Personal Computer

MZ-1500

BASIC LANGUAGE MANUAL



SHARP

Personal Computer

MZ-1500

BASIC LANGUAGE MANUAL

(BASIC解説書)

©1984 SHARP CORPORATION

ご注意

本書は、パーソナルコンピュータ MZ-1500 のシステム・ソフトウェア BASIC（5Z001）に基づいて作成されています。

（1）多目的パーソナルコンピュータ MZ-1500 では、システムソフトウェアをすべてファイル形態のクィックディスクなどによってサポートされます。各システムソフトウェアおよび本書の内容は、改良のため変更することがありますので、ファイルバージョンナンバーには特にご注意されるよう、お願い致します。

（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ください。

（3）パーソナルコンピュータ MZ-1500 の BASIC（5Z001）のソフトウェアは、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。

BASIC（5Z001）の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。

目　　次

第1章　入門編……1
1．さあ、コンピュータの世界へ出発しよう！……2
2．BASICの世界へ……3
3．コンピュータに仕事をさせる！……4
(1)　プログラムとは……4
(2)　コンピュータに仕事をさせる順序……5
4．プログラムの種類によるMZ-1500の使い方……6
5．BASICを覚えさせよう……7
6．デモンストレーションプログラムを走らせてみよう……9

第2章　基本編……11
1．数値データと文字データ……12
(1)　数値データ……12
(2)　文字データ……12
2．定数、変数、配列……13
(1)　定数……13
(2)　変数……14
(3)　配列変数……16
3．式……17
(1)　算術式……18
(2)　文字式……20
(3)　関係式……20
(4)　論理式……21
4．関数……22
(1)　数値関数……22
(2)　文字関数……24
(3)　数値・文字関数……26
(4)　ランダム関数……28

第3章　ソフトウェア編……29
1．動作モード……30
(1)　2つの動作モード……30
(2)　直接モード……30
(3)　間接モード（プログラムモード）……33
2．プログラムを使おう！……34
(1)　プログラムを入力してみよう……35
(2)　プログラムの入出力について……36
(3)　プログラムを実行しよう……37
(4)　プログラムを編集・修正しよう（テキスト編集命令）……38
3．基本命令……43
(1)　データは変数と呼ばれる箱に入れよう（代入文）……43
(2)　データを表示しよう（PRINT 命令）……44
(3)　データを入力しよう（INPUT 命令）……47
(4)　プログラムの流れを変えよう（GOTO 命令、GOSUB 命令）……48
(5)　条件を判断しよう（IF～THEN～：ELSE 命令）……51
(6)　くり返してみよう（FOR～NEXT 命令）……55
(7)　ON～GOTO 命令、ON～GOSUB 命令……57
(8)　READ～DATA 命令……58
(9)　注釈をつけよう（REM 命令）……59
(10)　デファイナブルキーの使い方……60
4．応用編……61
(1)　画面表示……62
(2)　初期設定……63
(3)　画面に応じた表示の手順……65
(4)　色……66
(5)　座標……68
5．ファイル……69
(1)　ファイルの分類……69
(2)　周辺機器への入出力……69
(3)　ファイルについて……70
(4)　ファイルの使い方……73
(5)　WOPEN と ROPEN……74
6．音楽を演奏する……75

(1) ステレオを楽しむ……76
(2) 音楽の演奏……77
(3) 音を作る……79
7．プリンタを使う……80
(1) 2つのモード……80
(2) 使用プリンタによるモードの使いわけ……80
(3) 各モードの設定命令……81
(4) アスキーコード表の見方……82
8．配列……83
9．乱数……85
10．ジョイスティック……87
11．通信……88
12．コンピュータの内部を扱う……90
(1) 機械語を使う……90
(2) BASICから機械語へのアプローチ……91

第4章　命令・ステートメント……93
1．基本命令……94
AUTO……94
DELETE……95
LIST……96
SEARCH……97
RENUM……98
NEW……99
NEW ON……100
CLR……101
CONT……102
RUN……103
CLS……104
CONSOLE……105
CURSOR……106
REM……107
LET……108
STOP……109

END ……110
FOR～NEXT ……111
LABEL……113
GOTO ……114
ON～GOTO……115
GOSUB～RETURN ……116
ON～GOSUB ……117
IF～THEN～ELSE ……118
IF～GOTO ……119
IF～GOSUB ……120
PRINT ……121
PRINT USING ……123
INPUT ……127
GET ……128
DIM ……129
READ～DATA ……130
RESTORE ……131
DEF FN ……132
TRON ……133
TROFF ……134
DEF KEY ……135
KEY LIST ……136
INIT ……137
PRTY ……143
BYE ……144
BOOT ……145
WAIT ……146
2．ファイル命令 ……147
LOAD ……147
SAVE ……148
LOAD ALL ……149
SAVE ALL ……150
RUN ……151
VERIFY ……152

CHAIN ……153
MERGE ……154
WOPEN # ……155
ROPEN # ……156
PRINT # ……157
INPUT # ……158
DIR ……159
EOF (#) ……160
CLOSE # ……161
RENAME ……162
KILL # ……163
DELETE ……164
DEFAULT ……165
3．グラフィック命令 ……166
COLOR ……166
CCOLOR ……168
PAL ……169
SET ……171
RESET ……173
CSET ……175
CRESET ……176
LINE ……177
BLINE ……179
BOX ……181
CIRCLE ……182
PAINT ……184
PATTERN ……185
POSITION ……187
KPATTERN ……188
FONT$ ……189
FPRINT ……192
POINT ……193
4．ミュージック命令 ……194
MUSIC ……194

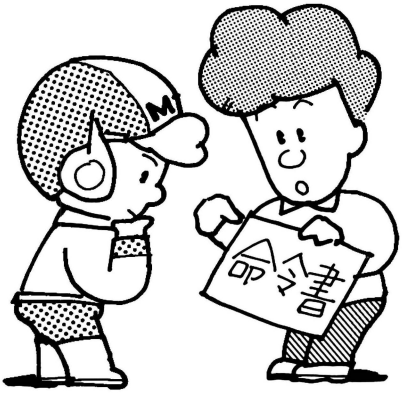
第1章　入　　門　　編

3．コンピュータに仕事をさせる！

コンピュータに仕事をさせるには、まず、コンピュータに何をさせるのかという目的をはっきりさせ、次に目的に合わせてプログラムを作らなければなりません。

（1）プログラムとは

私達が何か仕事をする場合、今までの経験に基づいて考えたり、作業指示書などにしたがって仕事をします。コンピュータの場合、コンピュータ自身が考えて何か仕事を処理するということはなく、あくまでもあなたの指示命令によって処理します。コンピュータに何をどのように実行させるのか、という命令書（仕事の手順書）のことをプログラムといいます。



プログラムは、多くの命令の組み合わせでできており、1つ1つの命令のことを文（ステートメント）といいます。命令をどのような順番に実行していくかをアルゴリズムといい、アルゴリズムに基づいて文を組み合わせたものがプログラムです。（プログラムを作ることをプログラミングといいます。）

※アルゴリズムの考え方は、どのようなプログラム言語でも似たようなもので、プログラムの基本といえます。

（2）コンピュータに仕事をさせる順序（プログラミング）

❶コンピュータにさせたいことがある。



❷させたいことの内容を調べて、プログラムを考える。



❸プログラムを作る。

※アプリケーションプログラムが用意されている場合は、作らなくてもよい。



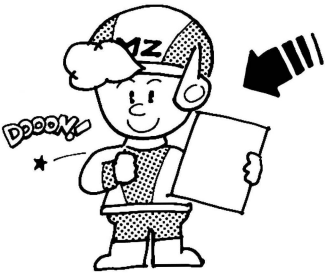
❹プログラムをコンピュータに実行させる。



・プログラムをキーボードで入れる。

⇩

・プログラムが入ったら、キーボードで"実行しなさい"という命令を入力する。



❺コンピュータが仕事をする。

4．プログラムの種類による MZ-1500 の使い方

プログラムを使って、コンピュータに仕事をさせる場合、自分でプログラムを作るのか、アプリケーションプログラム（市販されているプログラムや MZ-1500 に同梱されているデモンストレーションプログラム）を使用するのかで使い方は異なってきます。

〈BASIC を起動する〉（7 ページ参照）



BASIC のプログラムを自分で作る

BASIC のアプリケーションプログラムを使う





・第 4 章をよくお読みください。

プログラムとして使う間接モード

（33ページ参照）

電卓的な使い方直接モード

（30ページ参照）

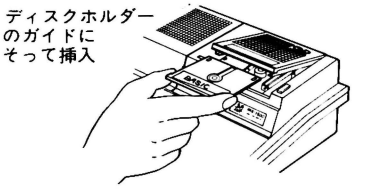
5． BASIC を覚えさせよう

最初は、コンピュータの頭のなかには、なにも記憶されていません。したがって、電源スイッチを入れてもすぐには働かせることはできません。まずスイッチを入れて、次にコンピュータを働かせるための BASIC という言葉を覚えさせてやることが必要です。これは、私達が英語で会話する場合に、まず英語を覚えるのと同じことです。

BASIC の起動方法

❶別冊、「初めてお使いになる人のために」の説明にしたがってディスプレイ（テレビ）を接続してください。

❷コンピュータ本体の後面にある電源スイッチを入れる前にクイックディスクに BASIC の入ったディスクをセットしてください。

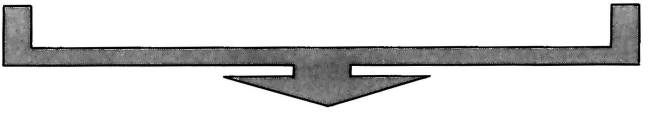


BASIC の入ったディスクを本体にセットしないで電源を ON にするとメッセージが表示されます。



❸コンピュータ本体の電源スイッチを ON にしてください。
クイックディスクの BUSY ランプが点灯します。

クイックディスクに BASIC の入ったディスクをセットした後、メッセージにしたがって入力してください。Q を入力するとクイックディスクの BUSY ランプが点灯します。



❹ BASIC がコンピュータに読み込まれ、画面に次ページの内容が表示されます。

これでコンピュータが BASIC 言語を覚えました。それでは，次にキーを押してみましょう。
キーボードの使い方については別冊の「オーナーズマニュアル」をご覧ください。

```
BASIC interpreter   MZ-5Z001 VX.XX
 Copyright (C) 1984 by SHARP CORP.

 XXXXX bytes free
Ready
```

●プログラムファイル" AUTO RUN "の実行

" AUTO RUN "の実行が前記の働きのうちに含まれています。BASICが読み込まれ、プログラムエリアの大きさを示すバイトサイズが表示されると、ふたたび、クイックディスクがアクセスされます。これはBASICの読み込みを終えたとき，自動的に、

```
RUN "AUTO RUN"
```

コマンドを実行するからです。すなわち、" AUTO RUN "というファイル名のプログラムをクイックディスクから読み出し、そのプログラムの先頭から実行しているのです。クイックディスク上には、デファイナブルファンクションキーを定義するプログラムをこのファイル名で登録しています。また、このプログラムの最後に、NEW文を置いているので実行後、" Ready "を表示しカーソル点滅となる前にプログラムは消去されているのです。(一度、LOAD " AUTO RUN "を実行してリストを調べてみてください。)

もし、BASIC読み込みに続いて、あるプログラムをスタートさせたい場合、そのプログラムのファイル名を" AUTO RUN "としてクイックディスクに書き込んでおけばいいのです。

6．デモンストレーションプログラムを走らせてみよう

コンヒュータは、BASICという言葉を覚えただけではなにもしてくれません。今度はあなたが、BASICを使ってコンヒュータに命令を与える必要があります。

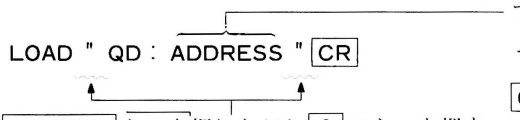
命令の使い方や、詳しい説明は「第3章」、「第4章」を見ていただくとして、ここではデモンストレーションフログラムを使って本機を動かしてみることにしましょう。

デモンストレーションフログラムは、MZ-1500に同梱されているディスクの中に入っています。

それでは、次の手順にしたがって本機を動かしてみてください。

①7ページの手順で読み込ませたBASICのディスクを裏返しにして、クイックディスクに入れてください。

次にキーボードを使って



とキー入力してください。

②フログラムの読み込みが終わると画面にはReadyと表示されカーソル（■）が点滅します。

③デモンストレーションフログラムがコンヒュータの頭のなか（メモリ）に入りました。これから実行させてみましょう。

RUN CR とキー入力すると実行します。

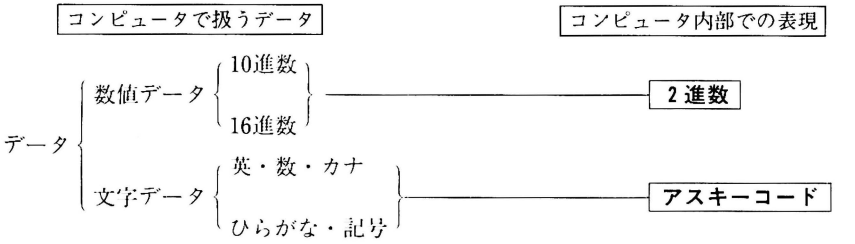
④フログラムの実行を中断させるには SHIFT キーを押しながら BREAK キーを押してください。画面にはBreakと表示され、続いてReadyの表示が出てカーソルが点滅します。

⑤再びフログラムを実行させたいときは、RUN CR とキー入力します。

第2章　基　　本　　編

1．数値データと文字データ

コンピュータで扱うデータは、数値データと文字データに分けられます。



（1）数値データ

BASICでは、私達が普通使っているのと同じように、数字を使うことができます。ただし、コンピュータで数字を扱う場合、コンピュータ内部の情報は、すべて電気信号のON･OFFによって表されています。そこで、コンピュータ内部では、"0"と"1"だけで表示できる2進数が使われています。

また、10進数として扱うケタ数の最大値を8ケタと定めています。これを、有効ケタといいます。

（2）文字データ

文字は、コンピュータ内部では数値として扱われています。コンピュータで扱われる文字（英・数・カナ・ひらがな・記号）には、0～255までの番号が付けられ、この番号のことをアスキーコードといいます。（258ページ参照）たとえば、"A"という文字には、65という番号（＝アスキーコード）が付いているので、コンピュータの内部では、"A"という文字は、65という数値で記憶されているわけです。

なお、コンピュータの内部を扱う場合には、16進数がよく使われるため、アスキーコードも16進数として表されることも多くあります。



2．定数、変数、配列

BASICで使用するデータは、プログラム中での使い方から定数・変数・配列に区別されます。







（1）定数

・定数には数値定数・文字定数があります。

数値定数は有効ケタ8ケタ以内で
指数範囲は10^{-38}～10^{38}です。
文字定数は最大文字数255
文字までです。

123　-12.34
1.23E+2（1.23×10^2）　} 数値指定
"MZ-1500" "シャープ"　文字定数

●数値定数

数値定数は、符号・数字・小数点・指数（E）で表されます。整数や小数の表し方は、数学で使うものと同じです。

[例]

１２３
－１２３．４
＋１２

指数は、小数や整数では表しにくい、大きな数などを表すときに使われ、Eで表されます。これは、Eの後にある数字の回数だけ10倍する（マイナスの場合は割る）ことを意味し、Eの前にある数値を仮数部、Eの後にある数字を指数部といいます。指数部の値の範囲は、仮数部を小数点第1位になおした場合の2桁の整数（－38～＋38）です。符号は＋の場合は省略できます。

[例]

１．２３Ｅ＋２（１．２３×１０×１０）→１２３
－１．２Ｅ－１（－１．２÷１０）→－０．１２

16進数のデータは、前にドルマーク（$）を付けて表します。

[例]

```
CHR$(＄１Ｂ)
USR （＄Ｅ０００）
```

●文字定数

文字定数は、ダブルクォーテーションマーク（"　"）で囲んで表します。文字定数の長さは最大255文字です。また、使用している行の有効長に制限されます。

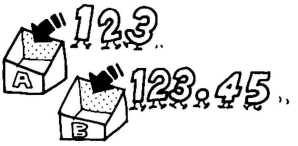
[例]

```
"ＡＢＣ"
"１２３４５"
"ＭＺ－１５００"
```

注）DATA文ではダブルクォーテーションマークを省略することができます。

（２）変数

変数は、数値データや文字データを入れるための箱にたとえられます。この箱についている名前を変数名、箱に入っている値を変数値といいます。変数値はプログラム中に必要に応じて変えることができ、また変数名を指定することによりその値を取り出すことができます。変数には数値変数・文字変数・システム変数があります。

数値変数	文字変数	システム変数
		
数値データのみを与えることができます。	文字データのみを与えることができます。	コンピュータが自分で定義している変数です。

●数値変数

数値変数の変数名は、次の規則にしたがわなくてはなりません。

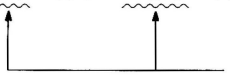
・変数名は何文字でもかまいませんが、最初の２文字だけが変数名として識別されますので、３文字以降の文字は無視されます。

●数値変数

数値変数の変数名は、次の規則にしたがわなくてはなりません。

・変数名は何文字でもかまいませんが、最初の2文字だけが変数名として識別されますので、3文字以降の文字は無視されます。

［例］ ＡＢ・ＸＹ・ＡＢＣ・Ａ１２３



同一のアルファベットの大文字と小文字は同一の変数として扱われます。

［例］ ＡＢという変数は、ａＢ、ａｂ、Ａｂすべて同じ変数です。

また、変数名は最初の1文字は必ずアルファベット（A～Z）でなければなりませんが、あとの2文字目はアルファベットと数字が使えます。

・変数名としてBASICで使っている予約語（命令や関数などの名前）は使えません。なお、データが代入されるまでは、変数値は0となっています。

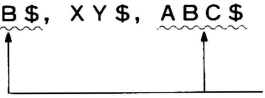
［例］ ＰＲＩＮＴ…画面に表示しなさいという命令語なので変数名には使えません。

Ｃ@……特殊文字（アルファベットと数字以外の文字）を含んでいるので変数名には使えません。

●文字変数

文字定数の変数名は、数値変数の場合と同じ規則にしたがいますが、変数名の最後にドルマーク（$）を付けて表します。文字変数の最大文字数は255文字です。なお、データが代入されるまでは、変数値はNull（"　"）となっています。（Nullとは何もない文字、すなわち文字数が0である文字変数であり、空白とは意味が違います。）

［例］ ＡＢ$，ＸＹ$，ＡＢＣ$



●システム変数

BASICを動かすために必要な変数を、コンピュータが自動的に定義しているものがあります。これをシステム変数といいます。

システム変数には下記のものがあります。

システム変数	説　　明
TI$	時刻を表す。 電源をONにしたときは TI$＝"000000"から始まります。 TI$＝"192035"ならば、19時20分35秒です。 35秒 20分 19時（午後7時）
SIZE	BASICで使用できるメモリの残りのバイト数を表す。
ERN	エラー発生時のエラー番号を表す。
ERL	エラー発生時に、エラーの発生した行の行番号を表す。
CSRH	文字画面上のカーソルのX座標を表す。 0≦CSRH≦39
CSRV	文字画面上のカーソルのY座標を表す。 0≦CSRV≦24
POSH	グラフィック画面上のポジションポインタのX座標を表す。 0≦POSH≦319
POSV	グラフィック画面上のポジションポインタのY座標を表す。 0≦POSV≦199

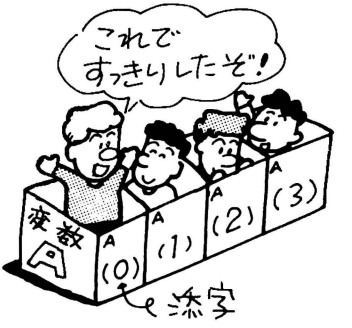
（３）配列変数

変数は、一つのデータを入れる箱でしたが、これでは多くのデータをまとめて使うときに不便です。いくつもの変数を１つにまとめて名前をつけて、取り扱おうというものが配列です。



配列は、データを入れる箱をいくつも持っていて、この箱には、他の箱と区別するため、番号がついています。この番号を添字といい、プログラム中では、（　）で囲んで表わします。

箱の数が多い場合、１つの添字だけではなく、２つの添字の組合せで表わすこともあります。添字が１つのものを１次元配列、２つのものを２次元配列といいます。また変数名の次に＂＄＂マークをつけると文字配列変数とみなされます。なお、プログラムで配列を使う場合、かならず最初にDIM文（129ページ）で、配列の名前と大きさを指定しなくてはなりません。



[例]

```
 10  DIM  A (3, 3), B$(3, 3)
 20  FOR  I=1  TO  3
 30  FOR  J=1  TO  3
 40  INPUT  "ナンバー" ; A (I, J)
 50  INPUT  "データ" ; B$(I, J)
 60  NEXT  J, I
 70  FOR  I=1  TO  3
 80  FOR  J=1  TO  3
 90  PRINT  "ナンバー" ; A (I, J)
100  PRINT  "データ" ; B$(I, J)
110  NEXT  J, I
120  END
```

上記プログラムを実行してみてください。数値配列変数の2次元配列と、文字配列変数の2次元配列を行番号10のDIM文(129ページ)で宣言しています。入力するデータの数は、数値データ、文字データともに9です。データ入力が終了すると、入力したデータの内容を全て表示します。

3．式

BASICで計算するときには式を使います。BASICの式も、数学の式と同じで、定数・変数・演算子・関数で構成されます。式には、数値データを扱う算術式、文字列データを扱う文字式、データの大小関係を扱う関係式、論理値を扱う論理式があります。

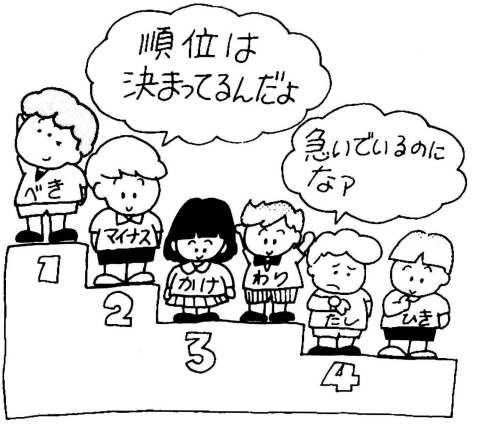
(1) 算術式

算術式は、数値データを扱う式で、演算結果として1つの数値データを与えます。算術式は、数値定数・数値変数・算術演算子・数値関数で構成されます。数値定数・数値変数については、すでに説明していますので、ここでは算術演算子と数値関数について説明します。

●算術演算子……算術演算子には、四則演算子とべき乗演算子があります。

演算子	意　味	例	優先順位
↑	べき乗	X ↑ Y	1
－	負符号	－X	2
＊	乗算(かけ算)	X ＊ Y	3
／	除算(わり算)	X／Y	3
＋	加算(たし算)	X＋Y	4
－	減算(ひき算)	X－Y	4

演算は、優先する順位にしたがって実行されます。ただし、カッコ「(　)」がある場合は、カッコ内の演算が優先されます。また、同順位の演算が続く場合は、左から右の順に計算されます。



（2）文字式

文字式は、文字データを扱う式で、演算結果として1つの文字列を与えます。文字式は、文字定数・文字変数・結合子・文字関数で構成されます。文字定数・文字変数については、すでに説明していますので、ここでは結合子と文字関数について説明します。

●結合子

算術演算子には6種類の演算子がありましたが、文字式には結合子（＋）という演算子しかありません。

[例]

```
PRINT  "ABC"+"DEF"
ABCDEF

PRINT  "A"+"B"+"C"
ABC
```

（3）関係式

数値データや文字データの大小を比較する式を関係式といいます。結果は論理値で表されます。論理値とは、真・偽の2つを表す値です。MZ-1500では真の場合は－1を、偽の場合は0を使います。関係式は、定数・変数・算術式・文字式と比較演算子とから構成されます。比較演算子には次のものがあります。

＝	等しい	X＝Y
＜	～より小さい	X＜Y
＞	～より大きい	X＞Y
＜＝	～以下	X＜＝Y
＝＜	〃	X＝＜Y
＞＝	～以上	X＞＝Y
＝＞	〃	X＝＞Y
＜＞	等しくない	X＜＞Y
＞＜	〃	X＞＜Y

（4）論理式

真・偽の2つを－1または0で表すものを論理値といいます。論理式は、論理値・関係式・論理演算子で構成されます。論理演算子には次のものがあります。

NOT	論理の否定	␣NOT X
AND	論理の積	X␣AND Y
OR	論理の和	X␣OR Y
XOR	排地的な論理の和	X␣XOR Y

(注)␣で示す部分は、必ずスペースが必要です。

文字データを比較する場合は、文字列の大小関係はアスキーコードの大小関係にしたがいます。たとえば、A＜B＜C…＜Z＜ア＜イ＜……＜ワ＜ヲ＜ンのようになります。また、比較する文字列は2文字以上でもかまいません。その場合、先頭から順に比較していきます。

4．関数

（1）数値関数

SIN や COS など、数値を計算するための関数を数値関数といいます。数値関数には次のものがあります。

関　数	BASIC の命　令	例	説　　明
絶対値	ABS (X)	A＝ABS (X)	変数 X の値の絶対値 | X | を変数 A に代入する。 (例) A＝ABS (2.9)→ A＝2.9 A＝ABS (−5.5)→ A＝5.5
符　号	SGN (X)	A＝SGN (X)	変数 X の値の正、負、ゼロを判定し、次の例のように結果を変数 A に代入する。 1 (X＞0)　　(例) A＝SGN (0.4) 0 (X＝0)　　のとき、変数 A に −1 (X＜0)　　は 1 が代入される。
整数化	INT (X)	A＝INT (X)	変数 X の値について X を超えない最大の整数を求めて変数 A に代入する。 (例) A＝INT (3.87)　→ A＝3 A＝INT (0.6)　→ A＝0 A＝INT (−3.87)→ A＝−4
平方根	SQR (X)	A＝SQR (X)	変数 X の値について、X の平方根 (X) の値を求めて変数 A に代入する。X は正、または 0 の値でなければならない。
指数関数	EXP (X)	A＝EXP (X)	変数 X の値について、e^x の値を求めて変数 A に代入する。

関　数	BASICの命　令	例	説　　明
三角関数	SIN (X)	A＝SIN (X) A=SIN (30＊π/180)	変数Xの値(ラジアン)について、sinの値を求めて変数Aに代入する。 度のデータを入れる場合、1度＝π/180ラジアンの関係がありますので30°の値のとき、30＊π/180ラジアンとなる。 同様の処理がCOS、TAN、ATNにも必要。
	COS (X)	A＝COS (X) A=COS (X＊π/180)	変数Xの値(ラジアン)について、cosXの値を求めて変数Aに代入する。 変数Xが度のとき。
	TAN (X)	A＝TAN (X) A=TAN (X＊π/180)	変数Xの値(ラジアン)について、tanXの値を求め変数Aに代入する。 変数Xが度のとき。
	ATN (X)	A＝ATN (X) A=ATN (X)＊180/π	変数Xの値について、$\tan^{-1}X$の値(ラジアン)を求め、変数Aに代入する。 計算結果は$-\pi/2$と$\pi/2$の間の値となる。 度で出力するとき。
常用対数	LOG (X)	A＝LOG (X)	変数Xの値について、常用対数logXの値を求めて変数Aに代入する。Xは正の値でなければならない。
自然対数	LN (X)	A＝LN (X)	変数Xの値について、自然対数$\log_e X$の値を求めて変数Aに代入する。 Xは正の値でなければならない。
円周率	PAI (X)	A＝PAI (X) もしくは A＝π＊X	円周率πのX倍をAに代入する。 (PAI (1)＝3.1415927)
ラジアン	RAD (X)	A＝RAD (X)	X度をラジアンに変換してAに代入する。

（2）文字関数

文字データを加工するための関数を文字関数といいます。文字関数には次のものがあります。

書　式	説　明	例
LEFT$ (x$, n) x$：文字データ n：数値データ	文字データx$について左からn文字で構成する文字列を与える。	B$=LEFT$ (A$, 2) B$ ← AB CDEFG 左から数えて2番目までの文字が、変数B$に代入される。したがって、 B$＝" AB "となる。 注意：0≦n≦255
MID$ (x$, m, n) x$：文字データ m, n：数値データ	文字データx$について、左からm番目の文字からn文字までで構成する文字列を与える。 (MID：middle)	B$=MID$ (A$, 3, 3) B$ ← AB CDE FG 左から3番目の文字を起点に、その文字から3文字分が、変数B$に代入される。したがってB$＝" CDE "となる。 注意：1≦m≦255 0≦n≦255
RIGHT$ (x$, n) x$：文字データ m, n：数値データ	文字データx$について右からn文字で構成する文字列を与える。	B$=RIGHT$ (A$, 2) B$ ← ABCDE FG 右から数えて2番目より終りまでの文字が変数B$に代入される。したがって、 B$＝" FG "となる。 注意：0≦n≦255

※例の結果は、A$＝" ABCDEFG "のとき

また、PRINT 文とともに使用される関数には次のものがあります。(ただし、FPRINT では使用できません。)

書　式	説　明	例
TAB (n) n：数値データ	現在ある行の左端より数えてn＋1文字目にカーソルを移動させる。 (タビュレーション)	? " A " ; TAB (3) " ABC " A　　ABC 0 1 2 3 4 5←(実際には表示されない。) ↑ 3 文字目から表示される。 注意：現在のカーソル位置より左側にタビュレーションが指定されたときは無効となる。
SPC (n) n：数値データ	n個の連続した空白を出力する。	? " A " ; SPC (3) ; " ABC " A　　ABC 0 1 2 3 4 5 6 ←(実際には表示されない。) └ 3 文字分の空白

（3）数値・文字関数

数値データと文字データを変換する関数には、次のものがあります。

書　式	説　明	例
STR$(n) n：数値データ	数値データnを、文字データに変換する。	A$＝STR$(－12) A$に、文字列として"－12"が代入される。 B$＝STR$(70＊33) B$に、文字列として"2310"が代入される。 C$＝STR$(1200000＊5000) C$に、文字列として"6E＋09"が代入される。 注意：正の数値の表示では正符号(＋)を省略したという意味の1つのスペースが数値の前に表示される。 STR$関数でストリングに変換されたものには、このスペースは付加されない。
VAL(x$) x$：文字データ	文字データx$を、数値データに変換する。 ただし、16進のA～Fまでの数値は変換されない。 (STR$(x)の逆の働きをする。)	A＝VAL("123") "123"という文字列が、123という数値に変換されて、変数Aに、代入される。
ASC(x$) x$：文字データ	文字データx$の先頭の文字に対応するアスキーコードを与える。	X＝ASC(" A ") キャラクタAのアスキーコード65が、変数Xに代入される。 Y＝ASC(" SHARP ") 先頭の文字"S"に対応するアスキーコード83が、変数Yに代入される。

書　式	説　明	例
CHR$(n) n：数値データ	数値データnについて、アスキーコードがXである1つのキャラクタを与える スペースとしてのCHR$（32）については PRINT "　" または PRINT　SPC（1）を使用。	**A$＝CHR$ (65)** 文字変数A$に、アスキーコード65に相当する文字" A "が代入される。 **PRINT CHR$ (107)** キーにないキャラクタ ⊠ が、画面上に表示される。 アスキーコードを続けて指定する場合は次のように指定できる。 **A$＝CHR$ (65, 66, 67, 68)**
LEN (x$) x$：文字データ	文字データx$の長さを与える。	**A＝LEN (" ABC ")** 数値変数Aに、文字列" ABC "の長さ3（3文字）が代入される。
HEXCHR$ (x$) x$：文字データ	16進数を表す文字列を、アスキーコードとみなして対応する文字列に変換する。 文字列にスペース（空白）があるときは区切りとみなし無視して次へ進む。	**A$＝HEXCHR$ (" 3441 ")** 4 A 文字変数A$にアスキーコード34と、41に相当する文字列" 4A "が代入される。 A$＝CHR$ ($34、 $41)
ASCCHR$ (x$) x$：文字データ	文字列に対応するアスキーコードを、16進数を表す文字データとして与える。 HEXCHR$の逆の働きをする。	**A$＝ASCCHR$ (" AB ")** 文字" A "に対応するアスキーコード41、文字" B "に対応するアスキーコード42が文字変数A$に代入され、 A$は4142となる。

（4）ランダム関数

数値データについて、擬似乱数を与えます。

書　式	説　明	例
RND (n) n：数値データ	・〈数値データ〉について、0.00000001から0.99999999までの値の擬似乱数(pseudo random numbers)を与える。 ・〈数値データ〉＞0の場合、BASICインタプリタが発生する擬似乱数の系列で、前に与えた乱数へ引き続く、次の乱数を与える。 ・〈数値データ〉≦0とした場合は、RND関数による擬似乱数の系列の発生を初めに戻し、先頭の擬似乱数の値を与える。この初めに戻す操作によって乱数によるシミュレーションなどに再現性を持たせることができる。	一般にNからMまでの整数の乱数を発生させる場合は、次の式を使う。 INT (RND (X)＊(M－N＋1)＋N) 乱数によって得た座標に乱数で得た半径の円をつぎつぎに描いて行くプログラムです。 10　FOR　A＝1　TO　100 20　B＝RND（1）＊320 30　C＝RND（1）＊200 40　D＝RND（1）＊100 50　E＝INT（RND（1）＊8） 60　CIRCLE［E，0］B， 　　C，D，0．5 70　NEXT　A 80　END

第3章　ソフトウェア編

本書中では、キーの入力のしかたとして次の略式を使っています。

SHIFT ＋ BREAK ：左側に示されたキーを押しながら右側のキーを押すことを意味します。

＋ ― 押しながら

1．動作モード

（1） 2つの動作モード

BASICには、命令をすぐに実行する直接モード（ダイレクトモード）と、命令（プログラム）で記憶しておいてあとで何回でも利用できる間接モード（プログラムモード）の2つの動作モードがあります。

★それでは、コンピュータに実際に命令をして行くことにしましょう★

（2） 直接モード

まず

PRINT␣8　　␣はスペースキーを押すことを示します。（Tと8の間は、PRINT 8でも、PRINT␣␣8でもかまいません。）

と、キーを押してみてください。画面には、なんの変化もありません。これは、コンピュータの画面に文字は表示されても、コンピュータ内部にはそのデータが読み込まれていないからです。

CRキーは、画面に表示された文字列を意味のあるデータとして、コンピュータ内部へ送る働きをします。プログラムや命令を入力する場合は、必ず行の最後にCRキーを入力する必要があります。

それでは次に、

PRINT　8 CR

と、キー入力してみてください。

画面には、「8」という数字が表示されます。つまり、「8」という数字を画面上に表示せよということを示しています。

[例]

PRINT　8 CR ……「8」という数字を表示しなさい。 8　　……「8」という数字を表示します。 Ready　　……命令の実行が終りました。

①直接モードの電卓的な使い方

```
PRINT　8＋3 CR
```

と、キー入力してみてください。

画面には、8＋3の答え、つまり11と表示されます。

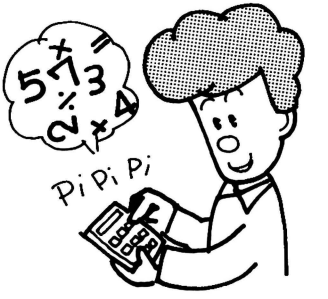
[例]

```
PRINT　8＋3
  11
Ready
```

同様に、PRINT 8－3 CR とすれば「5」が表示されます。ところで、「8×3」や「8÷3」の値を表示させたい場合、×（かけ算の記号）や÷（割り算の記号）は、コンピュータでは使えません。

四則演算の記号は、コンピュータでは次のようになります。

たし算の記号	＋
ひき算の記号	－
かけ算の記号	＊　（アスタリスク）
割り算の記号	／　（スラッシュ）
べき乗の記号	↑



それでは、電卓がわりに使ってみましょう。

[例]

```
PRINT　5＊8＋2 CR ……… 42
PRINT　5＊(8＋2) CR ……… 50
PRINT　8／2↑2 CR ……… 2
```

＊とか＋とか↑などが1つの式の中にあるとき、どの順で計算していくかの優先順位は、まず「(　)」で、2番目に「↑」、3番目に「＊,／」、4番目が「＋,－」となります。同じ優先順位のものは、左から右へ実行されます。

②文字の表示

次に、8という数字を「"」(ダブルクォーテーションマーク)で囲ってみてください。(「"」)は SHIFT キーを押しながら 2 のキーを押します。

```
PRINT  "8" CR
8
Ready
```

と、画面に表示されているはずです。前に、「"」を付けずに入力したときは、8という数字がPRINTのRの下にきていましたが、今回は行の先頭にきています。これは、8という数字が文字として扱われたからなのです。(コンピュータでは数値の場合は先頭に符号が入り、正の数の時はスペースで表しますが、文字の場合はスペースは入りません。)

次に、

```
PRINT  "8+3" CR
8+3
```

と、キー入力してみてください。画面には8+3という文字列が表示されます。つまり、「"」と「"」の間の文字の並びを表示せよという意味になるのです。「"」と「"」の間の文字の並びを文字列といいます。「"」で囲むことができる文字列の長さは、最大248文字までです。

それでは、いろいろな文字列を表示させてみましょう。

[例]

```
PRINT  "SHARP" CR

PRINT  "SHARP"+"MZ-1500" CR
                  ↑――― 文字列と文字列は+記号で結ばれます。
PRINT  "シャープ␣エムゼット-1500" CR
                  ↑――― 空白も1つの文字として扱います。
```

（３）間接モード（プログラムモード）

パーソナルコンピュータをコンピュータらしく使うためには、間接モードを使います。

あなたが、コンピュータへ伝える命令（コマンド）などをコンピュータ内部のメモリの中に、**行番号（１～65535の整数値）**をつけて順番に記憶しておき、行番号の順に実行して行きます。

このモードでは、いろいろな命令を組み合わせて、複雑な処理を実行させることもできます。行番号をつけた命令のならびをプログラムといい、プログラムの中で使う命令のことを文（ステートメント）といいます。

一度使ったプログラムは使回でも利用でき、また何度でも修正することができます。また、そのプログラムをクイックディスクやカセットテープなどに保存することもできます。

プログラムを実行するには、RUN という命令（コマンド）を使います。

[例]

行番号

```
10  PRINT  8 CR
20  PRINT  8＋3 CR
30  PRINT"8＋3" CR
40  PRINT"SHARP"＋"シャープ" CR
```

行番号をつけた命令の←ならびを**プログラム**といいます。

RUN CR……と、キー入力してください。

```
 8
 11
8＋3
SHARPシャープ
```

と、画面上に表示されます。

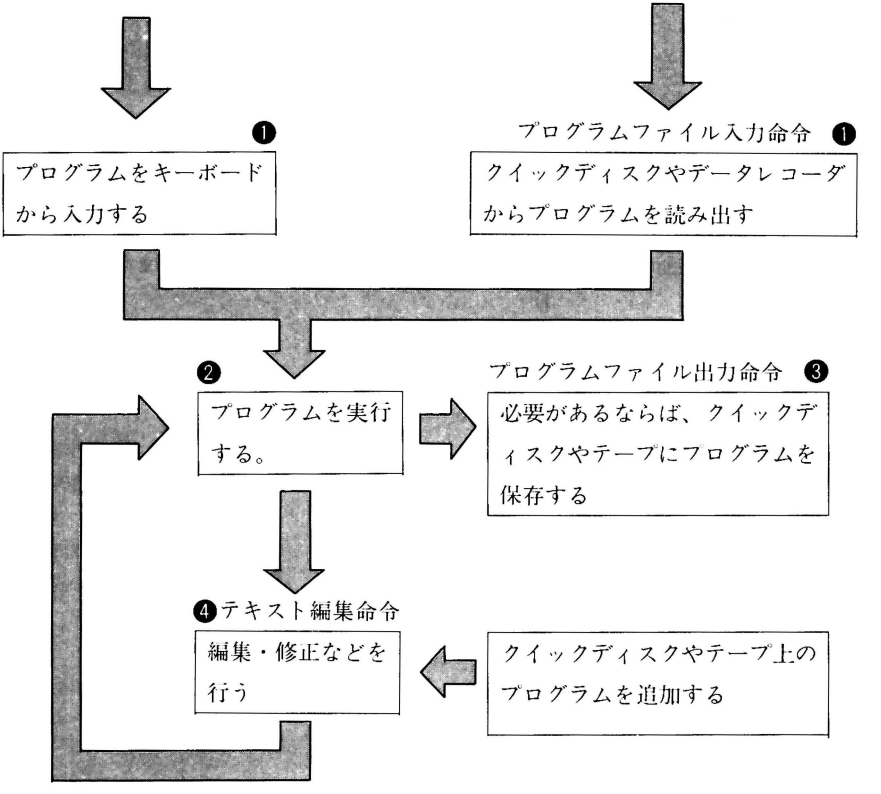
行の左端についている10、20、30、40、という数字のことを行番号と呼びます。行番号は、プログラムの途中に別の命令や文を挿入する場合がありますので、普通は10きざみに行番号をつけておきます。

2．プログラムを使おう！

BASICプログラムの使い方を示します。

自分で作るプログラム

クイックディスクやカセットテープ（テープ）に記録されているプログラム



（1）プログラムを入力してみよう

コンピュータは、電源を切ったり、新しいプログラムをクイックディスクやカセットテープから読み出したりしない限り、前に入力したプログラムの内容が残っています。次に何かプログラムを入力しようとした場合、前のプログラムがじゃまになることがあります。そこで、一度前のプログラムを消して、新しいプログラムを入力する必要があります。前のプログラムを消してみましょう。

```
NEW [CR]
```



と、キー入力してみてください。

これで、前の不要なプログラムが消えました。

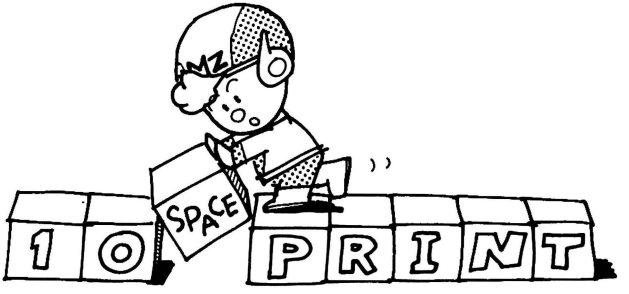
コンピュータは、命令を行単位で記憶して行きます。[CR]キーは、記憶させるためのキーなので、次の行に移るときは必ず[CR]キーを押してください。

なお、行とはプログラム上での行であり、画面上では数行になることもあります。ただし、1行は最大255文字ですが、あまり長いとわかり難くなりますので、適当な長さで区切りましょう。（プログラムの種類によっては255文字まで使えない場合もあります。）

