



Title	Basicによる1200BPSでのパソコンコンピューターの端末制御プログラム (t s s - y y) の作成
Author(s)	吉村, 幸雄
Citation	大阪大学大型計算機センターニュース. 1982, 47, p. 69-79
Version Type	VoR
URL	<a href="https://hdl.handle.net/11094/65548">https://hdl.handle.net/11094/65548</a>
rights	
Note	

*The University of Osaka Institutional Knowledge Archive : OUKA*

<https://ir.library.osaka-u.ac.jp/>

The University of Osaka

# Basic による 1200BPS でのパーソナルコンピューターの端末制御プログラム (tss - yy) の作成

四国女子大学

栄養生理学教室 吉村 幸雄

## 1. はじめに

パーソナルコンピューターのインテリジェント端末化の方法については、すでに多くの報告がある<sup>1-6)</sup>。それらの大部分は 300 BPS の速度によるものである。しかも、Basic 言語による 1200 BPS の速度でのインテリジェント端末化のプログラムは不可能で、一部機械語によらなければならぬだろうと報告している。

ところで、今年度の初頭になって 1200 BPS の音響カップラーが米国製ではあるが日本でも入手できるようになって、電々公社への手続きなしに端末制御のソフトウェアさえあれば、実に簡単に大型計算機との接続ができる。当研究室でも Recal-Vdic 社 (VI3412J) の音響カップラーを購入し、パーソナルコンピューターのインテリジェント端末化を試みた。

言語は機械語なしで Basic のみで可能と信じソフトウェアの作成を試みた。実の所、1200 BPS での端末制御ソフトウェアは無理だろうという事を知らず、音響カップラーを購入してしまったのが本音である。

使用したパーソナルコンピューターは Bubcom 80 であるが、このコンピューターは若干処理速度が遅く、多量のデータを受信する時はフロッピーディスクの書き込みができないなどの問題があつて悩んでいたところ、徳島大学医学部栄養生理学教室の PC 8800 を使う機会があったので、PC 8800 用のソフトウェアに変更してテストした所、期待した通り、十分使用できるので、このソフトウェアを公開します。この時変更した点は、両機種とも Micro-soft 系の Basic であったから、まったくキーワードが異なるのは RS 232C を定義する所だけであった。したがって他の Micro-soft 系の Basic を使用したパーソナルコンピューターであればわずかな変更で使用できると思う。さらに喜ばしいことであるが、PC 8800 及び Bubcom 80 ともに標準フロッピー構成であったので、ソース形式でセーブしたプログラム ( save "〇〇〇", A ) やデータはフロッピーディスクで互換がとれ、お互いのソフトウェアが共有できる。プログラムの作成にあたって、横浜国大の有澤博氏の作成したプログラム<sup>1)</sup>を無断ではあるが多くを利用させていただきましたことをおことわりし、ここに謝意を表します。又、徳島大学栄養生理学教室には PC 8800 を使用させていただき深く感謝します。

## 2. プログラムの概要及び使用法

### (1) アルゴリズムの改良点

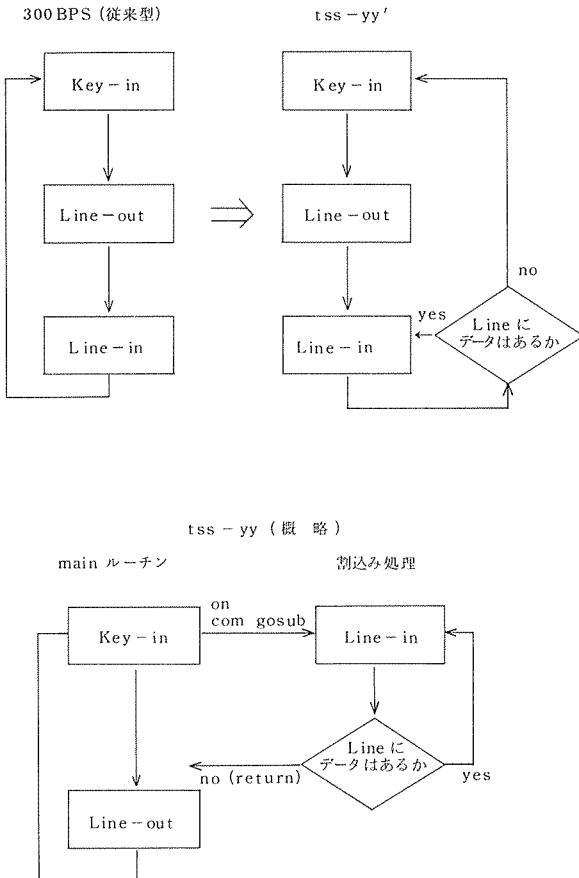


Fig. 1

すべての TSS 端末制御プログラムの基本的な部分は次の 3 点から構成される。

① 送信するデータをキーボードより入力するルーチン(Key-in)

