

7

第 7 章

RX-78が進化する

あなたはBS-BASICをほとんどマスターしてしまった。しかし、とっておきの命令というのがまだ残っている。それは、カセットやジョイスティック、あるいはプリンタと接続したときに使う命令だ。

そして、エラーを処理するというありがたい命令、さらには、RX-78の能力を数十倍も引き出してしまう機械語とのやりとりのための命令というのがある。

余裕を見せながら、最後までじっくりとやっていこう。

STEP 1 手際よくエラーを処理する

当然のことながら、プログラムにエラーがあると、RX-78はそこで止まってしまう。どんなに一所懸命に動かそうとしてももうダメ、どこでもうごいてはくれない。

ほんとにそうかな？

実は、プログラムにエラーがあっても、RX-78に動いてもらう方法があるんだ。

エラーには番号がある

エラーには、いろんなものがあるが、それぞれに番号が付いている。もっともよく現われる、文法上のまちがいを指摘してくれる Syntax

Error は、1番 (*Error 1)

READ-DATA文を使うとよく発生する Out of DATAは24番 (*Error 24)

というように、エラーメッセージのそれぞれに番号が付いているわけだ。(参考164ページ)

この番号によって、RX-78は、今どんなエラーがおこっているかを、教えてくれるんだ。

STEP 1 サンプルプログラム

```

10 REM ** エラー シヨリ サンプル **
20 PRINT CHR$(6)
30 ON ERROR GOTO 90
40 INPUT "X="; X
50 Y=SQR(X)
60 PRINT X;" ノ ハイホウコン ハ"
70 PRINT Y;" テス" :PRINT
80 GOTO 40
90 REM ** エラー ノ シヨリ **
100 IF [ERL=50]*[ERN=3] THEN 140
110 PRINT "ERL="; ERL
120 PRINT "ERN="; ERN
130 STOP
140 PRINT "フノ カズ ハ イケマセン"
150 RESUME 30

```

注釈文

画面に出ている内容をすべて消す。

エラーが発生したら90へとぶ。

Xを入力する。

Xの平方根をYに代入する。

Xの内容と「ノ ハイホウコン ハ」と画面に出す。

Yの内容と「デス」と画面に出し、行をかえる。

行番号40へとぶ。

注釈文

エラー行番号が50でエラーナンバーが3なら140へとぶ。

「ERL=」とERLの内容を画面に出す。

「ERN=」とERNの内容を画面に出す。

プログラムをストップする。

「フノ カズ ハ イケマセン」と画面に出す。

行番号30へプログラムの実行を移す。

〔解説〕

このプログラム、ちょっと見ると行番号80まででいいような気がする。しかしよく考えてみよう。行番号50のSQR関数、つまり平方根を求める数値関数だけど、この関数に負の数を与えたたらどうなるだろうか、たちまち *Error 3 in 50となってプログラムは止まってしまう。ところがこのプログラムだと、たとえばX=-120と入れてもエラーにならず、「フノ カズ ハ イケマセン」というメッセージが出て、プログラムは行番号40に戻る。

そして、もしこれ以外のエラーがおこってしまったならば、エラーの行 (ERL) とエラー番号を書き出すようにしてある。

```

X=6
6.4494897 デス
X=8
8.4320508 デス
X=5
5.236068 デス
X=-8
フノ カズ ハ イケマセン
X=4
4.1 ハイホウコン ハ
X=■

```

エラーが発生したら—ON ERROR GOTO

使用例

ON ERROR GOTO 1000

プログラム実行中にエラーが発生したら、行番号1000にジャンプせよという命令。

一般型式 ON ERROR GOTO 番号 ジャンプさせたい行

- ON ERROR GOTO命令よりあとでエラーがおこった場合プログラムは止まらずにこの命令で指定した行番号にジャンプし、その行以降を実行する。

さらに、IF ERN、IF ERLを使って、エラーの種類やエラーがおこった行番号などから、プログラムの中でエラーに対して処理をおこなっていくことができる。

```

LIST REM *** イラージュ サンプル 2 ***
10 PRINT CHR$(6):COLOR 4,1
20 ON ERROR GOTO 70
30 READ X,Y
40 READ X,Y
50 LINE POS X,POS Y,X,Y
60 GOTO 40
70 IF (ERL=40)*(ERN=24) THEN
80 PRINT "イラ- ERL=";ERL;"ERN=";
":ERN:GOTO 100
90 PRINT "テ-9 カ"タリマセン"
100 END
110 DATA 100,40,50,65,80,76,9
9,180,145
Ready

```

このプログラムは、*Error 24 in 40 というエラーになってしまふ。つまり、「データが足りません！」、というよくやってしまうエラーだ。

しかし、実際には*Error 24 in 40 なんて画面に出すに「データ ガ タリマセン」と出る。これが行番号30とその飛び先である行番号70~90の働きだ。これなら、エラー番号24は何だったのかと調べる必要がない。

