SHARP®

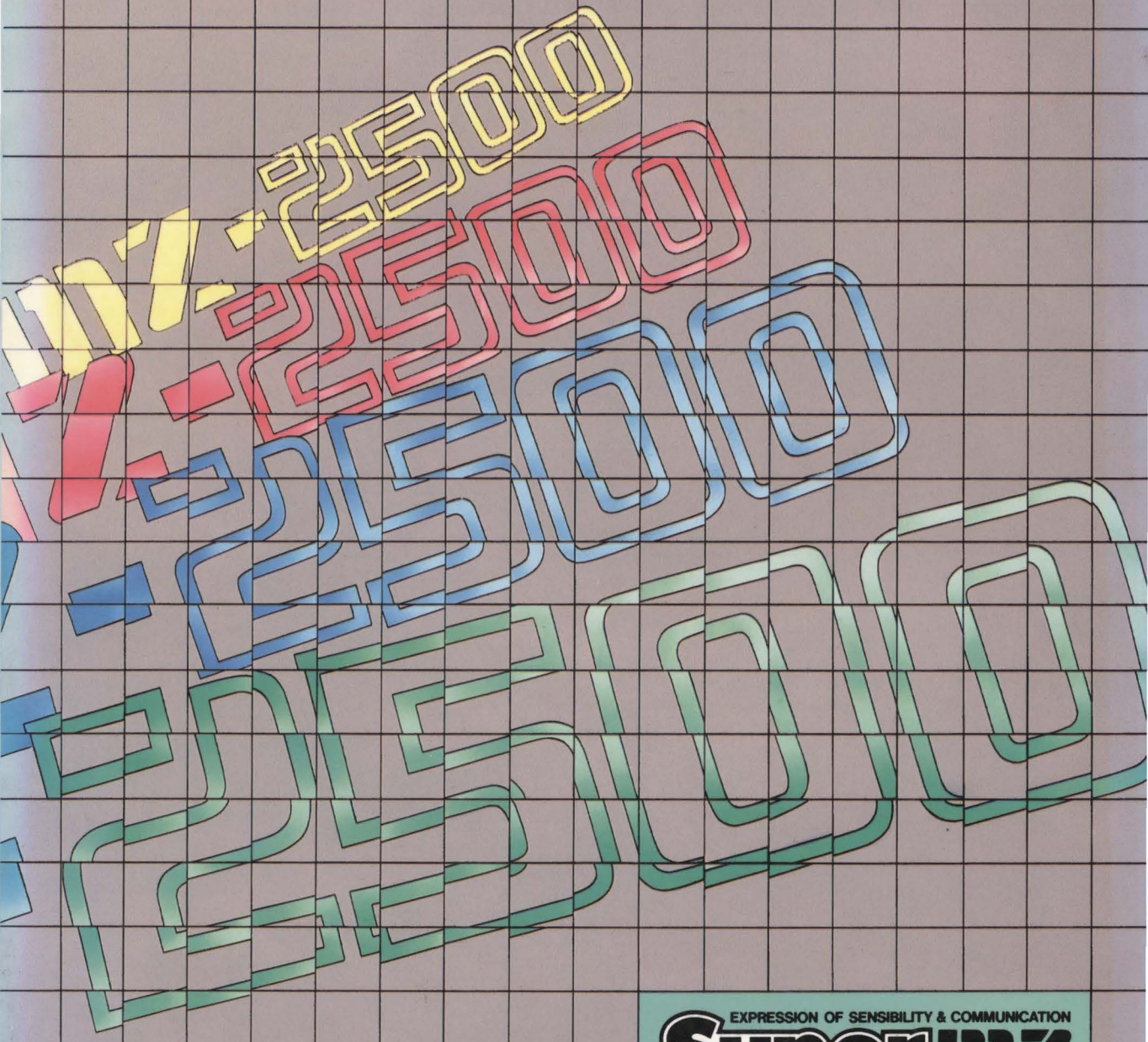
パーソナルコンピュータ

MZ-2500

形名

MZ-2511
MZ-2521

BASIC-S25 マニュアル

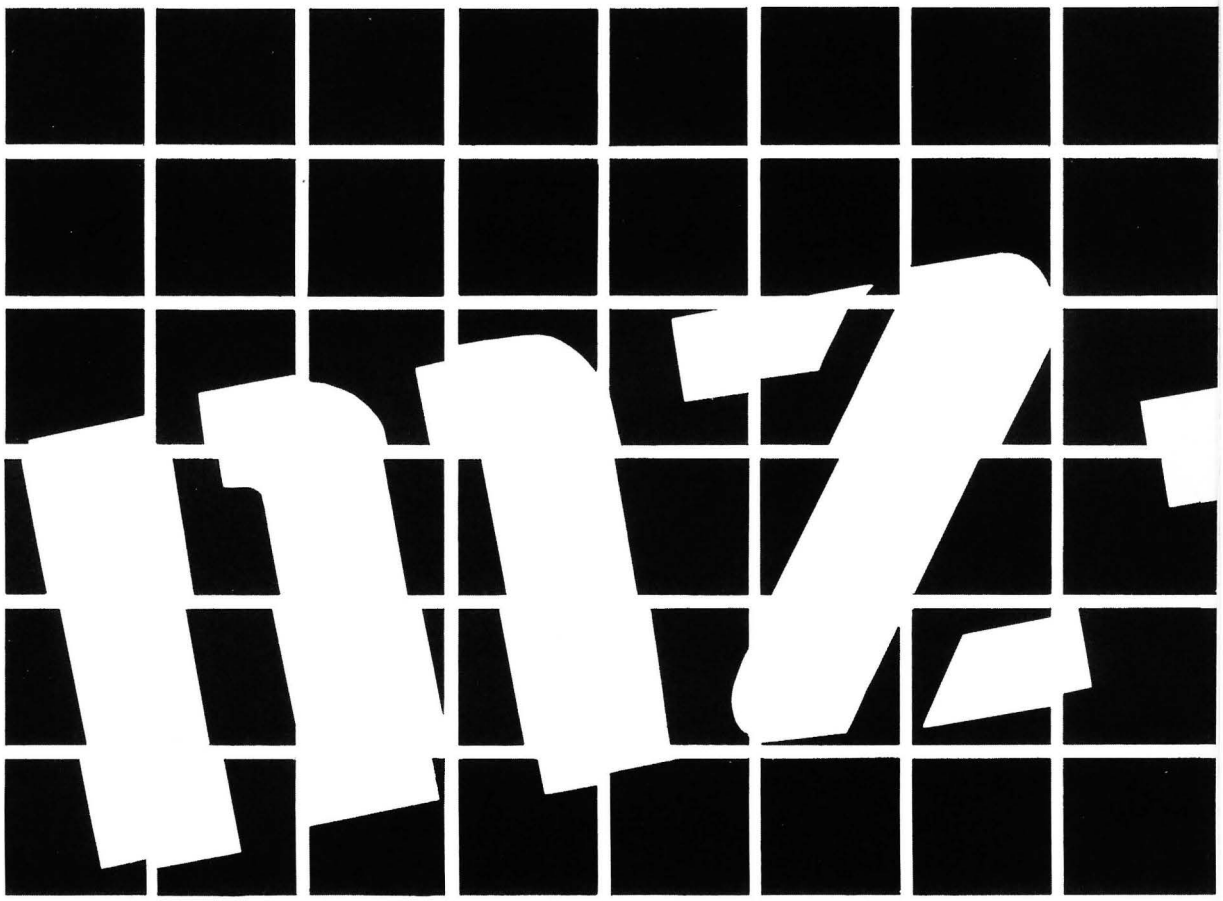


EXPRESSION OF SENSIBILITY & COMMUNICATION

SuperMZ

MZ-2500

BASIC-S25 マニュアル



ご注意

(1) ソフトウェアおよび本書の内容は、改良のため予告なく変更することがありますので、あらかじめご了承ください。

(2) 本書は内容について十分注意し作成していますが、万一ご不審な点、お気付きのことがありましたら、もよりのお客様ご相談窓口までご連絡下さい。

(3) パーソナルコンピュータ本機のBASICソフトウェアは、シャープ株式会社のオリジナルソフトウェアであり、著作権法上の権利は当社が保有しております。

ソフトウェアならびに本書の内容を無断で複製することは禁止します。

(4) 当社は、本ソフトウェアの使用に際して生ずる金銭上の損害および逸失利益などについては、一切の責任を負いかねますのであらかじめご了承ください。

はじめに

シャープパーソナルコンピュータMZ-2500シリーズにはプログラミング言語としてBASIC-M25とBASIC-S25の2つのBASICをもっています。
本書ではこのうちBASIC-S25について解説しています。
BASIC-S25はシャープMZシリーズのMZ-80B、2000、2200などのすぐれたBASICの流れを引き継いでおり、次の構成で書かれています。

第1部　BASICの概要
　ここではBASIC-S25のプログラミングに必要な知識と構造を述べています。

第2部　コマンド・ステートメントの解説
　ここではBASIC-S25で使用できる全コマンドとステートメントの文法や具体的な使用例を詳しく説明しています。

付　録

　以上の3部で構成されています。

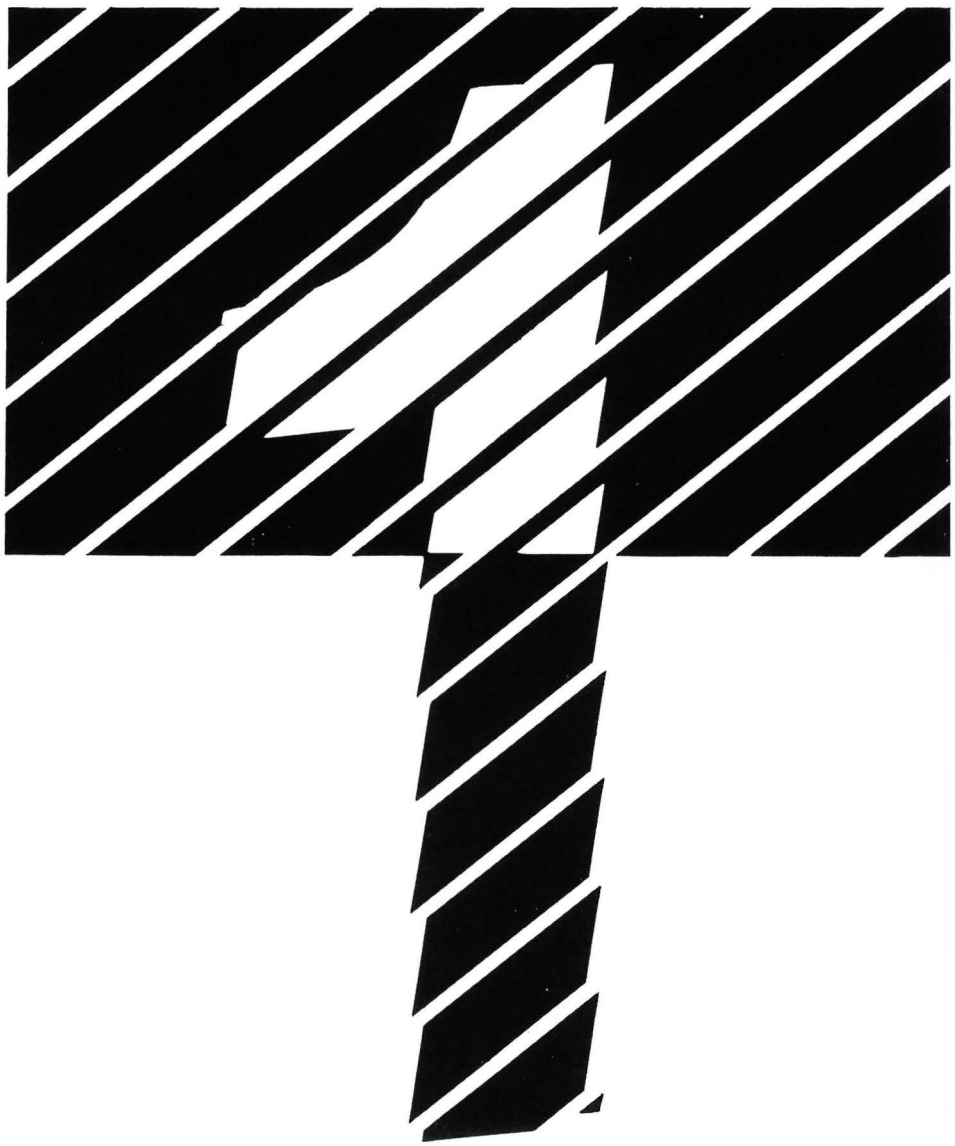
本書を十分にお読みいただき、MZ-2500のすぐれた機能を自在に使いこなしていただけますよう切望します。

目　次

第1部　BASICの概要

第2部　コマンド・ステートメントの解説

第1部
BASICの概要

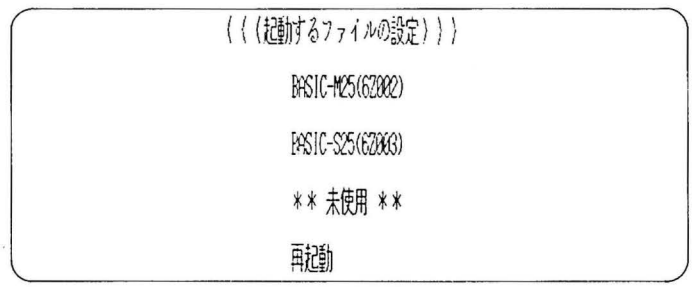


1 BASICの起動

同梱されているBASICシステムディスクは、出荷状態では**BASIC-M25**が起動するようになっています。**BASIC-S25**を起動させる場合は、次の操作を行なってください。

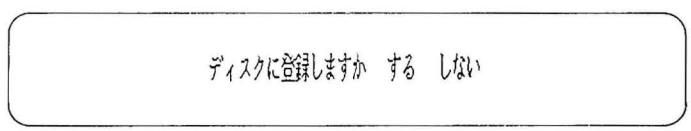
1．本体の電源を切った状態でBASICシステムディスクをドライブ1にセットしてください。

2．[HELP] キーを押し続けたまま、電源を入れてください。[HELP] キーは、次のような表示が出るまで押し続けてください。



3．カーソルの上下移動キーを押し、**BASIC-S25**を選択して [↵] キーを押してください。（選択されたものは、リバース表示になります。）

4．次のような表示が出ますのでカーソル左右移動キーを押し、「しない」を選択して [↵] キーを押してください。



5．以上で**BASIC-S25**が起動します。

4の操作で「する」を指定すると、それ以後自動的に**BASIC-S25**が起動するようにシステムディスクを書き換えます。この操作をされる場合は、安全のため同梱のシステムディスクに直接行なうのではなく、コピー（バックアップ）したディスクで行なってください。

この起動選択操作の詳細は、オーナーズマニュアルの第4章 MZ-2500の起動を参照してください。

"auto-run. s25"の実行は、プログラムの自動起動のための機能です。BASIC起動後に自動実行させたいプログラムがある場合そのプログラムの名前を"auto-run. s25"として、マスターディスクに登録しておくと便利です。

起動を開始してしばらくすると、次のような初期画面が表示されます。自動的にフロッピーディスクから、コンピュータが"auto-run. s25"と名前の付いたプログラムを読み込み、実行します。

画面に"Ready"が表示され、カーソルが点滅して命令の入力待ち状態になります。

起動時の初期設定

起動時の初期状態

BASICの起動時は次の初期化が行なわれます。

- newコマンドの実行により行なわれる初期化。
- 画面の初期化。(init "CRT2:" : init "CRT1:,,,1")
  (文字画面はスムーススクロール状態になっていますので、漢字入力を行なう場合はinit "CRT1:,,,0"としてラインスクロール状態にしてください。)
- 予約語の表示は小文字になります。(option list lcase)

"auto-run.s25"の実行

"auto-run. s25"の実行により、次の初期設定を行ないます。

- ファンクションキーの定義。
  (ファンクションキーラベルはS25のほうを使用してください。)
- new on 2の実行。
  この命令の実行により別売のマウス(MZ-1X10)、ボイスボード(MZ-1M08)、辞書ROMボード(MZ-1R28)を使用した機能を消去しています。これらの機能を使用される場合は、"auto-run. s25"プログラムの行番号990を書き換えてディスクへ再登録して、BASICの再起動を行なってください。
  メモリ標準実装(128キロバイト)の場合、これらの消去された機能を使用するとフリーエリアが少なくなります。詳しくはBASIC-M25マニュアルの付録「BASICのフリーエリア」を参照してください。

2 編集と実行

当BASICを十分に使いこなしていただくためには、基本的な約束ごとを覚えていただかなければなりません。ここでは、BASICの基本仕様から、プログラム作成時の編集操作方法、またその実行方法までを説明しています。

実行モード

BASICが起動されると、"**Ready**"が表示され、命令待ちになります。この状態で命令を実行させる方法として、直接実行モードと間接実行モードがあります。

直接実行モード

ダイレクトモードとも呼ばれ、命令が入力されると直ちに実行されます。コマンドやステートメントのスペル(単語)が入力されると、BASICはこれを解析して実行します。

変数やモードの変化として、実行結果を確かめることはできますが、実行手順を残すことはできません。

間接実行モード

一連の命令を手順を追って実行する場合は、ひとまずメモリ内へ手順どおりに命令を記憶させたのちに、実行させます。この命令が手順どおりに記憶されているものをプログラムと呼び、記憶されているプログラムを実行させるモードを間接実行(またはプログラム)モードと呼びます。

BASICでは実行の順番を示す行番号のあとに命令を書いて入力すると、プログラムとして行番号の順番で記憶されます。**run**コマンドまたは、**goto**文や**gosub**文を直接実行することにより、プログラムの実行を開始します。

命令の種類

BASICの命令は機能的に次の様に分けることができます。

コマンド

プログラムの編集や入出力、実行にかかわる命令で、おもに直接実行モードで実行されます。直接実行モード以外にプログラム中で実行可能な命令もあります。

ステートメント

おもにプログラム中に書かれる命令で、データの入出力や実行手順の制御、グラフィック命令などがあります。一部を除いてほとんどのステートメントが直接実行モードでも使用できます。

関数

関数は、それ自体で何かの実行を行なうものではなく与えられた引数(関数で処理を行なうデータ)により、数値演算や、文字列の加工を行ない結果を返すもので、それ自体が値をもちます。

また、関数の中には、入出力装置に対して情報を入出力するための入出力関数もあります。

システム変数

システム変数は、関数と同様にそれ自身が値をもちます。関数のように引数はありませんが、目的に応じて内容を設定できるものがあります。

システム変数の内容には時計機能やBASICの実行にかかわる情報を値としてもつものなどがあります。

プログラムの書式

行の最初に番号を付けて入力するとプログラムとして扱われます。プログラムの1行の書式は次のようになります。

<行番号>〔<ラベル>：〕<文>〔：<文>〕…

〔 〕内は省略可能なことを示します。

行番号

行番号は、プログラム内の位置を示し1から**65535**までの整数値が使えます。プログラムの各行の最初には、必ず行番号が必要です。

プログラムの各行をどのような順番で入力しても(行番号順でなくても)コンピュータ内部では、行番号の順に記憶されています。

ラベル

ラベルは**goto, gosub**文や編集命令などで行指定をするときに行番号の変わりに名前(ラベル名)で指定するものです。

ラベル名は、最初の文字をアスタリスク(*)にして、2文字目は半角のアルファベットやカタカナまたは漢字(文字コードが**&H889F**以降の全角文字)で指定します。3文字目以降は全角文字(文字コードが**&H82F4**以降のもの)または半角文字のアルファベット、数字、アンダーバー(_)などが使え最初のアスタリスクを除いて半角文字相当で200文字までの任意の長さまで指定できます。また、予約語のスペルをラベルの中に使うこともできます。

ラベルを入れる位置は行番号の次で、あとに文が続く場合はコロン(：)で句切ります。ラベルを文の途中に入れることはできません。

行の長さ

画面上の表示では1行が40または80文字ですが、BASICでは論理的な行として、行番号を含めて最大**255**文字まで可能です。(画面上では複数の行にわたって表示します。)

文

文はコマンドまたはステートメントで構成され、一行の中には必ず文を必要とします。ただし、ラベルのみの行にすることは可能です。また、マルチステートメントとして、文と文をコロン(：)で句切ることにより、同一行に複数の文を続けて書くことができます。

プログラム中のスペース

プログラム中でスペース(空白)を使用する場合は、いくつかの注意しなければいけない点があります。

予約語に続いて半角文字の数字、アルファベットやカタカナまたは全角文字がくる場合は、変数名と区別できなくなるので、必ず1個以上のスペースをあけて入力してください。ただし、いくつのスペースをあけてもBASIC内部では、1個のスペースとして扱われます。(リスト表示では1個分のスペースになります。)これは、予約語の前(左側)についても同様のことが起こります。

演算記号、比較記号、句切記号などの左右にスペースを入れた場合は、BASICでは詰めて扱われます。

行番号に続いて、スペースを入れずに文を書いた場合は、BASICでは1個のスペースを行番号の次に入れます。

次のような場合はスペースが入力したとおりの個数として扱われます。

- 行番号から文が始まるまでの間。
- マルチステートメントの句切記号であるコロン(:)の左右。
- rem文およびアポストロフィ(')以後のコメント部分。
- data文のデータ部分。
- ダブルクォーテーション(")で囲まれた文字定数。

行番号は右づめ表示をしますので、5桁に満たない左側の部分はスペースになります。

例) 入力とlistコマンドによる出力

```
10for   a=  1  to  200  :  next
list 10
   10 for A=1 to 200  :  next
                         └──────そのままの個数


200   if a$ = "a A"then   99     elseprint a
list 200
  200   if A$="a A" then 99 ELSEPRINT A
                              └──────予約語になりません
```

BASICで使用する文字

全角文字と半角文字

BASICで使用する文字はBASIC内部では、文字コードで扱われ、1バイトのコードで表される半角文字と、2バイトのコードで表される全角文字があります。表示上では全角文字の幅は半角文字の倍の幅があります。

半角文字 ┬ アルファベット・数字・英記号
　　├ カタカナ・カナ記号
　　└ グラフィック文字

全角文字 ┬ 漢字
　　└ ひらがな・カタカナ・記号・英数字

BASIC上の命令(予約語)は半角文字の英数字、記号で扱われ、全角文字の英数字、記号で書くことはできません。

アルファベットの大文字と小文字

BASICの予約語や変数名、配列名、定義関数名、ラベル名で、半角文字のアルファベットを使用した場合、大文字と小文字は同じ物として扱われ、入力時は、どちらを用いてもかまいません。

ただし、**list**コマンドなどによってプログラムを出力する場合は大文字と小文字のいずれかに統一されます。

予約語は、大文字で出力するか、小文字で出力するかの選択を、**option list**文で行なうことができ、最初は小文字で出力するようになっています。

変数名、配列名、定義関数名、ラベル名では、半角文字のアルファベットはすべて大文字で出力します。

特別な記号

BASICは半角文字の記号に特別な意味をもたせているものがあります。

	空白	原則として無視されますが、予約語の前後に句切りとして必要な場合があります。
＝	イコール	代入、等号
＋	プラス	加算、正符号、文字列の結合
－	マイナス	減算、負符号、行の範囲指定(ハイフンとして)
＊	アスタリスク	乗算、ラベルの接頭文字
／	スラッシュ	除算
＾	山記号	べき乗算
￥	円記号	整数除算
％	パーセント	整数型
！	感嘆符	単精度実数型
＃	シャープ記号	倍精度実数型
＄	ダラー	文字列型、16進数表記の接頭文字
＆	アンパサンド	2,8,16進数表記の接頭文字
’	アポストロフィ	注釈文
？	クエスチョン	**print**(予約語)の代用
，	カンマ	データの句切り
；	セミコロン	データの句切り
”	ダブルクォーテーション	文字定数の引用符
．	ピリオド	小数点、行番号(実行中断時)、予約語の省略
：	コロン	マルチステートメントの句切り
＿	アンダーバー	変数名、ラベル名の中に使用可
（）	括弧	演算時に括弧内を優先
＜＞	大小記号	比較演算子

予約語と省略形

予約語

予約語とは、BASICでコマンドやステートメント、関数として使用される単語のことです。変数名や配列名に予約語を使うことはできません。(ただし、予約語のスペルを含むものは可能です。)

予約語は半角文字の英数字、記号で表します。入力時は大文字、小文字のいずれでも可能ですが、**list**コマンドなどによる出力時は、大文字、小文字のいずれかに統一されます。リスト表示の予約語の大文字と小文字の選択は、**option list**文によって行なわれ、BASICでの初期状態は小文字になっています。

予約語に続いて半角文字の英数字やカタカナ、アンダーバー(_)、全角文字がある場合には変数名と区別できません。このような場合は、予約語のあとにスペースを挿入してください。

予約語を書くときは、1つの予約語の途中にスペースを入れることはできません。

省略形

BASIC上でよく使用する予約語や、長い単語の予約語に対して、すべてのスペルを書かずに省略できる機能があります。

省略形は予約語のスペルの一部分を書き、省略していることを表すためにピリオド(.)を付けます。

たとえば**print**の省略形は**p.**となります。

プログラム入力時に省略形で入力してもリスト表示を行なうと、入力された予約語がフルスペル(省略しない形)に直されます。

省略は、最も短かい省略可能な形だけでなくピリオド(.)を付けることにより省略範囲を広げることもできます。

たとえば**list**の最短の省略形は**l.**ですが、**li.**や**lis.**も可能です。

次に予約語とその最短の省略形の表を示します。

予約語	省略形
<a>	
abs	ab.
akcnv$	ak.
all	al.
and	an.
angle	ang.
aopen	ao.
apss	ap.
as	as
asc	asc
ascchr$	asc.
atn	at.
attr$	att.
auto	a.
<b>	
base	ba.
beep	b.
bin$	bi.
bline	bl.
boot	boo.
box	bo.
<c>	
call	ca.
cblock	cb.
ccolor	cc.
cdbl	cd.
cflash	cf.
cgen	cg.
cgpat$	cgp.
chain	cha.
character$	char.
chdir	ch.
chr$	chr$
cint	cin.
circle	ci.
click	cli.
close	clo.
clr	clr
cls	cl.
cmt	cm.
color	col.
com	com
common	com.
console	cons.
cont	c.
copy	cop.
cos	cos
crev	cr.
csng	cs.
csrh	csr.
csrv	csrv
cursor	cu.
cvd	cvd
cvi	cv.
cvs	cvs
<d>	
data	da.
date$	date$
day$	day$
dbl	db.
def	def
deg	deg
degrees	deg.
delete	d.
devf	devf
devi$	dev.
devo$	devo$
dim	di.
dir	dir
dtl	dt.
<e>	
edit	e.
else	el.
end	en.
eof	eo.
eqv	eq.
erase	er.
erl	erl
ern	ern
error	err.
exp	ex.
<f>	
fac	fa.
filed	fie.
fill	fi.
fix	fix
for	f.
fpos	fpo.
fpset	fp.
frac	fr.
<g>	
get	ge.
gload	gl.
go sub	go s.
go to	go .
gosub	gos.
goto	g.
gsave	gs.
<h>	
hcopy	h.
help	hel.
hex$	he.
hexchr$	hex.
<i>	
if	if
imp	im.
init	ini.
inkey$	ink.
inp@	inp@
input	i.
input$	input$
instr	ins.
int	int
interval	inte.
<j>	
jis$	j.
<k>	
kacnv$	ka.
kanji	kan.
key	k.
kill	ki.
klen	kle.
klist	kl.
kmode	km.
kpos	kp.
ktn$	kt.
<l>	
lcase	lc.
left$	lef.
len	len
let	le.
lfo	lf.
limit	lim.
line	lin.
list	l.
ln	ln
load	lo.
loc	loc
lock	loc.
lof	lof
log	log
lpos	lpos
lpout	lp.
lset	ls.
<m>	
map	map
max	ma.
merge	m.
mid$	mid$
mirror$	mi.
mkd$	mkd$
mkdir	mk.
mki$	mki$
mks$	mks$
mod	mod
mon	mo.
mouse	mou.
move	mov.
music	mu.

予約語	省略形	予約語	省略形	予約語	省略形
<n>		return	re.	time	tim.
new	new	right$	ri.	to	to
next	n.	rmdir	rm.	tone	ton.
not	no.	rnd	rn.	troff	trof.
		roll	rol.	tron	t.
<o>		ropen	ro.		
oct$	oc.	rset	rs.	<u>	
off	of.	run	r.	ucase	uc.
on	o.			unlock	unl.
option	op.	<s>		until	u.
or	or	save	sa.	using	us.
out@	ou.	scr	scr		
		screen	sc.	<v>	
<p>		search	sea.	val	va.
pai	pai	set	se.	verify	ve.
paint	pa.	sgn	sg.	view	vi.
pattern	pat.	sin	si.	voice	v.
peek	pee.	size	siz.		
pen	pe.	sng	sn.	<w>	
point	poi.	sound	so.	wait	w.
poke	po.	space$	spa.	wend	we.
poly	pol.	spc	sp.	while	wh.
posh	posh	sqr	sq.	width	wi.
position	pos.	step	ste.	window	win.
posv	posv	stick	sti.	wopen	wo.
print	p. ?	stop	s.		
priority	prio.	str	str	<x>	
put	pu.	str$	str$	xopen	x.
pwd$	pw.	strig	str.	xor	xor
		string$	strin.		
<r>		sum	su.		
rad	rad	swap	sw.		
radians	rad.	symbol	sy.		
randomize	ra.	sysid$	sys.		
read	rea.				
rem	rem	<t>			
rename	rena.	tab	ta.		
renum	ren.	tan	tan		
repeat	rep.	tel	tel		
replace	repl.	tempo	te.		
reset	rese.	term	ter.		
restore	res.	then	th.		
resume	resu.	ti$	ti$		

プログラムの編集

プログラムはつねにメモリ上に記憶されていて、1行単位で管理されます。ここではプログラムの入力、変更などの基本的な操作について説明します。

プログラムの入力

行番号の付いた1行が入力されるとプログラムの1行として記憶されます。続いて別の行番号を付けて入力すると、行番号を比較して、番号順に並べて記憶します。こうして異なる行番号を付けることにより、複数の行を記憶することができ、行番号の順番が実行する順番になります。

続けてプログラムを入力する場合に、自動的に行番号を発生させるautoコマンドがあります。また、入力されたプログラムを確認するためにlistコマンドが用意されています。

プログラムの変更

すでに記憶されているプログラムの変更も行単位で行なわれます。すでに記憶されているプログラムと同じ行番号で入力すると、以前に記憶されている方の行が消去され、新たに入力された行の内容になります。

変更する内容がその行の一部分の場合は、あらかじめ目的の行を画面に表示させ、カーソル移動により表示を書き換え、[↵] キーを入力することにより変更ができます。この機能をスクリーンエディット機能と呼びます。

プログラム全体をスクリーンエディットする命令として、editコマンドが用意されています。

プログラムの消去

特定の行を消去するときは、その行番号のみを入力します。入力が行なわれるとすでに記憶されているプログラム中の指定行番号の内容が消去されます。

行番号の範囲を指定して消去する命令として、deleteコマンドがあります。また、すべてのプログラムを消去する命令としてnewコマンドが用意されています。

ピリオド(.)の特別機能

直接実行モードでlistやeditなどの行番号を指定する編集コマンドにおいて、行番号の指定にピリオド(.)を使うことができます。ピリオドは現在BASICが注目している行として次のような場合に行番号の設定が行なわれます。

- プログラムの実行中に、エラーが発生して命令待ちに戻ったときのエラー発生行。
- stop文の実行および SHIFT ＋ BREAK キーの操作などによりプログラムの実行を一時中断して命令待ち状態に戻ったときに実行していた行。
- プログラムの編集が行なわれた場合の最後の入力行。
  このような場合list .とすると目的の行を確認できます。

プログラムの実行

メモリ上に記憶されているプログラムの実行はrunコマンドで行ないます。プログラムの実行は行番号の順に行なわれますが、go to文、gosub文、if～then文などの実行により順番を変えたり、for～next文、repeat～until文、while～wend文などによって繰り返し実行(ループ)させることもできます。

実行が終了されるのはend文を実行したとき、またはプログラムの最後で次に実行する行がなくなった場合です。また、SHIFT ＋ BREAK キーの操作やstop文による中断の場合はcontコマンドによってプログラムの実行を再開することができます。

プログラムの実行中に BREAK キーを押すことにより、実行を一時停止させることができます。一時停止の状態より実行を再開させる場合は、任意の文字キーを押します。

プログラムの実行状態を確認するためのtronコマンドやブレークポイントを設定するstopコマンド、またステップ実行をするためのstepコマンドなどのデバッグコマンドも用意されています。

プログラムが中断され、contコマンドやstepコマンドで中断した所からのプログラムの再開が可能なときは、Readyのあとにピリオド(.)を表示します。

プログラムの中にある誤りをバグ(BUG・虫)といいます。このバグを発見して修正する作業をデバッグと呼びます。

編集のためのキー操作

基本的な文字入力はオーナーズマニュアルを参照していただき、ここでは編集のための CTRL キーや SHIFT キーを使った操作を中心に説明します。

カーソルの移動

↑ ↓ → ←　カーソル移動キーを押すことにより、矢印の方向にカーソルが１文字分移動します。

CTRL ＋ ^　↑ キーと同じです。

CTRL ＋ _　↓ キーと同じです。

CTRL ＋ ¥　→ キーと同じです。

CTRL ＋]　← キーと同じです。

CTRL ＋ F
SHIFT ＋ →　右方向へ単語の先頭まで移動します。

CTRL ＋ B
SHIFT ＋ ←　左方向へ単語の先頭まで移動します。

単語移動は英数字の並びを1つの単語と見て、半角文字のスペースや記号が句切りとみなされます。

CLR/HOME
CTRL ＋ K　画面の左上(カーソルホームポジション)に移動します。

CTRL ＋ →　行末へ移動します。

CTRL ＋ ←　行頭へ移動します。

CTRL ＋ J　カーソル位置で行を分けます。カーソル位置から右側の文字を次の行へ送り出します。カーソル位置より左側の文字は、↵ キーを押した場合と同じように入力されます。

CTRL ＋ ↑　画面の左上に移動します。CLR/HOME キーと同じです。

SHIFT ＋ ↑　両面の左上に移動します。(editコマンドによるエディットモード時は画面半ページをスクロールダウンします。)

CTRL ＋ ↓　画面の左下に移動します。

SHIFT ＋ ↓　画面の左下に移動します。(エディットモード時は画面半ページをスクロールアップします。)

TAB　現在のカーソル位置の右側のタブ位置に移動します。タブの設定は CTRL ＋ T キーで行ないます。
BASICの起動時は8文字ごとの設定になっています。

CTRL ＋ I　TAB キーと同じです。

タブの設定と解除

CTRL ＋ T　現在のカーソル位置にタブを設定します。

CTRL ＋ Y　現在のカーソル位置にタブが設定されている場合、設定を解除します。

文字の挿入

SHIFT ＋ INST/DEL キーまたは CTRL ＋ R キーで、カーソル位置より行末までの文字を１文字分右にずらし、カーソル位置に空白を挿入します。

CTRL ＋ SHIFT ＋ INST/DEL キーまたは CTRL ＋ A キーで、インサートモードになりカーソルの点滅が速くなります。このときに文字を入力すると、カーソル位置に文字を挿入します。 ↵ キーを押して入力を完了するか、カーソル移動キーを押すことにより、カーソルの点滅が元に戻り、モードの解除が行なわれます。

インサートモードのときに、次のキーが特別な意味をもちます。

CTRL ＋ TAB　カーソル位置よりタブ位置まで空白を挿入します。

挿入処理において文字が行末まで詰っているときは、次の行以降をスクロールダウンさせます。

文字の削除

INST/DEL	カーソル位置の１文字を削除します。 カーソルの右側より行末までの文字は左に１文字分移動します。
⇦	カーソルの左側の１文字を削除します。 カーソル位置より行末までの文字は左に１文字分移動します。
CTRL ＋ H	
CTRL ＋ ⇦	行頭よりカーソル位置の左側の文字を削除します。 カーソル位置より行末までの文字は行頭へ移動します。

文字の消去

CTRL ＋ INST/DEL	カーソル位置より行末までを消去します。
CTRL ＋ E	
CTRL ＋ X	カーソルのある行をすべて消去します。カーソル位置は、行頭になります。
CTRL ＋ Z	カーソル位置より画面の終わり(右下)までを消去します。
SHIFT ＋ CLR/HOME	全画面を消去してカーソルを左上に移動します。
CTRL ＋ L	
CTRL ＋ TAB	カーソル位置より右側のタブ位置までを消去します。操作後、カーソルはタブ位置に移動します。

行の挿入

CTRL ＋ N	現在カーソルのある行から上の行をスクロールアップします。
CTRL ＋ O	現在カーソルのある行から下の行をスクロールダウンします。

いずれもカーソルのある行は空白が挿入され、カーソル位置は変化しません。

文字のコピー機能

文字の入力中に画面上の別の位置に表示されている文字を、現在の入力位置にコピーすることができます。

COPY キーを押すと現在のカーソル位置に点滅をしないカーソルを残し、点滅するカーソルをカーソル移動キーで移動させることができます。点滅するカーソルをコピーしたい文字の位置に合わせて ↵ キーを押すと元の位置に文字がコピーされ、両方のカーソルが右に移動して、次の位置に進みます。続けて ↵ キーを入力するごとに1文字ずつコピーされ、再度 COPY キーを押すか、あるいはカーソル移動や ↵ キー以外のキーを押すことにより、この機能が解除され、元の位置にカーソルが点滅して通常の状態に戻ります。また、行末までコピーを行なうことによっても自動的に解除を行ないます。

そのほかの制御操作

CTRL ＋ W　次の行と接続して論理的につながった行とします。

SHIFT ＋ BREAK　プログラムの実行を中止します。

BREAK　プログラム実行を一時停止させます。任意の文字キーを押すと実行を再開します。

CTRL ＋ G　ベル音を鳴らします。

CTRL ＋ SHIFT ＋ CLR/HOME　先行入力のキーバッファを消去します。

CTRL ＋ M　↵ キーと同じ動作をします。

CTRL ＋ D　画面表示の初期化や音楽演奏の中止を行ないます。

（初期化される命令）

console　console@　window　view　view@
screen　color　ccolor　cgen　crev
cflash　priority　cblock　color=

3 BASICで扱うデータ

BASICで扱われるデータは数値型データと文字列型(ストリング)データがあり、数値型データには整数型,単精度実数型,倍精度実数型の種類があります。

・整数型データ

-32768から32767の範囲の整数で、内部では、2バイトで16ビットの符号付き2進数で扱われています。

・単精度実数型データ

0および$\pm 2.9387359 \times 10^{-39}$から$\pm 1.7014118 \times 10^{38}$までの範囲の、有効桁8桁の浮動小数点方式です。内部では、5バイトのデータとして扱われています。

以後、単精度型と略して呼ぶ場合があります。

・倍精度実数型データ

0および$\pm 2.938735877055719 \times 10^{-293}$から$\pm 1.701411834604692 \times 10^{308}$までの範囲の、有効桁16桁の浮動小数点方式です。内部では、8バイトのデータとして扱われています。

以後、倍精度型と略して呼ぶ場合があります。

・文字列型データ

0から255文字までの長さの文字の並んだものとして扱われます。内部では、半角文字は1バイト、全角文字は2バイトの文字コードの並びとして扱われ、最大の長さは255バイトです。

注) ここで表現しているデータのバイト数は、データのみの数です。実際に変数や配列に代入した場合は、変数名や配列名の管理用のメモリを必要とします。

定数

定数とは、プログラム中で直接書かれるデータのことです。

整数型

数値のあとに%記号を付けます。

入力時-32768から32767までの定数は自動的に整数型とみなされます。

例）１２３４５％　－１２３％　１１５６　－１５００

整数型は10進数のほかに2、8、16進数での表現ができます。

2進数　……　&Bを数の先頭に付けます。

&B 0～&B 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

8進数　……　&Oを数の先頭に付けます。

&O 0～&O 1 7 7 7 7 7

16進数　……　&Hを数の先頭に付けます。

&H 0～&H F F F F

最上位ビットは符号になりますので1の場合は負の値になります。たとえば、16進数の&H8000～&HFFFFは-32768～-1になります。

通常の表示は10進数で行なわれますが、BIN$, OCT$, HEX$の関数を使用すると、それぞれ2、8、16進数で表すことができます。

漢字対応の特別な定数として次のものがあります。

いずれも内部ではシフトJISに変換して扱われます。

JISコード　……　&Jを先頭に付けて4桁の16進数で表します。

&J2121～&J7E7E　（範囲をこえた場合はエラーになります。）

例）print　chr$(&J3021),hex$(&J3021)

亜　889F　……　JISの&H3021はシフトJISでは&H889Fになります。

区点コード　……　&Kを先頭に付けて区と点をそれぞれ2桁の10進数で表します。

&K0101～&K9494　（範囲をこえた場合エラーになります。）

例）print　chr$(&K1601),hex$(&K1061)

亜　889F　……　16区、1点はシフトJISでは&H889Fになります。

単精度実数型

数のあとに！記号を付けます。

整数型の範囲をこえる8桁以内の整数、または小数点付きで有効桁が8桁以内の数は、自動的に単精度実数型とみなされます。

指数記号は、Eが用いられます。

例）１２３！　１２３４５６７　１０．５

１４３Ｅ＋０３　……　$143\times10^{3}=143000$

１１２８Ｅ－０５　……　$1128\times10^{-5}=0.01128$

倍精度実数型

数のあとに＃記号を付けます。

有効桁が9桁以上の数は、自動的に倍精度実数型とみなされます。

指数記号は、Dが用いられます。

例）１２３＃　１２３４５６７８９　１．２３Ｄ＋１２

文字列型

引用符としてダブルクォーテーション(")記号で囲みます。

例）"１２３"　"ＢＡＳＩＣ"　""

"記号を続けて書くと文字長が０のＮｕｌｌ(空文字)文字列になります。(Ｎｕｌｌ文字列＝ナルストリング)

変数

メモリに名前を付けて、名前によりデータを記憶したり、呼び出すことができます。これを変数と呼びます。

変数名

変数を区別するための名前が変数名になります。名前は、最初の文字を半角文字のアルファベットやカタカナまたは漢字(文字コードが**&H889F**以降の全角文字)で指定します。2文字目以降は半角文字のアルファベットやカタカナ、数字、アンダーバー(_)および全角文字(文字コードが**&H82F4**以降のもの)を使って指定でき、最大半角文字相当で200文字までの任意の長さで指定できます。また、最後に型を示す記号(**%, !, #, $**)を付けます。

予約語(コマンド、ステートメントのスペル)と同じ変数名は使用できませんが、予約語のスペルを含むものは使用することができます。

例) ＡＢＣＤ％　ＡＢＣＤ！　ＡＢＣＤ＃　ＡＢＣＤ$

…… 同じ名前を使用していますが型が異なりますので別のものとして扱われます。

変数の型

通常は変数名で型指定を省略した場合、変数名が半角のアルファベットで始まるものは単精度実数型、変数名が漢字または半角文字のカタカナで始まるものは倍精度実数型として扱われます。データの型を指定する場合は次の記号を付けます。

%	整数型
!	単精度実数型
#	倍精度実数型
$	文字列型

変数名が半角文字のアルファベットで始まる変数の型指定を省略したとき、自動的に決定される型を指定する命令として、**def int**、**def sng**、**def dbl**、**def str**の命令が用意されています。

配列

変数データをまとめて使用するとき、グループ全体を１つの名前で呼び、個々の要素の指定を番号で行なうことができます。これを配列と呼んでいます。

配列変数

グループ全体の名前が配列名で、変数名と同じ付け方ができ、個々の要素の指定は配列名のあとに番号をカッコ( )で囲んで書きます。このカッコでの指定を添字と言い、配列変数を添字付き変数と呼ぶことがあります。

例）A(10)　ABCD$(5)　価格＃(2)

要素の指定を行なう番号は変数や数式を使うこともできます。そのため、すべての要素を順番に参照するようなときでも、要素を指定する変数をループなどで変化させることにより簡単に行なうことができます。

配列宣言

配列を使用する場合は、あらかじめ配列名と最大要素番号を指定する必要があります。dim文がそれで、配列宣言と呼んでいます。配列宣言を行なうことによりBASICは、メモリ上にエリアを確保します。

例）ｄｉｍ　Ａ(１００),ＮＡＭＥ$(１００)

通常、配列宣言を行なうと要素番号は0から最大要素までとなり、最大要素が100であれば要素数は101個になります。要素の開始を１からに変更する命令としてoption base文が用意されています。

配列宣言を行なわずに配列要素を参照した場合、自動的に10までの配列宣言が行なわれたものとみなされます。宣言せずに11番以上の要素を参照したり、一度宣言されている配列を再度、宣言することはできません。

宣言された配列を消去する命令としてerase文が用意されています。

多次元配列

添字が１個の配列を１次元配列、２個の配列を２次元配列、３個のものを３次元配列と呼び、多次元配列が使えます。

例）`dim AAA(10,10),HYOU(10,200)`

……２次元配列

`dim X$(3,3,3)`

……３次元配列

２次元配列で添字を(10,20)と指定してoption base 0の場合、0から10までと0から20までの組み合わせになるので、要素の合計は、11×21で231になります。

次元の最大は255次元まで、各添字の最大は32767まで指定できます。ただしプログラムの１行の長さやメモリ容量の制約があるため、実際には指定できないことがあります。

１つの配列でメモリ使用量が65535バイトをこえる指定はできません。

式と演算子

式

結果が数値になる式を数式、結果が文字になる式を文字列式と呼びます。式は、演算される定数や変数などの項と、四則演算記号などの演算子により構成されます。単に定数や変数のみでも演算子の省略された式(単項式)としてみることができます。

算術演算

算術演算の優先順位は、( )内が最も優先され、以下は表の順番になります。2番目の負符号は単項演算子で、符号の前が項でない場合にあとに続く項の値を負にします。

¥の整数除算は、¥の左右の項を整数型に変換したあとに除算を行ない、結果の小数点以下を四捨五入します。

modの剰余演算は、整数除算の余りを求めます。

¥,modのいずれも左右の項の値が整数型の範囲をこえる場合にエラーになるので注意が必要です。

算術演算子は次のとおりです。

演算子	優先順位	意味	
^	1	べき乗算	例) 2^3 …2^3
−	2	負符号	例) −A 5*−3
*	3	乗算	
/	3	除算	
¥	4	整数除算	
mod	5	剰余演算	
+	6	加算	
−	6	減算	例) A−B 3−5

比較演算

比較演算は、数値や文字の大きさを比較して、比較演算子と合えば-1になり、合わなければ0の整数型の値になります。整数型の-1は16ビットの2進数ですべての桁が1に、0はすべての桁が0になり、論理演算と組み合わせて複雑な判断に用いられます。

演算子	意味
＝	等しい
<> ><	等しくない
<	より小さい
>	より大きい
<= =<	より小さいか等しい
>= =>	より大きいか等しい

等号は､代入のときにも使用される記号ですから注意が必要です。

比較は数値または、文字列どうしで行なわれ、数値と文字列の比較はできません。

比較演算の優先順位は算術演算の次になり比較演算どうしでは、順位が同じです。

文字列の比較は、最初から1文字目どうし、2文字目どうしと順番に行ない、すべての文字が同じであれば等しいとみなします。途中で文字が異なっていれば、その時点で文字コードの大きな方の文字列が大きいとみなします。文字列の長さが異なり、短いほうの文字が終了するまで同じであれば、長い方の文字列が大きいとみなされます。

例）"ＡＢＣＤ" ＝ "ＡＢＣＤ"
　　"ＡＢＣＤ" ＜ "ＡＢＣＥ"
　　"ＡＢＣＤ" ＞ "ＡＢＣ"
　　"ＡＢＣ　" ＞ "ＡＢＣ"
　　"" ＜ "Ａ"

文字長が0の文字列、Null文字列(空文字)は、最も小さな文字列となります。

空白(スペース)にも文字コードがありますので文字列の中に空白があれば､1つの文字として扱われ比較を行ないますので注意が必要です。

論理演算

論理演算は整数型で行なわれ、16ビットの2進数の各ビットどうしでブール代数の演算をします。※1コラム

そのとき、目的の項を整数型に変換するため、値が整数型の範囲をこえる場合エラーとなりますので注意が必要です。

演算子	順位	意味
not	1	否定
and	2	論理積
or	3	論理和
xor	4	排他的論理和(exclusive or)
imp	5	包含(implication)
eqv	6	同値(equivalence)

notは論理演算中、最も優先順位が高く、単項演算子になります。notに続く項のビットを反転させます。

X	not X
1	0
0	1

X	Y	X xor Y
1	1	0
1	0	1
0	1	1
0	0	0

X	Y	X and Y
1	1	1
1	0	0
0	1	0
0	0	0

X	Y	X imp Y
1	1	1
1	0	0
0	1	1
0	0	1

X	Y	X or Y
1	1	1
1	0	1
0	1	1
0	0	0

X	Y	X eqv Y
1	1	1
1	0	0
0	1	0
0	0	1

ブール代数の演算
数学の『ブール代数』の規則に従う演算方法で、演算の結果が2つの値のうちの1つをとる演算です。2つの値は0と1などで表したりします。

関数

BASICの組み込み関数として、平方根や三角関数などの数値関数、文字列処理関数、入出力関数などが用意されています。

数値関数

数値関数として、次のものが用意されています。

種　類	書　式	説　　明
絶対値	abs(x)	数式xの絶対値(｜x｜)を求めます。
符号	sgn(x)	数式xの符号(正負)を調べ次の値を返します。 x＞0　…　1 x＝0　…　0 x＜0　…－1
整数化	int(x)	数式xの値をこえない最大の整数を求めます。 例）A=int(2.5):B=int(-2.5) …変数Aには2、変数Bには－3を代入します。
	fix(x)	数式のxの小数点以下を切りすて、整数部を求めます。 例）A=fix(2.5):B=fix(-2.5) …整数Aには2、変数Bには－2を代入します。
小数部	frac(x)	数式xの整数部を取り去り、小数点以下の値を求めます。 例）A=frac(2.5):B=frac(-2.5) …変数Aには0.5、変数Bには－0.5を代入します。
三角関数	sin(x)	数式xを角度として正弦(sin x)を求めます。
	cos(x)	数式xを角度として余弦(cos x)を求めます。
	tan(x)	数式xを角度として正接(tan x)を求めます。
	atn(x)	数式xを正接として角度を求めます。(arctan x)
平方根	sqr(x)	数式xの平方根($\sqrt{X}$)を求めます。
指数関数	exp(x)	数式xの指数関数(e^x)を求めます。
対数	ln(x)	数式xの自然対数(log eX)を求めます。
常用対数	log(x)	数式xの常用対数($\log_{10}X$)を求めます。
円周率	pai(x)	円周率のx倍を求めます。
角度変換	rad(x)	数式xを度としてラジアンに変換した値を求めます。
	deg(x)	数式xをラジアンとして度に変換した値を求めます。
階乗	fac(x)	数式xの階乗(x！)を求めます。
型変換	cint(x)	数式xを整数型に変換した値を求めます。
	csng(x)	数式xを単精度実数型に変換した値を求めます。
	cdbl(x)	数式xを倍精度実数型に変換した値を求めます。
乱数発生	rnd(x)	0よりも大きく、1よりも小さい値の一様乱数を求めます。 数式xの値により次のようになります。 x＞0　…新しい乱数値を求めます。 x＝0　…直前と同じ乱数を求めます。 x＜0　…初期化を行ないます。

- 表の中のxは引数(関数演算をさせる値)で、定数や変数、関数などを含めた数式で指定します。引数による結果がその関数の値となりますので、関数も、数式の一部であると考えられます。
- 三角関数(sin, cos, tan, atn)の角度単位は、度とラジアンのいずれかになります。角度単位は、option angle文で切り換えることができ、最初(BASICの起動時やrun, clrコマンドなどの実行時)は、ラジアン単位になります。
- 乱数を発生させるrnd関数は、範囲内の一様乱数を発生させます。発生する乱数系列は、randomize文により選択することができます。また、引数を負にすることにより、その乱数列の初期状態から発生させますので、再現性をもたせた、乱数発生が可能になります。