また、文字と文字の間に空白を表したいときには、スペースキーで入力します。

```
10␣PRINT"ABC"
   ↑
   └──スペースキーを押す
```

と入力します。プログラムはコンピュータのメモリに記憶されます。



（2）プログラムの入出力について

プログラムをクイックディスクやデータレコーダから読み出したり、保存したりするには、LOAD 命令（147ページ）と SAVE 命令（148ページ）を使います。

①プログラムを読み出す

クイックディスクやデータレコーダに記憶されているプログラム（たとえば、DEMO という名前のプログラム）を使いたい場合には、LOAD 命令を使います。LOAD とは、プログラムを読み出せという命令です。

[例]

```
LOAD"QD:DEMO" CR
```

- DEMO：プログラムの名前です。（記憶させるときに指定した名前）
- QD：プログラムが記憶されている装置の名前（たとえば QD はクイックディスク、CMT はデータレコーダ）を指定します。

②プログラムを記憶させる

コンピュータのメモリ上のプログラムは、本体の電源を切ると、すべて消えてしまいます。そのため、メモリ上のプログラムを保存するためには、クイックディスクやカセットテープなどにそのプログラムを記憶させておく必要があります。たとえば、現在コンピュータのメモリ上にプログラムがあるとします。これに名前を付けてクイックディスクやデータレコーダに記憶させたい場合には、SAVE 命令を使います。SAVE とは、プログラムを記憶せよという命令です。

[例]

```
SAVE"QD:AAAA" CR
```

- AAAA：プログラムの名前です。（8文字以内の英数字で好きな名前）
- QD：プログラムをファイルとして記憶させておく装置の名前を指定します。

（3）プログラムを実行しよう

プログラムを実行するためには、RUN 命令（103ページ）を使います。

```
RUN CR
```

プログラムの実行が終了すると画面に

```
Ready
```

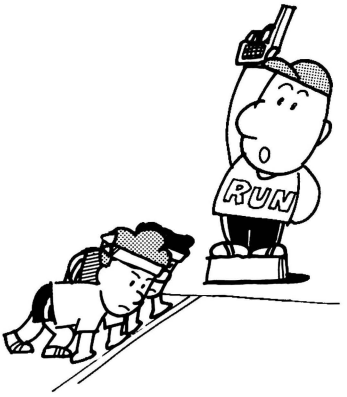
と表示されます。

また、プログラムの実行を途中で終了したい場合には、SHIFT キーを押しながら、BREAK キーを押すとプログラムが中止され、カーソルが点滅して入力待ちの状態になります。

```
Break in 30
         ↑――実行が中止された場所（行番号）
Ready．←ピリオド（．）はCONT可能であることの表示。

■……カーソルの点滅
```

なお、ふたたびプログラムを実行させる場合には、CONT 命令（102ページ）を使うことにより、プログラムを中止させた場所より続けて実行させることができますが、CONT できない場合はピリオド「．」が表示されません。



（4）プログラムを編集・修正しよう（テキスト編集命令）

実際にプログラムを作るときに、プログラムの編集（入力、変更、削除など）に使われる命令をまとめて簡単に説明します。

①編集・修正の確認をする。

コンピュータのメモリに入力されているプログラムを画面に表示させるには、

```
LIST [CR]
```

とすれば、入力されたプログラムがすべて表示されます。

```
LIST 10—100 [CR]
```

などとして、プログラムの一部だけを表示することもできます。この表示されたプログラムをLIST（リスト）といい、プログラムの確認や変更を行う場合はこのリストを見ながら行います。

間違いや変更したい行が見つかったときは、変更したい行の行番号と文を入力し、[CR]キーを押せば、入力したい行が新しいプログラムになります。

[例]

```
10 PRINT 8
20 PRINT 8＋3
30 PRINT "8＋3"
40 END
20 PRINT 8＋5
```

⇨

[結果]

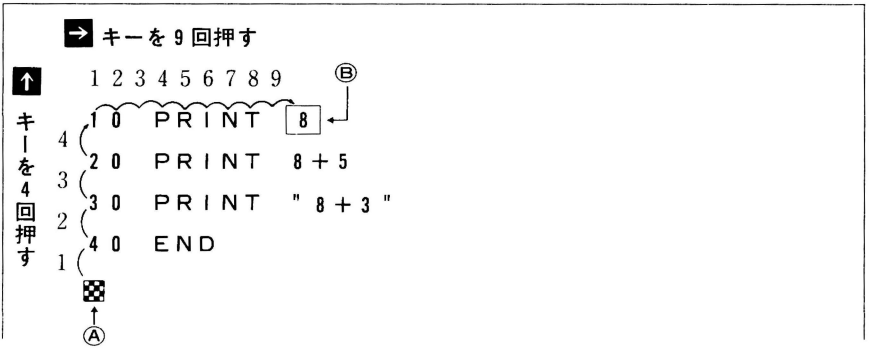
```
LIST [CR]
10 PRINT 8
20 PRINT 8＋5
30 PRINT "8＋3"
40 END
```

行番号20の8＋3を8＋5に変更したい場合、変更したい行番号と命令を入力します。変更されたことは、LIST命令で確認してください。

行番号20のPRINT 8＋3がPRINT 8＋5に変わっています。

行全体を変更するのではなく、一部を変更するときはリストを表示し、カーソルコントロールキー（[←][→][↑][↓]）を使ってカーソルを変更したい所に移動させます。そこでプログラムを変更し[CR]キーを押せば、一部分の変更もできます。

たとえば、[例] の行番号10のプログラムにある 8 という数字を 9 に変えるには、



カーソルコントロールキーの ↑ キーを 4 回、→ キーを 9 回押してカーソルをAの位置からBの位置まで移動させます。（カーソルは好きな順序で押すことができます。）8 の上にカーソルを重ね 9 CR と入力します。

また、プログラムの一部を追加したり、取り消したりするには、INST キーと DEL キーを使います。キーボードの使い方については「オーナーズマニュアル」を参照してください。

次に行番号20のプログラムを 20 PRINT 40＊8＋5 と変更してみましょう。まず、カーソルを 8 の上に重ね、INST キーを 3 回押してください。8＋3 が右に 3 文字分動いて、

```
20 PRINT ␣␣␣8＋5
          ↑
          └──カーソルの位置
```

となります。次に、40＊ CR とすれば文字の追加は終わります。

[結果]

```
20 PRINT 40＊8＋5
```

こんどは、行番号30のプログラムから＋を取り除いて、30 PRINT "83" とします。行番号30のプログラムへ、カーソルを移動させ 3 の上に重ね DEL キーを押すと＋が消えて

```
30 PRINT "83"
```

となりますので、次に CR キーを忘れず入力してください。

プログラムを変更したあとは、実行させる前に LIST 命令を実行し、プログラムの点検をしておくようにしましょう。

②プログラムの追加

すでに入力したプログラムの途中に新しい行を追加したいことがあります。たとえば、行番号10と行番号20の間にプログラムを追加したいときは、

```
15 PRINT"ABC" CR
```

とすれば、行番号10と行番号20の間に行番号15が追加されます。

[例]

```
10 PRINT 9
20 PRINT 40*8+5
30 PRINT "83"
15 PRINT "ABC" CR
```

⇨

[結果]

```
LIST CR
10 PRINT 9
15 PRINT "ABC"
20 PRINT 40*8+5
30 PRINT "83"
```

と入力します。

③プログラムの取り消し

プログラムの中の1つの行を取り消したい場合は取り消したい行の行番号を入力し、続いて CR を押せばその行が取り消されます。連続した何行かのプログラムを取り消したいときは DELETE 命令（95ページ）を使います。

行番号15～20を削除したい場合 DELETE␣15－20 CR と入力します。

[例]

```
10 PRINT 9
15 PRINT "ABC"
20 PRINT 40*8+5
30 PRINT "83"
DELETE␣15－20 CR
```

⇨

[結果]

```
LIST CR
10 PRINT 9
30 PRINT "83"
```

④プログラムの整理

プログラムを作るときに追加や取り消しが多く、行番号の間がつまって追加するすきまがなくなったり、行番号が不ぞろいなためにプログラムが見にくくなることがあります。このようなときは、**RENUM** 命令（98ページ）を使い、行番号を付けなおしたり、行番号を整理することができます。このとき、プログラムの中に出てくる **GOTO** 命令（114ページ）などの行き先きとして使われる行番号も新しく変更された行番号にもとづき変わります。

下の［例］のプログラムに対しRENUM CRを実行すると、右の［結果］となります。

［例］

```
10  PRINT  9
12  PRINT  8+3
15  PRINT  8-3
20  END
RENUM CR
```

［結果］

```
LIST CR
10  PRINT  9
20  PRINT  8+3
30  PRINT  8-3
40  END
```

⑤行番号の自動発生

プログラムを入力するとき、行番号を各行ごとに入力しますが、長いプログラムになるといちいち入力するのは手間がかかります。そこで **AUTO** 命令（94ページ）を使うと、コンピュータが自分で行番号を付けてくれます。

```
AUTO CR
```

とすると、最初の行番号「10」が次の行に表示され、その行の入力待ちの状態となります。すべての入力が終り、AUTO 命令を取り消したい場合は、次の行番号が表示されているとき、SHIFT キーを押しながら BREAK キーを押します。

[例]

```
AUTO [CR]
10  PRINT  9 [CR] ←――プログラムを入力する
           ⋮
110  END [CR]
120 [SHIFT] + [BREAK] ←行番号110行までのプログラムを入力し120行目
                       が表示されたところでAUTO命令を取り消す。
                       [SHIFT] + [BREAK] の「+」記号は左側の
                       キーを押しながら右側のキーを押すことを意味
                       します。
Ready
```

●誤差について

コンピュータを使う場合は、どうしても、誤差について考えなければなりません。というのは、誤差を少なくするために数値データの扱うケタ数を多くすることが考えられますが、いくらコンピュータでも、無限のケタ数を扱うというわけにはいきません。また、ケタ数を多くするとそれだけ時間がかかります。ですから、誤差の発生する原因を知り、誤差の少ないプログラムを作成するという考え方が大切になってきます。(例　5÷3×6のときは5×6÷3と手順を変える。)

MZ-1500のBASIC (5Z001) を使用する場合は次の事項にご注意ください。

①切り捨て誤差

たとえば、1/3を小数で表すと、　1/3=0.333333333333……

有効数字8ケタ ――――――

③ケタ落ち

ひき算を行った場合、誤差の影響が大きくなることをいいます。誤差を含んだケタの上に・を付けて表すと右のようになります。ひき算を行う前は、100012に対し±1（約0.001%）以内であった誤差が、11に対し±1（約10%）となり、誤差の影響が大きくなるのです。

$$\begin{array}{rr} & 100012\dot{} \\ -) & 100001\dot{} \\ \hline & 11\dot{} \end{array}$$

④近似値計算による誤差

コンピュータ内部では、べき乗の計算や三角関数、対数計算は、近似値計算を行っていますので、どうしても近似しただけ誤差が発生します。

3．基本命令

（1）データは変数と呼ばれる箱に入れよう（代入文）

データには、定数と変数がありました。BASICで扱う変数は、数値や文字列を入れるための箱にたとえることができます。この変数という箱に数値や文字列を入れることを代入するといいます。

[例1]

```
10  A＝5
20  A＝A＋1
30  PRINT  A
RUN
```

数値データ
代入記号
Aという名前の変数
A＋1、つまり5＋1の値が新たにAに代入されます。



[結果]

```
 6
```

[例２]

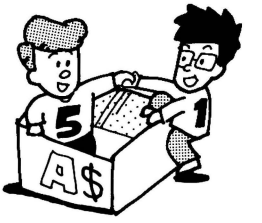
```
10  A$="5"
```

文字列を表す記号

数字としてでなく文字としての５を意味します。

$は文字変数を表し、A$で変数名となります。

```
20  A$=A$+"1"
30  PRINT  A$
RUN
```



[結果]

```
5 1
```

注)・A＝BとB＝Aでは意味が異なり、Aが１、Bが２の場合、A＝Bを実行するとAもBも２となり、B＝Aを実行するとAもBも１となります。

・変数名は最初の２文字で識別されますので、最初の２文字が同じ変数は同一の変数であるとみなされます。

［例］　ABCとABDは同一の変数

（２）データを表示しよう（PRINT命令：121ページ）

データを表示するためにはPRINT命令があります。

たとえば、変数Aに６、変数Bに－８と代入し、同一行に、２つの変数の値を表示する場合、２通りの表示方法があります。

[例１] ＰＲＩＮＴ　Ａ，Ｂとした場合

```
10  A=6
20  B=-8
30  PRINT  A, B←変数Aと変数Bをコンマで区切る
```

このプログラムを実行すると、－８は左端から10文字あけて表示されます。

[結果]

```
6              -8
```

[例２] ＰＲＩＮＴ　Ａ；Ｂとした場合

```
１０　Ａ＝６
２０　Ｂ＝－８
３０　ＰＲＩＮＴ　Ａ；Ｂ←変数Ａと変数Ｂをセミコロンで区切る
```

実行結果は、次のように間をあけずに表示されます。

６－８



・［例１］のプログラムの行番号30を次のように変えても同じ結果が得られます。

```
３０　ＰＲＩＮＴ　Ａ，
４０　ＰＲＩＮＴ　Ｂ
```

・単に改行する場合は、PRINTだけを入力します。

・PRINTを入力する場合は「？」と省略できます。ただし、リスト上ではPRINTと表示されます。

［例］　？＂ SHARP ＂　→はPRINT ＂ SHARP ＂と同じ意味です。

●マルチステートメント

上記の［例１］のプログラムは、次のように１行でも表せます。

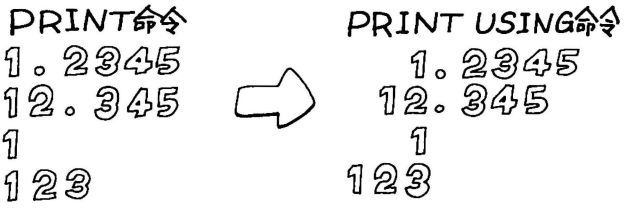
```
１０　Ａ＝６：Ｂ＝－８：ＰＲＩＮＴ　Ａ，Ｂ
```

このように、：（コロン）で命令を区切って、１行に複数の命令を書くことをマルチステートメントといいます。

●書式の指定（PRINT USING）

PRINT 命令を使って数値を画面に出力させたとき、数値のケタや小数点の位置がそろっていた方が見やすい場合があります。PRINT USING 命令は、データの表示のし方を指定したいときに使います。

PRINT USING "書式の指定" ; 変数 [; 変数……]



PRINT USING の命令の書式の指定には数値型・文字型・文字定数の出力機能・USING の分離があります。

①数値型

出力されるデータが数値のときに使います。数値のケタや小数点位置を指定したり、数値を3ケタごとに区切り、そこに「，」（コンマ）を入れたりすることができます。また、数値に¥（円記号）や＋（プラス）、－（マイナス）をつけて表示させる機能もあります。

②文字型

表示されるデータが文字のときに使います。文字列の最初の1文字のみを表示する機能や、指定した文字列を左づめで表示する機能があります。

③文字定数の表示機能

USING 命令で使う文字列のほかにそれとみなされないキャラクタがある場合は、それをそのまま表示する機能です。

④ USING の分離

PRINT 命令中、通常の PRINT 命令と USING 命令を組み合わせて使用することができます。

（3）データを入力しよう（INPUT 命令：127ページ）

プログラム実行中に、キーボードから入力したデータを変数に代入するときには、INPUT 命令を使います。

[例 1]

```
10  INPUT  A
20  PRINT  A
30  END
RUN CR
?  99 CR
 99
```

このプログラムを RUN させると、まず INPUT 命令で、画面に「？」と表示が出力され、入力待ちの状態になります。
ここでたとえば、99 CR と入力すると、A という変数に99という数値が代入されます。次に、行番号20で画面に変数 A の値「99」が表示され、実行を終ります。

[例 2]

```
10  INPUT  A
20  INPUT  B
30  PRINT  A ; B
```

行番号10の INPUT 命令でたとえば、5 CR と入力すると、行番号20の INPUT 命令では次の行にふたたび「？」が表示され、変数 B の入力待ちとなります。そこでたとえば10 CR と入力します。これで変数 A には 5、変数 B には10が入力され、行番号30の PRINT 文で変数 A、変数 B の値が表示されます。

[例 3]

```
10  INPUT  A , B
20  PRINT  A ; B
```

[例 2]と同じように 2 つの変数 A、B を入力するプログラムです。行番号10の入力は「？」に続き、10, 20 CR のようにデータを「 , 」（コンマ）で区切って入力する

か、10 CR でまず最初のデータを入力し、次に「？」が表示されたときに、20 CR と次のデータを入力します。これで変数A、変数Bに代入された値が行番号20のPRINT命令によって表示されます。

[例4]

```
10  INPUT"A=";A
20  INPUT"B=";B
30  PRINT  A;B
RUN CR
A=10 CR
B=20 CR
  10   20
```

上の［例1、例2、例3］では、入力要求が「？」だけです。しかし、［例4］のようにメッセージを入れることもできます。このときは、上の例のように「？」は出てきません。

（4）プログラムの流れを変えよう
（GOTO命令（114ページ）、GOSUB命令（116ページ））

プログラムは、原則として行番号の順に実行して行きますが、GOTO命令を使うと、これを希望する順番に実行させることができます。簡単なプログラムでその使い方をみてみましょう。

[例1]

```
10  INPUT  A
20  B=(A+5)/2
30  PRINT  B
40  GOTO  10
```

行番号40のGOTO命令がなければ、行番号10～30を何度も実行させようとすると、いちいちRUN命令を入力しなければなりません。そこで行番号40にGOTO命令を入れることによって、GOTO命令にぶつかると、GOTO命令で指定している行番号（この

場合10）のところに実行が移るようにしておきます。これを「指定する行番号へジャンプする」といいます。この例では、何度でも実行がくり返されて終わりませんから、「？」が表示されたときに SHIFT ＋ BREAK キーを押して実行を終了してください。

●ラベル

GOTO 命令のジャンプ先は、行番号で指定する以外に、ラベルを使うこともできます。ラベルは、ジャンプ先の名前を" （ダブルクォーテーション）で囲んで表します。使用できる文字は、英、数字のほか、カナ文字、記号があります。LABEL 命令（113ページ）は、次のように書きます。

LABEL " ラベル名 "

[例 2]

```
10  LABEL"ハジメ" : INPUT  A, B
20  C=A+B
30  PRINT  C : GOTO"ハジメ"
40  END
```

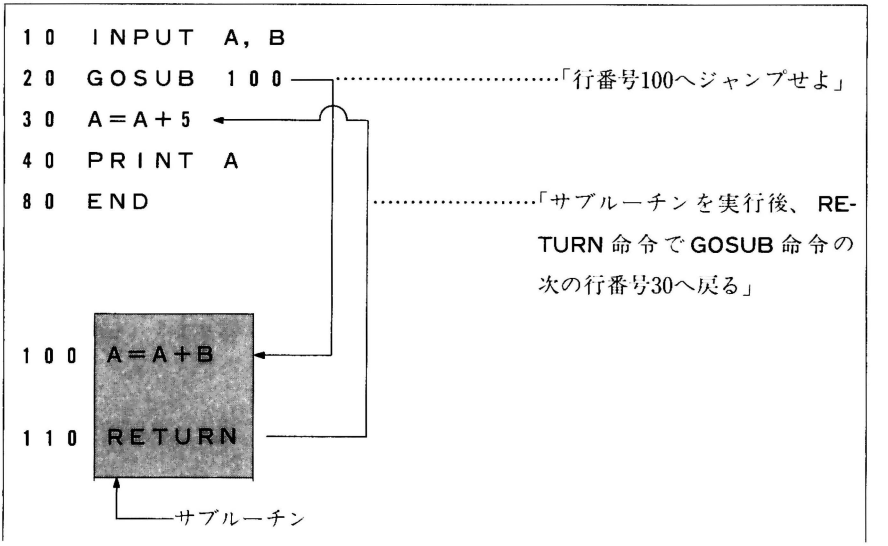
[例 3]

```
10  PRINT"SHARP" : GOTO10
```

このように GOTO 命令は、自分自身の行番号へジャンプすることもできます。

GOTO 命令では、指定の行番号またはラベルへジャンプすると、それきりですが、またもとに戻ってくるのが GOSUB 命令です。GOSUB の SUB というのは、サブルーチンのことです。GOSUB 命令は、「サブルーチンへジャンプせよ」という意味になります。プログラムで何度も実行する部分をサブルーチン化することによって、プログラムを見やすく短くすることができます。

[例1]



GOSUB 命令と RETURN 命令は、つねにペアで使用されなければなりません。

[例1]に次のプログラムを追加してみましょう。

```
50 GOSUB 100
60 A=A+10
70 PRINT A
```

このプログラムを追加することで、行番号100～110のサブルーチンを2回呼び出すことになります。このように同じサブルーチンを何度も実行させることで、GOSUB～RETURN 命令の効果が表れます。また、GOTO 命令と同じように、GOSUB 命令も、LABEL 命令が使えます。

（５）条件を判断しよう（IF～THEN～：ELSE 命令：118ページ）

日常、私達はいろいろな条件を判断して行動しています。たとえば、「明日、もし雨が降れば、運動会は中止します。そうでなければ予定通り行います。」



というような文章をBASICに置き換えてみますと「もし～ならば～そうでなければ～」にあたるものがIF～THEN～：ELSE命令です。

```
IF　×××　THEN　□□□　：ELSE　△△△
```

IFの後に続く×××の条件が満たされていれば、□□□を実行しなさい、そうでなければ、△△△を実行しなさい、というのがIF～THEN～：ELSE～の命令です。□□□、または△△△が行番号の場合は、その行番号へジャンプします。ラベル名の場合は、ラベル名が付けられている行にジャンプします。また、命令の場合はその命令を実行します。

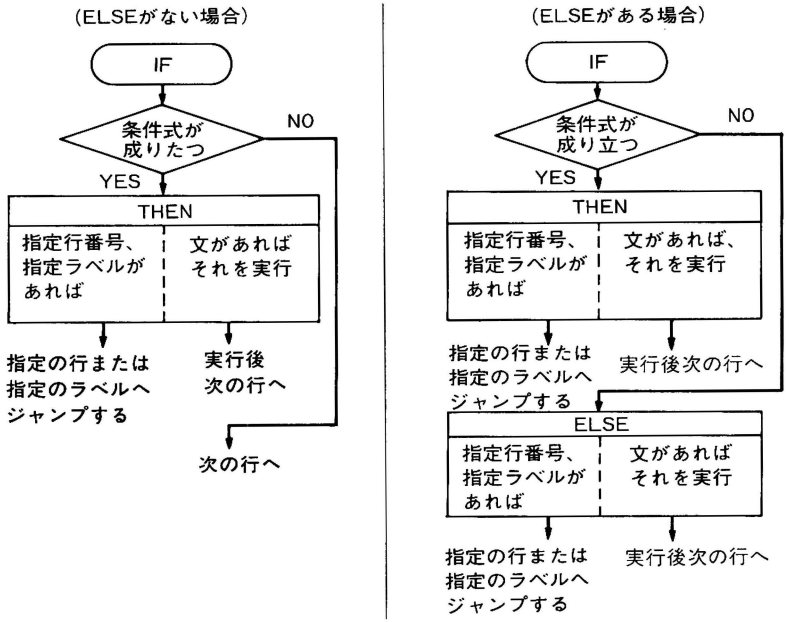
たとえばAが10より大きいならば、Aから20を引くというプログラムは、

```
ＩＦ　Ａ＞１０　ＴＨＥＮ　Ａ＝Ａ－２０
```

となります。

条　件	例	意　味
＜	A＜B	AはBより小さい
＞	A＞B	AはBより大きい
＝	A＝B	AとBは等しい
＜＞，＞＜	A＜＞B，A＞＜B	AとBは異なる
＞＝，＝＞	A＞＝B，A＝＞B	AはBより大きいか等しい
＜＝，＝＜	A＜＝B，A＝＜B	AはBより小さいか等しい

ELSE がある場合 IF～THEN～（：ELSE～）命令のプログラムの流れは下図のようになります。



[例１]

```
10  INPUT X
20  IF X=0 THEN END
30  IF X>0 THEN PRINT"0ヨリオオキイ":ELSE PRINT
"0ヨリチイサイ"
40  GOTO 10
```

[結果]

```
RUN CR
?  3  CR ←――――  3を入力します。
0ヨリオオキイ
?  −5 CR ←――――  −5を入力します。
0ヨリチイサイ
?  0  CR ←――――  0を入力するとプログラムが終ります。
Ready
```

●論理計算

コンピュータは判断によってYESかNOの結果を出す仕事が少なくありません。ここでは、いくつかの条件を判断するときに使う論理式というものについて考えてみましょう。論理式では、NOT、AND、OR、XORなどの論理演算子を使います。論理演算では、2進数を使用しその結果を出してきます。「正ならば1」を、「負ならば0」を使い計算します。

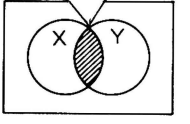
かなり、むずかしい話になってしまいましたが、ベン図を使った実際の例で、その働きを見てみましょう。

－1を2進数で表すと

① X AND Y （論理積＝and）

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

X AND Y
ANDというのはXであり、かつYであるという意味です。



このような図をベン図といいます。

－1 AND 4は？

－1＝ 1111 1111

4＝ 0000 0100

・－1 AND 4は

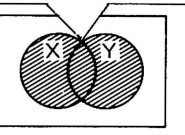
プラスをあらわす 00000100

＝4 答 4です

② X OR Y （論理和＝inclusive or）

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

X OR Y
XまたはYという意味です。



50 OR 44は？

50＝ 0011 0010

44＝ 0010 1100

・50 OR 44は？

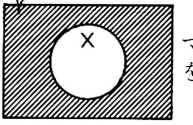
0011 1110

＝62 答 62です

③ NOT X （否定＝not）

X	NOT X
1	0
0	1

NOT X
Xではないという意味です。



NOT 13は？

13＝ 0000 1101

・NOT 13は

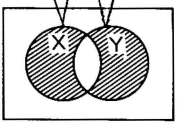
マイナスをあらわす 1111 0010

＝－14 答 －14です

④ X XOR Y （排他的論理和＝exclusive or）

X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	0

X XOR Y
XかYのいずれかが１のときだけ、結果が１になるのがXORです。



3 XOR 2は？

3＝ 0000 0011

2＝ 0000 0010

・3 XOR 2は

0000 0001

＝1 答 1です

論理演算はプログラムでは、前に述べたIF～THEN～：ELSE命令の中で使うことが多いのです。たとえば、次のように使われます。

・IF －30＜X AND Y＜20 THEN 120 ： ELSE 100

（Xの値が－30＜Xでかつ Y＜20ならば120行へ、それ以外ならば100行へジャンプせよ、という命令です。）

・IF －30＜X OR Y＜20 THEN 120 ： ELSE 100

（Xの値が－30＜Xまたは Y＜20ならば120行へ、それ以外ならば100行へジャンプせよ、という命令です。）

（6）くり返してみよう（FOR～NEXT命令：111ページ）

同じ種類の仕事をくり返して行うプログラム、たとえば" SHARP "という文字を10回書かせるプログラムを考えてみましょう。今までの命令で作ると、次のようなプログラムが考えられます。

[例1]

```
10  I＝0
20  PRINT  "SHARP"
30  I＝I＋1
40  IF  I＝10  THEN  60
50  GOTO  20
60  END
```



ここで、FOR～NEXT命令を使用すると、行番号10、30、40、50を1つの命令で行うことができます。

[例2]

```
10  FOR  I＝1  TO  10  STEP1      …「Iという変数の値を1から10まで1ずつ変化させる」
20  PRINT"SHARP"
30  NEXT  I ……「Iに1を加えた結果、Iが10と等しいか、もしくは小さいときは10へ戻り、それ以外は次の行に進む」
40  END
```

FORとNEXTはつねにペアにして使います。行番号10の1≦I≦10という条件が成立している間はFOR～NEXTの間のプログラムがくり返し実行されます。

FOR～NEXT命令の形式は、

```
FOR  変数＝最初の値  TO  最後の値  STEP  増分
  ⋮
NEXT  変数
```

となります。なおSTEP以下は、増分が1のときは省略することができます。上のプログラムの行番号10はFOR　I＝1　TO　10でも同じことになります。

FOR～NEXTで囲まれた部分を一般にループといい、1つのループの中に他のループを作ることもできます。

●ネスティング（多重化）

FOR～NEXT ループはループを多重に重ねることができます。ループの重ね合わせは、必ず入れ子型（ネスト構造）となるようにし、内側のループは外側のループに完全に含まれなければなりません。また、それぞれのループは別々の制御変数を使わなければなりません。

```
10  FOR  X=1  TO  9
20  FOR  Y=1  TO  9
30  FOR  Z=1  TO  9
40  PRINT  X+Y+Z
50  NEXT  Z
60  NEXT  Y
70  NEXT  X
```

内側のループ　外側のループ

（正）

```
10  FOR  X=1  TO  9
20  FOR  Y=1  TO  9
30  FOR  Z=1  TO  9
40  PRINT  X+Y+Z
50  NEXT  Z, Y, X
```

（正）

1行でNEXT命令を使う場合は、中のループから順番に書きます。

```
10  FOR  X=1  TO  9
20  FOR  Y=1  TO  9
30  PRINT  X+Y
40  NEXT  X
50  NEXT  Y
```

（誤）

ループが交差してはいけません。

```
10  FOR  X=1  TO  9
20  FOR  X=1  TO  9
30  NEXT  X
```

（誤）

同じ制御変数を使ってはいけません。

多重ループでのネスト数は、文法上の制限はありませんが、プログラム実行時における残りエリアの大きさによって制限を受けます。

（7） ON～GOTO 命令：115ページ、 ON～GOSUB 命令：117ページ

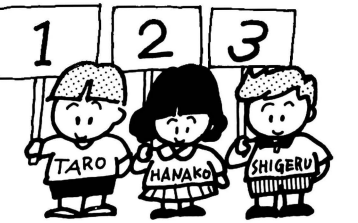
変数の値によって、次に行う実行を判断したいときには、次のように番号と名前を対応させ、その番号を入力することによって名前を出力させることができます。たとえば、名前に

1…TARO,　2…HANAKO,　3…SHIGERU

という番号を付けてプログラムを作ると

[例1]

```
10  INPUT BANGO
20  IF BANGO=1 THEN 50
30  IF BANGO=2 THEN 60
40  IF BANGO=3 THEN 70
50  PRINT "TARO" : END
60  PRINT "HANAKO" : END
70  PRINT "SHIGERU" : END
```



となります。しかし、数が多くなると行番号20～40の部分が多くなりすぎて、長いプログラムになってしまいます。そこで次のような命令を使い、行番号20～40を、

```
20  ON BANGO GOTO 50, 60, 70
```

とします。この ON～GOTO 命令は、

[ON　変数　GOTO　行番号，行番号，・・・・]

と書きます。「変数の値が1ならば、GOTO に続く1番目の行番号に、2ならば2番目の行番号に・・・」というふうに、GOTO での行き先を与えることのできる命令（分岐命令）です。

ON～GOTO 命令と同類の命令に ON～GOSUB 命令があります。これを使うと、上のプログラムは次のようになります。

[例2]

```
10  INPUT BANGO
20  ON BANGO GOSUB 50, 60, 70
30  END
50  PRINT"TARO" : RETURN
60  PRINT"HANAKO" : RETURN
70  PRINT"SHIGERU" : RETURN
```

（8） READ～DATA 命令：130ページ

INPUT 命令や GET 命令のほかに、コンピュータに変数を与える方法に READ 命令があります。READ 命令は、DATA の後に示す内容の中からデータを順番に１つ１つ読み込む命令です。

[例１]

```
10  READ  A, B, C, D
20  PRINT  A ; B ; C ; D
30  END
40  DATA  10, 100, 50, 60
RUN
 10  100  50  60
```

READ～DATA 命令により A には１０、B には１００、C には５０、D には６０が代入されます。

[例２]

```
10  READ  X$, A1, Y$
20  PRINT  X$ ; A1 ; Y$
30  END
40  DATA  ABC, 1, DE
RUN ┌──── 符号用（正のときはスペース）
ABC␣1DE
```

DATA 命令における定数は数値でも文字列でもかまいませんが、読み込まれる変数の型と一致していなくてはなりません。

READ 命令の変数の順序と DATA 命令の順序さえ一致していれば、READ～DATA 命令は何行かにまたがって書かれていても、結果的には同じです。

READ 命令の変数の個数と DATA 命令のデータ数は一致しなければなりません。

READ 命令で DATA の途中まで読んだ後、次の DATA 命令を実行するとその次のデータから引き続き読み込まれます。

次に実行する READ 命令で読み込むデータの並びの位置を決めるためには、RESTORE 命令を使います。RESTORE 命令を使うことによって読み込むデータを指定した行番号（またはラベル）から読むことができます。

※ DATA 命令を使う場合、データの多い場合は余ったデータは無視され、少ない場合は、エラーとなります。

（9）注釈をつけよう（REM 命令：107ページ）

プログラムが長くなってくると、どこに何が書いてあるのかわからなくなってしまいます。そこで、REM 命令を使って、プログラム中に注釈を付け、プログラムリストを見やすくします。他の人に見せるプログラムには、わかりやすいように REM 命令を入れておくと親切です。

プログラムの実行は、REM 命令の行を無視して、次の行番号より実行します。

[例]

```
10  REM  サンプル ←―――――――― プログラムのタイトル
20  INPUT  "タテ＝" ; A
30  INPUT  "ヨコ＝" ; B
40  REM  メンセキ  ヲ  モトメル ←―――― 注釈
50  C＝A＊B
60  PRINT  "メンセキ＝" ; C
```

注) REM 命令は、「行の終わりまでを１つの文として扱う」ということになっていますので、REM 命令の右にマルチステートメントの形で他の文を書くことはできません。



（10）デファイナブルキーの使い方

キーボード左上に並んでいる F1 ～ F5 と書かれたキーは、そのまま使うか SHIFT キーを押しながら使うことによって、10種類の文字列を、コンピュータに覚えさせることができます。また、「＋」記号でCHR$関数を付けて登録することもできます。このキーを使うことにより、たとえばプログラム中に同じ文字列が何度も出てくる場合などに、この文字列をデファイナブルキーに登録しておけば登録させたデファイナブルキーを押すと、この文字列が画面に表示され、入力が楽になります。また、実行おいても偉力を発揮します。

設定の方法

デファイナブルキー F1 に「INPUT」という文字列を登録するには、

```
DEF KEY (1) ="INPUT"
```

と入力します。この文字列に CR のはたらきも付け加えたい場合は、"INPUT"＋CHR$ (13) と入力します。よく使う命令などを登録しておけば便利です。

また、PRINT 50＊10というような命令の実行をデファイナブルキー F2 に登録しておきたい場合には、

```
DEF KEY (2) ="PRINT 50＊10"+CHR$(13)
```

と入力します。登録されたデファイナブルキーの F2 を押せば、PRINT 50＊10と表示され、次の行には500と命令の実行結果が表示されます。

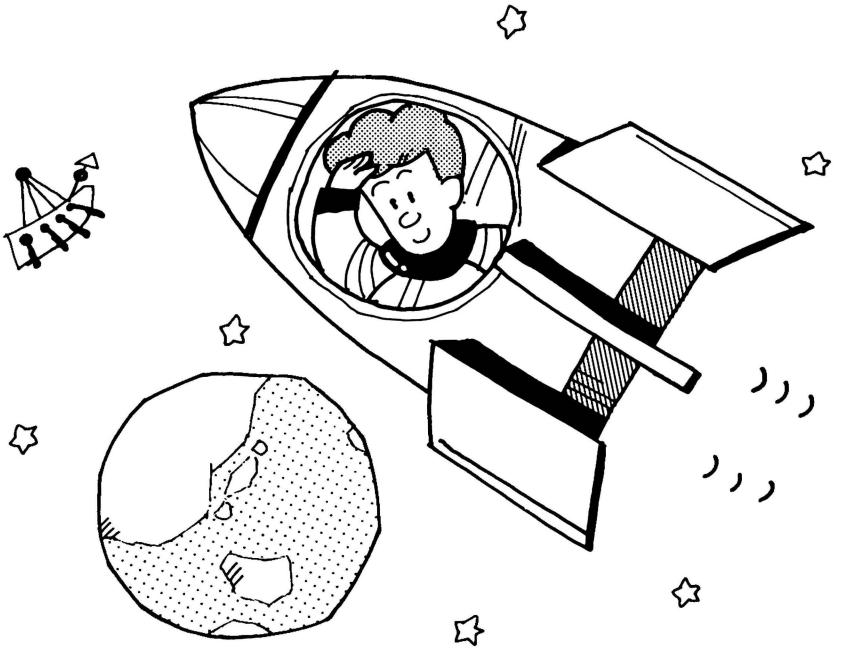
これらの、キーの内容は、本体の電源を切る（リセット時）か、または別の文字列を再登録するまで有効です。

現在、デファイナブルキーにどのような文字列が登録されているかを見るには、KEY LIST 命令(136ページ)を実行することにより画面に表示させることができます。

4．応用編

地球儀を見ると、日本列島やアメリカ大陸、ユーラシア大陸といった各大陸が地球という惑星を構成しているのがわかります。MZ-1500も画面表示やサウンド機能、ファイル機能といった数々の特長により、その性能が発揮されるように作られています。

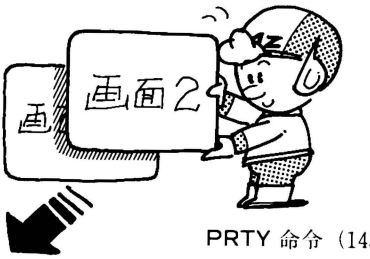
ここでは、私たちが地球儀を見て私達の住む世界を確認するように、MZ-1500の持つ機能を、全体からとらえていただくための説明です。また、その機能は、多くの命令や文により引き出されます。



（1）画面表示

テレビの画像は、放送局が送り出してくれますが、コンピュータではあなたが与えた命令にしたがって画像が作られます。

❶画面の優先順位



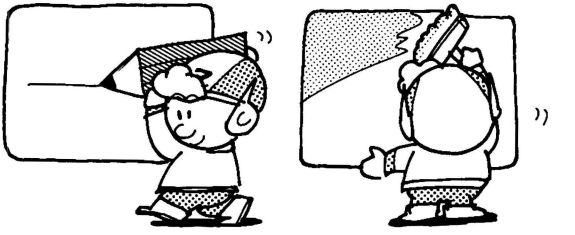
PRTY 命令（143ページ）でどちらの画面を優先するかきめます。

❷色を付ける。



PAL（169ページ）命令などでどんな色を使うか指定します。

❸描画を行う。

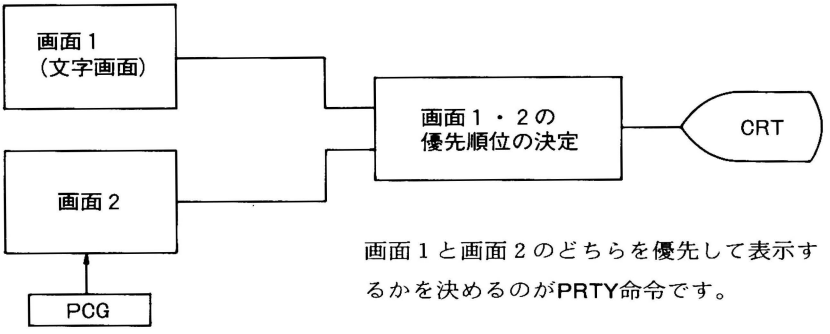


描画のための数々の命令を使い画面に線を書いたり画面をぬりつぶしたりします。

（２）初期設定

①モードの選択

本機には２つの画面があり、画面１は文字表示用に使用されます。画面２の使用方法は、２つのモード（イメージモード・グラフィックモード）により異なります。



画面１と画面２のどちらを優先して表示するかを決めるのがPRTY命令です。

イメージモードとグラフィックモードの切り替えは、INIT 命令で行います。

②イメージモード（INIT " CRT：I " で選択）

画面２をすべて PCG（次ページ参照）用画面として割り当てます。グラフィックの表示はできませんが、PCG を最大1000個まで使用できます。

③グラフィックモード（INIT " CRT：G " で選択）

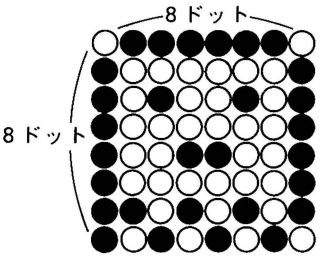
画面２を、グラフィックと PCG のために割り当てます。320×200ドットのグラフィック画面と、最大24個の PCG とを同時に表示することができます。

● PCG とは

数百種類もの文字や記号などのキャラクタパターンを MZ-1500 は覚えています。私たちはこれらのキャラクタを自由に使うことができます。しかし、その中にもないようなキャラクタを使いたいときには、自分でそのキャラクタを作ることができます。そのときに使うのが PCG です。

PCG とは、Programmable Character Generator の頭文字をとったもので、「利用者が定義できるキャラクタ発生装置」という意味です。

PCG にキャラクタを定義するには、**FONT$** 命令(189ページ)を使います。また、PCG に定義したキャラクタを使うときは **FPRINT** 命令(192ページ)などが使えます。



● PCG の特長

PCG を使う利点は、文字を表示する速さと同じ速さで、自由なパターンを表示できることです。1ドットずつ描いていくグラフィック命令を使うよりは速く同じパターンが PCG を使って描けます。

●定義できる数

イメージモードでは、PCG は1000種類まで定義でき、1ドットずつの色を決めることが可能です。

グラフィックモードでは、PCG は24種類定義でき、1ドットずつの色を決めることが可能です。

（3）画面に応じた表示の手順

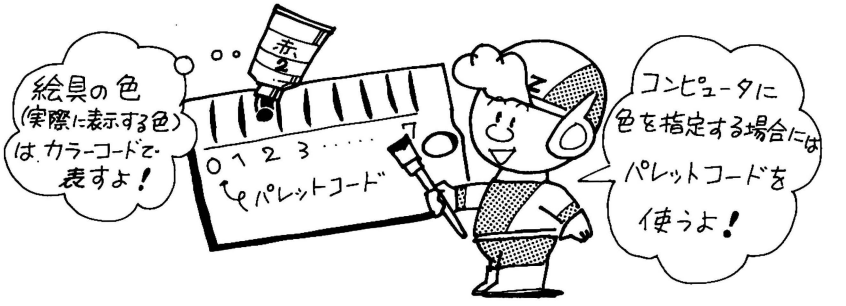
本機は、多彩なディスプレイ画面表示機能を持っています。いろいろな文字を使ったり、色鮮やかなコンピュータグラフィックを楽しむために、数々の命令が用意されています。