④プログラムにエラーがなくなったら、これらのエラー処理命令は取りさっててしまう。ただし、INPUT、GET命令などを使ってキーボードから入れる数字や文字が原因でエラーになりそうな場合には、そのエラー処理命令はもちろん有効だ。

エラーがこわくない—— IF ERN, IF ERL

イフ

エラーナンバー

イフ

エラーライン

使用例**① IF ERN=44 THEN 1050**

エラー番号が44であれば、行番号
1050へジャンプせよという命令

② IF ERL=350 THEN 1090

エラー発生行番号が350であれば、
行番号1090へジャンプせよという
命令

一般型式**IF ERN**=エラー番号 **THEN** ジャンプさせ

たい行番号

IF ERL=エラーがおこった行番号**THEN** ジャンプさせたい行番号**説明**

●ERNは、エラーがおこったときの、そのエラー番号が入っているRX-78の（システム）関数で、ERLは、エラーがおこった行の行番号が入っているRX-78の（システム）関数である。

●IF文の使い方は今までとまったく同じで、=, <, >, +（論理和）、＊（論理積）が使える。

●ERN=1, ERL=150というように数字を入れることなどはできないが、ふつうの数値関数のようにPRINT命令で中身を書き出すことなどができる。

エラー処理は完成!

RESUME

使用例

① RESUME

エラーが発生した命令に再びもどる。

② RESUME NEXT

エラーが発生した命令の次の命令へもどる。

③ RESUME 400

行番号400へもどる。

一般型式

RESUME

もどる行番号

説明

- エラーが発生し、ON ERROR GOTOなどでプログラムの実行がエラー処理に移ったとき、エラー処理終了後、エラーの発生した行、またはその次の行、あるいは指定した行に戻すことができる。
- もどる行番号を指定しなかった場合には、エラーが発生した行にもどり、NEXTと指定した場合には、エラーが発生した次の行にもどる。
- もどり行番号を0にすると、プログラムの先頭にもどる（当然、先頭の行番号を指定してもよい）。

```

LIST REM ** RESUME サンプル ***
10 PRINT CHR$(63)
20 ON ERROR GOTO 70
30 INPUT "A=";A
40 B=1/A
50 PRINT B
60 REM ** エラー処理 ***
70 IF (ERL=500)*(ERN=2) THEN P
RINT "0 ÷ 0 です。":RESUME 40
80 END
Ready

```

これは $1/A$ を求めるプログラムだが、0で割るというのは、数学では定義されていないので、RX-78でも当然エラーが出る。エラー番号2というのはOverflow Errorである。したがって、0で割る場合は、別に考えてやらなければならない。ON ERROR GOTOによって、0が入力された場合は、“0ハダメ”と表示してやるようにする。しかしこのままでは先に進まないので、行番号40にもどるように、RESUME40としているのである。もちろん、どこに戻ればいいのか？と考えるよりも前に、RESUME0として、プログラムを先頭からやってもよい。

STEP 2 カセットは頭脳カプセル

さて、今までいろいろなコマンドや、ステートメントを勉強してきた。それらを使つたいろんなプログラムも作ったけど、サア電源を切ると、どうなるだろうか？そう、今まで苦労して作ったプログラムは、一瞬のうちに、なくなってしまう。もう一度電源を入れても、再びもどつてこない。さて、どうしたらいいのだろうか？もう一生電源が切れない！？何かいい方法はないだろうか？リストをメモしておく？それも一つの方法だ。何かに、リストをメモしてしまっておくのもいい。実は、もっといい方法がある。RX-78の中のプログラムをカセットテープにしまっておくことができるのである!! カセットテープにプログラムを記録しておけば、再び打ち込むこともなく、簡単にプログラムを使えるわけだ。ようするに、カセットテープは、RX-78の頭脳カプセルになるわけだ。



カセットにしまおう!

セーブ
SAVE

使用例

SAVE "A"

Aというプログラム名でプログラムをカセットテープに記録する。

一般型式

SAVE

"プログラムの名前"

説明

●RX-78の中に作ったプログラムをカセットテープに記録させる命令。

●SAVEのうしろのプログラム名は16文字以内。よって、プログラムには16文字以内の名前をつけておくとよい。

●一本のプログラムだけをセーブしておくなら、ファイル名はなくてもよい。

二本以上しまっておくときは、ファイル名によって、あとで呼び出すので必ずファイル名をつけよう。

プログラムをセーブする手順

①RX-78という名でプログラムをセーブするには、**SAVE"RX-78"** ①と打つ。

SAVE " RX-78 "

②カセットレコーダの録音ボタンを押した後、を押す。

③ただ今、記録中！

③
SAVE " RX-78 "
WRITING RX-78

④テープへのセーブ 完了!!