文字列処理関数

文字列処理関数として、次のものが用意されています。

関数名（書　式）	説　　明
left$(x$, n)	x$の左からnバイト分の文字列を取り出します。 A$=left$("ABCD", 2) … A$には、"AB"が代入されます。
mid$(x$, m, n)	x$のmバイト目からnバイト分の文字列を取り出します。 A$=mid$("ABCD", 2, 2) … A$には、"BC"が代入されます。
right$(x$, n)	x$の右からnバイト分の文字列を取り出します。 A$=right$("ABCD", 2) … A$には、"CD"が代入されます。
asc(x$)	x$の最初の文字の文字コードを求めます。 A=asc("ABCD"):B=asc("漢字") … A$には、65、Bには-30017(&H8ABF)が代入されます。
chr$(n[,n]…)	nを文字コードとして文字に変換します。 print chr$(65) … "A"を表示します。
val(x$)	数字の並びの文字列x$を数値データに変換します。 A=val("1234") … Aに1234を代入します。
str$(n)	nの値を数字の並びの文字列に変換します。 A$=str$(2*3) … A$に"6"を代入します。
len(x$)	x$の文字長(バイト数)を求めます。 A=len("漢字") … Aには、4が代入されます。
mirror$(x$)	x$の1バイトずつの文字コードを、8ビットの2進数で左右対称にした文字列を求めます。 print ascchr$(mirror$(hexchr$("F08C"))) … "0F31"と表示します。

space$(n) spc(n)	n個分のスペースを並べた文字列を求めます。 A$=space$(10) … A$に10個分のスペースを代入します。
string$(x$, n)	x$をn個、並べた文字列を求めます。 print string$("AB", 3) "ABABAB"と表示します。
hex$(n)	nの値を16進数に変換した文字列を求めます。 print hex$(255) "FF"と表示します。
oct$(n)	nの値を8進数に変換した文字列を求めます。 print oct$(255) "377"と表示します。
bin$(n)	nの値を2進数に変換した文字列を求めます。 print bin$(255) "11111111"と表示します。
hexchr$(x$)	16進数の並びの文字列x$の2文字づつを1バイトの文字列に変換した文字列を求めます。 print hexchr$("414243") "ABC"と表示します。
ascchr$(x$)	x$の1バイトずつを2桁の16進数に変換した文字列を求めます。 print ascchr$("ABC") "414243"と表示します。
instr([n,]x$, y$)	x$中のnバイト目から、y$と同じ部分を探し、その最初の位置が何バイト目であるかを求めます。

- 引数のn, mは整数値、x$, y$は文字列型データで指定します。
- asc関数は、kmode 1 で漢字モードになっている場合、x$の1バイト目のコードが全角文字に相当するときは、2バイトの文字コードを求めます。
- chr$関数は、2バイトの文字コードで指定することもできます。また、カンマ(,)で句切って複数の文字コードを指定することができます。
- len関数の結果は、x$のバイト数になります。全角文字を含む文字列の文字数を求める関数としてklenがあります。
- hex$, oct$, bin$の各関数の引数nは、-32768～32767までの整数型の範囲で指定します。ただし、32768～65535までの値も、指定が可能で、16進数の場合&H8000～&HFFFFに変換します。
- instr関数による特定文字のさがし出しは、表示形式で行ないますのでkmode 1 で漢字モードとなっている場合、文字コードを順に調べて2バイトコードになるものは、2バイトをまとめて判断します。

関数定義

BASICでは最初から用意されている組み込み関数のほかに、自由に関数名と演算内容を決めて使うことができます。これを定義関数と呼び、FNで始まる関数名と式を定義し、あとは定義した関数名で演算させることができます。詳しい定義方法についてはdef FN文の解説を参照してください。

システム変数

システム変数

システム変数とはBASIC内で最初から予約されている変数のことで、現在時刻やBASIC内部の状態を示すポインタなどがあります。

主なシステム変数には次のものが用意されています。

時計、タイマ関係	**ti$　day$　date$　time**
ポインター	**dtl　lpos**
文字表示関係	**csrh　csrv**
グラフィック表示	**posh　posv**
エラー処理	**ern　erl**

システム定数

システム変数と同じようにBASICで予約されています。

sysid$　BASICバージョン(型式)の文字列になります。

データの型変換

BASICの数値型データは必要に応じて型変換が行なわれます。特に精度の異なる型変換が行なわれると、予定外の結果が出てしまう可能性があります。型変換の状況を良く理解して上手に活用してください。

- **精度が低くなる変換**

  精度が低くなる場合は、四捨五入が行なわれます。

  例）
```
print  cint(12.58)
 13
print  csng(3.141592653589793#)
 3.1415927
```

- **代入時の変換**

  変数に代入するときは、変数の型に合わせた変換が行なわれます。

  例）
```
10    ABC%=123.4
20    ABC#=123.4
30    XYZ#=123.4#
40    print  ABC%,ABC#,XYZ#
run
 123          123.400000005965          123.4
```

  …… 変数ABC#には単精度の定数を倍精度に変換して代入されますので、変換後の精度は単精度と同じになります。倍精度のデータとして代入する場合は、変数XYZ#への代入のように倍精度の定数(倍精度の型記号#を付けるか、9桁以上とする)を書く必要があります。

- **演算時の変換**

整数型専用のものを除いた演算や関数は、単精度型または倍精度型で行なわれ、精度の異なるデータどうしの場合、精度の高い方の精度に変換して行なわれます。また、関数の場合は引数の型に合わせた結果を求めます。

例） print 2/3＃

.6666666666666667

print sqr(5),sqr(5＃)

2.236068 2.23606797749979

- **整数型演算時の変換**

modや￥(整数除算)、論理演算などの整数型で行なわれる演算は、演算するデータをあらかじめ整数型に変換してから演算を行ないます。

例） print 100 ￥ 2.5

33 …… 2.5が整数型に変換するときに四捨五入されて3になります。

エラー

命令の入力時やプログラム実行途中に書式の間違いや範囲をこえるデータの使用、または周辺装置の異常などがあれば、コンピュータはエラーを発生して実行を中断します。このときエラーの内容に応じたエラーメッセージを表示します。文法上のエラーの場合次のように表示されます。

- **直接実行モード時**

  Syntax error

- **関接実行モード時**

  プログラム実行時は発生した行番号も表示されます。

  Syntax error in nn …… nnが発生行番号

  プログラム実行中にエラーが発生したとき、編集コマンドの行番号指定にピリオド(.)を使用すると、ピリオドにはエラー発生行番号として扱われます。

エラーの処理

プログラム実行中にエラーが発生したとき、実行を中断せずに発生したエラーの内容に応じた処理をして、元の処理を再開させることができます。

エラー処理関係の命令には次のものが用意されています。

on error goto	……	エラー発生時に実行する行の指定。
ern	……	エラー番号を求める。
erl	……	エラー発生行を求める。
resume	……	エラー処理を終了して復帰する。

エラー番号はエラーの種類に応じて割り当てられています。エラーメッセージ表(付録)を参照してください。

演算誤差

BASICで演算処理を行なった場合有限桁の値を2進数で計算していますので、どうしても誤差が発生します。以下に誤差の発生する原因をいくつか示しますので演算処理での参考にしてください。

- **丸め誤差**

  計算結果の桁数がコンピュータで扱われる桁数をこえた場合、扱う桁数以下の部分は四捨五入(一般に「丸める」という)されてしまいます。単精度型では8桁までの数を浮動少数点形式(M＊10^nの型)で扱うので、1.23456789は内部的に1.2345679になり、2/3の計算結果は0.66666667に丸められます。また、12345678＋0.02は、12345678になってしまいます。

- **2↔10進変換誤差**

  BASIC内部では数を2進数に変換して処理します。小数点以下の場合、10進数では桁数の少ない数値でも、2進数に直すと無限の桁数になることがあります。たとえば、10進数の0.1を2進数に変換すると0.00011001100…となり誤差が生じることになります。

- **近似値計算**

  sin, **cos**, **exp**, べき乗などの関数は、近似値計算をしてその値を求めています。その結果、近似した分だけ、誤差が生じることになります。

- **桁落ち**

  近似値どうしの減算の場合、誤差を含んだ桁数を増やしてしまうことがあります。たとえば、1の位に誤差を含んでいる数どうしの100012－100001の計算をコンピュータで行なうと、それぞれの誤差(絶対誤差)は±1以内、誤差率(相対誤差)は±0.01%になります。計算を行なうと答えは11になります。このときも誤差は±1以内ですが誤差率は10%になってしまい誤差の影響が大きくなってしまいます。

これらが原因になって起こる誤差を最小限度にくい止めるには、絶対数の小さな数から計算して行き、除算はできるだけあとに行ない、そして関数計算を最後に行なうようにしてください。

4 漢字の扱いについて

漢字を文字列型データとして扱うことができ、プログラム中で、変数や配列名、ラベル名、定義関数名、コメントなどに使用できます。

BASIC内部では1文字の漢字を2バイトのシフトJISコードで扱い、通常のアルファベットの2倍の幅で表示を行ないます。漢字コードには漢字以外の文字(非漢字)であるアルファベットや記号がありますので、1バイトのコードで表されるアルファベットや数字、記号、カタカナなどを半角文字と呼び、2バイトのコードで表される漢字や記号等を全角文字と呼びます。(JIS漢字コード → オーナーズマニアル第9章参照)

全角文字のシフトJISコードとは、JISで定められた2バイトのJISコードを半角文字のアルファベットやカタカナと重ならないようにずらせたコードのことです。シフトJISの第1バイト目は16進数で&H81～9Fと&HE0～FCの範囲になります。これは半角文字では図形(グラフィック)文字のコードに相当します。そのため半角図形文字と全角文字のいずれかを選ぶ必要があるため切り換え命令として、**kmode**文を用意しています。

kmode 0 …… この状態ではすべての文字が半角文字として扱われます。

kmode 1 …… 文字コードが&H81～&H9F、または&HE0～&HFCの範囲にあるときは、次に続く1バイトと合わせて2バイトコードで扱われます。

JIS漢字コード
JIS(日本工業規格)により規格化されているコードです。当用漢字を中心とした約3000種の漢字、英文字、数字、特殊文字がJIS第1水準と呼ばれます。

漢字モードにより次の命令の実行内容が変化します。

コマンド …**search**
表示 …**symbol**、**console**
printや**list**などでの文字表示
関数 …**instr**

コードの変換

全角文字は内部では、シフトJISコードで扱われていますが、JIS16進コードやJIS区点コードとシフトJISコードを相互交換する機能があります。

JISコード定数は**&J**を付けて16進数で表します。

例）**print　chr$(&J2122, &J3021)**

区点コード定数は**&K**を付けて10進2桁ずつで表します。

例）**print　chr$(&K0102, &J1601)**

シフトJISよりJISコードや区点コードに変換する関数が用意されています。

jis$(X$) …… **X$**をJIS16進コード文字列に変換します。

ktn$(X$) …… **X$**を10進4桁の区点コード文字列に変換します。

これらの関数は引数（例では**X$**）の最初の文字についてのみ有効であとに続く文字は無視します。

全角文字と半角文字の変換

英数字や一部の記号、カタカナは全角文字と半角文字の両方にあります。そのため相互の変換を行なう関数が用意されています。

akcnv$(X$) ……X$の半角文字を全角文字列に変換します。

kacnv$(X$) ……X$の全角文字を半角文字列に変換します。

そのほかの漢字関数

klen(X$) ……X$が全角と半角文字が混在した状態の文字数を求めます。

kpos(X$, n) ……X$のn番目の文字が、何バイト目にあるかを求めます。

5 ファイルの処理

フロッピーディスクドライブなどの装置に対してプログラムやデータを読み書きするための様々なファイル処理命令が用意されています。

BASICで扱うファイルの種類は次のとおりです。

BTX	BASICのプログラムファイル
BSD	シーケンシャルアクセス・データファイル
BRD	ランダムアクセス・データファイル
OBJ	機械語プログラムファイル
DIR	階層ディレクトリ管理用ファイル

- **BTXファイル**

  BTXファイルはインタプリタ上で作成したプログラムをインタプリタの内部形式でファイルしたもので、saveコマンドにより書き込み、loadコマンドにより呼び出しが行なわれます。また、プログラムを分割して使用する命令としてchain文やswap文が用意されています。

- **BSDファイル**

  BSDファイルはデータを順番にアクセス(読み書き)します。データ1個当たりの長さがまちまちでも順番に詰めて記録されますので、容量の効率も良くなります。ただし、アクセスはファイルの先頭より順番に行なう必要があり、途中のデータを直接アクセスすることはできません。

- **BRDファイル**

  BRDファイルは、任意の位置のデータを直接アクセスできるようになっています。ただし、データが固定長になるのでデータ長がまちまちの場合、容量の効率が悪くなります。

- **DIRファイル**

  DIRファイルはファイルの登録数が多くなるようなときグループ別にディレクトリを分けて参照する階層ディレクトリの管理を行なうファイルです。

ファイルの指定方法

ファイルはファイル名を付けて各装置に登録してあります。そのファイルを指定する場合は次のような文字データにより行ないます。

"<装置名><ファイル名>"

装置名

装置名はフロッピーディスクドライブやカセットテープレコーダなどの装置を示します。

装置名	指定される装置	プログラム入出力	シーケンシャル		ランダム入出力
			入力	出力	
FD1: | FD4:	フロッピーディスクドライブ	○	○	○	○
MEM:	メモリディスク	○	○	○	○
COM1: | COM2:	RS−232C インターフェイス	−	○	○	−
CMT:	カセットテープレコーダ	△(入力のみ)	○	−	−
CRT1:	ディスプレイ画面	−	−	○	−
LPT:	プリンタ	−	−	○	−
KB:	キーボード	−	○	−	−

- フロッピーディスクドライブは最大4基までサポートされています。ランダムアクセスが可能で、階層ディレクトリによるファイル管理が可能となっています。外部にディスクドライブを接続した場合にディスクドライブの種類に合わせてトラック数やタイミングを**init "FD:"**によりドライブ番号ごとに設定できます。
- メモリディスクは、グラフィクV-RAMやメインメモリの一部を使用し、フロッピーディスクと同様のランダムアクセスを行ないます。メモリを使用しますのでフロッピーディスクよりも高速でアクセスができ、プログラムやデータの一時的な記録やランダムアクセスファイルのデータ検索などに使用すると効果があります。ただし、コンピュータ本体の電源を切ると内容が消えてしまうので、必要なファイルはあらかじめフロッピーディスクへ移しかえておく必要があります。プログラムファイルの移しかえは**load**や

saveコマンドで行ないます。データファイルの移しかえはプログラムを組む必要がありますが、**"フォーマット&コピー"**ユーティリティを利用することもできます。☜コラム

"フォーマット&コピー"ユーティリティについては、オーナーズマニュアルの第5章の1「ユーティリティの使いかた」を参照してください。

- カセットテープは、プログラムやシーケンシャルアクセスファイルの読み出しに使用します。BASIC上の命令でファイルの書き込みはできませんが、アナログ録音やテープの位置をチェックできるようにデジタルトラックを使用しての頭出し信号のマーキングが可能です。また、**"フォーマット&コピー"**ユーティリティを使用すると、カセットテープへの書き込みが可能となります。
- RS-232Cインターフェイスはシーケンシャルアクセスファイルとして扱うことができます。BASICのプログラムもBSD形式ファイルで利用できます。
- ディスプレイ画面は出力専用の、ファイル装置として使用します。また、ファイルオープン時に**"CRT:CG"**として指定するとコントロールコードを実行させずに表示させることもできます。
- プリンタは出力専用のファイル装置として使用します。
- キーボードは入力専用のファイル装置として使用します。

ファイル名

ファイル名はファイルごとに付ける名前のことです。装置内に複数のファイルがある場合、どのファイルが指定されたのかを見分ける必要がありますので、同じファイル名を付けることはできません。ファイル名には次の文字以外は、半角文字や全角文字も使用可能で、最大16文字まで可能です。

ファイル名に使用できない文字

…… : ／ ? ＊ （すべて半角文字）

ディレクトリ

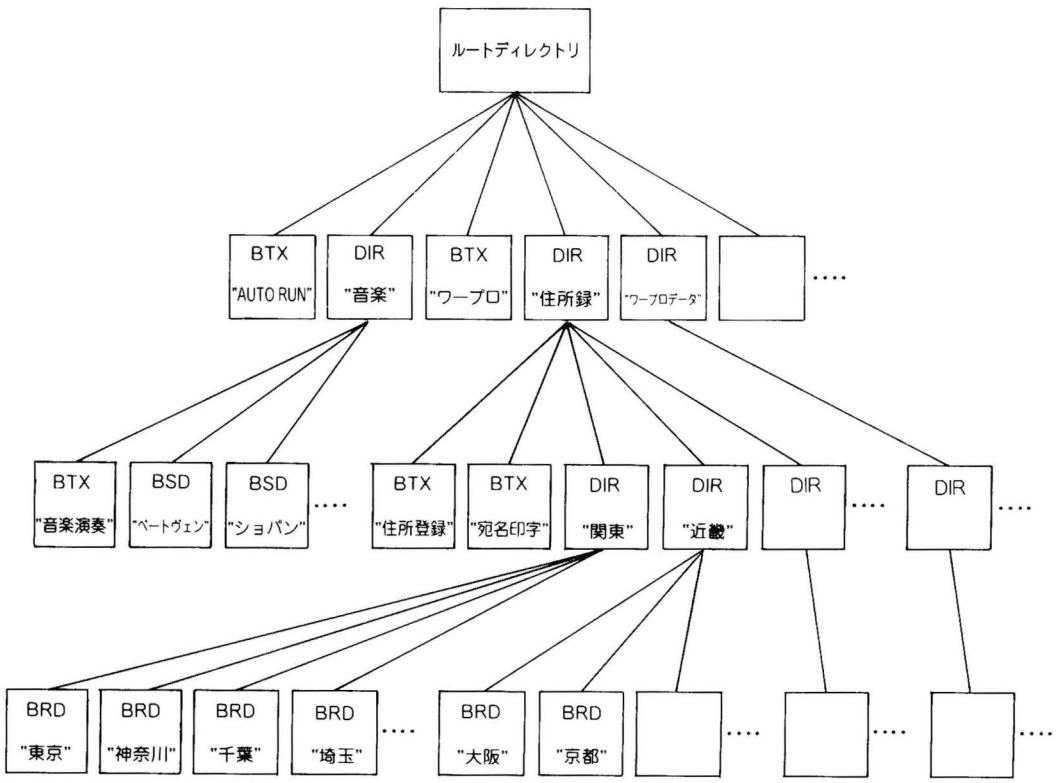
ディレクトリ(directory)とは台帳ということでファイルの管理台帳ということになります。ここにはファイルごとのファイル名、ファイル形式、書き込み禁止の有無、記録容量、記録位置、登録時間などの情報が管理されます。

たとえば、フロッピーディスクの場合、ディスクごとにこのディレクトリをもち、それぞれのディスク内のファイルを管理しています。当BASICでは1つのディレクトリ内に最大63個のファイルが登録できます。

階層ディレクトリ

1つのディレクトリ内に多くのファイルを登録した場合は、その登録状態を管理するのは大変です。そこでディレクトリを階層構造(木構造)にして1枚のフロッピーディスク内でグループ分けで整理したり、多くのファイルを登録することができます。

階層ディレクトリを図で表すと次のようになります。



基本となるディレクトリを木の根にたとえルートディレクトリと呼び、その中に**BTX**や**BSD**などのファイルにまじって**DIR**のファイルがあります。このファイルの中で別のディレクトリ(子ディレクトリ)を管理しています。子ディレクトリの中で一般ファイルのほかさらに、子ディレクトリを作成することができます。このように階層構造(木構造…図では根が上にあり幹や枝葉は下にのびています)でファイルを管理すると多くのファイルが扱いやすくなります。

階層ディレクトリの操作

フロッピーディスクは初期状態ではルートディレクトリのみとなっています。そのままの状態でも63個までのファイル登録ができますが、新しく子ディレクトリを作成するには、

mkdir "<ディレクトリ名>"

とします。これで新たなディレクトリができます。しかし、このままではdirコマンドでファイルのリストを表示してもルートディレクトリ中にあるファイルしか参照できません。

そこで、現在のディレクトリより子ディレクトリに移るには、

chdir "<ディレクトリ名>"

とします。これで現在は子ディレクトリにいることになります。この状態でdirコマンドによりファイルのリストを出すと

DIR＊ "."

DIR＊ ".."

の2つのファイルが表示されます。これはディレクトリの階層を管理するファイルで ".."は元の親のディレクトリを意味し、"." は現在のディレクトリを意味します。これらのファイルは直接消去することはできません。これらのディレクトリ管理ファイルはディレクトリ自身を消去すれば同時に消去されます。

現在参照しているディレクトリをカレントディレクトリと呼びカレントディレクトリから親ディレクトリに戻ることもできます。

chdir ".."

とすると現在のディレクトリの親ディレクトリに戻ることができ、新しいカレントディレクトリは戻った親ディレクトリとなります。

このようにして階層ディレクトリ構造はmkdir文により子ディレクトリを作り、子ディレクトリ内にまた子ディレクトリを作り、と次々に階層を増やして行くことができます。

階層は最大10階層まで作ることができます。

ディレクトリ階層の直接指定

ここまではchdir "<ディレクトリ名>"で子ディレクトリへ、chdir " .."で親ディレクトリへと、カレントディレクトリの移動について説明しました。しかし、カレントディレクトリより直接ほかの階層内のファイルをアクセスしたり、親や子ディレクトリ以外へ直接カレントディレクトリを移動することもできます。
ファイル指定の場合次のようになります。

"〔<装置名>〕〔/〕<ディレクトリ名>/〔<ディレクトリ名>/〕…<ファイル名>"

先のファイル指定と比べて<装置名>と<ファイル名>の間にスラッシュ(/)や<ディレクトリ名>が指定されています。これはディレクトリ階層の道すじを示すもので、ディレクトリパス(path)と呼びます。

最初のスラッシュ(/)はルートディレクトリを表し、<装置名>とスラッシュを省略するとカレントディレクトリからの指定になり、最初の<ディレクトリ名>で指定されたディレクトリファイルのあるディレクトリを指します。それ以後、スラッシュを区切りとして<ディレクトリ名>の流れを指定し、最後に<ファイル名>を指定します。

図の例で行くと、現在のカレントディレクトリが "住所録"の中だとすると、dirコマンドを実行すれば、BTXの "住所登録"や "宛名印字"、DIRの"関東"や"近畿"のファイルが表示されます。

このときのディレクトリパスは "/住所録"になります。ここでルートディレクトリ内のDIR "音楽"の中にあるBSD "ベートーヴェン"を指定するには "/音楽/ベートーヴェン"となり、カレントディレクトリ内のDIR"関東"の中にあるBRD"埼玉"を指定するには、"関東/埼玉"となります。

カレントディレクトリを移動するchdirコマンドの場合は、ファイル指定の場合の<ファイル名>とその直前のスラッシュを除いた形のディレクトリパスを指定すれば、直接の移動ができます。

ディレクトリパスの指定で最初のルートディレクトリを表すスラッシュを省略しても、<装置名>が指定されているとルートディレクトリからとみなします。カレントディレクトリからとするには、<装置名>と最初のスラッシュとの両方を省略する必要があります。

一般ファイルの操作

・ファイルの登録

プログラムファイルの登録はsaveコマンドにより行なわれ、BTXまたはBSDファイルになります。

データファイルはwopen文またはxopen文でファイルオープンを行ない、close文によりファイルの書き込みが正常に終了すれば登録が行なわれます。

・ファイルの読み込み

プログラムファイルの場合は、loadコマンドによりメモリ上に読み込みます。（またはrun＜ファイル指定＞によるload&runも可能です。）

データファイルの場合は、ropen文によりファイルオープンをしてデータを読み、close文により終了します。

・ファイルの消去

登録されているファイルは、deleteコマンドによりディレクトリより消去されます。

・ファイルのプロテクト

登録されているファイルを誤って消去しないように、ファイルにプロテクトをかけることができます。lockコマンドを使用してプロテクトをかけたあと、dirコマンドでファイルのリストを表示させると、BTXやBSDなどのファイル形式のあとに＊が表示され、deleteコマンドでの消去やデータファイルの書き換えを行なうとエラーになります。

プロテクトをはずすのはunlockコマンドで行ないます。

6 表示機能

文字表示機能

文字表示には縦横の文字数、表示色数、スクロール方式、１つの文字を構成するドット数などのモードと文字ごとの表示色、ブリンク表示、リバース表示などの指定、文字のフォントを設定するPCGや外字の機能があります。

表示文字数

横方向の桁数は、80桁と40桁、縦方向の行数は、25、20、12の行数の指定ができます。ただしグラフィック表示が256色モードのときは40桁のみになります。文字表示が８色のときに40桁の指定をすると、文字画面が２面になり、**screen**文での表示切り換えや重ね合わせ表示ができます。

文字の表示色

文字の表示色は、８色または64色で表現することが可能です。表示色のモードは、グラフィックの表示色数の設定で決まり、グラフィックが４色または16色モードのときは、８色になり、256色モードのときは、64色になります。64色のときは、桁数が40桁のみになります。(64色モードで表示する場合は、アナログ入力のディスプレイが必要です。)

スクロールモード

画面の最下行で改行した場合は新らしい行を表示するために、表示されている文字が上にスクロールします。スクロールの方法としてラインスクロールとスムーススクロールがあり、ラインスクロールは、行単位で移動しますが、スムーススクロールは文字を構成するドット単位でなめらかに移動して見やすくなります。200ラスターディスプレイを使用したときは１ドット単位、400ラスターディスプレイを使用したときは２ドット単位のスクロールになります。

スムーススクロールのときはconsole文による表示範囲の指定ができなくなり、一番下の行は表示に使用されなくなりますのでファンクションキーを表示させたり、漢字入力を行なうことはできなくなります。　40桁で8色表示の場合は文字画面を2面もちますのでスムーススクロールはできなくなります。

グラフィックの表示色数と文字表示色数、桁数の組み合わせとスムーススクロールは次のようになります。

グラフィック表示色	文字表示色	桁数	スムーススクロール	画面数
16または4色	8色	80	可	1
		40	不可	2
256色	64色	40	可	1

文字フォント

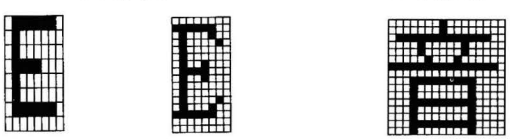
アルファベットなどの半角文字のドット構成を8×8ドットと8×16ドットの2種類を選択することができます。ただし、縦12行表示では8×16ドットになり、200ラスターディスプレイを使用したときは、縦20または25行の表示では8×8ドットになります。

使用するディスプレイによる、行数と文字フォントの関係は次のようになります。

ディスプレイ	行数	フォント	
		8×8ドット	8×16ドット
400ラスター	25, 20	○	○
	12	×	○
200ラスター	25, 20	○	×
	12	×	○

文字フォントの例

半角文字　　全角文字



8×8ドット　8×16ドット　16×16ドット

グラフィック表示機能

グラフィックは縦横のドット数、同時表示色によりモード指定が行なわれます。モードの設定は**init "CRT2:"**文により行ないます。

横のドット数は320と640、縦のドット数は200と400、同時表示色数は、4、16、256色があり、組み合わせは次のようになります。

ドット数		同時	最大	重ね合わせ表示
横	縦	表示色	画面数	
320	200	16	4	0と1または2と3
320	200	256	2	—
640	200	16	2	—
640	400	4	1	—
640	400	16	1	—

最大画面数は、別売の増設ビデオRAMボード(MZ-1R27)を取り付けたときの画面数で、取り付けていない場合、640×400の4色モード以外は半分の画面数になります。したがって、増設ビデオRAMボードを取り付けていない場合は640×400の16色モードは指定できません。

複数の画面をもった場合は0から始まるスクリーン番号(画面番号)で画面の指定を行ないます。したがって、640×200ドットの16色モードのときは、増設ビデオRAMボードを取り付けていると0、1、2、3の4つのスクリーン番号が割り当てられます。

重ね合わせ表示は、複数の画面をもったときの重ね合わせ表示可能なスクリーン番号を示しています。

縦400ドットのモードは400ラスターディスプレイを使用しないと表示できません。400ラスターディスプレイで縦200ドットの指定は可能です。

同時表示色が4色または16色のときは文字表示色は8色となり、256色のときは文字表示色が64色で、文字の桁数は40桁となります。(グラフィックの16色モードで表示する場合は、R.G.B.I.方式またはアナログ入力方式、256色モードで表示する場合は、アナログ入力方式のディスプレイが必要です。)

色コード

文字やグラフィックの表示色は色に番号を対応させたカラーコードにより指定します。BASICでは8色、16色、512色の3種類のカラーコードがあります。

16色コード

グラフィックの16色または4色モードでの色コードで、R. G. B. I. の4種の色信号により0～15で表します。

R …Red 赤
G …Green 緑
B …Blue 青
I …Intensity 明るさ

4つの信号の組み合わせで次の色が表現でき、色コードが割り当てられます

16色コード	表示色	色信号 I	R	G	B
0	黒	0	0	0	0
1	青	0	0	0	1
2	赤	0	0	1	0
3	マゼンタ	0	0	1	1
4	緑	0	1	0	0
5	シアン	0	1	0	1
6	黄	0	1	1	0
7	白	0	1	1	1

16色コード	表示色	色信号 I	R	G	B
8	灰	1	0	0	0
9	明青	1	0	0	1
10	明赤	1	0	1	0
11	明マゼンタ	1	0	1	1
12	明緑	1	1	0	0
13	明シアン	1	1	0	1
14	明黄	1	1	1	0
15	明白	1	1	1	1

0…信号が無い　　1…信号がある

8色コード

このコードは文字画面が8色モードのときに使用され、基本的にはR. G. B. の3種の信号の組み合わせで0～7で表します。ただし、コードが0の黒以外はI信号を使用していますので、実際の表示色は1～7が16色コードの9～15に対応します。

512色コード

512色コードはグラフィックが256色モードのときにグラフィックと文字の表示色に使用します。512色はR.G.B.の各信号が8段階に変化する組み合わせで表示され、0～511の値で表します。

コード番号は各信号を3桁の2進数に展開したものを上の桁からG.R.B.の順に並べた9桁の2進数に対応します。

色信号	G			R			B		
レベル	4/7	2/7	1/7	4/7	2/7	1/7	4/7	2/7	1/7
2進数	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

2進数では桁が多くなりますので通常は各信号を8進数で表現します。

8進数	2 進 数			信号レベル
	2^2	2^1	2^0	
0	0	0	0	0
1	0	0	1	1/7
0	0	1	0	2/7
3	0	1	1	3/7
4	1	0	0	4/7
5	1	0	1	5/7
6	1	1	0	6/7
7	1	1	1	1

いちばん輝度の高い黄色は次のようになります。

2進数　1 1 1　1 1 1　0 0 0　…&B111111000
8進数　7　7　0　…&O770
色信号　G　R　B

文字の色指定

8色モード

グラフィックが4色または16色モードのときは、文字色は8色コードで表します。文字の表示色はグラフィックの16色パレット機能では、変化しませんが別売のパレットボード(MZ-1M10)を取り付けた場合の4096色パレット機能では表示色が変化します。ただし8色コードの1～7の色は、カラーコード9～15に対応した変化となります。

8色コードと16色コードの対応は次のようになります。

8色コード	0	1	2	3	4	5	6	7
16色コード	0	9	10	11	12	13	14	15

64色モード

グラフィックが256色のときの文字表示は40桁の64色モードになります。このときの色指定はグラフィックと同様に512色コードで行ないます。

512色コードで指定を行なった場合は、R.G.B.各色信号の一番変化の少ない信号は0になります。これは8進数で色コードを指定した場合、奇数の桁は1が引かれて偶数になると見るとわかりやすくなります。

例) 指定値　　実際の設定値

&O777 → &O666

&O135 → &O024

&O247 → &O246

グラフィックの色指定

パレットコード

グラフィックの16色モードは1つのスクリーンにつきR. G. B. I. の4つのプレーンを使用します。この4プレーンの組み合わせでできるコードをパレットコードと呼び、実際の描画命令での色指定になります。

パレットコードは、プレーンをI. G. R. B. の順に上の桁から並べた4桁の2進数に対応し、0～15の値になります。

パレット機能

パレットコードはプレーンの組み合わせで0～15で表し、16色コードは信号の組み合わせで0～15で表されます。グラフィックが初期化されたときは、パレットコードと16色コードは1対1で対応していますが、このパレットと色コードの対応を変更できるのがパレット機能となります。

パレット機能はcolor=文で設定することができ、各パレットコードに対する色コードを設定できます。この機能はディスプレイへの出力色を指定するもので、異なったパレットコードに同じ色コードを対応させることもできます。また、グラフィックV-RAMの書き換えを行なわなくても、画面の表示色を変更することも可能です。

4096色パレット

このパレット機能は、別売のカラーパレットボード(MZ-1M10)を取り付け、アナログ入力のディスプレイを使用したときに指定可能です。4096色はR. G. B. の各信号を16段階に変化をするようにしたもので、各信号のレベルを0～15で指定します。このパレット機能も、color＝文で設定し、1～15の各パレットコードに対してディスプレイに出力される色を指定します。(パレットコードの0には、色信号を設定することはできません。)

4色モード

このモードは640×400ドットとグラフィックの解像度を上げたときのモードで、BとIの2つのプレーンの組み合わせで、0～3のパレットコードとなります。

4色モードのときも、16色パレットや4096色パレット機能を使って自由に表示色を設定することができます。

256色モード

このモードはグラフィックドット数を320×200としたときに可能となるモードで8つのプレーンを組み合わせて表示されます。このモードのときの描画色は、512色コードで指定するわけですが、512色すべてを出力しようとすると9つの信号が必要となります。
そこで9つの信号のうち1つの信号だけは、出力されずに常時0の値とする方法を採っています。出力されない信号は、R.G.B.それぞれの色のいちばん変化の少ない信号の中から1つを選び、その設定は**cblock**文で行ないます。

各プレーンは次のように並んでいて**cblock**文によるカラーブロックの指定により次のように割り当てられます。

256色モード時のプレーンの並び

I_1	G_1	R_1	B_1	I_0	G_0	R_0	B_0

cblockによる信号の割り当て

色信号		G			R			B		
レベル		4/7	2/7	1/7	4/7	2/7	1/7	4/7	2/7	1/7
指	**0**	G_1	G_0	I_1	R_1	R_0	I_0	B_1	B_0	0
定	**1**	G_1	G_0	I_1	R_1	R_0	0	B_1	B_0	I_0
値	**2**	G_1	G_0	0	R_1	R_0	I_1	B_1	B_0	I_0

0となる部分が使用されない信号です。

256色モード時に512色コードで色指定が行なわれるとこの**cblock**文の指定で出力されない信号にあたる部分は無視されます。これは512色コードを8進数で表したときに**cblock**の指定に該当する色の桁が奇数の場合1を引いて偶数になると見るとわかりやすくなります。

例) 色コードの指定が**&O777**であれば

cblock 0 → &O776

cblock 1 → &O767

cblock 2 → &O677

プライオリティ機能

文字とグラフィックを同時に表示させた場合や、グラフィックを2画面同時に表示させた場合に、グラフィックどうしの表示が重なる所やグラフィックと文字の重なる所ではどちらかの表示が優先されて表示されます。この表示の優先順位(プライオリティ)を指定することができます。

文字とグラフィック表示

グラフィックが4または16色モードのときに、各パレットコードに対して文字との優先順位を設定することができます。初期設定は文字の方がすべてのパレットコードに対して優先順位が高くなっています。

グラフィックとグラフィック表示

グラフィックが320×200ドットの16色モードのときに2面同時に表示させた場合はこの2面の順位の入れ換えが可能です。また、文字との優先順位はこのグラフィック2画面が合成された信号に対して行なわれます。

文字画面どうしの表示

文字画面は8色モード(グラフィック4、16色モード)のときに桁数を40とすると2画面もつことができ、同時表示も可能となります。この場合は優先順位が固定で文字スクリーン番号0の方が1の方より優先されます。

これらの画面の重ね合わせ表示の指定は**screen**文、優先順位は、**priority**文により指定します。

文字表示のアトリビュート

画面に表示される文字の１文字ごとに色や、リバース、ブリンクなどのアトリビュート(属性)を指定することができます。

表示色の指定

文字の表示色は**ccolor**文で文字カラーとして指定します。 指定後に表示される文字の色はこの指定色になります。

文字カラーは８色モードでは0～7の整数値になり、０はグラフィックの色と同じですが1～7の色はグラフィックの明るい方の色になる9～15に対応しています。

リバース表示

文字のリバースは**crev 1**とすることによりそれ以後表示される文字がリバース(反転)状態となります。

リバースの解除は**crev 0**とします。

文字のフォントは8×8または8×16ドットで構成され、リバース状態は通常光っている部分を消し、消えている部分を光らせます。PCG1～PCG3を組み合わせてドット単位で色指定ができるPCGは、各ドット色が補色になります。

通常表示　　リバース表示

ブリンク表示

文字のブリンクは**cflash 1**とすることによりそれ以後表示される文字がブリンク(点滅)状態となります。

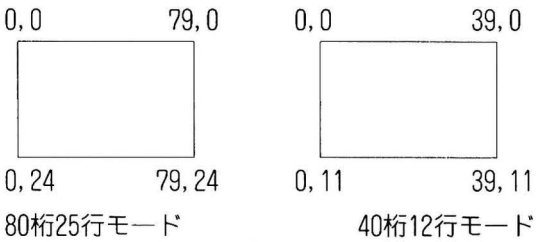
ブリンクの解除は**cflash 0**で、以後の表示される文字は通常の状態に戻ります。

文字座標

文字の表示位置を示すために文字座標が決められています。文字座標は文字の表示位置の移動や表示範囲の指定などに使用されます。

文字座標は左上の位置を0,0として横方向は右へ、縦方向は下へ向かって値が増加します。

文字座標の例



文字座標により指定を行なう命令には次のものが用意されています。

console …… 文字のスクロール範囲の指定。
console@ …… 文字の表示出力範囲の指定。
ccolor@ …… 指定範囲の文字色を変更。
crev@ …… 指定範囲の文字表示の反転。
cflash@ …… 指定範囲の文字表示の点滅。
cursor …… 文字表示位置の指定。
get% …… 指定範囲の文字データを配列に取り込む。
put% …… 配列の文字データを指定範囲へ表示する。

PCGと外字機能

PCGはProgrammable Character Generator（プログラマブル・キャラクタ・ジェネレータ）の略で文字のフォントを自由に定義することができます。本体内にはPCG-0～PCG-3のそれぞれ2Kバイトの PCG用メモリがあります。

各PCGの2Kバイトの容量は8×8ドットで256文字に相当します。したがって合計すると1024個の文字が定義できます。

PCG-1～PCG-3を組み合わせてドットごとに色を定義することができます。PCG-0～3の単独での使用時は、文字フォントの色はそのときの文字カラーの指定によりすべて同じ色（単色）になりますが、組み合わせて使用すると8色モードのときは、PCG-1～3にB,R,Gの色を割り当て、合成することによりドット単位の8色指定が可能になり、64色モードでは、2つの組み合わせを使用してドット単位の64色指定が可能になります。

PCGの表示は、**cgen**文によりPCG-0～3のいずれか単独での使用、またはPCG-1～3の組み合わせでの使用かを指定するとPCGの表示モードとなり、それ以後表示される文字コードに応じた表示がされます。

表示や定義のときの文字コードとの対応は次のようになります。

- 8×8ドットのときは、各PCGに256種の定義ができ、文字コードの0～255に対応します。
- 8×16ドットのときは、各PCGに128種の定義ができ、0～255のうち偶数値で指定します。これは8×8ドットのときの偶数奇数と続く部分を使用するためです。
- 64色モードでPCG-1～3を組み合わせて使用する場合は、指定コードと指定コードに128を加えたコードのPCGが使用されます。（指定コードが128以上の場合は指定コード-128のコードになります。）色信号は指定コードの側が変化の多い方の色信号に使用されます。

外字機能

外字は16×16ドットの全角文字となります。PCG-1～PCG-3を使用して合計192種の定義ができます。ただし、ドット単位での色指定はできません。文字コードは、**&HEC9F**～**&HECFC**, **&HED40**～**&HED7E**, **&HED80**～**&HEDA2**の範囲でJISコードでは**&J7821**～**&J787E**, **&J7921**～**&J797E**, **&J7A21**～**&J7A24**となります。外字はこのコードにより、表示および定義を行ないます。

PCGおよび外字の定義や、定義の読み出し命令は次のものが用意されています。

def chr$ … PCG、外字の定義。
cgpat$ … PCGや漢字ROMよりフォントの読み出し。

グラフィック座標

グラフィック画面の位置は座標により指定します。グラフィック座標は、使用目的により絶対スクリーン座標、スクリーン座標、ワールド座標の3種の座標指定を行ないます。

絶対スクリーン座標

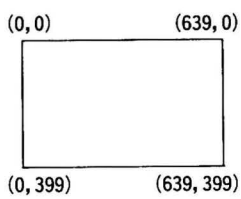
絶対スクリーン座標は、グラフィックとして表示可能な範囲の左上を原点(0,0)としてX軸(横方向)は右方向にグラフィック表示の1ドット単位に対応して増えて行き、Y軸(縦方向)は下に向かってグラフィック表示の1ドット単位に対応して増えて行きます。

この座標は画面の1ドット単位に対応した最も基本となる座標で、グラフィックの描画命令によって図形の描かれる範囲の指定やディスプレイへの出力範囲の指定のときに使用されます。

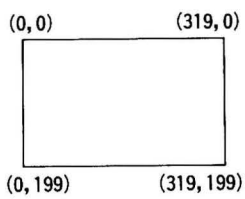
絶対スクリーン座標を使用する命令には次のものが用意されています。

view	……	グラフィックの描画範囲の指定。
view@	……	グラフィック表示がディスプレイへ出力される範囲の指定。
mouse	……	マウスの機能設定を行なうステートメント。
mouse(n)	……	マウスより座標情報を入力する関数。

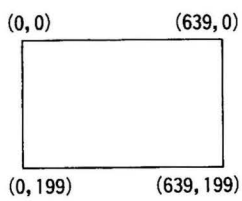
絶対スクリーン座標の例



640×400ドット



320×200ドット



640×200ドット

スクリーン座標

スクリーン座標は、グラフィック図形の描画範囲としてされたビューポートの左上を原点(0,0)としてグラフィック表示の1ドットずつに対応する座標です。

この座標は、ドットを複数集めた面積をもつ描画や、ビューポート内の範囲を指定する場合に使用されます。

スクリーン座標を使用する命令には次のものが用意されています。

pattern	……	パターンを描く。
symbol	……	文字パターンを描く。
color replace	……	指定範囲内の色を書き換える。
move	……	指定範囲の図形を移動させる。
get@	……	表示データを配列に取り込む。
put@	……	配列内の表示データを表示する。
gsave	……	グラフィック画面のデータをファイルする。
gload	……	グラフィックファイルを表示する。

BASICの起動後やグラフィック画面の初期化を行なったときは、ビューポートが、画面いっぱいに設定されていますので、絶対スクリーン座標とスクリーン座標は同じになります。

ビューポート

ビューポートは、グラフィック描画命令により、図形が描かれる範囲のことで、ビューポートの外側へはみ出るような描画指定がされていても、範囲外には、図形が描かれません。

ビューポートは絶対スクリーン座標により、**view**文で指定を行ないます。

ワールド座標

ワールド座標は、縦横それぞれの方向に単精度実数型の範囲となる-1.7014118E38～1.7014118E38までの広い範囲をもった論理的な座標です。点や線で描画するほとんどのグラフィック命令は、このワールド座標によって描画を行ないます。

ワールド座標はたいへん広い範囲の座標を扱いますが、実際に図形を描くのは、ごく一部の範囲となる場合が多いと考えられます。また、ワールド座標がいくら広い範囲の座標であっても、グラフィック画面のドット数は最大でも640×400ドットです。

ワールド座標により図形を描くには、まず、ワールド座標の中で実際に描画を行なう範囲をウィンドウとして設定します。そうしてこのウィンドウが実際の画面の描画範囲となるビューポートに割り当てられることになります。

ウィンドウの設定は、**window**文によって行ない、ビューポートの左上と右下のワールド座標を設定します。スクリーン座標は表示の１ドット単位に対応していますが、ワールド座標は範囲を自由に設定できますので座標の拡大や縮小、小数点の付いた実数座標の指定ができます。また座標の増加方向もスクリーン座標では固定ですが、ワールド座標では**window**文で設定した座標による増加方向となり、負の座標指定も可能です。

BASICの起動時やグラフィック画面の初期化を行なった場合は、ビューポートが画面いっぱいに設定され、ウィンドウもスクリーン座標と同じに設定されていますので、絶対スクリーン座標、スクリーン座標、ワールド座標は同じになります。

ワールド座標によるグラフィック命令には次のものが用意されています。

set …… ドットをセットする。
reset …… ドットを消す。
line …… 直線を引く。
circle …… 円を描く。
poly …… 多角形を描く。
paint …… 指定色で塗りつぶす。

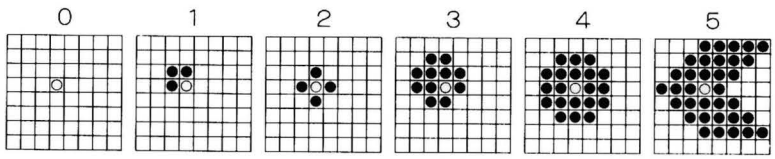
グラフィック描画時のペン機能

グラフィックの描画命令でドットや線で描画する場合のペン形状を選択または定義ができます。

ペン形状は指定された座標に打たれる点を8×8までの複数のドットを使って指定できます。BASICには0～5の6つのペンが用意されていて、あらかじめペン形状が設定されていますが、1～5のペンについては形状を定義することができます。

このペン形状を選択することにより、特定パターンの点を打ったり、太い線を引くことができます。

BASICで用意されているペン形状は次のとおりで、左上から右下へ向って4ドット目が描画の中心座標になります。



ペン形状の選択および定義はpen文により行ないます。

指定ペン形状で描画する命令には次のものが用意されています。

set …… 点をセットする。
reset …… 点をリセットする。
line …… 線を引く。
circle …… 円を描く。
poly …… 多角形を描く。

7 キーボードの制御機能

先行入力

BASICはハードウェアの割り込みを利用して文字の先行入力を行なっています。通常のキー入力は直接実行モードの状態やinput文の実行時にカーソルが出てコンピュータがキー入力待ちになったときに行ないますが、先行入力はコンピュータがキー入力待ちのとき以外のほかの処理をしているときにも受け付けることができます。

たとえば、現在BASICに何らかの時間のかかる処理を行なわせていて次に行なうキー操作が決定している場合、あらかじめ次の操作に必要なキーを押しておくと、現在の処理が終了したのち、あらかじめ行なっていたキー入力の内容により自動的に次の処理に進ませることができます。

BASIC内部では、ほかの処理中にキーが押された場合に入力された文字を31文字まで記憶するキーバッファをもっており、そこへ次々と入力された文字が蓄えられます。そうしてBASICで入力状態となったときに記憶した順番にキーバッファから1文字ずつ取り出されます。なお、キーバッファがいっぱいになった場合は、先に入力された文字が取り出されるまで入力はできなくなります。

先行入力を利用する場合、次の点に注意してください。

- 処理中のエラーの発生や SHIFT + BREAK キーの操作により命令待ちに戻るときは、キーバッファを消去します。
- 処理中の一時停止のために BREAK キーを押した場合は、キーバッファを消去します。また、一時停止の解除のために押したキーも入力されません。
- ファンクションキーを押した場合はそのキーに定義されている文字がキーバッファに入ります。また、途中でキーバッファが、いっぱいになってあふれた文字は入力されません。
- 先行入力中はキーリピート動作はしません。
- フロッピーディスクやカセットテープの操作中は割り込み禁止となりますので、先行入力は行なわれません。click onが指定されキークリック音が出る場合は、しばらくキーを押し続けてクリック音が出れば文字が入力されたことを確認できます。

キーバッファの内容を設定する命令として**def key(0)** が用意されていて、キー入力のシミュレートやキーバッファの消去に利用できます。また CTRL ＋ SHIFT ＋ CLR/HOME キーの操作によってもキーバッファを消去することができます。

キーボードの配列

キーボード配列は標準的なJIS規格に準じた英数字やカナ配列になっていますが、**init "KB:1"**の実行により、カナ配列を50音順に変更することができます。ただし、キートップはJIS配列となっていますので本体に添付しているキーラベルを貼り付ける必要があります。

クリック音

キーを押したときに発生するクリック音を制御することができます。**click on**とするとクリック音が発生し、**click off**とするとクリック音は発生しません。

クリック音はコンピュータが有効な入力として受け取ったときに発生しますので、 SHIFT キーや CTRL キーのみを押しただけでは発生しません。

キーリピート

キーを押した続けた場合にはリピートさせる機能があります。**repeat on**とすると、リピート状態となり**repeat off**とするとキーを押し続けてもリピートしません。また、**repeat**文はリピート速度を4段階で指定する機能もあります。

ファンクションキー

ファンクションキーは1つのキーに複数の文字を定義することができ、BASICではF1～F20の20種をサポートしています。ファンクションキーのキートップはF1～F10までですが、SHIFT キーを押しながら、F1～F10のキーを押すとF11～F20に対応します。

ファンクションキーに関する命令として次のものが用意されています。

def key …… ファンクションキーの定義
klist …… 定義内容の最下行への表示制御
key list …… 定義内容のリストアップ

キー割り込み機能

プログラムの実行中に、ファンクションキーや HELP キー、SHIFT ＋ BREAK キーを押したときに割り込みをかけ、指定のサブルーチンを割り込み処理として実行させることができます。割り込み処理として実行するサブルーチンは、それぞれ別々に設定でき、特にファンクションキーはF1～F20の20種の割り込みとして使用できます。

キー割り込み用の命令として次のものが用意されています。

on key gosub　　key on/off/stop
…… ファンクションキーによる割り込み処理の定義と制御。

on help gosub　　help on/off/stop
…… HELP キーによる割り込み処理の定義と制御。

on stop gosub　　stop on/off/stop
…… SHIFT ＋ BREAK キーによる割り込み処理の定義と制御。

8 時計機能

コンピュータ内にはバッテリーバックアップによるリアルタイムクロックが内蔵されて、時分秒、年月日、曜日の管理をしています。

時分秒は、システム変数の**ti$**で6桁の文字列で表され、代入を行なうことにより現在時刻を設定することができます。

例） **ti$="1 2 3 4 5 6"**

…12時34分56秒に設定します。

年月日は、システム変数の**date$**で8桁の文字例で表され、代入を行なうことにより日付を設定することができます。

例） **date$="8 5/1 2/2 5"**

…85年12月25日に設定します。

曜日は、システム変数の**day$**で3桁の文字例で表され、代入を行なうことにより曜日設定することができます。

例） **day$="Wed"**

…水曜日に設定します。

0.1秒単位のタイマーがシステム変数の**time**として用意されています。値は0～65535(0～6553.5秒)で、代入を行なうことにより、設定ができます。

時計による割り込み機能

ti$があらかじめ設定している時刻になった場合や、一定間隔で割込みを発生させるインターバルタイマによる割り込みが発生した場合に特定のサブルーチンを割り込み処理として実行させる機能があります。

割り込み処理用の命令としては次のものが用意されています。

on ti$ gosub	……	ti$による割り込み処理の定義。
ti$ on/off/stop	……	ti$による割り込み処理の制御。
interval	……	インターバルタイマの周期設定。
on interval gosub	……	インターバルタイマによる割り込み処理の定義。
interval on/off/stop	……	インターバルタイマによる割り込み処理の制御。

9 サウンド機能

コンピュータ内には音楽用のLSIとしてOPNが内蔵されていて、いろいろな音を出して音楽演奏ができます。

OPN内にはFM音源部とSSG音源部があり、それぞれ3チャンネルの音を出力し、合計6パートでの音楽演奏が可能です。特にFM音源は、複雑な処理が行なえ、様々な楽器音を出すことができます。

音楽演奏は**music**文で行なわれ、演奏データを設定するとすぐに次の処理に進めますので、音楽演奏とプログラム処理が同時に行なわれます。また、演奏の終了時に割り込みをかける機能がありますので、プログラム中のBGMとして常時、音楽演奏をさせておくこともできます。

FM音源の音色の設定は大変複雑なのでBASICでは、あらかじめ30種類の音色を用意していますが、**tone**文により自由に音色の設定ができるようになっています。また、BASICで用意された音色データを取り込む**tone copy**文があり、音色のデータ作りの参考データとすることもできます。