	画面1	画面2	
	文　字	PCG（1000種類） （イメージモード）	グラフィック・PCG（24種類） （グラフィックモード）
1.スクロール範囲	CONSOLE		
2.色	PAL		
	CCOLOR	（168ページ参照）	COLOR （各命令中で指定）
3.座標指定	CURSOR	POSITION	
4.描画	CSET CRESET		SET（点の描画） RESET（点の消去） LINE（直線の描画） BLINE（直線の消去） CIRCLE（円の描画） BOX（四角形の描画） PAINT（ぬりつぶし） PATTERN（パターンの作成） POINT （グラフィック座標のパレットレコード）
		FONT$	
		FPRINT	

（4）色

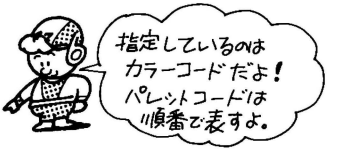
電源をONにした状態では、文字の色は白、背景の色は黒になっています。これを変えるには、各命令の中の色指定か、文字の場合はCCOLOR命令を、グラフィックの場合はCOLOR命令を使います。ただし背景の色はCCOLOR命令で指定された色になります。

色は、0～7の番号（パレットコード）で指定します。これは、絵を書くときのパレットからきた名前で、これに好きな色の絵具を当てることができます。(この絵具にも、0～7の番号がついていて、これをカラーコードといいます。)この割り当てを指定するのがPAL命令です。
ただし、電源ONの状態では、パレットコードはカラーコードと同じになっています。



注)カラーコードは次の通りです。

カラーコード	0	1	2	3	4	5	6	7
色	黒	青	赤	紫	緑	水色	黄	白



①PAL命令

[例] PAL 1, 3, 2, 4, 5, 0, 7, 6

指定されたカラーコードを左から順にパレートコード0～7に割り当てます。つまり、パレットコード0の色をカラーコード1（青）に、パレットコード1の色をカラーコード3（紫）に……と割り当てて行きます。なお、PAL CR で初期状態に戻ります。また CTRL ＋ D キーでも初期状態に戻ります。

②文字の色

PRINT 命令または CCOLOR 命令で指定します。

[例1] PRINT〔6, 4〕"ABC"

文字の色をパレットコード6の色、背景の色をパレットコード4の色にして、" ABC "を表示します。

[例2] CCOLOR 20, 10, 4, 6

……文字座標（20、10）の文字の色をパレットコード4の色、背景の色をパレットコード6の色に指定します。

③グラフィックの色

グラフィックの各命令で色の指定を行います。次の形で色を指定します。

〔パレットコード、ぬりつぶしコード〕

ただし、ぬりつぶしコードは0または1です。

- 0ならば、前の色を消して新しい色を描きます。
- 1ならば、前の色に重ね合せて(注)新しい色を描きます。

[例] SET 〔5, 0〕 100, 50

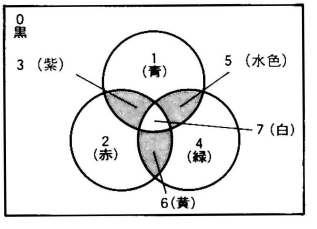
……グラフィック座標（100、50）の点を、前の色を消して、パレットコード5の色にします。

・グラフィックの各命令で色を指定しない場合は、COLOR 命令で指定された色になります。

[例] COLOR 4, 0

……グラフィックの色をパレットコード4、ぬりつぶしコードを0にします。

注)パレットコードの重ね合わせ方は、次のようになります。カッコの中は、電源をONにした状態の色です。



[例]・パレットコード1の上に2を重ね合わせるとパレットコード3になります。

・パレットコード1の上に4を重ね合わせるとパレットコード5になります。

（５）座標

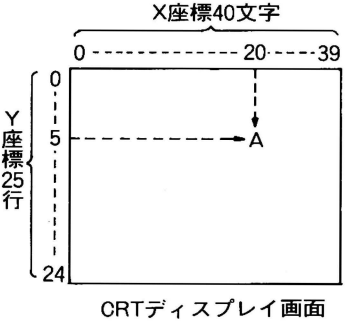
①位置の表しかた

画面上の任意の文字やグラフィックを描くためには、書きはじめる位置を決めなければなりません。この位置を決めるための命令がイメージモードでは、CURSOR 命令、グラフィックについては、POSITION 命令です。CURSOR 命令はカーソルの位置を、POSITION 命令はポジションポインタの位置を決めます。位置の表し方は座標という考えを使います。

座標は、将棋盤や碁盤と同じように（横の位置）と（縦の位置）を組み合わせて表します。座標にはカーソル座標とグラフィック座標の２種類があります。

②カーソル座標

文字画面で使われる座標です。横方向をＸ座標、縦方向をＹ座標といい、一般に（Ｘ座標、Ｙ座標）で表わします。左上を原点（０、０）とします。Ｘ座標の方向に40文字、Ｙ座標方向に25行、つまり1000文字表示できます。

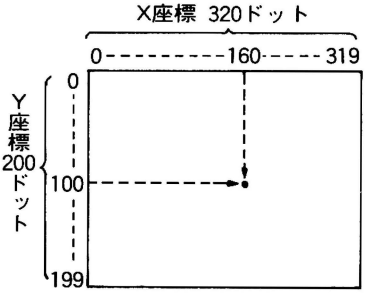


[例]右図の座標（20、５）にＡを表示します。

```
10  CURSOR  20, 5
20  PRINT"A"
```

③グラフィック座標

画面２のグラフィックモードで使われる座標です。Ｘ座標、Ｙ座標の考え方は、文字座標と同様です。Ｘ座標方向、Ｙ座標方向それぞれ320ドット、200ドットあります。



[例]右図の座標（160、100）に赤点を描きます。

```
10  SET [2, 0] 160, 100
```

5．ファイル

コンピュータは、周辺機器（クイックディスク・カセットテープなど）との間で、プログラムやデータのやりとりをすることができます。このプログラムやデータの単位をファイルと呼びます。



（1）ファイルの分類

ファイルには、プログラムファイルとデータファイルがあります。

ファイル
- プログラムファイル……プログラムをそのまま保存します。
  - BASIC プログラム……BTX ファイル
  - 機械語プログラム ……OBJ ファイル
- データファイル……数値・文字などをデータとして保存します。
  - BASIC シーケンシャルデータ ……BSD ファイル

（2）周辺機器への入出力

ファイルを扱う場合、どのような周辺機器（装置）に対してどのような名前のファイルの入出力を行うかを指定する必要があります。

"装置名：ファイル名"
- 8 文字までの英数字
- コロン（忘れないように。）
- 入出力させたい装置名（device name）
  - QD　：……クイックディスク
  - CMT：……カセットテープ
  - RAM：……RAMファイル　　など

［例］　"QD：SHARP"

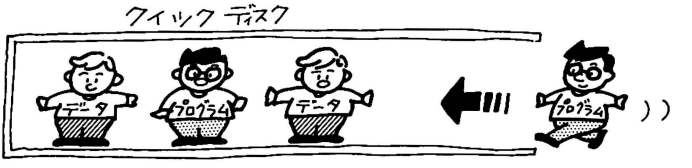
（3）ファイルについて

カセットテープやクイックディスクなどの記憶装置上のデータをファイルと呼ぶ場合もありますが、ここでは、プリンタ、RS-232C など装置に対する入出力データも含めてファイルと呼んでいます。

●クイックディスク上のファイル

クイックディスクは、カセットテープと同じように、プログラムやデータなどを順番に書き込んで行きます。書き込まれるファイルはつねに最新の部分に追加され、途中に追加することはできません。また、ファイル単位で消去することもできません。（ただし、DELETE UTILITY を使用することにより可能となります。）

クイックディスク上のファイルを消去する場合は、INIT 命令でクイックディスクを初期化（イニシャライズ）して、全てのファイルを消去する必要があります。なお、新しいクイックディスクを使うときにも、初期化することが必要です。



※なお、ファイルを読み出す場合も、前から順に探して行きます。

注1) RAM ファイル(MZ-1R18)は、RAM ファイルとして使用されるだけではなく、プリンタ用のバッファ（データを一時記憶しておくもの）としても使用されます。ですから、64K バイトのメモリをすべて RAM ファイルとして使用できるわけではありませんので注意が必要です。なお、この設定には、**INIT** 命令を使用します。

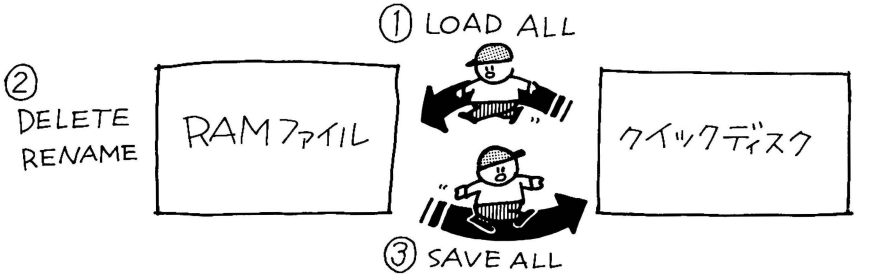
RAM ファイル（MZ-1R18）

管理用メモリ	←（16バイト）
RAM ファイル用メモリ	←（INIT 命令で容量を指定）
プリンタバッファ用メモリ	←（64K－(16＋RAM ファイル用メモリ)バイト）

注2) クイックディスクでは、**DELETE** 命令（ファイルの取り消し）や **RENAME** 命令（ファイルの名前の変更）は使用できませんが、RAM ファイルを使って、これの処理をすることができます。

つまり、**LOAD ALL** 命令でクイックディスクの内容をすべて RAM ファイルに読み出してから、RAM ファイル上で **DELETE／RENAME** 命令を実行し、そのあと、**SAVE ALL** 命令で RAM ファイルの内容をクイックディスクに書き込むわけです。

ただし、RAM ファイル用のメモリ容量およびクイックディスクの記憶容量が小さいとエラーとなりますので注意が必要です。



●ファイルの入出力装置一覧表

装置名	クイック ディスク	RAM ファイル	カセット テープ	ディス プレイ	プリンタ	RS-232C
	QD：	RAM：	CMT：	CRT：	LPT：	RS1~4：
INIT	○	○	×	○	○	○
DIR	○	○	×	×	×	×
DEFAULT	○	○	○	○	○	○
DELETE	×	○	×	×	×	×
RENAME	×	○	×	×	×	×
LOAD	○	○	○	×	×	×
SAVE	○	○	○	×	×	×
CHAIN	○	○	○	×	×	×
MERGE	○	○	○	×	×	×
WOPEN #	○	○	○	×	×	○
ROPEN #	○	○	○	×	×	○
LIST	×	○	×	○	○	×
RUN	○	○	○	×	×	×
INPUT #	○	○	○	×	×	○
PRINT #	○	○	○	×	○	○

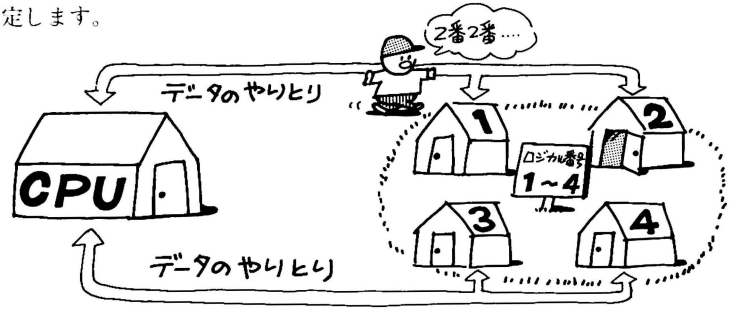
（4）ファイルの使い方

ファイルにはデータファイルとプログラムファイルがありました。その使いかたはどちらも一般的には次のようになります。

①データファイル

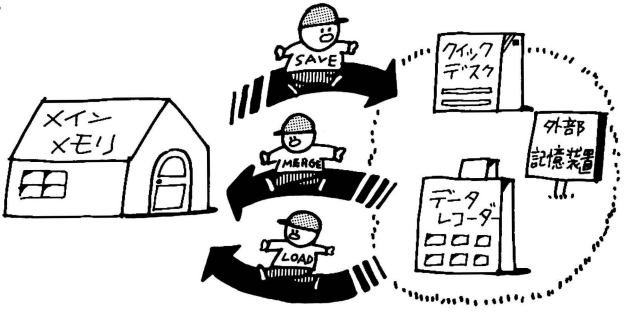
ファイルをオープンして、ロジカル番号で指定された周辺機器と接続します。	⇨	データの読み出し、書き込みをします。	⇨	処理が終われば、クローズして、周辺機器との接続を切ります。

注）ロジカル番号と周辺機器との対応づけは、WOPEN、ROPENにおいて任意に指定します。



②プログラムファイル

特にプログラムファイルでは、外部記憶装置との間で、LOAD、SAVE、MERGE、を行います。



SAVE……プログラムを外部記憶装置に記憶させる。

LOAD……プログラムを外部記憶装置からメモリに移す（今までに記憶されていたプログラムは消去されます。）

MERGE…メモリ上のプログラムと、外部記憶装置から取り出してきだプログラムをつなぐ。

（5） WOPEN と ROPEN

書き込みには **WOPEN**（Write Open）、読み込みには **ROPEN**（Read Open）を使います。

[例1]

```
10  WOPEN #1, "QD:ABCD" ……開く
20  PRINT #1, 8 ……処理
30  CLOSE……閉じる
```

行番号10で、出力用ファイルとして、クイックディスクにある ABCD というファイルをオープン（開く）します。クイックディスクのロジカルナンバは1番とします。

行番号20で、ABCD というファイルにデータの8を書き込みます。

行番号30で、ファイルをクローズ（閉じる）します。

[例2]

```
10  ROPEN #2, "QD:EFGH" ……開く
20  INPUT #2, A ……処理
30  CLOSE……閉じる
```

行番号10で、入力用ファイルとして、クイックディスクにあるＥＦＧＨというファイルをオープン（開く）します。

行番号20で、ＥＦＧＨというファイルから変数Ａにデータを入力します。

行番号30で、ファイルをクローズ（閉じる）します。

注）起動時の装置名として、**QD**：をとっています。つまり、装置名を指定しなければ、クイックディスクを指定したとみなされます。

6．音楽を演奏する

本機には、オーディオ端子と内蔵スピーカがあり、BASICを使っていろいろなメロディーを作曲することができます。また、オーディオ製品と組み合わせて使うこともできます。

音域は6オクターブあり、音の強弱・長さ・テンポの指定が可能で、和音は6重和音まで演奏することができます。

演奏はMUSIC命令（194ページ）を使います。演奏用のデータ文字列で指定した音符にしたがって6つまでのパートを同時に演奏させることができます。

●演奏用のデータ文字列の並べ方

オクターブ	音名	音長

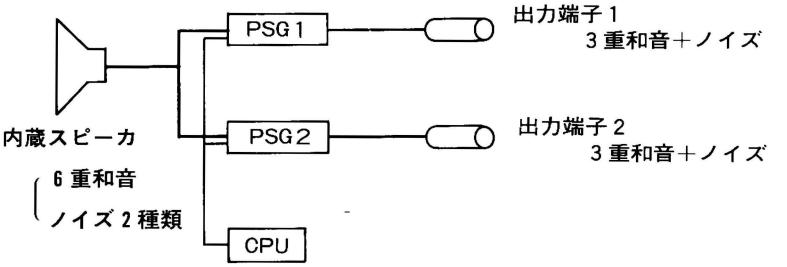
1　　2　　3 ……1 2 3……1 2 3……くり返し

オクターブや音長・テンポが変わる場合は、そのつど指定します。テンポの指定はTEMPO命令（199ページ）を使います。

演奏以外の音はSOUND命令（197ページ）とNOISE命令（200ページ）を使います。SOUND命令は指定した音を指定した長さで鳴らすことができます。爆発音などの効果音は、SOUND命令を使って簡単に作ることができます。また、SOUND命令を使うと、PSG（プログラマブル・サウンド・ジェネレータ）を直接制御してスピーカを鳴らすことができ、簡単なシンセサイザとして使うことも可能です。NOISE命令も効果音を作るときに使います。音の指定には、MUSIC命令と同じく演奏用のデータ文字列を使用します。



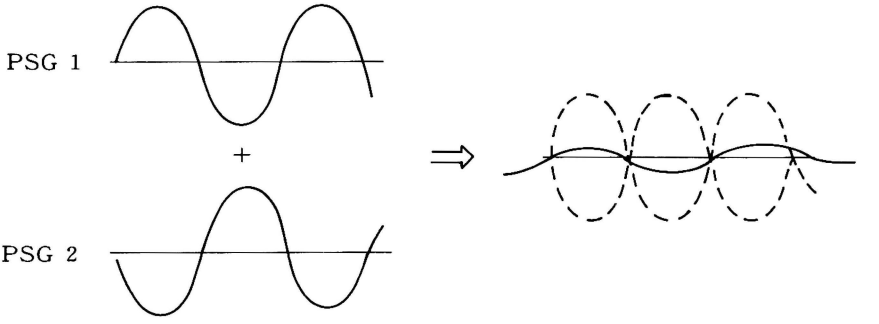
（1）ステレオを楽しむ



MZ-1500 は、PSG（プログラマブル・サウンド・ジェネレータ）を 2 つもっています。PSG は、6 オクターブの音域と 3 重和音、ノイズ 1 種類を鳴らすことができますから、MZ-1500 は、6 重和音とノイズ 2 種類を演奏できることになります。また、PSG 1 個につき 1 つずつ出力端子が用意されているので、2 個の外部スピーカと接続することができ、ステレオ演奏も楽しめます。

注) MZ-1500 では、2 つの PSG を使用して、音を発生させていますが、同じタイミングで 2 つの音を発生させているのではなく、少しタイミングをずらしています。

ですから、同じ音を別の PSG から発生させる場合、音の周波数によっては偶然に位相が90°近くずれて、音が小さくなる可能性があります。



このように、音が極端に小さくなるときは、一方の音を休止符に変更するなどして対応してください。

（２）音楽の演奏

演奏は、MUSIC 命令と TEMPO 命令を使用します。MZ-1500 は、６オクターブの音域をもち、６重和音まで演奏することができます。

MUSIC 命令では、演奏中のデータ文字列を使って音を指定します。メロディーの各パートの間はセミコロン（；）、１つのメロディーから次のメロディーに移るときはコンマ（，）で区切ります。演奏速度は、TEMPO 命令で指定します。１が最も遅く、７が最も速くなります。メロディーは音の続きで表し、オクターブは、

＋　（１つ上のオクターブ）
　　（基準のオクターブ）
－　（１つ下のオクターブ）

の記号で表し、基準のオクターブの高さはＯ０からＯ６で指定します。（Ｏ２の場合ラの音が440Hz）音名・音長は次のようになります。

①［書式］

MUSIC 〈メロディー１〉［；〈メロディー２〉］……［；〈メロディー６〉］，
　　〈メロディー１〉［；〈メロディー２〉］……

〈メロディー--〉＝〈音名〉［〈音長〉］

TEMPO 〈数値データ〉（数値データは１～７の整数）

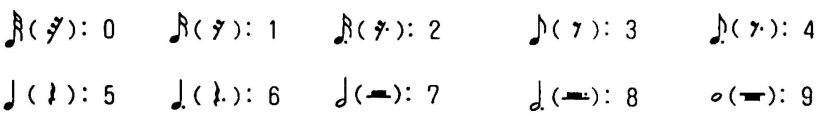
・音名

音	ド	レ	ミ	ファ	ソ	ラ	シ	休符
音名	C	D	E	F	G	A	B	R

半音上げる場合は音名の前に#記号をつけます。
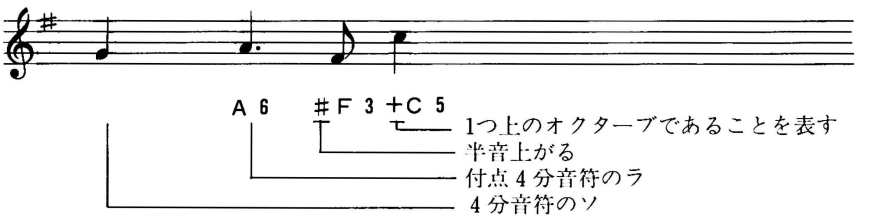
・音長

音長は、０～９の数字で指定します。前の音と同じ長さの場合は省略できます。ただし、＂で囲まれる部分の最初の音の音長は省略できません。

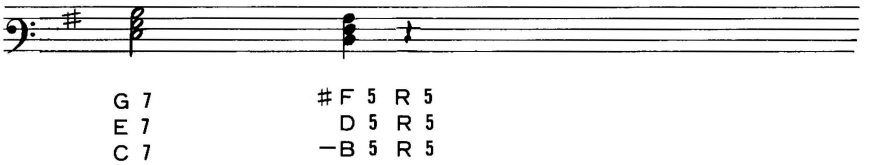


②音楽プログラムの例

（パートA）



（パートB）



[例]

```
10  A$="O2G5A6#F3+C5"
                基準のオクターブの高さを指定
20  B1$="O1G7#F5R5"
30  B2$="E7D5R5"
40  B3$="C7-B5R5"
50  MUSIC  A$;B1$;B2$;B3$
           メロディー1  メロディー3
60  END        メロディー2      メロディー4
```

行番号50を次のように変えると

```
50  MUSIC  A$;B1$;B2$;B3$,A$;B1$;B2$;B3$
                          メロディーをつなぐ場合は
                          コンマで結びます。
```

行番号10～40のフレーズが2回くり返されます。

（3）音を作る

演奏するのではなく、単に音を発生させたいときには、SOUND 命令や NOISE 命令を使います。爆発音などの効果音を作りたい場合、SOUND 命令や NOISE 命令を使うと非常に簡単に作ることができます。SOUND 命令は、指定された音程の音を、指定された長さの間スピーカで鳴らし、簡単なシンセサイザとして使うことができます。NOISE 命令は、指定された音程の雑音を、指定された長さの間、スピーカで鳴らすもので、効果音を作るときに使用されます。NOISE 命令では、2つの雑音を同時に鳴らすことができ、音の指定には演奏用のデータ文字列を使用します。音域は、6オクターブまで使えます。

[書式]　SOUND 音程、音長

（音域）＝12～83までの整数で指定します。

NOISE 〈メロディー1〉[；〈メロディー2〉]
[，　〈メロディー1〉[；〈メロディー2〉]] ……

〈メロディー〉＝〈音名〉[〈音長〉]

[例1]

```
10 FOR I=12 TO 83
20 SOUND I, 10
30 NEXT I
```

[例2]

```
10 NOISE"C5F6";"E5A6"
```

これは次のようにもできます。

```
10 NOISE"C5";"E5","F6";"A6"
```

7．プリンタを使う

コンピュータとの対話はおもにディスプレイ画面を通じて行われますが、これをプログラムとしてカセットやディスクなどに残しておかないかぎり、電源を切ってしまえばそのプログラムは残りません。プリンタを使うことによって、演算結果やプログラムリストを印字したり、一画面に表示しきれない長いプログラムを一度に見ることができます。

（1） 2つのモード

プリンタを使用するときに注意することにモードの指定があります。モードには、計算結果やプログラムリストなどの、文字や数字を印字する場合に使用するテキストモードと、図形を描く場合に使用するグラフィックモードの2種類があります。テキストモードでは、1行を何文字にするかによって印字の大きさを変えることができます。2つのモードの指定・切り換えは、BASICの命令を使用します。

（2）使用プリンタによるモードの使いわけ

①カラープロッタプリンタ

テキストモード、グラフィックモードの両方を使用することができます。ただし、一方のモードのとき、他方のモード命令は使用することができません。電源を入れた時点ではテキストモード（1行40文字）にセットされています。

②ドットプリンタなどの他のプリンタ

テキストモード以外は、使用できません。



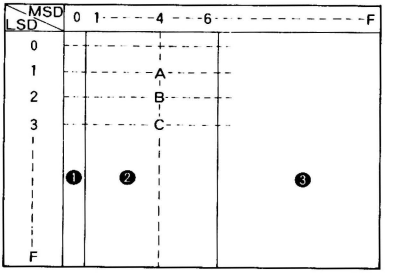
（3）各モードの設定命令

モードの種類	モード設定命令	使用可能プリンタ	モード中に使用できる命令
テキストモード 1行40字	PMODE TN	カラープロッタ プリンタ(MZ-1P09) ドットプリンタ (MZ-1P08)	PTEST PSKIP PAGE PCOLOR
テキストモード 1行26文字	PMODE TL	カラープロッタ プリンタ(MZ-1P09) ドットプリンタ (MZ-1P08)	※ LIST/P ※ PRINT/P ※ DIR/P
テキストモード 1行80行	PMODE TS	カラープロッタ プリンタ(MZ-1P09) ドットプリンタ (MZ-1P08)	※：印のみドットプリンタで使用できます。 （他はカラープロッタプリンタ用命令です）
グラフィック モード	PMODE GR	カラープロッタ プリンタ(MZ-1P09)	PLINE RLINE PMOVE RMOVE PHOME HSET GPRINT AXIS PCIRCLE PCOLOR

各モードにおける命令については各プリンタのマニュアルを参照してください。

（4）アスキーコード表の見方

MZ-1500 側から、プリンタにデータを送るには、下表のようなアスキーコードを使用します。



注)アスキーコード表では、表のMSD（行）は上位4ビット、LSD（列）は下位4ビットを表しています。
たとえば、AはMSDが4、LSDが1なので16進数の**41**と表されます。(10進数の65)

アスキーコード表は次の3つに分類できます。

❶プリンタをコントロールするためのコード

❷英数をあらわすコード

❸カタカナ・記号をあらわすコード

表中の❶はプリンタをコントロールするためのコードです。

表中の❷は英数・記号、表中❸はカタカナ・記号を表すアスキーコードです。

ABC とプリンタで印字するとき、アスキーコードを使用しない場合は次のようになります。

[例1]

```
10 PRINT/P "ABC"
RUN
```

アスキーコードを使うと

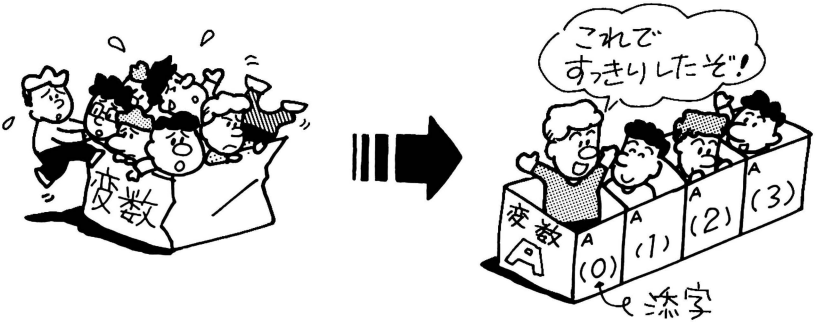
[例2]

```
10 PRINT/P CHR$(&41);CHR$(&42);CHR$(&43)
RUN
```

のようになり、同じ結果を得ることができます。

8．配列

変数とは、一時的にデータを入れておく箱であると前に説明しました。ところが、これでは多くのデータをまとめて使うときに不便です。たとえば、統計などで数多くの数値がでてきた場合、それらの数値をまとめると扱いやすくなります。このように、いくつもの変数を一組にまとめて並べたものを配列といいます。



配列は、データを入れる箱をいくつも持っていて、それぞれの箱には他の箱と区別するために番号がついています。この番号のことを添字といいます。また、配列にも普通の変数と同じように、名前をつけて使用します。この名前のことを配列名といいます。

箱の数が多い場合、１つの添字だけでなく、２つの添字を組み合わせて表すことがあります。添字が１つの配列を１次元配列、添字が２つの配列を２次元配列といいます。４次元配列まで行えます。

例　A（3，10，15，2）

プログラム中で配列を使う場合は、必ず最初にDIM命令で配列名と大きさ(プログラムの大きさや配列に格納されるデータの数によって制限を受けます。)を宣言しなければなりません。また、添字は（　）で囲んで表します。

1次元配列を使った簡単な例で配列をみてみましょう。

合計3回行ったテストの点数を入力して、各回の点数および3回の合計点と平均点を出します。

[例]

```
 10  DIM  A(5) ……配列の宣言
 20  FOR  I=1  TO  3
 30  PRINT  I;"カイメ  ノ  テンスウ"
 40  INPUT  A(I)
 50  A(4)=A(4)+A(I)
 60  NEXT  I
 70  A(5)=A(4)/3
 80  FOR  I=1  TO  3
 90  PRINT  I;"カイメ",A(1);"テン"
100  NEXT  I
110  PRINT  "ゴウケイ",.A(4);"テン"
120  PRINT  "ヘイキン",A(5);"テン"
130  END
```

行番号10で配列の宣言を行っています。これより、A(0)，A(1)，…，A(5)までの合計6の配列変数が普通の変数と同じように扱えます。

行番号20～60で、A(1)，A(2)，A(3)にそれぞれ、1回目、2回目、3回目の点数が代入され、A(4)には合計点が代入されます。(この例ではA(0)は使用していません。)

行番号70で合計点A(4)を回数3で割ったもの、つまり平均点をA(5)に代入します。

行番号80～100で各回の点数としてA(1)、A(2)、A(3)が表示されます。番号110では合計点としてA(4)が表示され、行番号120では平均点としてA(5)が表示されます。

9．乱　数

最近、「乱数」という言葉が、よくプロ野球などで使われています。しかし、乱数とは何かというと、あまり知られていません。そこで、次のような実験をしてみれば、乱数とは何かということがわかるでしょう。

用意するもの：サイコロ、エンピツ、紙 実験方法　　：サイコロをころがし、出た目を紙に記録します。 　　この作業を10回くりかえしてください。

その結果、

3　4　1　3　5　5　2　4　1　6

というような目が出たとします。これらの数字は、何の規則もないもので、デタラメ（ランダム）に出たものです。それを乱数と呼びます。

乱数の使い方は、一般的に知られているものとして、ゲームなどがあります。テレビゲームによく登場するキャラクタはデタラメ（ランダム）に動いていますが、これはコンピュータが発生する乱数に基づいて動いているのです。

BASICで、乱数を発生させる方法は、次の通りです。

```
RND (X)
```

で、乱数が発生します。RND関数の値は、0.00000001～0.99999999までの値をとります。また、Xは数値データ（数字）で、以下のような意味があります。

- **X≦0の場合**：プログラム実行中のたびに同じ乱数を発生させます。これを再現性のある乱数といい、プログラムやデータの確認などに使われます。同じパターンの乱数を発生させるには、最初にX≦0の乱数を発生させ、次からはX＞0の乱数を発生させればよいわけです。
- **X＞0の場合**：プログラム実行のたびに新しいパターンの乱数が発生します。

一般に、n～N（n＜N）までの乱数を発生させる場合、次の式を使います。

```
INT (RND (X) * (N-n+1)) +n
```

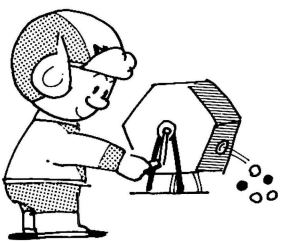
[例1] 78から97までの整数を5個出力するプログラム

```
10  FOR I=1 TO 5
20  A=INT(RND(1)*(97-78+1))+78
30  PRINT A;
40  NEXT I
50  END
```

注)20行目の97—78＋1を20にしてもよい

[例2] ジャンケンゲーム

```
10  PRINT"1.グー  2.チヨキ  3.パー"
20  INPUT"ナニヲ ダシマスカ":A
30  GOSUB 200
40  ON B GOTO 50,60,70
50  PRINT"ワタシハ グー":GOTO 80
60  PRINT"ワタシハ チヨキ":GOTO 80
70  PRINT"ワタシハ パー"
80  IF A=B THEN PRINT"アイコ"
90  IF A—B=—1 OR A—B=2 THEN PRINT
    "アナタノカチ"
100  IF A—B=1 OR A—B=—2 THEN PRINT
     "アナタノマケ"
110  GOTO 10
200  B=INT(RND(1)*3)+1
210  RETURN
```



10. ジョイスティック

ジョイスティックでは座標の指定や、スイッチからの入力ができますので、ゲームなどに最適です。MZ-1500には2個のジョイスティック端子が付いていますので、ジョイスティック(MZ-1X03：オプション)を2個使って、2人で遊ぶこともできます。

ジョイスティックにはJOY関数を使用します。

関数	意味	関数の値
JOY(0)	スティック1のX軸の値	0～255
JOY(1)	スティック1のY軸の値	0～255
JOY(2)	スティック2のX軸の値	0～255
JOY(3)	スティック2のY軸の値	0～255
JOY(4)	スティック1のスイッチ1	0(OFF)か－1(ON)
JOY(5)	スティック1のスイッチ2	0(OFF)か－1(ON)
JOY(6)	スティック2のスイッチ1	0(OFF)か－1(ON)
JOY(7)	スティック2のスイッチ2	0(OFF)か－1(ON)

[例]

```
 10  PRINT "■"
 20  GOSUB 100
 30  IF S1=0 THEN A=7
 40  IF S1=-1 THEN A=2
 50  CURSOR X,Y
 60  PRINT[A, 0]"♥"
 70  FOR A=0 TO 20:NEXT A  ←表示時間を長くします。
 80  GOTO 10
100  REM JOYSTIC サブルーチン
110  X=JOY(0)/255*39 ←X軸の値を横40行分に直します。
120  Y=JOY(1)/255*23 ←Y軸の値を縦24行分に直します。
                          (最後の行には表示しません。)
130  S1=JOY(4)←――――スイッチ1のON・OFFを調べます。
140  RETURN
```

ジョイスティック1を動かした方向に文字(♥)を動かすプログラムです。スイッチ1を押すと文字の色が赤になります。

11. 通信

コンピュータの間で通信を行うときには、RS-232Cが広く使用されています。RS-232Cとは、コンピュータと端末装置を結ぶインターフェイスの規格であり、EIA（米国の電子工業会）で制定されています。モデムや音響カプラなどで使用されています。

RS-232Cを使って通信を行うには次のようにします。ただし、RS-232Cインターフェイス（MZ-8BI03）を、内蔵スロットまたは拡張ユニットにセットすることが必要です。

①INIT命令で設定を行います。

②通信の入出力用にファイルをオープンにします。
（WOPEN #、 ROPEN #）

③データを入出力します。
（INPUT #、 PRINT #）

④ファイルをクローズ（CLOSE）します。

ファイルに使う装置名は次の通りです。

装置名		
RS 1	チャンネルA	インターフェイスを拡張ユニットにセットした場合
RS 2	チャンネルB	
RS 3	チャンネルA	インターフェイスを内蔵スロットにセットした場合
RS 4	チャンネルB	

[送信例]

```
10  INIT"RS1:CHR$($03)+CHR$($87)"
20  WOPEN #1,"RS1:"
30  PRINT #1,"アナタ ガ スキデス"
40  CLOSE #1
50  END
```

[受信例]

```
10  INIT"RS1:CHR$($03)+CHR$($87)"
20  ROPEN #2,"RS1:"
30  INPUT #2,A$
40  PRINT A$
50  CLOSE #1
60  END
```

12. コンピュータの内部を扱う

（1）機械語を使う

機械語とは、コンピュータがそのままの形で理解、実行できる言葉のことです。コンピュータにとっては、母国語のようなものです。機械語は、0と1から成る2進数8ケタを16進数2ケタに直した形で表現され、人間にとってはわかりにくい言葉です。そこで私達は、人間にとってわかりやすいBASICと呼ばれる言葉を使ってきました。しかし、コンピュータはこのBASICで書かれたプログラムを直接実行することはできません。BASICのプログラムを1行ずつ機械語に翻訳して、実行しているのです。それに、BASICを機械語に翻訳しているのも、機械語で書かれたプログラムです。

機械語を使う利点は以下の2点に要約できます。

●一度翻訳しなければならないBASICプログラムよりも、機械語プログラムの方が実行速度が速い。

●BASICでは表現できない特殊な操作をコンピュータに行わせることができる。

たとえば、インベーダーゲームのような条件反射型の、実行の速さが要求されるプログラムでは、ほとんどが機械語で書かれています。

機械語を扱うときには、240ページの「モニタコマンド」を参照してください。

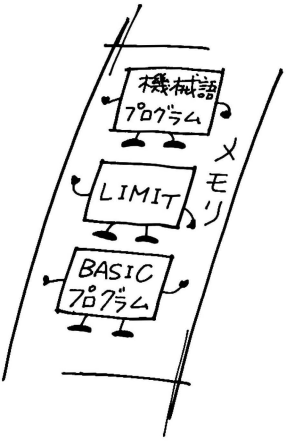


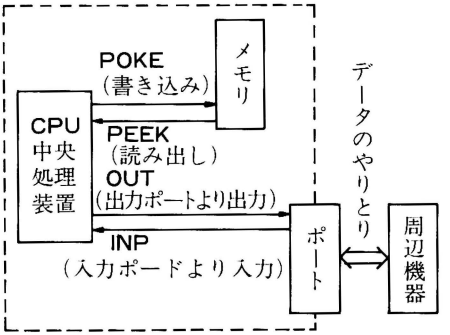
（2） BASIC から機械語へのアプローチ

コンピュータの内部を扱うときには、BASIC では次のような命令を使います。

- LIMIT　BASIC のプログラムと機械語のプログラムを連結（リンク）して使うときに、BASIC のメモリ領域を制限して、機械語プログラムが使うメモリ領域を確保する。
- PEEK　メモリ上の任意のアドレスのデータを読み出す。
- POKE　メモリ上の任意のアドレスにデータを書き込む。
- USR　利用者が定義した機械語サブルーチンを呼び出す。
- INP　入力ポートからデータを読み込む。
- OUT　出力ポートからデータを出力する。

コンピュータの頭脳である CPU（中央処理装置）では、データの入出力を記憶装置であるメモリと周辺機器などに対して行います。周辺機器などへの入出力はポートを通して行います。PEEK、POKE、INP、OUT はこれらの入出力を行う命令です。





第4章　命令・ステートメント

書式 では、記述方法について説明します。ただし、記号には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の総称を表します。
ただし、一番広い意味を表すものです。
（注）
〈変数〉は、〈数値変数〉と〈文字変数〉を含みます。
〈変数〉は、〈配列要素〉も含みます。
〈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。

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

本書中では、キーの入力のしかたとして次の略式を使っています。

SHIFT ＋ BREAK：左側に示されたキーを押しながら右側のキーを押すことを意味します。

└ 押しながら

1 基本命令

AUTO (auto・オート)

行番号を自動的に発生させる

書式 AUTO [〈スタート行番号〉] [, 〈増分〉]

省略形 A.

説明

・AUTO 命令を実行すると、BASIC は行番号を表示してステートメントの入力を待ちます。

・その行を入力して、最後に CR キーを押すと、改行して次の行番号(前の行番号＋増分)を表示してステートメントの入力を待ちます。

・このように、AUTO 命令はスタート行番号から一定の増分がある行番号を発生し、プログラム作成がスムーズに行えるようにしています。

・デフォルト値(なにも指定しなかった場合に、BASIC が決める値)は、スタート行番号、増分ともに10です。AUTO 命令を解除するには SHIFT ＋ BREAK キーを押します。

例

```
AUTO 300,5
```

…行番号を、300、305、310、315…と増分 5 で順番に発生します。

```
AUTO 100
```

…行番号を、100、110、120……と増分10で順番に発生します。

```
AUTO,20
```

…行番号を、10、30、50、70……と増分20で順番に発生します。

```
AUTO
```

…行番号を、10、20、30、40……と増分10で順番に発生します。

DELETE（delete・デリート）

プログラムの一部を消去する

書　式　DELETE［〈スタート行番号〉］［－］［〈エンド行番号〉］

省略形　D.

説　明
・プログラム中の〈スタート行番号〉から〈エンド行番号〉までの部分を消去します。

例

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

```
DELETE　－150
```
……プログラムの始めから行番号150までのプログラムを消去します。

```
DELETE　400－
```
……行番号400から最後までのプログラムを消去します。

```
DELETE　150
```
……行番号150の行を消去します。

LIST（list・リスト）

メモリに記憶されているプログラムのリストを画面上に表示する

書　式　LIST［／P］［〈スタート行番号〉］［－］［〈エンド行番号〉］

省略形　L.

説　明
- 画面上に、プログラムのリストを表示します。
- リスト表示の実行中にスペースキーを押さえると、リスト表示が停止します。もう一度スペースキーを押すとリスト表示が再開されます。また、SHIFT ＋ BREAK キーを押すと、リスト表示が中断されて命令待ち状態に戻ります。
- ／Pを指定すると、画面上にではなく、プリンタに出力します。

例

・全部をリストする場合

```
LIST
```

……〈スタート行番号〉〈エンド行番号〉を省略するとプログラムの全体を表示します。

・一部のリストを表示させる場合

```
LIST 300－500
```

……〈スタート行番号〉(300) と〈エンド行番号〉(500) をマイナス記号［－］で結ぶと、行番号300から500までのプログラムリストを表示します。

```
LIST 250－
```

……〈スタート行番号〉(250) の後に、マイナス記号［－］を記述すると行番号250以降のプログラムを表示します。

```
LIST 150
```

……行番号150の文のみを表示します。

SEARCH（search・サーチ）

指定した文字列を含む行を探し出す

書式　SEARCH［／P］〈文字データ〉

説明
- BASICのプログラム中で、〈文字データ〉で示す文字列を含む行を探し出し、画面上に表示します。
- 実行中にスペースキーを押すと、表示が一時停止し、もう一度押すと表示が再開されます。
- SHIFT ＋ BREAK で、実行が中止されます。
- 〈文字データ〉の、アルファベットの大文字と小文字、ひらがなとカタカナは区別できません。たとえば、" abc "も、" ABC "と同じ文字列とみなします。
- 〈文字データ〉として、ダブルクォーテーションマーク（"）を使用する場合は、CHR$（34）を使用します。
- ／Pを指定すると、画面上ではなく、プリンタに印字します。

例

```
SEARCH "ABC"
```
……プログラム中で、" ABC "という文字列を含む行を、探し出し画面上に表示します。

```
SEARCH "PRINT"+CHR$(34)+"A"+CHR$(34)
```
……文字列PRINT " A "を含む行を探し出します。

RENUM (renumber・リナンバー)

プログラムの行番号を変更する

書式　RENUM [〈新行番号〉] [, 〈旧行番号〉] [, 〈増分〉]

省略形　REN.

