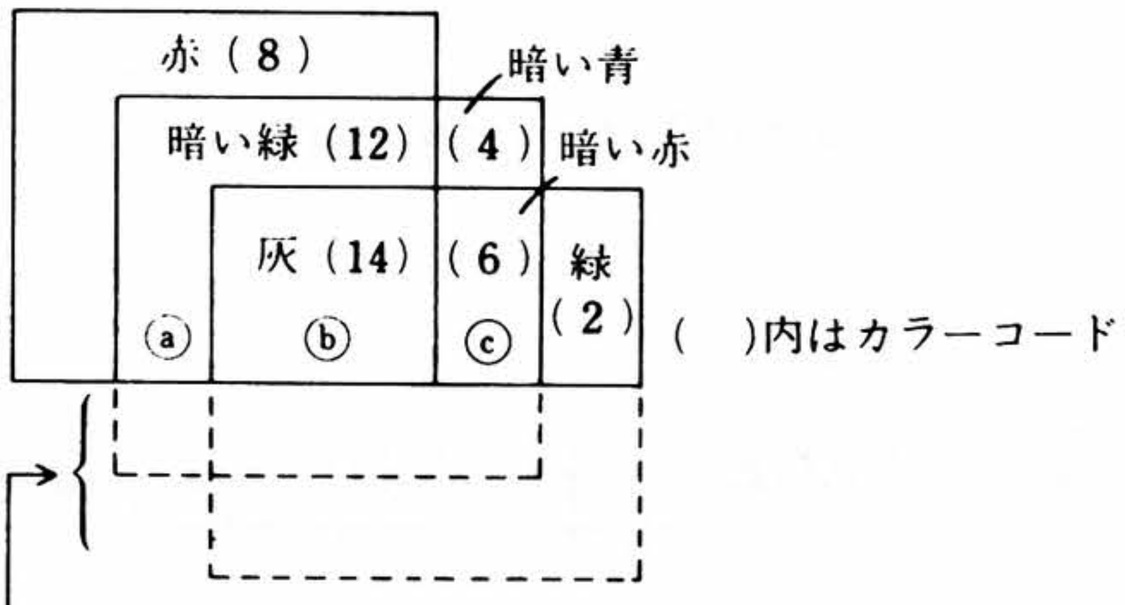
例）次のプログラムを実行すると，下図のようになります.（カラーコードは初期状態）

```
10 COLOR ,1:SCREEN 5,3
20 SPRITE$(0)=STRING$(32,CHR$(&HFF))
30 COLOR SPRITE(5)=8
40 COLOR SPRITE(6)=&H44
50 COLOR SPRITE(7)=&H42
60 PUT SPRITE 5,(100,100),,0
70 PUT SPRITE 6,(110,110),,0
80 PUT SPRITE 7,(120,120),,0
90 GOTO 90
```