*カセットテープへのセーブが完了すると、ピピッと音がします。

④
SAVE " RX-78 "
WRITING RX-78
READY

テープのプログラム再生は LOAD

使用例

LOAD "A" RETURN Aというプログラム名のプログラムをカセットテープから読み出す。

一般型式

LOAD "プログラム名" RETURN

説明

- カセットテープに SAVE したプログラムを RX-78 に読み込む。
- カセットテープから、RX-78 に、プログラムをロードすると、今まで RX-78 に入っていたプログラムは無効、すなわち、消えてなくなってしまうので注意しよう。
- プログラム名がなくてもロードしてくれるが、1 本のテープにいくつものプログラムが入っているときは、プログラム名をつけて LOAD を命じると、その名前のプログラムを捜してロードしてくれる。

プログラムをロードする手順

① **LOAD "A"**

(ここでは、A というプログラム名のプログラムをロードする。)

② RETURN を押した後、カセットレコーダの再生ボタンを押す。

③ プログラム A よりも前にセーブされていた ZOO というプログラムがあり、その名前が出た。

④ プログラム A がみつかったので、RX-78 に LOAD をはじめた。

⑤ プログラム A のロード完了!!

* RX-78 へのロードが完了すると、ピピッと音がします。

① LOAD "A"

③ LOAD "A"
FOUND "ZOO"

④ LOAD "A"
FOUND "ZOO"
FOUND "A"
LOADING "A"

⑤ LOAD "A"
FOUND "ZOO"
FOUND "A"
LOADING "A"
READY

正しくセーブできたかな? —— VERIFY

ベリファイ

使用例

VERIFY "A"

RETURN

カセットテープにセーブされているAというプログラム(ファイル)名のプログラムと、RX-78が現在記憶しているプログラムが同じものであるかどうかを確かめる。

一般型式

VERIFY "プログラム名" RETURN

説明

●プログラムをカセットテープにセーブしたとき、正しくセーブできているかを確認するときに使う。つまり、現在RX-78が覚えているプログラムと、カセットテープにセーブしたプログラムの内容を比較して、同じかどうかを調べる命令だ。

プログラムをVERIFYする手順

①テープを巻きもどす。

②Aというプログラム名のプログラムをVERIFYする。

③RETURNを押した後、カセットレコーダの再生ボタンを押す。

④VERIFYしたいプログラムの前に別のプログラムがあれば、そのプログラム名(この場合ZOO)を出す。

⑤プログラムAをみつけたので、内容を比較している。

⑥確認が終了するとOKがでてプログラムは正しくセーブされたことを示す。

このとき、プログラムが正しくなかったならば、*Error70のチェックサム エラーというになる。

もう1度、セーブしなおそう!

*VERIFYが終るとピピッと音がします。

② VERIFY "A"

④ VERIFY "A"
FOUND "ZOO".

⑤ VERIFY "A"
FOUND "ZOO"
FOUND "A"
VERIFYING "A"

⑥ VERIFY "A"
FOUND "ZOO"
FOUND "A"
VERIFYING "A"
OK
READY

RX-78とテープをミックス — APPEND

アpend

使用例

APPEND "A"**RETURN**

今RX-78の中にあるプログラムと
テープにセーブしてあるAという
プログラムを、まぜ合わせてくれ
る。

一般型式

APPEND

"プログラム"

RETURN

説明

●RX-78に今Bという名前のプログラムを作ったとしよう。テープにセーブされているAという名前のプログラムといっしょにしたい、と思った時に活躍してくれるのが、APPENDである。

〔例〕

RX-78のプログラムB

```

10 X=10
20 X=20
30 Z=X+Y
50 PRINT Z
60 END
    
```

テープにセーブされたプログラムA

```

10 X=5
20 Y=10
40 C=X*Y
60 PRINT C
70 END
    
```

APPEND "A" とすると、どうなるか？

まず行番号10はAにもBにもある。そうすると、A（カセットテープのプログラム）の方が優先され、行番号10にはX=5が入る。番号20も同様。ところが、行番号30は、Bにしかないからそのままそれが残る。行番号40はAにしかないので、Aの行番号40が残り、行番号50、60、70についても同様、よって次のようなプログラムができる。

```

10 X=5
20 Y=10
30 Z=X+Y
40 C=X*Y
50 PRINT Z
60 PRINT C
70 END
    
```

このようにして、APPENDを用いてRX-78のプログラムと、カセットにセーブされているプログラムと一緒にすることができる。カセットにセーブしたプログラムに新しく、サブルーチンなどをくっつけるときに便利だ。