説明

・プログラムの中の行番号を付けかえる命令です。(GOTO、GOSUB、ON～GOTO、ON～GOSUB)などのジャンプ先行番号も同時に変更されます。

・新行番号と旧行番号を指定する場合、新行番号は旧行番号より大きくなければいけません。 また、新しく発生する行番号が65535より大きくなる場合は、エラーとなります。

例

```
RENUM
```
……現在のプログラムの行番号の並びを行番号10から増分10で増える行番号の並びに変更します。

```
RENUM 100
```
……行番号100から増分10で増える行番号の並びに変更します。

```
RENUM 100,50,20
```
……行番号50から始まるプログラムを、行番号100から増分20で増える行番号の並びに変更します。

(実行前)		(実行後)
50 A=1		100 A=1
60 A=A+1	→	120 A=A+1
70 PRINT A		140 PRINT A
100 GOTO 60		160 GOTO 120

NEW （new・ニュー）

メモリに記憶されているプログラムを消去し、BASICの初期化を行う

書式　NEW

説明

・NEW命令を実行すると、BASICエリア内にあるプログラムを消去し、同時に変数エリア、配列エリアなどのプログラムワークエリアも消去します。

・LIMIT文によってBASICエリアが制限されている場合は、BASICエリア内だけが消去され、機械語プログラムは消去されません。

・LOAD命令の実行時にもローディングの前に、NEW命令と同じ操作が行われます。（LIMIT文の制限は有効）

例

```
NEW
```

……プログラムを消去します。

NEW ON （new on・ニュー オン）

ユーザープログラムエリアを拡張する

書 式

NEW ON { 1 / 2 }

説 明

BASICプログラムエリアを拡げる場合に使用します。

NEW ON命令の指定により、ボイスボードやプロッタプリンタに関するBASICインタープリタの一部を削除します。

オプションのボイスボード（MZ-1M08）に関する命令を使用しない場合は1を指定してください。

オプションのボイスボード（MZ-1M08）とプロッタプリンタ（MZ-1P09）に関する命令を使用しない場合は2を指定してください。

例

NEW ON1

……ボイスボードに関するBASICインタープリタの一部を削除します。

CLR（clear・クリア）

変数、配列をすべて未定義状態とする

書　式　CLR

説　明

・変数については、数値変数をすべて0に、文字変数の内容をすべてNull（" "）にします。

・配列は、すべての配列を未定義の状態に戻し、またすでに実行されているDIM文（129ページ）をすべて無効とします。したがって、CLR文の実行後に、配列を使用するには、新たにDIM文によって必要とする配列の定義を行わなければなりません。

・すでに実行されているDEF　FNもすべて無効とします。したがって、CLR文の実行後に、定義関数を使用するには、新たにDEF FN文によって必要とする定義関数の定義を行わなければなりません。

・FOR～NEXTの中、およびサブルーチンの中にCLR命令を置くことはできません。

例

```
10  A＝3
20  B＝12
30  C$＝"SHARP"
40  PRINT  A；B；C$
50  CLR
60  PRINT  A；B；C$
70  END
```

……CLR文の実行により、変数A、B、C$が未定義の状態になります。数値変数A、Bは0になり、文字変数C$はNull（" "）になります。

CONT（continue・コンティニュ）

中断またはストップしたプログラムの実行を継続する

書　式　CONT

省略形　C.

説　明

・プログラムの実行中に SHIFT ＋ BREAK キーの入力、またはSTOP文の実行によって、プログラムが中断した状態にあるときプログラムの実行を継続して再開します。

・END で終了している場合は使用できません。

・CONT 命令が使用できるのは、Ready の後に.（ピリオド）が表示されている場合です。

例

```
10  INPUT  A, B
20  C=A*B
30  STOP
40  PRINT  "A*B=" ; C
50  END
```

……行番号30の STOP 文によりプログラム実行が一時停止します。CONT 文を実行すると A＊B の値 C を行番号40の PRINT 文で画面に表示します。

RUN（run・ラン）

プログラムを実行する

書式　RUN［〈スタート行番号〉］

省略形　R.

説明
- スタート行番号を書かないRUN命令では、始めにCLR文が実行されたのと同じ様に、変数エリアと配列エリアは消去され、その後、先頭行からプログラムが実行されます。一方RUN〈スタート行番号〉命令では、消去する操作は行われません。
- BASICは、プログラム中のEND文、STOP文の実行またはプログラムの終りで命令待ち状態に戻ります。(途中でエラーが発生した場合や SHIFT ＋ BREAK キーが押された場合も、命令待ち状態に戻ります。)
- プログラム中でUSR関数を使っている場合は、ユーザエリアの機械語プログラムやモニタサブルーチンなどへ一時コントロールが移されます。(この場合、コントロールはBASICの管理を離れることになり、SHIFT ＋ BREAK キー操作などは受けつけなくなります。)
- スタート行番号のかわりにラベルの指定を行うことはできません。RUN " 文字列 " とした場合、その文字列を名前にもつプログラムを読み込み、ただちに実行する意味にとられます。

例

```
RUN
```
……プログラムを、その先頭行から実行します。

```
RUN  200
```
……行番号200から実行します。

参考　RUN（151ページ）

CLS （clear screen・クリアスクリーン）

画面を消去する

書　式

CLS { 1 / 2 / 3 }

説　明

・CLS 1 ならば、画面 1 を消去します。
・CLS 2 ならば、画面 2 を消去します。
・CLS 3 ならば、画面 1 と画面 2 を同時に消去します。
・CLS 1，3 では CONSOLE 命令で指定された範囲に関係なく画面 1 を全て消去します。

例

```
10 CLS 1
```

……画面 1 を消去します。

参　考

CONSOLE
INIT
画面 1 、画面 2 については62ページ参照。

CONSOLE（console・コンソール）

スクロールする範囲を指定する

書　式　CONSOLE［〈開始行〉，〈行数〉［〈開始ケタ〉，〈ケタ数〉］］

省略形　CONS.

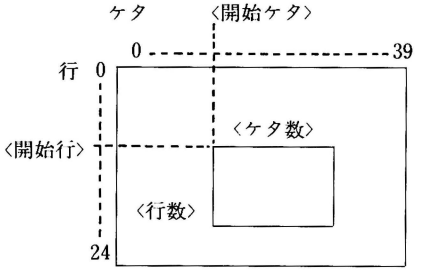
説　明

・スクロールする範囲を指定します。PRINT " C " によって消される範囲は、このCONSOLE文で指定されている行、またはケタのみです。

・画面1に対してのみ有効です。

・PLOT ON命令を実行時には、この命令は使用できません。

・〈行数〉，〈ケタ数〉が小さくなると、画面の編集が困難になるのでスクロール範囲には注意してください。なお、CTRL ＋ D キーでスクロール範囲の解除を行うこともできます。



例

```
CONSOLE  0，25，0，40
```
……全画面をスクロール範囲にします。

```
CONSOLE  5，15，5，30
```
……スクロールの範囲を5行目から15行、5ケタ目から30ケタにします。

```
CONSOLE  10，5
```
……スクロールの範囲を10行目から5行とします。

CURSOR （cursor・カーソル）

座標を指定してカーソルを移動させる

書　式

CURSOR 〈X 座標〉, 〈Y 座標〉

〈X 座標〉：0～39
〈Y 座標〉：0～24
} カーソル座標

省略形

省略形　CU.

説　明

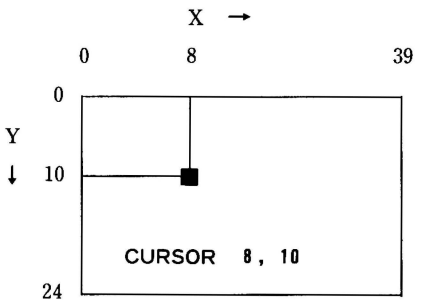
・画面上の座標位置を指定して、そこへカーソル位置を移動させます。PRINT 文、INPUT 文とペアで使用し、画面上の左上すみを（0，0）、右下すみを（39，24）とした座標（40ケタ×25行）の範囲の任意の位置にカーソルを移動します。

・X，Y の値が整数でない場合は、小数点以下は切り捨てられます。

例

```
10 CURSOR 8, 10
```

……カーソルを X 座標 8 、Y 座標10の位置に移動させます。PRINT 文や INPUT 文を実行すると、この位置から実行されます。



参　考

TAB

SPC

REM（remark・リマーク）

プログラムの中に注釈を書き込む

書　式　REM（注釈）

説　明

・REM 文は非実行文で、プログラムリストを見やすくするための注釈文などの書き込みができます。

・プログラムの実行は、REM 文をとばして、次の実行文へ移されます。したがって、REM 文の後にマルチステートメントを置いても実行されません。

例

```
10  REM  サンプル
20  INPUT  "ハンケイ＝";R
30  REM  エンシュウ  ヲ  モトメル
40  L＝R＊2＊π
50  REM  メンセキ  ヲ  モトメル
60  S＝R＊R＊π
70  PRINT  "エンシュウ＝";L
80  PRINT  "メンセキ＝";S
90  END
```

……円の半径から、円周と面積を求めるプログラムです。このようにプログラムを見やすくすることができます。

LET（let・レット）

右辺の値を左辺の変数に代入する

書式

LET 〈変数〉＝〈式〉

説明

・〈式〉の値を〈変数〉に代入します。

・〈式〉の値と、それを代入する〈変数〉の型は、同じデータ型（数値・文字列）でなければなりません。

・LETは、省略することができます。省略した場合、

```
A=A+3
```

というような文ができますが、これは、関係式ではなく、変数Aの値に3を加算して、その結果を新たに変数Aに代入させるという意味になります。

例

［正しい例］

```
10  LET  N=32
```

……変数Nに32を代入します。

```
20  LET  A$="SUNDAY"
```

……文字変数A$に" SUNDAY "という文字列を代入します。

```
30  LET  A$=A$+CHR$(255)
40  R.(D)=SIN(TH*π/180+C)+SIN
    (2*TH*π/180+E)
```

```
A=1
```

……直接（ダイレクト）モードでの代入例

［誤まった例］

```
50  LET  D$=A+B
```

……左辺が文字変数なのに右辺が数式となっています。

```
60  LET  A$(2,1)=A*2
```

……左辺が文字配列変数なのに、右辺が数式となっています。

STOP （stop・ストップ）

プログラムの実行を中断する

書　式　STOP

省略形　S.

説　明

・プログラムの実行を一時停止し、その停止された行番号を表示してから、命令待ちの状態になります。

・STOP 文はプログラムの実行を一時的に中断するために使用し、プログラム中どこにあっても何回使ってもかまいません。プログラムは一時的に中断されただけですから、変数に記憶されている値を、直接モードの PRINT 命令で調べたりすることができます。CONT CR とキー入力すると、コンピュータは実行を再開します。

・STOP 文は END 文と異なり、ファイルの CLOSE（クローズ）を実行しません。

例

```
10  READ  A，B
20  X＝A＊B
30  STOP
40  Y＝A／B
50  PRINT  X，Y
60  END
70  DATA  15，5
RUN
Break  in  30
Ready.
```

←（ピリオド）は CONT 可能であることを表します。

……ここで直接モードで、たとえば PRINT A, B, X CR とし、A, B, X の値を調べることもでき、また CONT CR とキー入力することで、コンピュータは実行を再開することもできます。

参　考　CONT

END（end・エンド）

プログラムの実行を終わる

書　式　END

省略形　E.

説　明

・END 文を実行するとシステムは

Ready

を表示して、BASIC が命令待ちの状態になったことを示します。

・プログラム実行中で、オープンされたままのファイルがあれば、それを CLOSE（クローズ）します。

例

```
10  INPUT  A,B
20  IF  A>=1000  THEN  END
30  PRINT (A+B)/2
40  GOTO  10
```

……入力した2つの数値の平均を表示しますが、変数Aに1000以上の数を入力すると、そこでプログラムが終了します。

FOR～NEXT （for～next・フォア～ネクスト）

FOR と NEXT との間の文をくり返し実行する

書　式

FOR 〈制御変数〉 ＝ 〈初期値〉 TO 〈終値〉 [STEP 〈増分〉] ～ NEXT 〈制御変数〉

省略形

F. ～N.

説　明

・〈制御変数〉は、FOR～NEXT 文によりくり返される文をコントロールするための変数を与えます。

・NEXT とともに用い、変数のとる値の〈初期値〉から〈終値〉までを〈増分〉ずつ変えて、FOR～NEXT の間をくり返します。変数の値が終値をこえると NEXT の次の文に実行が移ります。

・〈増分〉は、正の値のほかに負の値を使うこともできます。このとき終値が初期値より小さくなるまで実行されます。

・STEP 〈増分〉を省略すると、増分は 1 で実行されます。

・FOR～NEXT 文を重ねてくり返しを行う場合、次の点に注意しなければなりません。FOR～NEXT 文の重ね合わせは、必ず入れなければなりません。FOR～NEXT 文の重ね合せは、必ず入れ子型（ネスティング57ページ参照）となるようにし、内側のくり返し文は、外側のくり返し文に完全に含まなければなりません。またそれぞれのくり返し文は別々の制御変数を使わなければなりません。

重ね合わせたくり返し文が同じ場所で終わる場合は、NEXT は 1 つにまとめてしまうことができます。そのとき制御変数名はコンマで区切って、内側にあるくり返し文の制御変数名から順番に書いていきます。

くり返し文の重ね合わせには制限はありませんが、プログラム実行時における残りエリアの大きさによって制限を受けます。

NEXT 文が最も近い FOR 文に対応する場合は、その制御変数名を省略することができます。もし対応する FOR 文のない NEXT 文がある場合は、エラーが発生します。

・FOR～NEXT 文の中では CLR 文と LIMIT 文は使えません。

例

```
10  A=0
20  FOR  N=0  TO  10  STEP  2
30  A=A+2
40  PRINT  "N=";N,
50  PRINT  "A=";A
60  NEXT  N
```

……このプログラムでは行番号20において、変数Nに0が代入されます。行番号20～50まで実行し、AとNの値を表示します。行番号60でNの値にSTEPの値2をプラスし、最終値10より大きいかどうかの判定を行います。大きくなれば、行番号20以降をくり返します。

Nの値が10をこえたときに、くり返しのループを脱出し、次の命令に移って行きます。

上のプログラムではループを6回行います。

STEPを省略した場合は、Nはループを1回まわるごとにNに1がプラスされます。上のプログラムでSTEPを省略すると増分は1となり、ループは11回行われます。

```
FOR  N=0  TO  10  STEP  2
       ↑      ↑       ↑
    初期値   終値    増分
```

LABEL（label・ラベル）

ラベルを設定する

書式

LABEL　"〈ラベル名〉"

〈ラベル名〉：249文字までの文字列

説明

・GOTO文や、GOSUB文などのジャンプ先を表すラベルを設定することができます。これによってプログラムを見やすく、またわかりやすくすることができます。

例

```
10  PRINT  "SAMPLE"
20  GOSUB  "ABC"
30  PRINT  "END"
40  END
100  LABEL  "ABC"
110  PRINT  "LABEL  SAMPLE"
120  RETURN
```

……行番号20のGOSUB文のジャンプ先を、行番号100で"ABC"というラベルを設定しています。したがって、GOSUB文により行番号100以降のサブルーチンが呼び出されます。

参考

GOTO

GOSUB

GOTO （go to・ゴーツー）

プログラムの実行を、指定された行番号へ無条件に移す

書式

GOTO { 〈行番号〉 / 〈ラベル〉 }

説明

・GOTO 命令を実行すると、指定した〈行番号〉,〈ラベル〉からプログラムが実行されます。

・〈行番号〉の場合は、指定する行の番号を直接指定しなくてはなりません。〈ラベル〉の場合はラベル名を指定してください。

例

```
10  INPUT  A
20  B=A-1
30  LABEL  "START"
40  A=A*B
50  IF  B=1  GOTO  80
60  B=B-1
70  GOTO  "START"
80  PRINT  A
90  END
```

……入力した数値の階乗を求めて表示するプログラムです。

参考

LABEL

GOSUB

ON～GOTO（on～goto・オン～ゴーツー）

算術式の値にしたがって、いくつかの指定された行番号の行へジャンプする

書　式

ON 〈算術式〉 GOTO { 〈行番号〉 [，〈行番号〉] … / 〈ラベル〉 [，〈ラベル〉] … }

説　明

・ON～GOTO 文によるジャンプ先は、GOTO に続いて、いくつかの〈行番号〉を，（コンマ）で区切って書きます。ジャンプ先の数は1行（行番号も含み255文字まで）に記述できるならいくつでもかまいません。また、〈行番号〉のかわりに〈ラベル〉で指定することもできます。

・第1番目から第n番目までのジャンプ先に対して、整数1，2…nが対応し、〈算術式〉が、整数値であれば、指定した数値のジャンプ先へプログラムの実行が移ります。

・〈算術式〉が整数でない場合、小数部分を切り捨てた整数値によってジャンプ先が決められます。したがって数値の内部表現に対する注意（IF～THEN 文中の注意を参照）が必要です。たとえば、Aの値の内部表現が、1.99という値であったとしたら、ON 文で扱われる値は、この整数部の値、すなわち1となります。

・〈算術式〉が1以下、またはnよりも大きい数の場合は、ON～GOTO 文の次の文へ実行が移ります。

例

```
10  INPUT "A=";A
20  ON A GOTO 40,"TEST",90
30  GOTO 10
40  PRINT "グウ"
50  GOTO 10
60  LABEL "TEST"
70  PRINT "チヨキ"
80  GOTO 10
90  PRINT "パー"
100  GOTO 10
```

参　考

GOTO

ON GOSUB

GOSUB～RETURN (gosubroutine～return・ゴーサブルーチン～リターン)

プログラムの実行をサブルーチンへ移し、RETURN 文で戻る

書　式

GOSUB { 〈行番号〉 / 〈ラベル〉 }
⋮
RETURN

省略形

GOS. ～RET.

説　明

・プログラムの実行を GOSUB 文の行番号またはラベルによって指定されたサブルーチンへ移し、RETURN 文によって GOSUB 文の次の文へ戻ります。

・サブルーチンとは、プログラムの中で同じ処理方法の仕事が何回でもくり返し出てくるようなとき、その仕事だけを小プログラムとしてまとめたものです。

・サブルーチンの多重化（ネスティング）には制限はありませんが、そのプログラムの残りのメモリで制限されます。

・呼び出されるサブルーチンの最後には必ず RETURN 文がなければなりません。

例

```
10  INPUT  A,B
20  GOSUB  100
30  B=C
40  GOSUB  100
50  PRINT  C
60  END
100  C=A↑2+B
110  RETURN
```

……メインプログラム

……サブルーチンプログラム

ON～GOSUB (on go subroutine・オン～ゴーサブルーチン)

算術式の値にしたがって、いくつかの指定された行番号のサブルーチンを呼出す

書　式

ON 〈算術式〉 GOSUB { 〈行番号〉 [, 〈行番号〉] ‥ / 〈ラベル〉 [, 〈ラベル〉] ‥ }

省略形

ON～GOS.

説　明

・ON～GOSUB 文の基本的動作は、ON～GOTO 文と同様ですが、ジャンプ先は、すべてサブルーチンとなります。

・サブルーチンが終ると、ON～GOSUB 文の次の文に戻ります。したがって、どのサブルーチンへジャンプしても、RETURN 文で戻る所は同じになります。

例

```
10  INPUT"A";A
20  ON A GOSUB 100,"TEST",150
30  GOTO 10
100  PRINT"グウ"
110  RETURN
120  LABEL"TEST"
130  PRINT"チヨキ"
140  RETURN
150  PRINT"パー"
160  RETURN
```

……行番号10の INPUT 文で1を入力した場合は、行番号100へジャンプし、2を入力した場合は、ラベル"TEST"へジャンプして、3を入力した場合は、行番号150へジャンプします。それぞれの処理（この場合はグウ、チヨキ、パーのいずれかを表示する）が終ると、RETURN 文によりジャンプした次のステートメントを実行します。（この場合は行番号30の GOTO 文）

IF～THEN～：ELSE (if～then～else・イフ～ゼン～エルス)

条件の判断を行う

書　式

IF { 〈関係式〉 / 〈論理式〉 } [THEN] { 〈文〉 / 〈行番号〉 / 〈ラベル〉 } [：ELSE { 〈文〉 / 〈行番号〉 / 〈ラベル〉 }]

説　明

・〈関係式〉または〈論理式〉の条件が成立する（式の値が真）ならば、THEN 以後：ELSE の前まで実行されます（：ELSE 省略時はその行の終りまで）。
条件が成立しない（式の値が偽）ならば ELSE 以後が実行されます。（：ELSE 省略時は次の行を実行）

・〈行番号〉、〈ラベル〉を記述すると、その行に実行が移ります。

・文字データを比較する場合、文字列の大小関係はアスキーコードの大小関係にしたがいます。たとえば、A＜B＜C…Y＜Z＜ア＜イ＜…＜ワ＜ヲ＜ンのようになります。
また、比較する文字列は2文字以上でもかまいませんが、その場合は先頭から順に比較しています。

例

```
10 IF C<1 THEN C=3 :ELSE C=C-1
```
……もし数値変数Cが1よりも小さければ、Cに3を代入します。そうでなければCから1を引いた値(C－1)をCに代入します。

```
10 IF C=D THEN 150 :ELSE END
```
……もし変数Cの値が変数Dの値と等しければ行番号150以下を実行します。そうでなければプログラムの実行を終了します。

```
10 IF A$="ABC" THEN A$=A$+"DEF"
```
……もし文字変数A$が"ABC"ならば、文字変数A$にA$＋"DEF"、したがって"ABCDEF"を代入します。

IF～GOTO（if～goto・イフ～ゴーツー）

条件を判断し、その結果にしたがってジャンプする

書式

IF { 〈関係式〉 / 〈論理式〉 } GOTO { 〈行番号〉 / 〈ラベル〉 }

省略形

IF～G.

説明

・IF～THEN と同様に条件付きジャンプを行う文ですが、条件が成立する場合 GOTO 〈行番号〉 または GOTO 〈ラベル〉 の行へ実行を移します。

・条件が成立しない場合は、次の行へプログラムの実行が移ります。IF～GOTO 〈行番号〉 または 〈ラベル〉 と IF～THEN 〈行番号〉 または 〈ラベル〉 は同じ働きです。

・IF～GOTO 文の後には、マルチステートメント（45ページ）を置いてもプログラム実行にとって無意味となるので注意が必要です。

例

```
10 IF A$="ABC" GOTO 50
```

……文字変数 A$ の内容が " ABC " ならば行番号50へ実行を移します。

参考

GOTO

IF THEN

IF GOSUB

IF～GOSUB (if～go subroutine・イフ～ゴーサブルーチン)

条件を判断し、その結果にしたがってサブルーチンを呼び出す

書　式

IF { 〈関係式〉 / 〈論理式〉 } GOSUB { 〈行番号〉 / 〈ラベル〉 }

省略形

IF～GOS.

説　明

・IF～THEN 文と同様の条件付きジャンプを行う文ですが、条件が成立する場合、GOSUB 〈行番号〉 または 〈ラベル〉 で示されるサブルーチンを呼び出します。

・IF～GOSUB の条件が成立してサブルーチンジャンプを実行し、RETURN 文により戻ってきた場合、マルチステートメントであれば、そのステートメントを実行します。ただし条件が成立しなかった場合は、マルチステートメントは無視され次の行番号のステートメントから実行します。

・RETURN 文によるサブルーチンからの復帰は、IF～GOSUB 文の次の実行文へ移ります。

例

```
10 IF A=3 GOSUB 100
```

……数値変数 A が 3 ならば行番号100のサブルーチンを呼び出します。

参　考

GOSUB～RETURN

IF THEN

IF GOTO

PRINT （print・プリント）

画面上にデータを表示する

書　式

PRINT [[〈パレット1〉,〈パレット2〉]] "〈データ〉" [{ ; / , } 〈データ〉]…

{ パレット1：文字の色（パレットコード）
{ パレット2：背景の色（パレットコード）

省略形

?

説　明

・画面上にデータを表示します。
・〈データ〉を省略すると、改行のみを実行します。
・〈パレット1〉では文字の色のパレットコード（66ページ）を、〈パレット2〉では背景の色のパレットコードを設定します。省略すると、座標指定を行わないCCOLOR文で指定されたパレットコードとなります。
・〈データ〉の区切り記号として、セミコロン（；）を使用すると前の〈データ〉に続けて次の〈データ〉を表示します。
・〈データ〉の区切り記号として、コンマ（，）を使用すると、10文字分のタビュレーションを行ってから、次の〈データ〉を表示します。
・数値データの表示形式には実数形式と指数形式があります。
1×10^{-8} から 1×10^{8} 未満のデータについては実数形式で、それ以外のデータについては指数形式で表示します。

例

```
10 PRINT [6,5] "ABC";123
```

……文字データ" ABC "と数値データ１２３を続けて表示します。文字の色はパレットコード6、背景の色はパレットコード5となります。

```
20  PRINT [,4] "ABC",123
```

……文字データ" ABC "と数値データ１２３を10文字のタビュレーションをつけて表示します。背景の色はパレットコード４になります。文字の色は省略されていますので、座標指定を行っていないCCOLOR文で指定したパレットコードになります。

```
30  PRINT [7,] "ABC"
```

……文字の色はパレットコード７になり、背景の色は省略されていますので座標指定を行っていないCCOLOR文で指定したパレットコードになります。最後にセミコロン（；）を追加すると改行は行われません。

```
40  PRINT  "ABCDE"
50  END
```

● PRINT命令の特殊な使用方法は次のとおりです。

PRINT"Ⓒ"　画面全部をクリアしてカーソルを左上すみに持っていく。

PRINT"Ⓗ"　画面全部はそのままにしてカーソルを左上すみに持っていく。

PRINT"➔"　カーソルを一字分右へ持っていく。

PRINT"←"　カーソルを一字分左へ持っていく。

PRINT"↑"　カーソルを一行分上へ持っていく。

PRINT"↓"　カーソルを一行分下へ持っていく。

入力の方法は下記の通りです。

それぞれのキャラクタの入力方法は GRAPH キーを押すと、カーソル形状が■に変化します。その状態で編集用キーの CLR、HOME →、←、↑、↓、を押します。もとにもどすには 英数 キーを押してください。

参考　CCOLOR

PAL

PRINT USING（print using・プリントユージング）

書式指定されたデータを画面上に出力する

書式

PRINT [[〈パレット1〉，〈パレット2〉]] USING

" 〈書式指定〉 " ；〈データ〉 [{ ； / ， } 〈データ〉]

省略形

? USI.

説明

・書式指定されたデータを画面上に出力します。書式指定以外の項目については PRINT 文（121ページ）と同じです。
書式指定の意味は次の通りです。

例

(1)数値形

(a) #

表示したい最大のケタを指定し、出力は、ケタの範囲内で右づめの表示となります。

```
PRINT USING"####";A
␣123……A の値が123の場合
```

(b) .

固定小数点の表示で、小数点位置をそろえるという指定です。小数点以下のケタ数指定も行えます。

```
PRINT USING"###.##";A
␣12.34……A の値が12.34の場合
```

(c) ,

数値を3ケタごとに区切ってそこに，（コンマ）を入れて表示します。数字は右づめ表示となります。

```
PRINT USING"#,###,###";A
6,345,123……A の値が6345123の場合
PRINT USING"###,###.##";A
123,456.00……A の値が123456の場合
```

(d) ＋と－

＋と－の符合の位置を数の後につけたり、正の数のとき、␣（スペース）ではなく＋をつけたりする指定ができます。

PRINT USING"####＋"；A　または

PRINT USING"####－"；A　と指定すると、符合が数の後につきます。たとえばAが－13のとき、␣␣13－と表示され、Aが＋34の場合は␣␣34＋と表示されます。

ただし、－（マイナス）の指定で、正の数のとき、符号は␣（スペース）となります。

PRINT USING"＋####"；B　と指定すると、符号が数の前につきます。たとえばBが25のとき␣␣＋25と表示されます。

なお、

PRINT USING"－####"；C　という指定はないので、マイナス符合は単なる文字として表示されます。

(e) ＊＊

ケタ指定をし、その範囲内で、空白がある場合、全て＊で埋めるという指定をします。＊指定は必ず＊＊としてください。

PRINT USING"＊＊####"；A

＊＊＊＊12……Aが12の場合

(f) ￥￥

数字の前に￥（円記号）を付けます。

￥指定は、必ず￥￥としてください。

PRINT USING"￥￥####"；A

␣␣￥123……Aが123の場合

(g) ↑↑↑↑

指定表項の数値を制御します。

```
PRINT USING"##.##↑↑↑↑";A
```

␣5.11E+14…Aが51.123×10[13]の場合、なお、書式指定の最初の#は正負の判断に使用されます。

(h) **一行連続**

1つのUSINGで、複数の変数を記述するとその変数全てに対して、USINGを指定したとみなされます。

```
PRINT USING"##.###";{A;B;C}
                    {A,B,C}
```

どちらでも実行結果は同じです。

```
␣5.300␣6.900␣7.123
```

Aが5.3、Bが6.9、Cが7.123の場合

(2)文字型

(a) **!**

!の指定は、文字型変数にある文字列の最初の1文字のみを出力するものです。

```
PRINT USING"!";A$
```

C……A$が"CDE"の場合

(b) &␣␣␣␣&

指定文字列の中から2+n個出力するという命令で、&と&の間の空白(␣)の個数がn個です。文字は左づめの表示となり、2+n個以下であれば、残りを空白で埋めます。

```
PRINT USING"&␣␣␣␣&";A$
```

ABCDEF……A$が"ABCDEFGH"の場合指定した数より多い文字(この場合"GH")は切捨てられます。

```
PRINT USING"&␣␣␣&";A$
```

XY␣␣␣……A$が"XY"の場合

(3)文字定数出力機能

USINGの指定子とみなされないキャラクタがある場合は、それをそのまま出力します。

```
PRINT USING"DATA####";A
DATA␣123……Aが123の場合
```

(4) USINGの分離

PRINT文中、通常のPRINTとUSINGを組み合わせて使用することができます。

```
PRINT A;B;USING"####";C
  通常機能        USING機能
−12␣14␣␣12
……Aが−12、Bが14、Cが12の場合
```

INPUT（input・インプット）

プログラムの実行中にキーボードからデータを読み込む

書式　INPUT［〈メッセージ〉；］〈変数〉［，〈変数〉］……

説明

・プログラム実行中に、キーボードから入力されたデータを読み込み、〈変数〉に代入します。

・プログラム中でINPUT文を実行すると、〈メッセージ〉がある場合にはそのメッセージを、〈メッセージ〉がない場合には"？"マークを表示して、カーソルが点滅し、入力待ちの状態であることを示します。この状態で、必要なデータをキーボードから入力し、CRキーを押すとそのデータが〈変数〉に代入され処理が行われます。入力するデータと〈変数〉は同じ型でなくてはなりません。

・文字定数は"　"（ダブルクォーテーションマーク）なしで入力できますが、前後のスペースは無視されます。前後にスペースを与える場合やコンマを与える場合は全体をダブルクォーテーションマークで囲みます。

例

```
10  INPUT  A,B$
```

……入力待ちの状態となり、"？"マークを表示します。入力が行われると、最初のデータを数値変数Aに、次のデータを文字変数B$に代入します。

```
20  INPUT  "A=";A
```

……メッセージ"A="を表示して入力待ちの状態になります。

GET（get・ゲット）

プログラム実行時に、押されているキーを調べそのデータを変数に代入する

書式　GET 〈変数〉

説明　GET 文が実行された時点で、キーが押されていたら、そのデータを変数に代入します。キーが押されていない場合、数値変数には 0 が代入され、文字変数には Null（" "）が代入されます。

・GET 文によりキー入力できるデータは、数値データであれば、1～9のうちいずれかの整数であり、文字データであれば、1文字が入力されます。

・数値変数に対しては、数字キー以外のキーを押しても無視されます。

例

```
10 GET A$:IF A$="" THEN 10
20 PRINT A$
30 END
```

……キーボードから入力された文字を表示します。ただし、通常キー以外は表示しません。

DIM （dimension・ディメンジョン）

配列変数を宣言し、必要なメモリ領域を確保する

書　式

DIM 〈変数名〉(〈添字〉) [, 〈変数名〉(〈添字〉)] …

DIM 〈変数名〉(〈添字〉, 〈添字〉) [, 〈変数名〉(〈添字〉), 〈添字〉)] …

説　明

・配列変数を使用する場合は、DIM 文によって、あらかじめ配列の名前と大きさを宣言し、必要なメモリを確保する必要があります。

・〈配列名〉は、英大文字＋英数字の２文字が有効です。

・１次元から４次元までの配列が使用できます。

・〈添字〉の大きさには制限はありませんが、メモリの使用状況によって制限されます。

・DIM 文を実行した時点では、宣言した配列の、すべての要素の値を０または Null（" "）とします。

・CLR（101ページ）が実行されると、配列宣言はすべて無効になります。

・すでに配列宣言を行っている配列に対して、より大きい配列を宣言すると、ERROR が発生します。（添字が１つでも前の添字より大きいとき）

例

```
10 DIM A(99)
```

……数値配列 A を、A（0）～A（99）までの100個の大きさの配列として宣言します。

```
20 DIM B$(9, 9)
```

……文字配列 B$ を、B（0, 0）～B（9, 9）までの100個の大きさの配列として宣言します。

READ～DATA（read～data・リード～データ）

DATA 文に記述したデータ並びを読み、変数に代入する

書　式

READ 〈変数〉［，〈変数〉］…
　　⋮
DATA 〈定数〉［，〈定数〉］…

省略形

REA. ～DA.

説　明

・DATA 文は、１つの文あるいは複数の文で、データ並び（データテーブル）を構成し、READ 文は、そのデータを順番に１つ１つ読み出して、入力並びの１つ１つの変数に代入します。
・データ並びのデータと、対応する入力並びの変数のデータの型は、一致しなければなりません。
・READ 文で、データ並びの途中のデータまで読んだあと、次の READ 文を実行すると、その次のデータからひき続いて読み出されます。
・RESTORE 文を用いると、次の実行する READ 文で読み出す DATA 文の位置を決めることができます。
・DATA 文はプログラム中のどこに置いてもかまいません。
・DATA 文中の文字データは" "（ダブルクォーテーション）を省略することもできます。ただし、その文字定数で空白を使っている場合には省略できません。

例

```
10  READ  A$,B$,C
20  PRINT  A$;B$;C
30  DATA  "SHARP  ","MZ－",1500
40  END
```

……DATA 文からデータを読み込んで、それを表示するプログラムです。

参　考

RESTORE

RESTORE（restore・リストア）

READ文と対応するDATA文の位置を指定する

書式

RESTORE [{〈行番号〉 / 〈ラベル〉}]

説明

・〈行番号〉を書かないかあるいは、0を指定すると、データ並びの先頭、すなわち、DATA文のうちもっとも小さい行番号を持つDATA文のデータから読み出します。

・〈行番号〉,〈ラベル〉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持つDATA文のデータから読み出します。

例

```
10  DATA  "パーソナル  コンピュータ  "
20  DATA  "MZ－1500"
30  READ  A$, B$
40  PRINT  A$, B$
50  RESTORE  20
60  READ  C$
70  PRINT  C$
80  RESTORE
90  READ  D$
100  PRINT  D$
110  END
RUN
パーソナル  コンピュータ  MZ－1500
MZ－1500
パーソナル コンピュータ
```

参考

READ～DATA

DEF FN (define function・デファインファンクション)

関数を定義する

書式　DEF FN 〈関数名〉(〈変数〉) = 〈算術式〉

説明

・数式を組み合わせたものを、新しい関数として定義します。

・定義する関数名は、FNに続けてアルファベットの大文字1文字で指定します。したがって、同時に定義できる定義関数は最大26となります。

・CLR文、NEW文が実行されると、定義関数はすべて未定義となります。

例

```
DEF　FNA (X) =2*X↑2+3*X+1
```

……2 X^2+3 X+1をFNA (X) と定義します。

```
DEF　FNE (V) =1/2*M*V↑2
```

……$1/2MV^2$をFNE (V) と定義します。

```
10　DEF　FNB (X) =TAN (X-π/6)
```

(誤った例)

```
10　DEF　FNK (X) =SIN (X/3+π/4),
　　FNL (X) =EXP (-X↑2/K)
```

……DEF文は2つの関数定義式を置くことはできません。

質量5.5で初速度が3.5、3.5×2、3.5×3のときの運動エネルギーを求めます。

```
10　DEF　FNE (V) =1/2*M*V↑2
20　M=5.5:V=3.5
30　PRINT　FNE (V),FNE (V*2),FNE (V*3)
40　END
```

結果は

```
 33.6875   134.75   303.1875
```

TRON（trace on・トレース オン）

プログラムの実行を追跡する

書　式　TRON［／P］

省略形　TRO.

説　明
- プログラムの実行を追跡します。
- TRONが実行されると、プログラムを実行させたときに、現在どの行を実行しているかをしめす行番号が［　］（かぎカッコ）で囲まれて表示されます。
- ／Pを指定するとプリンタに出力されます。

例

```
10  DEF  FNA (X, Y) =X*Y
20  I=I+1 : READ  A1, A2, A3, A4
30  W=FNA (A1, A2) : GOSUB  100
40  W=FNA (A2, A3) : GOSUB  100
50  W=FNA (A3, A4) : GOSUB  100
60  DATA  4, 5, 6, 7
70  END
100  IF  W>20  THEN  PRINT"ABCD"
110  RETURN
```

……このプログラムを実行する前に、TRON CR と入力してからRUNしてみると

```
RUN
[10][20][30][100][110][40]
[100] ABCD
[110][50][100] ABCD
[110][60][70]
```

……［　］でくくった行番号の表示が出てくるので、プログラムがどのように実行されているかよくわかります。この動作を中止させるにはTROFF命令を入力してください。

参　考　TROFF

TROFF （trace off・トレース オフ）

TRON 文の機能を解除する

書　式　TROFF

省略形　TROF.

説　明　・TRON の機能を解除する。

参　考　TRON

●プログラムには「虫」が付きもの

プログラムは、人間が作り、操作するものです。ですから、何らかのミスをおかしているかもしれません。このようなミスを「虫（バグ）」と呼んでいます。

プログラムが複雑になればなるほど、バグは増えます。また一見したところ小さなバグでも、大変大きな影響を及ぼすことがよくあります。バグは発見しだい退治しなければなりません。コンピュータを正しく操作しても、プログラムの実行結果が間違ったものとなるからです。そして、このバグ取り除き作業を「デバッグ」と呼んでいます。

プログラム中のバグを発見するために、いろいろテストを行うわけですが、バグのまったくないプログラムを作るのは、なかなかむずかしいようです。

TRON などの命令は、バグの発見のために用意されています。

DEF KEY (definable key・デファイナブルキー)

デファイナブルキーを定義する

書式

DEF KEY (〈キーの番号〉) = " 〈定義文字〉 "

説明

・5個のデファイナブルキーに対して、その機能を定義します。

・〈キーの番号〉に1～10のいずれかを指定し、〈定義文字〉を、示すストリングや命令などをそのまま書き込みます。〈キーの番号〉6～10は SHIFT キーを押しながらデファイナブルキーを押す場合の設定になります。

・同じデファイナブルキーについて、新たにDEF KEYが実行されると前の定義が無効となります。最大登録文字数は15文字までです。ただし（"）で囲まれた文字数でスペースも含みます。

例

```
10 DEF KEY (1) = "SHARP"
```

……F1 に " SHARP " 定義します。

```
10 DEF KEY (2) = "RUN" +CHR$(13)
```

……F2 に RUN CR を定義します。

注) CHR$ (13) は CR コードで、CR を押すのと同じ意味を持ちます。

KEY LIST （key list・キーリスト）

デファイナブルキーの定義状態を表示する

書　式　KEY LIST

省略形　K．L.