ⓐ部…… 8 OR 4 =12

ⓑ部…… 8 OR 4 OR 2 =14

ⓒ部…… 4 OR 2 = 6

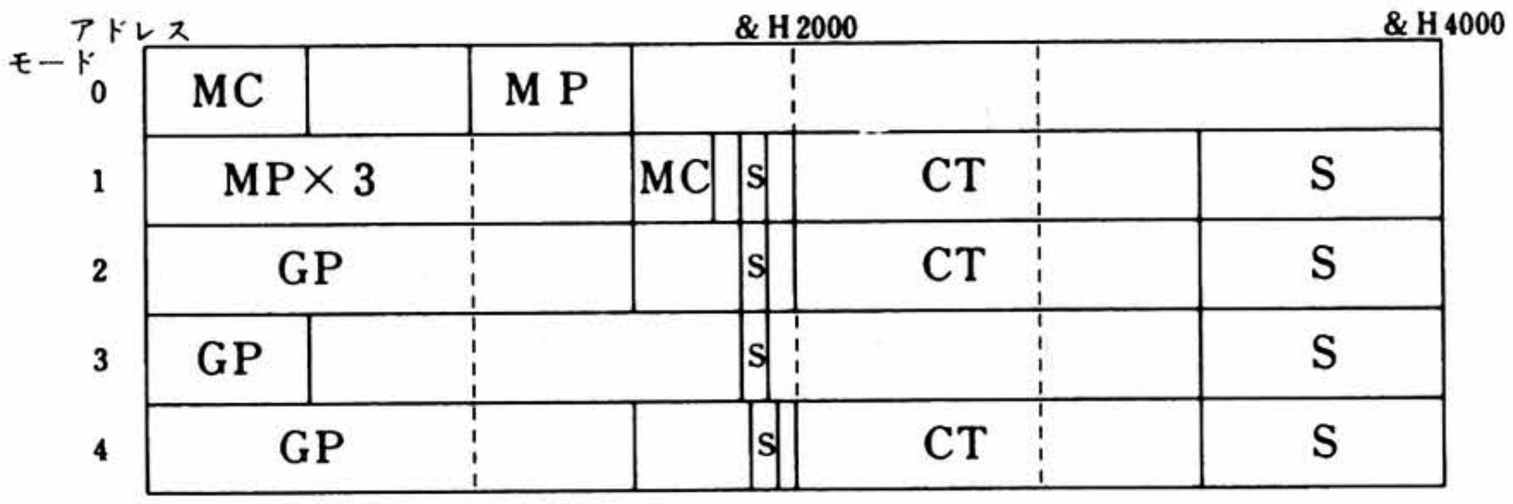


〈ご注意〉

- 画面に表示する順番はスプライト番号の小さい順に表示するようにしてください.
  順番が異なると，スプライトが重なった部分の色が異なることがあります.

VRAMについて,

SCREEN 0 ～ 4 はVRAMの 0 ～ &H4000番地を使用しており概略の内訳は下図の通りです.



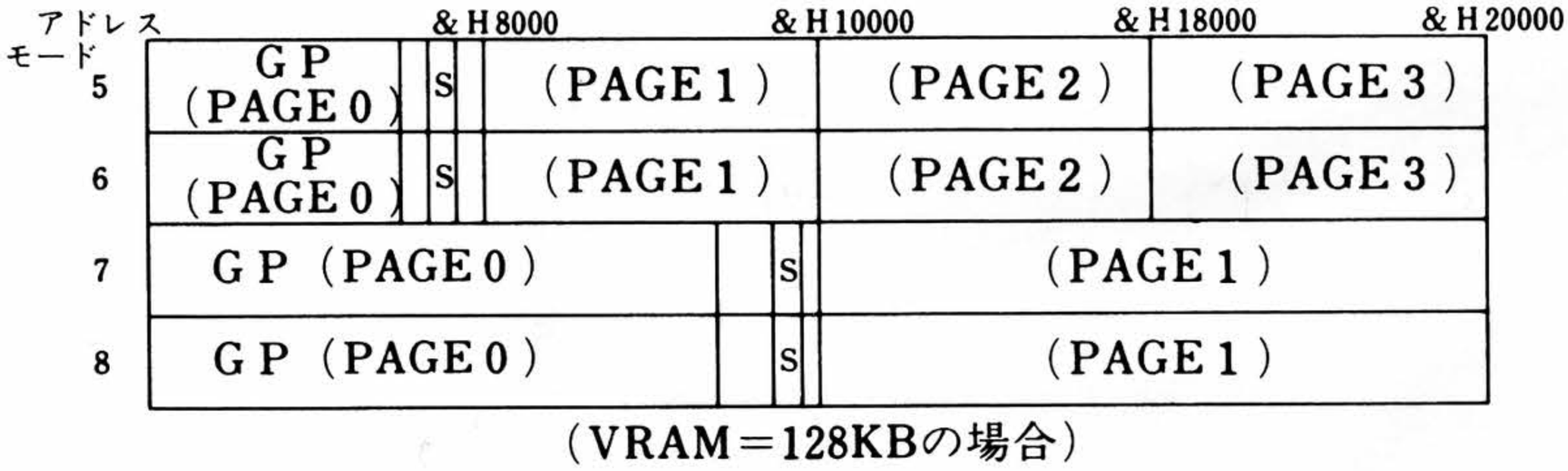
MC：文字コード領域　　MP：文字パターン領域

S：スプライト関係領域　　CT：カラーテーブル領域

GP：グラフィックパターン領域

SCREEN 2 と4ではGP領域の1ドットが 1 画素に対応しています. CTはGPの各バイトと対応しておりGPの 1 バイト(8 画素) 毎の前景色と背景色を定義しています. SCREEN 3 では, 2 ブロック(4 × 4 ドット／ブロック) に対してVRAM 1バイトを割り当てています. VRAMの上位4ビットで一方の色(16色)を, 下位 4 ビットで他方の色を指定しています.

SCREEN 5 ～ 8 では,VRAM を以下のように割り当てます.



(VRAM＝128KBの場合)

図はPAGE 0 だけについて割り当てを書いていますが他のページも同様です.