※APPENDが終るとピピッと音がします。

残りのメモリーのサイズは? —— SIZE

サイズ

使用例

PRINT SIZE

RETURN

BASICテキストエリアの未使用バ
イトを表示する。

一般型式 SIZE

説明 ●現在RX-78のBASICテキストエリアには、どれくらいのメモリーが残っているかを知りたいときに、SIZEを使って残りのバイト数を出す。

- RX-78のBASICテキストエリアは12KB（正確には、12104バイト）であり、1バイト1文字と考えてよい。つまり約12000文字分のプログラムが作れるというわけである。

④ モルフップ (160 °C まき) 奈照

STEP 3 プリンタと組めば楽しさ10倍

前にカセットテープにプログラムを記録する方法を勉強したが、

ほかにも記録の手段はある。

プリンタに、プログラムを打ち出させて残しておくという方法だ。

プリンタに記録するのと、カセットテープに記録するとの違いは、なんだろう。

まず

- ① 目に見える(プリンタ)のと見えない(カセット)。
- ② RX-78に打ち込まなくてはいけない(プリンタ)のとすぐロードできる(カセット)の違い。

が大きいものだ。

①について

目でリストを直接見ることができると、デバッグ(プログラムの間違いを直す作業)に便利だし、ほかの人にプログラムを見せて、理解してもらうことができる。さらには、RX-78のないところでも、プログラムを考えたり、なおしたりできる。

テープだと、直接目で見えるわけではないから、RX-78にロードしないかぎり、プログラムをいじったりすることは不可能だ。

②について

プリンタにリストを打ち出しておいても、再びRX-78で同じプログラムを実行したいときには、もう一度打ち込まなければならない。長いプログラムなどでは、ちょっと大変である。

ところが、カセットテープに記録しておけば、LOAD命令によって、すぐにプログラムをロードできるというわけだ。それぞれの場合においてより適切な方法をとるといい。

さて、APPEND(アペンド)命令についても、さきほど見ただけたように、行番号100は必ず最初にある。そうすると、A-100(アートデータ(プログラム)の行)が省略され、行番号101にはXから始まる行番号も出現ところが、行番号38とRにしかないので、Aを省略する。行番号38とRについても同様、よって次のようなプログラムができる。

```

10 APPEND
20 PRINT A
30 PRINT B
40 C=X+Y
50 PRINT Z
60 PRINT C
70 END

```

このようにしてAPPENDを利用してRX-78のプログラムと、カセットテープ上で組み合ったデータと一緒にできることができる。内蔵メモリにデータを組み合わせたり、データをオーバーライドなどをいつわけるときに便利だ。

STEP 3 サンプルプログラム

```

10 REM ** シュウエイロク **
20 PRINT CHR$(6)-----
30 INPUT "ナミエ? "; A$-----
40 INPUT "シユウエイ(1/2)? "; B1$-----
50 INPUT "シユウエイ(2/2)? "; B2$-----
60 INPUT "TEL NO.? "; C$-----
70 L1=LEN(A$)+3 :L2=LEN(B1$)+2 -----
80 L3=LEN(B2$)+2 :L4=LEN(C$)+2 -----
90 L=L1 -----
100 IF L<L2 THEN L=L2 -----
110 IF L<L3 THEN L=L3 -----
120 IF L<L4 THEN L=L4 -----
130 ST$=STRING$("+" ,L) -----
140 PRINT/P TAB(5);ST$ -----
150 PRINT/P -----
160 PRINT/P TAB(5);A$;" サマ" -----
170 PRINT/P TAB(7);B1$ -----
180 PRINT/P TAB(7);B2$ -----
190 PRINT/P SPACE$(L+7-L4);C$ -----
200 PRINT/P -----
210 PRINT/P TAB(5);ST$ -----
220 END

```

注釈文
 画面にでている内容をすべて消す。
 名前を入力する。
 住所 (1/2) を入力する。
 住所 (2/2) を入力する。
 電話番号を入力する。
 L 1 は名前の長さに 3 を加えた値である。
 L 2 は住所 (1/2) の長さに 2 を加えた値である。
 L 3 は住所 (2/2) の長さに 2 を加えた値である。
 L 4 は電話番号の長さに 2 を加えた値である。
 L は L 1 である。
 L が L 2 より小さいなら L は L 2 である。
 L が L 3 より小さいなら L は L 3 である。
 L が L 4 より小さいなら L は L 4 である。
 文字変数 ST\$ は "+" が L 個の変数である。
 プリンタを 5 文字分横にとばし、文字変数 ST\$ を書く。
 プリンタを改行する。
 プリンタを 5 文字分横にとばし、名前と " サマ " を書く。
 プリンタを 7 文字分横にとばし、住所 (1/2) を書く。
 プリンタを 7 文字分横にとばし住所 (2/2) を書く。
 プリンタに L + 7 - L4 個の空白と電話番号を書く。
 プリンタを改行する。
 プリンタを 5 文字分横にとばし、文字変数 ST\$ を書く。
 プログラムの終りを示す。