説　明　・デファイナブルキーの定義状態（定義動作）を見るため、画面上へ文字定義リストの表示を行います。

例

```
KEY LIST CR
DEF KEY( 1)="RUN␣␣␣"+CHR$(13)
DEF KEY( 2)="LIST"
DEF KEY( 3)="AUTO"
DEF KEY( 4)="RENUM"
DEF KEY( 5)="DIR␣"
DEF KEY( 6)="CHR$("
DEF KEY( 7)="DEF KEY("
DEF KEY( 8)="CONT"
DEF KEY( 9)="SAVE␣␣"
DEF KEY(10)="LOAD␣␣"
Ready
```

……デファイナブルキーの初期設定時の内容です。

INIT （initialize・イニシャライズ）

各装置の初期化およびモードの設定を行う

書　式

（1） INIT " QD："

（2） INIT " RAM：〈バイト数〉"

（3） INIT " LPT：［M{0/1/2}］［, Sn］［, CR コード］"

（4） INIT " RS{1/2/3/4}：〈モニタリングコード〉

〈初期設定コード〉［,〈終了コード〉］"

（5） INIT " CRT：{G/I}"

説　明

書式（1）

・クイックディスクの初期化を行います。INIT を実行すると、画面に「OK？［Y／N］」と表示されます。問いかけにしたがって初期化を行ってください。

例

ＩＮＩＴ　"ＱＤ："

……クイックディスクの初期化を行います。"Y" を入力するとクイックディスクのイニシャライズが行われ、"N" を入力するとBASICコマンドレベル待ちになります。

ＩＮＩＴ　"ＱＤ：Ｙ"

……無条件にクイックディスクを初期化します。

説　明

書式（2）

・オプションのRAMファイル（MZ-1R18）を初期化し、そのうち〈バイト数〉（バイト数＝$0010～$FFFF）で指定したメモリ容量をRAMファイル用に割り当て、残りの部分をプリンタ・バッファに割り当てます。

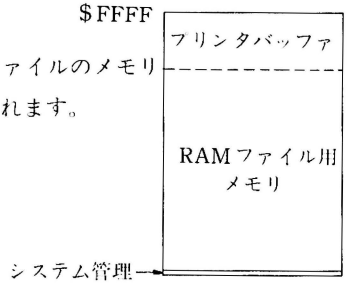
・〈バイト数〉を省略した場合、RAMファイルの設定を変えないで初期化します。

・RAMファイル用に割り当てたメモリ容量または、プリンタバッファ用に割り当てたメモリ容量が小さすぎる場合、どちらか一方が使用できなくなる場合もありますので注意してください。

例

INIT "RAM:"

……を実行すると、RAMファイルのメモリ構成は次のように変更されます。



説 明

書式（3）

・RAMファイルの一部のメモリをプリンタ・バッファとして使用した場合の使用情況の決定やプリンタの指定を行います。

・［M］はプリンタバッファのモード設定です。

- M 0：ダイレクトモード（バッファを初期化します。）
- M 1：スプールモード（バッファを初期化します。）
- M 2：ダイレクトモード（モード設定の前がスプールモードであれば、バッファの内容をすべて打ち出した後、ダイレクトモードとなります。）

・CTRL ＋ N で、スプールモード時にプリンタへの出力が停止できます。

・RAMファイルの未使用時には、M 1、M 2は使えません。

・スプールモードのとき、MZ-80KP5を使いプリンタにイメージプリントコード（0BH＋0BH）を出力するとモードエラーになります。この場合、INIT " LPT：M 2 " を実行してふたたび希望する命令を行なってください。

・［S］はプリンタの指定を行います。

S 0 ： MZ-1P09
S 1 ： MZ-80P4K
S 2 ： MZ-80KP5
S 3 ： MZ-1P08
S 4 ： JIS コードに変換します
S 5 ：コードスルー（改行コード 0DH）
S 6 ：コードスルー（改行コード 0AH）

・INIT でS0 ～ S3を指定した場合で、PRINT/P命令を使うと以下に示すコードが変換されます。

CHR$ （$11）または ↓ → $09に変換されます。
CHR$ （$12）または ↑ → $0Bに変換されます。
CHR$ （$15）または H → $0F に変換されます。
CHR$ （$16）または C → $0C と $0A に変換されます。

・「S4」指定のJIS コードは、05Hの後、06Hまたは改行までの間のA ～ Zについては、a ～ z（英小文字）にコード変換され80H ～ BFH のカタカナは、A0H ～ DFH に変換されます。

・CR コードは、0DH 以外に設定する場合にそのコードを指定します。

例

```
INIT "LPT:M1,S3"
```

……RAM ファイルの一部のメモリをプリンタバッファとして使用し、プリンタバッファのモードをスプールモードに設定します。使用するプリンタは MZ-1P08 を指定します。

●スプールモードとは

パーソナルコンピュータがプリンタに印字する場合は、普通データが全て印字されるのを待ってから次の処理を始めます。しかし、プリンタが印字する速度はコンピュータの処理速度に比べてかなり遅いため、この待ち時間はコンピュータにとってかなり長いものとなります。

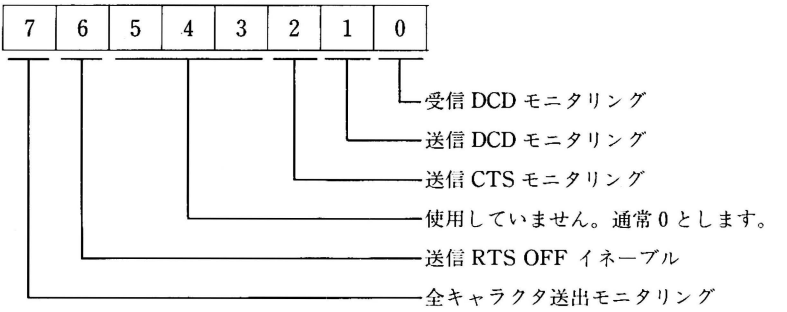
そこで、プリンタに出力するデータを一時記憶しておくメモリ（バッファ）を使い、コンピュータはバッファにデータを送るだけで、あとはバッファから自動的に印字が行われるようになると、待ち時間がなくなり、プリンタが印字している間に、コンピュータは別の処理を行うということが可能になります。

このように、プリンタとコンピュータが並列に処理することを、スプールモードといい、MZ-1500 では RAM ファイル（MZ-1R18）の一部のメモリを使って実施します。

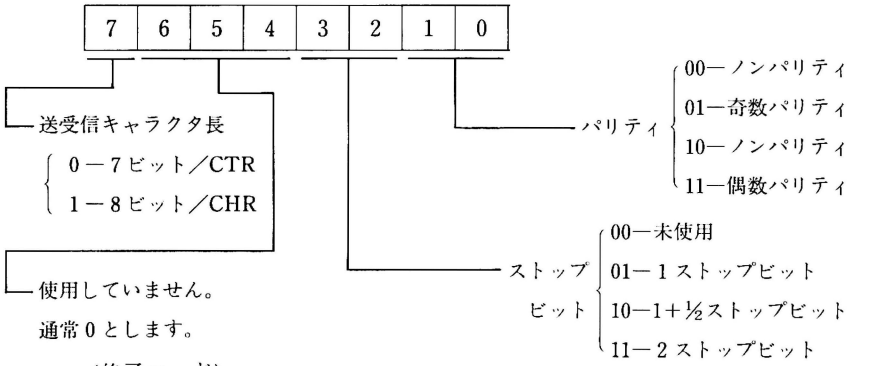
説　明　　**書式（4）**

・RS232C のモードを設定します。（High アクティブ）

〈モニタリングコード〉



〈初期設定コード〉



〈終了コード〉

0 ～255（$00～$FF）までの数値

例　　2 台の MZ-1500 間で情報交換を行う場合次のような結線を施したケーブルを用意してください。

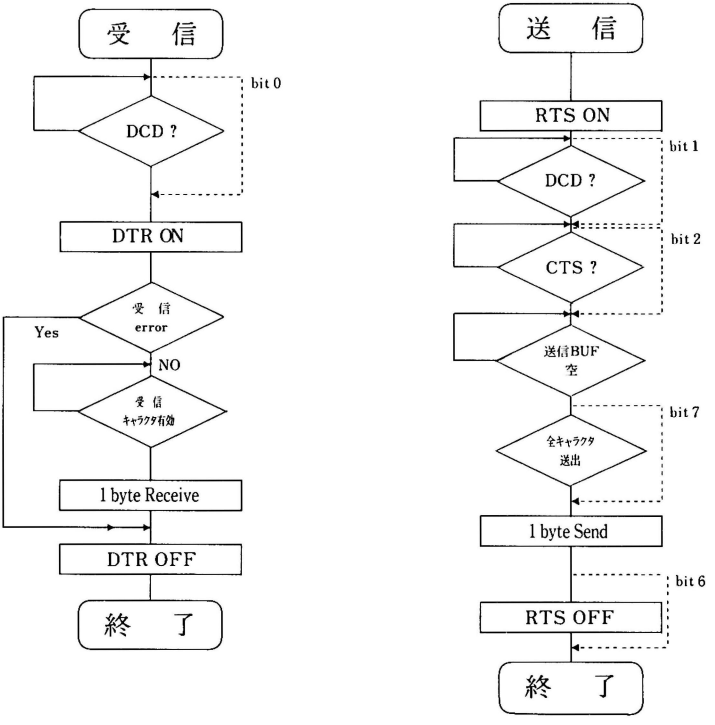
信 号 名	ピン番号
TXD	2
RXD	3
RTS	4
CTS	5
DTR	6
DCD	7
アース	1，8

ピン番号	信 号 名
2	TXD
3	RXD
4	RTS
5	CTS
6	DTR
7	DCD
1，8	アース

・両方ともターミナルモード(RS—232Cに関する取扱説明書参照)で使用します。

・2台のMZ-1500間で情報交換を行うプログラム例でA$の内容0～9までの数字を送受信するプログラムです。

〈フローチャート〉



〔受信側〕

```
10  INIT"RS1:$00,$8C"
20  ROPEN#1,"RS1:DATA"
30  INPUT#1,A$
40  PRINT  A$
50  CLOSE#1
60  END
```

〔送信側〕

```
10  INIT"RS1:$00,$8C"
20  A$="0123456789"
30  WOPEN#2,"RS1:"
40  PRINT#2,A$
50  CLOSE#2
60  END
```

書式（5）

・ディスプレイの画面2つのモードを設定します。

G：グラフィックモード

I：イメージモード

・モードを省略した場合は、イメージモードになります。

例

```
INIT  "CRT:I"
```

……ディスプレイ画面2のモードをイメージモードに設定します。

```
INIT  "CRT:G"
```

……ディスプレイ画面2のモードをグラフィックモードに設定します。

注）イメージモード、グラフィックモードについては、63ページを参照してください。

PRTY （priority・プライオリティ）

画面１と画面２の優先順位を設定する

書　式

PRTY [{1 / 2}]

説　明

・画面１のデータと画面２のデータのどちらかを優先して表示するかを設定します。

・１ならば、画面１のデータを優先して表示します。省略した場合も同じです。

・２ならば、画面２のデータを優先して表示します。

・背景色は、画面１の背景色となります。

例

```
10  PRTY  1
```
……画面１のデータを優先して表示します。

```
20  PRTY  2
```
……画面２のデータを優先して表示します。

参　考

画面１と画面２の違いは63ページを参照してください。

BYE（bye・バイ）

BASICインタプリタから、モニタ（RAM領域）へ移る

書式　BYE

省略形　B.

説明　・BASICインタプリタから、モニタ（RAM領域）へ移ります。モニタコマンドについては、240ページを参照してください。

例　BYE

……BASICインタプリタから、BASICモニタへ移ります。（RAM領域）

BOOT （boot・ブート）

IPL の起動を行う

書　式　　BOOT

説　明　　・IPL の起動を行います。すなわち電源スイッチを ON にしたのと同じ状態になります。

例　　BOOT

……システムを新しく起動します。

WAIT （wait・ウエイト）

プログラムの実行を一定時間停止する

書式　WAIT 〈数値データ〉

説明　・プログラムの実行を〈数値データ〉で指定された時間の間、停止します。単位は、1/1000秒です。

例

```
10 WAIT 100
```

……プログラムの実行を、0.1(100/1000)秒だけ停止します。

2 ファイル命令

LOAD （load・ロード）

プログラムファイルを読み出す

書 式

LOAD "［〈装置名〉：］〈ファイル名〉 "［〈, A〉］

ただし、〈装置名〉は、QD、RAM、CMTのみ。

省略形

LO.

説 明

・外部記憶装置からプログラムを読み出します。

・読み出すべきファイルは、〈装置名〉と〈ファイル名〉の記述によって指定します。

・DEFAULT文（165ページ）で指定された装置（あるいはデフォルト状態にある装置を指定する場合は、〈装置名〉は省略できます。

・オプション" A "を付加するとBSDファイルをアスキー形式で読み出します。読み出し後、変換に時間がかかります。

・ファイル名は、省略してはいけません。ただし、カセットテープ上のファイルに限りファイル名が省略されていても呼び出すことができます。

例

```
LOAD"QD:HELLO"
```

……クイックディスクから" HELLO "という名前のファイルを読み出します。

```
LOAD"CMT:MONDAY",A
```

……カセットテープから、" MONDAY "という名前のBSDファイルをアスキー形式で読み出します。

SAVE （save・セーブ）

プログラムをファイルとして登録する

書　式

SAVE " ［〈装置名〉：］〈ファイル名〉 " ［〈, A〉］

ただし、〈装置名〉は、QD、RAM、CMTのみ。

省略形

SA.

説　明

・プログラムを外部記憶装置などにファイルとして登録します。

・DEFAULT文で指定された装置あるいはデフォルト状態にある装置を指定する場合は、〈装置名〉は省略できます。

・オプション " A " を付加するとアスキー形式でSAVEします。ファイルタイプはBSDファイルとしてSAVEしますので、WOPEN #、PRINT #文で作成したデータファイルと、混同しないように、ファイルネームを付けてください。

・SAVE命令により作成されるファイルのタイプはBTXファイルまたはBSDファイルです。

・〈ファイル名〉は省略できません。

例

```
SAVE"QD:PROG"
```

……クイックディスク上に、プログラムを、" PROG " という名前をつけて登録します。登録されたファイルのタイプはBTXとなります。

```
SAVE"QD:DATA",A
```

……クイックディスク上にプログラムを、" DATA " という名前をつけてアスキー形式で登録します。登録されたファイルのタイプはBSDとなります。

LOAD ALL (load all・ロード オール)

クイックディスクの内容を、すべてRAMファイル上に読み出す

書式

LOAD ALL

説明

・クイックディスクの内容をRAMファイル上に読み出します。ただし、この命令を実行するには、オプションのRAMファイル（MZ-1R18）が必要です。

・この命令を実行するには、あらかじめRAMファイルを初期設定にする必要があります。初期設定のし方についてはINIT命令(137ページ)を参照してください。なお、初期設定を行ってもLOAD ALLできない場合は、RAMファイルの容量が足りないので必要とするバイト数を、INIT命令を使ってRAMファイルに割り当ててください。(最大約63KB)

例

```
LOAD  ALL
```

……クイックディスク上にあるファイルをすべて(片面のみ)RAMファイル上に読み出します。

参考

SAVE ALL

SAVE ALL （save all・セーブオール）

RAMファイルの内容をすべてクイックディスクに登録する

書式

SAVE ALL

説明

・この命令を実行するには、あらかじめクイックディスクを初期化する必要があります。

・RAMファイルの内容ををすべてクイックディスクに登録します。ただし、この命令を実行するには、オプションのRAMファイル（MZ-1R18）が必要です。

・RAMファイル上に登録されているファイルが復数の場合、RAMファイル上ではギャップ（レコードなどの曲と曲との間の無録音部分）はありませんが、クイックディスク上に書き込まれる場合はある程度のギャップが必要なために、RAMファイル上の全てのファイルをクイックディスク上に登録できない場合があります。

例

SAVE ALL

……RAMファイル上の全てのファイルをクイックディスク上に書き込みます。

参考

LOAD ALL

RUN （run・ラン）

プログラムを実行する

書　式

RUN " [〈装置名〉：] 〈ファイル名〉 " [{ 〈, R〉 / 〈, A〉 }]

ただし、〈装置名〉は、QD、RAM、CMTのみ。

省略形

R.

説　明

・〈ファイル名〉をつづけて記述する場合は、BASICを初期化（NEW命令と同じです。）して、クイックディスクなどにファイルとして記憶されているBASICプログラムを読み出し、つづけてその先頭からプログラムを実行します。

・DEFAULT文で指定された装置、あるいはデフォルト状態にある装置を指定する場合は、〈装置名〉は省略できます。

・RUNのみの場合は、テキストエリアにあるプログラムを実行します。

・オプション " A " を付加するとBSDファイルをアスキー形式とみなし読み込み実行します。またオプション " R " を付加するとOBJファイルをIPLからの読み込みと同じメモリ状態にして読み込み、実行します。

・指定するファイルはBTX, BSD, OBJファイルです。

例

```
RUN"QD：PROG"
```

……クイックディスク上の " PROG " という名前のBTXファイルを読み出し、実行します。

```
RUN"QD：DATA",A
```

……クイックディスク上のBSDファイル " DATA " をアスキー形式とみなし読み出します。

参　考

RUN （103ページ）

VERIFY （verify・ベリファイ）

メモリ上にあるプログラムとカセットファイルを比較します

書　式　VERIFY　" 〈ファイル名〉 "

説　明
・プログラムをカセットテープに SAVE したとき、正しく SAVE されているかを確認するため、コンピュータ内部のプログラムと、ファイルの内容を比較します。両者の内容が同一であれば " Ready " が表示されます。違っている場合は、" DATA ERROR " が表示されますので、SAVE しなおしてください。
・この命令はカセットテープに対してのみ有効です。

例

```
10 VERIFY"NAME"
```

……カセットテープ上の"NAME"というファイルとメモリにあるプログラムを比較します。

CHAIN （chain・チェイン）

実行中のプログラムからファイル上のプログラムに実行を移す

書　式

CHAIN " [〈装置名〉:] 〈ファイル名〉 "

ただし、〈装置名〉は QD、 RAM、 CMT のみ。

説　明

・現在実行中のプログラムから、ファイル上の別のプログラムへプログラムの実行を移します。

・CHAIN 文も１つのファイルのオープンとみなされます。

・CHAIN 文は、プログラム中で RUN 命令を実行したのと似た働きがありますが、チェインするさい、もとのプログラムで使用した変数、配列などの内容はそのまま新しいプログラムへ渡されます。

・DEFAULT 文で指定された装置、あるいはデフォルト状態にある装置を指定する場合は〈装置名〉は省略できます。

例

```
10  A=1
20  B=2
30  CHAIN"QD:PROG"
40  END
```

……クイックディスクにセットされているディスク上の " PROG " という名前の付いたファイルのプログラムが実行されます。

この場合、変数 A、 B の値はそのままで、A＝１、B＝２となっています。

MERGE （merge・マージ）

プログラムにファイル上のプログラムを追加する

書　式

MERGE ［" ［〈装置名〉：］〈ファイル名〉 " ］

ただし、〈装置名〉は QD、 RAM、 CMT のみ。

省略形

ME.

説　明

・〈装置名〉で指定したファイル上のプログラムを読み込み、現在メモリ中にあるプログラムの後につけ加えます。

・DEFAULT 文で指定された装置あるいはデフォルト状態にある装置を指定する場合は〈装置名〉は省略できます。

・コンピュータ内部にあるプログラムの行番号とファイルから読み込んだプログラムの行番号が同じ場合は、ファイルからのプログラムが優先し、前のプログラムは消えます。

例

（メモリ上のプログラム）

```
10  B=2
30  PRINT  B
50  END
```

（クイックディスク上のプログラム）

" PROG "

```
10  A=1
20  PRINT  A
40  END
```

……を、MERGE " QD：PROG " でマージすると次のようになります。

```
10  A=1
20  PRINT  A
30  PRINT  B
40  END
50  END
```

……LIST 命令（96ページ）で確認してください。

WOPEN＃（write open・ライト オープン）

ファイルを書き込み用にオープンする

書 式

WOPEN＃〈ロジカル番号〉，＂［〈装置名〉：］〈ファイル名〉＂

説 明

・ファイルに書き込むための準備を行う文であり、ファイルのロジカル番号と名前を指定します。

・DEFAULT 文で指定された装置あるいはデフォルト状態にある装置を指定する場合は〈装置名〉は省略できます。

例

```
10 WOPEN#1,"QD:DATA"
```

……〈ロジカル番号〉1に定義する＂DATA＂というファイルを書き込み用にオープンします。

```
10 WOPEN#1,"RS0:DATA1"
```

……PRINT #命令においての出力を RS-232C に設定します。

```
10 WOPEN#2,"DATA2"
20 FOR I=1 TO 99
30 PRINT #2,1
40 NEXT I
50 CLOSE #2
60 END
```

……1～99までの数をファイルに書き込みます。

参 考

PRINT＃
ROPEN＃
CLOSE＃

ROPEN #（read open・リード オープン）

ファイルを読み出し用に、オープンする

書 式

ROPEN # 〈ロジカル番号〉 , " [〈装置名〉:] 〈ファイル名〉 "

説 明

・ファイル中のデータを読み出すためにファイルをオープンします。

・ROPEN #文は、データ読み出しを行うための準備であり、読み出すべきファイルを〈装置名〉と〈ファイル名〉によって指定し、それを〈ロジカル番号〉に設定します。

・DEFAULT 文で指定された装置あるいはデフォルト状態にある装置を指定する場合は〈装置名〉は省略できます。

例

```
10  ROPEN#1,"QD:DATA"
```
……クイックディスクのBSDファイル" DATA "を読み出します。

```
10  ROPEN#1,"RS1:DATA1"
```
……INPUT #命令において入力をRS-232Cに設定します。

```
10  ROPEN #2,"DATA"
20  FOR I=1 TO 99
30  INPUT #2,A
40  PRINT A
50  NEXT I
60  CLOSE #2
70  END
```
……WOPEN #命令（155ページ）の例で作成されたファイルを読み出しデータを表示します。

参 考

INPUT #

WOPEN #

CLOSE #

PRINT #（print・プリント）

ファイルにデータを書き込む

書　式

PRINT #〈ロジカル番号〉，〈データ〉[，〈データ〉] ……

説　明

・WOPEN #文によって書き込み用にオープンされているファイル上へ、データを順次書き込んでいきます。

・書き込みを行う対象となるファイルは、オープンした時のロジカル番号で正しく指定します。

・〈データ〉は、数値データもしくは文字データです。

例

```
10  WOPEN  #1,"DATA"
20  PRINT  #1,1,2,3
30  CLOSE  #1
40  END
```

…… 〈ロジカル番号〉1に書き込み用にオープンされているファイルに数値データ1，2，3を書き込みます。

参　考

WOPEN #

CLOSE #

INPUT #（input・インプット）

ファイルからデータを読み出す

書　式

INPUT #〈ロジカル番号〉，〈変数〉[，〈変数〉，……]

説　明

・ROPEN #文によって、読み出し用にオープンされているファイルの先頭データから順次データを読み出し、〈変数〉に代入します。

・〈変数〉は配列要素でもかまいません。

・データの読み出しを行うファイルは、ROPEN#文で〈ロジカル番号〉に設定しているファイルになります。

・READ～DATA 文の場合と同様に、データと、〈変数〉のデータ型が一致しないとエラーが発生します。

・ファイルのデータの終りは EOF #関数で調べることができます。

例

```
10  ROPEN  #2, "DATA"
20  INPUT  #2, A, B, C
30  PRINT  A, B, C
40  CLOSE  #2
50  END
```

……ロジカル番号２に読み出し用にオープンされているファイルから数値データを読み出し、数値変数Ａ，Ｂ，Ｃに代入します。

参　考

ROPEN #

CLOSE #

EOF #

DIR （directroy・ディレクトリ）

ディレクトリの内容を表示する

書　式

DIR ［／P］［〈装置名〉］

ただし、装置名は QD、 RAM、 CMT のみ。

説　明

・各装置に登録されているファイルに関する情報、すなわちディレクトリ（台帳）の内容を表示します。

・／P を記述すると、プリンタにディレクトリの内容が出力されます。

・DEFAULT 文で指定された装置あるいはデフォルト状態にある装置を指定する場合は〈装置名〉は省略できます。

・DIR で指定した装置は DEFAULT 命令で指定したのと同じ意味をもちます。

・補足（ファイル分類の説明）

ＢＴＸ： BASIC プログラム

ＯＢＪ：機械語プログラム

ＢＳＤ： BASIC シーケンシャルデータ

例

ＤＩＲ　ＱＤ

……クイックディスクのディレクトリを行います。

参　考

DEFAULT

EOF (#) (end of file・エンド オブ ファイル)

データファイルのデータ終了をみつける関数

書 式 EOF (# 〈ロジカル番号〉)

省略形 EO. #

説 明

・ファイルを読み出す前に、データの終りを見つけるために使われる関数です。

・ファイル上のデータの読み出しを行ったとき、データが終了したにもかかわらずデータ読み出しを続けた場合にエラーは発生しないで、IF 文と組み合わされて使用した場合条件が真となります。

・ただしデータを読み出す装置が CMT の場合はエラーが発生します。

・この関数は、IF 文と組合せて使用され、各 INPUT #文の後に置きます。

例

```
10  ROPEN  #3,"DATA"
20  INPUT  #3,A
30  IF  EOF(#3) THEN  END
40  PRINT  A
50  GOTO  20
```

……このプログラムでは " DATA " というファイルから順次データを読み出し、画面に表示します。そして、読み出すデータがなくなったら（エンドオブファイル）この処理を終了します。

参 考 INPUT #

CLOSE #（close・クローズ）

ファイルをクローズする

書　式　CLOSE #〈ロジカル番号〉

省略形　CLO. #

説　明
- ロジカル番号にオープンされているファイルをクローズして、この番号を未定義状態に戻します。
- WOPEN #に対する CLOSE
  WOPEN #、PRINT #文によるファイルの書き込みを終了して、このファイルを正式に登録し、使用したロジカル番号を未定義に戻します。
- ROPEN #に対する CLOSE
  ROPEN #によってデータを読み出し用にオープンした BSD ファイルをクローズし、使用したロジカル番号を未定義にします。
- ロジカル番号を指定しない場合は、現在オープンされているすべてのファイルをクローズして、ロジカル番号をすべて未定義状態に戻します。

例
```
10 CLOSE #1
```
……ロジカル番号1のファイルをクローズします。
```
10 CLOSE
```
……すべてのファイルをクローズします。

参　考　WOPEN #
ROPEN #

RENAME （rename・リネーム）

RAM ファイル上のファイル名を変更する

書 式　RENAME " RAM：〈旧ファイル名〉 " , " 〈新ファイル名〉 "

説 明
- ファイル名変更は、現在のファイル名、新しいファイル名の順に指定します。
- 新しいファイル名は、その装置上にそれ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。

例　RENAME " RAM : OLDPROG " , " NEWPROG "

……RAM ファイル上の " OLD PROG " という名前のファイルを " NEW PROG " という名前に変更します。

KILL #（kill・キル）

ファイルの登録を中止する

書　式　KILL #［〈ロジカル番号〉］

説　明

・〈ロジカル番号〉にオープンされたファイルの登録を中止します。すなわち、WOPEN #文によってファイル作成を準備し、あるいはそれに続いてPRINT #を実行しているファイルの正式登録を途中で中止します。

・〈ロジカル番号〉を指定しない場合、現在作成中のファイルの正式登録を中止するとともに、すでにオープンされているすべてのファイルをクローズし、その〈ロジカル番号〉をすべて未定義状態に戻します。

・KILL命令は直接実行命令として使用できますので、カセットテープやクイックディスクなどのメディアを交換する直前にこの命令を実行させ、すべてのファイルをクローズすることによってメディア（記憶媒体）の内容を保護することができます。

例

```
KILL #3
```

……〈ロジカル番号〉#3に定義されファイルオープンされたファイルの正式登録を途中で中止し、ロジカル番号を未定義状態に戻します。

DELETE（delete・デリート）

RAMファイル上の指定されたファイルを消去する

書式　DELETE＂RAM：〈ファイル名〉＂

省略形　D.

説明　・〈ファイル名〉で指定されたファイルを消去します。ただしこの命令を実行するにはオプションのRAMファイル（MZ-1R18）が必要です。

例　DELETE＂RAM：SAMPLE＂

……RAMファイル上のファイル＂SAMPLE＂を消去します。

DEFAULT（default・デフォルト）

装置名の設定を行う

書　式　DEFAULT "〈装置名〉"

説　明

・命令文で〈装置名〉を省略した場合に有効となる装置名を定義します。

・装置名は、次のようになります。

QD：　クイックディスク
RAM：　RAM ファイル
CMT：　データレコーダ
LPT：　プリンタ
RS{1：/2：/3：/4：}　RS-232C インターフェイス

例　DEFAULT"QD："

……各命令で〈装置名〉を省略した場合には、クイックディスクとみなされます。

3 グラフィック命令

COLOR （color・カラー）

グラフィックの色を指定する

書式　COLOR ［〈パレットコード〉］［，〈重ね合せモード〉］

説明

・グラフィック命令(SET, RESET, LINE, BLINE, PATTERN, KPATTERN, BOX, PAINT, CIRCLE)で、カラー指定を省略した場合に有効になるパレットコードと重ね合せモードとを指定します。

・〈重ね合せモード〉は色の上に新しく色を重ねた場合の処理方法についての色指定を行います。0ならば、強制的に新しい色に変更します。1ならば、古い色と新しい色のOR（論理和）をとります。

・論理和とは…たとえば古い色が緑（カラーコード4）で新しい色が赤（カラーコード2）の場合は……

古い色　緑

カラーコード4 ―（2進数）→ 100 ┐
新しい色　赤　　　　　　　　　　の論理和は…
カラーコード2 ―（2進数）→ 010 ┘

```
  100
+ 010
  110は →カラーコード6に
          なるので黄色とい
          うことになります。
```

例

```
10  INIT  "CRT:G"
20  COLOR  3,0
30  FOR  I=0 TO 10  STEP2
40  SET  100,I
50  NEXT  I
60  END
```

…… 〈パレットコード〉3、重ね合せモード0で座標（100, 0）（100, 2）～（100, 10）に点を描きます。

例

```
10  CLR
20  PRINT"■"
30  INIT"CRT:G"
40  INPUT"パレット[0−7]";A
50  INPUT"カサネアワセ[0,1]";B
60  COLOR A,B
70  FOR Y=9 TO 199 STEP 20
80  LINE X,Y,X+319,Y
90  NEXT Y
100 GOTO 40
```

……行番号40でパレットコードを指定し、行番号50で重ね合せモードを設定して、行番号60のCOLOR文でカラーを設定しています。行番号80のLINE文で描かれる色は、行番号60で設定したカラーとなります。

CCOLOR (character color・キャラクタカラー)

文字とその背景の色を指定する

書　式

CCOLOR 〈X 座標〉, 〈Y 座標〉, 〈パレット 1〉, [, 〈パレット 2〉]

{パレット 1 ：文字の色（パレットコード）
{パレット 2 ：背景の色（パレットコード）

説　明

・画面上の文字とその背景の色指定を行います。

・〈パレット 1〉で文字の色のパレットコードを、〈パレット 2〉で背景の色のパレットコードを設定します。

・〈パレット 1〉を省略すると、文字の色のパレットコードは 7 に指定され、〈パレット 2〉を省略すると背景の色はパレットコード 0 に指定されます。

・座標指定を省略して色指定のみ行なった場合は、現在画面上に表示されているものに対しては何ら影響はなく、次にくる RRINT 命令 CSET 命令、CRESET 命令……などで色指定を省略した場合に有効となります。

・座標指定を行って色指定を行った場合は、指定された座標にある文字のみ指定した色に変更します。ただしこの場合は PRINT 命令、CSET 命令、CRESET 命令などで色指定を省略した場合に有効となるデフォルト値には何ら影響を与えません。

例

```
10  CURSOR  20, 10
20  PRINT  "A"
30  CCOLOR  20, 10, 2, 6
40  END
```

……行番号30でその位置の文字の色をパレットコード 2 に、背景の色をパレットコード 6 に指定します。

参　考

COLOR

PAL（palette・パレット）

パレットコードを設定する

書式

PAL［カラーコード〉，［カラーコード〉，［カラーコード〉，……［カラーコード］

説明

・パレットコード0～7に対応するカラーコード0～7を設定します。

カラーコード	0	1	2	3	4	5	6	7
色	黒	青	赤	紫	緑	水色	黄	白

・〈パレットコード〉を全部省略した場合には、次の設定となります。

```
PAL 0,1,2,3,4,5,6,7
```

・変更しない〈パレットコード〉は省略可能です。

例

```
10 PAL 0,1,6,3,4,5,6,7
```

……パレットコード2の色をカラーコード6の色（黄）に設定します。

```
10 PRINT"C"
20 GOSUB"PRI"
30 PAL
40 CURSOR 10,5:PRINT[1,2]
   "AAAABBBBCCCCDDDD";
50 CURSOR 10,6:PRINT[1,2]
   "AAAABBBBCCCCDDDD";
60 GOSUB"PAL"
70 GOSUB"MESSAGE"
80 PAL,M,H
90 GOSUB"GET"
100 GOTO 60
110 LABEL"PAL"
120 M=INT(RND(1)*8)
130 H=INT(RND(2)*8)
140 RETURN
```

```
150 LABEL"MESSAGE"
160 CURSOR 13,8:PRINT"M=";M,"H
    =";H
170 RETURN
180 LABEL"GET"
190 GET A$:IF A$=""THEN 190
200 RETURN
210 LABEL"PRI"
220 CURSOR 12,13:PRINT
   "PUSH ANY KEY !"
230 CURSOR 12,18:PRINT
   "M ハ モジノイロ"
240 CURSOR 12,20:PRINT
    "H ハ ハイケイノイロ"
250 RETURN
```

参考

RND 関数

GET

SET（set・セット）

指定した色で任意の1ドットをセットする

書　式

SET［〈カラー指定〉］〈X座標〉，〈Y座標〉

ただし、〈カラー指定〉＝〔［〈パレット〉］［，〈重ね合せ〉］〕

説　明

・グラフィックモード時に、画面2上の任意の1ドットをセットします。

・〈X座標〉〈Y座標〉は数値データ（数値定数・数値変数・算術式）であり、指定範囲は次のとおりです。

　−16384≦＜X座標＞≦16383

　−16384≦〈Y座標〉≦16383

・〈パレット〉では、セットするドットの色のパレットコードを指定します。

・〈重ね合せ〉は、0または1で指定します。0を指定すると、現在どのパレットコードがセットされているかにかかわらず、強制的に、〈パレットコード〉で指定された色を表示します。

1を指定すると、現在セットされているパレットコードに、〈パレットコード〉の色を重ね合わせます。（パレットコードの論理和（OR 53ページ参照）をとったパレットコードが指定されます。）

・〈カラーコード〉を省略するとCOLOR文で指定されたカラー指定にしたがいます。

例

```
10  SET［4，0］100，50
```

……座標（100，50）の点をパレットコード4の色、重ね合せモード0でセットします。

```
10  PRINT"■"
20  INIT"CRT：G"
30  GOSUB"SET"
40  GOTO  30
50  LABEL"SET"
60  INPUT"パレット［0−7］"；A
70  INPUT"カサネアワセ［0，1］"；B
```

```
80 FOR X=0 TO 319 STEP 2
90 SET [A,B] X,100
100 NEXT X
110 RETURN
```

……SET 文を使って点線を描きます。

参考　RESET

RESET（reset・リセット）

指定した色で任意の1ドットをリセットする

書　式

RESET［〈カラー指定〉］〈X座標〉，〈Y座標〉

ただし、〈カラー指定〉＝〔［〈パレット〉］［，重ね合せ〉］〕

説　明

・グラフィックモード時に、画面2上の任意の1ドットをリセットします。

・〈X座標〉〈Y座標〉は数値データ（数値変数・数値定数・算術式）であり、指定範囲はSET文（171ページ）と同じです。

・〈パレット〉では、リセットする色のパレットコードを指定します。

・〈重ね合せ〉は、0または1で指定します。1を指定すると、〈パレットヲで指定された色をリセットします。

0を指定すると、〈パレット〉で指定された色の反転色をセットします。パレットコードNの反転色とは、パレットコード（7-N）の色です。たとえば、パレットコード2の反転色は、パレットコード5（＝7－2）となります。

例

```
10 RESET［2，0］100，50
```

……座標（100，50）の点をパレットコード2の反転色パレットコード5の色でセットします。CRT：G"：PAL

```
100 PRINT"C"
110 INIT"RT：G"：PAL
120 GOSUB"SET"
130 GOSUB"RESET"
140 GOTO 120
150 LABEL"SET"
160 INPUT"SET パレット［0－7］"；A
170 INPUT"　　カサネアワセ［0，1］"；B
180 FOR X＝0 TO 319 STEP 2
190 SET［A，B］X，100
200 NEXT X
210 RETURN
```

```
220 LABEL"RESET"
230 INPUT"RESET パレット[0—7]";C
240 INPUT"          カサネアワセ[0,1]";D
250 FOR X=0 TO 319 STEP 2
260 RESET[C,D]X,100
270 NEXT X
280 RETURN
```

参考　SET

CSET（character set・キャラクタセット）

簡易グラフィックで点をセットする

書　式

CSET 〈X座標〉，〈Y座標〉[，〈パレットコード〉]

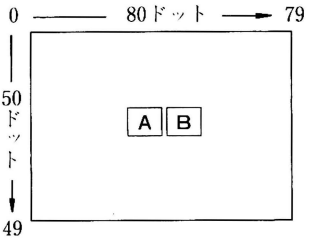
〈X座標〉：0～79

〈Y座標〉：0～49

説　明

・簡易グラフィックで、点を〈パレットコード〉の色にセットします。

・簡易グラフィックとは、1文字の1/4を1ドットとして扱い、文字として表します。この文字を使って簡単な図形や絵を描くことができます。



・1文字が4分割となっているため、同一文字の1分割の色を変えた場合、ほかの3ドット分も色が変わります。

・〈パレットコード〉を省略すると、CCOLOR文で指定した色となります。

例

```
10 CSET 20,10,1
```

……簡易グラフィックの座標（20，10）をパレットコード1の色で点を描きます。

参　考

CRESET

CCOLOR

CRESET (character reset・キャラクタリセット)

簡易グラフィックで点をリセットする

書式

CRESET 〈X座標〉, 〈Y座標〉[, 〈パレットコード〉]

〈X座標〉: 0～79

〈Y座標〉: 0～49

説明

・簡易グラフィックで、点をリセットします。すなわち現在の画面の背景の色に変更します。

例

```
10 CRESET 20,10
```

……簡易グラフィックの座標（20, 10）をリセットします。

参考

CSET

LINE（line・ライン）

指定された色で任意の直線を描く

書　式

LINE［〈カラー指定〉］〈X 座標〉，〈Y 座標〉，〈X 座標〉，〈Y 座標〉［，〈X 座標〉，〈Y 座標〉］……

ただし、〈カラー指定〉＝〔［〈パレット〉］［，〈重ね合せ〉］〕

説　明

・グラフィックモード時に、画面 2 上の任意の点を結ぶ直線を、指定された色で描きます。

・〈X 座標〉〈Y 座標〉は数値データ（数値定数・数値変数・算術式）であり、指定範囲は SET 文と同じです。

・〈カラー指定〉の機能は SET 文と同じです。これを省略すると COLOR 文のカラー指定にしたがいます。

・表示エリア外に座標が指定された場合は、表示エリアの線分のみが描かれます。

例

```
10  LINE[6,0]10,20,260,180,
    380,60
20  END
```

……座標（10, 20）と（260, 180）と（380, 60）を結ぶ直線を、パレットコード 6 の色、重ね合せモード 0 で描きます。

```
10  INIT"CRT:G"
20  FOR X1=0 TO 319 STEP 3
30  LINE 159,99,X1,0
40  NEXT X1
50  FOR Y1=0 TO 199 STEP 3
60  LINE 159,99,319,Y1
70  NEXT Y1
80  FOR X2=319 TO 0 STEP -3
90  LINE 159,99,X2,199
100 NEXT X2
```

```
110 FOR Y2=199 TO 0 STEP -3
120 LINE 159,99,0,Y2
130 NEXT Y2
```

……画面の中心座標（159，99）から画面のすみまで3ドット間隔で直線を引きます。

参考　BLINE

SET

BLINE (blank line・ブランクライン)

指定した色で任意のラインを消去する

書式

BLINE [〈カラー指定〉] 〈X 座標〉, 〈Y 座標〉, 〈X 座標〉, 〈Y 座標〉 [, 〈X 座標〉, 〈Y 座標〉] ……

ただし、〈カラー指定〉 = 〔[〈パレット〉] [, 〈重ね合せ〉]〕

説明

グラフィックモード時に、画面 2 上の任意の点を結ぶ直線を消去します。

- 〈X 座標〉〈Y 座標〉は数値データ（数値定数・数値変数・算術式）であり、指定範囲は SET 文と同じです。
- 〈カラー指定〉の機能は RESET 文と同じです。これを省略すると COLOR 文のカラー指定にしたがいます。
- 表示エリア外に座標が指定された場合は、表示エリアの線分のみが消去されます。

例

```
10 BLINE [5,0] 10,10,100,100
```

……パレットコード 5 の色の反転色パレットコード 2 の色で直線を描きます。

```
10 LINE [6,0] 0,0,319,199
20 BLINE [6,2] 32,20,192,120
30 END
```

……LINE 文で描いた線の一部を消去します。

```
10 INIT"CRT:G"
20 PRINT"C"
30 CURSOR 19,12:PRINT"◆"
40 GOSUB 110
50 X=INT(RND(1)*320)
60 Y=INT(RND(2)*200)
70 LINE 155,99,X,Y
80 GOSUB 110
```

```
90 BLINE 155,99,X,Y
100 GOTO 40
110 GET A$:IF A$="" THEN 110
120 RETURN
```

……画面上の中心座標（159, 99）からランダムに発生させた座標へ直線を描きます。その後任意のキーを押すことにより実行がくり返されます。

参考

LINE

RESET

RND 関数

GET

BOX（box・ボックス）

指定された2点を頂点とする四角形を描く

書　式

BOX［〈カラー指定〉］〈X座標1〉,〈Y座標1〉,〈X座標2〉,〈Y座標2〉［,〈パレットコード〉］

ただし、〈カラー指定〉＝〔［〈パレット〉］［,〈重ね合せ〉］〕

説　明

・グラフィックモード時に、画面2上の指定された2点（〈X座標1〉,〈Y座標1〉）と（〈X座標2〉,〈Y座標2〉）を頂点とする四角形を描きます。

・〈カラー指定〉の機能はSET文と同じです。これを省略するとCOLOR文のカラーコントロールにしたがいます。

・最後に〈パレットコード〉を記述すると、内部をその色でぬりつぶします。省略すると枠のみを描きます。

例

```
10  INIT"CRT:G"
20  PRINT"C"
30  BOX[1,0]20,20,60,60,4
40  END
```

……画面上に四角形を描きその中をパレットコード4の色でぬりつぶします。

参　考

SET

CIRCLE（circle・サークル）

スクリーン上に、だ円（円）または、円弧（扇）を描く

書　式

CIRCLE 〈カラー指定〉〈X 座標〉,〈Y 座標〉,〈半径〉[,〈比率〉][,〈開始角度〉,〈終了角度〉[0]]

ただし、〈カラー指定〉＝〔[〈パレット〉][,〈重ね合せ〉]〕

説　明

・グラフィックモード時の画面 2 上に、だ円（円）または、円弧（扇）を描きます。

・〈カラー指定〉の機能は SET 文と同じです。これを省略すると COLOR 文のカラー指定にしたがいます。

・〈X 座標〉,〈Y 座標〉は円の中心の座標を、〈半径〉は、円の半径を指定します。範囲は下記の通りです。

$-16383 \leqq$ 座標 $\leqq 16383$

$0 \leqq$ 半径 $\leqq 16383$

ただし画面上に表示される範囲は（0，0）～（319，199）です。

・〈比率〉は、（Y 軸方向の半径）／（X 軸方向の半径）です。〈比率〉が 1 以下の場合は、〈半径〉は X 軸方向の半径を、〈比率〉が 1 以上の場合は、〈半径〉は Y 軸方向の半径を示します。

・〈比率〉を省略すると自動的に 1 に設定されます。

・〈開始角度〉,〈終了角度〉を指定すると円弧または扇を描きます。角度の単位はラジアンです。省略すると、だ円（円）を描きます。

・〈開始角度〉,〈終了角度〉を指定し、0 を記述すると扇を描き、0 の記述を省略すると円弧を描きます。

例

```
10  INIT"CRT:G"
20  CIRCLE[1,0]100,100,80,.5
30  GOSUB 80
40  CIRCLE[4,0]50,130,60,.5,0,
    π/4,0
50  GOSUB 80
60  CIRCLE 159,99,50
70  END
```

```
80 GET A$:IF A$="" THEN 80
90 RETURN
```

……円、だ円、円孤（扇）を描きます。

参考　SET

GET

PAINT（paint・ペイント）

指定された境界色で囲まれた範囲内を指定された色でぬる

書式

PAINT［〈カラー指定〉］〈X座標〉，〈Y座標〉，〈境界色〉［，〈境界色〉］……

ただし、〈カラー指定〉＝〔［パレットコード〉］　，［〈重ね合せ〉］〕

〈境界色〉はパレットコードで指定する。

説明

［〈パレットコード〉］の記述を省略すると、COLOR文で設定されたカラー指定にしたがいます。

・〈境界色〉は最大7色まで指定できます。

・〈X座標〉，〈Y座標〉によって指定された座標を囲む境界色の範囲のエリアを〈パレットコード〉によって指定された色でぬりつぶします。その際、そのエリアが指定された境界色によって完全に囲まれた状態（これを閉ループ状態といいます。）となっていなければ、そのエリア外に対してもぬりつぶしを行います。

・〈X座標〉，〈Y座標〉を境界線上またはすでに境界色で描かれている場所に指定した場合は、ぬりつぶしを行いません。

・図形は、すべてドット単位で描かれています。したがって、たがいに近接した直線や曲線が表示されているとき、その交点付近に、いくつかの小さな閉ループが構成されることがあります。この場合それらの閉ループ内の座標を指定しなければ、ぬりつぶしは行いません。

例

```
10  INIT"CRT:G"
20  PRINT"■"
30  CIRCLE[2,0]160,100,50
40  PAINT[4,0]160,100,2
50  END
```

……座標（160，100）を、パレットコード2の色で囲むエリアを、パレットコード4の色でぬりつぶします。

PATTERN（pattern・パターン）

グラフィックパターンを定義する

書　式

PATTERN［〈カラー指定〉］，〈積み重ね段数〉〈数値データ〉〈文字データ〉〔，［〈カラー指定〉］〈数値データ〉〈文字データ〉……〕

ただし、〈カラー指定〉＝〔［〈パレットコード〉］，［〈重ね合せ〉］〕

説　明

・グラフィックモード時に、画面2上に、指定された色で任意のグラフィックパターンを描きます。

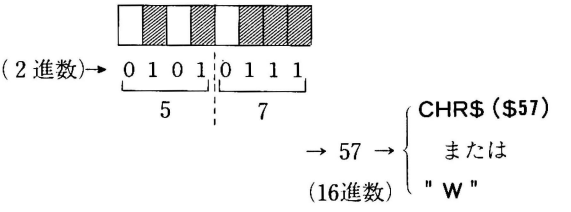
・〈カラー指定〉の機能はSET文と同じです。これを省略するとCOLOR文のカラー指定にしたがいます。

・パターンの指定は〈数値データ〉と〈文字データ〉で指定され、〈数値データ〉は積み重ね段数（±1～±24）を表し、文字データは8ビット単位のドットデータを表します。

・表示はグラフィックのポジションポインタより行われ、〈数値データ〉が正の場合には下から上に、負の場合には上から下に積み段数の数だけ表示されます。

・段数を越えると、8ビット分右の位置（1文字分右）からデータを表示します。

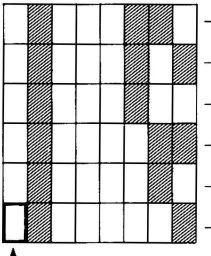
・ブラフィックデータを表す〈文字データ〉は、セット・リセットを2進数で表したものを、アスキーコードとみなして表します。



例

```
10 POSITION 100,100
     (ポジションポインタ指定)
20 PATTERN[4,0],6,"ABCDEF"
30 END
```

……このプログラムは下のパターンをパレット4の色で描くプログラムです。



	2進数	16進数または文字データ
→	0 1 0 0 0 1 1 0	→ 4 6 → " F "
→	0 1 0 0 0 1 0 1	→ 4 5 → " E "
→	0 1 0 0 0 1 0 0	→ 4 4 → " D "
→	0 1 0 0 0 0 1 1	→ 4 3 → " C "
→	0 1 0 0 0 0 1 0	→ 4 2 → " B "
→	0 1 0 0 0 0 0 1	→ 4 1 → " A "

実行前の位置ポインタ

ただし、行番号20は次の2通りの文でもそれぞれ同じです。

```
20 PATTERN[4,0],6,CHR$($41,$42,
   $43,$44,$45,$46)

20 PATTERN[4,0],6,HEXCHR$("4142
   434445 46")
```

参考 POSITION

POSITION （position・ポジション）

スクリーン上のポジションポイントの位置を設定する

書　式　POSITION 〈X 座標〉, 〈Y 座標〉

省略形　POS.