② 送信ルーチン (Line-out)

③ 受信ルーチン (Line-in)

従来型 300 BPS のプログラム<sup>2-6)</sup>

は図 1 に示す通り 3 つのルーチンをループしている。この欠点は、たとえば次々と RS232C にデータが入力されると予想されてもプログラムの制御は必ず、キーボードの入力ルーチンに移り、ついで受信ルーチンにプログラムの制御が移るから、無駄なステップを踏む。

そこで図の右上のフローを見ればわかるように、RS232C にデータが入力されているかいないかを調べて、もしデータが存在すれば、受信ルーチンのみをループし、計算機は受信に専念できる。

しかしながら、上記の二点のアルゴリズムは基本的には 3 つのルーチンをループしている事には変りがない。このままでは困る事がある。例えば、Key-in ルーチンでプログラムの制御がとまつた時に、仮にデータが RS232C に入力されても、データは Line buffer overflow を起こすだろう。又、プログラムの拡張性がない（プログラムを大きくするとそれだけ一回りループするのに時間を多く消費する）。

そこで、tss - yy に見るように、メインルーチンと受信ルーチンをそれぞれ独立させ、並列処理が可能になるようにした。すなわち普通はプログラムの制御はメインルーチンにあるが、もしデータが RS232C に入力されると割込みされ、制御は受信ルーチンに移る。したがってメインルー

チングがどんなに長くても、又どんな仕事をしていても(ただし `input` 文の実行時は、割込みを受けつけない) プログラムの制御は受信ルーチンに移るというわけである。

## (2) プログラムの概要

普通の TSS 端末にもとめられる機能の上にパーソナルコンピューターの持つインテリジェンスを追加した。

- 1) パーソナルコンピューターの持つ画面編集機能が使用できる。
- 2) オフラインの状態(もちろんオンラインでも可)で送信すべきデータ、プログラムを前もって編集し、それをたくわえて、(本プログラムではテキストと呼ぶ)後に一括して送信する。

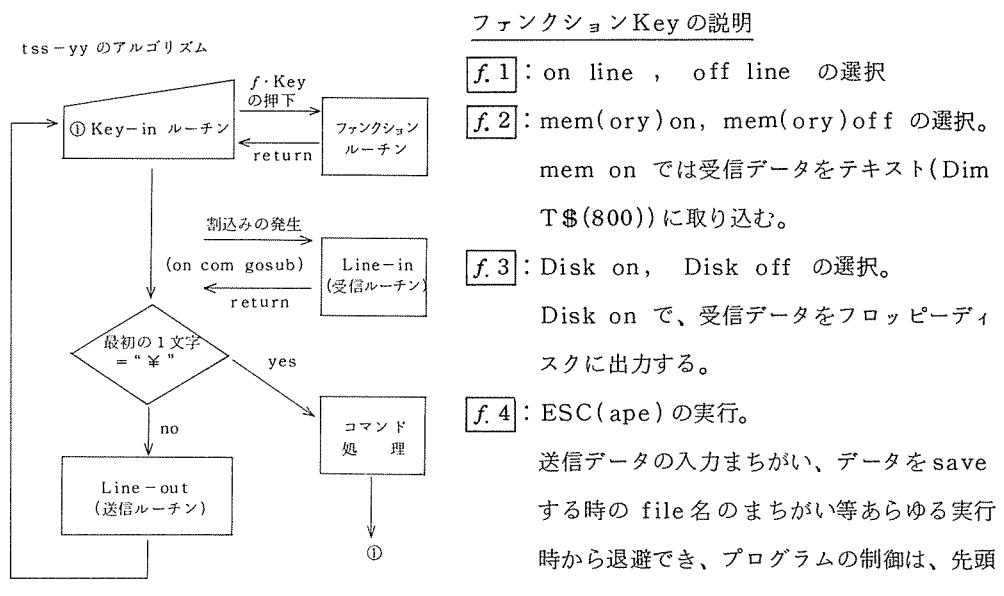


Fig. 2

f. 6 : Di(sk)  $\rightarrow$  TX Disk にある File のデータをテキストに入力する。

f. 7 : no LP, LP の選択。 LP を選択すると、この時点から会話のデータや File の dump、テキストの List を Line printer に出力する。