[解 説]

RX-78につないだプリンタから、名前、住所と電話番号を打ち出す(印字する)プログラムである。

プログラムのリストを見てくれるとわかると思うのだが、新しく出てきた命令は行番号140~210のPRINT/P。これをそっくりPRINTにかえれば画面に結果ができる。つまり、プリンタに結果をだすにはPRINTをPRINT/Pにするだけ。思ったより簡単みたいだ。

このプログラムを使って住所録を作ったり、手紙やハガキの宛名や自分の住所を書くなど、利用のしかたはいろいろだ。

* サンプルプログラム中に出てくる〔 〕はキーボード上で()と同じです。

プリンタに打て!—PRINT/P, LIST / P

使用例

① PRINT/P A [RETURN]

変数Aに入っているなかみを、プリンタに打ちだす(印字する)。

② PRINT/P [RETURN]

プリンタを1行分あける。

① LIST/P [RETURN]

RX-78が記憶しているプログラムのすべてを、行番号順にプリンタに打ちだす(印字する)。

② LIST/P 10-100 [RETURN]

行番号10から行番号100までのプログラムを、行番号順にプリンタに打ちだす(印字する)。

一般型式

PRINT/P 変数または式 [RETURN]

LIST/P はじまり行番号ーおわり行番号 [RETURN]

説明

●画面にだしていたのをプリンタに打ち出す(印字する)

命令で、PRINT、LISTと同じ使い方をする。

PRINT/P

●ストリング、数値をプリンタに打ちだす命令で、変数

は文字変数、数値変数のどちらでもよい。また、スト
リング、数式を直接書いてもよい。

●PRINT命令と、「;」と「,」の使い方をはじめとして、TAB

(59ページ)、USING(36ページ)の使い方も同じである。
また、PRINT/Pとだけ書くと、行を1行あける。

LIST/P

●RX-78が記憶しているプログラムを、行番号順にプリ
ンタに打ち出す。「はじまり行番号」や「おわり行番号」
を書くことによって、プログラムの見たいところだけ
をプリンタにだすこともできる。

●LIST命令と同じように、「はじまり行番号」、「おわり
行番号」をそれぞれ略すことができる。

```

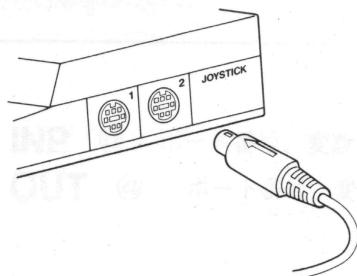
STEP 3
18 READ DATA FROM PORT A
28 PRINT CHR$(C6H)
38 INPUT "THIS IS ";B14
48 INPUT "IS IT? ";B14
58 INPUT "TEL NO. S ";C4
68 LISTLEN(C4)+3:=LEN(B14)+3
78 L=IF(L>3 THEN L=3
88 IF L>3 THEN L=3
98 IF L>3 THEN L=3
108 IF L>3 THEN L=3
118 IF L>3 THEN L=3
128 IF L>3 THEN L=3
138 STRINGS("L");L
148 PRINTTAB(ABC2)+3
158 PRINTTAB(TABC2)+3
168 PRINTTAB(TABC2)+3
178 PRINTTAB(TABC2)+3
188 PRINTTAB(SPACE(2)+3)
198 PRINTTAB(TABC2)+3
208 PRINTTAB(TABC2)+3
218 PRINTTAB(TABC2)+3
228 PRINTTAB(TABC2)+3
238 END

```

STEP 4 ジョイステイックはスポーツ感覚

RX-78には、2つのジョイステイックがつなげる。ジョイステイックを使えば、ゲームも本格的になってくるし、なんといっても操作が簡単なので、スピードも増し、スリリングになってくる。RX-78にジョイステイック、すなわち、スポーツ感覚になってくる。