サウンドに関する命令は次のものが用意されています。

music	……	音楽演奏をする。
music init	……	音楽演奏の初期化をする。
music wait	……	音楽演奏の終了を待つ。
tone	……	配列から音色データを設定する。
tone copy	……	BASICで用意されたFM音源の設定データを配列に取り込む。
on music gosub	……	音楽演奏終了時の割り込み処理を定義する。
music on/off/stop	……	音楽演奏終了時の割り込み処理を制御する。
tone lfo	……	トレモロとビブラートの設定をする。
sound	……	OPNのレジスタをセットする。
beep	……	スピーカを一瞬鳴らす。

10 ボイスレコーダの制御機能

ボイスレコーダは、一般のオーディオデッキで使用されている片側2チャンネルの録音方式を採っていますが、右チャンネルをアナログ信号専用に左チャンネルをデジタル信号専用に使用しています。

アナログ信号側は、内蔵マイクや外部にマイクを取り付けての音声録音やコンピュータ内部のライン入力によりコンピュータによる音楽演奏などを録音でき、再生は、内蔵スピーカーまたはイヤホン端子から再生音を聞くことができます。また、別売のモデムホン(MZ-1X19)とボイスコミュニケーションインターフェイス(MZ-1E26)を接続することにより電話回線に対する録音、再生ができます。

デジタル信号側は頭出し信号を録音して、カセットテープの位置制御に使用します。そのほか、BASICの命令ではプログラムやデーターの書き込みは行ないませんが、"**フォーマット&コピー**"ユーティリティを使用してカセットテープへの書き込みが可能です。[コラム] またMZシリーズのほかの機種で作成したデータファイルの入力として使用できます。

> "フォーマット&コピー"ユーティリティについては、オーナーズマニュアルの第5章の1「ユーティリティの使いかた」を参照してください。

カセットのメカニズムは、電磁メカを使用していますので前面にある操作ボタンによるメカニズムの操作はすべてプログラムコントロールが可能で0.1秒単位での動作時間の設定もできます。

ボイスレコーダの制御に関する命令として次のものが用意されています。

init "CMT:" …… 録音入力のマイクとラインの切り換え、操作ボタンによる制御の禁止、オートリワインドやオートプレイ機の設定、プログラムとの同時処理の指定を行ないます。

cmt= …… カセットメカのコントロール、頭出し信号の録音などを行ないます。

cmt(関数) …… カセットの状態を調べる関数です。

apss …… 頭出し信号を見つける命令です。

11 マウスの操作

別売のマウス(MZ-1X10)を取り付けることにより、グラフイック画面上のマウスカーソルを移動させ、座標入力を行なうことができます。

マウスの機能設定

mouse文により、次のようなマウス機能の設定を行ないます。

- マウスの使用開始の宣言と機能の初期化
- マウスカーソルの直接移動。
- マウスカーソルの形状設定。
- マウスカーソルの移動比率の設定。
- マウスカーソルの移動範囲の設定。
- マウスカーソルの表示色の設定。
- マウスの使用終了の宣言。

マウス入力

mouse関数によりマウスからの入力を行ないます。マウスから入力する情報は次のとおりです。

- 現在のマウスカーソル位置。
- 現在のボタン状態。
- ボタンが押されたときの座標。
- ボタンが離されたときの座標。
- 相対移動量。

マウスによる割り込み処理

マウスのボタン操作や移動により割り込みをかけ、特定のサブルーチンを割り込み処理として実行させることができます。割り込み処理に関する命令として次のものが用意されています。

on mouse gosub	……	割り込み処理行の定義。
mouse(n) on/off/stop	……	要因番号nによる割り込み処理の制御。

12 プリンタ制御機能

"プリンタ選択"のユーティリティを使用することにより、各種のプリンタで漢字の印字や画面のコピーを可能としています。

文字の印字

BASICではプリンタへ文字を表示させる命令として次のものが用意されています。

print/p　list/p　list＊/p　search/p　dir/p　tron/p

これらの命令は同機能で画面に出力されるのと同様の書式でプリンタに印字されます。

漢字などの全角文字の印字は漢字印字の可能なプリンタに対しては画面と同様に**kmode**文の設定により印字が可能です。

注）非漢字プリンタを使用して漢字印字を行なわせている場合、1行の途中で**kmode**文により、設定変更を行なわないでください。

スプーラ機能

プリンタの動作は本体の動きに比べてたいへん遅く、連続して印字を行なう場合はプリンタがそれまでのデータをすべて印字するまで待たなければなりません。そこでBASICでプリンタへの印字データがあるとメモリ上のバッファへデータを書き込むだけで、次のプログラム処理に進み、スプーラがプログラムのデータ受取の可能な状態を見てデータを送る作業を行ないます。こうするとBASICの処理とスプーラの作業が並列に行なわれ、BASICはプリンタの動きを待たなくても良くなります。

画面のコピー

画面に表示されている文字や図形をプリンタにコピーする機能として**hcopy**文と COPY キーの操作があります。

コピー機能	hcopy文	キー操作
文字画面のみ	**hcopy 1**	SHIFT ＋ COPY
グラフィック画面のみ	**hcopy 2**	CTRL ＋ COPY
文字とグラフィック画面	**hcopy 3**	SHIFT＋CTRL＋COPY

ファイル装置としてのプリンタ

ファイル処理において文字出力用の装置として"LPT:"の装置名で使用することができます。

プリンタの直接制御

プリンタ自身のコントロールのために、制御コードを直接送る場合は、lpout文を使用します。print/p文の場合は漢字モードの切り換えやシフトJISの変換、ビットイメージでの印字を行なっていますので制御コードが送れない場合があります。

13 RS-232C インターフェイスの制御

コンピュータ内には2チャンネルのRS-232Cインターフェイスをもっています。 本体後部の25ピンコネクタがチャンネルAで9ピンコネクタがチャンネルBとなります。BASIC上では装置名でチャンネルを指定し、チャンネル**A**は**"COM1:"**、チャンネル**B**は**"COM2:"**としています。

BASICでのRS-232Cの操作はファイル装置としての使用や**term**コマンドによりターミナルモードとして行なわれます。

RS-232CのBチャンネルはマウスインターフェイスとしても使用されていますので、マウスと同時にBチャンネルを使用することはできません。(チャンネルBとマウスコネクタは別になっていますのでどちらで使うかの選択は、命令の操作により行なわれます。)

機能設定

RS-232Cの機能設定は、**init "COMn:"**文により行なわれます。設定できる内容は次のとおりです。

- ボーレイト、パリティビット、データビット長、ストップビット長のデータフォーマット。
- X-ON, X-OFFまたはRR-RS信号による通信制御。
- カナ文字の伝送コード。
- DELコードの処理。
- 改行コードの設定。
- 全角文字の伝送コード。
- 通信の終了コード。

ターミナルモード

termコマンドにより、コンピュータを通信端末装置として使用することができます。ターミナルモードでは、画面表示のコントロールのためにエスケープシーケンスがサポートされています。

(機能設定とターミナルモードの詳細はBASIC-M25マニュアルの付録「RS-232Cインターフェイスとターミナルモード」を参照してください。)

14 ハードウェアの直接制御

BASICではメモリの内容を直接読み書きしたり、I/Oポートより直接データの入出力、機械語で書かれたプログラムをサブルーチンとして実行するなどの直接ハードウェアを制御する機能をもっています。これらの制御は、Z-80Bの制御信号やプログラミング、メモリ構成、I/Oポート入出力による内部回路の制御などを理解する必要がありますが、上手に利用するとBASICと合わせて効率的な処理が可能となります。

機械語エリア

BASIC上で機械語プログラムを利用する場合、あらかじめ機械語プログラムを作成するメモリを確保する必要があります。BASICの初期状態ではBASICがすべてのメモリを使用していますので、不用意にメモリを書き換えるとBASICを破壊するおそれがあります。そこで、limit文でBASICで使用しない機械語エリアを指定します。

limit文では、機械語エリアとなる先頭アドレスを指定し、その指定アドレス以後&HFFFF番地までのメモリが機械語エリアとなり、指定アドレスの直前までが、BASICで使用するエリアとなります。したがって機械語エリアを確保するとその分のBASICのフリーエリアが少なくなります。

機械語エリアのアクセス

limit文で確保された機械語エリアは、BASIC内では使用されませんので自由に読み書きが行なえ、機械語プログラムを作成することもできます。

BASICでは機械語エリアのアクセスのために、次の命令が用意されています。

poke …… メモリにデータを書き込む
peek …… メモリよりデータを読み出す。

以上の命令を使用して自由に機械語エリア内にプログラムを作成し、loadコマンドやsaveコマンドでOBJ形式のファイルを作成することもできます。

I/Oポートのアクセス

Z−80BCPUはメモリアドレスのほかにI/Oアドレスをもっていて、コンピュータ内部の制御や周辺機器の入出力に利用しています。

I/Oアドレスに対する入出力命令として次のものが用意されています。

inp@ …… 指定したI/Oアドレスよりデータを読み取る。
out@ …… 指定したI/Oアドレスにデータを出力する。

機械語プログラムの実行

機械語エリアに作成した機械語プログラムは、サブルーチンとしておくと、BASIC上から呼び出して実行することができます。

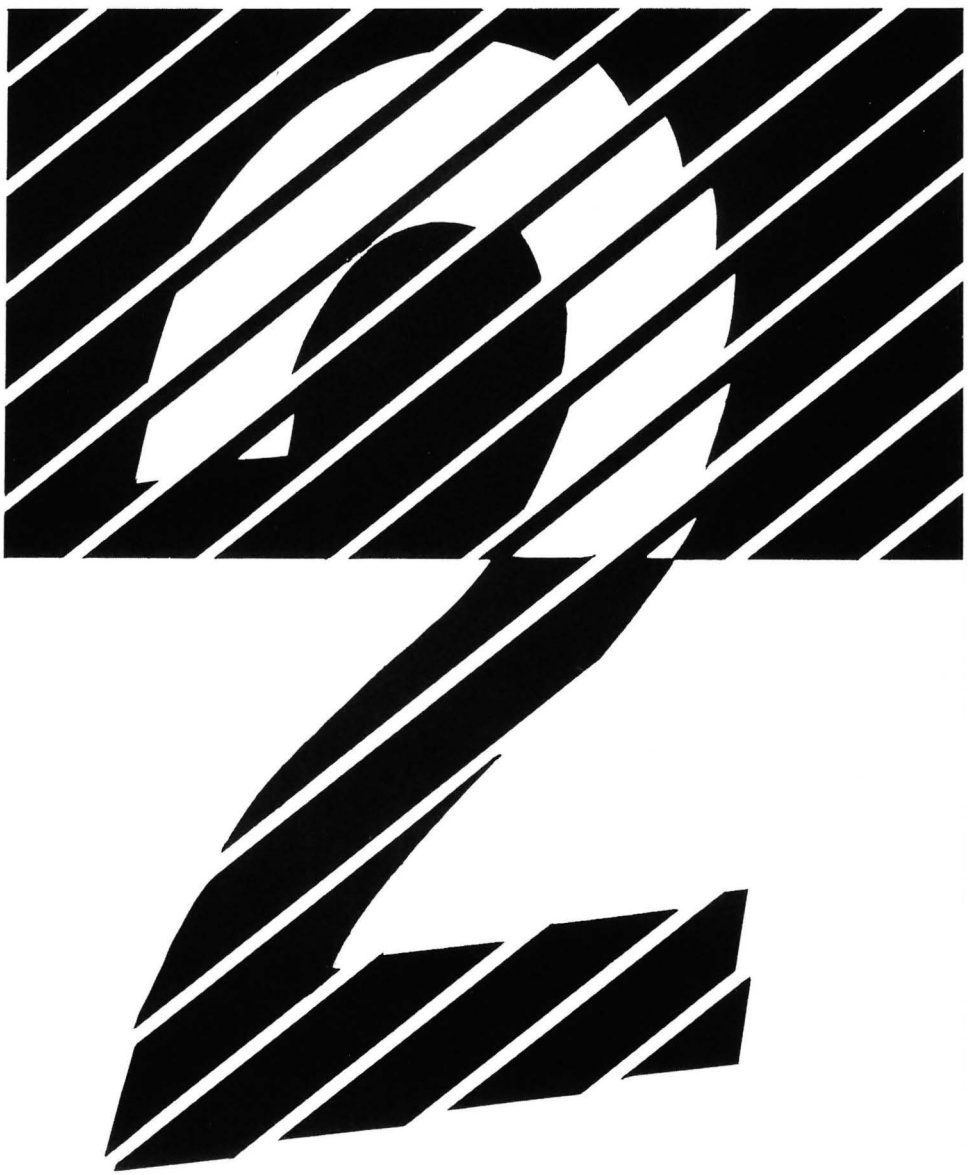
機械語サブルーチンの呼び出しは、usr関数または**call**文により行ないます。**usr**関数は、引数としてデータを機械語サブルーチンに引き渡し、機械語サブルーチンよりデータを受け取りこの関数の値とします。**call**文は1個または0個のデータを変数として引き渡し、機械語サブルーチンでデータの内容を書き換えることにより、BASICに戻ったときに書き換えたデータが変数に代入されます。

機械語プログラムの実行に関する命令として次のものが用意されています。

def usr …… usr関数のアドレスを定義する。
usr …… 機械語プログラムを関数として実行。
call …… 機械語プログラムの実行。

第2部

コマンド・ステートメントの解説



ここでは、各コマンド・ステートメント・関数・システム変数について説明します。

書　式　命令の記述方法を示します。ただし、以下の記号は実際に入力するものではなく、次のような意味があります。

〈　〉　：変数やデータのような、記述するものを表します。
ただし、一番広い意味を表すものです。

(注)

〈変数〉は、〈配列要素〉も含みます。
〈変数〉は、〈数値型変数〉と〈文字列型変数〉を含みます。
〈データ〉は、〈変数〉と〈定数〉を含みます。

〔　〕　：〔 〕で囲まれた部分は省略可能であることを表します。ただしこの後に他の指定が行なわれる場合はカンマやセミコロンの句切記号は必要です。

〔　〕…　：〔 〕…で囲まれた部分は、省略またはくり返し指定が可能であることを表します。

$\left\{\begin{matrix}A\\B\end{matrix}\right\}$　：AまたはBのどちらかを選択します。

省略形　命令内で使用する予約語の最短の省略形を示します。ただし、省略形のないものまたは省略の意味がないものははぶいてあります。
（例　init→ini.）

説　明　命令の働き、各指定の方法などを解説しています。

例　命令の使用例を示します。場合により解説を付けています。

参　考☞　関連する命令や解説部を示します。

サンプル　命令を使用したサンプルプログラムや実行例を示します。

コマンド・ステートメント一覧

RUN （ラン）

プログラムの実行を開始します。

書　式　run〔〈スタート行〉〕

省略形　r.

説　明

- ＜スタート行＞を指定しない場合はプログラムの最初の行から実行され、＜スタート行＞を指定した場合は指定行から実行します。
- ＜スタート行＞を指定しない場合、clr文の機能も実行され、変数や配列および定義関数の消去、操作途中のファイルの放棄、角度単位やデータの読み出し行の初期化などが行なわれます。（clr文参照）
- プログラムの実行が開始されたのちは、プログラム中でのend文、stop文の実行またはプログラムの終了により命令待ちに戻ります。
- プログラム実行中にエラーが発生した場合、エラー内容とエラーが発生した行番号を表示して命令待ちに戻ります。
- プログラム実行中に SHIFT ＋ BREAK キーが押されると、実行を中断し、中断した行番号を表示して命令待ちに戻ります。
- プログラム実行中に BREAK キー押すと実行を一時停止します。この状態で任意の文字キーを押すと、実行を再開します。
- ＜スタート行＞は、行番号またはラベルで指定できます。
- runのあとにファイル名を指定するとフロッピーディスクなどからプログラムを読み込み、実行します。（run　＜ファイル＞参照）

例

```
run          …… プログラムの先頭から実行します。
run 100      …… 行番号100から実行します。
run *開始    …… *開始というラベル名の付いた行から実行します。
```

参　考　run　＜ファイル＞　…… ファイルを呼び出しての実行。

CONT （コンティニュー・continue）

中断したプログラムの実行を再開します。

書　式　cont

省略形　c.

説　明

- プログラム実行中に SHIFT ＋ BREAK キーを押すか、プログラム中のstop文またはstopコマンド、stepコマンドにより実行が中断された状態のときに、中断された所から実行を再開します。
- end文による終了やエラーにより中断したとき、または中断後にプログラムの変更や消去を行なった場合には、プログラムを再開することはできません。
- この命令の実行が可能なときは、Readyのあとにピリオド(.)を表示します。

参　考☞

step （コマンド）　…… プログラムのステップの実行。
stop （コマンド）　…… ブレークポイントの設定。
stop （ステートメント）　…… 実行の中断。

サンプル

```
list
   10 A=5
   20 B=7
   30 stop
   40 C=A+B
   50 print A;"+";B;"=";C
Ready
run
*Stop in 30
Ready.
print A,B,C
 5          7          0
Ready.
cont
 5+ 7= 12
Ready
```

STEP （ステップ）

プログラムのステップ実行を行ないます。

書 式

step〔〈行数〉〕〔：{ run〔〈スタート行〉〕 / goto 〈スタート行〉 }〕

説 明

- contコマンドのように中断されたプログラムを再開しますが、＜行数＞で指定された行数だけ進むと"step in nn"と表示して実行を中断します。nnの部分は中断時の行番号を示しますが、その行自体は実行していません。再度、stepコマンドの実行または、contコマンドの実行で中断した行から実行します。
- ＜行数＞は1～65535の範囲が指定でき、省略した場合は1になります。
- 特別な処理としてあとにrunコマンドまたはgoto文を続けることにより、プログラムの先頭や指定行から実行を開始させることができます。この場合は入力行も1行として扱います。
- あとにrunコマンドまたはgoto文を続けない場合は、実行の再開ができる状態（Readyのあとにピリオド(.)が表示される）でないとcontコマンドと同様にエラーになります。
- ＜スタート行＞は、行番号またはラベルで指定できます。
- この命令は直接実行モード以外では使用できません。

例

```
step        …… 現在位置より1行だけ実行します。
step 5      …… 現在位置より5行進みます。
step:goto 100
    …… 行番号100からの指定ですがコマンド行を1行と数えますので行番号
       100の先頭で中断されます。
```

STOP （ストップ）

プログラムのブレークポイントを設定します。

書　式　stop〈ストップ行〉

省略形　s.

説　明

- この命令で＜ストップ行＞を指定してプログラムをスタートさせると、＜ストップ行＞で指定された行まで進むと実行を中断し、 "stop in nn" とnnの所に行番号を表示して命令待ちに戻ります。このように、プログラムの中断が行なわれる特定の位置をブレークポイントと呼びます。
- 実行を中断した位置は、＜ストップ行＞の先頭になりますので、＜ストップ行＞自体はこの時点ではまだ実行していません。
- 一度ブレークポイントを設定すると、ブレークポイントへ進むごとに実行を中断します。設定されたブレークポイントの解除は、ほかの位置にブレークポイントの設定を移すか、＜ストップ行＞の指定を0にします。
- ＜ストップ行＞は、行番号またはラベルで指定できます。
- プログラム中のstop文の実行によってもブレークポイントとすることができますが、解除を行なう場合は、プログラムの変更になりますので、contコマンドによるプログラムの再開ができなくなります。この命令でのブレークポイントは、プログラムの変更を行なわずに設定や解除ができます。
- 実行を中断した状態から続けて処理を行なう場合は、contコマンドまたはstepコマンドを使用します。
- この命令は直接実行モード以外では使用できません。

例　stop 100 …… ブレークポイントを行番号100に設定します。

参　考　cont …… 中断されたプログラムの再開。
step …… 中断されたプログラムのステップ実行。

NEW （ニュー）

プログラムを消去します。

書　式　new

説　明

- BASIC上でのプログラムをすべて消去します。同時に**clr**文の機能が実行され、変数や配列および定義関数の消去、操作途中のファイルの放棄、角度単位やデータの読み出し行の初期化などが行なわれます。（**clr**文参照）
- **limit**文によってBASICエリアが制限されている場合は、BASICエリアのプログラムのみを消去し、機械語プログラムは消去しません。
- **load**コマンドによるBASICプログラムの読み込み実行時にも、プログラムを読み込む前にこの命令と同じ動作が行なわれます。

例

```
new ……  プログラムを消去します。
9999 new
  ……  プログラム中にも書くことができますが、実行するとその時点でプロ
       グラムが消去されますので注意してください。
```

参　考☞　**delete** ……　プログラムの部分的な消去。

NEW ON （ニューオン）

プログラムエリアを変更します。

書　式　new on〈番号〉

説　明

- BASICの機能の一部を消去してフリーエリアを広げます。newコマンドと同様にBASICプログラムや変数を消去します。
- ＜番号＞は0～5の値で指定します。指定により消去される内容は次のとおりです。

  0 … 電卓やカラーシミュレーションなどのアルゴ機能。

  1 … 音声合成、マウス機能。

  ```
  voice   voice@
  on mouse gosub   mouse on/off/stop   mouse   mouse(関数)
  ```

  2 … 1の内容と辞書ROMによる漢字変換機能。

  3 … 1～2の内容と漢字変換機能とターミナル機能。

  4 … 1～3の内容とグラフィック機能。

  ```
  cblock  color=  priority  view  view@  window  color
  fill  pen  set  reset  line  bline  box  circle  poly
  paint  pattern  symbol  move  roll  roll@
  color replace  map  position  point  posh  posv
  get@  put@  gsave  gload
  ```

  5 … 1～4の内容とプリンタ印字機能。

  ```
  list/p  list/p*  search/p  dir/p  tron/p  print/p
  hcopy  lpos
  ```

- この命令で機能の消去を行なうとBASIC上の命令では元に戻すことはできません。したがって番号1～5の指定は現在の状態より大きな値を指定することはできますが、小さい値を指定することはできません。
- この命令で消去された命令を使用すると、**can't execute error**になります。
- 消去された命令や機能を使用する場合は、**IPL**よりBASICを再起動してください。
- 付属のマスターディスクは出荷時に "**auto-run. s25**" プログラム内で**new on 2**と指定しています。（行番号**990**）**new on 2**で消去される機能をお使いになるときは、"**auto-run. s25**" プログラムを書き換えてBASICを再起動してください。

TRON　（トレース オン・trace on）
TROFF　（トレース オフ・trace off）

プログラムの実行を追跡します。

書　式

tron〔/p〕〔〔〈表示開始行〉〕ー〔〈表示終了行〉〕〕
troff

省略形

t.　　trof.

説　明

- tronコマンドでトレースモードにしてプログラムを実行させると、実行している行番号を[ ]で囲んで表示します。
- ＜表示開始行＞と＜表示終了行＞でトレース範囲を指定した場合、指定範囲内を実行したときにトレース状態を表示します。
- tron/pとすると画面のかわりにプリンタにトレース状態を印字します。
- 解除するときは、troffコマンドを実行します。

例

```
tron
```

…… トレースモードにします。それ以後、実行するとトレース状態を画面に表示します。

```
tron 200-300
```

…… 行番号200から300までの範囲を実行したときにトレース状態を表示します。

サンプル

```
list
   10 T=1
   20 for A=2 to 5
   30   T=T*A
   40 next
   50 print T
Ready
tron
Ready
run
[10][20][30][40][30][40][30][40][30][40][50] 120
Ready
troff
Ready
run
 120
Ready
```

TERM （ターミナル・terminal）

ターミナルモードにします。

書 式

term "COMn:〈ボーレイト〉,〈パリティ〉〈データビット長〉
〈ストップビット長〉〈通信制御〉〈カナ〉〈DEL受信〉〈CR送信〉
〈CR受信〉〈全角文字〉〈エンドコード〉"[,〈モード〉]

説 明

- RS-232Cポートの通信条件や制御信号の設定を行ない、本機をターミナルとして使用します。ターミナルモードの終了は SHIFT ＋ BREAK キーです。
- "COM1:"または"COM:"としたときはチャンネルA、"COM2:"としたときはチャンネルBへの指定になります。"COM2:"はマウス使用時は指定できません。
- RS-232Cの設定は次のようになり、〔 〕内は省略時の値です。

＜ボーレイト＞	75, 150, 300, 600, 1200, 2400, 4800, 〔9600〕, 19200
＜パリティ＞	E　O 〔N〕
＜データビット長＞	7 〔8〕
＜ストップビット長＞	1　2 〔3〕 (1, 1½, 2ビットに対応します。)
＜通信制御＞	X　R 〔N〕
＜カナ＞	S 〔N〕
＜DEL受信＞	N　B 〔D〕
＜CR送信＞	C 〔L〕
＜CR受信＞	C 〔L〕
＜全角文字＞	J　M　P 〔N〕
＜エンドコード＞	文字コードで&H40～5Fまたは&H60～7Fに相当する1文字を&H0～1Fに対応させます。スペースの指定または省略したときはエンドコードは設定しません。

ボーレイトのあとにはカンマ(,)が必要で、ほかの指定は、すべて1文字になります。また、それぞれの指定の省略時はそれぞれ省略値が設定されますが、途中の指定を省略することはできません

- ＜モード＞はFまたはHで指定し、Fは全二重モード、Hは半二重モードになります。
- ターミナルモードではファンクションキーによる機能設定やエスケープシーケンスがサポートされています。
- 通信機能やエスケープシーケンスの詳細はBASIC-M25マニュアルの付録の「RS-232Cインターフェイスとターミナルモード」を参照してください。

例

```
term "COM:1200",H
```

…… COM1:を使用して1200ボーの半二重モードでターミナルモードにします。

MON （モン・monitor）

モニタへ制御を移します。

書　式　mon

説　明

- BASICからモニタへ移り、モニタコマンドによりメモリ内容の表示や変更、または機械語プログラムの実行などが行なえます。モニタコマンドについては、BASIC-M25マニュアルの付録の「モニタコマンド」を参照してください。
- モニタからBASICに戻るには、モニタコマンドの**R**コマンドを使用します。
- **limit**文で最初に機械語エリアを指定しておくと、モニタへ移ったときに、機械語プログラムを使用する場合のメモリ状態になります。

BOOT （ブート）

BASICの処理をやめ、IPLに移ります。

書　式　boot

説　明

- この命令を実行すると**IPL**に移り、電源をONしたときと同じ状態にします。したがって、BASIC言語およびBASICプログラムは消去されます。

参　考　**IPL**(Initial Program Loader)
…… 起動用補助プログラム。（オーナーズマニュアル参照）

LIST （リスト）

プログラムリストを出力します。

書　式　list〔/p〕〔〔〈開始行〉〕ー〔〈終了行〉〕〕

省略形　l.

説　明

- listとするとプログラムのリストを表示します。
- list/pはプリンタにプログラムリストを印字します。
- リストの表示中に BREAK キーを押すと一時停止します。その状態で任意の文字キーが押されると解除され表示を続けます。
- 表示中に SHIFT ＋ BREAK キーを押すと、表示が中止され命令待ちに戻ります。
- ＜開始行＞と＜終了行＞を指定して部分的に出力させることもできます。
- ＜開始行＞と＜終了行＞は、行番号またはラベルのほかにピリオド(.)での指定ができます。

例

```
list  …… すべてのリストを画面に表示します。
list 50  …… 行番号50のみを表示します。
list/p -100
  …… プログラムの先頭から行番号100までのリストをプリンタへ出力します。
list *ABC-
  …… *ABCというラベル名の付いた行からプログラムの最後までを表示します。
list 70-220
  …… 行番号70から行番号220までのプログラムを表示します。
```

LIST 〈ファイル〉(リスト〈ファイル〉)

ファイルに対してリスト出力を行ないます。

書 式 list 〈ファイル指定〉[,[〈開始行〉]−[〈終了行〉]]

省略形 l.

説 明

- ＜ファイル指定＞で指定された装置にプログラムリストを出力します。"FD1:"～"FD4:"と"MEM:"には＜ファイル名＞の指定が必要でBSD形式のファイルとして記録され、"COMn:"へは文字コードで出力されます。"CRT1:"は通常のlistコマンド、"LPT:"はlist/pと同じ働きをします。
- ＜開始行＞、＜終了行＞の指定方法はlistコマンドと同じです。
- ＜ファイル指定＞は次の書式の文字列型データで指定します。
  "[＜装置名＞][＜ディレクトリパス＞]＜ファイル名＞"
  ＜装置名＞……FD1:～FD4:, MEM:, COM1:～COM2:, CRT1:, LPT:
- ＜ファイル名＞は、16バイト以内の半角文字および全角文字で指定します。ただし、":"と"/"の2つの半角文字は使用できません。
- ＜ディレクトリパス＞は階層ディレクトリのパス(道すじ)を指定します。(chdirコマンド参照)
- ＜装置名＞と＜ディレクトリパス＞の指定を省略すると、chdirコマンドで設定されているカレントディレクトリ内でのファイル指定になります。

例

```
list "COM2:"   …… RS-232Cのチャンネルbへリストを出力します。
list "FD1:ABC"
   …… フロッピーディスクへBSD形式のファイルとして記録します。
```

LIST＊　（リスト アスタリスク）

コメント行のみをリストします。

書　式　list＊〔〈ファイル指定〉〕〔〔〈開始行〉〕ー〔〈終了行〉〕〕
list/p＊　〔〔〈開始行〉〕ー〔〈終了行〉〕〕

省略形　l.＊　l./p＊

説　明

- プログラム中でアポストロフィ(')で始まる行のみの内容を行番号とアポストロフィを除いて表示します。行の途中から始まるアポストロフィまたはrem文によるコメントは対象となりません。
- ＜ファイル指定＞を指定するとlist ＜ファイル＞コマンドと同様に指定装置にBSD形式のファイルを作成します。ただし行番号は除かれていますのでloadコマンドでプログラムとして読み込むことはできません。
- ＜開始行＞、＜終了行＞の指定はlistコマンドと同じです。
- list/p＊はプリンタにコメント行のリストを印字します。

参　考　auto＊　……　コメント行の連続入力。

OPTION LIST （オプション リスト）

予約語表示の大文字、小文字を切り換えます。

書 式

option list { ucase / lcase }

省略形

op. l. uc. lc.

説 明

- listコマンドなどで表示されるプログラム中の予約語部分を大文字で表示するか、小文字で表示するかを指定します。 ucaseを指定すると大文字に、lcaseを指定すると小文字になります。
- BASIC起動時はlcaseの小文字での表示になっています。
- 注釈文やダブルクォーテーション(")で囲まれた部分、data文中のデータなどは予約語ではありませんので入力したとおりに出力されます。
- この命令は表示の指定になりますので、入力時は大文字、小文字のいずれでも予約語を入力することができます。
- 変数名や配列名、定義関数名、ラベル名は、つねに大文字で表示されます。

例

```
option list ucase
```

…… 以後のプログラム表示中の予約語を大文字で表示します。

AUTO （オート）

行番号の自動発生を行ないます。

書　式　auto〔〈開始行〉〕〔,〈増分〉〕

省略形　a.

説　明

- この命令を実行すると、自動的に行番号を発生させ、行番号のあとにカーソルを出して入力状態とします。1行分の入力が終わり、[↵] キーが押されると、次の行番号を表示して入力状態になります。
- 発生させた行がすでに使用されていればその内容を表示し、文の先頭にカーソルを表示します。
- この状態は、[SHIFT] ＋ [BREAK] キーを押すことにより解除されます。
- ＜開始行＞と＜増分＞は、それぞれ省略することができます。省略した場合は、それぞれに10が指定されたものとします。
- ＜開始行＞は、行番号またはラベルで指定できます。

AUTO＊ （オート アスタリスク）

アポストロフィ(')付の行番号の自動発生を行ないます。

書　式　auto＊〔〈開始行〉〕〔,〈増分〉〕

省略形　a.＊

説　明

- autoコマンドと同様に行番号を自動的に発生させますが、行番号のあとに、アポストロフィ(')を表示して入力状態になります。アポストロフィは注釈文の始まりを意味します。（rem文を参照）
- この機能は注釈文を続けて入力する場合に便利で、アポストロフィで始まる行のみを表示する命令のlist＊コマンドと組み合わせて簡単なワードプロセッサーとして活用することができます。

参　考　rem …… 注釈文。
list＊ …… コメント行の出力。

EDIT （エディット）

エディットモードにして編集を行ないます。

書　式　edit〔〈編集行〉〕

省略形　e.

説　明

- 画面を消去して、＜編集行＞を中心とするリストを画面全体に表示し、スクリーンエデットが行なえる状態にします。編集キーとして次のキーが特別な働きをし、ほかのキーはいままでと同様の動きになります。ただし CTRL ＋ D キーは何も行ないません。

  ↑　カーソルを上方向に移動します。カーソルが最上行にある場合は画面をスクロールダウン（下方向への移動）させ行番号の小さい方を表示します。

  ↓　カーソルを下方向に移動します。カーソルが最下行にある場合は画面を上に１行分スクロールアップさせ、行番号の大きい方を表示させます。

  SHIFT ＋ ↑　画面を半ページ分（画面に表示されている約半分）スクロールダウンします。

  SHIFT ＋ ↓　画面を半ページ分スクロールアップします。

- エディットモード時は入力した文字の色が変わり、↵ キーが押されて入力が完了すると通常の表示色に戻して再表示を行ないます。
- ＜編集行＞は、行番号またはラベルのほかにピリオド（.）での指定ができます。省略した場合は、プログラムの先頭から行ないます。
- エディットモードは、SHIFT ＋ BREAK キーの操作または、命令の直接実行により解除されます。また、このときに CTRL ＋ D キーの操作と同じ内容が実行され、表示の初期化などが行なわれます。

SEARCH （サーチ）

プログラム中より指定文字列をさがし出して表示します。

書　式　search〔/p〕〈文字データ〉〔,〔〈開始行〉〕ー〔〈終了行〉〕〕

省略形　sea.

説　明

- BASICのプログラム中の指定範囲内で、＜文字データ＞と同じ文字を含む行をさがし出し表示します。
- 画面表示の場合、＜文字データ＞と同じ部分の色が変化します。
- search/pとするとプリンタに印字します。
- さがし出す行の範囲指定を省略するとプログラム全体からさがします。
- 表示中に、 BREAK キーを押すと一時停止することができ、任意のキーを押すと表示を続行します。
- 表示中に SHIFT ＋ BREAK キーを押すと、表示を中止し、命令待ち状態に戻ります。
- ＜文字データ＞にダブルクォーテーション(")が含まれる場合、直接ダブルクォーテーションを書くことができません。この場合、chr$(34)を使って文字式を書くか、または文字変数に代入して、指定を行ないます。
- プログラムはlistコマンドで表示される形により＜文字データ＞と比較しますので、予約語や変数の大文字と小文字などの変化には注意してください。（アルファベットの大文字と小文字は違った文字として扱われます。）
- kmode 1で全角文字を表示する状態のときは、表示される状態にしたがったサーチを行ないます。たとえば、"年"の文字はkmode 0で表示させると"ｲN"と表示します。"年"が含まれるプログラムでsearch "N"を行なうとkmode 1の状態では、"年"の文字は見つかりませんがkmode 0の状態で行なうと"年"のある行を表示します。ただし、画面の表示は"ｲN"となります。
- ＜開始行＞と＜終了行＞は行番号またはラベルのほかにピリオド(.)での指定ができます。

例

```
search "goto"
```
…… プログラム全体の中からgotoの文字を含む行をさがし出し表示します。

```
search "print "+chr$(34)+"ｵﾜﾘ"+chr$(34)
```
…… プログラム全体の中からprint "オワリ"の文字を含む行をさがし出し表示します。

DELETE （デリート）

プログラムの一部を消去します。

書　式　delete〔〈開始行〉〕〔ー〕〔〈終了行〉〕

省略形　d.

説　明

- ＜開始行＞から＜終了行＞までの指定範囲のプログラムを消去します。
- ＜開始行＞のみの指定の場合は、指定された行のみを消去します。
- この命令のあとに指定がない場合はエラーになります。
- 行の指定を省略してー のみの指定の場合はすべてのプログラムを消去します。ただし変数などは消去されません。
- ＜開始行＞と＜終了行＞は、行番号またはラベルのほかにピリオド(.)での指定ができます。

例

```
delete 150-350
```
…… 行番号150から行番号350までのプログラムを消去します。

```
delete -150
```
…… プログラムの最初から行番号150までの部分を消去します。

```
delete 400-
```
…… 行番号400からプログラムの最後までを消去します。

```
delete 150
```
…… 行番号150のみを消去します。

```
delete -
```
…… すべてのプログラムを消去します。

参　考注ぴ　new …… すべてのプログラムの消去。

RENUM （リナンバー・renumber）

プログラムの行番号を付け直します。

書　式　renum〔〈新行番号〉〕〔,〈増分〉〕〔,〔〈開始行〉〕ー〔〈終了行〉〕〕

省略形　ren.

説　明

- プログラムの指定された範囲の行番号を＜新行番号＞から＜増分＞のステップで付け直します。goto文, gosub文, if～goto文, およびon～goto文などの行番号の参照部分も自動的に付け直します。
- 新たに付け直す行番号が65535をこえる場合や実行の順序が入れかわるようなときは、実行できません。
- 行番号を付け直すことにより、1行の長さが255文字をこえる場合、listコマンドによる表示はできますが編集しようとすると255文字以上は入力できませんので注意してください。行を分割するか、小さな行番号に付け直してください。
- ＜新行番号＞と＜増分＞は、それぞれ省略することができます。省略した場合は、それぞれに10が指定されたものとします。
- ＜開始行＞と＜終了行＞は、行番号またはラベルのほかにピリオド(.)での指定ができます。

例

```
renum
```

…… プログラム全体の行番号を10から10ステップで番号を付け直します。

```
renum 1000,10,100-
```

…… 行番号100からを1000から10ステップで付け直します。

LET （レット）

変数にデータを代入します。

書 式　〔let〕〈変数〉＝〈式〉

説 明

- イコール右辺の＜式＞の値を左辺の＜変数＞に代入します。
- ＜変数＞と＜式＞のデータ型は文字型どうし、または数値型どうしでなければいけません。数値型(整数型, 単精度型, 倍精度型)相互の代入は、自動的に型が変換されます。ただし、変換時にオーバーフローが起こったり、代入される変数に対しての誤差が大きくなる場合があります。
- 半角のアルファベットで始まる変数名で型指定記号（**%, !, #, $**）によりデータ型を指定しないときは単精度型として扱われますが、 **def int/def sng/def dbl/def str**の命令により省略時の型を変更できます。
- 漢字または半角文字のカタカナで始まる変数名で型指定記号が省略されたときは、倍精度型変数として扱われます。
- letは、完全省略できますので、＜変数＞＝＜式＞とすることができます。

例

```
A=3+4
```
…… 3＋4の結果の７を左辺の変数Ａに代入します。

```
D#=2.1
```
…… 単精度の値2.1を倍精度に変換して倍精度変数D＃に代入します。この場合、単精度から倍精度に変換されるため、代入値の誤差は単精度の誤差となり、倍精度に対しては大きな誤差になります。倍精度の値として代入させる場合は**D＃＝2.1＃**として値のあとに倍精度を示す記号＃を付けます。

参 考　**def int/sng/dbl/str** …… 型指定の省略時の型宣言。

DEF INT （ディファイン インテジャー・define integer）
DEF SNG （ディファイン シングル・define single）
DEF DBL （ディファイン ダブル・define double）
DEF STR （ディファイン ストリング・define string）

変数の型宣言を行ないます。

書　式

def int 〈変数名の範囲〉[,〈変数名の範囲〉]…　（整数型）
def sng 〈変数名の範囲〉[,〈変数名の範囲〉]…　（単精度型）
def dbl 〈変数名の範囲〉[,〈変数名の範囲〉]…　（倍精度型）
def str 〈変数名の範囲〉[,〈変数名の範囲〉]…　（文字列型）

説　明

- 変数の型は変数名の最後の型指定記号（%, !, #, $）によって指定されますが、半角文字のアルファベットで始まる変数名の型指定を省略すると単精度型として扱われます。この命令により省略時の変数型を別の型にすることができます。
- ＜変数名の範囲＞は、変数名の１文字目のアルファベットで指定します。A-Dとすると、変数名がA, B, C, Dで始まる変数の指定になります。この範囲の指定は半角文字のアルファベットで行い、全角文字による指定はできません。
- 指定された範囲内のアルファベットで始まる変数を別の型で使う場合は、型指定を付ければ、型指定された方が優先します。
- この宣言は、配列宣言における配列名や使用時の個々の要素の参照、定義関数の関数名にも有効となります。
- この宣言は、繰り返して使用することができ、変数名の範囲が重なった場合あとから宣言された方が有効となります。
- この宣言は、newコマンドやclr文またはrunコマンドにより解除され、初期状態に戻ります。（初期状態はdef sng A-Zが宣言されているとみなすことができます。）
- 漢字または半角文字のカタカナで始まる変数名や配列、関数名の型指定を省略した場合は、この命令での宣言には関係なく、倍精度型になります。

例

```
def dbl A-D
```
…… 変数名がA, B, C, Dで始まる変数に型指定がなければ倍精度型とします。

```
def str N,M,X-Z
```
変数名がN, M, X, Y, Zで始まる変数に型指定がなければ文字列型とします。

参　考 変数（第１部　BASICの概要参照）

DEF FN (ディファイン ファンクション・define function)

関数を定義します。

書 式

def FN〈関数名〉(〈引数〉[,〈引数〉]…)＝〈演算式〉

説 明

- 演算内容を定義して新しく関数を作ることができます。
- ＜関数名＞の付け方は変数名と同じ方法で、型指定もできます。(使用時にはFNも合わせてひとつの関数名として扱われます。)
- ＜引数＞は変数名を書きますが、これは＜演算式＞の評価をするためのもので、実際の変数には影響しません。関数名の最後に型指定をすることにより、結果の型が指定できます。
- ＜演算式＞は＜引数＞を使った式を書きますが、引数以外の変数名を使った場合は、定義関数を使用したときの変数の値によって演算が行なわれます。
- ＜演算式＞の中にほかの定義関数を使うこともできます。(定義関数の入れ子型)
- 定義後はFNも含めた関数名で使い、引数として実際のデータを指定すれば、引数に対して演算処理をして、値を返します。
- 定義関数を再定義することはできません。同じ関数名で定義しようとするとエラーになります。
- 半角文字のアルファベットで始まる＜関数名＞や＜引数＞、＜演算式＞中の引数や変数名の型指定を省略すると、def int文などで宣言されている型として扱われます。したがって、定義時(この命令の実行時)と使用時の省略時の型が異なっている場合、異なる関数名として扱われたり、正しい結果が求められない場合がありますので注意してください。このような場合、省略時の型指定を途中で変更しないようにするか、＜関数名＞などに型指定を付けるようにしてください。
- clr文やnewコマンドまたはrunコマンドの実行によりすべての定義を消去します。

例

```
def FNROOT3(X)=X^(1/3)
```

…… 関数名FNROOT3(X)の関数を定義します。(立方根の演算をします。)

```
print FNROOT3(27)
```

…… 定義した関数をprint文で使用します。(結果は3と表示されます。)

```
def FNASCE(X$)=asc(right$(X$,1))
```

…… 関数名FNASCE(X$)で、文字列の最後の文字コードを求める関数を定義します。

DIM （ディメンジョン・dimension）

配列の宣言を行ないます。

書　式　dim 〈配列名〉(〈添字〉〔,〈添字〉〕…)〔,〈配列名〉(〈添字〉〔,〈添字〉〕…)〕…

説　明

- 配列を使用する場合、あらかじめ配列名と添字の最大を宣言する必要があります。添字の最大は32767まで、次元の最大は255次元までですが、実際にはメモリ容量（フリーエリアの制限）や配列全体の大きさ、あるいは１行に書ける文字数などにより制約されます。
- 宣言時の各要素の内容は、数値型は0、文字型は文字長が0のNull（空文字）になります。
- 宣言したときのメモリ使用量は次のとおりです。数値型はデータが代入されても、使用量は変化しませんが、文字型は代入された文字数に応じて使用量が増えます。

整数型	個数＊２＋α　バイト	αは、配列名や次元の管理に使用される分です。（10～数百バイトまで、次元が多くなり大きくなります。）
単精度型	個数＊５＋α　バイト	
倍精度型	個数＊８＋α　バイト	
文字型	個数＊４＋α　バイト	

- １つの配列でメモリの使用量が64キロバイトをこえる宣言はできません。
- 添字の下限は0ですが、option base文により、下限を1に変更できます。
- 宣言されている配列のを再宣言や、宣言されていない配列の参照、または添字の上限をこえた参照はエラーになります。ただし、宣言が行なわれずに添字が10までの配列を参照した場合は、自動的に10までの配列宣言が行なわれます。
- 変数名と同じ配列名を使うことができ、別の物として扱われますが、次元が異なっていても同じ配列名を使うことはできません。
- 宣言された配列は、newコマンドやclr文、またはrunコマンドが実行されたときに消去されます。また特定の配列を消去する命令としてerase文があります。

例

```
dim A(100),B$(100)
```

…… 数値配列Aと文字配列B$で100までの配列宣言をします。

```
dim XDATA(10,10)
```

…… XDATAの配列名で2次元配列を宣言します。

参　考　option base … 添字の下限を設定。

erase … 配列の消去。

OPTION BASE （オプション ベース）

配列の添字の下限を設定します。

書 式　option base {0 / 1}

省略形　op.　ba.

説 明
- 配列の添字の下限を0または1に設定します。この命令はプログラムの実行開始後、配列の宣言を行なうまでに実行してください。配列の宣言後に行なうとエラーになります。
- この設定の解除は、newコマンドやclr文またはrunコマンドの実行により解除され、0が下限になります。

例　option base 1 …… 配列の下限を1に設定します。

参 考　dim …… 配列の宣言。

ERASE （イレーズ）

配列を消去します。

書 式　erase 〈配列名〉[,〈配列名〉]…

省略形　er.

説 明
- 配列名を指定して配列を消去することができます。配列を消去すると消去した配列名で、再び宣言することができます。
- 消去した配列に使用していたメモリが解放されますので、フリーエリアが広くなり別の目的に使用することができます。

参 考　dim …… 配列の宣言。

SEARCH （サーチ）

配列内の一致するデータを捜します。

書　式　search(〈配列名〉,〈文字列式〉[,〈開始位置〉])

省略形　sea.

説　明

- 1次元の文字列において、＜文字列式＞と同じ内容の配列要素を指定された位置よりさがし出し、最初に見つけた位置を値として返す関数です。同じ内容のものが見つからない場合は、結果が-1になります。
- ＜配列名＞は目的の一致する文字を捜す文字配列名で、すでに宣言されている1次元の文字配列でない場合は、エラーになります。
- ＜開始位置＞はさがし始める位置を要素番号で指定します。範囲をこえる場合はエラーになります。また、指定を省略すると配列の最初からさがします。配列の最初はoption base文の指定により0または1になります。

CLR （クリアー・clear）

変数、配列の消去およびBASICエリアの制限を行ないます。

書　式　　clr

説　明

- 変数や配列および定義関数の消去を行ないます。このときにはfor～next文、repeat～until文、while～wend文などのループ処理、gosub～return文の戻り先、if～end if文などの管理をしているスタックも消去されますので、これらの命令の途中でこの命令を実行すると、正常な処理ができなくなるので注意が必要です。
- この命令では、ほかに次の初期化が行なわれます。
  - 操作の途中のファイルを放棄します。書き込み途中のファイルは登録されず、読み出し途中のファイルは閉ざします。(kill)
  - 変数名の型指定を省略すると単精度型として扱われます。(def sng A-Z)
  - 角度の単位をラジアンにします。(option angle radians)
  - 配列の下限値を0にします。(option base 0)
  - データの読み出し位置をプログラムの先頭にします。(restore)
  - エラー処理行の定義を解除します。(on error goto 0)
  - 割り込み処理行の定義と、制御を解除します。(on key gosub文、key off文など)

SWAP （スワップ）

変数の内容を交換します。

書　式　swap〈変数〉,〈変数〉

省略形　sw.

説　明
- 2つの変数の内容を交換します。2つの変数の型は同じ型で指定します。（整数型，単精度型，倍精度型，文字型の区分がされ、たとえ数値型どうしでも型が異なっているとエラーになります。）
- 変数として同じ型であれば配列の要素も指定できます。

例　swap A#,B#　……　倍精度型変数どうしの内容の交換をします。

MID$=（ミドル ダラー イコール・middle$=）

文字列型変数の内容の一部を置き換えます。

書　式　mid$(〈文字列型変数〉,〈開始位置〉[,〈文字数〉])＝〈置き換え文字列〉

説　明
- ＜文字列型変数＞の内容の＜開始位置＞から＜文字数＞（バイト数）分の文字を、＜置き換え文字列＞と置き換えます。
- ＜文字数＞の指定が＜置き換え文字列＞の長さより大きい場合や＜文字数＞を省略した場合は＜置き換え文字列＞すべてが置き換えの対象となります。
- ＜文字列型変数＞の＜開始位置＞より残りの部分が置き換える文字数より少ない場合、もとの＜文字列型変数＞の長さをこえる分は置き換えません。
- この命令の実行では、＜文字列型変数＞の長さは変化しません。

OPTION ANGLE （オプション アングル）

角度の単位を変更します。

書 式

option angle { radians / degrees }

省略形

op. ang. rad. deg.

説 明

- 三角関数やグラフィック命令のcircleやpolyにおいての角度単位として、ラジアンと度のいずれを使うかを設定できます。radiansがラジアン単位、degreesが度単位の指定になります。
- BASICの初期状態はラジアンですが、この命令でdegreesを指定することにより以後は度で扱われます。
- option angleはプログラム中で何回でも指定できます。
- この設定は、newコマンドやclr文またはrunコマンドにより解除され、ラジアン単位の状態になります。

RANDOMIZE （ランダマイズ）

乱数の発生系列を変更します。

書 式

randomize〔〈初期値〉〕

省略形

ra.

説 明

- rnd関数で発生する乱数の初期化を行ないます。
- ＜初期値＞は、-32768～32767の整数型のデータを指定し、同じ初期値を与えると同じパターンで発生する再現性があります。
- ＜初期値＞を省略したときは、BASICで任意の初期値を設定して乱数の初期化をします。
- ＜初期値＞によっては発生周期が短い場合がありますので注意してください。
- BASIC起動時の＜初期値＞は6になっています。

参 考

rnd … 乱数発生関数

REM　(リマーク・remark)

注釈を入れます。

書　式　　rem〔〈注釈〉〕

説　明

- この命令は非実行文でプログラムの実行とは関係なく、プログラムリストを見やすくするための注釈(コメント)を付けます。
- BASICは実行時にrem文があるとそれ以後の文字は無視して次の行に進みます。したがって、remのあとにはどんな文字があっても実行には影響しません。
- rem文はマルチステートメント(コロン(:)で2つの文をつなぐこと)で行の途中から始めることもできますが、そのあとにマルチステートメントで文を書いても実行されません。
- rem文のかわりにアポストロフィ(')を使用して、注釈を付けることができます。
- アポストロフィはrem文と同機能ですが、行の先頭からアポストロフィで始まる注釈行は、list＊コマンドでまとめて表示することができます。また、アポストロフィで始まる注釈行を連続して入力するための行番号を発生する命令としてauto＊コマンドがあります。

例

```
1000 rem ﾒﾆｭｰ表示
```

参　考☞　　list＊　……　アポストロフィで始まるコメント行のみの表示。
auto＊　……　コメント行の連続入力。

PRINT （プリント）

データの表示を行ないます。

書 式

print〔/p〕〔[〈色〉]〕〔〈データ〉〕〔{ , ; } 〈データ〉〕…

省略形

p. p./p または ? ?/p

説 明

- ＜データ＞を画面に表示します。
- print/pはプリンタに＜データ＞を印字します。
- printのかわりに？を使うことができますが、リスト表示をさせるとprintの5文字に変わります。
- ＜色＞は表示色の指定で、省略した場合はccolor文で設定されている色で表示します。また、あとに続く＜データ＞の直前に色指定を行なうこともできます。
- データの句切りとしてセミコロン(;)を使うと、前の＜データ＞に続けて次の＜データ＞を表示します。
- データの句切りとしてカンマ(,)を使うと10文字ごとのタブ位置に進めて、次のデータを表示します。
- 数値データを表示する場合、値が正のときは左側に1個のスペースを表示し、負のときはマイナス(－)記号を左側に表示します。
- プリンタの印字桁数(文字数)を指定する命令としてwidth print/p文があります。
- データの直前にtab関数を使って左からの間隔を設定することができます。
- 文字コードが31以下の文字データを画面に表示しようとすると、それぞれのコードに対応した制御を実行します。(制御内容はBASIC-M25マニュアルの付録のコントロールコード表を参照してください。)

参 考 lpout …プリンタへ制御コードを送る。

PRINT USING （プリント ユージング）

データを指定された書式で表示します。

書　式　print〔/p〕 using〈書式〉;〈データ〉〔{,/;}〈データ〉〕…

省略形　p. us.　p./p us.　または　? us.　?/p us.