f. 8 : Dump 指定した File を CRT, printer に出力する。

f. 9 : end プログラムの終了。

f. 10 : ID と pass word をホストに送信する。

次に実行時の実際の画面を下図に示します。

```
*RUN TLIBH/PMLTRG,R
DO YOU WANT TO USE A FILE FOR INPUT(YES OR NO)
= NO
ENTER NUMBER OF VARIABLES (N) AND NUMBER OF OBSERVATIONS (M)
= 2,10
ENTER DATA...X(1,1),X(1,2),X(1,3),...,X(1,M),Y(1)
           X(2,1),X(2,2),X(2,3),...,X(2,M),Y(2)
N...NUMBER OF OBSERVATIONS
M...NUMBER OF VARIABLES
= $TE
1 = 1,2,3
2 = 2,3,4
3 = 3,4,5
4 = 5,6,7,10
5 = 10,20,30
6 = 6
key in デキマス TEXT ハ ?          text 5  receiv 167  send 7
1:on line 2:mem off 3:disk off 4:esc 5:brake 6:Di->tx 7:naLP 8:dump 9:end
```

Fig. 3

日本電気提供の TSS

/ LIB-6

統計計算編の重回帰分析プログラムを起動した所である (TLIBH / PMLTRG, R)。

次に観測データの入力要求があり、\$TE コマンドを使いそのデータを端末側のテキストに作成中である (実

際はプログラムを起動する前に、テキストにデータを作成してからホストに送信するのが望ましい)。 CRTの24、25行がTSSの状態を表示する領域です。25行にf.1～f.9の状態を表示し、一目で確認できます。24行には、例えば現在の状態すなわち TEXT作成中及びKey in できる状態(“Key in デキマス”が表示されていないとKey in はできません)を表示しています。さらに TEXTは5行つくられており、今は6行目を作成中です。その横にはカウントした受信データ(receive) および送信データ(send)の量を表示します。

#### コマンドの説明

Key入力したデータの先頭の文字が“\$”であれば送信データでなくコマンドと解釈します。

\$TE (XT) or \$TX

これを入力するとテキスト作成モードになり、レコード数を先頭に表示し、“=”以降にデータを入力します。又このTEXTモードは画面編集機能を利用でき、すでにつくられたTEXTを訂正できます。例えば、TEXTの2行目の2,3,4を2,3,1と変更する場合、カーソルを上に移動させ2,3,1と訂正しリターンキーを入力すれば2行目が2,3,1と訂正できます。もしこの時“2=”を消してリターンキーを入力すれば、行番号が見当らないので、現在作成中の行番号、すなわち6行目にデータが入力されます。

\$DE (DELETE) or \$DE n<sub>1</sub>, n<sub>2</sub>, n<sub>3</sub> or \$DE n<sub>1</sub> - n<sub>2</sub>

テキスト行の削除。\$DEでは、削除すべき行番号を次に聞いてきます。“,”で区切ると n<sub>1</sub>, n<sub>2</sub>, n<sub>3</sub> 行のみ削除、\$DE n<sub>1</sub> - n<sub>2</sub>と入力すると n<sub>1</sub> 行から n<sub>2</sub> まですべて削除します。

¥IN(SERT) or ¥IN1 or ¥IN1.2

¥IN を入力すると insert する行番号を次に聞いてきます。上記の右 2 つの命令はいずれも 1 行目から 2 行目の間に TEXT を 1 行追加します。

¥ED( IT ) or ¥EDn

先の ¥TE でデータの訂正が十分可能だが画面に表示されないテキストは訂正できないので、 ¥EDn と行番号を指示してやると、その行を表示して訂正可能になる。ただしその行のみの訂正である。

¥LI or ¥LIN

¥LI と入力すれば、テキストを最初から終りまで表示する。途中表示を中止する場合は、 f.4 (ESC) を押す。¥LIN と行番号も同時に入力するとその指定した行から 21 行のみを表示する。