ジョイステイックのつなぎ方



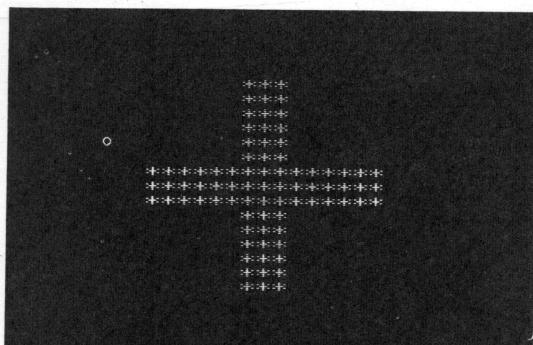
STEP 4 サンプルプログラム

```
10 REM ** ホシヲウコウカス **
20 PRINT CHR$(6)-----注釈文。
30 X=14 : Y=12-----画面に出ている内容をすべて消す。
40 CURSOR X, Y-----XとYを初期設定する。
50 PRINT "*"-----カーソルの位置を決める。
60 JOYS1 C-----「*」を画面に出す。
70 ON C GOSUB 140, 150, 160, 170, 180, 190, 200, 210-----ジョイステイックからのキーの入力を待つ。
80 IF X<0 THEN X=0
90 IF X>28 THEN X=28
100 IF Y<0 THEN Y=0
110 IF Y>21 THEN Y=21 }-----ジョイステイックのキーの入力により、サブルーチンへジャンプする。
120 GOTO 40-----画面から、はみ出ないようにする。
130 REM ** ホウコウヲキメル **
140 X=X+1 : MUSIC "C1" : RETURN
150 X=X+1 : Y=Y-1 : MUSIC "D1" : RETURN
160 Y=Y-1 : MUSIC "E1" : RETURN
170 X=X-1 : Y=Y-1 : MUSIC "F1" : RETURN
180 X=X-1 : MUSIC "G1" : RETURN
190 X=X-1 : Y=Y+1 : MUSIC "A1" : RETURN
200 Y=Y+1 : MUSIC "B1" : RETURN
210 X=X+1 : Y=Y+1 : MUSIC "C+1" : RETURN }-----行番号40へとぶ。
                                         -----注釈文。
                                         -----上下、左右、斜方向の移動を決め、音を出す。
```

〔解説〕

ジョイステイックを使って*を動かすばかりでなく、*の動く方向によって音楽を決めている。だから、簡単な曲だったら演奏だってできる。こんなはなれワザを持っている命令は、行番号60のJOYS1というたったこれだけの命令だ。もちろんこれだけで、全部がかたずいてしまうわけではないけど、やっぱり画期的だ。

ジョイステイックの1を操作して*の方向と音を決めているのが、行番号70~120と行番号130~210のサブルーチンだ。こんなに短いプログラムなのに本当に楽しくなってしまう。ジョイステイックを使うと、RX-78のおもしろさが急上昇する。



* サンプルプログラム中に出てくる〔 〕はキーボード上で()と同じです。

おもしろ急上昇! —— JOYS1, JOYS2

使用例

① JOYS1 A

1番のジョイスティックで押された数字（位置に対応した数字）を変数Aに入れる。

② JOYS2 B

2番のジョイスティックで押された数字（位置に対応した数字）を変数Bに入れる。

一般型式

JOYS1

数値変数……1番のジョイスティック

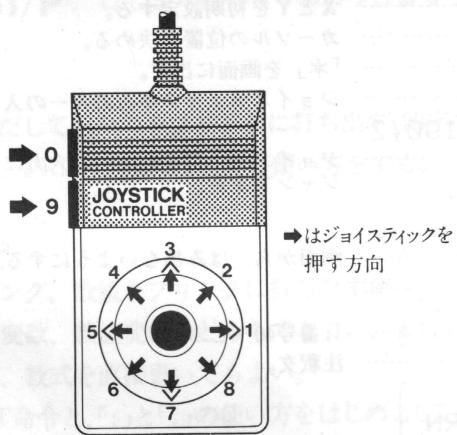
JOYS2

数値変数……2番のジョイスティック

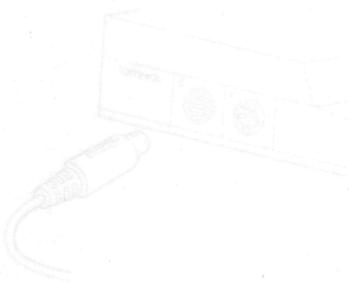
説明

●ジョイスティックで押された数字を変数に読み込む。

押された位置と数字との対応は下図のようになっている。
る。



●どの位置も押されていないと10が入り、入力の優先順位は、9、0、その他となる。



```

10 REM ** JOY サンプル ***
20 PRINT CHR$(6)
30 JOYS1 A
40 JOYS2 B
50 PRINT "JOYS1=";A,"JOYS2=";B
60 PRINT
70 FOR I=0 TO 1000 :NEXT I
80 GOTO 30

```

1番、2番それぞれジョイスティックのどの位置を押すと、どの数字が入るのかを画面に出してみた。
何にも押していないときは10の数字が入っているのがわかる。

JOYS1= 6	JOYS2= 6
JOYS1= 5	JOYS2= 10
JOYS1= 7	JOYS2= 8
JOYS1= 0	JOYS2= 5
JOYS1= 4	JOYS2= 0
JOYS1= 0	JOYS2= 0
JOYS1= 4	JOYS2= 9
JOYS1= 10	JOYS2= 6
JOYS1= 7	JOYS2= 8
JOYS1= 2	JOYS2= 8
JOYS1= 7	JOYS2= 2