説　明

- ＜書式＞にしたがったフォーマットで＜データ＞を表示します。
- 書式の制御文字は次のとおりです。

数値型

＃	数値表示の桁を指定し、出力は桁の範囲内で右づめで表示します。
.	固定小数点の表示で、小数点位置をそろえる指定です。小数点以下の桁数指定もできます。
,	数値を3桁ごとに句切り、そこにカンマ(,)を入れて右づめで表示します。
＋	数値を指定する記号の前後に付けると、その位置に符号を表示します。
－	数値を指定する記号のあとに付けると、数値が負のときに数値のあとにーを表示します。
＊＊	桁の範囲内で空白となる部分ができたとき、その部分を＊で、埋めます。＊＊も表示する値の2桁分の領域を確保します。
￥￥	表示する数値の前に￥記号を付けます。￥￥で、表示する値の2桁分の領域を確保しますが、1桁分は￥の領域として使われます。
＊＊￥	￥￥と＊＊の機能を合わせたものです。＊＊￥で表示する値の3桁分の領域を確保しますが1文字は￥の領域として使われます。
＾＾＾＾	桁指定のあとに付けると指数形式で表示します。
_	制御文字を単に1文字として表示させたいときに使います。_に続く1文字を制御機能をもたない文字として扱います。制御文字以外はそのまま表示します。

文字型

!	文字列の最初の1文字のみを表示します。
& &	&の間に複数のスペースを指定するとスペースと&を含めた桁数で文字を表示します。文字の長さが桁数に満たない場合は左づめにします。
@	@の部分に文字列を代入します。1つの書式中に複数の指定ができますが、あとに続くデータの個数が少ない場合は不足する部分の@以降を表示しません。

サンプル

```
list
   10 rem --- print using ---
   20 A=-1234.45
   30 A$="Suzuki":B$="Taro"
   40 print using "#######";A
   50 print using "####,###";A
   60 print using "####,###.#";A
   70 print using "+###,###.#";A
   80 print using "####,###.#-";A
   90 print using "####,###.#+";6543.21
  100 print using "**##,###";6543.21
  110 print using "¥¥##,###";6543.21
  120 print using "**¥#,###";6543.21
  130 print using "  ####.#^^^^";6543.21
  140 print using "!";B$,A$
  150 print using "& &";B$;A$
  160 print using "&     &";B$;A$
  170 print using "Dear Mr. @ ";A$
Ready
```

```
run
  -1234
  -1,234
  -1,234.4
  -1,234.4
   1,234.4-
   6,543.2+
***6,543
  ¥6,543
**¥6,543
   654.3E+01
TS
TarSuz
Taro   Suzuki
Dear Mr. Suzuki
Ready
```

関数

TAB （タビュレーション・tabulation）

画面の左端よりの文字表示位置を指定します。

書　式　tab(〈数式〉)

説　明

- print文中に用いる関数で、カーソルを現在位置より行の左端から＜数式＞分の間隔を取った位置に移動します。＜数式＞の値は0～255まで指定でき、0が左端になります。
- 現在のカーソル位置が＜数式＞による指定位置より右側にあるときは無効となり、カーソル位置は変化しません。

例

```
print tab(12);"A";tab(20):"B"
```

…… 画面の左端から13文字目に"A"、21文字目に"B"を表示します。

INPUT （インプット）

キーよりデータの入力を行ないます。

書　式　input〔〈コメント〉;〕〈入力変数〉〔,〈入力変数〉〕…

省略形　i.

説　明

- この命令を実行させると、カーソルを点滅させキー入力状態になり、データが入力されて変数に代入します。
- ＜コメント＞は文字定数または文字定数で始まる文字列式で指定します。実行時には＜コメント＞を表示し、そのあとにカーソルを表示します。＜コメント＞を省略すると、"？　"を表示して入力状態となります。
- 入力はカーソルが点滅したあとでデータをタイプし、[↵] キーを押すことにより完了します。＜入力変数＞を複数並べた場合は、カンマで句切って入力します。
- 文字列型変数の入力はダブルクォーテーション(")で囲むのを省略できます。ただし、カンマはデータの句切りとして扱われますのでカンマを含む文字列を入力するときはダブルクォーテーションで囲む必要があります。
- 変数の型や個数が、入力したデータと一致しない場合は、エラーを表示して、再度入力を要求します。
- 何も入力せずに [↵] キーを押すと数値変数には0、文字列変数には文字長が0のNull(空文字)を代入します。ただし、＜入力変数＞が複数のときは句切りのカンマが必要です。

LINE INPUT （ライン インプット）

キーより1行分のデータを入力します。

書 式　line input 〔〈コメント〉;〕〈入力文字型変数〉

省略形　lin. i.

説 明

- この命令を実行させると、カーソルを点滅させキー入力状態になり、データが入力されると、入力データの1行全体を文字型変数に代入します。
- 入力はカーソルが点滅したあとでデータをタイプし、[↵] キーを押すことにより完了します。
- input文と違い入力の途中にカンマ(,)があっても句切りとみなされず、ダブルクォーテーション(")も入力できます。カーソルが最初に点滅した所から行の終りまでを入力します。
- この命令は、入力時には"？　"を表示しません。

INPUT WAIT （インプット ウェイト）
LINE INPUT WAIT （ライン インプット ウェイト）

キー入力の時間制御を行ないます。

書 式　input wait 〈待ち時間〉,〔〈コメント〉;〕〈変数〉,〔〈変数〉〕…
line input wait 〈待ち時間〉,〔〈コメント〉;〕〈文字型変数〉

省略形　i. w.　　lin. i. w.

説 明

- ＜待ち時間＞で設定された時間だけキー入力を待ちます。＜待ち時間＞の単位は0.1秒単位で1～65535までの値で指定します。
- 待ち時間以外の機能はinput文またはline input文と同じですが、この命令のあとにコロン(:)でマルチステートメントにすると、制限時間内に入力された場合は実行され、入力されなかった場合は、コロン以降を実行せずに次の行から実行します。
- 制限時間内に入力されなかった場合の＜入力変数＞の値は以前のままで変化しません。

READ～DATA　（リード～データ）

プログラム中に書かれたデータを変数に読み込みます。

書　式

read 〈入力変数〉〔,〈入力変数〉〕…
data 〈定数〉〔,〈定数〉〕…

省略形

rea.～da.

説　明

- read文は、data文の中に＜定数＞で書かれているデータを＜入力変数＞に1つずつ読み込みます。＜入力変数＞は数値型でも文字型でも指定できますが、変数とデータの型が合わない場合はエラーになります。
- data文はread文で＜入力変数＞に読み取られるデータを書きます。文字型のダブルクォーテーション(")は省略できますが、句切り記号としてカンマ(,)が使われていますのでカンマを含むデータとしたいときは、ダブルクォーテーションで囲む必要があります。また、data文はプログラム中のどこにあってもよく、実行の順番がきても何もせずに次の行に進みます。
- データの読み込みはread文で＜入力変数＞が指定されるごとに、行番号の小さい方から順次data文をさがし、data文の中では左から右へと順番に行なわれます。順番にデータを読み込んで行き、読み取るデータが無くなっても読み込もうとするとエラーになります。
- restore文によって、読み取る行を変更することができ、同じデータを何回も読み直したり、いくつかのdata文を読み飛ばしたりすることができます。
- 現在のデータの読み取り位置を行番号で求めるシステム変数としてdtlがあります。

参　考☞

restore　……　データ読み取り行の変更。
dtl　……　現在のデータ読み取り行を求めるシステム変数。

RESTORE （リストア）

データの読み取り位置を変更します。

書　式　restore〔〈読み取り開始行〉〕

省略形　res.

説　明
- read文によるデータの読み取り位置を変更します。＜読み取り開始行＞を0または省略するとプログラムの最初に位置を移します。
- ＜読み取り開始行＞を指定するとその行のdata文または、それ以後で最も小さい行番号をもつdata文から読み取りを開始します。
- ＜読み取り開始行＞は、行番号またはラベルで指定します。

参　考☞　read～data　……　データを変数に読み込む。

システム変数

DTL （データライン・data line）

read文によるデータの読み取り行を求めます。

書　式　dtl

説　明
- read文によるデータ読み取り位置を記憶しているシステム変数で、値は行番号を示します。
- 値は直前に読み取った行を示しますので、まだデータを読み取っていない状態では0になります。

例
```
10 if dtl>500 then restore
```
……　データの読み取り位置が500をこえた場合は先頭に読み取り位置を移します。

参　考☞　restore　……　データの読み取り位置を変更する。

GOTO （ゴーツー）

指定された行へ実行を移します。

書　式　goto〈ジャンプ先行〉

省略形　g.

説　明
- ＜ジャンプ先行＞で指定された行へ実行が移り、そこからプログラムの実行を続けます。
- ＜ジャンプ先行＞は、行番号またはラベルで指定できます。
- この命令は、go toと離して書くこともできます。

GOSUB~RETURN （ゴーツーサブルーチン~リターン・go to subroutine~return）

サブルーチンを実行します。

書　式
```
gosub〈開始行〉
  ⋮
return〔〈戻り行〉〕
```

省略形　gos.～re.

説　明
- gosub文はgoto文と同様に指定された＜開始行＞へ実行を移しますが、その先にreturn文があると元のgosub文の次の文へ戻ります。
- サブルーチンの中でサブルーチンを実行することもでき、これをネスティング(入れ子型)と呼んでいます。ネスティングの深さ(回数)には制限はありませんが、メモリの残量で制約を受けることがあります。
- return文のあとに＜戻り行＞を指定すると元のgosub文の所以外へ戻ることができます。
- gosub文で呼ばれずにreturn文を単独で実行するとエラーになります。
- ＜開始行＞と＜戻り行＞は、行番号またはラベルで指定できます。
- この命令は、go subと離して書くこともできます。

ON ～ GOTO（オン～ゴーツー）
ON ～ GOSUB （オン～ゴーツーサブルーチン・on～go to subroutine）
ON ～ RETURN （オン～リターン）
ON ～ RESTORE （オン～リストア）
ON ～ RESUME （オン～リジューム）

式の値により参照行を変更します。

書　式

on〈数値式〉{ goto / gosub / return / restore / resume }〈行1〉〔,〈行2〉,〈行3〉…〕

省略形

o.～g.　　～gos.　　～re.　　～res.　　～resu.

説　明

- ＜数値式＞の値が1であれば＜行1＞、2であれば＜行2＞、3であれば＜行3＞…というように、＜数値式＞の値により＜行＞の並びの何番目であるかが決定され、goto文やgosub文などの各機能を実行します。
- ＜数値式＞の小数部は四捨五入が行なわれ、値が0以下または並びの個数をこえる場合は何も実行せず次の文に進みます。
- ＜行＞は、行番号またはラベルで指定できます。

参　考☞

goto …… 指定行へ実行を移す。
gosub～return …… サブルーチンの実行。
restore …… データの読み出し位置の設定。
resume …… エラー処理の終了。

IF～THEN～ELSE （イフ～ゼン～エルス）

条件により処理を変えます。

書　式

if 〈条件式〉〔 { then { 〈文〉〔:〈文〉〕… / 〈行〉 } / goto 〈行〉 / gosub 〈行〉〔:〈文〉〕… } 〕〔else { 〈文〉〔:〈文〉〕… / 〈行〉 } 〕

省略形

if～th. el.

説　明

- ＜条件式＞の条件に合う場合はthenに続く文を実行し、条件に合わない場合はelse以降の文を実行します。
- thenに続く文がgoto文の場合、then goto ＜行＞としますが、thenまたはgotoのどちらかを省略して、goto ＜行＞またはthen ＜行＞とすることができます。then gosub ＜行＞の場合はthenを省略してgosub ＜行＞とすることができます。
- else goto ＜行＞の場合は goto を省略してelse ＜行＞とすることができます。
- ＜条件式＞には関係演算子(＞＜や＝)を使った関係式、論理式、算術式などの結果が数値となる式を使うことができ、式の値が0でなければthen以降、0ならばelse以降の実行と判断します。
- if文の中にif文を書いてネスティング(入れ子型)にすることができます。

  if A=1 then if B=1 then Ⓐ else Ⓑ else Ⓒ

  このような場合ⒶはA＝1とB＝1の条件のときに実行、ⒷはA＝1とB≠1のときに実行、ⒸはA≠1のときに実行されます。

ELSE IF～THEN……END IF (エルス イフ～ゼン……エンド イフ)

IF～THEN～ELSE機能の複数行型式を行ないます。

書　式

```
if 〈条件式〉 then
  〈文〉
   ⋮
[else if 〈条件式〉 then
  〈文〉                    ]…
   ⋮
[else                       ]
  〈文〉
   ⋮
end if
```

省略形

el. if～th. ……en. if

説　明

- 行の最初から始まるif文で、thenのあとに何も書かず行が終わる場合、条件に合う処理を複数行に渡って書くことができます。条件に合わない処理は、elseだけ書かれている行の次から複数行に渡って書くことができ、終了はend if文になります。
- if～then文とend if文は1対1で対応させる必要があります。
- else if～then文を使うことにより次々に条件ごとの処理を書いて行くことができます。(次ページ参照)
- 複数行型式でのif～then文, else文, else if～then文, end if文のそれぞれの文は左右にマルチステートメント(コロン(:)で2つの文をつなぐこと)でほかの文を書くことはできません。

```
if  条件A  then
    処理1
else
    処理2
    if  条件B  then
        処理3
    else
        処理4
    end if
    処理5
end if
```

```
if  条件A  then
    処理1
else if  条件B  then
    処理3
else
    処理4
end if
```

〔例1〕

例1の左は条件のネスティングを使用したもので、右のelse if～thenと対比させたものです。各処理と条件の関係は次のようになります。

処理1	条件A…○	条件B…無関係	
処理2	条件A…×	〃 …無関係	○…条件に合う
処理3	条件A…×	〃 …○	×…条件に合わない
処理4	条件A…×	〃 …×	
処理5	条件A…×	〃 …無関係	

右側の各処理も左側と同じ条件での実行となりますが、else ifを使用しているのでend ifは1つですみます。

例1は処理の流れを理解するためのもので実際には次のような使い方をします。

```
if   <条件1>  then
   処理1
else if  <条件2>  then
   処理2
else if  <条件3>  then
   処理3
else if  <条件4>  then
   処理4
     ⋮
end if
```

〔例2〕

条件1に合う場合は処理1、条件2に合う場合は処理2…というように順に条件をチェックして各処理を行なう場合に便利な機能です。

FOR～NEXT （フォア～ネクスト）

forとnextの間の文を繰り返し実行します。

書　式

for〈カウント変数〉＝〈初期値〉 to〈最終値〉〔step〈増分〉〕

⋮

next〔〈カウント変数〉〕〔,〈カウント変数〉〕…

省略形

f.　～　n.

説　明

- for 文は＜カウント変数＞を＜増分＞ずつ増やしながら＜最終値＞をこえるまでnext 文の間を繰り返し実行します。
- ループの終了は、next文で＜カウント変数＞に＜増分＞を加え、＜最終値＞をこえるかの判断を行ないますので＜最終値＞が＜初期値＞より小さい場合でも繰り返し部分を1回は実行します。
- stepの指定を省略すると増分として1が指定されたとみなします。
- このループを終了したあとの＜カウント変数＞の値は、最後の繰り返し時の変数値に＜増分＞を加えた値になります。たとえば、for A=1 to 10のループの終了後のAの値は、最終の繰り返し値10に＜増分＞1を加えた値11になり、for A=1 to 10 step 2の場合は、最終の繰り返し値が9でループ終了後のAの値は11になります。

REPEAT～UNTIL （リピート～アンティル）

条件により繰り返し部分に戻るかを判断します。

書　式

repeat
　⋮
until〈条件式〉

省略形

rep.～u.

説　明

- untilのあとの＜条件式＞に合わなければrepeat文の次の文まで戻ります。
- 繰り返すかの判断はuntil文で行なわれますので、１回は繰り返し部分を実行します。
- repeat文とuntil文は1対1で対応し、ネスティング(入れ子型)させることができますが、ループの中から外へgoto文により飛び出した場合はループが終了したとみなします。
- repeat文実行時に対応するuntil文がない場合、またはrepeat文を実行せずにuntil文を実行した場合はエラーになります。

WHILE～WEND （ホワイル～ホワイル エンド・while～while end）

whileの条件に合えばwend間を繰り返します。

書　式

while〈条件式〉
　⋮
wend

省略形

wh.～we.

説　明

- ＜条件式＞の条件に合えばwhile文とwend文の間を実行し、条件に合わない場合はwend文の次の文から実行します。
- while文で繰り返すかの判断をしますので、繰り返し部分を1回も実行しない場合があります。
- while文とwend文は1対1で対応し、ネスティング(入れ子型)させることができますが、ループの中から外へgoto文により飛び出した場合はループが終了したとみなします。
- while文実行時に対応するwend文がない場合や、while文を実行せずにwend文が実行された場合は、エラーになります。

STOP （ストップ）

プログラムの実行を中止します。

書 式　stop

省略形　s.

説 明

- プログラムの実行を中止して、命令待ち状態にします。
- このときBASICは次のメッセージを出します。

  Stop in nn　　… nnは中止した行番号
- end文と異なりファイルを閉じたり各種の初期化は行なわれません。
- この命令で実行が中断された状態から、contコマンドやstepコマンドにより実行を再開することができます。ただし、実行の中断時にプログラムの変更を行なうとend文の実行と同様の処理が行なわれ、contコマンドやstepコマンドによるプログラムの再開ができなくなります。
- この命令で実行を中断するときに割り込み処理の制御(key on/off/stopなど)がonになっている場合、stop状態にして中断します。contコマンドやstepコマンドによる実行再開により、割り込み処理の制御がonになります。

END （エンド）

プログラムの実行を終了します。

書 式　end

説 明

- end文を実行するとReadyを表示して命令待ちに戻ります。
- この命令の実行で次の処理が行なわれます。
  - 操作途中のファイルをすべて放棄します。書き込み途中のファイルは登録されず、読み出し途中のファイルは閉ざします。(kill)
  - データの読み出し位置をプログラムの先頭にします。(restore)
  - エラー処理行の定義を解除します。(on error goto 0)
  - 割り込み処理行の定義と、制御を解除します。(on key gosub文、key off文など)
- プログラムの最後にend文を省略しても実行を終了しますが、この場合は以上の処理は行ないません。

WAIT （ウェイト）

プログラムの実行の一時停止をします。

書　式　wait〈停止時間〉

省略形　w.

説　明

- この命令を実行すると、〈停止時間〉で指定された時間が経過した後に次の処理に進みます。
- 〈停止時間〉は、0.1秒単位で1～65535の値で指定します。
- この命令で一時停止状態のときは、すべての割り込み制御はstopになり、停止状態で割り込みであった場合は、この命令の終了後に割り込み処理を実行します。

例

```
100 wait 100
```
…… 10秒間実行を停止します。

関数

SUM （サメーション・summation）

式の総和を求めます。

書　式　sum(〈変数〉,〈初期値〉,〈最終値〉,〈式〉)

説　明

- ＜変数＞を＜初期値＞から＜最終値＞まで変化させた式の総和を返します。

$$\sum_{\text{変数}=\text{初期値}}^{\text{最終値}} \text{式}$$

- ＜初期値＞と＜最終値＞は整数型で、小数点以下の部分があれば四捨五入します。また実行後の変数値は保証されませんので注意してください。
- ＜式＞は数値および文字式が可能です。
- ＜初期値＞より＜最終値＞が大きい場合には1度だけ〈式〉を実行しますが、結果が0またはNull(空文字)になります。

例

```
print sum(A,1,10,A*A)
```
…… 1～10のそれぞれの値を2乗した合計、385を表示します。

```
print sum(A,65,90,chr$(A))
```
…… 文字コードが65～90までのアルファベット、A～Zを並べた文字を表示します。

OPTION SCREEN （オプション スクリーン）

グラフィックV-RAMの使用目的を設定します。

書　式　option screen〈機能指定〉

省略形　op. sc.

説　明

- グラフィックV-RAMの一部を表示用以外のメモリディスクやプリンタ・スプーラ用メモリとして使用することができます。この命令ではV-RAMのどこまでを表示用として使用するかを設定します。
- ＜機能指定＞は1～4の値で指定します。指定による表示用とメモリディスクやプリンタスプーラ用の使用比率とそのときに設定可能なグラフィックモードは次のようになります。

機能	メモリブロック				320×200ドット		640×200ドット	640×400ドット	
指定	1	2	3	4	16色	256色	16色	4色	16色
1	G	G	G	G	○	○	○	○	○
2	G	G	M	M	○	○	○	○	×
3	G	M	M	M	○	×	×	×	×
4	M	M	M	M	×	×	×	×	×

G…グラフィック表示として使用　　M…メモリとして使用

○…設定可　　×…設定不可

メモリブロックは次のとおりです。

ブロック1 (32K)	ブロック2 (32K)	ブロック3 (32K)	ブロック4 (32K)

標準V-RAM　(64Kバイト)　　増設V-RAM　(64Kバイト)

増設V-RAMとは別売の増設ビデオRAMボード（MZ-1R27）のことです。

- この命令の実行時にメモリディスクやスプーラが設定されているとエラーになります。あらかじめinit "MEM:0,0"として設定を解除してください。
- ＜機能指定＞の4はすべてのグラフィックV-RAMをメインメモリと同様に扱い、テキストや配列などのフリーエリアとして使用します。したがって、4が指定されると、それ以後1～3への変更、グラフィック命令はエラーになり、スムーススクロールができなくなります。
- ＜機能指定＞で1～3が指定されると、グラフィック表示が320×200ドットの16色モードに変わります。(init "CRT2:")
- BASIC起動時の〈機能指定〉は1になります。

参　考　init "MEM:" …… メモリディスクとプリンタスプーラの設定。

INIT "CRT2:" (イニシャライズ シー アール ティー 2・initialize C,R,T,2)

グラフィック表示の初期化を行ないます。

書　式　　init "CRT2:〔〈横ドット数〉,〈縦ドット数〉,〈表示色数〉〕"

説　明

- グラフィック表示の解像度と同時表示色を設定します。
- 解像度は横と縦のドット数で指定し、＜横ドット数＞は320または640、＜縦ドット数＞は200または400で指定します。ただし、200ラスターのディスプレイを使用しているときは縦の400ドットは指定できません。
- 同時に何種類の色を表現できるようにするかを＜表示色数＞で指定します。＜表示色数＞は、4、16、256の3種類です。ただし4色は640×400ドット専用で、256色は320×200ドット専用になります。また、640×400ドットで16色の指定は別売の増設ビデオRAMボード(MZ-1R27)を取り付けてグラフィックV-RAMを128Kバイトに増設しておく必要があります。
- ＜表示色数＞が256のときの文字表示は64色に設定され、桁数は40桁になります。
- ドット数や表示色数をすべて省略してinit "CRT2:"とすると、320×200ドットで16色に設定されます。また、BASIC起動時もこの設定になります。
- この命令の実行により次の初期設定を行ないます。
  ウィンドウ、ビューポートや表示出力の初期化。（view:view@:window:cblock:roll @）
  グラフィックポインタ座標(0,0)の指定。（position 0,0）
  screen, color機能の初期化。
  パレットやプライオリティの初期化。
- この命令の実行では、グラフィックV-RAMの消去は行なわれませんが、256色と16または4色の切り換えが行なわれた場合は文字画面が初期化されます。
- グラフィックV-RAMの増設やoption screenの設定の関連で使用できないグラフィックモードを指定した場合はエラーになります。

この命令により指定可能な組み合わせ

〈横ドット数〉	〈縦ドット数〉	〈表示色数〉	備　考	
320	200	16		
		256	文字64色40桁	
640		16		
	400	4		400ラスター
		16	V-RAM増設要	ディスプレイ要

INIT "CRT1:" (イニシャライズ シー アール ティー 1・initialize C,R,T,1)

文字画面の初期化を行ないます。

書　式

init "CRT〔1〕:〔〈桁数〉〕〔,〈行数〉〕〔,〈フォント〉〕〔,〈スクロール〉〕"

説　明

- 文字画面に対して、桁数、行数、文字の種類、スクロールの方法を指定し、初期化を行ないます。このとき文字画面の消去も行なわれます。
- ＜桁数＞は40または80で指定しますが、64色表示(グラフィックが256色)のときは40桁以外の指定はできませんので、80桁を指定しても40桁になります。
- ＜行数＞の指定は12、20、25の指定ができます。
- ＜フォント＞は、半角文字の字体を8×8のドットにするか、8×16ドットにするかを指定します。0の場合は8×8ドット、1の場合は8×16ドットになります。この指定は、400ラスターのディスプレイを使用して、20、25行モードのときに有効となります。200ラスターのディスプレイを使用して20、25行モードのときは1が指定されても8×8ドットの表示になります。また、いずれのディスプレイを使用しても12行モードのときは8×16ドットになります。

ディスプレイと行数により表示可能な半角文字

使用ディスプレイ	＜行数＞	半角文字	
		8×8	8×16
400ラスター	12	×	○
	20/25	○	○
200ラスター	12	×	○
	20/25	○	×

- ＜スクロール＞は、スクロールの方法を指定します。0の場合は行単位のラインスクロール、1の場合はドット単位のスムーススクロールになります。スムーススクロールは200ラスターディスプレイ使用時には1ラスター単位、400ラスターディスプレイ使用時には2ラスター単位のスクロールになります。グラフィックの表示色が4または16色のときに40桁の指定をしたときはスムーススクロールは指定できません。

グラフィック＜表示色数＞	文字表示色数	〈桁数〉	＜スクロール＞	
			0	1
16/4	8	80	○	○
		40	○	×
256	64	40	○	○

- スムーススクロール時は、**console**文によるスクロール範囲の指定や**width**文での桁数の変更はできません。また、画面の最下行は表示には使用されず、ファンクションキーの表示や漢字入力はできなくなります。
- スムーススクロール時に**view@**文でグラフィック表示のディスプレイ出力を行なわない状態にすると、スムーススクロールが行なわれずラインスクロールと同じになりますが、グラフィック出力が行なわれるとスムーススクロール状態になります。
- すべての指定を省略すると文字表示色が8色の場合は次のように設定します。

  init "CRT1:80,25,1,0"　…400ラスターディスプレイ使用時。

  init "CRT1:80,12,0,0"　…200ラスターディスプレイ使用時。

  文字表示色が64色のときは桁数を40として他は同じ値です。
- 部分的な省略を行なった場合は、省略部分に上記の値を設定します。
- この命令の実行により次の初期設定が行なわれます。

  文字画面の消去。(cls 1)

  文字表示色の初期化。(**ccolor**)

  文字属性の初期化。(cgen:cflash:crev)

  文字入出力画面の初期化。(screen 0,(0))

  スクロール範囲の初期化。(console)

  文字画面の出力範囲の初期化。(**console@**)

  カーソルON。(**cursor ,,2**)

INIT "CRT3:" (イニシャライズ シー アール ティー・initialize C,R,T,3)

デジタル方式のモノクロディスプレイ出力を設定します。

書　式　**init "CRT3:**〔〈グラフィック出力〉〕〔**,**〈反転〉〕**"**

説　明

- この命令は、デジタル方式のモノクロディスプレイを使用したときに有効になり、グラフィック画面の出力と白黒の反転について設定をします。
- <グラフィック出力>は、**0**を指定するとディスプレイに出力され、**1**を指定すると、グラフィック画面は出力されず、ディスプレイには、文字のみを表示します。
- <反転>は白黒の反転を指定します。0を指定すると通常の状態、1を指定するとリバース(白黒反転)表示になります。
- <グラフィック出力>または、<反転>を省略すると、省略値は現在の状態となります。両方の指定を省略してinit "CRT3:"とした場合はどちらも0が設定され初期状態になります。

SCREEN （スクリーン）

文字およびグラフィック画面の入出力設定を行ないます。

書 式

screen 〔Ti〕〔,(To₁〔,To₂〕)〕〔,Gi〕〔,(Go₁〔,Go₂〕)〕〔,Pi〕〔,(Po₁〔,Po₂〕)〕

Ti … 〈文字入力〉　　To₁, To₂ … 〈文字出力〉

Gi … 〈グラフィック出力〉　Go₁, Go₂ … 〈グラフィック出力〉

Pi … 〈入力プレーン〉　　Po₁, Po₂ … 〈出力プレーン〉

省略形

sc.

説 明

- 文字画面やグラフィック画面で複数の面（スクリーン）をもった場合、描画する入力面とディスプレイに表示する出力面の指定および重ね合わせの指定を行ないます。また、グラフィック画面はプレーンごとの入出力も設定できます。
- 表示モードによる画面数は次のようになります。

横サイズ		縦サイズ	表示色	画面数	プレーン数	重ね合せ
文字表示	40	12・20・25	8	2	—	○
			64	1	—	×
	80	12・20・25	8	1	—	×
グラフィック表示	320	200	16	2(4)	4	0と1（または2と3）
			256	1(2)	8	×
	640	200	16	1	4	×
	640	400	4	1	2	×
			16	(1)	4	×

（ ）内は別売の増設ビデオRAMボード（MZ-1R27）を取り付けてグラフィックV-RAMを128Kバイトに増設した場合です。

- 文字表示は8色の40桁時に2面もつことができ、画面番号は0と1になります。
  ＜文字入力＞はprint文などで文字の描かれる面を画面番号で指定します。
  ＜文字出力＞はディスプレイ上に表示する面を指定し、0または1の1面か0と1の2面を同時に重ね合わせて表示させることができます。重ね合わせ表示でドットの重なる部分は画面番号0の方が優先表示されます。
  ＜文字出力＞の指定

  screen ,(0)　… 文字画面0のみ表示。
  screen ,(1)　… 文字画面1のみ表示。
  screen ,(0, 1)　… 文字画面0と1の重ね合わせ表示。

- ＜グラフィック入力＞はグラフィック描画命令により図形の描かれるスクリーン(面)を指定します。入力面に設定された面をアクティブスクリーンと呼びます。＜グラフィック出力＞はディスプレイ上に表示するスクリーンを指定します。スクリーン番号は320×200ドットモードでV-RAMを増設時には0～3の4面をもつことができ、0と1または2と3の重ね合わせが可能となります。グラフィックを重ね合わせる場合の優先順位はpriority文で指定します。

  ＜グラフィック出力＞の指定

  screen ,,,(＜スクリーン番号＞)　　…1面のみ出力

  screen ,,,(0, 1) または **(2, 3)**　　…2面の重ね合わせ出力

- ＜入力プレーン＞はアクティブスクリーンの図形が描かれるプレーンを指定し、これをアクティブプレーンと呼びます。＜出力プレーン＞は＜グラフィック出力＞に指定されたスクリーンのディスプレイ上へ表示されるプレーンを指定します。256色のプレーン指定は512色のコードではなく、物理的な8プレーンを0～255の値で指定します。8プレーンの出力色の割り当てはcblock文で行ないます。

16色表示でのプレーン指定

指定	I G R B	指定	I G R B	指定	I G R B	指定	I G R B
0	0 0 0 0	4	0 1 0 0	8	1 0 0 0	12	1 1 0 0
1	0 0 0 1	5	0 1 0 1	9	1 0 0 1	13	1 1 0 1
2	0 0 1 0	6	0 1 1 0	10	1 0 1 0	14	1 1 1 0
3	0 0 1 1	7	0 1 1 1	11	1 0 1 1	15	1 1 1 1

I, G, R, Bの各プレーンで1となる部分が入出力するプレーンになります。

256色表示でのプレーンの並び

2 進 数	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
プレーン	I_1	G_1	R_1	B_1	I_0	G_0	R_0	B_0

- 各指定の省略時は現在の設定値となりますが、すべての指定を省略した場合または初期設定は次のようになります。

  screen **0, (0), 0, (0), 15, (15)**　　…16色モード

  screen **0, (0), 0, (0), 255, (255)**　　…256色モード

CLS （クリア スクリーン・clear screen）

画面表示の消去を行ないます。

書　式　　cls〔〈画面指定〉〕

説　明

- ＜画面指定＞で指定された画面の消去を行ないます。＜画面指定＞は1～3の値で行ない機能は次のとおりです。

  1または省略　…　文字画面のみ消去します。
  2　…　グラフィック画面のみ消去します。
  3　…　文字およびグラフィック画面の両方を消去します。

- 文字画面の消去はscreen文で設定された＜文字入力＞面の、console文で設定されているスクロール範囲になります。
- グラフィック画面の消去(範囲)はscreen文で設定されているアクティブスクリーンのアクティブプレーンでview文で設定されたビューポート内の範囲になります。

例　　cls 3　……　文字とグラフィックの画面を消去します。

WIDTH （ウィドス）

文字の表示桁数や行数を設定します。

書　式　width〔〈桁数〉〕〔,〈行数〉〕

省略形　wi.

説　明

- 画面に表示する文字の〈桁数〉と〈行数〉の設定を行ないます。〈桁数〉は横方向の文字数で40または80、〈行数〉は縦方向の文字数で12、20、25の指定ができます。
- 文字表示色が64色のときは40桁以外の指定はできません。
- 文字表示色が8色のときに40桁の指定を行なうと2つの文字画面を使用することができます。2つの文字画面の切り換えはscreen文で行ないます。（実行直後はscreen 0,(0)になります。）
- 〈桁数〉または〈行数〉のいずれかを省略することができ、省略値は現在の桁数または行数になります。
- この命令を実行するとconsole文の設定は解除され、画面を消去（cls 1）します。
- スムーススクロールの設定がされている場合は、桁数の変更をすることはできません。
- ＜行数＞を12行にすると半角文字は8×16ドットで表示されます。20または25行のときは、200ラスタディスプレイを使用している場合8×8ドット、400ラスタディスプレイを使用している場合は、init "CRT1:"の＜フォント＞で設定されたドット数になります。

例　`width ,20` …… 桁数を変更せずに、行数を20行にします。

参　考☞　init "CRT1:" …… 文字画面の初期化。

CONSOLE （コンソール）

文字のスクロール範囲を指定します。

書　式　console〔[〈開始行〉],〈行数〉[,[〈開始桁〉],〈桁数〉]]

省略形　cons.

説　明

- 文字のスクロール範囲を指定します。指定後は文字の表示や消去(cls 1)、スクロールが、その範囲内で行なわれます。
- カーソルの上下左右の移動は範囲内に限られていますが、cursor文により設定範囲外へカーソルを移動させることができます。
- 〈開始行〉または〈開始桁〉を省略したときはそれぞれ現在の設定値になります。
- 最下行にファンクションキーを表示するモード(klist 1またはklist 2)のときに、最下行を範囲に含めるとファンクションキー表示を行ないません。また、漢字モード(kmode 1)のときに最下行を範囲に含めている場合は、漢字入力を行なうことはできません。
- すべての指定を省略すると現在の表示モードでの最大範囲になります。ただし、最下行はファンクションキー表示や漢字入力に使用されますので、漢字モード(kmode 1)やファンクションキー表示モード(klist 1またはklist 2)のときは最下行は範囲に含まれません。
- init "CRT1:"によりスムーススクロールモードに設定されている場合は、この命令を実行することはできません。
- この命令による設定はinit "CRT1："やwidth文の実行により解除され、スクロール範囲は最大になります。

例

```
console 0,25,0,80
```

…… 文字画面が80桁、25桁モードのときに最大の範囲をスクロール範囲とします。

参　考

init "CRT1:" …… 文字画面の初期化。

width …… 文字画面の最大桁数と最大行数を設定する。

CONSOLE@ （コンソール アットマーク）

文字画面のディスプレイの出力範囲を設定します。

書 式　console@ 〔〔〈開始行〉〕,〈行数〉〔,〔〈開始桁〉〕,〈桁数〉〕〕

省略形　cons.@

説 明

- ディスプレイに表示される文字表示の範囲を指定します。この命令は、画面への信号を制御しますので、範囲外の部分は表示されないだけでprint文による文字の書き込みやスクロールはconsole文で設定されたスクロール範囲で行なわれます。
- ＜行数＞または＜桁数＞を0にすると文字画面のディスプレイへの出力は行なわれません。
- スムーススクロールのときに、最下行を出力範囲に指定しても、最下行は出力範囲に含まれません。
- この命令による設定は、init "CRT1:"やwidth文の実行、または CTRL ＋ D キーの操作により初期化され、文字画面全体を出力します。
- すべての指定を省略すると初期化され、全体を出力します。

例　console@ ,0 …… 行数が0なので文字画面が表示されなくなります。

参 考☞　view@ …… グラフィック画面のディスプレイ出力範囲の設定。

サンプル

```
 10 rem --- console@ ---
 20 init "CRT1:80,25"
 30 for A=1 to 1920:print "C";:next
 40 for A=1 to 5
 50   for X=80 to 0 step -1
 60     Y=int(X*25/80)
 70     console@ 0,Y,0,X
 80   next : wait 5
 90   for X=79 to 0 step -1
100     Y=int(X*25/80)
110     console@ Y,25-Y,X,80-X
120   next : wait 5
130 next
```

CCOLOR （キャラクタ カラー・character color）

画面の表示色を設定します。

書　式　ccolor〔〈文字カラー〉〕〔,〈ボーダーカラー〉〕〔,〈バックモード〉〕

省略形　cc.

説　明

- ＜文字カラー＞は画面に表示する色を指定します。指定以後は指定色が文字色となります。
- ＜ボーダーカラー＞は文字やグラフィックのドットのない部分の色を指定します。グラフィックが16色または4色モードのときはcolor＝文のパレットコード0の色と同じ扱いになります。
- ＜バックモード＞を1とすると文字のバック部分が不透明となり、グラフィック表示が見えなくなります。0の指定により解除され、グラフィック表示が見えるようになります。これはグラフィックを消去せずに文字画面を見るときに利用できます。
- 各設定値を省略した場合、現在の設定値のままになります。
- init "CRT1:"文の実行や CTRL ＋ D キーの操作によっても初期設定が行なわれます。
- 設定値の範囲と初期値は次のとおりです。

表示色		文字	ボーダー
グラフィック16色(4色)	範囲	0～7	0～15
文字8色	初期値	7	0
グラフィック256色	範囲	0～&O777	0～&O777
文字64色	初期値	&O666	0

CCOLOR@ （キャラクタ カラーアットマーク）

指定範囲の文字色を変更します。

書　式　ccolor@　X_1, Y_1, X_2, Y_2, 〈色コード〉

省略形　cc. @

説　明

- 文字座標(X_1, Y_1)と(X_2, Y_2)を対角とする範囲に表示している文字の色を〈色コード〉で指定した色に変更します。
- 座標は、(X_1, Y_1)で左上、(X_2, Y_2)で右下を指定しますので、$X_1 \leqq X_2, Y_1 \leqq Y_2$でないとエラーになります。
- 〈色コード〉は8色モード時0～7、64色モード時0～&O777の値で指定できます。

例

```
ccolor@ 0,0,10,10,6
```

参　考☞　cflash@　……　指定範囲の文字を点滅させる。
crev@　……　指定範囲の文字を直転させる。

サンプル

```
 10 rem --- ccolor@  crev@ cflash@ ---
 20 init "CRT2:":cls 2:color=(0,3)
 30 init "CRT1:"
 40 for A=1 to 640:print "H";:next
 50 for A=0 to 7
 60   ccolor@ A*10,0,A*10+9,19,A
 70 next
 80 cflash@ 0,2,79,3,1
 90 crev@ 0,4,79,5,1
100 crev@ 0,6,79,7,1
110 cflash@ 0,6,79,7,1
120 ' 実行終了後 CTRL + D キーを押してください
```

CURSOR （カーソル）

カーソルの移動を行ないます。

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

省略形　cu.

説　明

- 文字座標の＜X座標＞,＜Y座標＞の位置にカーソルを移動します。移動したのちに、print文などで文字を表示させるとその位置から表示します。
- ＜カーソルスイッチ＞はカーソルの表示を制御し、次のようになります。
  - 0 …指定以後カーソルを表示しません。
  - 1 …カーソルを常時表示します。
  - 2 …直接実行命令およびinputなどのキー入力待ちのときにカーソルを示します。
- ＜X座標＞の省略値は0で、＜Y座標＞や＜カーソルスイッチ＞を省略したときは現在の値のまま変化しません。
- インサートモード(文字コード&H01)のときにこの命令を実行すると、インサートモードを解除します。(ただし、位置が変化しない場合は解除しません。)

例

```
cursor 10,10 …… 文字座標(10,10)へカーソルを移動します。
cursor ,10 …… 11行目の先頭(左端)へカーソルを移動します。
```

参　考☞　csrh …… カーソルの水平位置を求める。
csrv …… カーソルの垂直位置を求める。

CFLASH (キャラクタ フラッシュ・character flash)

文字を点滅させます。

書 式　cflash {〔0〕 / 1}

省略形　cf.

説 明
- cflash 1を実行すると、以後はprint文やlistコマンドなどで画面に表示される文字が点滅します。指定を0とするか省略することにより通常の表示に戻ります。
- この状態はinit "CRT1:"の実行や CTRL ＋ D キーを押すことによっても解除されます。

例　`cflash` …… 文字の点滅表示状態であれば解除します。

CFLASH@ (キャラクタ フラッシュ アットマーク・character flash@)

範囲を指定して文字の点滅を行ないます。

書 式　cflash@ X_1, Y_1, X_2, Y_2, 〈モード〉

省略形　cf. @

説 明
- 文字座標の(X_1, Y_1)と(X_2, Y_2)を対角とする範囲に表示している文字を点滅します。
- 座標は、(X_1, Y_1)で左上、(X_2, Y_2)で右下を指定しますので、$X_1 \leqq X_2, Y_1 \leqq Y_2$でないとエラーになります。
- 〈モード〉を1とすると範囲内の文字が点滅し、0とすると点滅を解除します。

例
```
cflash@ 0,0,10,10,1
```

CREV (キャラクタ リバース・character reverse)

文字を反転させます。

書 式　crev { [0] / 1 }

省略形　cr.

説 明

- crev 1を実行すると、以後はprint文やlistコマンドなどで画面に表示される文字が反転します。指定を0とするか省略することにより通常の表示に戻ります。
- この状態はinit "CRT1:"の実行や CTRL ＋ D キーを押すことによっても解除されます。
- 表示の反転は、文字のフォント(光っている部分)が透明(ボーダーカラー)になり、バックがいままでのフォントの色になります。また、1ドット単位で色指定ができるPCGは各ドットの色が補色になります。

例　`crev 1` …… 以後表示される文字を反転します。

CREV@ (キャラクタ リバース アットマーク・character reverse @)

範囲を指定して文字の反転を行ないます。

書 式　crev@ X_1, Y_1, X_2, Y_2, 〈モード〉

省略形　cr.@

説 明

- 文字座標の(X_1, Y_1)と(X_2, Y_2)を対角とする範囲に表示している文字を反転します。
- 座標は(X_1, Y_1)で左上、(X_2, Y_2)で右下を指定しますので、$X_1 \leqq X_2$, $Y_1 \leqq Y_2$でないとエラーになります。
- 〈モード〉を1とすると範囲内の文字が反転し、0とすると反転を解除します。

例

```
crev@ 0,0,40,10,1
```

CGEN (キャラクタ ジェネレータ・character generator)

キャラクタジェネレータの切り換えを行ないます。

書 式

cgen {〔0〕 / 1〔,〈PCG指定〉〕}

省略形

cg.

説 明

- 画面に表示する文字の種類を選択します。以後は、print文やlistコマンドなどで表示する文字を指定のキャラクタジェネレータにより表示します。

指　　定	選択される CG(キャラクタ ジェネレータ)
0 または省略	通常のROM CGによる表示になります。
1	PCG 1～3を組み合わせてドット単位で色を指定できるPCGになります。
1,0～3	4種のPCGのうちいずれかを選択します。

- 文字表示のときの文字コードの対応は次のようになります。

  8×8ドットのときは、各PCGに256種の定義がされており、文字コードの0～255に対応します。

  8×16ドットのときは、各PCGに128種の定義がされており、0～255のうち偶数値で指定します。これは8×8ドットのときの偶数、奇数と続く部分を使用するためです。

  64色モードでPCG-1～3を組み合わせて使用する場合は、指定コードと指定コードに128を加えたコードが使用されます。(指定コードが128以上の場合は指定コード-128のコードになります。) 色信号は指定コードの側が変化の多い方の色信号に使用されます。

例

cgen 1 …… ドット単位で色指定のできるPCG表示にします。

参 考☞

def chr$ …… PCGフォントの定義。

cgpat$ …… PCGフォントの読み出し関数。

CHARACTER$ （キャラクタ ダラー）

文字画面より連続した文字を取り出します。

書　式　character$(〈X座標〉,〈Y座標〉[,〈文字数〉])

省略形　char.

説　明

- 文字画面の指定位置より、指定長で表示しているデータを求める関数です。
- 文字座標の＜X座標＞,＜Y座標＞が開始位置になり、＜文字数＞分の文字列として表します。
- ＜文字数＞は、半角文字(全角文字は半角文字の2文字分)での数で、最大255まで指定できます。省略したときの文字数は1になります。
- 取り込み範囲の最初の文字が全角文字の右半分にかかる場合や、最後の文字が全角文字の左半分にかかる場合は、2バイトのコードになりますから、返される文字長が増えます。ただし、文字長が255をこえる場合は255をこえた範囲の文字は入力できません。

例

```
A$=character$(0,0,40)
```

…… 画面の左上から40文字分の表示データをA$に代入します。

CSRH （カーソル ホリゾンタル・cursor horizontal）
CSRV （カーソル バーチカル・cursor vertical）

カーソルの現在位置を求めます。

書　式

```
csrh
csrv
```

説　明

- どちらも現在のカーソル位置を求めるシステム変数です。csrhは、水平位置、csrvは垂直位置が設定されています。
- csrhは画面の左端を0として右に向って数が大きくなります。40桁モードのときは0～39、80桁モードのときは0～79の値となります。
- csrvは画面の上端を0として下に向かって数が大きくなります。25行モードでは0～24、20行モードでは0～19、12行モードでは0～11になりますが、スムーススクロールモードでは最大値が1つ少なくなります。

DEF CHR$ (デファイン キャラクタ ダラー・define character$)

PCGの定義を行ないます。

書　式　　def chr$(〈コード〉[,〈PCG指定〉])＝〈文字列〉

説　明

- 文字のフォント(形)を作るPCGにはドット数が8×8ドット、8×16ドット、16×16ドットの3種があります。表示の方法としてPCG0～3のいずれかを単独で使用するものと、PCG1～3を組み合わせてドット単位で色を指定できるものがあります。この命令では<コード>によりドット数、<PCG指定>によりPCG0～3の組み合わせが指定され、それぞれの大きさによるフォントデータを<文字列>により指定します。

<コード>	<PCG指定>	定義されるPCG	文字列の大きさ
00～FF(H) 8×8ドット	省　略	PCG1～3	24バイト
	0	PCG0	8バイト
	1	PCG1	8バイト
	2	PCG2	8バイト
	3	PCG3	8バイト
100～1FE(H) 8×16ドット 偶数値	省　略	PCG1～3	48バイト
	0	PCG0	16バイト
	1	PCG1	16バイト
	2	PCG2	16バイト
	3	PCG3	16バイト
EC9F～ECFC ED40～ED7E ED80～EDA2	16×16ドット外字 / 省略(指定しません)	PCG1～3	32バイト

- フォントの設定は<文字列>の1文字ずつの文字コードを横8ドットとしてフォントの上から下へ順番に割り当て、16×16ドットのときは左側の16バイトの次に右側の16バイトの指定になります。また、PCG1～3を組み合わせて定義する場合は、1、2、3の順に割り当てます。なお、8ドットの指定は左端がMSBになります。
- 定義したPCGの表示は、8×8ドットと8×16ドットの場合**cgen**文でPCG表示モードに切り換えます。画面の表示が8×8ドットのときは0～255のコードに対応したPCGを表示し、8×16ドットモードのときは0～254の偶数で指定します。(奇数で指定した場合は指定-1の値とします。)

16×16ドットは外字となり、kmode 1の状態で外字コードで指定すれば表示できます。外字はJISコード表現では&J7821～&J787E、&J7921～7971、&J7A21～7A24になります。

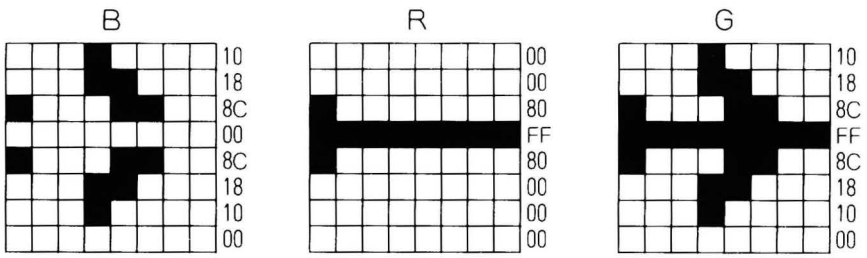
- PCG1～3の組み合わせで表示する場合、8色時はPCG1,2,3にそれぞれB.R.G.の色が割り当てられます。64色時は指定された文字コードと文字コード＋128または、文字コードが128以上の場合は文字コード－128のPCGと組み合わせます。したがって64色時は2つのPCGを定義してください。

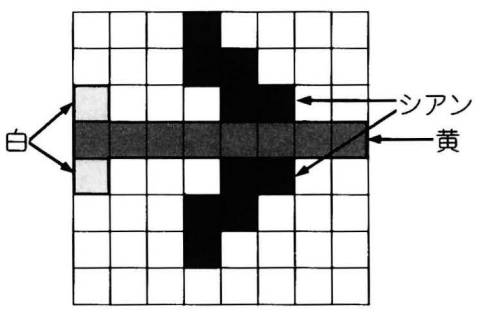
参　考☞　**cgen**　……　**PCG表示の切り換え。**

cgpat$　……　**PCGフォントの読み出し。**

サンプル

```
10 rem --- def chr$ ---
20 init "CRT1:40,25,0"
30 B$=hexchr$("10188C008C181000")
40 R$=hexchr$("000080FF80000000")
50 G$=hexchr$("10188CFF8C181000")
60 def chr$(64)=B$+R$+G$
70 cgen 1:print chr$(64)::cgen
```





CGPAT$ (キャラクタ ジェネレータ パターン ダラー・character generator pattern$)

文字のパターンを読みます。

書　式　cgpat$(〈コード〉[,〈PCG指定〉])

省略形　cgp.