¥ER(ASE)

TEXT のデータをすべて消去。

¥LP

これは、 f.7 (LP) と少し異なり、 TEXT のみの printer への出力。

¥SE(ND)

TEXT あるいは Disk の File の送信。一括送信も (“READ FDD” コマンドを入力) も可能である。

¥SA(VE)

TEXT data の Disk File への save。

¥LO(AD)

Disk file のデータをテキストを入力する。一部のデータのみ入力する場合はレコード番号を指定しなければならない。したがって、 file を dump して行番号を知る必要がある。

¥DU(MP)

Disk file のデータの出力。

¥EN(D)

プログラムの終了。

プログラム List その1

```

10 WIDTH 80,25
20 CONSOLE 0,25,0,1
30 DEFINT C,F,U,V,I
40 C2=400:DIM T$(C2)
50 ON COM GOSUB 230:ON KEY GOSUB 500,470,420,540,620,1740,580,1850,1930,660
60 ON ERROR GOTO 710
70 CLS:CONSOLE 0,23
80 COLOR 4:LOCATE 0,24:PRINT '1:on line 2:mem off 3:disk off 4:esc 5:brake 6
:Di->tx 7:noLP 8:dump 9:end':COLOR 7
90 VL=0:VT=0:VR=0:F1=0:F2=0:F3=0:F7=0
100 LOCATE 47,23:PRINT 'text 0':LOCATE 70,23:PRINT 'send 0'
110 LOCATE 57,23:PRINT 'receive 0'
120 KEY ON:COM ON
130 LOCATE 10,18:PRINT '— tss-yy ( PC <--> ACOS1000 ) by yoshimura v1.0 —
140 GOSUB 510:' RS232C on

150 ' main loop —
160 LOCATE 0,22
170 COLOR 7
180 GOSUB 290:' key-in sub
190 IF LEFT$(TA$,1)='%' THEN 780 ELSE GOSUB 370
200 LOCATE 51,23:PRINT VT;:LOCATE 63,23:PRINT VR;
210 GOTO 160

220 ' line-in sub —
230 COLOR 5:COM OFF
240 IP=PORT(1):VR=VR+1:IF IP THEN B$=INPUT$(IP,#1):PRINT B$;:ELSE COM ON:RETURN
250 IF F3 THEN PRINT #3,B$;
260 IF F2 THEN T$(VT)=B$:VT=VT+1
270 GOTO 240

280 ' key in -sub rem —
290 TA$="":LOCATE 0,23:PRINT '— key in テキマス —';:LOCATE 0,22
300 A$=INKEY$:IF A$="." THEN 300
310 IF A$=CHR$(&H1E) THEN LINE INPUT TA$:TA$=TA$+CHR$(&HD):GOTO 350
320 IF A$=CHR$(&H1C) OR A$=CHR$(&H1F) THEN 300
330 IF A$=CHR$(29) THEN PRINT CHR$(29);';CHR$(29);:TA$=LEFT$(TA$,LEN(TA$)-1):
GOTO 300
340 TA$=TA$+A$:PRINT A$;:IF CHR$(&HD)<>A$ THEN 300
350 LOCATE 0,23:PRINT SPC(18);:RETURN

360 ' line-out sub —
370 IF F7 THEN LPRINT 'S: ':TA$
380 IF F1=0 THEN 400
390 VS=VS+1:LOCATE 74,23:PRINT VS;:LOCATE 0,22:PRINT #1,TA$;:RETURN
400 LOCATE 0,22:PRINT 'not online':RETURN

410 ' f.3 disk on sub —
420 F3=1-F3:I3=1
430 LOCATE 23,24:IF F3 THEN COLOR 5:PRINT 'disk on ':ELSE COLOR 4:PRINT 'disk of
f';:CLOSE 3
440 LOCATE 0,22:IF F3 THEN INPUT '** save file (line ---> disk) ** input unit, f
ile name: ':UNIT$,NA$:OPEN UNIT$+':'+NA$ FOR APPEND AS #3
450 I3=0:GOTO 650