SCREEN 5は，GP 1バイトに対して2画素が割り当てられます．1画素当たり4ビットありますので，16色の表示が可能です．これに対して，SCREEN 6では，GP 1バイトに対して4画素が割り当てられます．このため1画素当たりのビットは2ビットとなります．4色の表示しかできないのはこのためです．

SCREE 7は，SCREEN 5と同様，GP 1バイトに対して2画素であるため16色の表示が可能です．ただし，横方向のドット数が倍となっているためGPの領域も倍となっています．

SCREEN 8はGP 1バイトに対して1バイトが割り当てられます。このため256色の表示が可能となっています。

前記VRAM割り当て図は簡略的に示したもので，一部省略している所もあります．

SCREEN 7，8は図だけで見るとVRAM 64KBの装置でも使えるように見えますが，実際には，使用できません．(2ページのペアで使用するようになっているためです．)

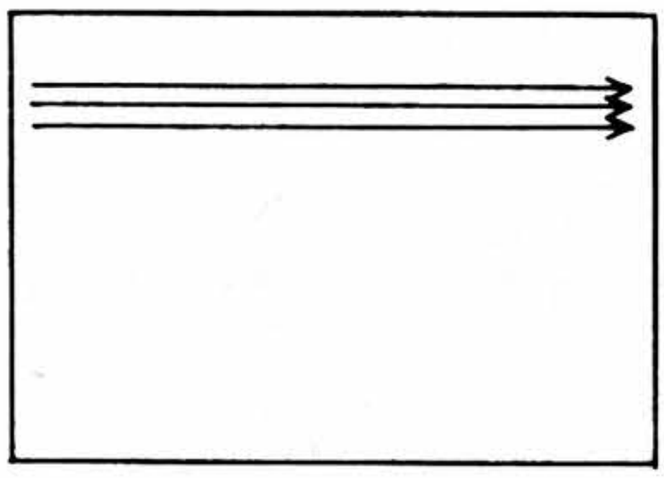
・SCREEN 5～8のGP(グラフィックパターン)の割当て

ビット	b7	b6	b5	b4	b3	b2	b1	b0
SCREEN 5	← 1ドット →				← 1ドット →			
SCREEN 6	← 1ドット →		← 1ドット →		← 1ドット →		← 1ドット →	
SCREEN 7	← 1ドット →				← 1ドット →			
SCREEN 8	← 1ドット →							

※1ドットと記した部分に，そのドットに対応したカラーコードが入っています．

・画面との対応

図のように最上段の走査線の左端より右端，2番目の走査線の左端より右端……の順序で，各ドットのカラーコードがグラフィックパターン領域の先頭より格納されています．



スーパーインポーズとディジタイズ

1) スーパーインポーズ

SET VIDEO 2,YM,,1,SU,VI

上記命令でスーパーインポーズモードとなります．ただし，テレビ画面を表示する部分の色および周辺色はカラーコード0(透明色)にしてください．

YMを"1"にすると，テレビまたはビデオの画面の輝度を落とします．SUは省略または0～3の数値で音声を制御します．VIは，ビデオ入力の制御です．SCREEN 5の画面中央の丸の中にテレビ画面を表示する例を次に示します．

```
10 SCREEN 5:COLOR ,4,0:CLS
20 SET VIDEO 2,,,1,,0
30 CIRCLE(128,100),80,0
40 PAINT(128,100),0
50 GOTO 50
```

注) スーパーインポーズ機能のないパソコンではスーパーインポーズはできません。くわしくはパソコンの説明書を参照してください。

2) ディジタイズ

COPY SCREEN文によりRGBマルチ(21ピンRGB)またはビデオ入力端子より入力された画像信号をパソコン画面に取り込むことができます．このような機能をディジタイズといいます．ただし機種によっては，ディジタイズ機能を持っていないものもあります．

画面を取り込みのためのプログラム例を次に示します．

```
10 SCREEN 8,,,,,1
20 VDP(7)=255
30 SET VIDEO 1,,1,1,,0
40 COPY SCREEN
50 SET VIDEO ,,0
60 IF INKEY$="" THEN 60 ELSE 30
```

何かのキーを押した瞬間の画面を取り込み静止画として表示します．

連続するフィールドを取り込む場合の例を次に示します.

```
10 SCREEN 8,,,,,3
20 SET PAGE 1,1:VDP(7)=255
30 SET VIDEO 1,,1,1,,0
40 COPY SCREEN 1
50 SET VIDEO ,,0
60 IF INKEY$="" THEN 60 ELSE 30
```

〈参考〉

- SCREEN 5 ～ 7 でも実行できますが,取り込んだ色はテレビ表示と異なったものになります.
- CTRL + STOP で実行を停止した時, 画面が細かく上下にゆれている感じの表示になります. これはインタレース表示となっているためです. SCREEN,,,,, 0 とSET VIDEO 0,, 0, 0 を実行すると通常の表示にもどります.