説　明

- PCGやROM CGのフォントパターンを文字列データとして読み出す関数です。**def chr$**文と逆の働きをします。
- ＜コード＞は読み出す文字のデータで、＜PCG指定＞はPCGの読み出しを行なうときに1面ずつの指定を行ない、省略時はPCG1～3の連続したデータとなります。

＜コード＞指定と読み出すデータ

文字種	＜コード＞	＜PCG指定＞	読み出すCG	バイト数
8×8		省　略	PCG1～3	24
ド	00	0	PCG0	8
ッ	～	1	PCG1	8
ト	FF	2	PCG2	8
PCG		3	PCG3	8
8×16		省　略	PCG1～3	48
ド	100	0	PCG0	16
ッ	～	1	PCG1	16
ト	1FE	2	PCG2	16
PCG	偶数値	3	PCG3	16
ROM8×8	200～2FF	－	ROM	8
ROM8×16	300～3FF	－	ROM	16
全角文字	8140～EAA2	－	漢字ROM	32
外　字	EC9F～FCFC ED40～ED7E ED80～EDA2	－	PCG1～3	32

- 読み出されたデータの構成(内容)は**def chr$**文の定義データと同じです。

GET%　（ゲット パーセント）

文字画面の表示データを取り込みます。

書　式　　get%　X_1, Y_1, X_2, Y_2, 〈配列変数名〉[(〈要素番号〉)]

説　明

- 文字画面に表示している(X_1, Y_1)と(X_2, Y_2)の範囲のデータを数値型配列変数に取り込みます。取り込まれるデータは文字コードだけではなく、色の情報についても行なわれます。
- 配列変数にはoption base文で設定されている下限から順に記憶されますが、記憶を開始する要素番号を指定することもできます。
- 読み取りは1文字につき4バイトずつ左上の文字から順に右側へ行き、範囲の右端にくると次の行の左端から順に行なわれます。
- データと取り込む配列変数はあらかじめ必要な大きさを宣言しておきます。1文字当り4バイトのデータになりますので、必要なバイト数は横巾×縦の巾×4となります。配列変数は型により1つの配列要素当りのバイト数が異なり、整数型は2、単精度型は5、倍精度型は8で先の必要バイト数を1つ当りの要素バイト数で割れば必要な配列の要素数が求められます。

  必要な要素数＝横巾×縦の巾×4÷要素のバイト数＋1

- 1文字当りのデータは次のように記憶されます。

  1バイト目　文字コード又は漢字ROMアドレス.（2バイト目のKが0の場合は文字コード、1の場合は、ROMアドレスになります。）

  MSB [文字コード] LSB　　MSB [ROMアドレス(low) | R | X] LSB

  2バイト目　　漢字属性

  MSB [K | L | ROMアドレス(high)] LSB

  3バイト目　　色属性8色時3ビット、64色時6ビットを使用します。

  MSB [- | - | G | R | B | G | R | B] LSB

  4バイト目　　文字属性

  MSB [Bl | Rv | S_1 | S_0 | P | C | - | -] LSB

  C　1　……　PCGによる表示
  　　0　……　ROM CGでの表示

  （このデータはテキストV-RAMの3バイトのデータのうち、色コードを64色モードでも表せるようにしたものです。各ビットの機能はオーナーズマニュアルを参照してください。）

- この命令で読み取ったデータは**put%**文により画面の任意の位置に表示ができます。

参　考　　**put%**　……　get%で読み取ったデータの表示。

PUT% （プット パーセント）

GET%文で読み取ったデータを表示します。

書　式　put% X_1, Y_1, X_2, Y_2,〈配列変数名〉[(〈要素番号〉)]

説　明

- GET%文で配列に読み取った文字表示のデータを(X_1, Y_1)－(X_2, Y_2)の範囲で画面に表示します。＜配列変数名＞だけの指定ではその配列の下限(option base文で0または1に設定されている)から順に取り出し、指定範囲の左上から右へ順に進み右端にきた場合は次の行の左端に進んで右側へと表示して行きます。
- ＜配列変数名＞のあとに＜要素番号＞を指定した場合は、指定された要素番号より読み取りを始めます。

参　考☞　get%　……　文字画面のデータを取り込む。

サンプル

```
 10 rem --- get%  put% ---
 20 init "CRT1:40,25"
 30 dim X%(52):C=1
 40 for A=65 to 90
 50   crev A mod 2
 60   ccolor C:print chr$(A);
 70 C=C+1:if C=8 then C=1
 80 next:color 7:cflash:crev
 90 get% 0,0,25,0,X%
100 put% 0,3,25,3,X%
110 put% 0,5,12,6,X%
120 put% 0,8,6,11,X%
130 put% 30,0,30,19,X%
140 put% 34,0,34,19,X%(12)
150 cursor 0,20:ccolor 7
```

CBLOCK （カラー ブロック・color block）

グラフィック表示色のブロック設定を行ないます。

書 式　cblock〔〈ブロック番号〉〕

省略形　cb.

説 明

- グラフィックが256色モードのときに、最もレベルの低い信号をどのように割り当てるかを指定します。ブロック番号により次のようにプレーンと色信号の設定を行ないます。

＜ブロック番号＞	色信号	G	R	B
	レベル	4/7 2/7 1/7	4/7 2/7 1/7	4/7 2/7 1/7
0		G_1 G_0 I_1	R_1 R_0 I_0	B_1 B_0 0
1		G_1 G_0 I_1	R_1 R_0 0	B_1 B_0 I_0
2		G_1 G_0 0	R_1 R_0 I_1	B_1 B_0 I_0

- この命令の指定によりR. G. B. いずれかの色が 4 段階の変化になります。512色の色指定では、0となっている部分は変化せずにつねに0になります。
- この命令はグラフィックが256色モードのときに有効で、そのほかのモードの時には表示に影響しません。
- ＜ブロック番号＞の省略時、または初期状態は0になります。 CTRL ＋ D キーの操作、またはinit "CRT2:"の実行で初期化されます。

例

```
cblock 2
```

…… 256色の表示を緑が4段階に変化する組み合わせに設定します。

COLOR＝　（カラー　イコール）

カラーパレットの設定を行ないます。

書　式

color=(⟨パレットコード⟩,⟨カラーコード⟩)[,(⟨パレットコード⟩,⟨カラーコード⟩)]…　(1)

color=(⟨カラーコード⟩,⟨G輝度⟩,⟨R輝度⟩,⟨B輝度⟩)[,(⟨カラーコード⟩,⟨G輝度⟩,⟨R輝度⟩,⟨B輝度⟩)]…　(2)

(1)…16×16色パレット用　　(2)…15×4096色パレット用

省略形

col.=

説　明

- グラフィック画面が4，16色モードのとき、0～15の＜パレットコード＞に対して画面上に出力される色を設定します。
- 書式(1)は0～15の＜パレットコード＞に対して0～15の＜カラーコード＞で指定します。
- 書式(2)は別売のカラーパレットボード(MZ-1M10)を取り付けた場合に有効で、G.R.B.の3原色に対してそれぞれ0～15の輝度レベルを設定して4096種の色を表現することができます。その色を16色×16色パレット回路を通って出てきた信号(＜カラーコード＞)に対して指定します。＜カラーコード＞は0～15までありますが、＜カラーコード＞0に対する指定は無効になります。
- 書式(1)ではグラフィック画面の表示色指定で文字表示色は変化しませんが、書式(2)では画面の表示色に対して指定が行なわれますので文字の表示色も同様の変化をします。この場合、文字の色コード1～7はカラーコードの9～15と同じになります。
- 書式(1)を実行すると、16色×16色パレット回路からディスプレイへ直接信号を出力(図1)します。書式(2)を実行すると、16色×16色パレット回路から、15色×4096色パレット回路を通してディスプレイへ信号を出力(図2)します。

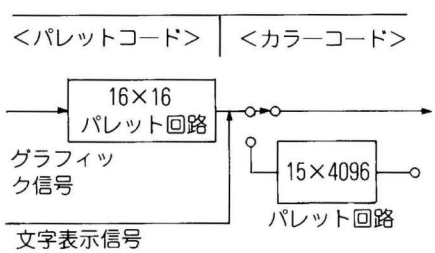


図1

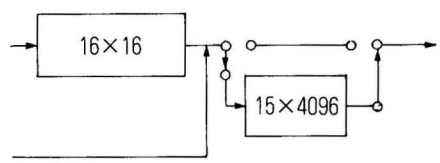


図2

したがって書式(1)を実行すると15×4096色の指定は無効となり、書式(2)を実行すると15×4096色の指定が有効となります。このような回路構成となっ

ていますので15×4096色パレットを使用されるときは16×16色パレットは初期状態のままで指定を行なわない方が、<パレットコード>と<カラーコード>が1対1で対応するため表示色がわかりやすくなります。

- この命令はグラフィックが256色モードのときは何も行ないません。

PRIORITY （プライオリティ）

表示の優先順位を設定します。

書　式　priority〔〈文字対グラフィック順位〉〕〔,〈グラフィック順位〉〕

省略形　prio.

説　明

- ＜文字対グラフィック順位＞は4、16色モードのときにグラフィック画面の各パレットコードに対して設定することができ、整数型データを16ビットの2進数として次のように対応します。

2進数	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
パレットコード	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
表示色（パレット初期状態）	明白	明黄	明シアン	明緑	明マゼンタ	明赤	明青	灰	白	黄	シアン	緑	マゼンタ	赤	青	黒

各ビットが1の場合、指定色は文字より優先し、0の場合は文字の方が優先します。

- ＜文字対グラフィック順位＞は、整数型データで指定しますので**-32768～32767（& H0～ & HFFFF）**の範囲になりますが、特別な処理として**32768～65535（& H8000～ & HFFFF**に対応）までの値による指定も可能です。
- ＜グラフィック順位＞は2つのグラフィック画面の重ね合わせを行なったときに、どちらを優先させるかを指定します。グラフィック画面の重ね合わせはスクリーン0と1、または2と3です。＜グラフィック順位＞を0とするとスクリーン番号が偶数の方が優先され、1とすると奇数の方を優先します。

グラフィック順位	組み合わせと優先	
0	0>1	2>3
1	0<1	2<3

大小記号の開いた方を優先します。

- すべての指定を省略すると初期状態になります。（priority　0, 0）
- CTRL ＋ D キーの操作またはinit　"CRT2:"の実行により初期化されます。

VIEW （ビュー）

グラフィックのビューポートを設定します。

書　式　view 〔X_1, Y_1, X_2, Y_2〔,〈領域色〉〕〔,〈境界色〉〕〕

省略形　vi.

説　明

- グラフィック画面上の実際にグラフィック描画が行なわれる範囲（ビューポート）を指定します。画面の左上を(0, 0)とする絶対スクリーン座標の(X_1, Y_1)と(X_2, Y_2)を対角とする範囲がその指定となり、そのときに＜領域色＞を指定すると、その範囲を指定された色で塗りつぶし、＜境界色＞を指定すると、指定色でビューポートの枠取りをします。
- ＜領域色＞や＜境界色＞を省略した場合はそれぞれの機能は行なわれません。
- すべての指定を省略すると初期化され、画面全体がビューポートになります。
- 座標は範囲の左上と右下を指定しますので、$X_1 \leqq X_2$、$Y_1 \leqq Y_2$でない場合はエラーになります。
- この指定により以後のグラフィック描画はビューポート内で行なわれ、範囲外へのグラフィック描画は行なわれません。また、ビューポートの左上がスクリーン座標(0, 0)になります。
- CTRL ＋ D キーの操作またはinit "CRT2:"の実行により初期化されます。

例

```
view 0,0,199,159
```

サンプル

```
 10 rem --- view ---
 20 init "CRT2:":cls 2
 30 option angle degrees
 40 color 7 :gosub 1000
 50 view 0,0,160,100
 60 color 9 :gosub 1000
 70 view 160,0,319,199
 80 color 10:gosub 1000
 90 view 80,80,240,120
100 color 12:gosub 1000
990 view:end
1000 poly 160,100,90,185,,13320
1010 return
```

VIEW@ （ビュー アットマーク）

ディスプレイ上に表示するグラフィック画面の範囲を指定します。

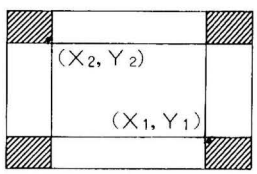
書 式　view@〔X_1, Y_1, X_2, Y_2〕

省略形　vi. @

説 明

- 画面の左上を(0, 0)とする絶対スクリーン座標の(X_1, Y_1)と(X_2, Y_2)を対角とする範囲をディスプレイ上に表示します。範囲以外の部分はディスプレイ上に表示されませんが、図形の描画などは行なうことができます。
- 横方向の指定のX_1とX_2は、横方向のドット数が640ドットの場合は8ドット単位、横方向のドット数が320ドットの場合は4ドット単位で表示が行なわれます。たとえば指定値をXとして640ドットの場合、**(X¥8)＊8**から8ドット分が指定されたことになります。(¥は整数除算の記号です。)
- 横方向または縦方向で開始位置が終了位置より1つ少ない場合は画面全体が出力されません。(例 **view@ 0,100,319,99**)
- 座標指定を省略すると初期化され、グラフィック画面全体を表示します。
- (X_1, Y_1)が表示開始位置、(X_2, Y_2)が表示終了位置の指定となりますので、X_1>X_2であれば範囲の上下、Y_1>Y_2であれば範囲の左右の部分が表示されることになり、範囲内が表示されなくなります。

  例) X_1>X_2, Y_1>Y_2であれば次のようになります。



▨部分が表示され、それ以外の部分は表示しません。

- CTRL ＋ D キーの操作またはinit "CRT2:"の実行によっても初期化し、画面全体を表示します。

例

```
view@ 0,0,100,100
```

WINDOW （ウインドゥ）

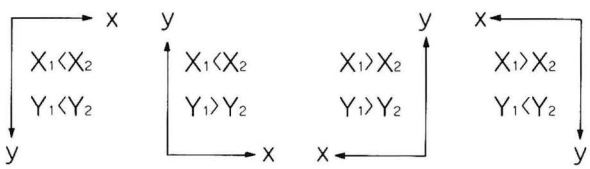
ビューポートに論理座標を設定します。

書　式　window 〔X_1, Y_1, X_2, Y_2〕.

省略形　win.

説　明

- view文で設定されたビューポートに(X_1, Y_1)と(X_2,Y_2)の論理座標(ワールド座標)を割り当てます。ビューポートの左上が論理座標(X_1, Y_1)、右下が論理座標(X_2, Y_2)になります。
- 座標指定を省略すると現在のビューポートに対してスクリーン座標と同じワールド座標を設定します。
- スクリーン座標は左上を起点としてX軸は右方向、Y軸は下方向に座標値が増加しますが、ワールド座標は、座標の増加方向を自由に指定することができます。これはX_1とX_2、Y_1とY_2の大小により決まり、次のようになります。



- CTRL ＋ D キーの操作またはinit "CRT2:"の実行により初期化され、画面全体に設定されたビューポートに対してスクリーン座標と同じワールド座標が割り当てられます。

例

```
window 0,199,319,0
```

…… スクリーン座標に対してY軸を反転させます。

サンプル

```
  10 rem --- window ---
  20 init "CRT2:":cls 2
  30 option angle degrees
  40 color 9 :gosub 1000
  50 window 0,399,319,0
  60 color 10:gosub 1000
  70 window 0,0,639,399
  80 color 12:gosub 1000
  90 window 0,0,639,199
 100 color 14:gosub 1000
 990 window:end
1000 poly 160,100,90,144,90,810
1010 return
```

COLOR （カラー）

グラフィックの描画色を設定します。

書　式　color〈描画色〉〔,〈機能〉〕

省略形　col.

説　明

- グラフィック描画命令で、描画色や描画機能を省略したときに、設定される内容を指定します。
- ＜描画色＞の指定は、4,16色モードのときはパレットコード、256色モードでは512色系のカラーコードで指定します。
- ＜機能＞は描画機能を0～3の値で指定し、省略時は0が設定されます。

  0　…　強制的に指定色に変更します。

  1　…　論理演算ORを行ないます。

  2　…　論理演算XORを行ないます。

  3　…　論理演算ANDを行ないます。

  それぞれの論理演算は、描画しようとする座表に描かれているデータと描画データで行ないます。**reset**文と**bline**文は機能の内容が異なります。
- この命令での設定は、init "CRT2:"文の実行、または、CTRL ＋ D キーの操作で初期化され、＜描画色＞は**15**または**&O777**(256色モード)、＜機能＞は0になります。

例

```
color 15,1
```

FILL （フィル）

グラフィック画面を指定色で塗りつぶします。

書　式　fill〈塗りつぶし色〉

省略形　fi.

説　明

- グラフィック画面のビューポート内を＜塗りつぶし色＞で塗りつぶします。
- ＜塗りつぶし色＞の指定は、4,16色モードのときはパレットコード、256色モードでは512色系のカラーコードで指定します。

例

```
fill 8
```

PEN　(ペン)

ドットおよび線描画のペン形状を指定します。

書　式

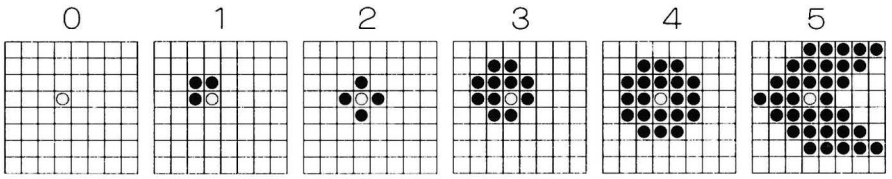
pen〔〈ペン番号〉〕　　…ペンの選択
pen〈ペン番号〉,〈ペン形状〉　　…ペン形状の指定

説　明

- グラフィック描画時のペン形状を指定します。ペン形状が有効なのは次の命令です。

  set　reset　line　bline　box　circle　poly

- ペンの形状は6種類あり、＜ペン番号＞は0～5の数で指定します。BASICの起動時は次の形状が設定されています。



○のドットが中心位置になります。

- ペン番号0以外は形状の指定ができます。＜ペン形状＞は8バイトの文字列データで指定し、1文字目がいちばん上の横8ドットで左端がMSB、右端がLSBになります。ペン番号5の形状指定をする場合は、**hexchr$("1F3E7CF87C3E1F00")**になります。また、中心となる位置は左側から4ドット目、上から4ドット目の場所になります。
- ペンの選択で＜ペン番号＞を省略した場合は0が指定されたものとします。

例

```
pen 5:circle [1] 160,100,90
```

SET （セット）

ドットのセットを行ないます。

書 式　set［[［〈色〉］[,〈機能〉]］］X,Y

説 明

- ワールド座標上の指定された位置に対して指定された＜色＞で点をセットします。
- (X,Y)がセットする位置でstep (X,Y)による相対座標での指定が可能です。
- 実行後グラフィックポインタはドットをセットした座標になります。
- ＜色＞は、4,16色モードではパレットコード、256色モードでは512色系のカラーコードで指定します。
- ＜機能＞は、描画機能を0～3の値で指定します。(color文参照)
- ＜色＞や＜機能＞を省略した場合は、color文で設定された値になります。
- pen文によりセットするドット形状の選択、設定が可能です。

例

```
set 100,100
```

…… ワールド座標(100,100)をcolor文で設定されている＜色＞，＜機能＞でセットします。

RESET （リセット）

ドットのリセットを行ないます。

書 式　reset［[［〈色〉］[,〈機能〉]］］X,Y

説 明

- set文の逆で指定座標をリセットします。 座標指定などはset文と同様で、＜機能＞が次のようになります。
  - 0 … 指定色を構成するプレーンのみをリセットします。
  - 1 … 指定色の補色でセットします。
- ＜機能＞を省略した場合は、color文で設定されている値になりますが、2と3が設定されている場合は、0になります。

LINE （ライン）

指定座標間を直線で結びます。

書 式

line 〔[〔〈色〉〕〔,〈機能〉〕〔,〈ラインスタイル〉〕]〕$X_1, Y_1, X_2, Y_2 \cdots$〔, X_n, Y_n〕

説 明

- ワールド座標上の$(X_1, Y_1), (X_2, Y_2), \cdots (X_n, Y_n)$の間を直線で結びます。実行後のグラフィックポインタは(X_n, Y_n)になります。
- それぞれの座標には、**step (X, Y)**による相対座標での指定が可能です。座標に使用した場合は、この命令の実行時点のグラフィックポインタから、それ以後の座標に使用した場合は、直前の座標からの相対座標指定になります。
- ＜色＞は、4, 16色モードではパレットコード、256色モードでは512色系のカラーコードで指定します。
- ＜機能＞は、描画機能を0～3の値で指定します。（color文参照）
- ＜色＞や＜機能＞を省略した場合は、color文で設定された値になります。＜ラインスタイル＞は16ドット単位のドットの並びを整数型データで指定します。

  ラインスタイルの例

  &HCCCC　C C C C

  &HFF3C　F F 3 C

- pen文により線を引くためのドット形状の選択、設定ができます。

例

```
line [3] 100,50,150,100,50,100,100,50
```

BLINE （ブランク ライン・blank line）

リセットした直線を引きます。

書 式 bline〔[〈色〉]〔,〈機能〉〕〔,〈ラインスタイル〉]]X_1, Y_1, X_2, Y_2…〔, X_n, Y_n〕

省略形 bl.

説 明

- line文の逆で指定座標間にリセットした直線を引きます。座標指定などは、line文と同様で＜機能＞が次のようになります。
  - 0 … 指定色を構成するプレーンのみをリセットした直線を引きます。
  - 1 … 指定色の補色で直線を引きます。
- ＜機能＞を省略した場合は、color文で設定されている値になりますが、2と3が設定されている場合は0になります。

BOX （ボックス）

2点を対角とする四角形を描きます。

書 式 box〔[[〈色〉]〔,〈機能〉]〔,〈ラインスタイル〉]]]X_1, Y_1, X_2, Y_2〔{〈タイルパターン〉 / 〈塗りつぶし色〉}〕

説 明

- ワールド座標の(X_1, Y_1)と(X_2, Y_2)を対角とする四角形を描きます。
- ＜色＞は、四角形を描く線(枠)の描画色で、＜ラインスタイル＞は、枠のラインスタイルをline文と同様に、16ドット単位で指定します。
- ＜塗りつぶし色＞を指定すると、四角形の枠内を指定色で塗りつぶし、＜タイルパターン＞を指定すると枠内をパターンで塗りつぶします。
- ＜色＞と＜塗りつぶし色＞は、4, 16色モードではパレットコード、256色モードでは512色系のカラーコードで指定します。
- ＜機能＞は、描画機能を0～3の値で指定します。(color文参照)
- ＜色＞や＜機能＞を省略した場合は、color文で設定された値になります。

CIRCLE （サークル）

円を描きます。

書　式

circle〔〈描画指定〉〕X,Y,〈半径〉〔,〈比率〉〕〔,〈開始角〉〕〔,〈終了角〉〕〔,O〕

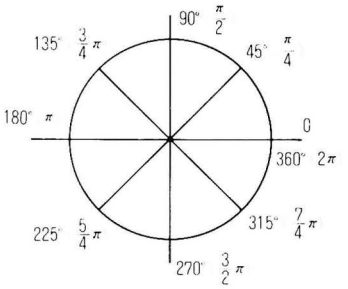
〈描画指定〉…[[〈色〉]〔,〈機能〉]]

省略形

ci.

説　明

- ワールド座標(X,Y)を中心として＜半径＞で指定された大きさの円を描きます。
- 中心座標の指定はstep (X,Y)による相対座標での指定が可能です。
- 実行後のグラフィックポインタは中心座標になります。
- ＜色＞は、4,16色モードではパレットコード、256色モードでは512色系のカラーコードで指定します。
- ＜機能＞は、描画機能を0～3の値で指定します。(color文参照)
- ＜色＞や＜機能＞を省略した場合は、color文で設定された値になります。
- ＜開始角＞と＜終了角＞は中心座標の右側から始まり左回りで角度が増えます。角度の単位はラジアンと度が使用でき、その選択はoption angle文によって行ないます。＜開始角＞の省略時は0、＜終了角＞の省略時は360度またはラジアンで2πになります。



角度指定(　)内はラジアン

- ＜比率＞を指定することにより、だ円を描くことができます。比率1が真円でそれ以外は次のようになります。(比率＝垂直半径／水平半径)

  ＜比率＞　＜　1　…横方向を1とした横長のだ円になります。

  ＜比率＞　＞　1　…縦方向を1とした縦長のだ円になります。
- ＜比率＞を省略した場合、640×200ドットモードのときは0.5、それ以外のときは、1の指定になります。
- 0を指定すると中心まで直線を引き扇形にすることができます。
- pen文により線を引くためのドット形状の選択、設定ができます。

例

```
circle 150,100,50
```

……中心座標(150,100)で半経50の円を描きます。

POLY （ポリゴン・polygon）

多角形を描きます。

書　式

poly〔〈描画指定〉〕X,Y〔〈半径〉〕〔,〈ステップ角〉〕〔,〈初期角〉〕〔,〈終了角〉〕
〈描画指定〉…[[〈色〉]〔,〈機能〉〕〔,〈ラインスタイル〉]]

説　明

- ワールド座標の(X,Y)を中心として、＜初期角＞から＜ステップ角＞ごとの頂点を＜終了角＞になるまで線を結んで行きます。＜半径＞が中心座標より頂点までの長さになります。
- 中心座標の指定はstep　(X,Y)による相対座標での指定が可能です。
- 実行後のグラフィックポインタは中心座標になります。
- ＜色＞は、4,16色モードではパレットコード、256色モードでは512色系のカラーコードで指定します。
- ＜機能＞は、描画機能を0～3の値で指定します。(color文参照)
- ＜色＞や＜機能＞を省略した場合は、color文で設定された値になります。
- ＜ステップ角＞、＜初期角＞、＜終了角＞の指定はラジアンと度が使用でき、その選択はoption angle文により行ないます。それぞれの省略時の値は＜ステップ角＞が1度＜初期角＞は0度、＜終了角＞は360度になります。
- ＜ラインスタイル＞はline文と同様に2バイトの整数形で指定し、16ドットの指定を行ないます。
- .pen文により線を引くためのドット形状の選択、設定ができます。

例

```
option angle degrees:poly 160,100,70,144,90,810
```

……　星形を描きます。

参　考☞　circle　……　円を描く。

PAINT （ペイント）

境界内を塗りつぶします。

書　式

paint〔[{〈色〉 / 〈タイルパターン〉}]]X,Y{〔,〈境界色〉〕… / 〔,not 〈境界色〉〕}

省略形

pa.

説　明

- ワールド座標の(X,Y)の位置から＜境界色＞で囲まれた範囲内を指定された＜色＞または＜タイルパターン＞で塗りつぶします。ただし、ビューポートの外側は塗りつぶしません。
- 塗り始める位置の指定はstep（X,Y)による相対座標での指定が可能です。
- この命令の実行後のグラフィックポインタは、塗り始めの位置になります。
- ＜色＞は塗りつぶしに使用する色で、省略した場合はcolor文で指定されている＜描画色＞になります。
- ＜色＞と＜境界色＞は、4,16色モードではパレットコード、256色モードでは512色系のカラーコードで指定します。
- ＜境界色＞は複数個の指定ができ、省略された場合は塗りつぶす＜色＞が境界になります。not ＜境界色＞は、＜境界色＞以外のすべての色を境界色とする指定になります。
- 塗り始める位置がすでに境界色となっている場合は、塗りつぶしを行ないません。
- ＜タイルパターン＞は単色の塗りつぶしではなく、横8ドットで縦は任意のドット数の模様で指定できます。指定は文字列データで、縦をnドットとすると4,16色モードのときはn×4バイト、256色モードのときはn×8バイトの長さになります。画面にはこのデータの1バイトずつを2進数に対応させて各プレーンに次のように展開します。

4,16色モード時

B	R	G	I

（4色モードのときは、G.I.プレーンに対する指定は何も行ないませんが、2段以上のパターン指定を行なう場合は、ダミーとして必要です。）

256色モード時

B_1	R_1	G_1	I_1	B_0	R_0	G_0	I_0

（I_0とI_1のプレーンは、cblock文により色信号が割り当てられます。cblock文参照）

例)　ＰＴ$=hexchr$（"ＡＡＣＣＦ０００ＡＡＣＣＦ０ＦＦ"）

16色モードでPT$をパターンデータとすると次のような8×2ドットのパターンの繰り返しになります。

サンプル

```
 10 rem --- paint ---
 20 init "CRT2:":cls 2
 30 window -160,-100,159,99
 40 option angle degrees
 50 X1=0:Y1=0
 60 for R=0 to 99 step .5
 70   X2=sin(12*R)*R:Y2=cos(12*R)*R
 80   line X1,Y1,X2,Y2
 90   line -X1,-Y1,-X2,-Y2
100   line Y1,-X1,Y2,-X2
110   line -Y1,X1,-Y2,X2
120   X1=X2:Y1=Y2
130 next R
140 circle 0,0,99
150 paint [9] 0,-95,15
160 paint [10] 95,0,15
170 paint [12] 0,95,15
180 paint [14]-95,0,15
190 PT$=hexchr$("AACCF000AACCF0FF")
200 paint [PT$]-160,-100,15
```

PATTERN （パターン）

グラフィック画面にパターンを描きます。

書　式

pattern〔〈描画指定〉〕〈段数〉,〈パターン〉
〈描画指定〉…[[〈フォント色〉]〔,〈機能〉〕〔,〈バック色〉]]

省略形

pat.

説　明

- 現在のグラフィックポインタから、指定された＜段数＞にしたがってパターンを描きます。グラフィックポインタはposition文で設定することができます。
- ＜段数＞は±255までの指定ができ、正の場合は上方向、負の場合は下方向に重ねます。
- ＜パターン＞は文字列型データで、1バイトずつ文字コードを2進数に展開して画面に表示します。文字列の長さが段数をこえる場合は、最初の位置より右に8ドット分ずれた位置より続けます。
- この命令の実行後のグラフィックポインタは、最後のパターンデータの次の位置になります。
- ＜フォント色＞はドットの部分、＜バック色＞はそれ以外の部分の色を指定します。指定を省略した場合はcolor文の指定による＜描画色＞と＜0＞になります。
- ＜色＞は4,16色モードではパレットコード、256色モードでは512色系のカラーコードで指定します。
- ＜機能＞は0～5の値で指定し、次のようになります。省略時はcolor文での設定値になります。

  0 … フォント色、バック色にしたがって描きます。

  1 … or
  2 … xor　すでに表示されているデーターとの論理演算をします。
  3 … and

  4 … フォント部分についてのみ＜フォント色＞による描画が行なわれ、バックの部分は変化しません。したがって、＜バック色＞を指定しても何も行ないません。

  5 … ＜フォント色＞に該当するプレーンのみ描かれ、該当しないプレーンには作用しません。バック部分は該当プレーンではリセットされます。＜バック色＞を指定しても何も行ないません。

SYMBOL （シンボル）

任意の大きさで文字パターンを描きます。

書　式

symbol〔〈描画指定〉〕X,Y,〈文字列〉〔,〈横倍率〉〕〔,〈縦倍率〉〕〔,〈角度〉〕〔,〈フォント〉〕
〈描画指定〉…[[〈色〉]〔,〈機能〉]]

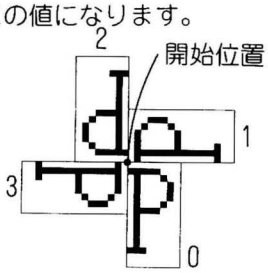
省略形

sy.

説　明

- ビューポート内のスクリーン座標(X,Y)の位置から、＜文字列＞による文字フォントを縦横の倍率を設定して描きます。また、4種類の角度を指定して文字の向きを変えることができます。
- 開始座標は文字の左上部分になります。また、開始座標の指定はstep(X,Y)による相対座標での指定が可能です。
- ＜横倍率＞と＜縦倍率＞は1～255までの整数で指定でき、省略した場合はそれぞれ1になります。
- ＜色＞は、4,16色モードではパレットコード、256色モードでは512色系のカラーコードで指定します。
- ＜機能＞は、描画機能を0～3の値で指定します。（color文参照）
- ＜色＞や＜機能＞を省略した場合は、color文で設定された値になります。
- ＜角度＞は90度ごとの4方向で0～3の整数で指定します。

  0　…　通常状態で省略時はこの値になります。
  1　…　左回転で 90度。
  2　…　左回転で180度。
  3　…　左回転で270度。



- この命令で描かれるのはフォント部分のみで、フォント以外の部分は変化しません。
- ＜フォント＞は半角文字のドット数の選択です。0を指定すると8×8ドット、1を指定すると8×16ドットになります。また、省略時は、現在半角文字表示に用いられているフォントになります。
- この命令の実行後のグラフィックポインタは、続いて同倍率、同角度で文字パターンを描く場合の次の位置になります。

例

```
symbol 16,16,"ABC",2,2
```

…… 縦，横2倍で"ABC"を描きます。

COLOR REPLACE （カラー リプレース）

グラフィック画面の表示色を変更します。

書 式

color replace X_1, Y_1, X_2, Y_2, 〈旧色〉, 〈新色〉〔, 〈旧色〉, 〈新色〉〕…

省略形

col. repl.

説 明

- グラフィック画面上の表示に対して、指定範囲内の特定色をほかの色に変更します。この場合の色変更は、 color= 文のようにディスプレイ出力の信号を切り換えるのではなく、グラフィックV-RAMに書かれているデータを変更します。
- (X_1, Y_1)と(X_2, Y_2)のエリア指定はビューポートの左上を(0, 0)とするスクリン座標に対して行ない、エリア内の＜旧色＞、＜新色＞の組み合わせで変更して行きます。
- 開始座標の(X_1, Y_1)と終了座標の(X_2, Y_2)には、**step (X, Y)**による相対座標での指定が可能です。開始座標に使用した場合は、この命令の実行時点のグラフィックポインタから、終了座標に使用した場合は、開始座標からの相対座標になります。
- この命令の実行後のグラフィックポインタは、(X_2, Y_2)の位置になります。
- ＜旧色＞と＜新色＞は、4, 16色モードではパレットコード、256色モードでは512色系のカラーコードで指定します。
- この命令では**screen**文で設定されているアクティブプレーンには関係なくすべてのプレーンに対して行なわれます。

例

```
color replace 0,0,100,100,1,2,2,1
```

…… 指定範囲の1と2の色を入れ換えます。

サンプル

```
 10 rem --- color replace ---
 20 init "CRT2:":cls 2:pen 3
 30 for R=1 to 90 step 3
 40   C=C+1:if C=16 then C=1
 50   circle [C] 160,100,R
 60 next:pen 0
 70 for A=2 to 15:B=16-A
 80   color replace 0,0,159,199,1,A,A,1
 90   color replace 159,0,319,199,15,B,B,15
100 next
```

MOVE （ムーブ）

グラフィック表示の移動を行ないます。

書　式　move〔[[〈色〉]〔,〈機能〉]]]X_1, Y_1, Lx, Ly, X_2, Y_2〔,〈移動処理〉〕

説　明

- グラフィック画面に描かれている図形を移動します。スクリーン座標の(X_1, Y_1)を左上の起点として、横幅がLx、縦の長さがLyの範囲の図形をスクリーン座標(X_2, Y_2)の位置へ移動します。
- 開始座標の(X_1, Y_1)と終了座標の(X_2, Y_2)には、step (X, Y)による相対座標での指定が可能です。 開始座標に使用した場合は、この命令の実行時点のグラフィックポインタから、終了座標に使用した場合は、開始座標からの相対座標になります。
- この命令の実行後のグラフィックポインタは、(X_2, Y_2)になります。
- ＜色＞は、移動する色の指定となりますが、実際には指定色を表示するために必要なプレーンのデータが移動します。たとえば、16色モードでコードが6の黄色を指定した場合、黄色を構成するコード2の赤とコード4の緑の色も移動します。＜色＞の指定を省略した場合はcolor文で指定されている＜描画色＞になります。
- ＜移動処理＞を0とすると移動後に以前のデータを消去し、1とすると以前のデータは消去されません。省略時は0の指定になります。
- ＜色＞は、4, 16色モードではパレットコード、256色モードでは512色系のカラーコードで指定します。
- ＜機能＞は、描画機能を0～3の値で指定します。(color文参照)
- ＜色＞や＜機能＞を省略した場合は、color文で設定された値になります。

例

```
move 0,0,16,16,16,16
```

サンプル

```
 10 rem --- move ---
 20 init "CRT2:":cls 2
 30 box 0,0,39,39,8
 40 option angle degrees
 50 poly [7] 20,20,18,144,90,810
 60 for A=0 to 79
 70   move A,A,40,40,A+1,A+1
 80 next
 90 for C=0 to 3
100   for A=80 to 279
110     move [2^C] A,80,40,40,A+1,80
120   next
130 next
```

ROLL （ロール）

グラフィック画面のスクロールを行ないます。

書 式　roll〈移動量〉

説 明

- グラフィック画面のビューポート内に描画されているデータ全体を上下にスクロール(移動)させます。
- ＜移動量＞はドットで表し、値が正の場合は上方向、負の場合は下方向に移動します。移動方向の反対側は、消去します。また、＜移動量＞がビューポートの縦のドット数より大きい場合は、全画面をすべて消去します。

例　`roll -16` …… 下方向へ16ドット分のスクロールを行ないます。

ROLL@ （ロール アットマーク）

グラフィックのディスプレイ表示のスクロールを行ないます。

書 式　roll@〔〈移動量〉〕

説 明

- グラフィック画面をディスプレイに出力する位置をスクロール(移動)します。
- ＜移動量＞は、現在の表示より移動を行なう量をドット数で指定します。値が正の場合は上方向、負の場合は下方向に位置を移します。
- 表示のスクロールはディスプレイへの出力信号を制御しますので、グラフィックの座標は移動にともない上下にずれます。
- 指定を省略すると、初期化します。
- 移動により、はみ出す部分は、移動方向と反対側に表示します。

例　`roll@ 100` …… グラフィック表示を上方向に100ドット分ずらします。

サンプル

```
10 rem --- roll@ ---
20 init "CRT2:":cls 2
30 option angle degrees
40 poly [10] 160,50,40,144,90,810
50 circle [12] 160,150,40
60 for A=1 to 400:roll@ 1:next
70 for A=1 to 400:roll@ -1:next
80 for A=1 to 400:roll@ 100:next
90 for A=1 to 400:roll@ 101:next
```

MAP （マップ）

スクリーン座標とワールド座標の相互変換を行ないます。

書 式 map(〈数式〉,〈機能〉)

説 明

- ＜数式＞をスクリーン座標または、ワールド座標として相互の変換を行ないます。変換内容は＜機能＞に0～3の値を指定します。

＜機能＞の値	変換内容
0 ……	＜数式＞をワールド座標のX座標として、スクリーン座標のX座標に変換した値を返します。
1 ……	＜数式＞をワールド座標のY座標として、スクリーン座標のY座標に変換した値を返します。
2 ……	＜数式＞をスクリーン座標のX座標として、ワールド座標のX座標に変換した値を返します。
3 ……	＜数式＞をスクリーン座標のY座標として、ワールド座標に変換した値を返します。

POINT （ポイント）

グラフィックの指定座標の表示色を求めます。

書 式 point(X, Y)

省略形 poi.

説 明

- ビューポート内の左上を(0, 0)とするスクリーン座標の(X, Y)の位置に表示されている色を求める関数です。
- 指定された値がビューポートの範囲外の場合は−1の値を返します。
- 求められる値は4, 16色モードのときはパレットコード、256色モードのときは512色コードになります。

POSITION （ポジション）

グラフィックポイントの位置を設定します。

書 式

position { X, Y
step (X, Y)
screen X, Y
screen step (X, Y) }

省略形

pos.

説 明

- グラフィック命令の描画位置を記憶しているグラフィックポイントの値を指定します。グラフィックポイントはグラフィック描画命令において、step (X, Y)による相対座標指定の起点となるもので、描画命令実行後はその命令での指定座標に変わります。
- グラフィックポイントの位置指定はワールド座標、またはビューポート内の左上を(0, 0)とするスクリーン座標で行ないます。screenを指定するとスクリーン座標、指定しない場合はワールド座標による設定になります。また、step (X, Y)は現在のグラフィックポイントに対する相対座標指定になります。

関数

POSH （ポジション ホリゾンタル・position horizontal）
POSV （ポジション バーチカル・position vertical）

グラフィックポイントの位置を求めます。

書 式

posh[(n)]
posv[(n)]

説 明

- グラフィックポイントの現在位置を求める関数です。poshは水平位置、posvは垂直位置を求めます。
- 引数のnは0を指定するとワールド座標、1を指定するとスクリーン座標での値になります。引数を省略してposh, posvとすると、0が指定されたものとしてワールド座標値で求めます。

GET@ （ゲット アットマーク）

グラフィック画面データを配列に取り込みます。

書　式

get@ X_1, Y_1, X_2, Y_2, 〈配列変数名〉[(〈要素番号〉)]

説　明

- ビューポート内の左上を(0, 0)とするスクリーン座標上の(X_1, Y_1)から(X_2, Y_2)のデータを数値型配列変数に取り込みます。
- 座標の指定はstep (X, Y)による相対座標指定が可能です。
- データは16色モードの場合、範囲の左上のドットから右方向に8ドットずつ、各プレーンをB, R, G, Iの順に読み取って行き、右端まで読むと次のラインの左端に行き8ドットずつ右へと読んで行きます。水平ドット数が8で割り切れない場合は範囲をこえたデータもそのまま読み取られます。(256色モードのときは$B_1, R_1, G_1, I_1, B_0, R_0, G_0, I_0$の順に8プレーン、4色モードはB, Rの2プレーンが読み取られます。)
- データを読み取る配列を＜配列変数名＞で指定し、＜要素番号＞は読み取ったデータの記憶を開始する要素番号を指定します。＜要素番号＞の指定を省略した場合はoption base文で指定されている下限より記憶されます。
- 読み取りデータの記憶される配列は、データのバイト数に応じた宣言をしておく必要があります。必要なバイト数は水平方向のドット数を8で割り、余りがある場合は1を加え、ライン数とプレーン数を掛け合わせ、5を加えた数になります。この必要バイト数を配列要素1つ当たりのバイト数で割り、余りが出た場合に1を加えた数が必要要素数になります。要素1つ当りのバイト数は整数型は2、単精度型は5、倍精度型は8になります。

  また、次の式で求めることができます。

  必要要素数＝(((abs(X_2−X_1)＋8)¥8)＊(abs(Y_2−Y_1)＋1)＊pl＋5＋(m−1))¥m

  pl…プレーン数　　m…要素ひとつ当たりのバイト数
- 読み取った8ドット単位のデータは表示上の左側がLSB、右側がMSBになります。
- この命令で読み取ったデータはput@文によりビューポート内の任意の位置へ表示することができます。

参　考☞

put@　…get@文で読み取ったデータの表示。

PUT@ （プット アットマーク）

グラフィックパターンの表示を行ないます。

書　式

put@ X, Y, 〈配列変数名〉〔(〈要素番号〉)〕 …(1)

put@ 〔〔〔〈フォント色〉〕〔,〈機能〉〕〔,〈バック色〉〕〕〕X, Y, Kanji(〈漢字コード〉) …(2)

説　明

- 書式(1)はget@文により、配列に取り込まれたデータをビューポート内の左上を(0, 0)とするスクリーン座標(X, Y)の位置より表示する機能です。
- 座標の指定は、step (X, Y)による相対座標指定が可能です。
- ＜配列変数名＞でデータの記憶されている配列名を指定し、＜要素＞がデータの開始要素を示します。＜要素＞を省略するとoption base文で指定されている配列の下限よりデータを取り出します。
- 書式(2)はビューポート内の左上を(0, 0)とするスクリーン座標(X, Y)の座標に漢字を1文字出力する機能です。座標(X, Y)は文字の左上の位置になります。
- ＜フォント色＞はドット色の部分の色。＜バック色＞はそれ以外の部分の色を指定します。＜フォント色＞を省略した場合は、color文で指定されている＜描画色＞になり＜バック色＞の省略値は0になります。
- ＜漢字コード＞はJISコードで表した整数型データで指定します。
- ＜機能＞は0～3の値で指定し、省略した場合は、color文で設定された値になります。
- ＜フォント色＞と＜バック色＞は、4, 16色モードではパレットコード、256色モードでは512色系のカラーコードで指定します。
  (この機能は漢字のコードにJISコードを使用しています。内部と同じシフトJISコードでお使いになる場合はsymbol文を使用してください。)

参　考☞　get@ …… グラフィックデータを配列に読み取る。

GSAVE （グラフィック セーブ・graphic save）

グラフィック表示をファイルに書き込みます。

書　式　gsave 〈ファイル指定〉〔,X_1,Y_1,X_2,Y_2〕

省略形　gs.

説　明

- ビューポート内の左上を(0, 0)とするスクリーン座標(X_1, Y_1)と(X_2, Y_2)の範囲のグラフィックデータをファイルとして記録します。
- 記録範囲を省略するとビューポートの全体を記憶します。
- この命令により記録される画面は screen 文により設定されている＜アクティブスクリーン＞の全プレーンデータです。
- ＜ファイル指定＞は次のようになり、記録したファイルはBRD形式のファイルになります。

  "〔＜装置名＞〕〔＜ディレクトリパス＞〕＜ファイル名＞"

  装置としてはFD1:～FD4:, MEM:が使用できます。

例

```
gsave "FD1:Argo",0,0,100,100
```

…… (0, 0)と(100, 100)の範囲の図形を"Argo"というファイル名でフロッピーディスクに登録します。

参　考☞　gload …… gsave文で書き込んだデータの呼び出し。

GLOAD （グラフィック ロード・graphic load）

ファイルからグラフィックデータを呼び出します。

書　式　gload 〈ファイル指定〉〔,X,Y〕

省略形　gl.

説　明

- gsave文で記録されたグラフィックファイルからデータを呼び出しグラフィック画面に表示します。
- 表示する左上の位置を、ビューポートの左上を(0,0)とするスクリーン座標(X,Y)で指定します。座標を省略すると書き込まれたのと同じ位置に表示します。
- ＜ファイル指定＞は次のようになります。
  "〔＜装置名＞〕〔＜ディレクトリパス＞〕＜ファイル名＞"
- 呼び出したデータが描かれるのは、screen文で設定されている＜アクティブスクリーン＞の＜アクティブプレーン＞になります。
- gsave文でファイルに書き込んだときの画面モードと、呼び出したときの画面モードが異なっている場合は、正常に描くことはできませんので注意してください。

例

```
gload "FD1:Argo"
```

……"Argo"というファイルから図形データを呼び出して、登録時と同じ位置に描きます。

参　考　gsave …… グラフィック画面をファイルに書き込む。

INIT "FDn:" (イニシャライズ フロッピーディスク・initialize Floppy disk drive)

フロッピーディスクドライブの種類を選択します。

書 式

init "〔FD〕n:{〈ドライブタイプ〉 / 〈シリンダ数〉,〈ステップレート〉,〈タイミング〉}"

説 明

- この命令は、ドライブ番号ごとに<ドライブタイプ>による代表的な機種での設定、または個別に各種の容量や各種のタイミングの設定を行ないます。(外部に別のフロッピーディスクドライブを接続した場合、使用するディスクドライブの種類により、記憶容量や各種コントロールタイミングが異なりますので、フロッピーディスクドライブに合わせた設定が必要になります。)
- フロッピーディスクドライブの装置名は"FD1："～"FD4："の4種でドライブ番号1～4に対応しています。"FD1："は、ドライブ番号を省略して"FD："として表すことができます。また、"FD："を省略して"1："～"4："のドライブ番号のみで表すこともできます。
- <ドライブタイプ>はF1～F3でフロッピーディスクドライブの選択をします。指定を行なうことにより次のように各設定が行なわれます。

<ドライブタイプ>と指定時の設定値

<ドライブタイプ>	機　種　名	シリンダ数	ステップレート	タイミング
F1	内蔵ドライブ	80	6msec.	F
F2	MZ-1F07/02	40	6msec.	F
F3	MZ-80BF	35	30msec.	S

- <シリンダ数>は、80, 40, 35のいずれかで指定します。
- <ステップレート>は、6msec., 12msec., 20msec., 30msec. の4種の設定を6, 12, 20, 30のいずれかで指定します。
- <タイミング>は、各種タイミングをF(fast)またはS(slow)で指定します。

F …… MOTOR-ON後の時間は 500msec. セットリングタイムは15msec. に設定されます。

S …… MOTOR-ON後の時間は1sec. セットリングタイムは60msec. に設定されます。

例

```
init "FD3:40,20,S"
```

……ドライブ番号3のフロッピーディスクドライブを40シリンダでステップレート20msec. タイミングをslowに設定します。

INIT "MEM:" (イニシャライズ メモリファイル・initialize memory file)

メモリディスクやスプーラの容量を設定します。

書　式

init "MEM:〔〈メモリディスクサイズ〉〕〔,〈スプーラサイズ〉〕"

説　明

- ＜メモリディスクサイズ＞はメモリをフロッピーディスクドライブと同じように、ファイル装置として使用する機能に対して割り当てるメモリサイズをキロバイト単位で0～128までの値で指定します。
- ＜スプーラサイズ＞は印字用のバァッファサイズをキロバイト単位で0～64までの値で指定します。
- それぞれの指定は0から最大値までの任意の整数で行ないますが、実際には、0以外は8Kバイトごとに切り上げて設定されます。また、0を指定するとこの機能は解除され(以前に設定されていたものは消去し)初期化します。
- それぞれで指定されたサイズのメモリはoption screen文で設定されたグラフィックV-RAMの表示に使用されないエリアを使用します。ただしV-RAMのエリアがこの命令で指定されたサイズより少ない場合はメインメモリも合わせて使用します。
- この命令で設定可能なメモリ容量は、size関数で調べることができます。
- ＜メモリディスクサイズ＞または＜スプーラサイズ＞の指定を省略した場合、省略した方はそれまでの設定値のままで内容の初期化は行なわれません。
- BASICの起動時はinit "MEM：0,0"に設定されます。
- ＜メモリディスクサイズ＞を設定した場合は、それ以後"MEM:"の装置名でファイル装置として使用できます。
- ＜スプーラサイズ＞を設定した場合は、init "LPT:"文により、スプーラ機能の設定をします。

参　考☞

init "LPT:" …… プリンタスプーラの設定。

option screen …… グラフィックV-RAMを、メモリディスクやスプーラ用メモリに設定する。

INIT "COMn:" (イニシャライズ コミュニケーションポート・initialize communication port)

RS-232Cインターフェイスの各種設定をします。

書　式

init "COMn:〈ボーレイト〉,〈パリティ〉〈データビット長〉
〈ストップビット長〉〈通信制御〉〈カナ〉〈DEL受信〉〈CR送信〉
〈CR受信〉〈全角文字〉〈エンドコード〉"

説　明

- RS-232Cインターフェイスの通信機能を設定します。
- "COM1："または"COM："としたときはチャンネルA、"COM2："としたときはチャンネルBへの指定になります。
- RS-232Cインターフェイスの通信条件や制御信号の設定を次のように行ないます。〔 〕内は省略値です。設定の詳細はBASIC-M25マニュアルの付録の「RS-232Cインターフェイスとターミナルモード」を参照してください。

＜ボーレイト＞	75, 150, 300, 600, 1200, 2400, 4800, 〔9600〕, 19200
＜パリティ＞	E　O 〔N〕
＜データビット長＞	7 〔8〕
＜ストップビット長＞	1　2 〔3〕　(1, 1$\frac{1}{2}$, 2ビットに対応します。)
＜通信制御＞	X　R 〔N〕
＜カナ＞	S 〔N〕
＜DEL受信＞	N　B 〔D〕
＜CR送信＞	C 〔L〕
＜CR受信＞	C 〔L〕
＜全角文字＞	J　M　P 〔N〕
＜エンドコード＞	文字コードで&H40～5Fまたは&H60～7Fに相当する1文字で指定し、受信時の&H0～&H1Fに対応します。

ボーレイトのあとにはカンマ(,)が必要で、ほかの指定はすべて1文字になります。また、それぞれの指定は省略でき、省略時はそれぞれ省略値が設定されますが、途中の指定を省略することはできません。

- この命令での設定は、装置名として"COMn："を使用するコマンドやファイルオープンに対して有効になります。termコマンドでRS-232Cインターフェイスを使用した場合は設定値が変わりますので注意してください。
- マウスを使用している場合は"COM2："を使用することはできません。あらかじめmouse 6を実行してマウスの使用を終了しておいてください。

例

```
init "COM:9600,N83XNDLLND"
```

DIR （ディレクトリ・directory）

登録されているファイルを表示します。

書　式

dir〔/p〕 "〔〈装置名〉〕〔〈ファイル名〉〕"

〈装置名〉 …… FD1:～FD4:, MEM:, CMT:

説　明

- dirとすると指定装置に登録されているファイルを画面に表示します。dir/pとするとプリンタに印字します。
- 表示は装置内の記録残量を表示したのちファイルごとのファイル形式、プロテクトの有無、ファイル名を表示します。なお、ファイル表示の順番はファイル名の文字コード順に並びかえて行なわれます。
- ＜装置名＞を省略するとchdirコマンドで設定されているカレントディレクトリのファイルを表示します。
- BREAK キーによる表示の一時停止、 SHIFT ＋ BREAK キーによる実行の中断が可能です。
- ＜ファイル名＞を指定しない場合はディレクトリ内のすべてのファイルを表示し、＜ファイル名＞を指定する場合、特定のファイルのみを表示することができます。また、ファイル名の指定に省略記号が使え、次の方法で特定の文字を含むファイルをすべて表示することができます。？や＊記号をワイルドカードと呼び複数のファイルを指定することができます。

  ？ …… この記号の部分は1文字分の代用を意味します。

  例） "A？？Z"と指定すると次のようなものが該当します。

  "AAZZ", "AXYZ", "AtoZ"

  ＊ …… この記号は0～複数文字の代用になります。

  例） "A＊Z"と指定すると次のようなものが該当します。

  "AZ", "ABCXYZ", "A＋Z"

- 階層ディレクトリが可能な場合は＜装置名＞と＜ファイル名＞の間に＜ディレクトリパス＞を指定できます。（chdirコマンド参照）

例

```
dir "FD1:"
```
…… FD1:にセットしているフロッピーディスクのディレクトリを表示します。

```
dir/p
```
…… プリンタにカレントディレクトリ内のすべてのファイルを表示します。

参　考☞　chdir　カレントディレクトリの変更。

CHDIR （チェンジディレクトリ・change directory）

カレントディレクトリを変更します。

書　式

chdir { "〈装置名〉" / "〔〈装置名〉〕〈ディレクトリパス〉" }

〈装置名〉 …… FD1:～FD4:, MEM:

省略形

ch.