```

プログラム List その2

```

460 ' f.2 memory on sub ——
470 F2=1-F2
480 LOCATE 13,24:IF F2 THEN COLOR 5:PRINT 'mem on ' ;ELSE COLOR 4:PRINT 'mem off'
;
490 GOTO 650

500 ' f.1 sub ——
510 F1=1-F1:REM IF F1 THEN OPEN 'COM:E71 S' AS #1
520 LOCATE 2,24:IF F1 THEN COLOR 5:PRINT 'on line ' ;ELSE COLOR 4:PRINT 'offline'
;
530 GOTO 650

540 ' f.4 escape ——
550 LOCATE 0,22:PRINT '** escaped **'
560 CLOSE 2:IF I3 THEN I3=0:F3=1:GOSUB 420
570 GOTO 565

580 ' f.7 hard copy ——
590 BEEP:F7=1-F7
600 LOCATE 61,24:IF F7 THEN COLOR 5:PRINT 'LP ' ;ELSE COLOR 4:PRINT 'noLP';
610 GOTO 650

620 ' f. 5 brake out ——
630 IF F1 THEN BEEP:FOR I=1 TO 80:OUT &H21,&H3F:NEXT I:OUT &H21,&H37
640 RETURN
650 LOCATE 0,22:COLOR 7:RETURN

660 ' f.10 send ID no & passw. ——
670 TA$='6700252545$password'+CHR$(&HD)
680 GOSUB 370
690 RETURN

700 ' error routine ——
710 COM OFF:KEY ON:IF ERR=23 THEN OPEN 'COM:E71 S' AS #1:COM ON:RESUME 160
720 COM ON:IF ERR=5 THEN RESUME 160
730 IF ERR=52 AND ERL=440 THEN OPEN UNIT$+":":+NA$ FOR OUTPUT AS #1:RESUME NEXT
740 IF ERR=53 THEN PRINT '** error ** file not found ':RESUME 160
750 IF ERR=8 THEN RESUME 160
760 LOCATE 0,22:PRINT 'system error';ERR;ERL
770 RESUME 160

780 LO$=MID$(TA$,2,2)
790 IF LO$='TE' OR LO$='TX' THEN 850 ELSE IF LO$='DE' THEN 1070 ELSE IF LO$='IN'
    THEN 960
800 IF LO$='ER' THEN 1640 ELSE IF LO$='LI' THEN 1500 ELSE IF LO$='ED' THEN 1220
810 IF LO$='DU' THEN 1850 ELSE IF LO$='LP' THEN 1590 ELSE IF LO$='SE' THEN 1290
820 IF LO$='SA' THEN 1680 ELSE IF LO$='LO' THEN 1680 ELSE IF LO$='EN' THEN 1930
830 LOCATE 0,22:PRINT 'command invalid'
840 GOTO 160

850 ' text #TX ——
860 LOCATE 0,23:PRINT '— key in テキマス — TEXT \ ?'
870 LOCATE 0,22:PRINT VT+1;
880 IF VT>C2 THEN 930
890 COLOR 4:PRINT CHR$(224);
900 COLOR 7:LINE INPUT A$:IF A$="" THEN 950 ELSE I=INSTR(A$,CHR$(224))
910 IF I THEN VL=VAL(LEFT$(A$,I-1)):A$=MID$(A$,I+1):T$(VL-1)=A$ ELSE 940
920 IF VL>0 AND VL<VT+1 THEN LOCATE 0,CSRLIN-1:PRINT SPC(79);:ELSE 940
930 LOCATE 0,CSRLIN:PRINT VL;CHR$(224);:PRINT A$:LOCATE 0,CSRLIN:GOTO 900
940 VL=0:T$(VT)=A$:VT=VT+1:LOCATE 51,23:PRINT VT;:LOCATE 0,22:PRINT VT+1;:
    GOTO 880
950 LOCATE 20,23:PRINT SPC(18);:GOTO 160