* サンプルプログラム中に出てくる〔〕はキーボード上で〔〕と同じです。

出船、入船、I/Oポート— INP@, OUT@

使用例

① INP@12, A

I/Oポート番号12にあるデータを、変数Aに読み込む。

② OUT@13, B

変数Bの値を、I/Oポート13番に出力する。

一般型式 **INP @ ポート番号, 変数**

OUT @ ポート番号, 変数

説明 ●RX-78にジョイスティックをつないで数値を入れたり、プリンタをつないで結果を打ち出させるというように、いろんな機器を自由につなぐことによって、RX-78にデータを与えたり、逆にRX-78で制御する(あやつる)ことが可能である。

INP@

●I/Oポート番号を指定して、そのポート上にあるデータを変数に読み込む。

OUT@

●変数のデータを外部機器(デバイス)に与えるため、I/Oポート番号を指定して、そのポートにデータを送る。

STEP 5 データファイル入出力

カセットテープを使って、データのファイルを作つてみよう。知人の住所や電話番号などを RX-78 とカセットテープの組み合わせでしっかりとファイルする。

その他にも君のアイディアで色々なデータのファイルを作り君のデータ・バンクにしよう。

STEP 5 サンプルプログラム

```

10 REM ** データ ファイル ***
20 DIM AA$(2, 100)
30 PRINT CHR$(6):COLOR7, 1
40 CURSOR 2, 1:PRINT "** ADDRESS PROGRAM"
50 PRINT:PRINT TAB(6); "1 サクセイ 2 ヘンコウ"
60 PRINT TAB(6); "3 カキコミ 4 ヨミトリ"
70 PRINT TAB(6); "5 ミル 6 プリント"
80 PRINT:PRINT TAB(6); "ト レニ シマスカ [1-6]"
90 GET A$:A=VAL(A$)
100 ON A GOTO 120, 210, 310, 480, 650, 740
110 GOTO 90
120 REM ** ADDRESS ノ サクセイ ***
130 PRINT CHR$(6):PRINT "ADDRESSノサクセイ(オワリ=E, マチガイ=M)":PRINT: IY=I: I=0
140 I=I+1: IF I>100 THEN PRINT "100 ニン マテデス":GOTO 40
150 PRINT "NO. ";:PRINT USING "###"; I; " ";:INPUT "NAME "; AA$(0, I):AA$(0, I)=LEFT$(AA$(0, I), 14)
160 IF AA$(0, I)="E" THEN I=I-1:GOTO 30
170 IF AA$(0, I)="M" THEN I=IY:GOTO 30
180 INPUT "TEL. "; AA$(1, I):AA$(1, I)=LEFT$(AA$(1, I), 14)
190 INPUT "ADDRESS "; AA$(2, I):AA$(2, I)=LEFT$(AA$(2, I), 50)
200 PRINT:GOTO 140
210 REM ** ヘンコウ ***
220 PRINT CHR$(6):PRINT
230 PRINT "ADDRESS ノ ヘンコウ (オワリ=E)"
240 PRINT:INPUT "ヘンコウスル NO.= "; H$
250 IF H$="E" THEN 30

```

注釈文。
配列宣言。
画面に出ている内容をすべて消す。カラーコードを白、画面のバックの色を青にする。

(2, 1)の位置にカーソルを移動し、" *PROGRAM"と画面に出す。
改行し、左から6文字分とばし、" 1_サクセイ_2_ヘンコウ"と画面に出す。
左から6文字分とばし、" 3_カキコミ_4_ヨミトリ"と画面に出す。

左から6文字分とばし、" 5_ミル_6_プリント"と画面に出す。
改行し、左から6文字分とばし、" ドレニ_シマスカ_(1-6)"と画面に出す。
文字変数A \$の値をキーボードから読み込み、それを数値化して変数Aの値に代入する。
変数Aの値が1なら行番号120へ、2なら行番号210へ、3なら行番号310へ、4なら行番号480へ、5なら行番号650へ、6なら行番号740へプログラムの実行を移す。
行番号90へとぶ。
注釈文。

画面に出ている内容をすべて消す。" ADDRESSノサクセイ(オワリ=E, マチガイ=M)"と画面に出す。改行し、変数IYにIを、Iに0を代入する。

変数Iの値を1つふやし、もしIの値が100より大きいならば" 100 ニン マテデス"と画面に出し行番号40にとぶ。

" NO. "と3桁でIの値を画面に出し、名前を入力する。名前は左から14文字以内とする。
もし名前が"E"ならIの値を1つひいて、行番号30へとぶ。
もし名前が"M"なら変数IYの値を変数Iに代入し、行番号30へとぶ。

電話番号を入力する。電話番号は左から14文字以内とする。

住所を入力する。住所は左から50文字以内とする。
改行し、行番号140へとぶ。

注釈文。
画面に出ている内容をすべて消す。改行する。
" ADDRESS ノ ヘンコウ (オワリ=E)"と画面に出す。
改行し、変更する番号を入力する。
もし変更する番号が"E"なら行番号30へとぶ。