説　明

- この命令により、カレントディレクトリを変更します。カレントディレクトリとは現在参照しているディレクトリのことでdirコマンドやファイルオープンなどのファイル指定を行なう命令で、＜装置名＞と＜ディレクトリパス＞を省略した場合に対象となるディレクトリのことです。
- ＜装置名＞のみを指定した場合は指定された装置のルートディレクトリにカレントディレクトリを変更します。
- ＜ディレクトリパス＞は階層ディレクトリのパス（道すじ）を指定します。＜ディレクトリパス＞を指定した場合は、その指定ディレクトリを新たなカレントディレクトリとします。＜装置名＞の指定または＜ディレクトリパス＞の最初にスラッシュ（/）を指定した場合は、ルートディレクトリからの指定になり、それらを省略するとカレントディレクトリより＜ディレクトリパス＞をたどり指定されたディレクトリにカレントディレクトリを変更します。
- ＜装置名＞や＜ディレクトリパス＞をすべて省略した場合は、現在の装置のルートディレクトリにカレントディレクトリを変更します。
- カレントディレクトリが、ルートディレクトリ以外にある場合にフロッピーディスクを入れ換えて次のファイル操作を行なうとエラーになります。このような場合、chdirとしてあらかじめカレントディレクトリをルートディレクトリに戻してから、次のファイル操作を行なってください。

例

```
chdir "FD2:"
```

…… フロッピーディスク装置の2番へカレントディレクトリを移します。

```
chdir "FD1:住所録/京都"
```

…… フロッピーディスク装置の1番のルートディレクトリより、住所録ー京都とディレクトリをたどって京都のディレクトリにカレントディレクトリを移します。

参　考

pwd$ …… カレントディレクトリのディレクトリパスを求めるシステム変数。

MKDIR （メイク ディレクトリ・make directory）

ディレクトリファイルを作成します。

書　式

mkdir "〔〈装置名〉〕〈ディレクトリパス〉〈ディレクトリ名〉"

〈装置名〉 …… FD1：～FD4：, MEM：

省略形

mk.

説　明

- 新たなディレクトリを作成し、親ディレクトリにディレクトリファイルとして登録します。
- DIR形式のファイルの消去はrmdirコマンドで行ないます。deleteコマンドでは消去できません。
- ＜装置名＞を省略すると、カレントディレクトリがある装置の指定になります。
- ＜ディレクトリパス＞はディレクトリファイルを登録する親ディレクトリを指定します。＜装置名＞が指定されている場合や＜ディレクトリパス＞の最初にスラッシュ(/)がある場合はルートディレクトリからパス(道すじ)をたどり、それ以外の場合は、カレントディレクトリよりパスをたどります。
- ＜装置名＞と＜ディレクトリパス＞の指定がない場合は、chdirコマンドで設定されている、カレントディレクトリの指定になります。
- ＜ディレクトリ名＞は＜装置名＞や＜ディレクトリパス＞によって指定されたディレクトリ内に作成されるディレクトリファイルのファイル名を指定します。
- 新たなディレクトリ内には"."と".." の2つのDIR形式のファイルが作成されます。".."は親ディレクトリを管理するファイルで、カレントディレクトリ内に、このファイルがある場合、chdir ".."とすると親ディレクトリへカレントディレクトリを移すことができます。 "." は自身のディレクトリを管理するファイルです。

参　考

rmdir …… ディレクトリファイルの消去。

chdir …… カレントディレクトリの変更。

RMDIR （リムーブ ディレクトリ・remove directory）

ディレクトリファイルを削除します。

書　式　rmdir "〔〈装置名〉〕〔〈ディレクトリパス〉〕〈ディレクトリ名〉"

〈装置名〉 …… FD1：～FD4：, MEM：

省略形　rm.

説　明

- ディレクトリファイルの削除を行ないます。
- ＜装置名＞を省略すると、カレントディレクトリがある装置の指定になります。
- ＜ディレクトリパス＞はディレクトリファイルが登録されている親ディレクトリを指定します。＜装置名＞が指定されている場合や＜ディレクトリパス＞の最初にスラッシュ(/)がある場合はルートディレクトリからパス(道すじ)をたどり、それ以外の場合は、カレントディレクトリよりパスをたどります。
- ＜装置名＞と＜ディレクトリパス＞の指定がない場合は、chdirコマンドで設定されている、カレントディレクトリの指定になります。
- ＜ディレクトリ名＞は＜装置名＞や＜ディレクトリパス＞によって指定されたディレクトリ内で消去されるディレクトリファイルのファイル名を指定します。
- 削除するディレクトリ内に"."と".."以外のファイルが残っていると削除できません。あらかじめ削除するディレクトリ内のファイルを消去しておく必要があります。

参　考☞　mkdir …… ディレクトリファイルの作成。

RUN 〈ファイル〉 (ラン)

指定ファイルを読み込み実行します。

書 式

run "〔〈装置名〉〕〈ファイル名〉"〔,C〕

〈装置名〉 …… FD1：～FD4：,MEM：,COM1：～COM2：,CMT：

省略形

r.

説 明

- 指定されたプログラムをメモリ上に読み込み、実行します。(loadおよびrun)
- BTX形式またはBSD形式のBASICプログラムを指定した場合は、指定ファイルを見つけたあとに現在のプログラムを消去(new)して、指定ファイルを読み込み実行(run)します。したがって、指定プログラムが見つからない場合はエラーになり、現在のプログラムは消去されません。
- OBJ形式の機械語プログラムを指定した場合は、limit文で指定された機械語エリアまたはBASICのフリーエリアへプログラムを読み込み実行します。機械語エリアやフリーエリアが不足しプログラムが読み込めない場合は、no memory errorになり、機械語プログラムの読み込むアドレスが&H4000以下であればmemory protection errorになります。
  機械語プログラムの読み込みアドレスや実行アドレスは機械語プログラムの登録時の設定によります。実行の終了は機械語プログラム中のRET(リターン)命令によりBASICに戻ります。
- Cの指定は機械語プログラムの実行時に行ない、指定されるとBASICインタプリタを消去して機械語プログラムを読み込み、実行を行ないます。この場合はmemory protection errorは出ません。ただし、BASICインタプリタは消去されていますので、機械語プログラム中のRET命令で終了してもBASICへは戻れません。
- ファイルの指定で＜装置名＞を省略した場合は、chdir文で指定されているカレントディレクトリから指定ファイル名をさがします。階層ディレクトリでファイルを指定する場合は＜装置名＞と＜ファイル名＞の間にディレクトリパスを指定します。

例

```
run "日本語ワープロ"
```

参 考

load …… プログラムの呼び出し。

run(コマンド) …… プログラムの実行。

LOAD （ロード）

プログラムファイルをメモリに読み込みます。

書　式　load　"〔〈装置名〉〕〈ファイル名〉"〔,〈アドレス〉〕

〈装置名〉　…FD1:～FD4:, MEM:, COM1:～COM2:, CMT:

省略形　lo.

説　明

- 指定装置より、BASICまたは機械語のプログラムを読み出します。
- ファイルの指定で＜装置名＞を省略した場合は、chdirコマンドで指定されているカレントディレクトリから指定ファイル名をさがします。階層ディレクトリでファイルを指定する場合は＜装置名＞と＜ファイル名＞の間にディレクトリパスを指定します。
- ＜ファイル名＞は、"CMT:"と"COM1："、"COM2："以外の装置では省略できません。"CMT:"で＜ファイル名＞省略した場合には、最初に見つかったファイルを呼び出します。
- BTXまたはBSD形式のBASICプログラムの場合、指定されたファイルをみつけたあとにメモリ上のプログラムを消去(new)して指定ファイルを読み込みます。したがって、指定ファイルがみつからなかった場合は、メモリ上のプログラムは消去されません。
- BSD形式のファイルも呼び出すことができますが、内容がBASICプログラムになっていない場合は途中でエラーを出し、実行を中止します。
- ＜装置名＞が"COM1:"～"COM2:"の場合はBSD形式のBASICプログラムの入力とみなし、＜ファイル名＞の指定は無視します。
- ＜アドレス＞は、OBJ形式の機械語ファイルの読み込み先頭アドレスの指定です。指定を省略した場合は、指定した機械語ファイルを作成したときのアドレスへ読み込みます。機械語プログラムを読み込むエリアは、あらかじめlimit文で設定しておいてください。

例

```
load "auto-run.s25"
```

…… "auto-run. s25"というファイル名の付いたプログラムを呼び出します。

参　考☞　save …… プログラムの記録

SAVE （セーブ）

プログラムを指定の装置に記録します。

書　式

save　"〔〈装置名〉〕〈ファイル名〉"〔,A〕　…(1)

save　"〔〈装置名〉〕〈ファイル名〉",〈開始アドレス〉,〈サイズ〉
〔,〈実行アドレス〉〕〔,〈読み出しアドレス〉〕　…(2)

〈装置名〉 …… FD1:～FD4:, MEM:, COM1:～COM2

省略形

sa.

説　明

- 書式(1)はBASICプログラムの記録です。最後のAを省略するとBTX形式のファイルになり、Aを指定するとBSD形式で記録します。
- 一つのディレクトリ内で同じファイル名を使うことはできません。
- ＜装置名＞を省略するとchdirコマンドで指定されているカレントディレクトリに登録されます。階層ディレクトリのディレクトリ階層を指定する場合は＜装置名＞と＜ファイル名＞の間にディレクトリパスを指定します。
- ＜装置名＞が"COM1:"～"COM2:"の場合はAを指定してBSD形式にしてください。
- 書式(2)はlimit文で指定している機械語エリアのプログラムをファイル名と開始アドレス、サイズ、実行アドレス、読み出しアドレスを指定しファイルとして登録します。
- ＜開始アドレス＞はメモリ上にある機械語プログラムの最初のアドレスを指定します。
- ＜サイズ＞は記録する機械語プログラムのバイト数を指定します。
- ＜実行アドレス＞はrun　＜ファイル＞コマンドによりプログラムをメモリ上に読み込んだあとに実行を開始するアドレスを指定します。この指定を省略すると、＜開始アドレス＞により指定されたアドレスが設定されます。
- ＜読み出し＞は、この命令により記録した機械語プログラムをrunコマンドやloadコマンドにより機械語プログラムを読み出すときの、メモリの先頭アドレスを指定します。
- 各アドレスは、整数型データで指定しますので、10進数では-32768～32767の値になりますが、この命令では32768～65535(&H8000～&HFFFFに対応)の値で指定することもできます。

参　考☞

run　＜ファイル＞ …… プログラムファイルのload&run。

MERGE （マージ）

プログラムを併合します。

書　式　merge "〔〈装置名〉〕〈ファイル名〉"

〈装置名〉 …FD1:～FD4:, MEM:, COM1:～COM2:, CMT:

省略形　m.

説　明

- メモリ上にあるプログラムに指定したプログラムファイルの内容を併合(合成)します。
- プログラムファイルの形式はBTXとBSDの両方のものが使えます。ただし、BSD形式のものでプログラムの書式に合っていないものは読み込み途中でエラーが起こる場合があります。
- 内容の併合はメモリ上のプログラムにプログラムファイルの内容が追加されるようになり、追加する側の行番号が追加される側の行番号と同じであれば結果は追加する側の内容になります。
- ＜装置名＞を省略した場合はchdirコマンドにより設定されているカレントディレクトリより指定ファイルを読み出します。
- 階層ディレクトリでファイルを指定する場合は＜装置名＞と＜ファイル名＞の間に＜ディレクトリパス＞を指定します。
- ＜装置名＞に"COM1:"～"COM2:"を指定するとBSD形式のファイルとみなし＜ファイル名＞は無視されます。

例　merge "ABC" …… 現在メモリ上にあるプログラムに"ＡＢＣ"というプログラムを併合します。

参　考☞　chain merge …… 指定プログラムを併合して実行する。

DELETE （デリート）

ファイルを消去します。

書　式

delete　"〔〈装置名〉〕〈ファイル名〉"

＜装置名＞　……　FD1:～FD4:, MEM:

省略形

d.

説　明

- 指定されたファイルを消去し登録を抹消します。
- ＜装置名＞の指定を省略するとchdirコマンドで指定されているカレントディレクトリ内の＜ファイル名＞で指定されたファイルを消去します。また、階層ディレクトリによりファイル指定を行なう場合は＜装置名＞と＜ファイル名＞の間にディレクトリパスを指定します。
- 指定されたファイルが存在しない場合はエラーとなります。
- 指定されたファイルがプロテクトされている場合は消去されずにエラーとなります。unlockコマンドによりあらかじめプロテクトをはずしておく必要があります。
- DIR形式のファイルはこのコマンドでは消去できません。（rmdir）

例

`delete "ABC"`　……　ファイル名が"ABC"のファイルを消去します。

参　考

unlock　……　ファイルのプロテクトをはずす。

rmdir　……　ディレクトリファイルの消去。

RENAME （リネーム）

ファイルの登録名を変更します。

書　式

rename　"〔〈装置名〉〕〈ファイル名〉","〈新ファイル名〉"

〈装置名〉 …… FD1：～FD4：,MEM：

省略形

rena.

説　明

- すでに登録してあるファイルの名前を＜新ファイル名＞に変更します。
- ＜装置名＞の指定を省略するとchdirコマンドで指定されているカレントディレクトリ内のファイル名で指定されたファイルをさがします。また、階層ディレクトリによりファイル指定を行なう場合は＜装置名＞と＜ファイル名＞の間にディレクトリパスを指定します。
- ＜新ファイル名＞の指定は、16バイト以内の半角文字および全角文字を使用します。ただし、":"と"/" の2つの半角文字は使用できません。
- プロテクトしてあるファイルに対して行なった場合は、エラーになります。あらかじめunlockコマンドによりプロテクトをはずしておく必要があります。

例

```
rename "ABC","abc"
```

…… "ABC"のファイル名を"abc"に変更します。

参　考☞

lock/unlock …… ファイルのプロテクトを設定する。

LOCK （ロック）
UNLOCK （アンロック）

ファイルにプロテクトをかけます。

書 式

{lock / unlock} "〔〈装置名〉〕〈ファイル名〉"

〈装置名〉 …… FD:1～FD4:, MEM:

説 明

- ファイルを誤操作によって消去したり、書き換えてしまうのをふせぐためにファイルごとにプロテクトをかけることができます。lockとするとそのファイルはプロテクト状態になり、dirコマンドによるファイル表示のときに、BTXやBSDなどのファイル形式のあとに＊を表示します。
- プロテクトされたファイルに対しては次の操作を禁止します。

  deleteコマンドによる消去。

  renameコマンドによる名前の変更。

  aopenによるファイルへの追加書き込み。

  ランダムアクセスファイルへの書き込み。(xopen時のput ＃やprint ＃)

  以上の操作を行なうとwrite protect errorになります。
- プロテクトの解除はunlockとします。
- ＜装置名＞の指定を省略するとchdirコマンドで指定されているカレントディレクトリ内のファイル名で指定されたファイルをさがします。また、階層ディレクトリによりファイル指定を行なう場合は＜装置名＞と＜ファイル名＞の間にディレクトリパスを指定します。

例

lock "ABC" …… ファイル名が"ABC"のファイルにプロテクトをかけます。

参 考☞

attr$ …… ファイルのプロテクト状態を調べる関数。

VERIFY ON/OFF （ベリファイ オン/オフ）

フロッピーディスクへの書き込みチェックを指定します。

書　式

verify {on / off}

省略形

ve.

説　明

- フロッピーディスクへの書き込みを行なったあとに、書き込んだデータを読み出してチェックを行なうかどうかを指定します。指定をonとするとチェックを行ない、offとするとチェックを行ないません。
- BASIC起動時はverify onとなっています。
- 書き込みを行なったあとにチェックを行なうことをリードアフターライト（read after write）といいます。

例

```
verify off
```
…… 以後リードアフターライトを行ないません。

CHAIN （チェイン）

現在実行中のプログラムから指定プログラムファイルを読み込み実行します。

書　式

chain 〈ファイル指定〉〔,〈実行開始行〉〕〔,all〕
chain merge 〈ファイル指定〉〔,〈実行開始行〉〕〔,all〕〔,delete 〈削除範囲〉〕

省略形

cha.

説　明

- chainは現在実行中のプログラムを消去して、指定プログラムファイルを読み込み実行します。
- ＜実行開始行＞は、指定プログラムを読み込んだあとに実行を開始する行を、行番号またはラベルで指定できます。省略した場合はプログラムの最初から実行します。なお、省略する場合はカンマ(,)も含めて省略してください。
- allを指定した場合は呼び出したプログラムにすべての変数や配列を引き渡します。allを指定しない場合はcommon文で指定されている変数や配列以外の変数や配列を消去します。ただし、allを指定しない場合でもcommon文が実行されていない場合はすべての変数や配列を引き渡します。
- chain mergeは現在のプログラムを消去せずに指定ファイルを現在のプログラムに併合します。この場合はdeleteの指定によって、あらかじめ現在のプログラムの一部を消去しておくことができます。＜削除範囲＞の指定は＜開始行＞－＜終了行＞でdeleteコマンドと同じです。
- ＜ファイル指定＞は次の書式の文字列型データで指定します。
  "〔＜装置名＞〕〔＜ディレクトリパス＞〕＜ファイル名＞"
  〈装置名〉　…FD1:～FD4:, MEM:, COM1:～COM2:, CMT:
- ＜ディレクトリパス＞は階層ディレクトリのパス(道すじ)を指定します。(chdirコマンド参照)
- ＜装置名＞と＜ディレクトリパス＞の指定を省略すると、chdirコマンドで設定されているカレントディレクトリ内でのファイル指定になります。

例

```
100 chain "ABC",all
```

…… 変数や配列を消去せずに"ABC"のプログラムを読み込み、プログラムの最初より実行を開始します。

参　考

merge　…… プログラムの併合。
common　…… 共通変数の定義。

COMMON （コモン）

共通変数の定義を行ないます。

書　式　common 〈変数名〉[,〈変数名〉] …

省略形　com.

説　明

- chain文の実行を行なったときに、次のプログラムへ引き渡す変数や配列を定義します。次のプログラムへ移ったとき、この命令で定義した変数や配列は、消去されずそのままの変数名や配列名で値が残ります。
- ＜変数名＞に配列変数を指定する場合は、配列名の後に( )を付けます。
- chain文ではcommon文で指定されている変数や配列を残すか、またはすべての変数や配列を残すかの選択ができます。

例

```
common A,B$,X()
```

…… 変数AとB$、配列X( )を共通変数として定義します。

参　考☞　chain …… 新しいプログラムを読み込み、実行する。

SWAP （スワップ）

指定プログラムファイルを一時的に呼び出し実行します。

書　式　swap〈ファイル指定〉

省略形　sw.

説　明

- この命令を実行すると、最初に現在実行中のプログラムをカレントディレクトリが設定されている装置に記録して退避させます。次に指定プログラムファイルを呼び出し、そのプログラムの最初から実行します。呼び出したプログラムが終了すると、呼び出したプログラムを消去して、最初の退避させたプログラムを呼び出し、次の文から実行を続けます。
- ＜ファイル指定＞は次の書式の文字列型データで指定します。
  "〔＜装置名＞〕〔＜ディレクトリパス＞〕＜ファイル名＞"
  〈装置名〉　…FD1:～FD4:, MEM:
- ＜ディレクトリパス＞は階層ディレクトリのパス（道すじ）を指定します。（chdirコマンド参照）
- ＜装置名＞と＜ディレクトリパス＞の指定を省略すると、chdirコマンドで設定されているカレントディレクトリ内でのファイル指定になります。
- 変数の内容や各種の定義は消去されずにそのまま呼び出したプログラムに引き渡されますのでプログラムファイルをサブルーチンとして呼び出すとみることができます。
- 呼び出したプログラムから元のプログラムに戻るのはend文またはテキストの終了により行なわれます。
- 呼び出したプログラム内でこの命令を実行することはできません。（ネスティングレベルは１のみ）
- 呼び出したプログラムの実行中にエラーが起こった場合や、実行を中止させ編集等を行いcontコマンドの実行が不可能になった場合は、メモリー上には呼び出されたプログラムが残り、元のプログラムを呼び出すことができなくなります。

例

```
1000 swap "XYZ"
```

…… "XYZ"というファイルを一時的に呼び出します。

参　考☞　chain …… 新しいプログラムを読み込み、実行する。

WOPEN （ライト オープン・write open）

ファイルの書き込みを開始します。

書 式　wopen　＃〈ファイル番号〉,〈ファイル指定〉

省略形　wo.

説 明

- シーケンシャルアクセスファイルの書き込みを開始するために＜ファイル番号＞とファイル名を指定します。
- この命令を実行したあとは、print ＃文でファイル番号を指定することにより、オープンされたファイルにデータを書き込むことができます。
- ＜ファイル番号＞は、1～127の任意の整数で指定します。ただし、すでに別のファイルオープンで使用中（close文またはkill文でファイル操作を終了していない）のファイル番号は、使用できません。
- close文によってファイルを閉じることによりファイル処理が終了して正式にファイルの登録が完了します。ファイルの終了処理を行わずにディスクを交換するとファイルの記録が正常に行なわれず双方のディスクの内容が破壊されますので注意が必要です。
- ＜ファイル指定＞は次の書式の文字列型データで指定します。
  "〔＜装置名＞〕〔＜ディレクトリパス＞〕＜ファイル名＞"
  〈装置名〉 …FD1:～FD4:, MEM:, COM1:～COM2:, CRT:, LPT:
- ＜ディレクトリパス＞は階層ディレクトリのパス（道すじ）を指定します。（chdirコマンド参照）
- ＜装置名＞と＜ディレクトリパス＞の指定を省略すると、chdirコマンドで設定されているカレントディレクトリ内でのファイル指定になります。
- すでに＜ファイル指定＞と同じファイル名のファイルが存在すればエラーになります。

サンプル

```
10 rem --- wopen ---
20 wopen #1,"test data 1"
30 for A=1 to 10
40   print A,A*A
50   print #1,A,A*A
60 next
70 close #1
80 ' このデータは ropen
90 ' のｻﾝﾌﾟﾙで読みだせます
```

ROPEN （リード オープン・read open）

ファイルの読み出しを開始します。

書 式　ropen　#〈ファイル番号〉,〈ファイル指定〉

省略形　ro.

説 明

- シーケンシャルアクセスファイルの読み出しを開始するために＜ファイル番号＞とファイル名を指定します。
- ＜ファイル番号＞は、1～127の任意の整数で指定します。ただし、すでに別のファイルオープンで使用中（close文またはkill文でファイル操作を終了していない）のファイル番号は、使用できません。
- ＜ファイル指定＞は次の書式の文字列型データで指定します。

  "〔＜装置名＞〕〔＜ディレクトリパス＞〕＜ファイル名＞"

  〈装置名〉　…FD1:～FD4:, MEM:, COM1:～COM2:, CMT:, KB:
- ＜ディレクトリパス＞は階層ディレクトリのパス（道すじ）を指定します。（chdirコマンド参照）
- ＜装置名＞と＜ディレクトリパス＞の指定を省略すると、chdirコマンドで設定されているカレントディレクトリ内でのファイル指定になります。
- この命令を実行した後は、input #文でファイル番号を指定することにより、オープンされたファイルからデータを読み取ることができます。
- 指定ファイルが存在しない場合はエラーになります。

サンプル

```
10 rem --- ropen ---
20 ropen #1,"test data 1"
30 for A=1 to 10
40   input #1,X,Y
50   print X,Y
60 next
70 close #1
```

AOPEN （アペンド オープン・append open）

ファイルの追加書き込みを開始します。

書　式　aopen　＃〈ファイル番号〉,〈ファイル指定〉

省略形　ao.

説　明

- すでに登録してあるシーケンシャルアクセスファイルに対してデータの追加書き込みを開始するために＜ファイル番号＞とファイル名を指定します。
- この命令を実行した後は、**print** ＃文でファイル番号を指定することにより、オープンされたファイルにデータを書き込みます。
- ＜ファイル番号＞は、1～127の任意の整数で指定します。ただし、すでに別のファイルオープンで使用中（**close**文または**kill**文でファイル操作を終了していない）のファイル番号は、使用できません。
- ＜ファイル指定＞は次の書式の文字列型データで指定します。
  "〔＜装置名＞〕〔＜ディレクトリパス＞〕＜ファイル名＞"
  〈装置名〉　…**FD1:～FD4:, MEM:**
- ＜ディレクトリパス＞は階層ディレクトリのパス（道すじ）を指定します。（**chdir**コマンド参照）
- ＜装置名＞と＜ディレクトリパス＞の指定を省略すると、**chdir**コマンドで設定されているカレントディレクトリ内でのファイル指定になります。
- **close**文によってファイルを閉じることによりファイル処理が終了して正式にファイルの登録が行なわれます。ファイルの終了処理を行なわずにディスクを交換するとファイルの記録が正常に行なわれずに双方のディスクの内容が破壊されますので注意が必要です。
- 指定されたファイルが登録されていないか、ファイルの種類が異なる場合、またはファイルがプロテクトされている場合はエラーになります。

XOPEN （クロス オープン・cross open）

ランダムアクセスファイルの処理を開始します。

書　式　xopen　＃〈ファイル番号〉,〈ファイル指定〉〔,〈レコード長〉〕

省略形　x.

説　明

- ランダムアクセス方式のデータファイルの操作を開始するためにファイル番号とファイル名を指定してオープンします。
- ＜ファイル番号＞は、1～127の任意の整数で指定します。ただし、すでに別のファイルオープンで使用中（close文またはkill文でファイル操作を終了していない）のファイル番号は、使用できません。
- ＜ファイル指定＞は次の書式の文字列型データで指定します。
  "〔＜装置名＞〕〔＜ディレクトリパス＞〕＜ファイル名＞"
  〈装置名〉　…FD1:～FD4:, MEM:
- ＜ディレクトリパス＞は階層ディレクトリのパス（道すじ）を指定します。（chdirコマンド参照）
- ＜装置名＞と＜ディレクトリパス＞の指定を省略すると、chdirコマンドで設定されているカレントディレクトリ内でのファイル指定になります。
- この命令を実行した後は、input ＃( )文, print ＃( )文で1レコード単位のデータアクセス、field文による宣言とget ＃文, put ＃文によってレコードフォマットを指定したデータアクセスをすることができます。
- 指定されたファイルが存在しない場合は新たにファイルを作成します。
- プロテクトされているファイル、またはディスク自体に書き込み禁止の処理がされている場合はinput ＃( )文, get ＃文による読み出しはできますが、print ＃( )文, put ＃文による書き込みはできません。
- ＜レコード長＞は、1レコードのバイト数を1～256までの値で指定します。ただし、ファイルへの記録は32×2^n（32, 64, 128, 256）バイト単位で切り上げます。したがって、この単位で使用するほうが記憶効率が良くなります。
- ＜レコード長＞を省略した場合は、32バイトが1レコードになります。

CLOSE （クローズ）

ファイル操作を修了します。

書　式　close〔＃〈ファイル番号〉〔,＃〈ファイル番号〉〕…〕

省略形　clo.

説　明
- ropen, wopen, aopen, xopenの各文でオープンされたファイルの終了処理をします。
- ＜ファイル番号＞は終了するファイルの指定で、複数の指定ができ、指定を省略した場合は、オープンされているすべてのファイルをクローズします。
- wopen, aopen, xopenでファイルオープンを行い、書き込みを行った場合はこの命令で終了しないと正常に登録されません。

例　`1000 close` …… 全てのオープン中のファイル操作を終了します。

KILL （キル）

ファイル処理を中止します。

書　式　kill〔＃〈ファイル番号〉〔,＃〈ファイル番号〉〕…〕

省略形　ki.

説　明
- オープンされているファイルの操作を中止します。
- wopen文により書き込み途中のファイルは登録されません。
- aopen文により追加中の物の追加部分は記録されません。
- xopen文による操作でオープン時の最大レコード番号を越える部分に書き込まれたものは登録されませんが、オープン時の最大レコード番号以内の部分で書き込まれた内容は残ります。

例　`kill` …… 全てのオープン中のファイル操作を中止します。

PRINT ＃　(プリント シャープ)

オープンしたファイルへデータを書き込みます。

書　式

print ＃〈ファイル番号〉,〈データ〉〔,〈データ〉〕…　(1)

print ＃〈ファイル番号〉(〈レコード番号〉),〈データ〉〔,〈データ〉〕…　(2)

省略形

p. ＃

説　明

- 書式(1)はシーケンシャルアクセスファイル用で、**wopen**文または**aopen**文でファイルオープンを行なったあとに、オープン時の＜ファイル番号＞で指定してデータを書き込みます。
- 書式(2)はランダムアクセスファイル用で、**xopen**文でファイルオープンされたファイルに対して＜ファイル番号＞と＜レコード番号＞を指定してデータを書き込みます。
- 書き込むデータが数値型の場合は**print**文による表示と同じ形式で文字並びとして書き込まれます。したがって有効桁を超える場合は指数形式になります。
- 文字型データの場合はそのまま書き込まれますが、シーケンシャルアクセスファイルの場合、データ中に文字コードが32よりも小さいコントロールコードがあると、読み出す時に正常に読み出せなくなるので注意が必要です。
- ランダムアクセスファイルの書き込みにおいて、書き込むデータの長さがレコード長を超える部分は記録されず、またレコード長に満たない部分はスペースで埋め、レコード長に合わせてデータを書き込みます。(レコード長は**xopen**文により設定されます。)

PRINT # USING (プリント シャープ ユージング)

書式設定をしてファイルに書き込みます。

書 式

print #〈ファイル番号〉, using 〈書式〉;〈データ〉[{ , / ; } 〈データ〉]…

省略形

p. #us. ? #us.

説 明

- BSD形式のシーケンシャルアクセスファイルに対する書き込みにおいて書式設定をした書き込みを行ないます。
- 指定する書式はprint using文と同じで、画面に表示されるものと同じように書き込みが行なわれます。
- 書き込むデータに対する注意はprint#文と同じです。

参 考☞

print using …… 画面に書式指定をしてデータを表示する。

INPUT ＃　(インプット シャープ)

オープンされたファイルよりデータを読み込みます。

書　式

input ＃〈ファイル番号〉,〈変数〉〔,〈変数〉〕…　(1)

input ＃〈ファイル番号〉(〈レコード番号〉),〈変数〉〔,〈変数〉〕…　(2)

省略形

i. ＃

説　明

- 書式(1)はシーケンシャルアクセスファイル用で、**ropen**文によりオープンされたファイルからデータを変数に読み出します。
- 書式(2)はランダムアクセスファイル用で、指定されたレコード番号からデータを変数に読み出します。複数の変数がある場合は左から順番にレコード番号を更新しながら変数に読み込みます。
- 数値変数への読み込みは数値として判断できないデータの場合エラーとなります。
- ランダムアクセスファイルにおいて文字変数に読み込む場合の文字長はレコード長となります。通常はデータの書き込み時にレコード長に満たない部分はスペースが入っていますのでデータの比較を行なうときは注意が必要です。

FIELD （フィールド）

ランダムアクセスファイルのレコードフォーマットを設定します。

書　式　field ＃〈ファイル番号〉,〈フィールド長〉 as 〈文字変数〉[,〈フィールド長〉 as 〈文字変数〉]…

省略形　fie.

説　明

- オープンされたランダムアクセスファイルの1レコード内をフィールドで分割するため、フィールドごとに＜フィールド長＞と入出力に使用する＜文字列型変数＞を指定します。＜ファイル番号＞はxopen文でファイルオープンされたときの番号になります。
- 1つのレコード内ではフィールド長の合計が、xopen文で設定されたレコード長をこえない範囲でいくつでも設定できます。
- field文によりランダムアクセスファイル用に割り付けられた変数を入力文（input文, read文, devi$文など）の入力変数として使用したり代入文の左辺に使用した場合は、フィールドの割り付けが無効となり、通常の文字変数として扱われます。field文で指定された変数にデータを設定するにはlset文やrset文で行ないます。

LSET （レフト セット・left set）
RSET （ライト セット・right set）

フィールドにデータをセットします。

書　式　lset〈フィールド変数〉＝〈文字列データ〉
rset〈フィールド変数〉＝〈文字列データ〉

省略形　ls. rs.

説　明

- field文でフィールドに割り付けられている変数にデータを設定する場合は、rset文またはlset文により行ないます。lset文は左づめ、rset文は右づめでデータをセットします。
- セットする＜文字列データ＞がフィールドで指定されている変数の長さより短い場合は、残りの部分にスペースがセットされます。また、データが長い場合は長い部分を切り捨てます。

PUT ＃　（プット シャープ）

ランダムアクセスファイルへデータを書き込みます。

書　式　put ＃〈ファイル番号〉〔,〈レコード番号〉〕

説　明

- ランダムアクセスファイルへのデータ書き込みはfield文で指定している変数にデータをセットしたのちに、put＃文により＜レコード番号＞を指定して書き込みます。
- ＜レコード番号＞は1以上の整数で指定し、省略するとそれ以前にput＃またはget＃文によりアクセスされた次のレコードになります。
- field文で指定された変数にデータを設定するにはlset文やrset文で行ないます。

GET ＃　（ゲット シャープ）

ランダムアクセスファイルからデータを読み込みます。

書　式　get ＃〈ファイル番号〉〔,〈レコード番号〉〕

説　明

- ランダムアクセスファイルからの読み込みは、get＃文でファイルオープンしたときの＜ファイル番号＞と＜レコード番号＞を指定することにより、field文で指定されている変数にデータがセットされます。
- ＜レコード番号＞は1以上の整数で指定し、省略するとそれ以前にput＃またはget＃文によりアクセスされた次のレコードになります。

サンプル

```
20 xopen #1,"random data",32
30 field #1,2 as A$,2 as B$,5 as C$
40 for A=1 to 10
50   lset A$=mki$(A)
60   lset B$=mki$(A*A)
70   lset C$=mks$(sqr(A))
80   put #1,A
90 next
100 close #1
```

```
20 xopen #1,"random data",32
30 field #1,2 as A$,2 as B$,5 as C$
40 for A=1 to 10
50   get #1,A
60   X=cvi(A$)
70   Y=cvi(B$)
80   Z=cvs(C$)
90   print X,Y,Z
100 next
110 close #1
```

MKI$ （メイク インテジャー ダラー・make integer$）
MKS$ （メイク シングル ダラー・make single$）
MKD$ （メイク ダブル ダラー・make double$）

数値データを内部表現に対応した文字列に変換します。

書 式

mki$(〈整数型データ〉)
mks$(〈単精度型データ〉)
mkd$(〈倍精度型データ〉)

説 明

- 数値データを内部表現に対応した文字列データに変換して返します。結果の文字長は整数型が2、単精度型が5、倍精度型が8文字になります。
- この関数は、おもにランダムアクセスファイルへのデータ書き込みのときに使用します。

CVI （コンバート ツー インテジャー・convert to integer）
CVS （コンバート ツー シングル・convert to single）
CVD （コンバート ツー ダブル・convert to double）

内部表現に対応した文字列データを数値データに変換します。

書 式

cvi(〈2文字の文字列式〉)
cvs(〈5文字の文字列式〉)
cvd(〈8文字の文字列式〉)

説 明

- 内部表現に対応した文字列データを数値データに変換して返します。
- それぞれの＜文字列式＞の長さは内部表現に合っている必要があります。
- この関数は、おもにランダムアクセスファイルより読み出したデータの変換に使用します。

DEVI$ (デバイス インプットダラー・device input$)

指定装置により1レコードのデータを読み出します。

書　式　devi$〈装置名〉,〈レコード番号〉,〈文字列型変数〉,〈文字列型変数〉

〈装置名〉…"FD1:"～"FD4:","MEM:"

省略形　dev.

説　明

- ランダムアクセスが可能な装置からレコード単位でデータを読み込みます。1レコードは256バイト単位で、0から始まる<レコード番号>で読み出し位置を指定します。読み出したデータは2つの文字列型変数に128バイトずつ代入されます。

DEVO$ (デバイス アウトダラー・device out$)

指定装置へ1レコードのデータを書き込みます。

書　式　devo$〈装置名〉,〈レコード番号〉,〈文字データ〉,〈文字データ〉

〈装置名〉…"FD1:"～"FD4:","MEM:"

説　明

- ランダムアクセスが可能な装置にレコード単位でデータを書き込みます。1レコードは256バイトで、0から始まる<レコード番号>で書き込み位置を指定します。書き込むデータは128バイトの文字型データ2つで指定します。
- この命令はファイル名で指定するようなBASICで管理された書き込みではなく、直接装置内の記録をアクセスします。不用意に使用するとファイルの内容が変わってしまったり、ファイル管理をしている部分を破壊するおそれがありますので注意が必要です。(特にファイル管理部分が破壊された場合、そのときは見かけ上正常でもあとで記録された容量が増えてきたときなどに異常が発生することがあります。)

INPUT$ （インプットダラー）

文字長を指定してファイルまたはキーボードから入力を行ないます。

書　式　input$(〈文字長〉[,[#]〈ファイル番号〉])

説　明

- ropen文によりオープンされたファイルまたはキーボードより、指定した文字長を読み取る関数です。
- 〈ファイル番号〉を省略して文字長のみを指定した場合は、キーボードからの入力になります。このとき、画面にカーソルが出ますが入力文字は表示されません。

例

```
print input$(5)
```
…… キーボードより5文字のデータを読み取り表示します。

EOF （エンド オブ ファイル・end of file）

ファイルエンドであるかどうかを調べます。

書　式　eof([#]〈ファイル番号〉)

説　明

- <ファイル番号>で指定されたファイルが最後までデータを読み取ったかどうかを調べる関数です。
- 結果は数値でファイルエンドであれば-1、ファイルエンドでなければ0を返します。
- 装置名が"COMn:"(RS-232インターフェイス)であれば、init "COMn:"文で設定されたエンドコードを読み出したときに、-1となります。
- シーケンシャルファイルの場合は、最後のデータの次のデータを読み取った後にこの関数の値が-1になります。
- ランダムアクセスファイルの場合は、記録されている最大レコード数よりも大きなレコード番号で読み取りを行なった場合にこの関数の値が-1になります。

例

```
if eof(#1) then close #1
```
…… もしファイル番号1がファイルエンドであればそのファイルを終了します。

LOC （エル オー シー・location counter of a file）

ファイル中での論理的な現在位置を求めます。

書 式　loc(〔#〕〈ファイル番号〉)

説 明

- オープンされているファイルに対して現在どの位置をアクセスしているかを求める関数です。ファイルの先頭からの位置を返します。
- シーケンシャルアクセスファイルの場合はそのファイルがオープンされてから読み書きされたブロック数、ランダムアクセスファイルの場合は最後に読み書きされたレコード番号を返します。
- シーケンシャルアクセスファイルの場合、1ブロックに句切記号や改行コードも含めて"FDn:"や"MEM:"の場合254バイト、"CMT:"の場合は256バイトが記録されます。
- 装置名が"COMn:"(RS-232Cインターフェイス)の場合は入力バッファ中に残っている文字数をバイト数で返します。ただし、init "COMn:"文でKI・KO(漢字イン・アウト)の設定がされている場合は、KI・KOコードも含まれていますので読み出したときの文字長が短くなる場合があります。
- 装置名が"KB:"(キーボード)の場合はキーバッファに残っている文字数をバイト数で返します。

FPOS （ファイル ポジション・file position）

ファイル中の物理的な現在位置を求めます。

書 式　fpos(〔#〕〈ファイル番号〉)

説 明

- <ファイル番号>でオープンされているファイルがこの関数の実行直前に読み出しまたは書き込みを行なった位置を装置の物理位置で求める関数です。
- 物理位置とは、その装置の最初から256バイト単位で句切られた0から始まるレコード番号になります。
- この関数は"FD1:"～"FD4:","MEM:"の装置に登録されているファイルに対して有効になります。それ以外の装置に対してはエラーになります。

LOF （エル オー エフ・ length of a file

ファイルの大きさを求めます。

書 式 lof(〔#〕〈ファイル番号〉)

説 明

- オープンされているファイルの大きさを求める関数です。
- 結果はシーケンシャルアクセスファイルの場合は使用セクタ数、ランダムアクセスファイルの場合は使用している最大のレコード番号になります。
- シーケンシャルアクセスファイルの場合、1セクタに区切記号や改行コードも含めて254バイト記録されます。
- 装置名が"FDn:"または"MEM:"のシーケンシャルアクセスファイルの場合、1セクタに句切記号や改行コードも含めて254バイトが記録されます。
- 装置名が"COMn:"の場合はバッファの残量をバイト数で返します。ただし、init "COMn:"文でKI･KO(漢字イン･アウト)の設定がされている場合は、KI･KOコードも含まれていますので読み出したときの文字長が短くなる場合があります。
- 装置名が"KB:"の場合は、キーバッファに残っている文字数をバイト数で返します。

ATTR$ （アトリビュート ダラー・attribute$)

ファイルのプロテクト状態を調べます。

書 式 attr$(〈ファイル指定〉)

省略形 att.

説 明

- 指定されたファイルのプロテクト状態を調べる関数で、結果は文字型データとなります。
- ファイルがプロテクトされている場合は結果が"P"、プロテクトされていない場合は結果が" "(1個のスペース)、指定ファイルが見つからない場合は結果が""(Null)になります。
- ファイルのプロテクト状態を変更するにはlock文, unlock文を使用します。
- ＜ファイル指定＞は次の書式の文字列型データで指定します。

  "〔＜装置名＞〕〔＜ディレクトリパス＞〕＜ファイル名＞"

  ＜装置名＞……FD1:～FD4:, MEM:

DEVF （デバイス フリー・device free）

ファイルデバイスの残量を求めます。

書 式　devf(〈装置名〉)

〈装置名〉…"FD1:"～"FD4:", "MEM:"

説 明
- 外部記憶装置の未使用の記憶容量をキロバイト単位で求める関数です。
- 1キロバイトに満たない容量は切り捨てます。

例

```
print devf("fd1:")
```

…… ドライブ1にセットしているフロッピーディスクの残量を表示します。

PWD$ （パスワード ダラー・pathword$）

カレントディレクトリの階層を求めます。

書 式　pwd$

省略形　pw.

説 明
- システム変数としてカレントディレクトリのディレクトリパスを文字列型データで返します。
- 階層が10段以上になるとデータを返すことができずにエラーになります。

例　`print pwd$` …… カレントディレクトリのディレクトリパスを表示します。

参 考　chdir …… カレントディレクトリの設定。

KMODE （漢字モード・kanji mode）

漢字モードにします。

書　式

kmode { [0] / 1 }

省略形

km.

説　明

- 漢字などの全角文字を扱うためのモード切り換えを行ないます。指定を1とすると漢字モードとなり、指定を省略または0とすると漢字モードを解除します。BASICの起動時はkmode 1になっています。
- 漢字モードのときは文字コードが16進数で81～9F、またはE0～FCの範囲であればそれに続く1バイトと合わせて、2バイトの全角文字として扱われ表示などを行ないます。このモード指定により実行内容が変化する文字表示以外の命令は次のとおりです。

  コマンド　……　search
  グラフィック　……　symbol
  関数　……　instr asc

  文字画面の最下行を漢字入力に使用するためwidth文、init "CRT1:" 文やconsole文によるスクロール範囲の初期化時に、最下行がスクロール範囲に含まれなくなります。
- 漢字モードのときは文字画面の最下行を使って漢字入力ができますが、次の条件に合わない場合は、漢字入力のキー操作を行なっても「ピッ」と音が鳴って入力状態になりません。

  漢字入力の条件
  - kmode 1により漢字モードに設定。
  - ラインスクロールモード。（init "CRT1:"文で設定します。）
  - 最下行が文字スクロール範囲外。（最下行がconsole文による指定範囲に含まれない。）

例

```
kmode 1　……　漢字モードにします。
```

AKCNV$ (アスキー 漢字 コンバート ダラー・ascii kanji covert$)

半角文字を全角文字に変換します。

書　式　akcnv$(〈変換文字列〉)

省略形　ak.

説　明

- ＜変換文字列＞中にある半角文字を全角文字に変換した文字列を求める関数です。ただし、変換が行なわれるのは数字、アルファベット、カナ文字や一部の記号などの全角と半角の相方にある文字のみです。全角文字や変換できない半角文字はそのままになります。

例

```
print akcnv$("ﾌﾟﾘﾝﾀ")
```

…… 全角文字で"プリンタ"の4文字を表示します。

KACNV$ (漢字 アスキー コンバート ダラー・kanji ascii convert$)

全角文字を半角文字に変換します。

書　式　kacnv$(〈文字列〉)

省略形　ka.

説　明

- ＜文字列＞中の全角文字を半角文字に変換した文字列を求める関数です。ただし、変換が行なわれるのは数字、アルファベット、カナ文字や一部の記号などの全角と半角の相方にある文字のみです。半角文字や変換できない全角文字はそのままになります。

例

```
print kacnv$("ＡＢＣあいうえお")
```

…… 半角文字で"ABCアイウエオ"と表示します。

```
print kacnv$("プリンタ")
```

…… 半角文字で"ﾌﾟ リンタ"の5文字を表示します。

JIS$ （ジス ダラー）

シフトJISコードよりJISコードに変換します。

書 式 jis$(〈変換文字列〉)

省略形 j.

説 明

- ＜変換文字列＞の最初の2バイトをシフトJISコードとみなして、4バイトのJIS16進文字列に変換する関数です。
- 16進JISコードに変換できないものはエラーになります。
- JIS16進の文字列は**hexchr$**関数を使用すると2バイトコードに変換できます。
- JIS16進よりシフトJISに変換するには**&J**を付けた定数が利用できます。

例 `print jis$("漢字")` …… "3441"と表示します。

KTN$ （区点 ダラー・kuten$）

シフトJISコードよりJIS区点コードに変換します。

書 式 ktn$(〈変換文字列〉)

省略形 kt.

説 明

- ＜変換文字列＞の最初の2バイトをシフトJISコードとみなして、4バイトのJIS区点コード文字列に変換する関数です。
- JIS区点コードに変換できないものはエラーになります。
- JIS区点コードよりシフトJISに変換するには**&K**を付けた定数が利用できます。

例 `print ktn$("漢字")` …… "2033"と表示します。

KLEN （漢字レングス・kanji length）

全角文字を含む文字列の文字数を求めます。

書　式　klen(〈変換文字列〉)

省略形　kle.

説　明
- 全角文字および半角文字が混在した＜文字列＞の文字数を求める関数です。したがって、全角文字は2バイトで1文字とみなされます。
- この関数はkmode文による設定に関係なく動作します。

例
```
print klen("漢字ｻｰﾏﾙﾌﾟﾘﾝﾀ")
```
…… 11と文字数を表示します。

KPOS （漢字ポジション・kanji position）

文字列の先頭より指定文字までのバイト数を求めます。

書　式　kpos(〈文字列〉,〈位置〉)

省略形　kp.

説　明
- ＜文字列＞の中の＜位置＞番目の文字が、先頭から何バイト目になるかを求める関数です。半角文字は1バイトで1文字、全角文字は2バイトで1文字とみなされます。
- ＜位置＞が＜文字列＞の文字数より大きい場合は結果が0になります。

例
```
print kpos("11月3日は文化の日",7)
```
…… 7文字目の"文"は10バイト目になります。

ON ERROR GOTO （オン エラー ゴーツー）

エラー処理の宣言を行ないます。

書　式　on error goto〈エラー処理開始行〉

省略形　o. err. g.

説　明

- プログラム実行中にエラーが発生した場合エラーに対応する処理を行なう行を<エラー処理開始行>として宣言します。
- この命令で宣言を行なっておくと、エラーが発生した場合コマンド待ちに戻らずに、システム変数のernとerlにエラー番号とエラー発生行の情報を設定して、<エラー処理開始行>に実行を移します。
- エラー処理内では、ernとerlの値により処理を行ない、resume文によりエラー処理を終了し、もとの処理に戻ります。
- エラーが発生して定義された処理行へ実行を移したあと、resume文を実行するまでにエラーが発生すると、エラー処理は行なわれずにエラーメッセージを出して命令待ちに戻ります。
- プログラム中に複数の宣言がある場合は、エラーの発生する直前に実行されている宣言の<エラー処理開始行>が有効となります。
- <エラー処理開始行>は、行番号またはラベルで指定できます。
- <エラー処理開始行>を0とするエラー処理の宣言を解除します。

例

```
on error goto *ERROR
```

…… エラーが発生した場合ラベル名＊ERRORの行からエラー処理を開始するように宣言します。

参　考☞　resume …… エラー処理の終了。
ern …… エラー番号を求めるシステム変数。
erl …… エラー発生行を求めるシステム変数。

ERN （エラー ナンバー・error number）

エラー番号を求めます。

書 式　ern

説 明
- プログラムの実行中にエラーが発生した場合に、エラーコードが設定されるシステム変数です。
- 直接実行モードでエラーが発生した場合は、ernに値を設定しません。
- エラーコードについては付録の「エラーメッセージ表」を参照してください。

例
```
if ern=1 then print "文法誤りです"
```
…… もしエラーコードが１であれば "文法誤りです"と表示します。

参 考　on error go to …… エラー処理の宣言。

ERL （エラー ライン・error line）

エラー発生行を求めます。

書 式　erl

説 明
- プログラムの実行中にエラーが発生した場合に、エラーが発生した行番号が設定されるシステム変数です。
- 直接実行モードでエラーが発生した場合は、erlに値を設定しません。

例
```
if erl=100 then resume next
```
…… もしエラーの発生した行が行番号100であれば、エラーの発生した次の文に戻ります。

参 考　on error goto …… エラー処理の宣言。

RESUME （リジューム）

エラー処理を終了してプログラムを再開します。

書 式

resume 〔 { next / 〈戻り行〉 } 〕

省略形

resu.

説 明

- on error goto文により、エラー処理に移り、エラー処理を終了して元のプログラムに戻る場合に使用します。この命令により、エラー時の処理行の再設定を行ないますので、この命令以外で元のプログラムへ移った場合は、次にエラーが発生してもエラー処理を行ないません。
- 戻る場所は次のように指定できます。

  resume …… エラーの発生した文に戻ります。

  resume next …… エラーの発生した次の文に戻ります。

  resume 0 …… プログラムの先頭に戻ります。

  resume ＜戻り行＞ …… 指定された行に戻ります。
- エラー処理中のエラーは、エラー処理は行なわれませんが、resume文でエラーが発生(戻り行などの指定の誤りがある場合)した場合は、エラー処理が行なわれますので注意が必要です。

例

```
if erl=1010 then resume *START
```

…… もしエラーの発生した行が行番号1010であれば＊STARTのラベル名の付いた行から実行を再開します。

参 考

on error go to …… エラー処理行の宣言。

ern …… エラー番号を求める。

erl …… エラー発生行を求める。

ERROR （エラー）

エラーを発生させます。

書　式　error 〈エラー番号〉

省略形　err.

説　明

- エラーを疑似的に発生させ、エラー処理のシミュレーションなどに利用します。
  ＜エラー番号＞は0～255までの整数値で指定し、実行時のこの値をernに設定します。エラー処理が宣言されていない場合はエラー番号に対応したエラーメッセージを出して実行を中止します。エラーが定義されていない番号の場合、エラーメッセージは unprintable error になります。
- 定義されていないエラー処理を利用してユーザー独自のエラーを定義することもできます。
- ＜エラー番号＞に範囲外のデータを指定すると data error が発生します。

例

```
100 error 3
```
…… 行番号100でエラーコード3のエラーを発生します。この場合、ernに3、erlに100が設定されます。

```
250 if X<0 then error 103
```
…… Xの値が負であれば103番のエラーが発生します。エラー処理の宣言が行なわれているとernに103、erlに250をそれぞれ設定してエラー処理の開始行に移ります。エラー処理の中で、このエラーコードに対応する処理を行なえばユーザー独自のエラー定義になります。

参　考　on error goto …… エラー処理の宣言。
ern …… エラー番号を求める。
erl …… エラー発生行を求める。

ON KEY GOSUB (オン キー ゴーツーサブルーチン・on key go to subroutine)

ファンクションキーによる割り込み処理行を定義します。

書　式　on key gosub 〔〈処理行1〉〕〔,〈処理行2〉〕…

省略形　o. k. gos.