説　明
- PATTERN 文によるドットパターンの表示位置を決めるためポジションポイントの位置を設定します。
- ポジションポイントとは、グラフィックモード時での画面 2 上の座標を示すポインタで PATTERN 文で使用されます。

例

```
10  POSITION  100,50
20  A$="ABCDEFGH"
30  PATTERN [4,0]-8,A$
40  END
```

……行番号10でポジションポイントが(100, 50)に設定され、その位置から PATTERN 文が実行されます。

KPATTERN（kanji pattern・漢字パターン）

漢字を画面上に表示する

書式

KPATTERN　[〈カラー指定〉]〈文字データ〉
……，[〈カラー指定〉]〈文字データ〉

ただし、〈カラー指定〉＝〔[パレットコード〉]，[〈重ね合せ〉]〕

説明

〈カラー指定〉で指定した色で漢字パターンをグラフィック画面に表示します。

〈文字データ〉はCHR$（27ページ）によって指定した16進のコードを与えます。（漢字コード表165ページ参照）

グラフィック画面への書き込み位置はPOSITION文で与えます。なお1つの漢字パターンは16×16ドットで表示されます。

偶数のデータに対して、それぞれ別個にカラーコントロールを省略した場合は、COLOR文で指定したカラーコントロールにしたがって漢字パターンを表示します。

この命令を実行するにはオプションの漢字ROMボード（MZ-1R23）が必要です。

```
10  INIT"CRT:G"
20  KPATTERN[4,],CHR$($30,$20)
30  END
```

……JIS漢字コード（$3020）の漢字が表示されます。

FONT$（font・フォント）

PCGのパターンを定義したり、参照したりする

書式　FONT$〈タイプ〉,〈コード〉

説明

・8ビット×8ビットのパターンを定義したり、CGやPCGのパターンを参照したりします。

・〈タイプ〉には下記のものがあり、それに対応するコードの範囲は下表の通りです。

〈タイプ〉	意味	〈コード〉の範囲
0 （画面1）	ROM・CG（I） 参照のみ。定義は不可	0～255
1 （画面1）	ROM・CG（II） 参照のみ。定義は不可	0～255
2 （画面2）	PCG イメージモード時のみ可	1～1000
3 （画面2）	PCG グラフィックモード時 のみ可	1～24

・CG（キャラクタジェネレータ）は、画面1で使われます。

・PCG（プログラマルキャラクタジェネレータ）は画面2で使われます。

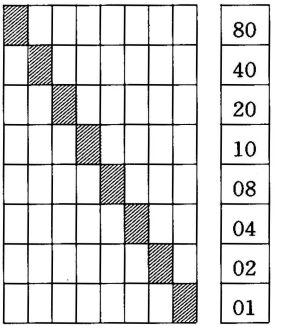
例

```
10  INIT"CRT:G"
20  P0$=CHR$($80,$40,$20,$10,
    $08,$04,$02,$01)
30  P1$=CHR$($20,$10,$88,$44,
    $22,$11,$08,$04)
40  P2$=CHR$($08,$04,$02,$01,
    $80,$40,$20,$10)
50  FONT$(3,1)=P0$+P1$+P2$
```

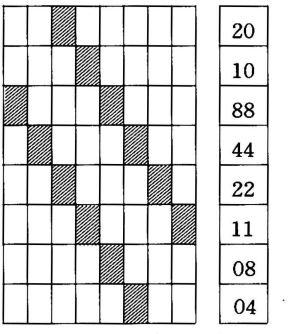
……グラフィックPCGのコード1として定義します。

```
60  FPRINT  1
70  END
```

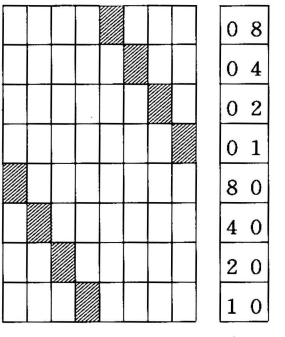
PCGはプレーン0，1，2各々8バイト，計バイトで定義します。
定義は上から行います。



プレーン0（P 0 $）

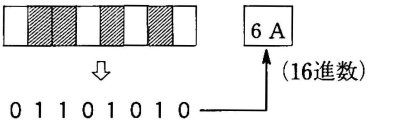


プレーン1（P 1 $）



プレーン2（P 2 $）

注1）
各データの意味は次の通りです。



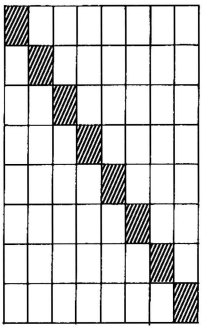
ドットがセットされているのを1、
リセットされているのを0とします。

注２）

プレーンと色の関係は次の通りです。

パレットコード	0	1	2	3	4	5	6	7
プレーン０	0	1	0	1	0	1	0	1
プレーン１	0	0	1	1	0	0	1	1
プレーン２	0	0	0	0	1	1	1	1

セットされているのを１、リセットされているのを０とします。
たとえば、パレットコード５の色で下図の位置をセットするならばプレーン０とプレーン２の次に示す位置をセットしてください。



（プレーン０とプレーン２をセットする）

注３）

行番号20～40で **CHR$** の代りに **HEXCHR$** を使用することもできます。たとえば、行番号20は次のようになります。

```
20  PO$=HEXCHR$("8040201008040201")
```

参考　FPRINT

FPRINT （font print・フォントプリント）

第2画面にFONT$で指定したCG・PCGを表示する

書　式

FPRINT［〈X座標〉，〈Y座標〉］〈キャラクタ〉，〈キャラクタ〉…
…［〈X座標〉，〈Y座標〉］〈キャラクタ〉，〈キャラクタ〉……

説　明

・現在FONT$で設定されているCGまたはPCGを、X，Y座標の位置より表示します。ただし、座標を省略すると前に表示した次の位置より表示します。
この場合、キャラクタ画面のカーソル位置に何ら影響をしません。

・FPRINT 0で、その位置のキャラクタを消去します。

例

```
10  INIT"CRT:G"
20  P0$=CHR$($80,$40,$20,$10,$0
    8,$04,$02,$01)
30  P1$=CHR$($20,$10,$88,$44,$2
    2,$11,$08,$04)
40  P2$=CHR$($08,$04,$02,$01,$8
    0,$40,$20,$10)
50  FONT$(3,1)=P0$+P1$+P2$
60  FPRINT 1
70  GET A$:IF A$="" THEN 70
80  FPRINT 10,10,0
90  END
```

……グラフィックPCGのコード1として定義します。

POINT （point・ポイント）

グラフィック画面上の点にセットされているパレットコードを与えます

書　式

POINT 〈X 座標〉, 〈Y 座標〉

説　明

・〈X 座標〉, 〈Y 座標〉で示すグラフィックスクリーン上の点にセットされているパレットコードを与えます。

例

```
10  INIT"CRT:G"
20  SET[3,0]100,100
30  A=POINT(100,100)
40  PRINT A
50  END
RUN
3
Ready
```

……行番号10で座標（100, 100）にパレットコード3の色がセットされるのでPOINT文を実行すると数値3が得られます。

4 ミュージック命令

MUSIC （music・ミュージック）

音楽を演奏する

書　式

MUSIC 〈メロディー1〉［；〈メロディー2〉］……［；〈メロディー6〉］，〈メロディー1〉［；〈メロディー2〉］……

〈メロディー〉＝［{±}］〈音名〉［〈音長〉］

説　明

・6つのパートのメロディーも同時に演奏できます。

・各パートのメロディーの間は、セミコロン（；）で、また、1つのメロディーから次のメロディーへ移る場合の区切りは、コンマ(，)で表します。

・メロディーは音の続きで表し、音はオクターブ(＋または－)、音名、音長で表します。なお、基準となるオクターブはOn（n＝0～6）で指定します。

（i）オクターブ（＋，－）

＋　：1つ上のオクターブ（基準がO6の場合は使用不可）

省略：基準のオクターブ

－　：1つ下のオクターブ（基準がO0の場合は使用不可）

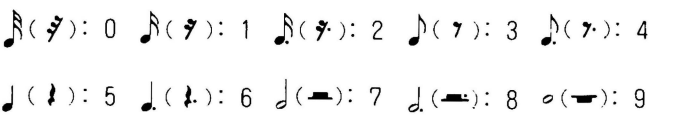
（ii）音名

音	ド	レ	ミ	ファ	ソ	ラ	シ	休符
音名	C	D	E	F	G	A	B	R

前に、#をつけるとシャープ（半音あげる）を表します。ただし、半音下げる場合は一つ下の音にシャープを付けて表します。たとえば、シ（B）のフラットならば、#Aで表します。

（iii）音長

音長は、0～9の数字で指定します。



・また、メロディーの中に次の指定をすることもできます。

	n＝値 （整数）	意　　味
On	n＝0～6	オクターブを指定します。省略すると2とみなされる。 O 2でA（ラ）の音が440Hzです。
Nn	n＝0～83	オクターブと音名からではなく、直接音程を指定する。nの値は次ページの音程表の通りですが、n＝0～8は休止符となる。
Tn	n＝1～7	TEMPO文と同じ。（199ページ参照）
Vn	n＝0～15	音量を設定する。15で最大、0で無音状態となる。
Sn	n＝0～7	エンベロープパターン（音の波形）を設定する。nの値は次ページのエンベロープ表の通り。 エンベロープ表参照
Mn	n＝1～255	Sコマンドとペアで使用し、エンベロープパターンの変化の速さを指定する。1で速度は最大であり、255に近づくにつれてゆるやかになり、図形の傾きもゆるやかになる。
Ln	n＝0～9	音の長さを指定する。nの指定は〈音長〉の指定と同じ。

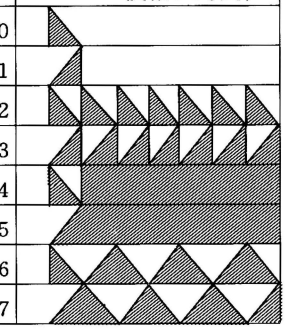
例　フォスター作曲「おおスザンナ」の一部をプログラムしてみます。

```
10  TEMPO  6
20  A1$="O2G1A1B3+D3+D3+E3+D3B3
    G4A1B3B3A3G3A6G1A1B3+D3+D3+
    E3+D3B3G4A1B3B3A3A3G6R3"
30  A2$="G1A1G3B3B3+C3B3G3G4G1G
    3G3G3G3E6G1A1G3B3B3+C3B3G3G
    4G1G3G3E3E3G6R3"
40  MUSIC  A1$;A2$
50  END
```

・音程表（Nn）

音程＼オクターブ	0	1	2	3	4	5	6
ド		12	24	36	48	60	72
ド#		13	25	37	49	61	73
レ		14	26	38	50	62	74
レ#		15	27	39	51	63	75
ミ		16	28	40	52	64	76
ファ		17	29	41	53	65	77
ファ#		18	30	42	54	66	78
ソ		19	31	43	55	67	79
ソ#		20	32	44	56	68	80
ラ	9	21	33	45	57	69	81
ラ#	10	22	34	46	58	70	82
シ	11	23	35	47	59	71	83

・エンベロープ表（Sn）

n	エンベロープ パターン（縦軸は音量、横軸は時間）
0	
1	
2	
3	
4	
5	
6	
7	

・一度 MUSIC 文を実行すると、音楽データはバッファに記憶され、コンピュータの処理とは関係なく音を発生させます。ですから、音楽を演奏しながら画面を動かすという処理なども可能になるのですが、逆に SHIFT ＋ BREAK キーで止めたり、画面と合わせて音楽を演奏するということもできません。

そこで、使用されるのが下の２つです。

MUSIC STOP：音の発生を止めます。

MUSIC WAIT：この命令では、音楽の演奏が終るまで、プログラムを一時中止して、演奏が終了してから次の命令を実行します。

MUSIC INIT：ミュージックおよびノイズを初期状態に戻します。全チャンネルに対し、" V15L5T4S8M255 " を実行した場合と同じです。

SOUND（sound・サウンド）

音を発生させる

書　式

SOUND 〈音程〉，〈音長〉

SOUND＝（〈レジスタ〉，〈データ〉）

説　明

・〈音程〉と〈音長〉とで指定した音を発生させます。

（ｉ）音程

音程＼オクターブ	1	2	3	4	5	6
ド		24	36	48	60	72
ド#		25	37	49	61	73
レ		26	38	50	62	74
レ#		27	39	51	63	75
ミ		28	40	52	64	76
ファ		29	41	53	65	77
ファ#		30	42	54	66	78
ソ		31	43	55	67	79
ソ#		32	44	56	68	80
ラ	21	33	45	57	69	81
ラ#	22	34	46	58	70	82
シ	23	35	47	59	71	83

オクターブ２のとき、ラの音が440Hz となります。

（ii）音長

秒の単位で長さを指定します。（０～65535）

（〈レジスタ〉，〈データ〉）で指定すると、音発生用の LSI である PSG（プログラマブルサウンドジェネレータ）の内容を直接指定することができます。

PSGは2個(PSG 0，PSG 1)あり、各々3音と1ノイズを発生させることができます。

各レジスタとデータを次に示します。レジスタ0～7はPSG 0の、レジスタ8～15はPSG 1のレジスタを表します。

レジスタ				データ
PSG 0	PSG 1			
0	8	トーン0	周波数	1～$2^{10}-1$の整数
1	9	トーン0	音　量	0～15の整数(注)
2	10	トーン1	周波数	レジスタ0と同じ
3	11	トーン1	音　量	レジスタ1と同じ
4	12	トーン2	周波数	レジスタ0と同じ
5	13	トーン2	音　量	レジスタ1と同じ
6	14	ノイズ	周波数	ノイズデータ
7	15	ノイズ	音　量	レジスタ1と同じ

周波数は下記の2バイトのデータで設定します。

第1バイト

D7	D6	D5	D4	D3	D2	D1	D0
1	レジスタ			周　波　数　1			
	R2	R1	R0				

第2バイト

D7	D6	D5	D4	D3	D2	D1	D0
0	×	周　波　数　2					

$$周波数=\frac{3.58}{32\times データ}(kHz)$$

ただし、nは上位6ビットを周波数2、下位4ビットを周波数1とする10ビットデータ。

注）音量は0で最大。15で無音状態となる。

データ	0	1	2	3	4	5	6	7	8	9	10
減衰量(dB)	0	2	4	6	8	10	12	14	16	18	20

11	12	13	14	15
22	24	26	28	30

TEMPO （tempo・テンポ）

MUSIC 文で演奏する音楽の演奏速度を決める

書　式　TEMPO 〈数値データ〉

ただし、〈数値データ〉は１～７の整数

省略形　TEM.

説　明

・MUSIC 文の実行は、６つのパートとも TEMPO 文の指定されたテンポにしたがいます。

・BASIC の起動時には、テンポ＝４（中位の速度）がデフォルト値となっていますが、〈数値データ〉＝１～７の範囲で任意に設定し直すことができます。

・テンポ１が最も遅く、テンポ７が最も速いテンポです。

```
TEMPO  1……最も遅い
  ⋮
TEMPO  4……中位の速さ
  ⋮
TEMPO  7……最も速い
```

NOISE （noise・ノイズ）

ノイズを発生する

書　式

NOISE 〈メロディー1〉[　〈メロディー2〉]
[,〈メロディー1〉[；〈メロディー2〉]

〈メロディー〉= { ＋ / － } 〈音名〉[〈音長〉]

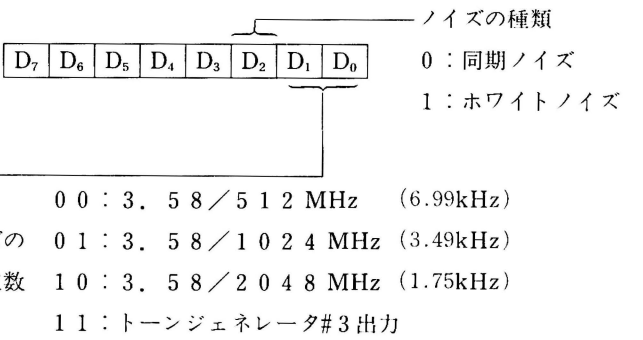
説　明

・〈メロディー〉の設定にしたがったホワイトノイズを発生させます。
〈メロディー〉の設定はMUSIC文と同じです。

・2つのノイズを同時に発生させることができ、2つのパートのメロディーの間は、セミコロン（；）で、また、1つのメロディーから次のメロディーへ移る場合の区切りは、コンマ（，）で表します。

ノイズは、同期ノイズまたはホワイトノイズを発生させることができます。下記の1バイトデータで設定を行います。

D7	D6	D5	D4	D3	D2	D1	D0
1	レジスタ			減衰量			
	R2	R1	R0				



例

```
10 NOISE"C3D1";"E3F1"
```

……2つのノイズを同時に発生させます。

VOICE（voice・ボイス）

指定の文字列、または固有メッセージを発声する

書　式

VOICE［〈スピードコード〉，〈数値読みパラメータ〉］
〈発生する文字列〉［，〈発生する文字列２〉，…

ただし、〈スピードコード〉１～３　１遅い　２標準　３速い
〈数値読みパラメータ〉　１棒読み　２ケタ読み

VOICE　@〈メッセージ固有の番号〉，…

説　明

①指定の文字列を発声します。

・文字列中には、カナ、アルファベット、数字、特殊文字（＋－＊／＝・）のいずれも許されます。

・ただし、アルファベットは日本語読み、数字はパラメータにより棒読みまたは桁読みされます。なお、数値桁読みの際、整数部が８桁をこえる場合は自動的に棒読みとなりますが、モードは変わりません。

・スペースは無音部となり、上記の発音文字以外は無視します。

・なお、〈スピードコード〉，〈数値読みパラメータ〉はそれぞれ省略可能で、省略時は直前値となります。

・起動時は〈スピードコード〉＝２，〈数値読みパラメータ〉＝１に設定されます。

②固定メッセージを発生します。ただし、スピードは標準のときと同じで一定です。

番号と固定メーセージの関係は次の通りです。

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	もう1回チャレンジしますか	19	点
		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）
11	命中	27	自動車（停止）
11	お好きなKEYを押して下さい	28	メロディー(葬送行進曲)
		29	メロディー（デキシーランド）
12	YかNのキーを押してください	30	メロディー（蛍の光）
13	最高得点	31	メロディー（おおスザンナ）
14	ボーナス点		
15	あなたの番です	32	メロディー（故郷の人々）
		33	メロディー（村のかじ屋）

例

```
10 VOICE [1] "ABCDE","アイウエオ", "123"
```

……"ABCDEF","アイウエオ","123"を発声します。

```
10 VOICE @4
```

……"よくできました"と発声します。

注) VOICEコマンドを実行させるには、ボイスボードMZ-1M08（オプション）が必要です。

5 プリンタ命令

PTEST（pen test・ペンテスト）

プロッタプリンタのテストを行う

書　式　PTEST

省略形　PTE.

説　明
・色の指定のチェックやインキ量、インキの乾燥をチェックするための命令で黒，青，緑，赤の順に正方形を描きます。
・テキストモードで有効です。



0　1　2　3
黒　青　緑　赤

例　PTEST
……ペンのチェックを行います。

PMODE (printer mode・プリンタモード)

プロッタプリンタのモード設定を行う

書 式

PMODE { TN / TL / TS / GR }

省略形

M.

説 明

・プロッタプリンタのモード設定を行います。

・TN ならば、グラフィックモードからテキストモードに戻し、1ライン40文字とします。

・TL ならば、グラフィックモードからテキストモードに戻し、1ライン26文字とします。

・TS ならば、グラフィックモードからテキストモードに戻し、1ライン80文字とします。

・GR ならば、テキストモードからグラフィックモード(図形を描くモード)に切り換えます。このモード切り換え直前の1ラインの文字数を BASIC プログラムの中で覚えておく必要があります。これは、ブレーク時または STOP 命令時に停止した場合、テキストモードに戻るためです。

```
*** CHARACTER MODE ***

90 ﾌﾂ ﾓｰﾄﾞ
ABCDEFGHIJKLMNOPQRSTUVWXYZ

40 ﾌﾂ ﾓｰﾄﾞ
ABCDEFGHIJKLMNOPQRSTUVWXYZ

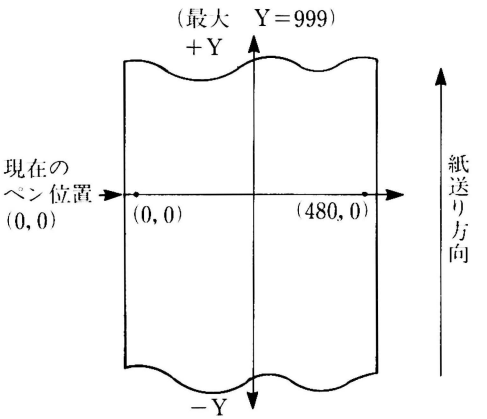
26 ﾌﾂ ﾓｰﾄﾞ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

例

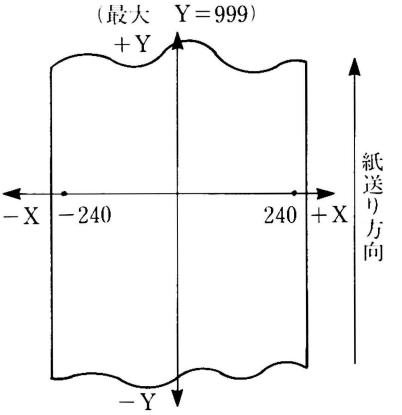
```
PMODE  TN
```

……1ライン40文字のテキストモードに設定します。

グラフィックモードで使用する命令は、MODE GR によって、有効となり、プリンタに X-Y 軸 が設定されます。 現在のペン位置（左端）を X＝0、 Y＝0に設定します。初期値はこのように設定されていますが、自由に原点を移動できますのでいろいろな使い方ができます。ただし、現在のペン位置から範囲外の X, Y を指定しないでください。紙がないところに印字させると、ペンをいためたり、故障の原因になりますので、範囲内の X, Y を指定してください。



MODE GR 命令実行後の X-Y 軸
このときは X＝0～480, Y＝－999～999



原点を中心へ変更後の X-Y 軸
（MOVE 240, －240 : HSET）

PCOLOR （pen color・ペンカラー）

ペンの色指定を行う

書　式

PCOLOR { 0 / 1 / 2 / 3 }

省略形

PC.

説　明

・印字する文字や線の色を指定する命令です。

・テキストモード、グラフィックモードともに有効です。

0：黒　色
1：青　色
2：緑　色
3：赤　色

例

```
10 PCOLOR 1
```

……ペンを青色にします。

PSKIP （paper skip・ペーパースキップ）

プロッタプリンタの紙送り

書　式

PSKIP 〈行数〉

〈行数〉：－２０～２０の整数

説　明

・紙送りをする命令で、〈行数〉が正のときは紙を送り、〈行数〉が負のときは紙を戻します。

・テキストモードで有効です。

例

```
１０　ＰＳＫＩＰ　１２
```

……プリンタ用紙を12行分送ります。

```
２０　ＰＳＫＩＰ　－６
```

……プリンタ用紙を６行分戻します。

PAGE （page・ページ）

1ページあたりの行数を指定します

書　式　PAGE 〈行数〉

〈行数〉：１～72の整数

説　明　・１ページあたりの行数を指定します。

例　１０　ＰＡＧＥ　２０

……１ページあたりの行数を20行とします。プリンタでフォームフィード動作を行うと20行改行されます。

PRINT/P (print/printer・プリント/プリンタ)

プリンタ上にデータを印字する

書 式　PRINT/P 〈データ〉[{ , / ; } 〈データ〉] ……

省略形　?／P

説 明
・PRINT / P 文は、PRINT 文の実行とほぼ同じ形式でデータ出力をプリンタ上へ行います。セパレータ" , "と" ; "の働き、TAB 関数の働きも同様です。・プリンタには各種の機能があり、コントロールコードを送ることにより、これらの機能を有効に活用できます。
・コントロールコードは次の文により送ることができます。
PRINT/P　CHR$ (コントロールコード)

例
```
10 PRINT／P"ABCD"
```
……" ABCD "を印字します。
```
10 PRINT／P CHR$(&0A)
```
……改行を行います。

注)プリンタに対するコントロールのうち、↑ ↓ を含む命令を連続して実行する場合は、行を改めるかマルチステートメントにしてください。↑ ↓ の個数でコントロール動作を判定しているため、結合子 (+) で連続したデータを送ると一連のデータとみなして誤動作するおそれがあります。

[×]　PRINT/P" ↓ "+" ↓ ↓ 4D"

⇩

[○]　PRINT" ↓ " : PRINT/P" ↓ ↓ 4D"

……PRINT/P " ↓ ↓ ↓ "+" 4D" とみなして処理されます。

PLINE（pen line・ペンライン）

プロッタプリンタで直線を描く

書　式

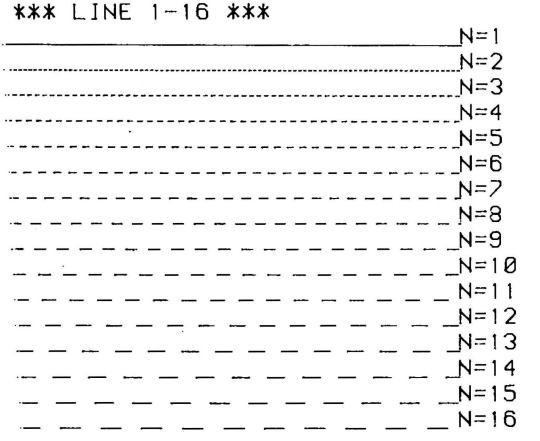
PLINE［%〈線の種類〉，］〈X座標1〉，〈Y座標1〉
……〈X座標n〉，〈Y座標n〉

〈線の種類〉：1～16の整数

〈X座標〉：－480～480の整数
〈Y座標〉：－999～999の整数

説　明

・現在のペン位置から順番に、指定された原点よりの（絶対）座標位置へ実線あるいは点線を引きます。
・グラフィックモードで有効です。
・〈線の種類〉は1から16までの整数で、1の場合実線を引き、2から16までの場合点線を引きます。%nの指定がないときは以前設定された値が保たれ、初期設定値は実線となります。〈線の種類〉＝Nの数値の指定により次のような線となります。



例

```
10 PMODE GR
20 PLINE %1,0,0,200,0,200,-200,0,-200,0,0
30 END
```

……一辺200の正方形を描きます。

RLINE (relative line・レラティブライン)

プロッタプリンタで相対的に線を描く

書　式

RLINE［%〈線の種類〉,］〈X座標1〉,〈Y座標1〉
…〈X座標n〉,〈Y座標n〉
〈線の種類〉：1～16の整数
〈X座標〉：－480～480の整数
〈Y座標〉：－999～999の整数

省略形

RL.

説　明

・現在のペン位置から相対的に、X座標，Y座標で指定された位置だけ順番に線を引きます。線の種類の値についてはPLINE命令と同じです。
・グラフィックモードで有効です。

例

```
10  PMODE  GR
20  SQ=INT(120*SQR(3))
30  RLINE  %1,240,0,-120,-SQ
        -120,SQ
40  PMODE  TN
```

……正三角形を実線で描きます。

PMOVE （pen move・ペンムーブ）

ペンを移動する

書　式

PMOVE 〈X座標〉，〈Y座標〉.

〈X座標〉：−480～480の整数

〈Y座標〉：−999～999の整数

省略形

M.

説　明

・現在の位置よりペンをアップして指定された点へ、ペンを移動します。

・グラフィックモードで有効です。

例

一辺480の十字形を作成します。

```
10  PMODE  GR
20  PLINE  0,0,480,0
30  PMOVE  240,240
40  PLINE  240,240,240,−240
50  PMODE  TN
```

※このプログラムを実行する前に必ずロール紙を紙送りしてください。

RMOVE （relative move・レラティブムーブ）

ペンを相対的に移動する

書　式

RMOVE 〈X 座標〉，〈Y 座標〉

……〈X 座標〉：－480～480の整数

〈Y 座標〉：－999～999の整数

省略形

RM.

説　明

・画面 2 のグラフィックモードにおいて、現在のペンの位置よりペンをアップして指定される座標値分（〈X 座標〉，〈Y 座標〉）相対的にペンを移動します。

・グラフィックモードで有効です。

例

```
10  PMODE  GR
20  PMOVE  240,0
30  PLINE  240,0,360,120
40  RMOVE  -120,0
50  PLINE  240,120,360,240
60  PMODE  TN
```

……斜線を 2 本引きます。

※このプログラムを実行する前に、必ずロール紙を紙送りしてください。

PHOME （pen home・ペンホーム）

ペンをホーム位置に戻す

書　式　PHOME

省略形　PH.

説　明
・現在あるペンの位置から、ペンをアップして、原点に戻します。
・グラフィックモードで有効です。

例

```
10 PMODE GR
20 PMOVE 240,-240
30 PCIRCLE 240,-240,50
40 PHOME        ……ペンをホーム位置へ戻します。
50 PMODE TN
```

HSET (home set・ホームセット)

現在のペンの位置を新しい原点とする

書 式 HSET

省略形 H.

説 明

・現在あるペンの位置を新しい原点に指定します。

・グラフィックモードで有効です。

・MODE GR 命令を実行すると原点は左端が原点となります。

いろいろ図形を書く場合などでは、原点をその図形を書く最も最適な座標におくことができます。たとえば、MOVE 命令で任意の位置にペンを移動させ、HSET 命令で原点を指定してその位置を基点として図形を書きます。

例

```
10 PMODE GR
20 PMOVE 240,-240
30 HSET ……(240,-240)を新しい原点(0,0)に
                設定します。
40 PMOVE 240,0
50 PLINE 240,0,0,-240,-240,0,
   0,240,240,0
60 PHOME
70 PMODE TN
```

GPRINT（graphic print・グラフィック プリント）

文字データをグラフィックとして印字する

書 式　GRINNT 〔[〈大きさ〉，〈傾き〉]，〕〈文字データ〉

ただし、〈大きさ〉：0～63の整数
〈傾き〉：0～3の整数

省略形　GP.

説 明
- 文字の大きさ、傾きを指定して文字データを印字します。
- グラフィックモードで有効です。
- 〈大きさ〉の値は、テキストモードでの80桁／ラインの場合は0，40桁/ラインの場合は1，26桁／ラインの場合は2です。
- 〈大きさ〉，〈傾き〉の指定がない場合は、以前設定された値が使用されます。またグラフィック命令直後の値は〈大きさ〉＝1，〈傾き〉＝0です。

例

```
10 PMODE GR
20 GPRINT"A"
```
……Aという文字をグラフィックモードで印字します。
```
30 PMOVE 240,-240
40 GPRINT[2,2],"A"
50 PHOME
60 PMODE TN
```

AXIS （axis・アクシス）

X—Y 座標の座標軸を描く

書　式

AXIS　〈軸〉，〈ピッチ〉，〈繰り返し〉

〈軸〉：0（Y 軸）または 1（X 軸）

〈ピッチ〉：−999〜999の整数

〈繰り返し〉：1〜255の整数

省略形

AX.

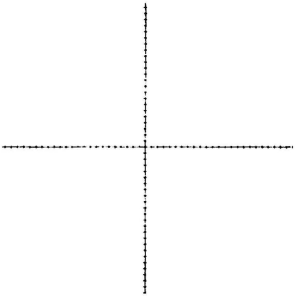
説　明

・X—Y 座標の座標軸を書きます。〈軸〉＝0のとき Œ 軸を、〈軸〉＝1のとき X 軸をそれぞれ書きます。〈ピッチ〉は座標軸に目盛を〈ピッチ〉の値ずつ、〈繰り返し〉個書きます。

・グラフィックモードで有効です。

例

```
10  PMODE  GR
20  PMOVE  240,0
30  AXIS  0,−10,48
40  PMOVE  0,−240
50  AXIS  1,10,48
60  PMODE  TN
```



……座標軸のスケール目盛を10として、X, Y それぞれ−240〜240までの座標軸を作成します。

実際にプログラムを作成する場合は、X 軸と Y 軸との交わる点を新しく頂点として、x＝−240〜240、y＝−240〜240の範囲内で使うことにより、数学と同じような感覚で使用できます。

PCIRCLE (pen circle・ペンサークル)

プロッタプリンタ上に円を描く

書式　PCIRCLE 〈X座標〉，〈Y座標〉，〈半径〉，〈開始角度〉，〈終了角度〉，〈ステップ角度〉

省略形　PC.

説明

・任意の座標を中心として、反時計まわりに円、円孤または多角形を描きます。座標〈x, y〉を中心として、スタート角度s度、エンド角度e度、ステップ角度d度で半径rの円を描きます。s＝0，e＝360，4＝0，2で円となります。

・グラフィックモードで有効です。

・多角形で描きますのでスムーズな円を描くためにはステップ角度(d)をできる限り小さくする必要があります。また、s（スタート角度）<e（エンド角度）でなければなりません。d＝0の場合は中心座標(x, y)と始点、また中心座標と終点を結ぶ直線となります。

例

```
10 PMODE GR:P=0
20 PMOVE 240,-240
30 HSET
40 FOR I=240 TO 40 STEP -60
45 PCOLOR P
50 PCIRCLE 0,0,I,0,360,2
55 P=P+1
60 NEXT I
70 PMODE TN
```

LIST／P（list／printer・リスト／プリンタ）

プログラムテキストをプリンタ上にプリントする

書　式　LIST／P［〈スタート行番号〉］［－］［〈エンド行番号〉］

省略形　L.／P

説　明
- 現在、BASICエリア内にあるプログラムのすべて、もしくは一部をプリンタ上にリストします。
- テキストモードで有効です。
- プログラムのすべてをリストする場合は、オペランドは記述しません。
- 一部分をリストさせる場合、〈スタート行番号〉と〈エンド行番号〉をマイナス記号"－"で結ぶか、どちらか一方を記述します。

例

```
ＬＩＳＴ／Ｐ
……プログラムの全部をプリンタ上にリストします。
ＬＩＳＴ／Ｐ　１０００－２９９０
……行番号1000から2990までのプログラムをプリンタ上にリストします。
ＬＩＳＴ／Ｐ　－２９９０
……行番号2990までのプログラムをプリンタ上にリストします。
ＬＩＳＴ／Ｐ　２９９０－
……行番号2990以降のプログラムをプリンタ上にリストします。
ＬＩＳＴ／Ｐ　２９９０
……行番号2990のプログラムのみをプリンタ上にリストします。
```

HCOPY （hard copy・ハードコピー）

画面の内容をプリンタに出力する

書　式

HCOPY { 1 / 2 / 3 }

説　明

・指定画面のハードコピーを行います。
・テキストモードで有効です。
・1または省略すると、画面1をハードコピーします。
・MZ-1P09使用時は画面1のハードコピーのみ可能です。ただしMZ-1P09のアスキーコードにないキャラクタはピリオド"．"で印字します。
・2のときは、画面2をハードコピーします。
・3のときは、画面1と画面2をハードコピーします。
・3はプロッタプリンタには使用できません。MZ-1P08にのみ有効な命令です。（MZ-80P4K、MZ-80KP5も使用可能です。）

PLOT（plot・プロット）

ディスプレイの代用としてカラープロッタプリンタを使用できるようにする

書　式

PLOT { ON / OFF }

省略形

PL.

説　明

・ON ならば、ディスプレイの代用として、カラープロッタプリンタを使用できるようにします。これにより、ディスプレイを使用できないところでも、手軽に MZ-1500 をご使用いただけます。

・テキストモードで有効です。

・この命令は、オプションのカラープロッタプリンタ MZ-1P09 が必要です。

・カラープロッタプリンタのキャラクタジェネレータにないキャラクタはピリオドで印字します。[INST]、[DEL] キーのキー操作および [←] キー使って一行前へカーソルを移動させることはできません。また、[CTRL] + [G] でペンの色を変更します。

・OFF ならば、PLOT ON の機能を解除します。

・PLOT ON 時は CONSOLE 命令は使用できません。

例

```
PLOT　ON
```

……プロッタプリンタをディスプレイの代用として使用します。

DIR／P（directory／printer・ディレクトリ／プリンタ）

ディレクトリの内容をプリンタに出力します

書　式　DIR／P［〈装置名〉］

説　明
・装置に登録されているファイルに関する情報、ディレクトリ（台帳）の内容をプリンタに出力します。
・〈装置名〉を省略すると、DEFAULT文で指定された装置と見なされます。
・この命令で指定された装置はDEFAULT命令で指定されたものと同じ意味をもちます。

例　ＤＩＲ／Ｐ　ＱＤ
……クイックディスクに登録されているファイルのディレクトリをプリンタに出力します。

6 機械語命令

PEEK（peek・ピーク）

メモリ上の任意のアドレスのデータを与える

書式　PEEK［@］（〈アドレス〉）

〈アドレス〉：数値データ（0～65535）、もしくは4ケタの16進数（$0000～$FFFF。）

説明

・〈アドレス〉の値の示すメモリ上のアドレスの内容を変数に代入します。メモリ上のデータは8ビットデータですので0から255の範囲が与えられます。

・53248（$D000）番地以降でユーザRAMエリアの任意のアドレスの内容を変数に代入する場合は、PEEK@命令を使います。

例　メモリ上のアドレス40960（$A000）番地から40975（$A00F）番地までのデータを表示します。

```
10 FOR AD=40960 TO 40975
20 ? PEEK(AD)
30 NEXT AD
```

POKE（poke・ポーク）

メモリ上の任意のアドレスにデータを書き込む

書式

POKE［@］　〈アドレス〉，〈データ〉

〈アドレス〉：数値データ（0～65535）、もしくは4ケタの16進数（$0000～$FFFF）

〈データ〉　：数値データ（0～255）、もしくは2ケタの16進数（$00～$FF）

説明

・〈アドレス〉に相当するアドレスに、〈データ〉に相当するデータをセットします。

・〈アドレス〉で指定するメモリ位置に直接1バイトのデータを書き込む文であり、アクセスするアドレスは任意です。

・POKE文はLIMIT命令と関係なく、メモリ空間上の任意の位置に対して実行可能ですからBASICやMONITOR自身を壊す可能性があるので、その使用には十分注意が必要です。

・53248（$D000）番地以降で、ユーザRAMエリアの任意のアドレスに指定したデータを書き込む場合は、POKE@命令を使います。$D000以降のPOKEは、ユーザRAM側を指定します。

例

```
POKE　$D000，$5F……16進数を使った場合
POKE　53248，95……10進数を使った場合
……2つは同じです。
```

INP@(input・インプット)

I/O ポート上にあるデータを変数に入力する

書式　INP@〈ポート番号〉，〈変数〉

説明
- 〈ポート番号〉の入力ポートより8ビットデータを入力し、それを10進数に変換した数値（0～255）を、〈変数〉に代入します。
- 〈ポート番号〉として、0～239（16進数の$00～$FF）までは、任意に決めることができます。128～255(16進数の$80～$FF)までは、外部機器用ポートとして専用に使用されています。

例