* サンプルプログラム中に出てくる〔 〕はキーボード上で〔 〕と同じです。

```

260 H=INT(VAL(H$)): IF [H=0]+[H>I] THEN
240 .....  

270 INPUT "NAME "; AA$(0,H): AA$(0,H)=LEFT$([AA$(0,H)], 14)
280 INPUT "TEL. "; AA$(1,H): AA$(1,H)=LEFT$([AA$(1,H)], 14)
290 INPUT "ADDRESS "; AA$(2,H): AA$(2,H)=LEFT$([AA$(2,H)], 50)
300 GOTO 240
310 REM ** カキコミ **
320 PRINT CHR$(6): PRINT
330 INPUT "WRITING NAME ? "; NA$
340 IF NA$="E" THEN 30
350 PRINT: PRINT "TAPE START"
360 FOR J=0 TO 1000: NEXT J
370 WOPEN/T NA$
380 PRINT/T I
390 FOR J=1 TO I
400 FOR K=0 TO 2
410 PRINT/T AA$(K,J)
420 NEXT K
430 NEXT J
440 CLOSE/T
450 PRINT: PRINT "カキコミ オワリ"
460 FOR J=0 TO 1000: NEXT J
470 GOTO 30
480 REM ** ヨミトリ **
490 PRINT CHR$(6): PRINT
500 INPUT "READING NAME ? "; NA$
510 IF NA$="E" THEN 30
520 PRINT: PRINT "TAPE START"
530 FOR J=0 TO 1000: NEXT J
540 ROPEN/T NA$
550 INPUT/T I
560 FOR J=1 TO I
570 FOR K=0 TO 2
580 INPUT/T AA$(K,J)
590 NEXT K
600 NEXT J
610 CLOSE/T
620 PRINT: PRINT "ヨミコミ オワリ"
630 FOR J=0 TO 1000: NEXT J
640 GOTO 30
650 REM ** ミル **
660 FOR J=1 TO I
670 PRINT USING "###"; J; "[NAME] "; AA$(0,J): PRINT "[TEL.] "; AA$(1,J)
680 PRINT " "; AA$(2,J)
690 USR($00D8): GETK$: IF K$=". " THEN 690
700 PRINT: NEXT J

```

変更する番号を数値化し変数Hに代入する。もし変数Hの値が0かIの値より大きいなら行番号240へとぶ。

変更する名前を入力する。変更する名前は左から14文字以内とする。

変更する電話番号を入力する。変更する電話番号は左から14文字以内とする。

変更する住所を入力する。変更する住所は左から50文字以内とする。

行番号240へとぶ。

注釈文。

画面に出てる内容をすべて消す。改行する。

テープに出力するデータ・ファイルの名前を入力する。もしデータ・ファイルの名前が"E"なら行番号30へとぶ。

改行する。"TAPE START"と画面に出す。

Jの値を0が1000まで変化させる。

カセットデータ・ファイルをオープンする。

変数Iの値をカセットテープに書き出す。

Jが1からIの間、行番号430までを繰り返す。

Kが0から2の間、行番号420までを繰り返す。

名前、電話番号、住所をカセットテープに書き出す。

行番号400の繰り返し命令の終りを示す。

行番号390の繰り返し命令の終りを示す。

オープンしているデータファイルをクローズする。

改行する。"カキコミ オワリ"と画面に出す。

Jの値を0から1000まで変化させる。

行番号30へとぶ。

注釈文。

画面に出てる内容をすべて消す。改行する。

テープから入力するデータ・ファイルの名前を入力する。

もしデータファイルの名前が"E"なら行番号30へとぶ。

改行する。"TAPE START"と画面に出す。

Jの値を0から1000まで変化させる。

データファイルをオープンする。

カセットテープのデータを変数Iに読み込む。

Jが1からIの間、行番号600までを繰り返す。

Kが0から2の間、行番号590までを繰り返す。

カセットテープのデータを名前電話番号、住所に読み込む。

行番号570の繰り返し命令の終りを示す。

行番号560の繰り返し命令の終りを示す。

オープンしているデータファイルをクローズする。

改行する。"ヨミコミ オワリ"と画面に出す。

Jの値を0から1000まで変化させる。

行番号30へとぶ。

注釈文。

Jが1からIの間、行番号700までを繰り返す。

Jの値、"[NAME]"、名前を画面に出す。"[TEL.]"、電話番号を画面に出す。

住所を画面に出す。

キーボードから連続的に読み出し可能にする。キーボードから1文字入力する。もしスペース・キーが押されていると行番号690へとぶ。

改行する。行番号630の繰り返し命令の終りを示す。

* サンプルプログラム中に出てくる〔〕はキーボード上で〔〕と同じです。