説　明

- F1～F20のファンクションキー(F11～F20は SHIFT キーを押しながら F1 ～ F10 のキーを押します。)を押したときに割り込み処理が行なわれる開始行を処理行としてそれぞれのキーに対して定義します。
- F1を押したときに処理する行を＜処理行1＞、F2を押したときに処理する行を＜処理行2＞というように、順次ファンクションキーの番号に合わせて割り込み処理の開始行を指定して行きます。
- この命令では割り込み処理行を定義をするだけで、割り込み処理の実行制御はkey on/off/stop文により行ないます。
- ＜処理行＞は、行番号またはラベルで指定できます。
- 割り込み処理は、割り込み発生時に＜処理行＞からサブルーチンとして実行しますので、割り込み処理の終了はreturn文になります。
- この命令での定義は、runコマンドおよびend文の実行により解除されます。
- この命令の実行時に＜処理行＞で指定された行がない場合はエラーになります。

例

```
on key gosub 100,200,,400
```

…… 割り込み処理行をF1に100, F2に200, F4に400と行番号で定義します。

参　考☞　key on/off/stop …… ファンクションキー割り込みの制御。

ON STOP GOSUB (オン ストップ ゴーツーサブルーチン・on stop go to subroutine)

SHIFT ＋ BREAK キーによる割り込み処理行を定義します。

書　式　on stop gosub〈処理行〉

省略形　o. s. gos.

説　明

- SHIFT ＋ BREAK キー押したときに、割り込み処理が行なわれる開始行を＜処理行＞として定義します。
- この命令では割り込み処理行を定義するだけで、割り込み処理の実行を制御するのは、stop on/off/stop文により行ないます。
- ＜処理行＞は、行番号またはラベルで指定できます。
- 割り込み処理は、割り込み発生時に＜処理行＞からサブルーチンとして実行されますので、割り込み処理の終了はreturn文になります。
- この命令での定義は、runコマンドおよびend文の実行により解除されます。
- この命令の実行時に＜処理行＞で指定された行がない場合はエラーになります。
- この割り込み機能がonされると本来の SHIFT ＋ BREAK キーの操作による実行の中断機能はなくなります。この機能はプログラム実行中に不用意にプログラムの実行を中断されないようにするものです。したがって、実行中に何らかの方法で実行を終了するようにプログラミングしておいてください。

例

```
on stop gosub 1000
```
…… 割り込み処理行を行番号1000と定義します。

```
on stop gosub *BREAK
```
…… 割り込み処理行を＊BREAKというラベルで定義します。

参　考　stop on/off/stop …… SHIFT ＋ BREAK キーによる割り込みの制御。

ON HELP GOSUB (オン ヘルプ ゴーツーサブルーチン・on help go to subroutine)

HELP キーによる割り込み処理行を定義します。

書　式　on help gosub〈処理行〉

省略形　o. hel. gos.

説　明

- HELP キーを押したときに、割り込み処理が行なわれる開始行を＜処理行＞として定義します。
- この命令では割り込み処理行を定義するだけで、割り込み処理の実行を制御するのは、**help on/off/stop**文により行ないます。
- ＜処理行＞は、行番号またはラベルで指定できます。
- 割り込み処理は、割り込み発生時に＜処理行＞からサブルーチンとして実行されますので、割り込み処理の終了は**return**文になります。
- この命令での定義は**run**コマンドおよび**end**文の実行により解除されます。
- この命令の実行時に＜処理行＞で指定された行がない場合はエラーになります。

例

```
on help gosub *HELP
```

…… 割り込み処理行を**＊HELP**というラベルで定義します。

参　考☞　help on/off/stop …… HELP キーによる割り込み制御。

KEY ON/OFF/STOP （キー オン／オフ／ストップ）
STOP ON/OFF/STOP （ストップ オン／オフ／ストップ）
HELP ON/OFF/STOP （ヘルプ オン／オフ／ストップ）

キー割り込み機能を制御します。

書 式

key〔(〈キー番号〉)〕〈制御〉 … ファンクションキーによる割り込み
stop〈制御〉 … SHIFT ＋ BREAK キーによる割り込み
help〈制御〉 … HELP キーによる割り込み
〈制御〉 … on, off, stop

省略形

k. s. hel. o./of./s.

説 明

- 各種のキーによる割り込み処理を制します。ファンクションキーの場合、＜キー番号＞を指定することによりF1～F20のキーごとの指定となりますが、＜キー番号＞の指定を省略するとすべてのファンクションキーに対する指定になります。
- onを指定するとキーによる割り込みが可能となり、それぞれの定義されている割り込み処理の開始行から、サブルーチンとして実行します。
- offを指定すると割り込み機能は解除され、該当するキーを押しても割り込み処理は行なわれません。
- stopを指定すると該当するキーを押しても割り込み処理は行なわれませんが、割り込みがあったことは記憶されていますので、あとでonが指定されたときに、割り込み処理を実行します。ただし、stop中にキーを何回押しても、実行されるのはそれぞれのキーに対して1回のみです。
- onまたはstopの状態でinput文やline input文による入力時に、該当するキーを押すとinput文をぬけ(onの場合は割り込み処理を行ない)、次の処理に進みます。このときのinput文の入力変数の値は変化しません。
- runコマンドおよびend文の実行やエラーの発生、または実行中断時のプログラム変更などにより、すべての割り込み処理の実行制御はoffになります。
- SHIFT ＋ BREAK キーの操作または、stop文の実行によりプログラムが中断され、contコマンドの実行が可能な場合は、プログラム中断時にonとなっていたものはstopと同じ扱いになり、プログラムの再開で、再びonになります。
- onまたはstopに指定されているキーは、本来の操作には使用できません。

ON COM GOSUB (オン コミュニケーション ゴーツーサブルーチン・on communication go to subroutine)

RS−232Cインターフェイスによる割り込み処理行を定義します。

書　式　on com gosub〈処理行〉

省略形　o. com gos.

説　明

- RS−232Cインターフェイスにデータが入力され割り込みが発生したときに、割り込み処理が行なわれる開始行を＜処理行＞として定義します。
- この命令では割り込み処理行を定義するだけで、割り込み処理の実行を制御するのはcom on/off/stop文で行います。
- ＜処理行＞は、行番号またはラベルで指定できます。
- 割り込み処理は、割り込み発生時に＜処理行＞からサブルーチンとして実行されますので、割り込み処理の終了はreturn文になります。
- この命令での定義はrunコマンドおよびend文の実行により解除されます。
- この命令の実行時に＜処理行＞で指定された行がない場合はエラーになります。

例

```
on com gosub *COMIN
```

…… RS-232Cインターフェイスによる割り込み処理行を＊COMINというラベルで定義します。

ON MOUSE GOSUB (オン マウス ゴーツーサブルーチン・on mouse go to subroutine)

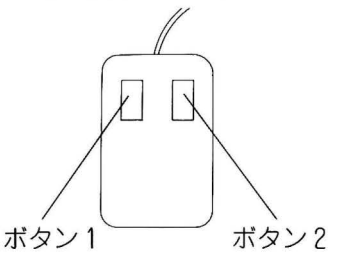
マウスによる割り込み処理を定義します。

書　式　on mouse gosub〔〈処理行1〉〕〔,〈処理行2〉〕……〔,〈処理行5〉〕

省略形　o. mou. gos.

説　明

- この命令は別売のマウス(MZ-1X10)操作による各種割り込み要因に対する割り込み処理を定義します。
  - 1 …… マウスが移動したとき。
  - 2 …… ボタン1が押されたとき。
  - 3 …… ボタン2が押されたとき。
  - 4 …… ボタン1が離されたとき。
  - 5 …… ボタン2が離されたとき。



- <処理行>の0～4は、割り込み要因の0～4のそれぞれに対応した割り込み処理の先頭行を行番号またはラベルで指定します。
- この命令では割り込み処理の行なわれる開始行を定義するだけで、割り込み処理の制御はmouse on/off/stop文で行なわれます。
- 割り込み処理は、割り込み発生時に<処理行>からサブルーチンとして実行されますので、割り込み処理の終了はreturn文になります。
- この命令での定義はrunコマンドおよびend文の実行により解除されます。
- この命令実行時に<処理行>で指定された行がない場合はエラーになります。

例

```
on mouse gosub ,*TRG_1
```

…… マウスのボタン1が押されたときに実行する割り込み処理行を
＊TRG_1というラベルで定義します。

参　考　mouse on/off/stop …… マウスによる割り込みの制御。

ON TI$ GOSUB (オン タイム ダラー ゴーツーサブルーチン・on time$ go to subroutine)

リアルタイムクロックでの割り込み処理を定義します。

書　式　on ti$="〈時〉〈分〉〈秒〉" gosub 〈処理行〉

省略形　o. ti$ gos.

説　明

- リアルタイムクロックが設定時刻になったときに割り込みを発生させ、割り込み時間と割り込み処理の開始行を定義します。
- この命令では割り込み処理の行なわれる開始行を定義するだけで、割り込み処理の制御はti$ on/off/stop文で行なわれます。
- ＜処理行＞は、行番号またはラベルで指定できます。
- 割り込み処理は、割り込み発生時に＜処理行＞からサブルーチンとして実行されますので、割り込み処理の終了はreturn文になります。
- この命令での定義はrunコマンドおよびend文の実行により解除されます。
- この命令の実行時に＜処理行＞で指定された行がない場合はエラーになります。
- 割り込みは＜分＞単位で処理され、＜秒＞の指定は"00"とみなします。ただし、＜秒＞の指定を省略することはできません。
- この命令の実行時には、システム変数のti$と割り込み時刻との差を計算しますので、ti$の設定はこの命令の実行前に行なってください。

例

```
on ti$="124500" gosub 1000
```

…… 12時45分になると行番号1000から処理をするように定義します。

参　考☞　ti$ on/off/stop …… リアルタイムクロックの割り込み制御。

INTERVAL （インターバル）

インターバルタイマの設定をします。

書　式　interval〈周期〉

省略形　inte.

説　明

- 一定間隔で割り込みを発生させるインターバルタイマの周期を設定します。
- ＜周期＞は0.1秒単位で1～65535までの整数で指定します。
- この命令の実行によりインターバルタイマが初期化されスタートします。したがって、＜周期＞を50とした場合、この命令の実行を起点として5秒ごとの割り込みが発生します。
- 割り込み処理をするには、on interval gosub文で割り込み処理の定義、interval on/off/stop文により割り込み処理の制御を行ないます。

ON INTERVAL GOSUB (オン インターバル ゴーツーサブルーチン・on interval go to subroutine)

インターバルタイマーによる割り込み処理を定義します。

書　式　on interval gosub 〈処理行〉

省略形　o. inte. gos.

説　明

- interval文により設定された一定間隔で発生する割り込みに対する割り込み処理の開始行を＜処理行＞として定義します。
- この文では割り込み処理の行なわれる開始行を定義するだけで、割り込み処理の制御はinterval on/off/stop文で行なわれます。
- ＜処理行＞は、行番号またはラベルで指定できます。
- 割り込み処理は、割り込み発生時に＜処理行＞からサブルーチンとして実行されますので、割り込み処理の終了はreturn文になります。
- この命令での定義はrunコマンドおよびend文の実行により解除されます。
- この命令の実行時に＜処理行＞として指定された行がない場合はエラーになります。

ON MUSIC GOSUB (オン ミュージック ゴーツーサブルーチン・on music go to subroutine)

音楽演奏終了時に発生する割り込み処理を定義します。

書　式　on music gosub〈処理行〉

省略形　o. mu. gos.

説　明

- music文による音楽演奏で、演奏が終了したときに発生する割り込みに対する処理の開始行を＜処理行＞として定義します。
- この命令では割り込み処理の行なわれる開始行を定義するだけで、割り込み処理の制御はmusic on/off/stop文で行なわれます。
- ＜処理行＞は、行番号またはラベルで指定できます。
- 割り込み処理は割り込み発生時に＜処理行＞からサブルーチンとして実行されますので、割り込み処理の終了はreturn文になります。
- この命令での定義はrunコマンドおよびend文の実行により解除されます。
- この命令の実行時に＜処理行＞として指定された行がない場合は、エラーになります。
- この命令はmusic on文を実行するまでに実行してください。音楽演奏をしていない状態ではすぐに割り込みがかかり正常に処理することができなくなります。

例

```
on music gosub 1000
```

…… 音楽演奏終了時の割り込み処理行を行番号1000として定義します。

参　考　music on/off/stop …… 音楽演奏終了時の割り込みの制御。

COM ON/OFF/STOP （コミュニケーション オン/オフ/ストップ・communication on/off/stop）
MOUSE ON/OFF/STOP （マウス オン/オフ/ストップ）
TI$ ON/OFF/STOP （タイムダラー オン/オフ/ストップ・time$ on/off/stop）
INTERVAL ON/OFF/STOP （インターバル オン/オフ/ストップ）
MUSIC ON/OFF/STOP （ミュージック オン/オフ/ストップ）

各種の割り込みの制御を行います。

書　式

com〈制御〉　　… RS232Cインターフェイスによる割り込み
mouse〔(〈要因番号〉)〕〈制御〉　… マウスによる割り込み
ti$〈制御〉　　… クロックによる割り込み
interval〈制御〉　　… インターバルタイマによる割り込み
music〈制御〉　　… 音楽演奏による割り込み
〈制御〉… on, off, stop

説　明

- 各種の割り込み処理を制御します。
- onを指定すると割り込みが可能となり、割り込み発生時にそれぞれの定義されている割り込み処理の開始行から、サブルーチンとして実行します。
- offを指定すると割り込み機能は解除され、割り込みが発生しても割り込み処理は行なわれません。
- stopを指定すると割り込み発生時には割り込み処理は行なわれませんが、割り込みがあったことは記憶されていますので、あとでonが指定されたときに割り込み処理を実行します。
- 割り込み処理はステートメントとステートメントの間に行なわれ、ステートメント実行中はstopの状態になります。したがってpaint文やpoly文などの処理時間がかかる処理中に割り込みがかかった場合は、その処理が終了してから割り込み処理が行なわれます。また、input文やline input文は入力終了後に割り込み処理を実行します。
- SHIFT ＋ BREAK キーの操作または、stop文の実行によりプログラムが中断され、contコマンドの実行が可能な場合は、中断時にonとなっていたものはstopと同じ扱いになり、プログラムの再開で、再びonになります。
- runコマンドおよびend文の実行やエラーの発生、または実行中断時のプログラム変更などにより、すべての割り込み処理の実行制御はoffになります。

例

```
com on
```
…… RS－232Cインターフェイスによる割り込みを可能とします。

MOUSE （マウス）

マウスの機能を設定します。

書 式 mouse〈機能〉〔,〈指定〉〕…

省略形 mou.

説 明

- 別売のマウス(MZ-1X10)に対して各種の機能を設定します。＜機能＞は0～6まであり、次のようになります。なおマウス処理の座標はすべて絶対スクリーン座標になります。

mouse 0

各種機能を初期化し、マウスを使用可能にします。初期化される機能は次のとおりです。

- マウスカーソルを左上に移動させ、オフ(表示しない)状態にします。
- マウスカーソル形状を設定し
  指示点は左上端になります。

- 縦横の移動比率(次ページ参照)を8にします。
- 移動範囲を画面全体として、カーソル色は青になります。

マウスのコントロールはRS－232Cインターフェイスのチャンネルを使用していますので、この機能を実行するときに"COM2:"がオープンされているとエラーになりますので"COM2:"がオープンされている場合はあらかじめクローズしておいてください。

mouse 1, X, Y, ＜カーソルスイッチ＞

マウスカーソルの移動と、オン、オフの制御をします。

X, Yはマウスカーソルを移動させる座標になります。マウスカーソルの指示点が指定座標になるような移動です。

＜カーソルスイッチ＞を1(オン)にするとマウスカーソルを表示し、0(オフ)にするとマウスカーソルは表示しません。

mouse 2, ＜横指示点＞, ＜縦表示点＞, ＜カーソル形状＞

マウスカーソルの形状と指示点を設定します。

マウスカーソルは16×16ドットで表しますので、＜カーソル形状＞は32バイトの文字列型データで指定します。

カーソル形状の文字列

1	2	3	4	…

⇨

MSB	LSB	MSB	LSB
1		2	
3		4	
⋮		⋮	
31		32	

＜横指示点＞は左側から＜縦指示点＞は上からのドット数で0～15の整数値で指定します。

この指示点がマウス入力時の座標になります。グラフィックが縦200ドットのときは、カーソルは1ラインおきに表示され、縦の指示点は1/2の位置になります。

mouse 3,＜方向＞,＜移動比率＞

マウスの移動比率を設定します。＜方向＞は0が横方向で、1が縦方向の指定になり、＜移動比率＞は1～255の整数値で表します。＜移動比率＞は、画面上の8ドットを移動するのに必要なマウスからのパルス数になり、1が最も大きく移動し、255が最も小さくなります。また、＜移動比率＞が、8よりも小さい場合は指示できない座標が出てきます。なお、マウスからのパルスは、約0.25mm移動するごとに発生します。

mouse 4, X_1, Y_1, X_2, Y_2

マウスの移動範囲を設定します。左上の座標X_1, Y_1と右下の座標X2, Y2を対角とする範囲の外へ指示点を移動させることはできなくなります。

mouse 5,＜色＞

マウスの表示色を指定します。＜色＞は0～2で指定し、次の色になります。

＜色＞の指定	4/16色モード	256色モード
0（青）	1	&O004
1（赤）	2	&O040
2（緑）	4	&O400

4/16色モードのときはパレットコードとなり、256色モードのときは512色系のカラーコードになります。

マウスカーソルはシングルプレーンで表示され、描かれている図形と重なる部分をXOR処理します。

mouse 6

マウスの処理を終了します。RS－232CインターフェイスのチャンネルBのマウスでの使用を終了します。マウスを使用した場合、この機能で終了しておかないと"COM2:"を使用できません。

- グラフィック画面の描画や消去を行なうときは必ずマウスカーソルをオフ（表示しない）状態にしてください。また、init "CRT2:"文を実行する場合は、必ずmouse 6を実行してマウス処理を終了しておいてください。
- この命令は"auto-run.s25"プログラム内でnew on 2を行なって消去されています。この命令を使用される場合は、"auto-run.s25"プログラムを変更して再起動してください。また、メモリ標準実装（128キロバイト）の場合、フリーエリアにより使い方が限定される場合があります。詳しくはBASIC-M25マニュアルの付録の「BASICのフリーエリア」を参照してください。

MOUSE(n)　(マウス)

マウスからの情報を読み取ります。

書　式　mouse (〈機能〉[,〈ボタン番号〉])

省略形　mou.

説　明

- 別売のマウス(MZ-1X10)から座標やボタンの情報を読み取る関数です。＜機能＞は0～8まであり、＜機能＞に応じて＜ボタン番号＞として1か2のボタン番号を指定します。この関数で求められる座標はすべて絶対スクリーン座標になります。

  mouse(0)
  　マウスカーソルのX座標上の現在値を求めます。

  mouse(1)
  　マウスカーソルのY座標上の現在値を求めます。

  mouse(2, ＜ボタン番号＞)
  　＜ボタン番号＞で指定されたボタンの状態を求めます。ボタンが押されている場合は−1、押されていない場合は0の値が返ります。

  mouse(3, ＜ボタン番号＞)
  　＜ボタン番号＞で指定されたボタンが最後に押されたときのX座標を求めます。

  mouse(4, ＜ボタン番号＞)
  　＜ボタン番号＞で指定されたボタンが最後に押されたときのY座標を求めます。

  mouse(5, ＜ボタン番号＞)
  　＜ボタン番号＞で指定されたボタンが最後に離されたときのX座標を求めます。

  mouse(6, ＜ボタン番号＞)
  　＜ボタン番号＞で指定されたボタンが最後に離されたときのY座標を求めます。

  mouse(7)
  　最後にmouse(7)を実行してからのX座標の移動量を求めます。

  mouse(8)
  　最後にmouse(8)を実行してからのY座標の移動量を求めます。

- この関数はmouse 0でマウスを使用可としていないと正しい値は返りません。

INIT "KB:" (イニシャライズ キーボード・initialize keyboard)

キーボードの機能を設定します。

書　式　init "KB:〔〈カナ配列〉〕〔,〈文字選択〉〕〔,〈LOCK〉〕〔,〈漢字入力〉〕"

説　明

- ＜カナ配列＞は、キーボードのカナ配列をJISで定められた配列にするか、50音順の配列にするかを選択します。
  - 0 … JIS配列。
  - 1 … 50音順配列。
- ＜文字選択＞は、英数、カナ、グラフィック文字の入力状態を指定します。
  - 0 … 英数入力。
  - 1 … カナ入力。
  - 2 … グラフィック文字入力。
- ＜LOCK＞は、アルファベットの大文字、小文字の入力を指定します。
  - 0 … 小文字入力。
  - 1 … 大文字入力。
- ＜漢字入力＞は、漢字入力状態を指定します。
  - 0 … 漢字入力状態を抜けます。
  - 1 … 漢字入力状態にします。
- 漢字の入力はラインスクロール状態で、最下行がスクロール範囲外でないとできませんので注意してください。
- それぞれの指定を省略した場合は、現在の状態を設定します。
- BASICの起動時は、すべての指定が0になっています。
- キーボード自体のカナ配列はJIS配列になっていますので、50音配列で使用するときにはキートップに50音順のキーラベル(添付)を貼り付けて使用してください。

例　`init "KB:1"` …… キーボードのカナ配列を50音順に設定します。

CLICK ON/OFF（クリック オン/オフ）

キーのクリック音を制御します。

書　式

click { on / off }

省略形

cli.　o./of.

説　明

- キーボードのキーを押すごとに出るクリック音を制御します。onを指定すると音が出るようになり、offを指定すると音は出なくなります。
- BASICの起動時はclick onの状態になります。
- クリック音はコンピュータが有効な文字入力を行なったときに発生します。したがって SHIFT キーや CTRL キーを単独に押しただけではクリック音は発生しません。また、フロッピーディスクの動作中は割り込み禁止になりますので文字を入力する場合、クリック音が出るまでキーを押し続ける必要があります。

例

```
click off　……　クリック音を止めます。
click on　……　クリック音が出るようにします。
```

REPEAT ON/OFF （リピート オン/オフ）

キーリピートを制御します。

書　式

repeat {on〔,〈スピード〉〕 / off}

省略形

rep.　o./of.

説　明

- キーを押し続けたときのキーリピートとリピート速度を指定します。onを指定するとリピート可能となり、offを指定するとキーを押し続けても、リピートを行ないません。
- repeat onの指定のときに<スピード>でリピート速度を指定します。

  <スピード>

  0　…リピートは行ないません。（repeat offと同様）

  1　…低速リピート。

  2　…中速リピート。（省略時はこの速度になります。）

  3　…高速リピート。

  4　…最高速リピート。

  <スピード>が4に設定されている場合は CTRL ＋ D キーの操作により<スピード>を2に設定します。また、<スピード>を省略した場合は2が指定されたものとします。
- BASICの起動時はrepeat onの状態になります。
- リピートは、ダイレクトモードまたはinput文の実行時などキー入力待ちのときにリピート状態になります。先行入力時はリピートを行ないません。

例

```
repeat on   ……  中速リピートに設定します。
repeat off  ……  キーを押し続けてもリピートしなくなります。
```

GET （ゲット）

キーボードより1文字入力を行ないます。

書　式　get〈入力変数〉

説　明

- キーボードより1文字のデータを＜入力変数＞に代入します。入力がなかった場合は入力を待つのではなく、値を0またはNull(空文字)として次の処理に進みます。
- ＜入力変数＞は、数値型、文字型のいずれの型でも使用できます。数値型の変数を使用した場合、1～9の数値キーが押されていればその値、それ以外のキーであれば0を代入します。文字型の変数を使用した場合は、1文字のデータを代入し、キーが押されていなければ、Nullを代入します。
- 入力は先行入力のキーバッファの最初の文字となりますので、先行入力によりすでに入力されている場合は、順次1文字ずつ入力します。あらかじめキーバッファを消去する場合はdef key(0)=""を実行してください。
- リアルタイム入力を行なうときはrepeat on,4としてリピート速度を最高速としておきます。ただしリピートが最高速の場合、通常のキー入力はできませんので、プログラムを終了する場合やinput文の実行前にはもとの速度に戻しておく必要があります。

例　get A$ …… 入力した1文字を変数A$に代入します。

システム変数

INKEY$ （インプット キー ダラー・input key$)

キーボードより1文字入力を行ないます。

書　式　inkey$

省略形　ink.

説　明

- キーボードより1文字のデータを入力するシステム変数で、文字列型データとして値を返します。
- 入力機能は、get文で文字列型の入力を行なうのと同様です。

DEF KEY （ディファイン キー）

ファンクションキーを定義します。

書　式　def key（〈キー番号〉）＝〈文字列式〉

省略形　def　k.

説　明

- 20種のファンクションキー（F11～F20は SHIFT キーを押しながら F1 ～ F10 のキーを押します。）に対して、それぞれのキーが押されたときの入力文字列を定義します。
- ＜キー番号＞は1～20で指定し、＜文字列式＞で押されたときの内容を指定します。＜文字列式＞は15バイトまで定義できます。
- klist文により、最下行ファンクション表示の制御や定義書式での定義内容の表示ができます。

例

```
def key (2)="list"+chr$(5,13)
```

…… F2 のキーに l , i , s , t , CTRL ＋ E , ↵ を定義します。定義後 F2 のキーを押すと"list"を表示して行末まで消去し、listコマンドを実行します。

参　考☞　key list …… ファンクションキーの定義状態を表示。

KEY LIST／KLIST （キーリスト）

ファンクションキーの定義状態を表示します。

書　式

key　list
klist 〔〈機能〉〕

省略形

k. l.　　kl.

説　明

- 文字画面の最下行へのファンクションキーの表示制御やすべての定義状態の表示を行ないます。
- ＜機能＞は、0～2と指定なし（省略）の4種の表示機能があります。
  - klist 0とすると機能1、2による最下行の定義状態の表示を消します。
  - klist 1とすると最下行にファンクションキーの定義状態の表示が行なわれます。
  - klist 2とすると最下行に定義状態と時計を表示します。
- key listまたはklistのみの場合はすべてのキーの定義状態を表示します。表示はdef key文による定義の書式になっていますので、スクリーンエディットにより定義の変更が簡単に行なえます。
- 機能1、2を指定したとき最下行がconsole文によるスクロール範囲に含まれる場合やスムーススクロールモードのときは表示されませんが、のちにラインスクロールで最下行がconsole文の指定範囲外になった場合は、klist 1またはklist 2の指定が行なわれていれば定義状態を表示します。
- 最下行の表示は、リバース（反転）文字で行なわれ、80桁表示のときは10個分、40桁表示のときは5個分の定義内容の表示になります。ただし機能2のときは時計表示に右端の1個分を使用しますので、定義の表示は1個分少なくなります。
- 定義文字中の文字コードが31（&H1F）以下のコントロールコードは、最下行表示の場合はCGの対応するフォントで表示し、定義書式による表示の場合は、chr$( )による表示になります。

例

```
klist 1
```
…… 最下行にファンクションキーの表示を行ないます。

参　考☞

def　key …… ファンクションキーの定義。

DEF KEY(0) （ディファイン キーゼロ）

キーボード入力のシミュレーションを行ないます。

書 式　def key (0)＝(〈文字列〉)

省略形　def k. (0)

説 明

- キーの先行入力に用いられるキーバッファへあらかじめ文字を定義しておき、キー入力を自動的に行なわせます。
- ＜文字列＞があらかじめ定義しておく文字で31文字まで指定できます。＜文字列＞には文字コードが&H20以下のコントロールコードも定義することができます。
- シミュレーションは直接実行モード時、input文、line input文およびget文、inkey $の実行時に行なわれます。
- BREAK キーによる一時停止を行なうとキーバッファは消去されますので注意が必要です。
- この命令を実行すると、それまでにキーバッファに残っているデータは消去され、新たに＜文字列＞の内容がキーバッファに入ります。

例

```
def key (0)="list"+chr$(5,13)
```
…… 直接実行モードで実行すると、listコマンドを実行します。

```
def key (0)=""
```
…… キーバッファを消去します。

関数

STICK （スティック）

ジョイスティック状態を調べます。

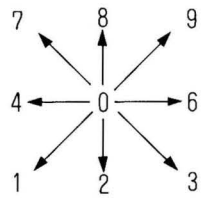
書 式 stick(〈スティック番号〉)

省略形 sti.

説 明

- ジョイスティックの状態を調べて値を返す関数です。
- <スティック番号>は、ジョイスティック1,2を次のように指定します。
  1 …… ジョイスティック1
  2 …… ジョイスティック2

スティックが倒されている方向により次の値を返し、スティックが倒されていない場合は、0を返します。



例

```
if stick(1)=4 then print "Left"
```

…… ジョイスティック1が左に倒された場合、"Left"と表示します。

STICK ON/OFF （スティック オン/オフ）

ジョイスティックでカーソルキーの代用をさせます。

書 式 stick {on / off}

省略形 sti. o./of.

説 明

- stick onとするとジョイスティック1をカーソルキーの代用として使用できます。
- stick関数を使用する場合はstick offとして通常の状態に戻してください。

STRIG （スティック トリガー・stick trigger）

ジョイスティックのトリガーボタンの状態を調べます。

書　式　strig(〈スティック番号〉)

省略形　str.

説　明

- ジョイスティックのトリガーボタンが押されているかを調べる関数です。
- ＜スティック番号＞は、ジョイステック1,2を次のように指定します。
- ＜スティック番号＞は、ジョイスティック1,2またはキーボードを次のように指定します。

  1 …… ジョイスティック1

  2 …… ジョイスティック2
- この関数の結果はトリガーボタンの状態により0〜3の値になり、次のように対応します。

値	トリガーボタン1	トリガーボタン2	
0	OFF	OFF	
1	ON	OFF	ON …押されている
2	OFF	ON	OFF…押されていない
3	ON	ON	

INIT "LPT:" (イニシャライズ エル ピー ティー・(initialize L.P.T.))

プリンタスプーラの設定を行ないます。

書 式

init "LPT:〔{0 / 1 / 2}〕"

説 明

- init "MEM:"文で設定されたプリンタスプーラの機能設定を次のように指定します。(プリンタスプーラについては第1章の「プリンタ制御機能」を参照してください。)

  0 …… ダイレクトモード
  バッファの初期化を行ない、ダイレクトモードにします。

  1 …… スプーラモード
  バッファの初期化を行ない、スプーラモードにします。

  2 …… ダイレクトモード
  この指定を行なう前にスプーラモードであった場合は、バッファ内のデータをすべてプリンタに出力したのちにダイレクトモードにします。

- 指定を省略してinit "LPT:"とするとバッファの初期化を行ないます。
- バッファの初期化を行なうと、それまでにバッファ内に記憶されているデータは消去します。
- プリンタが画面のコピーや非漢字プリンタでの漢字印字などのビットイメージ印字を行なっている途中にバッファの初期化を行なうと、以後のプリンタ印字が正常に行なわれない場合があります。このような場合、一度プリンタの電源を切って、プリンタ自身のリセットを行なってください。
- init "MEM:"文でスプーラ用メモリサイズを設定した直後は、0を指定したのと同様のダイレクトモードになります。

例

```
init "LPT:1" …… スプーラモードにします。
```

参 考

init "MEM:" …… メモリファイルとスプーラのサイズを設定する。

option screen
…… グラフィックV-RAMの一部をメモリファイルやプリンタスプーラに使用できるように設定する。

HCOPY （ハード コピー・hard copy）

画面の表示をプリンタにコピーします。

書 式　hcopy〔〈機能〉〕

省略形　h.

説 明

- 画面に表示されている文字や図形をプリンタにコピー(印字)します。
- ＜機能＞は、1～3で指定してコピーする画面を選択します。省略時は3の機能になります。
  - 1　…文字画面のみをコピーします。
  - 2　…グラフィック画面のみをコピーします。
  - 3　…文字画面とグラフィック画面の両方をコピーします。
- プリンタへのコピーはディスプレイ表示の次の設定の影響を受けます。

  console@ 文による文字表示出力範囲。

  view@文によるグラフィック表示出力範囲。

  screen文による出力プレーン。

  グラフィック画面どうしの優先順位。

  color=文で設定されたカラーコード0。
- この命令の終了後はプリンタが初期化され、改行ピッチは1/6インチになります。

例

```
hcopy
```

…… ディスプレイに表示されている画面全体のグラフィックと文字をプリンタへコピーします。

LPOUT (ライン プリンタ アウトプット・line printer output)

プリンタにデータを出力します。

書　式　lpout 〈出力データ〉

省略形　lp.

説　明

- print/p文の場合コントロールコードを送ろうとしても、漢字モードの切り換えやシフトJISコードからJISコードへの変換、またはイメージモードでの文字印字などを行なっていますので、そのままデータをプリンタへ出力することはできません。この命令は＜出力データ＞の内容をそのまま、プリンタに出力するために用意されています。
- ＜出力データ＞は文字列型の定数や変数、式で指定し、データの1バイトずつを文字コードとしてプリンタへ送り出します。また、print/p文と異なり、データの終了時に改行コードは送られませんので改行が必要な場合は、＜出力データ＞の中に改行コードを含める必要があります。

例

```
lpout chr$(&H1B)+"8"
```

…… プリンタへ&H1B、&H38の2バイトのデータを送ります。

WIDTH PRINT/P （ウィドス プリント・width print）

プリンタの印字幅を設定します。

書　式　width print/p 〈印字幅〉

省略形　wi.

説　明

- プリンタに出力する場合の、1行に印字される文字数を指定します。
- ＜印字幅＞は1～255までの値で指定でき、1行に表示される半角文字の数で表します。ただし、＜印字幅＞を255にした場合は、この機能はなくなります。
- BASIC起動時の＜印字幅＞は255になっています。
- この命令はプリンタのもつ印字幅より幅のせまい用紙を使用した場合などに使用してください。

例　`width print/p 72` …… 印字幅を72桁に設定します。

関数

LPOS （ライン ポジション・line position）

現在のプリンタのヘッド位置を求めます。

書　式　lpos(〈数式〉)

説　明

- print/p文によるプリンタ印字で、次に文字を印字する位置がどこにあるのかを求める関数です。
- ＜数式＞はダミーで数値データであればどのような値でもかまいません。通常は0を指定します。
- この関数の値は、次の印字位置が左端から半角文字で何文字分進んだ位置にあるかを示します。したがって、左端にある場合は、値が0になります。
- この関数でのヘッド位置は論理的な位置になりますので、実際のプリンタの物理的なヘッド位置とは異なります。画面上のカーソル水平位置を求めるcsrhをプリンタに対応させたものとみることができます。

例

```
print lpos(0)
```

…… 現在のプリンタのヘッド位置を画面に表示します。

システム変数

TI$ （タイム ダラー・time$）

時刻の読み出しおよび設定を行ないます。

書 式

ti$ …読み出し

ti$＝"〈時〉〈分〉〈秒〉" …設定

説 明

- 内蔵のリアルタイムクロックに対するシステム変数で、現在時刻を文字列型データとしてもち、代入を行なうことにより設定ができます。
- 時刻は24時間で6文字の文字列型データとなり、＜時＞、＜分＞、＜秒＞それぞれ2桁ずつで表します。

例

```
print ti$    …… 現在時刻を表示します。
ti$="180500"  …… 現在時刻を午後6時5分0秒に設定します。
```

システム変数

TIME （タイム）

タイマーの読み出しおよび設定を行ないます。

書 式

time …読み出し

time＝〈数式〉 …設定

説 明

- 内蔵タイマーに対するシステム変数で0.1秒単位の値をもち、＜数式＞の値を代入することにより設定ができます。
- タイマーは0.1秒単位で0～65535までの値をもちます。
- タイマーの値が65535になるとそのまま停止します。

例

```
print time   …… 現在のタイマーを表示します。
time=0   …… タイマーを0に設定します。
```

DATE$ （デート ダラー）

日付の読み出しおよび設定を行ないます。

書 式

date$ …読み出し

date$＝"〈年〉/〈月〉/〈日〉" …設定

説 明

- 内蔵のリアルタイムクロックに対するシステム変数で、現在の日付を文字列型データとしてもち、代入することにより設定ができます。
- 日付は8文字の文字型データとなり、＜年＞、＜月＞、＜日＞それぞれ2桁ずつで表し、句切にスラッシュ(／)を使用します。
- ti$が"235959"から"000000"になるときに自動的に次の日付に更新します。
  また、うるう年の処理も行ないますので年は西暦の下2桁で指定してください。

例

```
print date$ ……  現在の日付を表示します。
date$="85/12/03" ……  日付を1985年12月3日に設定します。
```


DAY$ （デイ ダラー）

曜日の読み出しおよび設定を行ないます。

書 式

day$ …読み出し

day$＝"〈曜日〉" …設定

説 明

- 内蔵のリアルタイムクロックに対するシステム変数で、現在の曜日を文字列型データとしてもち、代入を行なうことにより設定ができます。
- ＜曜日＞は3文字の文字列型データで表します。読み出し時は1文字目が大文字であとは小文字になりますが、指定時は大文字、小文字のどちらでも使用できます。

日	月	火	水	木	金	土
Sun	Mon	Tue	Wed	Thu	Fri	Sat

- ti$が"235959"から"000000"になるときに自動的に更新します。

例

```
print day$ ……  曜日を表示します。
day$="SAT" ……  土曜日に設定します。
```

MUSIC （ミュージック）

音楽演奏をします。

書　式

music 〈パート1〉[;〈パート2〉]…[;〈パート6〉][,〈パート1〉…;〈パート6〉]…
music init
music wait

省略形

mu.　mu. ini.　mu. w.

説　明

- ＜パート１＞から＜パート６＞による音楽演奏や演奏機能に関する制御を行ないます。
- ＜パート＞は音階や音長、オクターブ、音量、音色などの演奏データを文字列型データで表します。＜パート１＞～＜パート３＞はFM音源のチャンネル1～3に、＜パート４＞～＜パート６＞はSSG音源のチャンネルA, B, Cに対応し、同時に演奏が行なわれます。（演奏データの詳細は次ページ以降を参照してください。）
- ＜パート1～6＞の演奏はmusic文の実行とともに、同時に行なわれます。各＜パート＞の演奏時間が異なる場合の演奏時間は、一番長い＜パート＞の演奏時間となり、演奏時間の短かい＜パート＞は休符が続くものとします。
- 演奏の実行は、各＜パート＞の指定とともに開始され、プログラムは次の処理に進みますが、演奏中（以前のmusic文の実行による演奏が終了していない）であれば演奏の終了を待ち、演奏の終了後に各＜パート＞の設定を行ない演奏を開始し、次のプログラム処理に進みます。また、音楽演奏の終了時に割り込みをかける機能があり、on music gosub文で処理の定義、music on/off/stop文で処理の制御を行ないます。
- music initはオクターブや音長、音色などの初期化を行ないます。

＜パート＞	初期化に相当する演奏データ
1～3	"O4L5T4Q8@0@V100@M0"
4～6	"O4L5T4Q8V12M255"

- music waitはこの命令の実行時に音楽が演奏中であれば、演奏の終了を待ちます。通常は各パートを指定して演奏を開始するとすぐに次の処理に進み、プログラム処理と同時に演奏が行なわれますが、音楽演奏時にこの命令を実行すると音楽演奏が終了するまで待ち、次の処理に進みます。

＜パート＞の演奏データは次のようになります。

機能一覧

文 字	機能
C～G, A, B	音階
#	音階を半音上げる
R ℓ	休符（ℓ :0～9）
.	音長を1.5倍にする
Nx	音階コードxによる音の発生（x:0～95）
Qn	音の出る時間比率の設定（n:1～8）
&	前後の音を継ぐ（タイ）
{ } ℓ	連符。音長ℓ を { } 内の音符数で割った音を発生させる（ℓ :0～9）
L ℓ	省略時の音長指定（ℓ :0～9）
Ts	テンポ指定（s:1～7）
On	オクターブ指定（n:1～8）
>	オクターブを１つ上げる
<	オクターブを１つ下げる
@Vn	FM音源の音量設定（n:0～127）
@n	FM音源の音色設定（n:0～29）
Vn	SSG音源の音量設定（n:0～15）
Sn	SSG音源のエンベロープ形状設定（n:0～15）
Mn	SSG音源のエンベロープ周期設定（1～16383）
Yr, d	OPNのレジスタrにデータdをセット
@W ℓ	ℓ の長さだけ状態を維持する（ℓ :0～9）
@Mn	FM音源チャンネル３のモード設定（n:0～2）
=＜変数名＞;	値を変数で指定する

FM音源はパート1～3、SSG音源はパート4～6になります。

- 音符の指定

$$[\left\{\begin{matrix}+\\-\end{matrix}\right\}][\#]\text{＜音階＞}[\text{＜音長＞}][.]$$

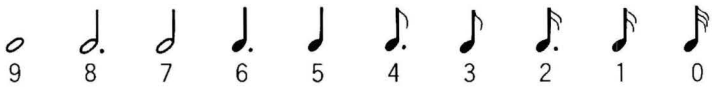
＋とーはオクターブの上下を指定します。指定を省略した場合は、Oで設定されているオクターブになり、＋を指定すると設定より1オクターブ上、ーを指定すると設定より1オクターブ下になります。

＜音階＞はアルファベットのC～G, A, Bを使用して次のように対応します。

C	D	E	F	G	A	B
ド	レ	ミ	ファ	ソ	ラ	シ

音階の前に＃を指定すると半音上がります。Bの半音上がった音は1オクターブ上のCになります。

＜音長＞は0～9で指定し、省略した場合は直前に指定されている音長またはLで設定された音長になります。＜音長＞は次のように対応します。



9	8	7	6	5	4	3	2	1	0

ピリオド(.)を指定すると音長を1.5倍にします。

- 休符の指定

R〔＜音長＞〕〔.〕

＜音長＞とピリオドの機能は音符の指定と同じです。この間は音が発生しませんが音色によっては前の音の余韻が残る場合があります。(エンベロープのRRの設定によります。)

- 連符の指定

{}〔＜音長＞〕

{} 内の音符を、＜音長＞を音符数で等分に割った長さで演奏します。＜音長＞の省略時はLで設定されている長さになります。音長 {} 内の音符数で割った値が32分音符より短い場合はエラーになります。

- 音を続ける(タイ)

&

この記号の前後の音を続けて1つの音とします。前後の音程が異なる場合は前の音だけがQ8の指定をされたのと同じになります。

- 音の発生比率の設定

Q＜比率＞

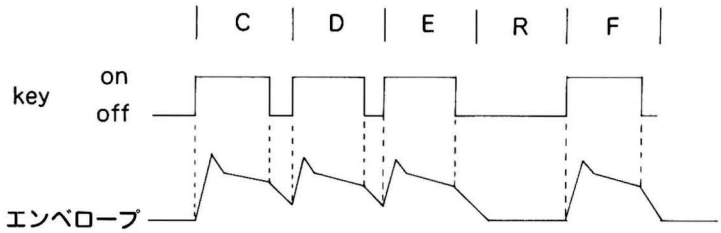
1つの音符で、その音長の間で実際に音を出している時間の指定です。＜比率＞は1～8で指定し、次のように対応しています。

指定	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
音の長さ	1/8	2/8	3/8	4/8	5/8	6/8	7/8	8/8

＜比率＞を小さくするとスタッカートになります。

音の発生をkey-on、停止をkey-offと呼びます。エンベロープとの関係は次のようになります。

例）music "Q6CDERF" とした場合



- 音階コードによる音の発生

N<音階コード>

<音階コード>による音階の音を発生します。音長はLにより設定された長さになります。

<音階コード>は0～95で指定し次のようになります。

	O1	O2	O3	O4	O5	O6	O7	O8
C	0	12	24	36	48	60	72	84
＃C	1	13	25	37	49	61	73	85
D	2	14	26	38	50	62	74	86
＃D	3	15	27	39	51	63	75	87
E	4	16	28	40	52	64	76	88
F	5	17	29	41	53	65	77	89
＃F	6	18	30	42	54	66	78	90
G	7	19	31	43	55	67	79	91
＃G	8	20	32	44	56	68	80	92
A	9	21	33	45	57	69	81	93
＃A	10	22	34	46	58	70	82	94
B	11	23	35	47	59	71	83	95

- 省略時の音長設定

L<音長>

音符や休符、**N**、{ }で音長を省略したときの音長を設定します。<音長>は**1～64**で音符の指定と同じです。初期化されたときは**"L 5 "**になります。

- テンポの設定

T<スピード>

<スピード>は1～7までの値で指定します。初期化されたときは **"T4"**になります。

<スピード>	1	2	3	4	5	6	7
テンポ(♩=)	32	61	90	120	165	210	255

- オクターブの設定

  O＜オクターブ＞

  ＜

  ＞

  オクターブの設定を行ないます。＜オクターブ＞は**1～8**で指定し、**"O4A"**が**440Hz**になります。**＞**は現在より1つ上へ、**＜**は現在より1つ下へのオクターブの変更になります。初期化されたときは**"O4"**になります。

  FM音源は音色により基本周波数が異なりますので**"O4A"**が**440Hz**にならない場合があります。

- **FM**音源の音量の設定

  @V＜音量＞

  FM音源のパート内で使用し、そのパートの音量を指定します。＜音量＞は**0～127**までで指定し、**127**が最大音量になります。初期されたときは**"@V100"**になります。この指定は**SSG**音源のパートでは無視します。

- **FM**音源の音色の設定

  @＜音色＞

  FM音源のパート内で使用し、そのパートの音色をあらかじめ用意された30種の音色の中から選びます。＜音色＞は**0～29**で指定し、初期化されたときは**"@ 0 "**になります。この指定は**SSG**音源のパートでは無視します。

- **SSG**音源の音量設定

  V＜音量＞

  SSG音源のパート内で使用し、そのパートの量を設定します。＜音量＞は、**0～15**で指定し、**"V15"**が量大音量になり**"V 0 "**は音が出ません。初期化されたときは**"V12"**になります。この指定を行なうと、**S**によるエンベロープ形状の設定は無効になります。この指定は**FM**音源のパートでは無視します。

- **SSG**音源のエンベロープ形状の設定

  S＜形状番号＞

  SSG音源のパート内で使用し、そのパートのエンベロープ形状を設定します。＜形状番号＞は、**0～15**で指定します。形状番号によるエンベロープ形状は**sound**文を参照してください。この指定を行なうと**V**による音量は無効になります。

  この指定は**FM**音源のパートでは無視します。

- **SSG**音源のエンベロープ周期の設定

  M＜周期＞

  SSG音源のパート内で使用し、そのパートのエンベロープ周期を設定します。＜周期＞は、**1～16383**までで指定し、**16383**がいちばん周期が長くなります。エンベロープの周期は＜周期＞を**7812.5**で割った値が秒単位の周期

になります。

この指定はFM音源のパートでは無視します。初期化された場合は"M255"になります。

- OPNのレジスタに値を設定

  Y＜レジスタ番号＞,＜データ＞

  OPNの内部レジスタへ直接データを設定します。＜レジスタ番号＞は0～180、＜データ＞は0～255で指定します。値を16進数で指定する場合は$を付けてください。

- 状態の維持

  @W[＜長さ＞]

  ＜長さ＞で指定された時間だけ現在の状態を保ちます。＜長さ＞は、音符の音長と同じで省略時は、Lの設定値となります。この指定はYによりレジスタを設定したあとに使用し、設定値による音を出すために用意されています。(key-onはレジスタ設定により行なってください。)

- FM音源のチャンネル3のモード切り換え

  @M＜モード番号＞

  FM音源のチャンネル3はモード切り換えが可能でパート3でこの指定を行ないます。＜モード番号＞は次のように対応します。

  0 …… 通常の音楽モード

  1 …… CSM(サイン波合成)モード

  2 …… 効果音モード

  この指定はSSG音源のパートでは無視します。

- 値の変数による指定

  ＝＜変数名＞;

  演奏データの中での数値指定を行なう所で使用し、そのときの指定された変数の値がその位置で使用されます。

例)
```
10 for A=10 to 120 step 10
20 music "@V=A;CDE"
30 next
```

…… 音量を10ステップで上げながらドレミの音を出します。

• 音色表

music文の"@n"の指定やtone copy文で指定される音色コードです。

音色番号	音色	
0	HARPSIC	ハープシコード
1	BRASS 1	ブラス系の音
2	BRASS 2	
3	TRUMPET	トランペット
4	STRING 1	ストリング系の音
5	STRING 2	
6	EPIANO 1	エレクトリックピアノ
7	EPIANO 2	
8	EPIANO 3	
9	GUITAR	ギター
10	EBASS 1	エレクトリックベース
11	EBASS 2	
12	EORGAN 1	エレクトリックオルガン
13	EORGAN 2	
14	PORGAN 1	パイプオルガン
15	PORGAN 2	
16	FLUTE	フルート
17	PICCOLO	ピッコロ
18	OBOE	オーボエ
19	CLARINET	クラリネット
20	GROCKEN	グロッケン
21	VIBRPHN	ビブラホン
22	XYLOPHN	シロホン
23	KOTO	琴
24	ZITAR	チター
25	CLAV	クラビネット
26	BELL	ベル
27	HARP	ハープ
28	HARMONICA	ハーモニカ
29	TIMPANI	ティンパニー

注） 音色により、基本周波数が異なりますので注意してください。（**"O4A"**の音が**440Hz**にならない場合があります。）

音色により、音量が異なりますので複数の音を演奏に使用される場合は、音色ごとに**@Vn**で音量を調整してください。

TEMPO （テンポ）

音楽演奏のテンポを設定します。

書　式　tempo〈スピード〉

省略形　te.

説　明
- music文による音楽演奏の速度を設定します。
- ＜スピード＞は1～7の値で指定し、次の速度に対応します。

＜スピード＞	1	2	3	4	5	6	7
テンポ(♩＝)	32	61	90	120	165	210	255

例　tempo 4

BEEP （ビープ）

スピーカを一瞬鳴らします。

書　式　beep

省略形　b.