```
10  FOR  1=0  TO  20
20  C=1+32
30  GOSUB  "SUB"
40  NEXT  1
50  END
60  LABEL  "SUB"
70  INP@  $FE,A
80  IF  NOT(A  AND  $0D)=0  THEN  70
90  OUT@  $FF,C
100  OUT@  $FE,$80
110  INP@  $FE,A
120  IF  NOT(A  AND  $0D)=1  THEN
     110
130  OUT@  $FE,0
140  RETURN
```

OUT@(OUT・アウト)

I／O ポート上にデータを出力する

書　式　OUT@〈ポート番号〉，〈数値データ〉

説　明

・〈ポート番号〉の出力ポートに、〈数値データ〉の値（0～255）を2進数に変換して出力します。

・〈ポート番号〉として、0～127（16進数の$00～$7F）までは、任意に決めることができます。128～255（16進数の$80～$FF）までは外部機器用ポートとして、専用に使用されています。

・周辺機器などの制御はI／Oポートにデータを出力して行っています。ですから、このOUT命令で〈ポート番号〉などを誤まると周辺機器などの異常動作の原因となる可能性もありますので、使用にあたっては十分注意してください。

例

```
10  FOR  I=0  TO  20
20  C=I+32
30  GOSUB  "SUB"
40  NEXT  I
50  END
60  LABEL  "SUB"
70  INP@  $FE,A
80  IF  NOT(A  AND  $0D)=0  THEN  70
90  OUT@  $FF,C
100  OUT@  $FF,$80
110  INP@  $FE,A
120  IF  NOT(A  AND  $0D)=1  THEN
     110
130  OUT@  $FE,0
140  RETURN
```

USR (user・ユーザ)

機械語プログラムを BASIC で呼び出して実行する

書 式

USR (〈アドレス〉[, 〈入力文字変数〉][, 〈出力文字変数〉])

〈アドレス〉：数値データ、もしくは 4 桁の16進数

省略形

U.

説 明

・BASIC プログラム実行中に機械語プログラムを呼び出して実行します。これは、機械語サブルーチンへの分岐命令、CALL〈アドレス〉と同じです。したがって、機械語プログラム中で、リターン命令(RET, RET cc)があると、USR 関数を実行した次のステートメントコントロールが移ります。

・機械語プログラムを呼び出した時点で、〈入力文字変数〉の次の値がレジスタにセットされます。

DE レジスタ：〈入力文字変数〉のメモリエリアの先頭アドレス

B レジスタ：〈入力文字変数〉の長さ

IX レジスタ：エラー処理ルーチンが宣言されている場合のアドレス

・機械語プログラムから戻る時点で、次のレジスタで示されるデータの値が〈出力文字変数〉となります。

DE レジスタ：〈出力文字変数〉のメモリエリアの先頭アドレス

B レジスタ：〈出力文字変数〉の長さ

・機械語プログラム中でエラー処理が必要な場合は、次の処理を行います。

① BASIC プログラム中で、ON ERROR GOTO 文によりエラー処理ルーチンを設定する。

② A レジスタにエラーコードを代入し、IX レジスタの示すアドレスにジャンプさせる。

LIMIT（limit・リミット）

BASICで使用するメモリエリアを制限する

書式　LIMIT 〈アドレス〉

〈アドレス〉：数値データ（0～6579）、もしくは4桁の16進数（$0000～$FFFF）

省略形　LIM.

説明

・BASICで使用するメモリエリアを、〈アドレス〉で指定するアドレスまでに制限し、その次のアドレスから$FFFF（65279）番地までをユーザエリアとします。

BASICで機械語プログラムとのリンク（連係）を行う場合、あるいは特殊なデータをメモリ上に構成しようとする場合は、それに必要なユーザエリアをこのLIMIT命令で制限して、BASICで使用するエリアと区別しておかなければなりません。

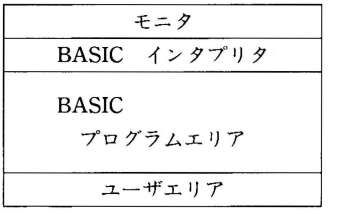
・$FF00～$FFFFまではモニタのワークとして使われますので、ユーザエリアとしては使えません。またLIMIT命令はプログラムの先頭文で使われなければなりません。

・LIMIT命令により、制限されたBASICの使用エリアを解除するにはLIMITMAX文を実行します。

例

```
LIMIT $AFFF
```

BASICプログラムエリアを$AFFF番地までに制限し、それ以降をユーザエリアとします。




```
LIMIT MAX
```

……制限されたBASICの使用エリアを解除します。

7 エラー命令

ON ERROR GOTO (on error goto・オンエラーゴーツー)

エラーが発生した時、プログラムの実行をエラー処理ルーチンに移す

書 式

ON ERROR GOTO { 〈行番号〉 / 〈ラベル〉 }

省略形

ON ERR. G.

説 明

・エラーが発生した時、プログラム実行を移す行番号を宣言します。

・ON ERROR GOTO 文によりエラー処理ルーチンを宣言することによって、エラー発生時に、停止させずに、プログラム内でエラー処理を行うことが可能になります。

・ON ERROR GOTO 文が実行されていると、プログラム実行中にどのようなエラーが発生した場合でもプログラム実行は指定した〈行番号〉または〈ラベル〉で始まるルーチンに移りますが、このルーチンでシステム変数 ERN や ERL によってエラー番号(ERN)とエラー発生行番号(ERL)の判別ができ、それぞれ適当な処理を行うことができます。

・エラー発生箇所にプログラム実行を戻すために、RESUME 文を使用することができます。

・新たに ON ERROR GOTO 文を実行すると、前の ON ERROR GOTO 文は無効となります。

・ON ERROR GOTO 文を設定後、CLR 命令を実行すると、ON ERROR GOTO 文の設定は無効となります。

例

```
10 ON ERROR GOTO 1000
20 INPUT "X=";X
30 Y=SQR(X)
40 PRINT "ANS.=";Y
50 END
1000 PRINT "テイギサレテ イマセン"
1010 RESUME 20
```

RESUME （resume・レジューム）

エラー処理ルーチンの実行後、エラーが発生したルーチンに実行を戻す

書　式

RESUME 〈行番号〉
RESUME　NEXT
RESUME　0
RESUME

省略形

RESU.

説　明

・エラー処理後、プログラム実行をエラーが発生したルーチンへ戻します。
・エラー発生時にはエラー発生位置が記憶されているので、エラー処理を終了した後、その文あるいは、その次の文へプログラム実行を戻したり、あるいは、他の任意の行へ戻すことができます。
・次の4通りの復帰文が記述できます。

RESUME……エラーが発生した文へ復帰します。
RESUME……NEXT…エラーが発生した文の次の文へプログラム実行を戻します。
RESUME 〈行番号〉…〈行番号〉で指定する行へプログラム実行を戻します。
RESUME　0……プログラムの先頭へプログラム実行を戻します。

・エラーが発生していないのにRESUME文にきた場合、ここでErrorが発生します。
・RESUME不能の場合、Errorが発生します。
・エラー処理ルーチンからメインプログラムへの復帰には、かならずRESUME文を使ってください。

第5章　付　　録

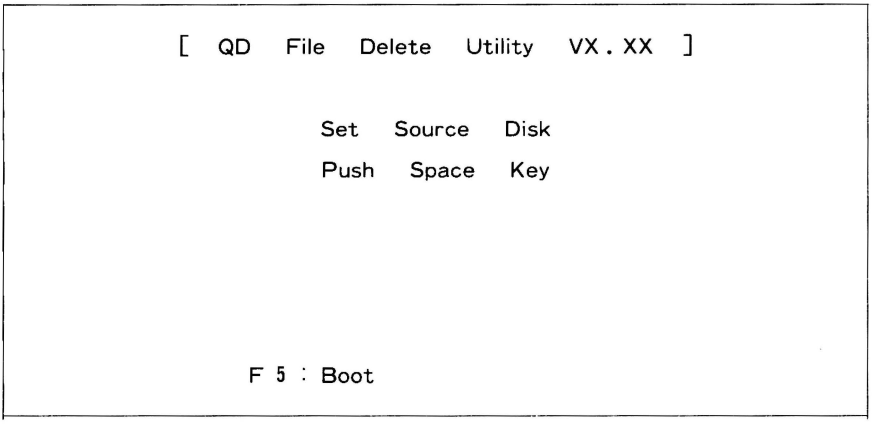
A. DELETE ユーティリティ

・このソフトはクイックディスクを有効に使用するために不必要なファイルを消去するものです。クイックディスク上に登録されているファイルが不必要になった場合、必要なファイルのみをコンピュータ本体のメモリ上に読み込んで、1時退避させておき、次にクイックディスクをイニシャライズします。その後、メモリ上に読み込んだ内容をクイックディスクに書き込むことにより、不必要なファイルを消去するものです。

・このソフトを動かすには次の命令を実行してください。

RUN"DELETE" CR

上記命令を実行することにより次のような初期画面が表れます。



［説明］

ここで消去したいファイルがあるクイックディスクをセットしてスペースキーを押してください。スペースキーを押すと、セットされたクイックディスクのディレクトリが表示され、画面は次のようになります。（次に１つの例を示します。）

```
          [  QD   File   Delete   Utility   VX.XX  ]

No.  Atr File   Name              Size      Delete
1    OBJ  ASEM                  44651     No
2    BTX  TEST                    299     No
3    OBJ  DATA                   7840     No
4    BTX  SAMPLE                10342     No
5    OBJ  DATA2                  1064     No




F1:RUN  F4:Again    All  Size  64196
        F5:Boot     Max  Size  57088
```

ディレクトリが表示されたらカーソル（▩）が表示されますので、消去したいファイルのところに[↑]、[↓]キーを使ってカーソルを移動させ" Y "キーを押してください。" Y "キーを押すと今まで" No "と表示されていたのが" Yes "にかわります。このようにして消去したいファイルを" Yes "に変更してください。その際、画面右下にある「All␣Size」のバイト数表示が減少します。また、もしまちがえたりして消去したくないファイルを" Yes "にしてしまったら、ファイルのところにカーソルを移動させ今度は" N "を入力してください。すると" Yes "が" No "に変更されます。

次に前の画面で出てきた" DATA "というファイルと" DATA 2 "というファイルを消去したいとすると、[↑]、[↓]キーを使ってカーソルを消去したいファイルのところまで移動させて、" Y "キーを押して" No "を" Yes "に変更すると画面は次のようになります。

```
          [  QD   File   Delete   Utility   VX.XX  ]

No.  Atr File   Name              Size      Delete
 1   OBJ  ASEM                  44651     No
 2   BTX  TEST                    299     No
 3   OBJ  DATA                   7840     Yes
 4   BTX  SAMPLE                10342     No
 5   OBJ  DATA2                  1064     Yes




F1:Run  F4:Again    All Size  55292
        F5:Boot     Max Size  57088
```

" No " を " Yes " にしただけではファイルを消したことにはなりません。次に F1 キーを押すと " Yes " にしたファイル以外（" No " と表示しているファイル）を本体メモリ上に読み込み画面右下に次のような表示がでます。

```
File is loading !
```

（この時、All Size＞Max Size の条件が成立すると、Memory Capacity error となります）。読み込みが終了すると画面右下に次のような表示がでます。

```
Loading OK !
Set Destination Disk
```

ここで、書き込むクイックディスクをセットしてください。ただし書き込むクイックディスクとして同じディスクを使用する場合は、入れかえる必要はありません。F1 キーを押すと、次の表示がでてコンピュータ本体のメモリ上に読み込んだファイルの内容がクイックディスクに書き込まれます。

```
File is Saving !
```

書き込みが終了すると、画面右下に次の表示がでます。

```
End of Job
```

ここで画面左下に表示されるメッセージについて説明します。

F 1 ： RUN ……指定した作業を実行させます。

F 2 ： Cont ……Error（エラー）などで実行が中断された場合、エラーが起こる直前の状態に戻ります。

F 4 ： Again……初期画面（初期状態）に戻ります。

F 5 ： Boot ……ROM モニタの初期画面に戻るか、または最初に見つけた OBJ ファイルを読み込みます。

B. QDCOPY ユーティリティ

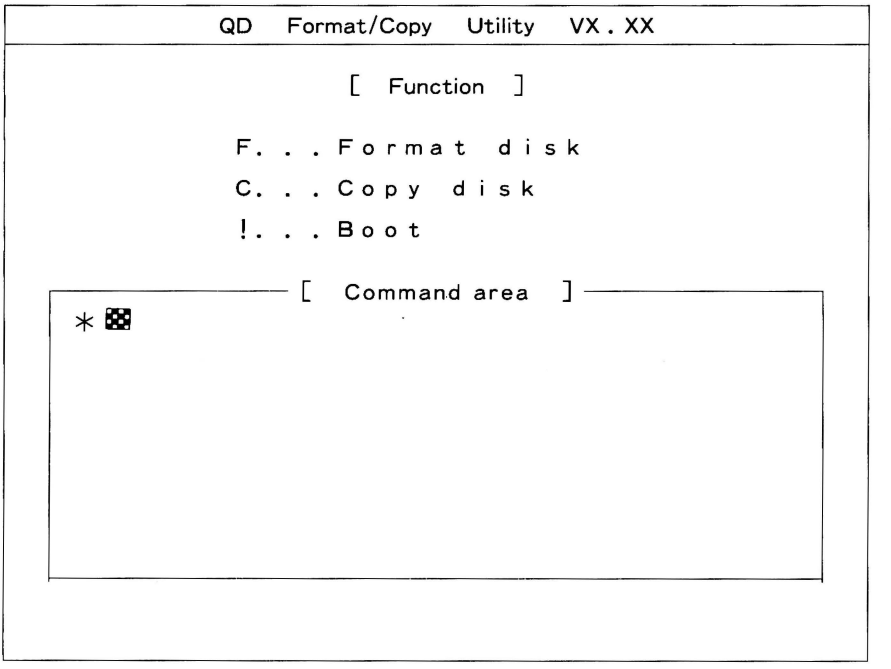
このソフトは、クイックディスクの内容をそっくりそのまま他のクイックディスクにコピーするものです。使用例として同梱されている BASIC を、このソフトを使ってコピーしてみましょう。

これは何らかの事故により、クイックディスクが使用不能となってもかわりのクイックディスクに同じものをコピーしておけば助かるからです。ふだん使用するときはコピーした方のディスクを使用し、同梱されていたディスクは保管しておくようにしましょう。

このソフトを動かすには次の命令を実行してください。

```
RUN"QDCOPY"CR
```

上記命令を実行することにより、次のような初期画面が表れます。



①まずはじめに、コピーディスクをフォーマット設定しなければなりません。
[F] キーを押すと「Command　area　」には次の表示が行われます。

```
[ Command  area ]
＊F
  Set  format  disk,  ↓  any  key ■
```

フォーマット設定を行うディスクをディスクドライブにセットして、キー（いずれかのキー）を押してください。すると、コピーディスクのフォーマット設定が行われ、終了すると" OK！" が表示され、画面表示は次のようになります。

```
[ Command  area ]
＊F
  Set  format  disk,  ↓  any  key
  OK!
＊■
```

②コピーする元のディスクの内容をコンピュータ本体に読み込ませなければなりません。[C] キーを押すと「Command　area」には次の表示が行われます。（前記述内容の表示は省略します。）

```
[ Command  area ]
＊C
  Insert  source  disk,  ↓  any  key
```

コンピュータ本体に読み込ませる、ディスクをディスクドライブにセットしてキー(いずれかのキー) を押してください。するとディスクの内容がコンピュータ本体に読み込まれます。読み込みが終了すると画面は次のようになります。

```
[ Command area ]
＊C
  Insert source disk, ↓ any key
  Insert destination, ↓ any key
```

今度はコピーディスクをディスクドライブへセットしてキー (いずれかのキー) を押してください。するとコンピュータ本体に記憶されている内容がクイックディスクに書き込まれます。書き込みが終了すると" OK！" と表示されます。(ただしコピーするファイル容量が大きかったり、ファイル数が多かったりすると一度ではコピーできない場合がありますので、②からの手順を" OK！" の表示が出るまでくり返して下さい。)

コピーが終ると画面は次のようになります。

```
[ Command area ]
＊C
  Insert source disk, ↓ any key
  Insert destinetion, ↓ any key
  OK!
＊■
```

クイックディスクのコピーが終了し次のコマンドまち状態になったことを示しています。次に [!] キー ([SHIFT] + [1]) を入力すると、ROM モニタの初期画面に戻るか、最初に見つけたOBJ ファイルを読み込みます。

C. TRANSユーティリティ

・ファイルコンバータは、BASICのディスク上に、" TRANS "というファイルネームで登録されています。これを実行させるには、

RUN "TRANS" CR

とキーを押してください。

最初の画面に出てくる指定項目を上から順に指定しますが、指定の内容はメニューエリアに表示される中から適切なものを選んでその番号をキー入力してください。以後は画面表示にしたがって操作してください。

D. モニタコマンド

(1) モニタコマンドの機能と使い方

機械語で作成されたプログラムは、BASICで作られたプログラムに比較し、むずかしく、わかりにくいものですが、反面、処理速度が速い、必要とするメモリが少なくてすむ、ハードウェアに即したプログラムが作成できるなど、多くの長所があります。

しかし、BASICでは機械語プログラムを作成することはできませんので、機械語プログラムを作成するために、モニタコマンドが用意されています。

ここでは、モニタコマンドについて、それぞれの機能と使い方の解説を行います。ただし、使用にあたっては次の注意が必要です。

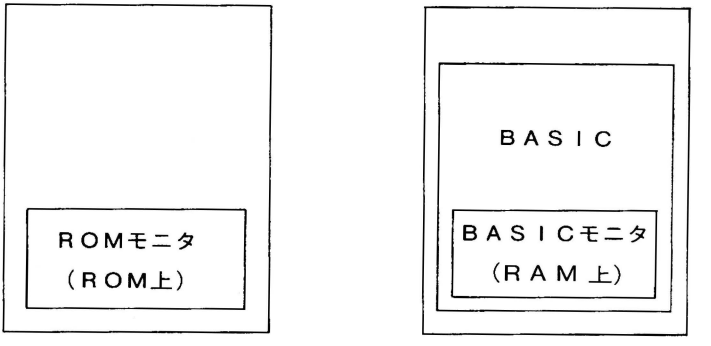
・コマンドあるいはデータのキー入力は、所定の場所に正しく行い、CR キーを押すことによって実行されます。スペースなどの不必要なデータをまじえたキー入力をすると、モニタコマンドはそのデータを受け付けません。
・コマンドを修飾するデータ入力あるいはデータ表示はすべて16進表示で行われます。したがって、1バイトデータは16進2ケタ、1つのアドレスデータは16進4ケタで扱われます。たとえば表示データが"15"であったとすれば16進数の$15(10進数の21)を表しており、キー入力を行う場合、たとえば、10進数の10をデータとして入力する場合は、所定の位置にきちんとケタ数を合わせて、"0A"と入力しなければなりません。(上位ケタの0を省略することはできません。)
・ファイル名の指定などで、モニタが受け付ける以上の文字数を入力した場合、その数を越す文字列は無効となります。
・各コマンドのアクセス範囲には、プロテクションが一切ありません。したがって、モニタプログラム自身の変更などが自由に行えますが、逆に、システムコントロールを行うプログラムを壊してしまい、CPUの暴走を引き起こす可能性もありますので、細心の注意をはらう必要があります。

(2) ROM モニタと BASIC モニタ

モニタとは、OS（オペレーティング・システム）の一種であり、BASIC などの言語の手助け：ファイルの管理・機械語プログラムの作成などの機能をもっています。

本機には、あらかじめ ROM に書き込まれている ROM モニタと、BASIC 本体の一部として RAM 上に読み出される BASIC モニタとがあります。

ROM モニタについては次ページから説明します。BASIC モニタについては 250 ページで説明します。



状　態	モニタ
・[M]を押しながらRESETスイッチを押した場合	ROMモニタ
・[M]を押しながら、電源をONにした場合	ROMモニタ
・起動時にモニタを指定した場合	ROMモニタ
・BASIC上でBYE命令を実行した場合	BASICモニタ

注1）254ページに載せているモニタサブルーチンのエントリアドレスは、ROM モニタのものであり、BASIC モニタのものではありません。

注2）BASIC モニタに戻った場合は、BASIC で使用されていた変数や配列の値がそのまま残っています。ROM モニタに戻った場合は保証されません。

(3) ROM モニタ

L（QL） コマンド

書式 L

QL

説明 機械語プログラムを読み出します。L ならばカセットテープから、QL ならばクイックディスクから読み出します。L の場合は " PLAY " と表示されますのでカセットレコーダの PLAY ボタンを押してください。QL の場合は、「File name？」とたずねてきますので、読み出すべきファイルの名前を入力します。もし入力しなければ、クイックディスク上の先頭のファイルを読み出します。

例 ＊ＱＬ CR

Ｆｉｌｅｎａｍｅ？ ＭＦＩＬＥ CR

S（QS） コマンド

書式 S

QS

説明 指定するメモリブロックの内容を、ファイル名を付けて記録します。S ならばカセットテープ上に、QS ならばクイックディスク上に記録します。

例 アドレス $6000～$60A3の機械語プログラムに、" MFILE " というファイル名を付けて、カセットテープ上に記録します。また、このプログラムをロードした場合は自動的に、$6050番地から実行がはじまります。

＊Ｓ CR

Ｆｉｌｅｎａｍｅ？ ＭＦＩＬＥ CR

Ｔｏｐ ａｄｒｓ ？ ６０００ CR

Ｅｎｄ ａｄｒｓ ？ ６０Ａ３ CR

Ｅｘｃ ａｄｒｓ ？ ６０５０ CR

↓ ＲＥＣＯＲＤ．ＰＬＡＹ

ここで RECORD ボタンを押します。（クイックディスクの場合には必要ありません。）

M　コマンド

書式　M〈先頭アドレス〉

説明　指定するアドレスから順に1バイトずつデータの書き替えを行います。

例　$C000～$C002の内容を全部$FFとし、$C010～$C013の内容をすべて$88とします。

```
＊MC000 [CR]
C000  00  FF
C001  00  FF
C002  00  FF
C003  00  [SHIFT] + [BREAK]
＊MC010
C010  00  88
C011  00  88
C012  00  88
C013  00  88
C014  00  [SHIFT] + [BREAK]
＊
```

Mコマンドをキャンセルしてモニタコマンドへ戻るには、[SHIFT]＋[BREAK]キーを入力します。

J　コマンド

書式　J〈アドレス〉

説明　指定するアドレスへコントロールを移します。正確にいうと、指定するアドレスをCPUのプログラムカウンタに与えることになります。

例　＊J1200……$1200番地へジャンプします。

P　コマンド

説明　プロッタプリンタに対して行う命令で、次のような命令があります。

＊PABC

ABCという文字を印字します。

＊P&T

テストパターンを印字します。

＊P&S

文字桁数を、80文字1行に変更します。

＊P&L

文字桁数を、40文字1行に変更します。

＊P&G

グラフィックモードに変更します。

＊P&C

ペンを、次の色に変更します。

D　コマンド

書式　D〈先頭アドレス〉〈終了アドレス〉

説明　〈先頭アドレス〉から〈終了アドレス〉までのメモリの内容を表示します。

例　＊D C 0 0 0 C 7 0 0 CR

……アドレス$C000-$C700までのメモリの内容を表示します。

V　コマンド（V：Verify）

書式　V

説明　カセットテープにセーブされたデータが、もとのメモリ上のデータと同じであるかどうか調べます。

例　Sコマンドの例でセーブしたプログラムを比較して調べます。

```
＊VMFILE CR
  PLAY
```

QC コマンド

書式　QC

説明　クイックディスク上のファイルを、コピーします。
まず、元になるクイックディスクをセットしてQC命令を実行するとコピーするファイルの名前を尋ねてきますので、ファイル名を入力します。
そのファイルを読み終ったら、「Set destination disk」と表示されますので、読み込んだファイルを書き込むクイックディスクをセットしてください。そして"Y"キーを入力すると読み込んだファイルの内容がセットしたディスクに書き込まれ、"N"を入力するとモニタのコマンドまち状態に戻ります。

例

```
QC CR
Filename? M F I L E CR
Set destination disk
OK! (Y/N)
```

QF　コマンド

書式　QF

説明　セットされているクイックディスクの初期化（フォーマット設定）を行います。
初期化を行うとクイックディスク上の全てのファイルが消去されます。

例　＊QF [CR]
QD：Formatting
OK？（Y／N）

QX　コマンド

書式　QX

説明　カセットテープ上のファイルをクイックディスク上にコピーします。
QX命令を実行すると、" PLAY " というメッセージが表示されますのでカセットレコーダーのPLAYボタンを押してください。
ファイルを読み出し、そのファイル名を画面に表示します。
そのファイルを読み終ったら、「Set destination disk」と表示されますので、クイックディスクをセットし、書き込む場合のファイル名を入力します。この場合、ファイル名は省略できません。したがって、そのファイル名が、読み出した場合のファイル名と同じであっても入力する必要があります。

例　QX [CR]
PLAY
Loading "MFILE"
Set destination disk
Filename？ [M] [F] [I] [L] [E] [CR] ←必ずファイル名を入力してください。
OK？（Y／N）
OK！←コピーが終了したことを表します。

QD　コマンド

書式　QD

説明　クイックディスクに登録されているファイルのディレクトリを表示します。
クイックディスクに登録されているファイルの名前や種類を調べる場合に使用されます。

例　＊QC CR
```
OBJ　BASIC
BTX　PROGRAM
BSD　DATA
OBJ　MFILE
```

#　コマンド

書式　#

説明　BASICを読み込ませて、RESETスイッチを押し、モニタの状態にしてから
＊# CR
と#コマンドを入力すると、BASICに戻ります。

B　コマンド

書式　B

説明　＊B CR
とBコマンドを入力すると、キーを押すごとにベル音を鳴らします。ベル音を止めたいときは、もう一度Bコマンドを入力します。

■ 機械語プログラムの作成

画面いっぱい、本機にあるキャラクタの全部を使って埋めつくすことを考えてみます。

これは要するにキャラクタジェネータ（CG）内のすべての文字を表示させることであり、ビデオRAMに、0から255の8ビットデータの全てを順次セットすればよいことになります。

機械語プログラムを機械語エリアに作成し、そのサブルーチンをBASICから呼び出して、以上の動作を行わせるようにしてみます。その例は次のようになります。

```
10 LIMIT $BFFF ····· BASICエリアを$BFFFに限定する。
20 GOSUB 50
30 USR($C000) ····· 機械語プログラムを呼び出す。
40 END
50 FOR I =49152 TO 49181 ·····
60 READ M                    ····· 機械語プログラムのデータを読み、
70 POKE I,M                        機械語エリアに書き込む。
80 NEXT I
90 RETURN                    ·····
100 DATA 197:REM        PUSH BC ····· 機械語データのはじまり。
110 DATA 213:REM        PUSH DE
120 DATA 229:REM        PUSH HL
130 DATA 22,0:REM       LD D,0
140 DATA 33,0,208:REM   LD HL,D000H
150 DATA 1,232,3:REM    LD BC,1000
160 DATA 243:REM        DI
170 DATA 211,227:REM    OUT (E3H),A····· メモリブロックのビデオ
180 DATA 114:REM    STO:LD (HL),D·····     RAM側に切り替える。
190 DATA 35:REM         INC HL
200 DATA 20:REM         INC D
210 DATA 11:REM         DEC BC         ··· ビデオRAMにディスプ
220 DATA 120:REM        LD A,B              レイコードをセットする。
230 DATA 177:REM        OR C
240 DATA 194,14,192:REM JP NZ,STO
250 DATA 211,225:REM    OUT (E1H),A····· メモリブロックのダイナ
260 DATA 251:REM        EI                  ミックRAM側に切り
270 DATA 225:REM        POP HL              替える。
280 DATA 209:REM        POP DE
290 DATA 193:REM        POP BC
300 DATA 201:REM        RET ····· BASICプログラムに戻る。
```

または、モニタで機械語エリアを作成しておき、テープへSAVEされた機械語プログラムファイル（DISPLAYCODE）を使用するときは、次のようにします。

```
110 LIMIT $BFFF
110 LOAD"DISPLAYCODE"
120 USR($C000)
```

(4) BASIC モニタ機能

BASIC モニタ部を設けています。スタックワークエリアとしては $FF00～FFFF を使用しています。

このモニタは、BASIC 同様スクリーンエディタを内蔵していて、次に示すエディト書式によりメインメモリ64KB のどのアドレスでも書きかえ可能です。

エディト書式

：アドレス＝データ　データ　データ

：(コロン)……エディト可能な行であることをしめすマーク（文番号とモニタのアドレスを区別するため）

アドレス　……16進数 4 桁で指定し、先頭のデータがはいるメインメモリのアドレス（$0000～$FFFF）

＝(符号)　……アドレス部とデータ部を区別するためのセパレータ

データ　……16進数 2 桁、あるいは；（セミコロン）＋文字で指定し、8 ビットのデータあるいは、指定文字のアスキーコードを指定のメモリアドレスに書き込みます。データ間は原則として、スペースで区切ります。

プリンタスイッチ（P コマンド）

＊P

D および F コマンドの出力を、プリンタにしたり画面にしたりします。モニタに入ったときは、画面モードとなっており、このコマンドを実行するたびにモードが反転します。プリンタがつながっていない場合は、しばらくしてから ERR？と出てコマンド待ちになりますから、プリンタをチェックするか、P コマンドを実行して画面モードに戻してください。

ダンプ（D コマンド）

＊D 〈先頭アドレス〉〈最終アドレス〉

メモリの内容を表示します。最終アドレスを省略すると先頭アドレスより 128 バイトを表示します。先頭アドレスも省略すると次のアドレスから 128 バイトを表示します。

次のフォーマットでダンプされます。

：HHHH＝HH HH HH HH HH HH HH HH／ABCDE G

↑

先頭アドレス　　8 バイト16進データ　　8 バイト文字データ

メモリ内容を変更する場合は、カーソルを変更したいデータ上に移動させて、訂正を行い、CR を入力することによってできます。

注）最後に出力される 8 バイト文字は、データのアスキーコードで、コントロールコードは．（ピリオド）で表示されます。BREAK キーのみで表示が止まり SHIFT ＋ BREAK でコマンド待ちに戻ります。

メモリセット（M コマンド）

＊M 〈先頭アドレス〉

メモリの内容を書き換えます。先頭アドレスを省略すると、現在のポインタから、かきかえます。このモードからぬけるには SHIFT ＋ BREAK を押してください。

アドレスとデータが表示されて、カーソルは、データの上にきますので、エディットの書式でデータを指定して CR を押してください。指定のデータ数、アドレスが加算されて次の行へすすみます。

ファインド（F コマンド）

＊F〈先頭アドレス〉〈最終アドレス〉〈データ〉〈データ〉…

先頭アドレスより最終アドレスまで、データで指定された個数の連続したデータをさがし、見つかったらそのアドレスとデータをダンプの様式で出力します。

SHIFT ＋ BREAK で止まります。

ゴーサブ（G コマンド）

＊G〈コールドアドレス〉

指定したコールアドレスをサブルーチンコールします。スタックポインタは$FFFFにあります。

トランスファ（T コマンド）

＊T〈先頭アドレス〉〈最終アドレス〉〈転送先頭アドレス〉

指定したアドレス間のデータを転送先頭アドレスより転送します。

セーブ（S コマンド）

＊S〈先頭アドレス〉〈最終アドレス〉〈実行アドレス〉："装置名：ファイル名"

指定したアドレス間のデータをカセットテープに記録します。実行アドレスとはIPLからLOADしたときの実行先頭アドレスです。ファイル名は、：(コロン)の後に指定してください。

ロード（L コマンド）

＊L 〈ロード先頭アドレス〉：" 装置名：ファイル名 "

指定のファイルをカセットよりLOADします。先頭アドレスを指定すると、そのアドレスよりLOADし、指定しないと、SAVEした情報のとおりLOADします。ファイル名は指定しないと最初に見つけたファイルを、LOADします。途中でBREAKしたりチェックサムエラーが出たときは、ERR？と表示して、コマンド待ちに戻り、エラーが出なかったときはそのままコマンド待ちに戻ります。

ベリファイ（V コマンド）

＊V 〈ファイルネーム〉

指定のファイルをカセットより読み、メインメモリの内容と比較します。正しくSAVEされたかどうかを調べるときに使い、正しくない場合ERR？と表示します。

リターン（R コマンド）

＊R

モニタをコールしたシステムへリターンします。SP（スタックポインタ）およびHLレジスタは保存していますのでMONの次の命令を実行します。

SPがFF00 ～ FFFFにあるようなシステムからコールされた場合や、スタックにリターンアドレスが、入ってない場合は、この命令ではリターンできません。Gコマンドより、そのシステムのホットスタートを、CALLしてください。

E．モニタサブルーチン

モニタ1(9Z-502M)のサブルーチンには次のようなものがあります。サブルーチン名は、サブルーチンの機能をシンボリックに表現したものです。

レジスタ保存は、そのサブルーチンをコールし、リターンしたときに、内容が保存されるレジスタを示しています。すなわち、それ以外のレジスタの内容は、リターン時に一般に変化することになるので、注意が必要です。

サブルーチン名およびエントリーポイント(16進番地)	サブルーチンの機能	レジスタ保存
CALL LETNL (0006)	行を替えて、次の行の先頭にカーソルをセットする。	AF以外は保存
CALL PRNTS (000C)	テレビ画面のカーソル位置にスペースを1個だけ表示する。	AF以外は保存
CALL PRNT (0012)	ACCにあるデータをアスキーコードと見て、テレビ画面のカーソル位置に表示する。アスキーコードと、キャラクタの関係は、258ページの表を参照。ただし、0Dコードのときは、キャリッジリターンが実行され、11～16コードのときは、それぞれカーソルコントロールが実行される。	AF以外は保存
CALL MSG (0015)	テレビ画面のカーソル位置から、メッセージを表示する。 メッセージの先頭アドレスは、あらかじめDEレジスタに指定しておくこと。メッセージは、アスキーコードで構成し、エンドマークは、0Dコードであること。ただし、キャリッジリターンは実行されない。 カーソルコントロール(11～16コード)は実行される。	全レジスタ保存
CALL BELL (003E)	高音のラ（約880Hz）を瞬間だけ鳴らす。	AF以外は保存
CALL MELDY (0030)	音楽データを演奏する。音楽データの先頭アドレスは、あらかじめ、DEレジスタに指定しておくこと。音楽データは「BASICのMUSIC命令」で示した文字列と同じく、音程、音長の順にアスキーコードで表現し、エンドマークは、0DまたはCB（キャラクタは■）でなければならない。 ただし、リターン時にCフラグが0なら演奏完了、Cフラグが1なら途中でBREAKキーが押されたことを示す。	AF以外は保存
CALL XTEMP (0041)	演奏テンポを設定する。テンポのデータ（01から07）をACCにセットしてコールする。 ACC←01　最も遅い ACC←04　中位の速度 ACC←07　最も速い	全レジスタ保存

サブルーチン名およびエントリーポイント（16進番地）	サブルーチンの機能	レジスタ保存
	注意が必要なのは、ACCに入れるコードは、1から7までのバイナリコードであり、"1"から"7"に相当するアスキーコード（31から37）ではない点です。	
CALL MSTA （0044）	指定された分周比の音を連続して鳴らす。 分周比nn'（2バイトのデータ）は、11A1番地にn'、11A2番地にnをストアしてコールする。 分周比と発生周波数の関係は895kHz/nn。	BC、DEだけ保存
CALL MSTP （0047）	音の発生を止める。	AF以外は保存
CALL TIMST （0033）	内蔵された時計をセットする。（時計はこのコールで起動される。） コールの条件は、 ACC←0（AM）、ACC←1（PM） DE←時刻を、秒に直したもの。（バイナリで2バイト）	AF以外は保存
CALL TIMRD （003B）	内蔵された時計の値を読み取る。リターン時の状態は、 ACC←0（AM）、ACC←1（PM） DE←時刻を、秒に直したもの。（バイナリで2バイト）	AF、DE以外は保存
CALL BRKEY （001E）	SHIFT＋BREAKが押されたかどうかをチェックする。 押されていれば、Zフラグはセット。 押されていなければ、Zフラグはリセット。	AF以外は保存
CALL GETL （0003）	キーボードから1行分を入力する。 あらかじめ、入力データをストアするスタートアドレスをDEレジスタに指定しておくこと。エンドマークはキャリッジリターンによる。 そして、エンドマークとして0Dコードがセットされる。 入力文字数は、0Dコードを含めて最大80文字。 キー入力にはエコーバックが行われ、カーソルコントロールも受けつけられる。 SHIFT＋BREAKが押されたら、DEレジスタの示すアドレスの先頭にBREAKコード、次にキャリッジリターンコードがセットされて戻る。 ひらがな、英小文字のキー入力の場合、たとえばLineA……という文字は4C　05　49　4E　45　06　41　……と入力される。ここで05は英小文字またはひらがなデータを表わすコードで、1番目に現われた05のあとの英文字またはカタカナは、英小文字またはひらがなとなり、次の06でその後の文字を英大文字、カタカナに戻す。そのため、ひらがな、英小文字が含まれている場合の入力文字数は80文字より少なくなる。	全レジスタ保存

<table>
<tr><th>サブルーチン名および
エントリーポイント
（16進番地）</th><th>サブルーチンの機能</th><th>レジスタ
保　存</th></tr>
<tr><td></td><td>る。また、この場合V－RAMのカラー情報データ（D800番地以降）の7ビット目をONにし、英小文字またはひらがながテレビ画面に表示される。</td><td></td></tr>
<tr><td>CALL GETKY
（001B）

GETKYによる特殊キーの取り込み</td><td>キーボードから1文字だけアスキーコードをACCに取り込む。
そのときキー入力がなければ、ACCに00がセットされて戻る

ただし、キー入力によるチャタリングは防止しておらず、エコーバックも行わない。
このGETKYを行う場合、DEL やCR などの特殊キーは、アスキーコードとは別の特殊なコードがACCに取り込まれる。
これらの特殊キーによってACCに取り込まれるコードを次に示す。
ここでディスプレイコードというのは、キャラクタジェネレータ内にあるキャラクタを呼びだすためのコード番号であり、この表では単に、ACCに取り込まれるコードのアスキーキャラクタに対応するディスプレイコードを示しているにすぎない。
<table>
<tr><th>特殊キー</th><th>ACCに取り込まれるコード</th><th>ディスプレイコード</th></tr>
<tr><td>DEL</td><td>60</td><td>C7</td></tr>
<tr><td>INST</td><td>61</td><td>C8</td></tr>
<tr><td>英数</td><td>62</td><td>C9</td></tr>
<tr><td>カナ</td><td>63</td><td>CA</td></tr>
<tr><td>BREAK</td><td>64</td><td>CB</td></tr>
<tr><td>CR</td><td>66</td><td>CD</td></tr>
<tr><td>カーソルキー↓</td><td>11</td><td>C1</td></tr>
<tr><td>カーソルキー↑</td><td>12</td><td>C2</td></tr>
<tr><td>カーソルキー→</td><td>13</td><td>C3</td></tr>
<tr><td>カーソルキー←</td><td>14</td><td>C4</td></tr>
<tr><td>HOME</td><td>15</td><td>C5</td></tr>
<tr><td>CLR</td><td>16</td><td>C6</td></tr>
</table>
</td><td></td></tr>
<tr><td>CALL ASC
（03DA）</td><td>ACCの下位4ビットを16進数と見なし、アスキーに変換したものをACCにセットしてリターンする。</td><td>AF以外は保存</td></tr>
<tr><td>CALL HEX
（03F9）</td><td>ACCの8ビットをアスキーと見なし、16進数に変換したものを、ACCの下位4ビットにセットしてリターンする。
リターン時のCF＝"0"　　ACC←16進数
リターン時のCF＝"1"　　ACCは保証されない。</td><td>AF以外は保存</td></tr>
<tr><td>CALL HLHEX
（0410）</td><td>連続した4個のアスキー列を16進数列と見なし、HLレジスタにセットしてリターンする。コールおよびリターン条件は次の通り。
DE ← ASCII列の先頭アドレス
（例"3""1""A""5"）
CALL HLHEX　　DE
CF＝0　　HL←16進数（例 HL＝31A5H）
CF＝1　　HLは保証されない。</td><td>AF、HL以外は保存</td></tr>
</table>

<table>
<tr><th>サブルーチン名および
エントリーポイント
（16進番地）</th><th>サブルーチンの機能</th><th>レジスタ
保　存</th></tr>
<tr><td>CALL 2HEX
（041F）</td><td>連続した2個のアスキー列を16進数列と見なし、ACCにセットしてリターンする。コールおよびリターン条件は次の通り。
DE ← ASCII列のアドレス（例"3""A"）
CALL 2HEX　　DE
CF＝0　　ACC←16進数（例 ACC＝3AH）
CF＝1　　ACCは保証されない。</td><td>AF、DE
以外は保存</td></tr>
<tr><td>CALL ??KEY
（0983）</td><td>カーソルを点滅させながら、キー入力を待つ。キー入力があるとディスプレイコードに変換してACCにセット後、リターンする。</td><td>AF以外は
保存</td></tr>
<tr><td>CALL ?ADCN
（0BB9）</td><td>アスキー値をディスプレイコードに変換する。コールおよびリターン条件は次の通り。
ACC ←アスキー値
CALL ?ADCN
ACC ←ディスプレイコード</td><td>AF以外は
保存</td></tr>
<tr><td>CALL ?DACN
（0BCE）</td><td>ディスプレイコードをアスキー値に変換する。コールおよびリターン条件は次の通り。
ACC ←ディスプレイコード
CALL ?DACN
ACC ← アスキー値</td><td>AF以外は
保存</td></tr>
<tr><td>CALL ?BLNK
（0DA6）</td><td>テレビ画面の垂直ブランキングをチェックする。垂直ブランキング期間になるまで待っており、ブランキングになったらリターンする。</td><td>全レジスタ
保存</td></tr>
<tr><td>CALL ?DPCT
（0DDC）</td><td>テレビ画面上のディスプレイをコントロールする。コール時のACCとコントロールの関係は次の通り。
<table>
<tr><th>ACC</th><th>コントロール内容</th><th>ACC</th><th>コントロール内容</th></tr>
<tr><td>C0H</td><td>スクローリング</td><td>C6H</td><td>CLR キーと同機能</td></tr>
<tr><td>C1H</td><td>↓キーと同機能</td><td>C7H</td><td>DEL キーと同機能</td></tr>
<tr><td>C2H</td><td>↑キーと同機能</td><td>C8H</td><td>INST キーと同機能</td></tr>
<tr><td>C3H</td><td>→キーと同機能</td><td>C9H</td><td>英数 キーと同機能</td></tr>
<tr><td>C4H</td><td>←キーと同機能</td><td>CAH</td><td>カナ キーと同機能</td></tr>
<tr><td>C5H</td><td>HOME キーと同機能</td><td>CBH</td><td>CR キーと同機能</td></tr>
</table></td><td>全レジスタ
保存</td></tr>
<tr><td>CALL ?PONT
（0FB1）</td><td>現在のテレビ画面におけるカーソル位置をHLにセットする。リターン条件は次の通り。
CALL ?PONT
HL ← テレビ画面上のカーソル位置(バイナリ)</td><td>AF、HL
以外は保存</td></tr>
</table>

F. アスキーコード、ディスプレイコード表

(1) アスキーコード表

表は、上位4ビットがカラム（列）に相当し、下位4ビットか、ロウ（行）に相当しており、たとえば、キャラクタ" A "のアスキーコードは、16進コードで41H、" Z "は5AHになっています。

ただし、表中の11H～16Hまでのコードは、カーソルコントロールのためのアスキーコードです。たとえばACCの内容が15Hのとき、CALL PRNT（モニターサブルーチン）を行うとカーソルホームが行われます。(" "を表示するのではありません) 枠のついていないカナ文字、たとえば" ア "（87H）などは小文字を表します。

アスキーコード表

LSD(下位) ＼ MSD(上位)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0 0000			SP	0	@	P	◆	日	↓	ー	タ	ミ	→	▗	▬	□
1 0001		↓	!	1	A	Q	H	月	。	ア	チ	ム	▐	口	♠	●
2 0010		↑	"	2	B	R	I	火	「	イ	ツ	メ	▄	田	▯	▁
3 0011		→	#	3	C	S	大	水	」	ウ	テ	モ	□	▌	▬	♥
4 0100		←	$	4	D	T	中	木	、	エ	ト	ヤ	▁	▐	▀	▯
5 0101		H	%	5	E	U	小	金	・	オ	ナ	ユ	□	▙	▔	◢
6 0110		C	&	6	F	V	¥	土	ヲ	カ	ニ	ヨ	▩	▐	▄	✕
7 0111			'	7	G	W	●	生	ァ	キ	ヌ	ラ	□	▀	▯	○
8 1000			(	8	H	X	☺	年	ィ	ク	ネ	リ	■	▄	▯	♣
9 1001			)	9	I	Y	♨	時	ゥ	ケ	ノ	ル	◤	▁	◣	□
A 1010			*	:	J	Z	▶	分	ェ	コ	ハ	レ	▐	▁	◹	◆
B 1011			+	;	K	[	◀	秒	ォ	サ	ヒ	ロ	▌	◹	◸	▦
C 1100			,	く	L	\	▙	円	ャ	シ	フ	ワ	○	◹	□	▩
D 1101	CR		−	=	M	]	▛	¥	ュ	ス	ヘ	ン	▙	▛	◹	▯
E 1110			.	>	N	↑	▟	£	ョ	セ	ホ	゛	▗	▚	◸	◣
F 1111			/	?	O	←	▀	↓	ッ	ソ	マ	°	▁	▚	□	π

(2) ディスプレイコード表

ディスプレイコードとは、キャラクタジェネレータ内にあるキャラクタを呼び出すためのコード番号です。テレビ画面へのキャラクタ表示はいつも、このコードをビデオRAMに転送することによって行われています。

モニタサブルーチンの"PRNT (0012H)"ルーチンや"MSG (0015H)"ルーチンでは、アスキーコードをこのディスプレイコードに変換して、カーソルの示すビデオRAMの位置へ、そのコードを転送しています。ただし表中のC1H～C6Hのコードはカーソルコントロール用の文字であり、枠のついていないカナ文字、たとえば"ア" (9EH) などは小文字を表しています。

DISPLAY (189ページ、FONT$ 参照：ROM CG I)

LSD(下位) ＼ MSD(上位)	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111
0 0000	SP	P	0		→	↑	π		↓	セ	ワ	厶	↓	日		SP
1 0001	A	Q	1		♠	く	!		チ	タ	ヌ		↓	月		
2 0010	B	R	2			[	"		コ	ス	フ	ィ	↑	火		
3 0011	C	S	3		■	♥	#		ソ	ト	ア	ュ	→	水		
4 0100	D	T	4		◆	]	$		シ	カ	ウ	ヲ	←	木		
5 0101	E	U	5		←	@	%		イ	ナ	エ	、	H	金		
6 0110	F	V	6		♣		&		ハ	ヒ	オ	ゥ	C	土		
7 0111	G	W	7		●	>	'		キ	テ	ヤ	ョ		生		
8 1000	H	X	8			(			ク	サ	ユ	°	H	年		
9 1001	I	Y	9		?	\	)		ニ	ン	ヨ	。	I	時		
A 1010	J	Z	−				+		マ	ツ	ホ	ェ	大	分		
B 1011	K		=				*		ノ	ロ	ヘ	ッ		秒		
C 1100	L		;						リ	ケ	レ	〝		円		
D 1101	M		/						モ	「	メ	゜	¥	¥		
E 1110	N		.						ミ	ァ	ル	ォ		£		
F 1111	O		,						ラ	ャ	ネ	ー				

V-RAM $D800～$DBFFの色情報エリアの第7ビットを用いた場合、英大文字は英小文字に、カタカナはひらがなとなります。そのときのディスプレイコード表を次に示します。ただし、表中のC1H～C6Hのコードはカーソルコントロール用の文字であり、枠のついていないひらがな、たとえば"あ"（9EH）などはひらがな小文字を表します。

DISPLAY　（189ページ、FONT$参照：ROM CG II）

LSD(下位) ＼ MSD(上位)	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111
0 0000	SP	p	0	▔	→	↑	π	□	↓	せ	わ	む	↓	日	▁	SP
1 0001	a	q	1	▏	♠	く	!	▏	ち	た	ぬ	」	↓	月	▁	▪
2 0010	b	r	2	▕	◥	「	"	▕	こ	す	ふ	い	↑	火	▁	▪
3 0011	c	s	3	▕	■	♥	#	▕	そ	と	あ	ゆ	→	水	～	▪▪
4 0100	d	t	4	▔	◆	」	$	▔	し	か	う	を	←	木	◤	▪
5 0101	e	u	5	▏	←	@	%	▏	い	な	え	、	H	金	◥	▪
6 0110	f	v	6	▔	♣	◢	&	▨	は	ひ	お	う	C	土	◸	▗
7 0111	g	w	7	▏	●	>	'	▨	き	て	や	ょ	◓	生	◸	▪▪
8 1000	h	x	8	▁	○	▚	(	▁	く	さ	ゆ	°	H	年	◸	▪
9 1001	i	y	9	▕	?	\	)	▕	に	ん	よ	•	工	時	K	▖
A 1010	j	z	ー	▃	○	▚	+	▃	ま	つ	ほ	ぇ	大	分	K	▗
B 1011	k	┼	=	◨	◪	▛	*	◧	の	ろ	へ	っ	◸	秒	◸	▚
C 1100	l	┗	;	▢	◹	▟	▞	▁	り	け	れ	゛	◸	円	◸	▄
D 1101	m	┛	/	▢	◣	▙	▩	▕	も	「	め	°	¥	¥	◸	▙
E 1110	n	┣	▪	▁	◢	▛	▟	▁	み	ぁ	る	ぉ	●	£	◸	▟
F 1111	o	┻	,	▕	:	▜	◹	▕	ら	ゃ	ね	ー	☺	～	▩	▦

G. エラーメッセージ一覧表

コンピュータ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テレビ画面上に、エラーメッセージが送られます。

①エラーの種類　ＥＲＲＯＲ　　（直接命令を実行中でのエラー）
②エラーの種類　ＥＲＲＯＲ　ＩＮ　文番号　（プログラム実行中でのエラー）

①直接命令を実行中に発生し、文番号に無関係なエラーとみなされる場合

②は特にプログラムの実行中において、文番号××××に発生したエラーとみなされる場合

エラー番号	表示ステートメント	エラーの内容
1	Syntax error	文法上の誤り
2	Over flow error	数値データが範囲外、演算結果がオーバフローした
3	Illegal data error	規定外の数値、変数が使われた
4	Type mismatch error	データと変数の型が一致しない
5	String length error	ストリングの長さが255文字を越えた
6	Memory capacity error	メモリ容量不足となった
7	Array def.　error	同じ配列変数を前より大きく定義したが、未定義の配列変数を使用した
8	Linelength error	１行の長さが制限を越えた
9		
10	GOSUB nesting error	GOSUB文のネスティングが15を越えた
11	FOR-NEXT error	FOR-NEXT文のネスティングが15を越えた
12	DEF FN nesting error	DEF FN文による関数定義のネスティングが６を越えた
13	NEXT error	対応するFOR文のないNEXT文が使われた
14	RETURN error	対応するGOSUB文のないRETURN文が使われた
15	Un def.　Function error	定義されていない関数が使われた
16	Un def.　line error	存在しない文番号，ラベルを参照しようとした

エラー番号	表示ステートメント	エラーの内容
17	Can't CONT error	CONT文によるプログラムの継続ができない
18	Memory protection error	BASICインタープリタの管理エリア内への書き込み要求をした
19	Instruction error	ダイレクトコマンドとステートメントを混同して使った
20	Can't RESUME error	RESUMEが実行できない
21	RESUME error	エラーが発生していないのにRESUMEしようとした
22		
23		
24	READ error	対応するDATA文のないREAD文が使われた
25		
26		
27		
28	System id error	BASIC以外のディスクをアクセスした
29	Framing error	フレミングエラー
30	Overrun error	オーバーランエラー
31	Parity error	パリティエラー
32		
33		
34		
35		
36		
37		
38		
39		
40	File not found error	存在しないファイルを参照した
41	Hardware error	ディスクドライブのハード上のエラーが発生した
42	Already exist error	すでに存在するファイル名を新たに登録しようとした
43	Already open error	すでにオープンされているファイルをさらにオープンしようとした

エラー番号	表示ステートメント	エラーの内容
44	Not open error	オープンされていないファイルを参照または CLOSE、KILLを実行した
45		
46	Write protect error	書き込み禁止ファイル（クイックディスクの書き込み禁止用ツメが折られている）
47		
48		
49		
50	Not ready error	ドライブがレディ状態でない
51	Too many files error	登録できる最大ファイル数を越すファイルを登録しようとした（最大63）
52	Disk mismatch error	ディスクが一致しない
53	No file space error	ディスク上のファイルスペースが無くなった
54	Unformat error	イニシャライズされていないディスク
55	Too long file error	1つのBSDファイルのデータの総バイト数が64Kを越えた
56		
57		
58	Dev. name error	装置名の記述が誤っている
59	Can'texecute error	実行できない装置に対して実行しようとした
60	Illegal filename error	ファイル名エラー
61	Illegal filemode error	ファイルモードエラー
62		
63	Out of file error	カセットテープのファイルデータ読み込みで、アウトオブファイルが起きた
64	Logical number error	ロジカルナンバーエラー
65	LPT: not ready error	プリンタが接続されていないか、OFF状態である
66		プリンタにメカトラブルが起きた
67		プリンタ用紙ぎれ
68	Dev. mode error	装置のモードエラー
69	Unprintable error	メッセージの定義されていないエラー
70	Check sum error	チェックサムエラー

H．英小文字、ひらがな、一部のグラフィックパターに対する注意

英小文字、ひらがなのキャラクタは、内部において特別なデータ形式となっていますので扱いには注意が必要です。

"ＳＨＡＲＰ"→内部のデータ並びは、$53、$48、$41、$52、$50
Ｓ　Ｈ　Ａ　Ｒ　Ｐ

"Ｓｈａｒｐ"→内部のデータ並びは、

$53	$05	$48	$41	$52	$50	$06
Ｓ	識別コード	ｈ	ａ	ｒ	ｐ	識別コード

"ＡｂｃｄＥ"→

$41	$05	$42	$43	$44	$06	$45
Ａ	識別コード	ｂ	ｃ	ｄ	識別コード	Ｅ

"あいうえお"→

$05	$91	$92	$93	$94	$95	$06
識別コード	あ	い	う	え	お	識別コード

このように、識別コード（$05、$06）が、文字体を変化させるときに付け加えられますので、内部での文字数が増加します。したがって、BASICのMID$、LEFT$、RIGHT$、LENなどの文字列を扱う関数を使用する場合は、内部での文字数に注意してください。