CTRL + STOP を押すタイミングによっては, 画面に何も表示されない等の異常になることがあります. このときはSET VIDEO 0,, 0, 0 を実行してください. 入力した文字は画面には表示されませんが ↵ キーを押すと表示されます. このようなことを極力防ぐために, プログラム例のように30, 40, 50 行の命令は 1 組として実行するようにしてください.

ご注意

SET VIDEO文によって, 外部同期, 内部同期を切り換えると, 同期が安定するまでに時間を要します. この同期の不安定な間にディジタイズした静止画は乱れた画面となります. したがって, 同期の不安定な間にディジタイズしないよう以下の点を配慮してください.( )で示した程度のソフトタイマを挿入すれば問題ありません.

- 同期を切り換えてしばらくはディジタイズしないこと.
  (FOR T=0 T0 200 : NEXT T)
- COPY SCREEN文実行後, ディジタイズが完了するまでの間(60m秒程度) 同期を切り換えないこと.
  (FOR T=0 T0 10 : NEXT T)

索引

あ行

アクティブページ ……177
アラーム …… 78,176
RGBマルチ……252
引数 ……16,61
インターバルタイマ ……116,204
インタレース ……172
インテンジャー …… 41
引用符 …… 5
エラー …… 71,114
エラーコード …… ·72
エラーメッセージ……18,72,241
演算 …… 11
演算の優先順位 …… 18
エンベロープ ……135,184

か行

拡張ステートメント …… 39
カーソル …… 59
型宣言文字 …… 7
型変換 …… 10
画面の取り込み……56,252
画面モード …… 19,171
カラーコード ……25,48
カラーコードの初期化 ……25,48
漢字コード……160,229
関数 ……16,61
関数演算 …… 13
機械語 …… 63,206
キャラクタコード……228
キャラクタセット …… 4
行 …… 3
行番号 …… 3
組込み関数……16
グラフィックキャラクタ…… 228
グラフィック座標系……22
グラフィックモード……22
コマンド……2
コントロールコード表……238

さ行

サインビット …… 6
サブルーチン …… 79
算術演算 …… 12
次元 …… 8
時刻 …… 78,176,220
システム変数 …… 9
周辺色 …… 49
ジョイスティック ……193,197
16進形式 …… 5
条件判断 ……15,85
定数 …… 5
使用メモリ …… 8
剰余 …… 12
書式制御 ……144
数値型 …… 4
ステートメット …… 3
スーパーインポーズ……178,252
スプライト …… 26,159,188
スプライトサイズ …… 26,171
スプライトパターン …… 26,189
整数型 ……5,7
前景色 …… 48

た行

ダイレクトモード …… 2
単精度 ……4,6,7,9,11,62
ディジタイズ……56,178,252
ディスプレイページ ……177
テキストモード …… 20
データ …… 4,60
デバイス名 …… 28
透明色…… 48,56
同期……178

な行

2進形式 …… 5
ヌルストリング …… 6
ノンインタレース …… 172

は行

背景色 …… 49
倍精度 …… 4,6,7,9,11,62
配列 …… 7,8,55,65
パスワード …… 174
パレット …… 25
日付 …… 78,176
ファイル …… 27,55,69,126
ファイル名 …… 29
ファイル番号 …… 28
ファイルスペック …… 28
ファンクションキー …… 96,97,98
VRAM …… 211,250
複文 …… 3
ページ …… 177,219
フリーエリア …… 77
プログラム …… 3
プログラムモード …… 2
ブロック …… 23
プロンプト …… 175
文 …… 2
変数 …… 6
変数名 …… 7
ボーレート …… 58,172

ま行

マルチステートメント …… 3
マルチカラーモード …… 23
メモリマップ …… 239
文字型 …… 4,7,9,62
文字定数 …… 5
文字の入力 …… 90,92,102
文字列 …… 4,17
文字列関数 …… 16

や行

ユーザー定義関数 …… 61
予約語 …… 227

ら、わ行

RAMディスク …… 27,39
ランダム関数 …… 169
リスト …… 104
論理演算 …… 14
割込み …… 29,95,97,188
割込み処理 …… 29
割込みの優先順位 …… 29

MSX 2 パーソナルコンピュータBASIC説明書

昭和60年11月　第1版　第1刷発行

発　行　　松下電器産業株式会社
情報機器部
〒571 大阪府門真市門真1006
電話（06）908－8801

編　集　　㈱アスキー

無断転載を禁ず

松下電器産業株式会社

情報機器部

〒571 大阪府門真市大字門真1006

電話 (06)908-1151(代表)

TFQF2033ZD

S0987-0