説　明
- この命令を実行するとスピーカから一瞬、「ピッ」という音が出ます。
- この音は文字コード7で発生する音(BELL)と同じです。
- この音の周波数は約880Hzで約0.055秒間(サイクル数48)、音が鳴ります。

例　beep　…　「ピッ」という音を出します。

TONE （トーン）

FM音源の音色を設定します。

書 式　tone〈整数配列名１〉[,〈整数配列名２〉][,〈整数配列名３〉]

説 明

- 整数型の配列に設定されている音色データをFM音源の各チャンネルに設定します。＜整数配列名＞の1～3はそれぞれFM音源の1～3チャンネルに対応します。また配列は２次元配列で添字の下限を０(option base　0)として(4, 9) の大きさが必要です。
- 配列内のデータは引数の右順が0のときは全体的な設定で、1～4はそれぞれのオペレータの指定になり、次の内容を設定します。

n	(0, n)	(1, n)～(4, n)
0	Feedback/Algorithm	Attack Rate
1	Operator mask	Decay Rate
2	Wave Form	Sustain Rate
3	Sync	Release Rate
4	Speed	Sustain Level
5	Pitch Modulation Depth	Output Level
6	Amplitude Modulation Depth	Keyboard Rate Scaling Depth
7	未使用	Multiple
8		Detune
9		Amplitude Modulation Sensitvity

(設定の内容は次ページを参照してください。)

例

```
tone PIANO%,BELL%
```

…… FM音源のチャンネル1に整数型配列PIANO%(4, 9)より、チャンネル2は整数型配列BELL%(4, 9)より指定値を設定します。

参 考☞　tone copy …… BASICで用意された音色データを配列に取り込む。

(0, 0);Feedback/Algorithm

以下の意味をもつ6ビットのデータになっています。

D7	D6	D5	D4	D3	D2	D1	D0
未　使　用		Feedback			Algorithm		

Feedback(0～7)

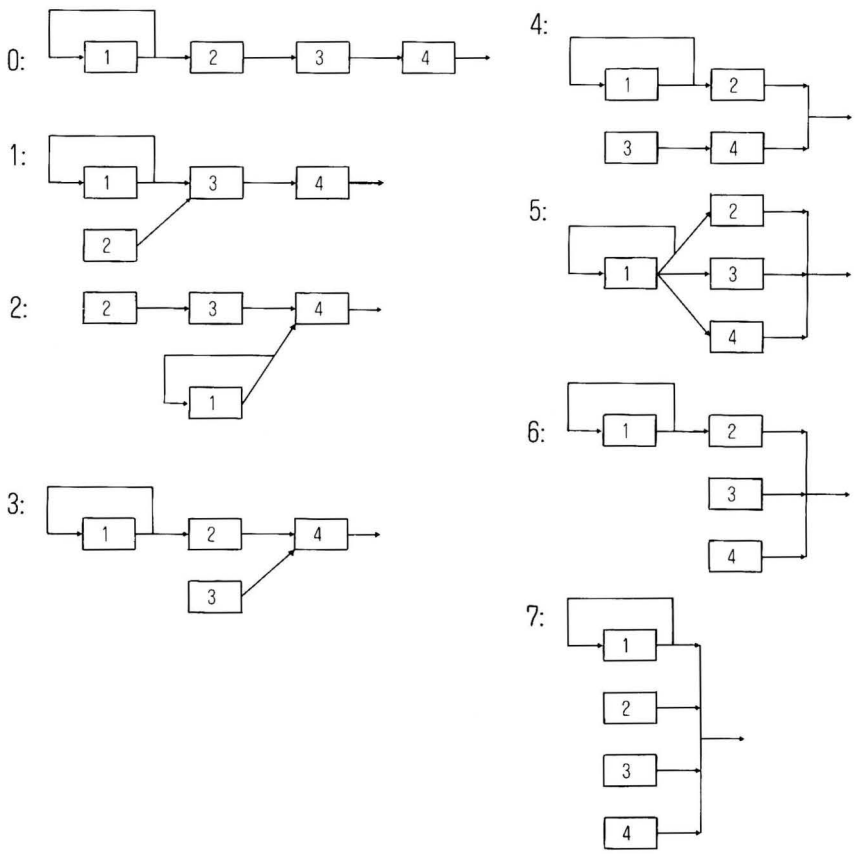
第1オペレータの変調度を設定します。

第1オペレータは自分自身の出力を変調信号としているため、この変調度をこれにより制御します。

	0	1	2	3	4	5	6	7
変調度	OFF	π/16	π/8	π/4	π/2	π	2π	4π

Algorithm(0～7)

4つのオペレータの接続の仕方を設定します。値はそれぞれ次のようなアルゴリズムに対応します。



(0,1):Operator mask(0～15)

各オペレータを使用するか否かを設定します。それぞれの値は次のとおりです。

値	OP4	OP3	OP2	OP1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

0…使用しない　　1…使用する

(0,2):Wave Form (0～3)

0　ノコギリ波

1　矩形波

2　三角波

3　ランダム

(0,3):Sync

Key onとLFO(Low Frequency Oscillator)の同期を指定します。

0　Key-onと無関係にLFOが動作します。

1　Key-onと同期してLFOが動作します。

(0,4):Speed(0～255)

LFOの速度を決めます。大きいほど速くなります。

(0,5):Pitch Modulation Depth(－127～127)

音階に対してLFOをかける深さを設定します。絶対値が大きいほど深くかかります。負の値にすると、波形の極性が逆になります。

(0, 6):Amplitude Modulation Depth(−127〜127)

音量に対してLFOをかける深さを設定します。絶対値が大きいほど深くかかります。負の値をとると、波形の極性が逆になります。

(0, 7):未使用

(0, 8):未使用

(0, 9):未使用

OP=1, 2, 3, 4(オペレータ番号)

(OP, 0):Attack Rate(0〜31)

(OP, 1):Decay Rate(0〜31)

(OP, 2):Sustain Rate(0〜31)

(OP, 3):Release Rate(0〜15)

(OP, 4):Sustain Level(0〜15)

(OP, 5):Output Level(127〜0)

これは減数量の指定で、0が最大出力になります。

(OP, 6):Keydoard Rate Scaling Depth(0〜3)

高音になるほどEGの変化速度を速くする度合を設定します。

(OP, 7):Multiple(0〜15)

周波数比を設定します。0で$\frac{1}{2}$になります。

(OP, 8):Detune(−3〜3)

音階のずれの度合を設定します。

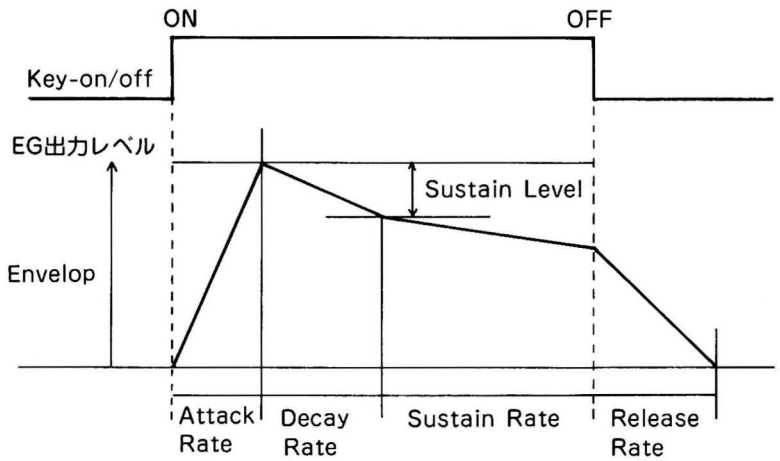
(OP, 9):Amplitude Modulation Sensitivity(0〜2)

音量に対するLFOをかける度合を設定します。数値が大きいほど変化の幅が大きくなります。

EG(Envelop Generator)

楽器の音は時間とともに音量、音色が変化します。この時間的な変化をエンベロープといいます。そして、このエンベロープを人工的に作る機能をEG(Envelop Generator)といいます。

EGは以下のパラメータをもっており、それを設定することによって、より自然の楽器に近づけることができます。



AR(Attack Rate)0～31

音が発生(Key-on)してからEGの出力レベルが最大になるまでの時間を設定します。数値が大きいほど短くなります。

DR(Decay Rate)0～31

EGの出力レベルが最大になってからSustain Levelに達するまでの時間を指定します。数値が大きいほど短くなります。

SR(Sustain Rate)0～31

Sustain LevelからEGの出力レベルが0になるまでの時間を設定します。数値が大きいほど短くなります。

RR(Release Rate)0～15

Keyを離してからEGの出力レベルが0になるまでの時間を設定します。数値が大きいほど短くなります。

SL(Sustain Level)0～15

DecayからSustainに移るときのレベル(減衰量)を設定します。数値が大きいほどEGの出力レベルが低くなります。

TONE COPY （トーン コピー）

BASICで用意された音色データを配列に取り込みます。

書 式

tone copy 〈音色番号〉,〈整数型配列名〉

説 明

- ＜音色番号＞で指定された音色のデータを＜整数型配列名＞で設定された配列に取り込みます。
- ＜音色番号＞はmusic文で使用される音色に対応し、0～29の値で指定します。（music文の音色表を参照してください。）
- 音色データを取り込む配列はあらかじめdim文で(4,9)の大きさで定義しておきます。また、添字の下限は0(option base 0)に設定してください。
- 配列内のデータの内容は、tone文でFM音源に設定する値と同じです。

例

```
tone copy 0,HARP%
```

…… 音色番号27の音色データを(4,9)の大きさの整数型配列変数HARP%に取り込みます。

参 考

tone …… 音色データをFM音源に設定する。

TONE LFO (トーン エル エフ オー・tone Low Frequency Oscillator)

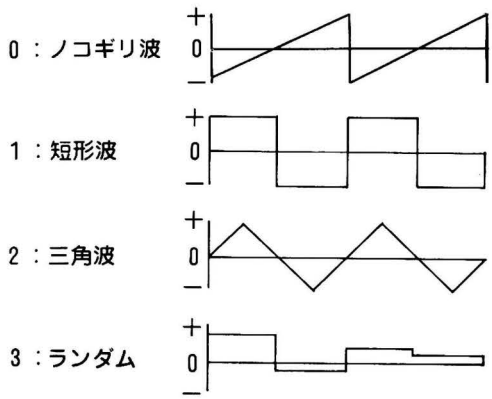
ビブラートやトレモロ機能を設定します。

書　式

tone lfo 〈チャンネル〉[,〈波形〉][,〈同期〉][,〈スピード〉][,〈PMD〉][,〈AMD〉]

説　明

- LFOとは(Low Frequency Oscillator)の略で、低い周波数を発生させ、ビブラート(音程を細かく変化させる)やトレモロ(音量を細かく変化させる)の周期に使用します。この命令では、LFOの周期や波形、変調の深さを設定します。
- <チャンネル>は、music文のパート番号1～6で指定します。
- <波形>は、LFOの変化を次の波形0～3選択します。



- <同期>は1とすると音の発生(key-on)と同時にLFO波形を発生させ、0とすると音の発生に関係なくLFO波形を発生します。
- <スピード>は0～255でLFOの周波数を指定し、値が大きくなるほど周波数が高くなります。
- <PMD>(Pitch Modulation Depth)の値は－127～127で、音程に対してLFOをかける深さを指定します。この絶対値が大きいほど変化の幅が大きくなります。負の値を推定した場合、LFOの波形の極性が逆転します。(ビブラートをかけるのに使います。)
- <AMD>(Amplitude Modulation Depth)の値は－127～127で、音量に対してLFOをかける深さを指定します。この絶対値が大きいほど変化の幅が大きくなります。負の値の場合、LFO波形の極性が逆転します。(トレモロをかけるのに使います。)

例

```
tone lfo 1,0,1,128,0,64
```

…… チャンネル1をノコギリ波で同期し、スピードは128、ビブラートをかけずトレモロを64の深さに設定します。

SOUND （サウンド）

SSG音源のレジスタセットや音階コードによる音を鳴らします。

書　式

sound＝(〈レジスタ番号〉,〈データ〉)　…(1)

sound 〈音階〉,〈音長〉　…(2)

省略形

so.

説　明

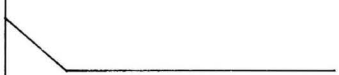
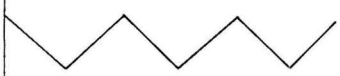
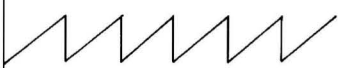
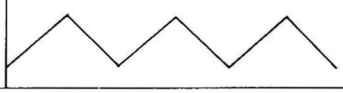
- (1)の書式は本体内のOPN(音楽用IC)のレジスタへデータを書き込み、OPNの直接制御を行います。
- (2)の書式は、音階コードにより、指定時間だけ音を鳴らします。＜音階＞は0～95の値で指定し、music文のNの指定と同じになります。＜音長＞は、0～255の値で指定し、0.1秒単位になります。
- ＜レジスタ番号＞は0～179(&H0～&HB3)まであり、＜データ＞は8ビットで、0～255(&H0～&HFF)の値で指定します。
- SSG音源のレジスタは次のとおりです。

レジスタ番号	レジスタ機能	ビット構成							
		7	6	5	4	3	2	1	0
0	チャンネルA	下位8ビットT_L							
1	周波数					上位4ビットT_H			
2	チャンネルB	下位8ビットT_L							
3	周波数					上位4ビットT_H			
4	チャンネルC	下位8ビットT_L							
5	周波数					上位4ビットT_H			
6	ノイズ周波数			5ビット					
7	チャンネル	$\overline{in}$/out		ノイズ			トーン		
	設定	IOB	IOA	C	B	A	C	B	A
8	チャンネルA音量				M	4ビット			
9	チャンネルB音量				M	4ビット			
10	チャンネルC音量				M	4ビット			
11	エンベロープ周期	下位8ビット E_L							
12		上位8ビット E_H							
13	エンベロープ形状					4ビット			

各チャンネルの周波数は2MHzを16で割りさらにT_HとT_Lの12ビット値で割った値になります。

ノイズ周波数は2MHzを16で割りさらに5ビット値で割った値になります。
チャンネル設定のIOBは0のインプット、IOAは1のアウトプットに固定され、ノイズとトーンのチャンネルA, B, Cはそのビットが0であれば選択になります。
エンベロープ周期は2MHzを256で割りさらにE_HとE_Lの16ビットデータで割った値の周波数による周期になります。
各チャンネルの音量は4ビットで0～15の値になりますがMのビットが1の場合は音量がきかず、エンベロープモードになります。

エンベロープ形状

R13の値	エンベロープパターン
0, 1, 2, 3, 9	
4, 5, 6, 7, 15	
8	
10	
11	
12	
13	
14	

VOICE （ボイス）

音声合成により文字列を発声します。

書　式　voice〔[[〈スピード〉]〔,〈数値読み〉]]]〈文字列〉〔,〈文字列〉〕…

省略形　v.

説　明

- 別売のボイスボード(MZ-1X08)を使用して＜文字列＞の内容を1文字ずつ発声します。発声する文字は数字、アルファベット、カナ、一部の記号(＋－＊／＝.)ですべて半角文字です。ただしアルファベットは日本語による発音で、記号は次のようになります。
  ＋…プラス －…マイナス ＊…カケル ／…ワル ＝…イコール ． …テン
- ＜スピード＞は1～3で指定し、1-低速、2-標準、3-高速になります。
- ＜数値読み＞は数字が並んでいる場合の読み方の指定で、1とすると1桁ずつの棒読みで発声し、2とすると位取りをしながら発声します。ただし、8桁をこえる場合や小数点以下は棒読みとなります。
- ＜スピード＞や＜数値読み＞は省略するとそれ以前の設定値となり、BASIC起動時は、＜スピード＞は2、＜数値読み＞は1になります。
- ＜文字列＞をカンマ(,)で句切って複数並べることができ、＜文字列＞の直前に[ ]を使って＜スピード＞と＜数値読み＞の指定が可能です。
- この命令は"auto-run. s25" プログラム内でnew on 2を行なって消去されています。この命令を使用される場合は、"auto-run. s25" プログラムを変更して再起動してください。また、メモリ標準実装(128キロバイト)の場合、フリーエリアにより使い方が限定される場合があります。詳しくはBASIC-M25マニュアルの付録の「BASICのフリーエリア」を参照してください。

例

```
voice [1] "ABC","ｱｲｳｴｵ"
```

…… 低速で"ABC"と"アイウエオ"を発声します。

参　考　voice@ …… あらかじめ用意された語句を発声する。

VOICE @ （ボイス アットマーク）

あらかじめ用意された語句を発声します。

書 式　voice@〈語句番号〉

省略形　v.@

説 明

- 別売のボイスボード(MZ-1X08)に用意されている語句を発声します。
- ＜語句番号＞は0～33で指定し、次のようになります。

番号	メッセージ	番号	メッセージ
0	まちがいです	16	私の勝ちです
1	正解です	17	文法上の誤りです
2	OK	18	私はMZです。
3	もう1回チャレンジし	19	点
	ますか	20	たす
4	よくできました	21	ひく
5	笑い声	22	イテテ
6	ゲームスタート	23	イテッ！
7	ゲームオーバー	24	自動車(25＋26＋27)
8	アウト	25	自動車（発車）
9	GO	26	自動車（走行）
10	命中	27	自動車（停止）
11	お好きなKEYを押して	28	メロディー(葬送行進曲)
	ください	29	メロディー(デキシーランド)
12	YかNのキーを押して	30	メロディー（蛍の光）
	ください	31	メロディー(おおスザンナ)
13	最高得点	32	メロディー（故郷の人々）
14	ボーナス点	33	メロディー（村のかじ屋）
15	あなたの番です		

- この命令は"auto-run. s25" プログラム内でnew on 2を行なって消去されています。この命令を使用される場合は、"auto-run. s25" プログラムを変更して再起動してください。また、メモリ標準実装(128キロバイト)の場合、フリーエリアにより使い方が限定される場合があります。詳しくはBASIC-M25マニュアルの付録の「BASICのフリーエリア」を参照してください。

例

```
voice@ 1
```
…… 「正解です」と発声します。

TEL （テレフォン・telephone）

モデムホンへの制御を切り換えます。

書 式

tel "COM{1 / 2}:{M / T}"

説 明

- RS-232Cインターフェイスから別売のモデムホン(MZ-1X19)に出力するデータを、モデムホンのコマンドとするか、伝送データとするかを切り換えます。
- M(Modem)を指定すると電話回線への伝送、T(Telephone control)を指定するとモデムホンへの制御データになります。
- 制御の切り換えは、RS-232CインターフェイスのER信号を、MのときはON、TのときはOFFにしています。
- ファイルオープン時はERをonにしますのでこの命令は、ファイルオープン後に実行してください。

例

```
tel "COM:T"
```

…… RS-232Cインターフェイスのチャンネルaをモデムホンへのコマンド状態にします。

INIT "CMT:" (イニシャライズ シー エム ティー・initialize C.M.T.)

ボイスレコーダの機能を設定します。

書　式

init "CMT:〔〈録音入力〉〕〔,〈操作ボタン〉〕〔,〈オートプレイ〉〕
〔,〈オートリワインド〉〕〔,〈プログラム同期〉〕"

説　明

- ボイスレコーダの各種機能を設定します。
- <録音入力>はアナログ録音時の入力を切り換えます。
  0…ライン入力　　1…マイク入力
- <操作ボタン>は前面パネルのEJECT以外のSTOP、PLAY、REC、FF、REWのカセット操作ボタンの入力について指定します。
  0…ボタン操作が可能　　1…ボタン操作は不可能
  この機能はプログラムでカセットの操作を行なっているときに不用意にボタンを押してテープの状態が変わってしまうのを防ぐためです。ただし、テープよりデータを読み取っているときは、この設定に関係なく、ボタン操作はできません。
- <オートプレイ>は巻き戻し動作中、または逆方向のAPSS動作中にすべてのテープが巻き戻され、メカニズムが自動停止したあとに再生状態にするかどうかの指定を行ないます。
  0…ストップ(STOP)状態　1…再生(PLAY)状態
- <オートリワインド>は録音(REC)や再生(PLAY)状態でテープの最後まで巻き取ったときに巻き戻し(REW)を行なうかどうかの指定を行ないます。
  0…ストップ(STOP)状態　1…巻き戻し(REW)状態
- <プログラム同期>はAPSS動作中やCMT=文で時間指定の操作を行なったときに操作の終了を待って次の処理に進むか、カセットの操作と次の処理を同時に行なうかの指定を行ないます。
  0…カセット操作の終了を待たずに次の処理に進みます。
  1…カセット操作の終了を待って次の処理に進みます。
  0を指定するとカセット操作とプログラムが同時進行しますが、カセットのメカニズムの切り換えを行なうときに一時的にプログラムの処理が停止することがあります。また、カセット操作をしたときにそれ以前の操作がまだ行なわれている場合は、先の操作が終了するまで待ちます。

- 指定を省略してinit "CMT："とした場合はすべての指定は0になります。
  た、BASICの起動時もこの設定になります。
- 指定の部分的な省略の場合は、省略された指定は以前のままになります。

例

```
init "CMT:1"
```

…… アナログの録音入力をマイクに設定します。そのほかの指定は以前のままです。

```
init "CMT:,,1"
```

…… オートリワインド機能のみを指定します。

参考

cmt(ステートメント)…… カセットの制御をする。

cmt(関数) …… カセットの状態を確認する。

apss …… カセットの頭出しをする。

CMT= (シー エム ティー イコール・Compact cassette Magnetic Tape=)

ボイスレコーダを制御します。

書　式　cmt=〈機能〉〔,〈時間1〉〔,〈時間2〉〕〕

説　明

- ボイスレコーダの制御を行ないます。＜機能＞は0～17までの整数値で指定し、内容に応じて＜時間1＞や＜時間2＞を指定します。時間は0.1秒単位で1～65535までの整数値で指定できます。
- cmt =0
  カセットのふたを開けます。(EJECT操作)
- cmt =1
  カセットを停止させます。(STOP操作)
- cmt =2〔,＜再生時間＞〕
  カセットを再生状態にします。＜再生時間＞を指定するとその時間だけプレイ状態になり停止します。省略するとそのまま再生状態を続けます。(PLAY操作)
- cmt =3〔,＜早送り時間＞〕
  カセットを早送り状態にします。＜早送り時間＞を指定するとその時間だけ早送りをして停止します。省略するとそのまま早送りを続けます。(FF操作)
- cmt =4〔,＜巻戻し時間＞〕
  カセットを巻戻し状態にします。＜巻戻し状態＞を指定するとその時間だけ巻戻しを行ない、省略するとそのまま巻戻しを続けます。(REW操作)
- cmt =5
  早送りによる頭出しをします。apss 1と同じです。
- cmt =6
  巻戻しによる頭出しをします。apss −1と同じです。
- cmt =7
  現在位置のデジタル側に頭出し信号を録音します。信号の前後を2秒程度の無音状態としますので、ほかの頭出し信号とは再生状態で5秒以上の間隔を開けてください。この機能の実行はアナログ側には影響しません。
- cmt =8〔,＜録音時間＞〕
  アナログ側およびデジタル側の録音をします。＜録音時間＞を指定するとその時間だけ録音状態になり停止します。省略すると録音状態を続けます。
- cmt =9〔,＜録音時間＞〕
  アナログ側のみを録音状態にします。録音時間を指定するとその時間だけ録音状態となり停止します。省略すると録音状態を続けます。(REC＋PLAY操作)

- **cmt =10〔,＜録音時間＞〕**
  デジタル側のみを録音状態にします。＜録音時間＞を指定するとその時間だけ録音状態になり停止します。省略すると録音状態を続けます。
- **cmt =11,＜再生時間＞〔,＜早送り時間＞〕**
  ＜再生時間＞だけ再生状態とし、＜早送り時間＞だけ早送りをして停止します。＜早送り時間＞を省略すると再生後、早送り状態を続けます。
- **cmt =12,＜再生時間＞〔,＜巻戻し時間＞〕**
  ＜再生時間＞だけ再生状態とし、＜巻戻し時間＞だけ巻戻しをして停止します。＜巻戻し時間＞を省略すると再生後、巻戻し状態を続けます。
- **cmt =13,＜再生時間＞〔,＜録音時間＞〕**
  ＜再生時間＞だけ再生状態とし、＜録音時間＞だけアナログ側のみの録音を行ない停止します。＜録音時間＞を省略すると録音状態を続けます。
- **cmt =14,＜早送り時間＞〔,＜再生時間＞〕**
  ＜早送り時間＞だけ早送り状態とし、＜再生時間＞だけ再生して停止します。＜再生時間＞を省略すると再生状態を続けます。
- **cmt =15,＜早送り時間＞〔,＜録音時間＞〕**
  ＜早送り時間＞だけ早送り状態とし、＜録音時間＞だけアナログ側のみの録音を行ない停止します。＜録音時間＞を省略すると録音状態を続けます。
- **cmt =16,＜巻戻し時間＞〔,＜再生時間＞〕**
  ＜巻戻し時間＞だけ巻戻し状態とし、＜再生時間＞だけ再生状態として停止します。＜再生時間＞を省略すると再生状態を続けます。
- **cmt =17,＜巻戻し時間＞〔,＜録音時間＞〕**
  ＜巻戻し時間＞だけ巻戻し状態にし、＜録音時間＞だけアナログ側のみの録音を行ない停止します。＜録音時間＞を省略すると録音状態を続けます。
- 各時間の設定は0.1秒単位で指定しますが、1秒以下の短かい指定の場合、カセットメカニズムの動作が正常に行なわれない場合があります。
- ＜早送り時間＞と＜巻戻し時間＞はそれぞれの操作を行なっている時間で、再生時の時間には対応しません。テープの巻取側リールの直径が大きくなれば、巻取量は多くなり、小さいほど少なくなります。
- ＜機能＞が2〜17はカセットテープがセットされていない場合は何も行ないません。
- ＜機能＞が7〜10、13、15、17の録音状態となるものはカセットの消去防止ツメが折られていると何も行ないません。

CMT (シー エム ティー・Compact cassette Magnetic Tape)

ボイスレコーダの状態を調べます。

書　式　cmt(〈機能〉)

説　明

- ボイスレコーダの状態を調べる関数です。確認できる内容は、テープが回転中であるか、テープがセットされているか、テープの消去防止ツメが折られているかの3種類で＜機能＞に0～2の値を指定することによりそれぞれの内容を調べます。
- ＜機能＞に0を指定するとテープが回転中であるかを調べ次の値を返します。
  テープが回転中である場合　… -1
  テープが停止中の場合　… 0
- ＜機能＞に1を指定するとテープがセットされているかを調べ次の値を返します。
  テープがセットされている場合　… -1
  テープがセットされていない場合　… 0
- ＜機能＞に2を指定するとテープの消去防止ツメが折られているかを調べ、次の値を返します。
  テープの消去防止ツメが折られていない場合　… -1
  テープの消去防止ツメが折られている場合　… 0
  テープがセットされていない場合はテープの消去ツメが折られている状態と同じになります。

例

```
CPLAY=cmt(0)
```
…… 変数CPLAYにテープが回転中かどうかの情報を代入します。
```
if not cmt(2) then print "not Rec."
```
…… もし消去防止ツメが折られていれば"not Rec. "と表示します。

参　考　cmt=（ステートメント）…… ボイスレコーダの制御。

APSS (エー ピー エス エス・Auto Program Search System)

カセットの頭出しを行ないます。

書 式 apss〔〈回数〉〕

省略形 ap.

説 明

- ボイスレコーダのデジタルトラックの頭出しを行ないます。頭出し信号は、CMT=7により書き込めます。
- ＜回数＞は-128から127までの整数値で指定し、値が正であれば早送り方向(FF)で、値が負であれば巻戻し方向(REW)の指定になり、指定回数の頭出し信号を見つけたのちに停止します。
- ＜回数＞の指定を省略すると1が指定されたものとします。
- 頭出し信号が見つからなかった場合、テープの終了まで巻取ったのちにオートストップがかかり停止状態(STOP)になります。ただし、巻戻し方向の場合init "CMT："文による＜オートプレイ＞の設定により再生状態になることがあります。

例

apss ……　現在位置より早送りで1回目の頭出し信号を見つけます。

apss -3 ……　現在位置より巻戻しで3回目の頭出し信号を見つけます。

参 考

init "CMT:" …… ボイスレコーダの機能設定。

cmt=(ステートメント) …… ボイスレコーダの制御をする。

LIMIT （リミット）

BASICエリアの制限を行ないます。

書 式　limit {〈アドレス〉 / max}

省略形　lim.

説 明

- BASICで使用するエリアを制限して自由に使用できるエリアを確保します。
- <アドレス>を指定すると、BASICはその直前までのメモリエリアを使用し、それ以後のメモリはBASICの管理外になりますので、メモリを自由に書き換えて、機械語プログラムを作成することもできます。
- <アドレス>は、**&H4000～&HFFFF**までの範囲で指定できますが、実際には、8キロバイト単位のメモリブロックで割り当てますので、指定値を8キロバイト単位で切り下げて設定します。たとえば**&HE000～&HFFFF**までの指定は、**&HE000**として設定します。
- BASICで全てのメモリを使用する場合は、<アドレス>の指定を**max**とします。
- この命令の実行時にBASICでの未使用のエリア（フリーエリア）が少ない場合は、エラーになります。
- <アドレス>は、整数型データで指定しますので**&H8000～&HFFFF**までは、10進数では**-32768～-1**になりますが、**32768～65535**で指定することもできます。

例

```
limit &HD000
```
… **&HD000**からの指定ですが、実際には**&HC000**からのエリアが自由に使用できます。

```
limit max
```
… すべてのエリアをBASICが使用します。

PEEK （ピーク）

メモリよりデータを読み出します。

書 式

peek(〈アドレス〉) …(1)

peek@(〈ブロック番号〉,〈オフセット〉) …(2)

説 明

- メモリより1バイトのデータを読み出す関数です。
- 書式(1)は機械語エリア使用時のメモリマップによる＜アドレス＞を指定しての読み出しです。
- 書式(2)は＜ブロック番号＞に8kバイトのメモリブロック番号と、ブロック内のアドレスに相当する＜オフセット＞を指定しての読み出しです。
- ＜アドレス＞は-32768～65535(&H0～&HFFFF)、＜ブロック番号＞は0～63(&H0～&H3F)、＜オフセット＞は0～8191(&H0～&H1FFF)の値で指定します。
- 読み出されるデータは8ビットのデータとなり、0～255(&H0～&HFF)の値になります。
- ＜アドレス＞は、整数型データで指定しますので、10進数では-32768～32767の値になりますが、この命令では32768～65535の値で指定することもできます。32768～65535は、整数型データの-32768～-1(&H8000～&HFFFF)に対応します。

例

```
A=peek(&HFF00)
```

…… メモリの&HFF00番地のデータを変数Aに代入します。

参 考

poke …… メモリへのデータの書き込み。

POKE （ポーク）

メモリにデータを書き込みます。

書　式

poke 〈アドレス〉,〈データ〉[,〈データ〉]…　…(1)
poke@〈ブロック番号〉,〈オフセット〉,〈データ〉[,〈データ〉]…　…(2)

省略形

po.

説　明

- メモリに1バイト単位のデータを書き込みます。
- 書式(1)は機械語エリア使用時のメモリマップによる＜アドレス＞を指定しての書き込みです。
- 書式(2)は＜ブロック番号＞に8Kバイトごとのメモリブロック番号と、ブロック内のアドレスに相当する＜オフセット＞を指定しての書き込みです。
- ＜アドレス＞は-32767～65535(&H0～&HFFFF)、＜ブロック番号＞は0～63(&H0～&H3F)、＜オフセット＞は0～8191(&H0～&H1FFF)の値で指定します。
- ＜アドレス＞は、整数型データで指定しますので、10進数では-32768～32767の値になりますが、この命令では32768～65535の値で指定することもできます。32768～65535は、整数型データの-32768～-1（&H8000～&HFFFF)に対応します。
- ＜データ＞は8ビットの範囲内の0～255(&H0～&HFF)までの値で指定します。また、＜データ＞を並べて書くと順次＜アドレス＞または＜オフセット＞を増やしながら＜データ＞を書き込みます。ただし、範囲をこえる＜アドレス＞または＜オフセット＞となるときは、エラーになります。
- この命令は内部のメモリを直接書き換えますのでclear文で指定された機械語エリア、またはV-RAMなどのメモリブロックに対して行なってください。

例

```
poke@ &H20,&H1000,&HFF
```

…… メモリブロック&H20のオフセット&H1000のメモリに&HFFのデータを書き込みます。

参　考

peek …… メモリの読み出し。
limit …… 機械語エリアの設定。

DEF USR （ディファイン ユーザ・define user）

機械語プログラムの開始アドレスを定義します。

書　式

def usr〔〈番号〉〕＝〈開始アドレス〉

説　明

- 機械語プログラムを関数として呼び出すusr0～usr9の関数に機械語開始アドレスを定義します。
- ＜番号＞は0～9までの整数で指定し、usr0～usr9の10種の定義ができます。＜番号＞を省略すると、usr0になります。
- ＜開始アドレス＞はusr関数を使用したときに実行される機械語プログラムの実行開始アドレスを-32768～65535（&H0～&HFFFF）の値で指定します。
- ＜開始アドレス＞は、整数型データで指定しますので、10進数では-32768～32767の値になりますが、この命令では-32768～65535の値で指定することもできます。32768～65535は、整数型データの-32768～-1（&H8000～&HFFFF）に対応します。
- 定義された関数は、その関数名で使用すると定義された＜開始アドレス＞より機械語プログラムをサブルーチンとして実行します。機械語プログラムの終了は、機械語プログラム内のRET（リターン）命令になります。
- 機械語プログラムはlimit文で指定した機械語エリア内で作成してください。

例

```
def usr=&HF000
```

…… usr0の開始番地を&HF000に指定します。

参　考

peek、poke …… メモリの読み出し、書き込み。

call …… 機械語プログラムの実行。

limit …… 機械語プログラムのエリア設定。

USR　(ユーザ・user)

機械語プログラムを関数として呼び出します。

書　式　usr〔〈番号〉〕(〈引数〉)

説　明

- def usr文で定義された機械語プログラムに<引数>を与えて実行し、機械語プログラムより渡されたデータがこの関数の値になります。
- <番号>は0～9の値で指定し、usr0～usr9になり、<番号>を省略するとusr0になります。
- <引数>は機械語プログラムへ渡すデータになり、データの型がCPUレジスタのBレジスタへ、データの記憶されているアドレスがCPUレジスタのHLレジスタにセットされます。
- この関数の結果は、機械語プログラムでCPUレジスタのBレジスタとHLレジスタに渡された情報によるデータになります。機械語プログラムで、これらのCPUレジスタを変更していない場合は<引数>の値がそのまま結果になります。
- 機械語プログラムに渡されるデータは次のようになります。

データ型	Aレジスタ	データの並び（HL←先頭アドレス）
整　数	2	上位 | 下位
単精度	5	指数 | 仮　数　部
倍精度	8	指数 | 仮　数　部
文字列	3	長さ | L | H

（文字列型のL, Hは、文字列が格納されているアドレスを示します。）

- 機械語プログラムに渡される情報は、ほかにCPUレジスタのIXに、エラー時の戻り先アドレスがセットされます。機械語プログラムでエラーにするときは、CPUレジスタのAにエラーコードを設定してJP(IX)を実行するとBASIC内のエラーコードとして扱われます。

例

```
A$=usr("ABC")
```

…… "ABC"を引数としてUSR0で定義されている機械語サブルーチンを実行し、結果を変数A$に代入します。

参　考　def usr …… 機械語関数の定義。

CALL （コール）

機械語サブルーチンを呼び出します。

書 式　call 〈アドレス〉[(〈変数〉)]

省略形　ca.

説 明

- 指定された<アドレス>から機械語サブルーチンとして実行します。このときに機械語サブルーチンへデータを渡すことができます。
- <アドレス>は-32768～65535(&H0～&HFFFF) の値を変数または定数で指定します。
- <変数>は機械語プログラムに渡すデータで、定数を使用することはできません。
- 機械語プログラムへはCPUのAレジスタへデータの有無(1…あり、0…なし) HLレジスタにデータが記憶されている先頭アドレス、Bレジスタにデータの型の情報が入り、指定されたアドレスからの実行になります。
- 機械語プログラムより、データを返す場合は機械語プログラム内で、BASICより渡されたデータを書き換えることにより行なわれ、BASICに戻ったときに<変数>に値が返ります。
- 機械語プログラムはlimit文で機械語エリアを設定してそのエリア内に作成してください。
- field文でフィールド変数として指定されている文字型変数をこの命令で使用するとフィールド変数の機能が解除されますので注意してください。

例

```
call &HF000
```

…… &H F000番地より機械語サブルーチンを実行します。

参 考　peek、poke …… メモリの読み出し、書き込み。
def usr、usr …… 機械語プログラムを関数としての定義と実行。
limit …… 機械語プログラムエリアの設定。

INP@ （インプットアットマーク・input@）

I/Oポートよりデータを入力します。

書　式　inp@ 〈ポートアドレス〉,〈数値変数〉

説　明

- CPUのI/Oアドレスを＜ポートアドレス＞として、入力した値を＜数値変数＞に代入します。
- ＜ポートアドレス＞は16ビットで、-32768～32767 の整数型データで指定します。また、32768～65535（&H8000～&HFFFFに対応）で指定することもできます。
  結果は8ビットで0～255（&H0～HFF）の値が返ります。
- この命令はコンピュータ内部のI/Oポートより直接入力する命令です。

例

```
inp@ &H10,A
```

…… ポートアドレス&H10により入力したデータを変数Aに代入します。

OUT@ （アウトプットアットマーク・output@）

I/Oポートにデータを出力します。

書　式　out@ 〈ポートアドレス〉,〈数式〉

説　明

- CPUのI/Oドレスを＜ポートアドレス＞として＜数式＞の値を出力します。
- ＜ポートアドレス＞は16ビットで、-32768～32767 の整数型データで指定します。また、32768～65535（&H8000～&HFFFFに対応）で指定することもできます。
- この命令はコンピュータ内部のI/Oポートを直接操作しますので、コンピュータ内部で使用されている場所に不用意に出力するとコンピュータが正常に動作しなくなるので注意が必要です。

例

```
out@ &H20,asc("A")
```

…… ポートアドレス&H20に"A"の文字コード(65)を出力します。

SIZE （サイズ）

メモリの未使用量を求めます。

書 式　size(〈機能〉)

説 明

- プログラム、変数、配列、プリンタスプーラやRAMファイルなどで使用されていないメモリ量をバイト数で求める関数で ＜機能＞ により次の値が返ります。
  - 0 … フリーエリアのサイズ。
  - 1 … フリーエリアの中でプログラム用として使用可能なサイズ。
  - 2 … フリーエリアの中で数値配列用として使用可能なサイズ。
  - 3 … フリーエリアの中で文字列用として使用可能なサイズ。
  - 4 … フリーエリアの中でRAMファイルやプリンタスプーラ用として使用可能なサイズ。
- フリーエリアは各種の目的に使用可能な未使用のエリアです。＜機能＞の1～4はそれぞれの目的だけで使用できる未使用のサイズを示しますので、それぞれの値や合計は、フリーエリアとは一致しません。

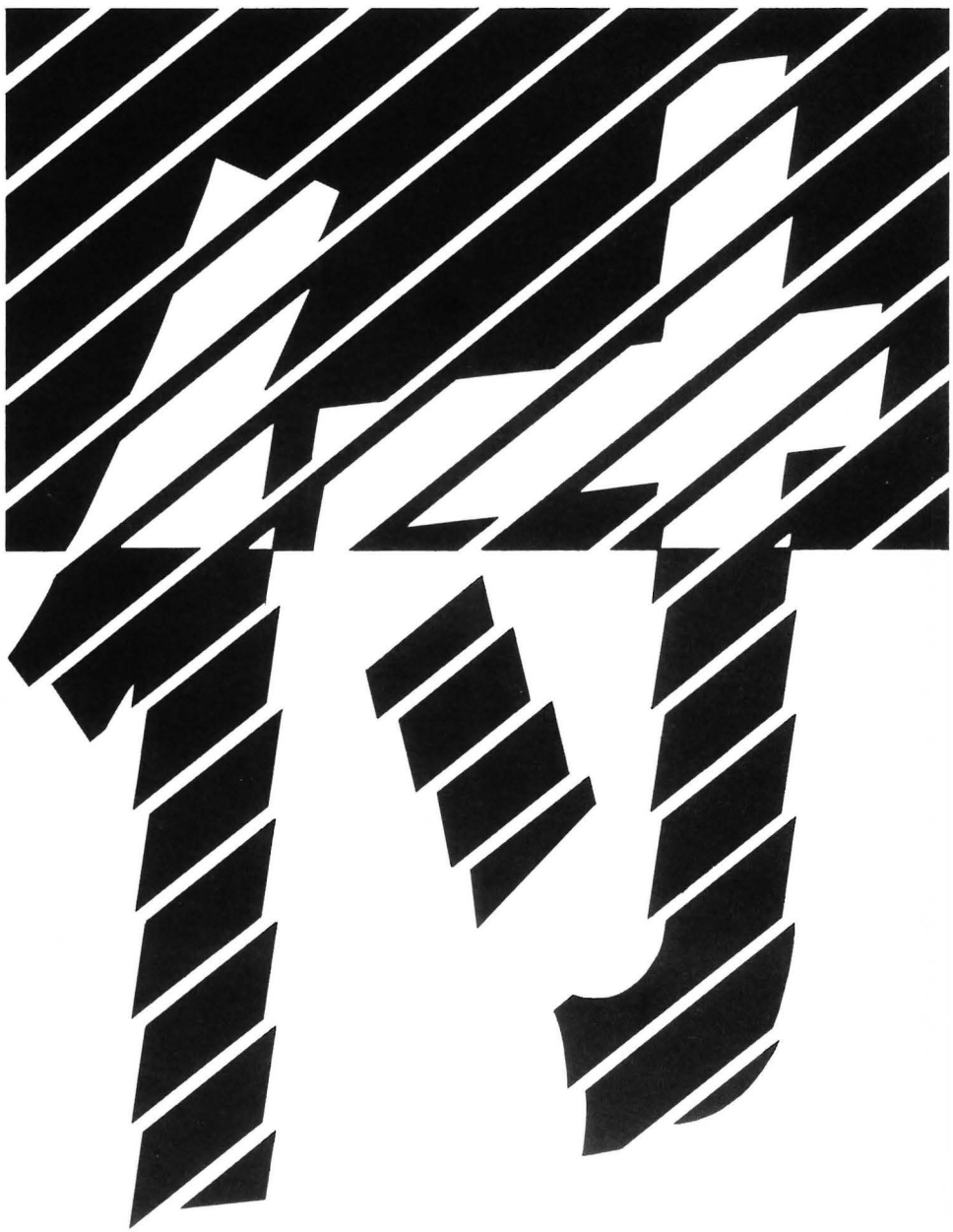
例

```
print size(2)
```

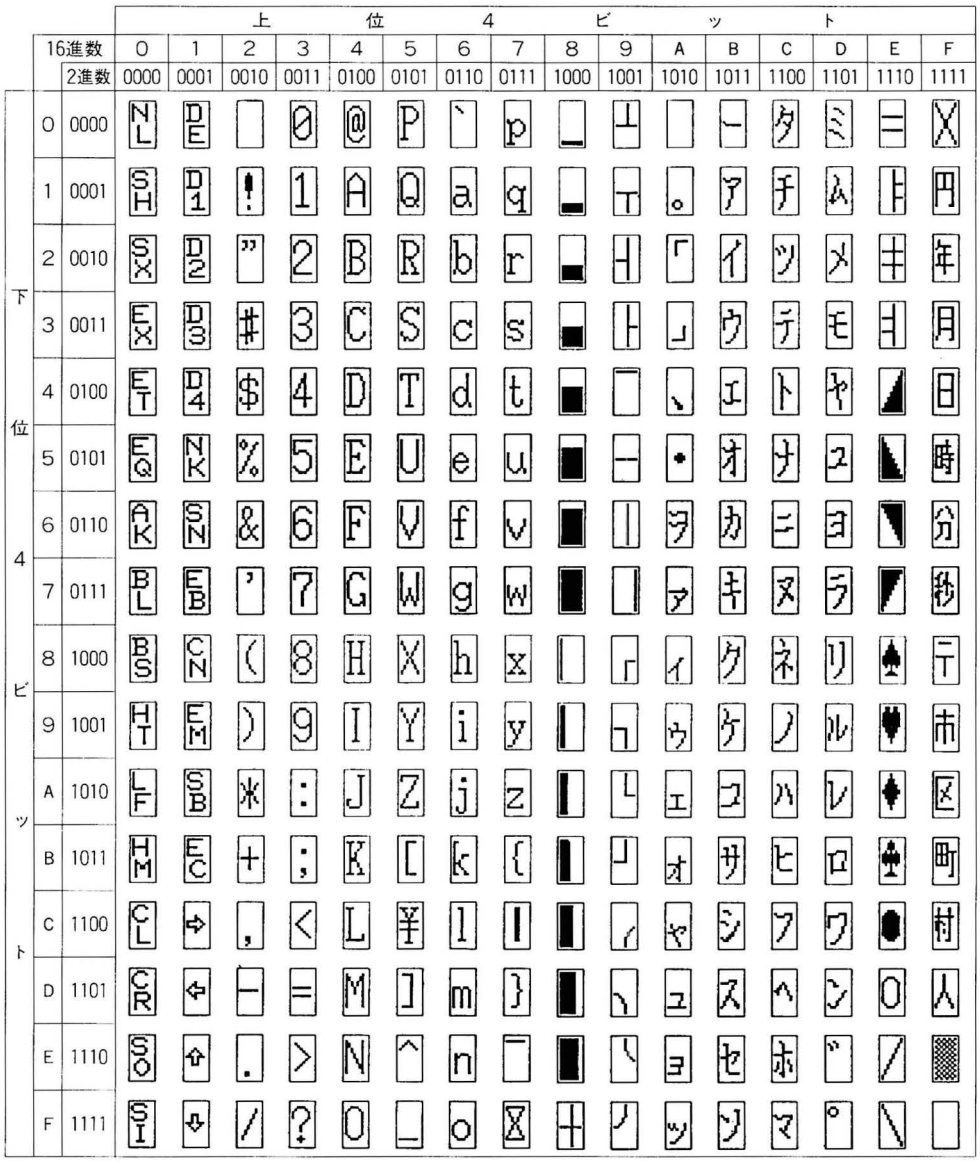
…… 数値変数や配列に使用できるサイズを表示します。

参 考☞　BASICのフリーエリア（BASIC-M25マニュアル付録）

付 録



文字コード表 （半角文字）



- 文字コードが**&H81～&H9F, &HE0～&HFC**はシフトJISコードの1バイト目になります。半角文字で使用する場合は**kmode 0**として漢字モードを解除してください。
- 文字コードが**&H1F**以下の文字はコントロールコードになりますので、通常の**print**文ではコントロールが実行されます。コントロールコードを画面に表示する場合は、**"CRT1:CG"**のファイル名でシーケンシャルアクセスのファイルオープンを行ない、**print#**文で書き込みを行なってください。

エラーメッセージ

BASICの動作中にエラーが発生した場合、次の形式でコンピュータよりディスプレイ上にエラーメッセージが表示されます。

エラーの表示

××××　error　　…直接実行モード(ダイレクトモード)

××××　error　in　nn　　…プログラム実行中

××××はエラーの内容、nnはエラーの発生行になります。

ファイル処理や周辺装置を使用している場合は、その装置名やファイル指定を左側に表示する場合があります。

エラー番号	表示メッセージ	エラーの内容
1	syntax error	文法上の誤りがある
2	over flow error	演算上で数値が扱える範囲をこえた(0での除算など)
3	data error	指定データが規定外である
4	type mismatch error	指定のデータ型が異なる
5	string length error	文字データの長さが255をこえた
		指定の文字データが規定長になっていない
6	no memory error	メモリ容量不足となった
7	w-def. FN error	定義関数を2重に定義しようとした
8	line length error	プログラムの1行の長さが制限をこえた
9	unprintable errer	未定義
10	IF-ENDIF mismatch error	ifとend ifが対応していない
11	FOR-NEXT mismatch error	forとnextが対応していない
12	WHILE-WEND mismatch error	whileとwendが対応していない
13	REPEAT-UNTEL mismatch error	repeatとuntilが対応していない
14	GOSUB-RETURN mismatch error	gosub文で呼ばれていないのにreturnしようとした
15	undef. array/FN error	定義されていない関数や配列を指定した
16	undef. line error	存在しない行番号またはラベルを指定した
17	can't CONT error	プログラムの実行再開が不可能なとき再開しようとした
18	memory protection error	使用できないアドレスのOBJファイルを読み込もうとした
19	Instruction error	ダイレクトコマンドとステーメントを混同した
		現在の状態では使用できないモードを指定した
20	sequence error	読み込もうとしたプログラムファイルの書式がおかしい

エラー番号	表示メッセージ	エラーの内容
21	RESUME error	エラー処理中でないのにresumeしようとした
22	count error	関数の引数や入力変数の個数が合わない
23	subscript error	配列の指定が不適当 配列の大きさが64Kバイトをこえた
24	READ-DATA mismasch error	対応するdata文がないのにreadしようとした
25	SWAP nesting error	スワップレベルが 1 をこえた(BASIC-S25)
26	too complex error	式が複雑で一度に処理できない
27	too many nesting error	ループまたはif文、サブルーチンなどのネスティングが多い
28	system id error	BASIC以外のディスクを使用した
29	framing error	RS-232C フレーミングエラー
30	over run error	RS-232C オーバーランエラー
31	parity error	RS-232C パリティエラー
32	unprintable error	未定義
33	unprintable error	
34	unprintable error	
35	unprintable error	
36	unprintable error	
37	unprintable error	
38	unprintable error	
39	unprintable error	
40	not found error	指定ファイルがみつからない
41	hardware error	ディスクのハードウェアトラブルが起こった
41	already exist error	すでに登録されているファイル名を使用した
43	already open error	すでにオープンされているファイルを指定した
44	not open error	オープンされていないファイルを使用した
45	not empty error	消去しようとしたディレクトリにまだファイルが残っている
46	write protect error	プロテクトされたファイルまたはディスクを書き換えようとした
47	field error	フィールド定義のできないファイルを指定した フィールドの合計がレコード長をこえた
48	open mode error	オープンされたファイルに対するアクセス型式が合わない
49	unprintable error	未定義
50	not ready error	指定装置が可動状態になっていない
51	too many files error	ファイルの登録数が制限をこえた

エラー番号	表示メッセージ	エラーの内容
52	disk mismatch error	ディスクの記録内容が異常(ディスクがクラッシュした)
53	no file space error	ファイルを作成するスペースがなくなった
54	unformat error	フォーマットされていないディスクを使用した
55	file length error	ファイルの大きさが制限をこえた
56	unprintable error	未定義
57	unprintable error	
58	device name error	装置名の指定が不適当
59	can't execute error	指定装置に使用できないファイル型式で指定した 実行できない命令の使用または実行できないモードを指定した
60	file name error	ファイルの指定が不適当
61	file mode error	指定されたファイルの型式が目的に合わない
62	unprintable error	未定義
63	file end error	ファイルが終了しているのに読み込もうとした
64	logical number error	ファイル番号の指定がおかしい
65	LPT：not ready error	プリンタが可動状態でない(PW-off,off line)
66	unprintable error	未定義
67	unprintable error	
68	device mode error	デバイスモード エラー
69	unprintable error	未定義
70	check sum error	カセットのチェックサム(ビット合計)が合わない
71 | 255	unprintable error	未定義

索　引（アルファベット順）

<a>

<b>

<c>

<d>

シャープ㈱株式会社

本　　社　〒545 大阪市阿倍野区長池町22番22号
電話(06)621−1221(大代表)
情報システム事業本部　〒639-11 奈良県大和郡山市美濃庄町492番地
電話(07435)3−5521(大代表)
国内情報システム営業本部　〒545 大阪市阿倍野区長池町22番22号
電話(06)621−1221(大代表)

お客様ご相談窓口

札　幌	(011)642-4649	仙　台	(0222)88-9141	宇都宮	(0286)35-1155
東　京	(03)893-4649	金　沢	(0762)49-4649	名古屋	(052)322-4649
大　阪	(06)643-4649	広　島	(082)874-4649	高　松	(0878)33-4649
福　岡	(092)572-4649	沖　縄	(0988)62-2231		

シャープエンジニアリング㈱株式会社

本　　社	〒114 東京都北区東田端2丁目13番17号	電話(03)800-1221(代表)
札幌支店	〒063 札幌市西区24軒1条7丁目3番17号	電話(011)641−4649
仙台支店	〒983 仙台市卸町東三丁目1番27号	電話(0222)88−9141
宇都宮支店	〒320 宇都宮市不動前4丁目2番41号	電話(0286)35−1155
東京支店	〒114 東京都北区東田端2丁目13番17号	電話 (03)800−1221
金沢支店	〒921 石川県石川郡野々市町字御経塚1096番地の1	電話(0762)49−4649
名古屋支店	〒454 名古屋市中川区山王3丁目5番5号	電話(052)332−2626
大阪支店	〒556 大阪市浪速区恵美須西1丁目2番9号	電話 (06)643−4649
広島支店	〒731-01 広島市安佐南区西原2丁目14番地の3	電話(082)874−2281
高松支店	〒760 高松市木太町1861番地の3	電話(0878)33−4649
福岡支店	〒816 福岡市博多区井相田2丁目12番地の1	電話(092)572−4655
沖縄シャープ電機㈱	〒900 那覇市曙2丁目10番地の1	電話(0988)62−2231

シャープビジネス㈱株式会社

本　　社	〒545 大阪市阿倍野区長池町22番22号	電話(06)621-1221(大代表)
札幌支店	〒063 札幌市西区24軒1条7丁目3番17号	電話(011)641−3631
仙台支店	〒983 仙台市卸町東三丁目1番27号	電話(0222)88−9151
東京支店	〒130 東京都墨田区石原2丁目12番3号	電話 (03)625−5111(代表)
千葉支店	〒280 千葉市南町1丁目5番20号	電話(0472)63−4043
横浜支店	〒235 横浜市磯子区中原1丁目2番23号	電話(045)751−3215
埼玉支店	〒330 大宮市宮原町2丁目107番地2号	電話(0486)63−5159
宇都宮支店	〒320 宇都宮市不動前4丁目2番41号	電話(0286)37−3576
新潟支店	〒950 新潟市上所中1丁目7番21号	電話(0252)83−1795
長野支店	〒380 長野市中御所2丁目11番3号	電話(0262)28−4618
名古屋支店	〒454 名古屋市中川区山王3丁目5番5号	電話(052)332−2631(代表)
金沢支店	〒921 石川県石川郡野々市町字御経塚1096番地の1	電話(0762)49−1240
大阪支店	〒556 大阪市浪速区恵美須西1丁目2番9号	電話 (06)643−3021(代表)
京都支店	〒601 京都市南区上鳥羽菅田町48番地	電話(075)661−7335
神戸支店	〒658 神戸市東灘区魚崎北町1丁目6番地18号	電話(078)452−8531
広島支店	〒731-01 広島市安佐南区西原2丁目14番地の3	電話(082)874−4925
高松支店	〒760 高松市木太町1861番地の3	電話(0878)33−4255
福岡支店	〒816 福岡市博多区井相田2丁目12番地の1	電話(092)572−2611
沖縄支店	〒900 那覇市曙2丁目10番地の1	電話(0988)61−7360(代表)

シャープ株式会社

5K 14.5-I(TINSJ1314ACZZ)②