```
10  A$="SHARP"
20  PRINT  LEN (A$)
```
→結果は5となります。

```
10  A$="Sharp"
20  PRINT  LEN (A$)
```
→結果は7となります。

BASICでプログラムリストを、出力した場合の画面表示と、カラープロッタプリンタへの印字は、次の表のようになります。

命　　令	キー入力時	画面表示	プロッタプリンタ印字
LIST	"ＡｂｃＤ" "アいうエ"	"Ａ☻ＢＣ☺Ｄ" "ア☻イウ☺エ"	
LIST／P	"ＡｂｃＤ" "アいうエ"		"Ａ．ＢＣ．Ｄ" "ア．イウ．エ"

カラープロッタプリンタが持っていないグラフィックパターンを印字させた場合は、ペンの色を変えて、16進アスキーコードで印字されます。

Ⅰ．漢字コード表

	コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
記号	2120									;	?	!					
	2130									仝	々	〆	○	―	―	―	／
	2140	＼	～	∥	｜	…	‥					（	）	〔	〕	［	］
	2150	｛	｝	〈	〉	《	》	「	」	『	』	【	】	＋	－	±	×
	2160	÷	＝	≠	＜	＞	≦	≧	∞	∴	♂	♀		′	″	℃	￥
	2170	$	¢	£	%	#	&	*	@	§	☆	★	○	●	◎	◇	
	2220		◆	□	■	△	▲	▽	▼	※	〒	→	←	↑	↓	＝	―
	2230	｜	ｒ	ｒ	ｊ	∟	├	┬	┤	┴	＋						
	2240		。	／	》												
	2320							*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
英・数字	2330	0	1	2	3	4	5	6	7	8	9						
	2340		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	2350	P	Q	R	S	T	U	V	W	X	Y	Z					
	2360		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
	2370	p	q	r	s	t	u	v	w	x	y	z					
ひらがな	2420		ぁ	あ	ぃ	い	ぅ	う	ぇ	え	ぉ	お	か	が	き	ぎ	く
	2430	ぐ	け	げ	こ	ご	さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た
	2440	だ	ち	ぢ	っ	っ	づ	て	で	と	ど	な	に	ぬ	ね	の	は
	2450	ば	は	ひ	び	ひ	ふ	ぶ	ふ	へ	べ	へ	ほ	ぼ	ほ	ま	み
	2460	む	め	も	ゃ	や	ゅ	ゆ	ょ	よ	ら	り	る	れ	ろ	ゎ	わ
	2470	ゐ	ゑ	を	ん		0	1	2	3	4	5	6	7	8	9	
カタカナ	2520		ァ	ア	ィ	イ	ゥ	ウ	ェ	エ	ォ	オ	カ	ガ	キ	ギ	ク
	2530	グ	ケ	ゲ	コ	ゴ	サ	ザ	シ	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ
	2540	ダ	チ	ヂ	ッ	ツ	ヅ	テ	デ	ト	ド	ナ	ニ	ヌ	ネ	ノ	ハ
	2550	バ	パ	ヒ	ビ	ヒ	フ	ブ	フ	ヘ	ベ	ヘ	ホ	ボ	ホ	マ	ミ
	2560	ム	メ	モ	ャ	ヤ	ュ	ユ	ョ	ヨ	ラ	リ	ル	レ	ロ	ヮ	ワ
	2570	ヰ	ヱ	ヲ	ン	ヴ	ヵ	ヶ							・		
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

※コードは16進で表現されています。

例えば“B”のコードは2340＋2＝2342となります。

	JIS	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ギリシア文字	2620		Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο
	2630	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω							
	2640		α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο
	2650	π	ρ	σ	τ	υ	φ	χ	ψ	ω							
	2660		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	2670		P	Q	R	S	T	U	V	W	X	Y	Z				
ロシア文字	2720		А	Б	В	Г	Д	Е	Ё	Ж	З	Н	Й	К	Л	М	Н
	2730	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
	2740	Ю	Я			0	1	2	3	4	5	6	7	8	9		
	2750		а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н
	2760	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э
	2770	ю	я		0	1	2	3	4	5	6	7	8	9			
ア	3020		亜	唖	娃	阿	哀	愛	挨	姶	逢	葵	茜	穐	悪	握	渥
	3030	旭	葦	芦	鯵	梓	圧	斡	扱	宛	姐	虻	飴	絢	綾	鮎	或
	3040	粟	袷	安	庵	按	暗	案	闇	鞍	杏						
イ	3040											以	伊	位	依	偉	囲
	3050	夷	委	威	尉	惟	意	慰	易	椅	為	畏	異	移	維	緯	胃
	3060	萎	衣	謂	違	遺	医	井	亥	域	育	郁	磯	一	壱	溢	逸
	3070	稲	茨	芋	鰯	允	印	咽	員	因	姻	引	飲	淫	胤	蔭	
	3120		院	陰	隠	韻	吋										
ウ	3120							右	宇	烏	羽	迂	雨	卯	鵜	窺	丑
	3130	碓	臼	渦	嘘	唄	鬱	蔚	鰻	姥	厩	浦	瓜	閏	噂	云	運
	3140	雲															
エ	3140		作	餌	叡	営	嬰	影	映	曳	栄	永	泳	洩	瑛	盈	穎
	3150	頴	英	衛	詠	鋭	液	疫	益	駅	悦	謁	越	閲	榎	厭	円
	3160	園	堰	奄	宴	延	怨	掩	援	沿	演	炎	焔	煙	燕	猿	縁
	3170	艶	苑	薗	遠	鉛	鴛	塩									
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
オ	3170								於	汚	甥	凹	央	奥	往	応	
	3220		押	旺	横	欧	殴	王	翁	襖	鴬	鴎	黄	岡	沖	荻	億
	3230	屋	憶	臆	桶	牡	乙	俺	卸	恩	温	穏	音				
カ	3230													下	化	仮	何
	3240	伽	価	佳	加	可	嘉	夏	嫁	家	寡	科	暇	果	架	歌	河
	3250	火	珂	禍	禾	稼	箇	花	苛	茄	荷	華	菓	蝦	課	嘩	貨
	3260	迦	過	霞	蚊	俄	峨	我	牙	画	臥	芽	蛾	賀	雅	餓	駕
	3270	介	会	解	回	塊	壊	廻	快	怪	悔	恢	懐	戒	拐	改	
	3320		魁	晦	械	海	灰	界	皆	絵	芥	蟹	開	階	貝	凱	劾
	3330	外	咳	害	崖	慨	概	涯	碍	蓋	街	該	鎧	骸	浬	馨	蛙
	3340	垣	柿	蠣	鈎	劃	嚇	各	廓	拡	撹	格	核	殻	獲	確	穫
	3350	覚	角	赫	較	郭	閣	隔	革	学	岳	楽	額	顎	掛	笠	樫
	3360	橿	梶	鰍	潟	割	喝	恰	括	活	渇	滑	葛	褐	轄	且	鰹
	3370	叶	椛	樺	鞄	株	兜	竈	蒲	釜	鎌	噛	鴨	栢	茅	萱	
	3420		粥	刈	苅	瓦	乾	侃	冠	寒	刊	勘	勧	巻	喚	堪	姦
	3430	完	官	寛	干	幹	患	感	慣	憾	換	敢	柑	桓	棺	款	歓
	3440	汗	漢	澗	灌	環	甘	監	看	竿	管	簡	緩	缶	翰	肝	艦
	3450	莞	観	諌	貫	還	鑑	間	閑	関	陥	韓	館	舘	丸	含	岸
	3460	巌	玩	癌	眼	岩	翫	贋	雁	頑	顔	願					
キ	3460												企	伎	危	喜	器
	3470	基	奇	嬉	寄	岐	希	幾	忌	揮	机	旗	既	期	棋	棄	
	3520		機	帰	毅	気	汽	畿	祈	季	稀	紀	徽	規	記	貴	起
	3530	軌	輝	飢	騎	鬼	亀	偽	儀	妓	宜	戯	技	擬	欺	犠	疑
	3540	祇	義	蟻	誼	議	掬	菊	鞠	吉	吃	喫	桔	橘	詰	砧	杵
	3550	黍	却	客	脚	虐	逆	丘	久	仇	休	及	吸	宮	弓	急	救
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	3560	朽	求	汲	泣	灸	球	究	窮	笈	級	糾	給	旧	牛	去	居
	3570	巨	拒	拠	挙	渠	虚	許	距	鋸	漁	禦	魚	亨	享	京	
	3620		供	侠	僑	兇	競	共	凶	協	匡	卿	叫	喬	境	峡	強
	3630	彊	怯	恐	恭	挟	教	橋	況	狂	狭	矯	胸	脅	興	蕎	郷
	3640	鏡	響	饗	驚	仰	凝	尭	暁	業	局	曲	極	玉	桐	粁	僅
	3650	勤	均	巾	錦	斤	欣	欽	琴	禁	禽	筋	緊	芹	菌	衿	襟
	3660	謹	近	金	吟	銀											
ク	3660						九	倶	句	区	狗	玖	矩	苦	躯	駆	駈
	3670	駒	具	愚	虞	喰	空	偶	寓	遇	隅	串	櫛	釧	屑	屈	
	3720		掘	窟	沓	靴	轡	窪	熊	隈	粂	栗	繰	桑	鍬	勲	君
	3730	薫	訓	群	軍	郡											
ケ	3730						卦	袈	祁	係	傾	刑	兄	啓	圭	珪	型
	3740	契	形	径	恵	慶	慧	憩	掲	携	敬	景	桂	渓	畦	稽	系
	3750	経	継	繋	罫	茎	荊	蛍	計	詣	警	軽	頸	鶏	芸	迎	鯨
	3760	劇	戟	撃	激	隙	桁	傑	欠	決	潔	穴	結	血	訣	月	件
	3770	倹	倦	健	兼	券	剣	喧	圏	堅	嫌	建	憲	懸	拳	捲	
	3820		検	権	牽	犬	献	研	硯	絹	県	肩	見	謙	賢	軒	遣
	3830	鍵	険	顕	験	鹸	元	原	厳	幻	弦	減	源	玄	現	絃	舷
	3840	言	諺	限													
コ	3840				乎	個	古	呼	固	姑	孤	己	庫	弧	戸	故	枯
	3850	湖	狐	糊	袴	股	胡	蝶	虎	誇	跨	鈷	雇	顧	鼓	五	互
	3860	伍	午	呉	吾	娯	後	御	悟	梧	檎	瑚	碁	語	誤	護	醐
	3870	乞	鯉	交	佼	侯	候	倖	光	公	功	効	勾	厚	口	向	
	3920		后	喉	坑	垢	好	孔	孝	宏	工	巧	巷	幸	広	庚	康
	3930	弘	恒	慌	抗	拘	控	攻	昂	晃	更	杭	校	梗	構	江	洪
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	3940	浩	港	溝	甲	皇	硬	稿	糠	紅	紘	絞	綱	耕	考	肯	肱
	3950	腔	膏	航	荒	行	衡	講	貢	購	郊	酵	鉱	礦	鋼	閤	降
	3960	項	香	高	鴻	剛	劫	号	合	壕	拷	濠	豪	轟	麹	克	刻
	3970	告	国	穀	酷	鵠	黒	獄	漉	腰	甑	忽	惚	骨	狛	込	
	3A20		此	頃	今	困	坤	墾	婚	恨	懇	昏	昆	根	梱	混	痕
	3A30	紺	艮	魂													
サ	3A30				些	佐	叉	唆	嵯	左	差	査	沙	瑳	砂	詐	鎖
	3A40	裟	坐	座	挫	債	催	再	最	哉	塞	妻	宰	彩	才	採	栽
	3A50	歳	済	災	采	犀	砕	砦	祭	斎	細	菜	裁	載	際	剤	在
	3A60	材	罪	財	冴	坂	阪	堺	榊	肴	咲	崎	埼	碕	鷺	作	削
	3A70	咋	搾	昨	朔	柵	窄	策	索	錯	桜	鮭	笹	匙	冊	刷	
	3B20		察	拶	撮	擦	札	殺	薩	雑	皐	鯖	捌	錆	鮫	皿	晒
	3B30	三	傘	参	山	惨	撒	散	桟	燦	珊	産	算	纂	蚕	讃	賛
	3B40	酸	餐	斬	暫	残											
シ	3B40						仕	仔	伺	使	刺	司	史	嗣	四	士	始
	3B50	姉	姿	子	屍	市	師	志	思	指	支	孜	斯	施	旨	枝	止
	3B60	死	氏	獅	祉	私	糸	紙	紫	肢	脂	至	視	詞	詩	試	誌
	3B70	諮	資	賜	雌	飼	歯	事	似	侍	児	字	寺	慈	持	時	
	3C20		次	滋	治	爾	璽	痔	磁	示	而	耳	自	蒔	辞	汐	鹿
	3C30	式	識	鳴	竺	軸	宍	雫	七	叱	執	失	嫉	室	悉	湿	漆
	3C40	疾	質	実	蔀	篠	偲	柴	芝	屡	蕊	縞	舎	写	射	捨	赦
	3C50	斜	煮	社	紗	者	謝	車	遮	蛇	邪	借	勺	尺	杓	灼	爵
	3C60	酌	釈	錫	若	寂	弱	惹	主	取	守	手	朱	殊	狩	珠	種
	3C70	腫	趣	洒	首	儒	受	呪	寿	授	樹	綬	需	囚	収	周	
	3D20		宗	就	州	修	愁	拾	洲	秀	秋	終	繍	習	臭	舟	蒐
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	3D30	衆	襲	讐	蹴	輯	週	酋	酬	集	醜	什	住	充	十	従	戎
	3D40	柔	汁	渋	獣	縦	重	銃	叔	夙	宿	淑	祝	縮	粛	塾	熟
	3D50	出	術	述	俊	峻	春	瞬	竣	舜	駿	准	循	旬	楯	殉	淳
	3D60	準	潤	盾	純	巡	遵	醇	順	処	初	所	暑	曙	渚	庶	緒
	3D70	署	書	薯	藷	諸	助	叙	女	序	徐	恕	鋤	除	傷	償	
	3E20		勝	匠	升	召	哨	商	唱	嘗	奨	妾	娼	宵	将	小	少
	3E30	尚	庄	床	廠	彰	承	抄	招	掌	捷	昇	昌	昭	晶	松	梢
	3E40	樟	樵	沼	消	渉	湘	焼	焦	照	症	省	硝	礁	祥	称	章
	3E50	笑	粧	紹	肖	菖	蒋	蕉	衝	裳	訟	証	詔	詳	象	賞	醤
	3E60	鉦	鍾	鐘	障	鞘	上	丈	丞	乗	冗	剰	城	場	壌	嬢	常
	3E70	情	擾	条	杖	浄	状	畳	穣	蒸	譲	醸	錠	嘱	埴	飾	
	3F20		拭	植	殖	燭	織	職	色	触	食	蝕	辱	尻	伸	信	侵
	3F30	唇	娠	寝	審	心	慎	振	新	晋	森	榛	浸	深	申	疹	真
	3F40	神	秦	紳	臣	芯	薪	親	診	身	辛	進	針	震	人	仁	刃
	3F50	塵	壬	尋	甚	尽	腎	訊	迅	陣	靱						
ス	3F50											笥	諏	須	酢	図	厨
	3F60	逗	吹	垂	帥	推	水	炊	睡	粋	翠	衰	遂	酔	錐	錘	随
	3F70	瑞	髄	崇	嵩	数	枢	趨	雛	据	杉	椙	菅	頗	雀	裾	
	4020		澄	摺	寸												
セ	4020					世	瀬	畝	是	凄	制	勢	姓	征	性	成	政
	4030	整	星	晴	棲	栖	正	清	牲	生	盛	精	聖	声	製	西	誠
	4040	誓	請	逝	醒	青	静	斉	税	脆	隻	席	惜	戚	斥	昔	析
	4050	石	積	籍	績	脊	責	赤	跡	蹟	碩	切	拙	接	摂	折	設
	4060	窃	節	説	雪	絶	舌	蝉	仙	先	千	占	宣	専	尖	川	戦
	4070	扇	撰	栓	栴	泉	浅	洗	染	潜	煎	煽	旋	穿	箭	線	
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	4120		繊	羨	腺	舛	船	薦	詮	賤	践	選	遷	銭	銑	閃	鮮
	4130	前	善	漸	然	全	禅	繕	膳	糎							
ソ	4130										噌	塑	岨	措	曾	曽	楚
	4140	狙	疏	疎	礎	祖	租	粗	素	組	蘇	訴	阻	遡	鼠	僧	創
	4150	双	叢	倉	喪	壮	奏	爽	宋	層	匝	惣	想	捜	掃	挿	掻
	4160	操	早	曹	巣	槍	槽	漕	燥	争	痩	相	窓	糟	総	綜	聡
	4170	草	荘	葬	蒼	藻	装	走	送	遭	鎗	霜	騒	像	増	憎	
	4220		臓	蔵	贈	造	促	側	則	即	息	捉	束	測	足	速	俗
	4230	属	賊	族	続	卒	袖	其	揃	存	孫	尊	損	村	遜		
タ	4230															他	多
	4240	太	汰	詑	唾	堕	妥	惰	打	柁	舵	楕	陀	駄	騨	体	堆
	4250	対	耐	岱	帯	待	怠	態	戴	替	泰	滞	胎	腿	苔	袋	貸
	4260	退	逮	隊	黛	鯛	代	台	大	第	醍	題	鷹	滝	瀧	卓	啄
	4270	宅	托	択	拓	沢	濯	琢	託	鐸	濁	諾	茸	凧	蛸	只	
	4320		叩	但	達	辰	奪	脱	巽	竪	辿	棚	谷	狸	鱈	樽	誰
	4330	丹	単	嘆	坦	担	探	旦	歎	淡	湛	炭	短	端	箪	綻	耽
	4340	胆	蛋	誕	鍛	団	壇	弾	断	暖	檀	段	男	談			
チ	4340														値	知	地
	4350	弛	恥	智	池	痴	稚	置	致	蜘	遅	馳	築	畜	竹	筑	蓄
	4360	逐	秩	窒	茶	嫡	着	中	仲	宙	忠	抽	昼	柱	注	虫	衷
	4370	註	酎	鋳	駐	樗	瀦	猪	苧	著	貯	丁	兆	凋	喋	寵	
	4420		帖	帳	庁	弔	張	彫	徴	懲	挑	暢	朝	潮	牒	町	眺
	4430	聴	脹	腸	蝶	調	諜	超	跳	銚	長	頂	鳥	勅	捗	直	朕
	4440	沈	珍	賃	鎮	陳											
ツ	4440						津	墜	椎	槌	追	鎚	痛	通	塚	栂	掴
	4450	槻	佃	漬	柘	辻	蔦	綴	鍔	椿	潰	坪	壺	嬬	紬	爪	吊
	4460	釣	鶴														
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
テ	4460			亭	低	停	偵	剃	貞	呈	堤	定	帝	底	庭	廷	弟
	4470	悌	抵	挺	提	梯	汀	碇	禎	程	締	艇	訂	諦	蹄	逓	
	4520		邸	鄭	釘	鼎	泥	摘	擢	敵	滴	的	笛	適	鏑	溺	哲
	4530	徹	撤	轍	迭	鉄	典	塡	天	展	店	添	纏	甜	貼	転	顚
	4540	点	伝	殿	澱	田	電										
ト	4540							兎	吐	堵	塗	妬	屠	徒	斗	杜	渡
	4550	登	菟	賭	途	都	鍍	砥	礪	努	度	土	奴	怒	倒	党	冬
	4560	凍	刀	唐	塔	塘	套	宕	島	嶋	悼	投	搭	東	桃	梼	棟
	4570	盗	淘	湯	涛	灯	燈	当	痘	禱	等	答	筒	糖	統	到	
	4620		董	蕩	藤	討	謄	豆	踏	逃	透	鐙	陶	頭	騰	闘	働
	4630	動	同	堂	導	憧	撞	洞	瞳	童	胴	萄	道	銅	峠	鴇	匿
	4640	得	徳	瀆	特	督	禿	篤	毒	独	読	栃	橡	凸	突	椴	届
	4650	鳶	苫	寅	酉	瀞	噸	屯	惇	敦	沌	豚	遁	頓	呑	曇	鈍
ナ	4660	奈	那	内	乍	凪	薙	謎	灘	捺	鍋	楢	馴	縄	暖	南	楠
	4670	軟	難	汝													
ニ	4670				二	尼	弐	邇	匂	賑	肉	虹	廿	日	乳	入	
	4720		如	尿	韮	任	妊	忍	認								
ヌ	4720									濡							
ネ	4720										禰	祢	寧	葱	猫	熱	年
	4730	念	捻	撚	燃	粘											
ノ	4730						乃	廼	之	埜	嚢	悩	濃	納	能	脳	膿
	4740	農	覗	蚤													
ハ	4740				巴	把	播	覇	杷	波	派	琶	破	婆	罵	芭	馬
	4750	俳	廃	拝	排	敗	杯	盃	牌	背	肺	輩	配	倍	培	媒	梅
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	4760	楳	煤	狽	買	売	賠	陪	這	蠅	秤	矧	萩	伯	剝	博	拍
	4770	柏	泊	白	箔	粕	舶	薄	迫	曝	漠	爆	縛	莫	駁	麦	
	4820		函	箱	硲	箸	肇	筈	櫨	幡	肌	畑	畠	八	鉢	溌	発
	4830	醗	髪	伐	罰	抜	筏	閥	鳩	噺	塙	蛤	隼	伴	判	半	反
	4840	叛	帆	搬	斑	板	氾	汎	版	犯	班	畔	繁	般	藩	販	範
	4850	釆	煩	頒	飯	挽	晩	番	盤	磐	蕃	蛮					
ヒ	4850												匪	卑	否	妃	庇
	4860	彼	悲	扉	批	披	斐	比	泌	疲	皮	碑	秘	緋	罷	肥	被
	4870	誹	費	避	非	飛	樋	簸	備	尾	微	枇	毘	琵	眉	美	
	4920		鼻	柊	稗	匹	疋	髭	彦	膝	菱	肘	弼	必	畢	筆	逼
	4930	檜	姫	媛	紐	百	謬	俵	彪	標	氷	漂	瓢	票	表	評	豹
	4940	廟	描	病	秒	苗	錨	鋲	蒜	蛭	鰭	品	彬	斌	浜	瀕	貧
	4950	賓	頻	敏	瓶												
フ	4950					不	付	埠	夫	婦	富	冨	布	府	怖	扶	敷
	4960	斧	普	浮	父	符	腐	膚	芙	譜	負	賦	赴	阜	附	侮	撫
	4970	武	舞	葡	蕪	部	封	楓	風	葺	蕗	伏	副	復	幅	服	
	4A20		福	腹	複	覆	淵	弗	払	沸	仏	物	鮒	分	吻	噴	墳
	4A30	憤	扮	焚	奮	粉	糞	紛	雰	文	聞						
ヘ	4A30											丙	併	兵	塀	幣	平
	4A40	弊	柄	並	蔽	閉	陛	米	頁	僻	壁	癖	碧	別	瞥	蔑	箆
	4A50	偏	変	片	篇	編	辺	返	遍	便	勉	娩	弁	鞭			
ホ	4A50														保	舗	鋪
	4A60	圃	捕	歩	甫	補	輔	穂	募	墓	慕	戊	暮	母	簿	菩	倣
	4A70	俸	包	呆	報	奉	宝	峰	峯	崩	庖	抱	捧	放	方	朋	
	4B20		法	泡	烹	砲	縫	胞	芳	萌	蓬	蜂	褒	訪	豊	邦	鋒
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	4B30	飽	鳳	鵬	乏	亡	傍	剖	坊	妨	帽	忘	忙	房	暴	望	某
	4B40	棒	冒	紡	肪	膨	謀	貌	貿	鉾	防	吠	頬	北	僕	卜	墨
	4B50	撲	朴	牧	睦	穆	釦	勃	没	殆	堀	幌	奔	本	翻	凡	盆
マ	4B60	摩	磨	魔	麻	埋	妹	昧	枚	毎	哩	槇	幕	膜	枕	鮪	柾
	4B70	鱒	桝	亦	俣	又	抹	末	沫	迄	儘	繭	麿	万	慢	満	
	4C20		漫	蔓													
ミ	4C20				味	未	魅	巳	箕	岬	密	蜜	湊	蓑	稔	脈	妙
	4C30	粍	民	眠													
ム	4C30				務	夢	無	牟	矛	霧	鵡	椋	婿	娘			
メ	4C30														冥	名	命
	4C40	明	盟	迷	銘	鳴	姪	牝	滅	免	棉	綿	緬	面	麺		
モ	4C40															摸	模
	4C50	茂	妄	孟	毛	猛	盲	網	耗	蒙	儲	木	黙	目	杢	勿	餅
	4C60	尤	戻	籾	貰	問	悶	紋	門	匁							
ヤ	4C60										也	冶	夜	爺	耶	野	弥
	4C70	矢	厄	役	約	薬	訳	躍	靖	柳	薮	鑓					
ユ	4C70												愉	愈	油	癒	
	4D20		諭	輸	唯	佑	優	勇	友	宥	幽	悠	憂	揖	有	柚	湧
	4D30	涌	猶	猷	由	祐	裕	誘	遊	邑	郵	雄	融	夕			
ヨ	4D30															予	余	与
	4D40	誉	輿	預	傭	幼	妖	容	庸	揚	揺	擁	曜	楊	様	洋	溶
	4D50	熔	用	窯	羊	耀	葉	蓉	要	謡	踊	遙	陽	養	慾	抑	欲
	4D60	沃	浴	翌	翼	淀											
ラ	4D60						羅	螺	裸	来	莱	頼	雷	洛	絡	落	酪
	4D70	乱	卵	嵐	欄	濫	藍	蘭	覧								
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

	コード	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
リ	4D70									利	吏	履	李	梨	理	璃	
	4E20		痢	裏	裡	里	離	陸	律	率	立	葎	掠	略	劉	流	溜
	4E30	琉	留	硫	粒	隆	竜	龍	侶	慮	旅	虜	了	亮	僚	両	凌
	4E40	寮	料	梁	涼	猟	療	瞭	稜	糧	良	諒	遼	量	陵	領	力
	4E50	緑	倫	厘	林	淋	燐	琳	臨	輪	隣	鱗	麟				
ル	4E50													瑠	塁	涙	累
	4E60	類															
レ	4E60		令	伶	例	冷	励	嶺	怜	玲	礼	苓	鈴	隷	零	霊	麗
	4E70	齢	暦	歴	列	劣	烈	裂	廉	恋	憐	漣	煉	簾	練	聯	
	4F20		蓮	連	錬												
ロ	4F20					呂	魯	櫓	炉	賂	路	露	労	婁	廊	弄	朗
	4F30	楼	榔	浪	漏	牢	狼	篭	老	聾	蠟	郎	六	麓	禄	肋	録
	4F40	論															
ワ	4F40		倭	和	話	歪	賄	脇	惑	枠	鷲	亙	亘	鰐	詫	藁	蕨
	4F50	椀	湾	碗	腕												
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

J．索引

[A] ABS ……22
AND ……53
ASC ……26
ASCCHR$……27
ATN ……23
AUTO ……94
AXIS ……217
[B] BLINE……179
BOOT ……145
BOX……181
BYE……144
[C] CCOLOR……168
CHAIN ……153
CHR$……27
CIRCLE ……182
CLOSE# ……161
CLR ……101
CLS ……104
COLOR ……166
CONSOLE ……105
CONT ……102
COS ……23
CRESET……176
CSET ……175
DSRH……16
CSRV……16
CURSOR……106
[D] DEFAULT……165
DEF FN ……132
DEF KEY ……135
DELETE……95、164
DIM ……129
DIR ……159
DIR/P ……222
[E] EOF (#) ……160
END……110
ERL ……16
ERN ……16
EXP ……22
[F] FONT$ ……189
FOR～NEXT……111
FPRINT ……192
[G] GET ……128
GOSUB～RETURN……116
GOTO ……114
GPRINT……216
[H] HCOPY ……220
HEXCHR$ ……27
HSET ……215
[I] IF～GOSUB ……120
IF～GOTO ……119
IF～THEN～ELSE ……118
INIT……137
INP@ ……225
INPUT ……127
INPUT# ……158
INT ……22
[K] KEY LIST ……136
KILL# ……163
KPATTERN ……188
[L] LABEL ……113
LEFT$ ……24

LEN ……27
LET ……108
LIMIT ……228
LINE ……177
LIST ……96
LIST/P ……219
LN ……23
LOAD ……147
LOAD ALL ……149
LOG ……23
M MERGE ……154
MID$ ……24
MUSIC……194
N NEW ……99
NEW ON ……100
NOISE……200
NOT ……53
O ON ERROR GOTO ……229
ON～GOSUB……117
ON～GOTO ……115
OR ……53
OUT@……226
P PAGE ……208
PAI……23
PAINT ……184
PAL ……169
PATTERN……185
PCIRCLE ……218
PCOLOR……206
PEEK ……223
PHOME ……214
PLINE……210
PLOT ……221
PMODE ……204
PMOVE ……212
POINT……193
POKE ……224
POSH……16
POSITION ……187
POSV……16
PRINT……121
PRINT# ……157
PRINT/P ……209
PRINT USING……123
PRTY ……143
PSKIP ……207
PTEST ……203
R RAD ……23
READ～DATA ……130
REM……107
RENAME ……162
RENUM ……98
RESET ……173
RESTORE ……131
RESUME ……230
RIGHT$ ……24
RLINE……211
RMOVE ……213
RND ……28
ROPEN#……156
RUN ……103、151
S SAVE ……148
SAVE ALL……150
SEARCH ……97
SET ……171
SGN ……22

SIN ·····23
SIZE ·····16
SOUND ·····197
SPC·····25
SQR ·····22
STOP ·····109
STR$ ·····26
[T] TAB ·····25
TAN ·····23
TEMPO ·····199
TI$ ·····16
TR OFF ·····134
TR ON ·····133
[U] USR ·····227
[V] VAL ·····26
VERIFY ·····152
VOICE·····201
[W] WAIT ·····146
WOPEN# ·····155
[X] XOR ·····54

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)
産業機器事業本部　〒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)
国内産機営業本部　〒162 東京都新宿区市谷八幡町８番地
電話(03)260－1161(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 仙台市六丁目字本新田東2番地の1	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 広島市安佐南区祇園町大字西原2249番地の1	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 仙台市六丁目字本新田東2番地の1	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 広島市安佐南区祇園町大字西原2249番地の1	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

4D 1002-I
TINSJ1165ACZZ