```

プログラム List その3

```

960 ' INSERT $IN _____
970 IF LEN(TA$)=4 THEN LOCATE 20,23:PRINT 'INSERT LINE \ ?':GOSUB 290 ELSE VL=
980 FIX(VAL(MID$(TA$,4))):GOTO 990
980 VL=FIX(VAL(TA$)):IF TA$=CHR$(&HD) THEN 1060
990 LOCATE 6,22:PRINT 'line '+CHR$(224);:IF VL>C2 THEN VL=C2
1000 IF VT+1>C2 THEN 1050
1010 LOCATE 12,22:LINE INPUT A$;I=INSTR(A$,CHR$(224)):IF I THEN A$=MID$(A$,I+1)
1020 IF CSRLIN<22 THEN LOCATE 10,22:PRINT A$
1030 FOR IN=VT TO VL STEP -1:T$(IN+1)=T$(IN):NEXT IN:T$(VL)=A$
1040 VL=VL+1:VT=VT+1:LOCATE 51,23:PRINT VT;:TA$='':GOTO 1060
1050 LOCATE 0,22:PRINT 'buffer full';
1060 LOCATE 20,23:PRINT SPC(15);:GOTO 160

1070 ' DELETE $DE _____
1080 IF LEN(TA$)=4 THEN LOCATE 20,23:PRINT 'DELETE LINE \ ?':GOSUB 290:VL=VAL(
1080 TA$)-1:T$(VL)='':GOTO 1140
1090 I1=4
1100 I2=INSTR(I1,TA$,','):IF I2 THEN T$(VAL(MID$(TA$,I1,I2-I1))-1)='':I1=I2+1:
1100 GOTO 1100
1110 T$(VAL(MID$(TA$,I1))-1)='':I1=4
1120 I2=INSTR(I1,TA$,','):IF I2 THEN VS=VAL(MID$(TA$,I1,I2-I1))-1:VL=VAL(MID$(TA
1120 $,I2+1))-1:FOR I=VS TO VL:T$(I)='':NEXT I:ELSE T$(VAL(MID$(TA$,4))-1)=''
1130 IF VL>C2 THEN VL=C2
1140 I=0:I1=0:K=0:I2=0
1150 IF T$(I)='':THEN I=I+1:I2=I2+1:ELSE 1170
1160 IF I2<VT THEN 1150
1170 T$(I1)=T$(I):I1=I1+1:I=I+1
1180 IF I<VT THEN 1150
1190 IF T$(VT-1)='':THEN I1=I1-1
1200 LOCATE 19,23:PRINT SPC(17);
1210 VT=I1:LOCATE 51,23:PRINT VT:GOTO 160

1220 ' edit $ED _____
1230 IF LEN(TA$)=4 THEN LOCATE 20,23:PRINT 'edit line \ ?':GOSUB 290 ELSE VL=
1230 VAL(MID$(TA$,4))-1:GOTO 1260
1240 IF TA$=CHR$(&HD) THEN LOCATE 20,23:PRINT SPC(13);:GOTO 160
1250 VL=VAL(TA$)-1:LOCATE 5,22
1260 LOCATE 0,22:PRINT VL+1;CHR$(224);T$(VL);
1270 LOCATE LEN(STR$(VL))+2,CSRLIN:LINE INPUT T$(VL):I=INSTR(T$(VL),CHR$(224)):
1270 IF I THEN T$(VL)=MID$(T$(VL),I+1)
1280 LOCATE 20,23:PRINT SPC(15):GOTO 160

1290 ' SEND $SE _____
1300 KEY ON:LOCATE 0,22:INPUT 'send data \ text (0) or file (1) ',I1
1310 IF I1 THEN INPUT 'input unit, & file name: ',UNIT$,NA$:OPEN UNIT$+'+'+NA$ F
1310 OR INPUT AS #2
1320 IF I1=0 THEN LOCATE 0,22:INPUT 'send data \ 1 record ツ'カ (0) or 1 ド'ニカ
1320 (1)';I2
1330 IF I1 OR I2 THEN LOCATE 0,22:PRINT CHR$(&H22);"READ FDD";CHR$(&H22);'ト ソウシ
1330 ナセヨ -----> READY ツナレハ" ソウシカノウ":GOSUB 290:GOSUB 370
1340 LOCATE 5,23:PRINT '** OK ? press any key **';
1350 IF I1 OR I2 THEN A$=INKEY$:IF A$='':THEN 1350
1360 IF I1 THEN 1440
1370 LOCATE 0,22
1380 FOR I=0 TO VT-1
1390 IF I2=0 THEN A$=INKEY$:IF A$='':THEN GOTO 1390
1400 PRINT T$(I)
1410 IF I2 THEN TA$=T$(I)+CHR$(&HA) ELSE TA$=T$(I)+CHR$(&HD)
1420 GOSUB 370
1430 NEXT I:GOTO 1470
1440 WHILE NOT(EOF(2))
1450 LINE INPUT #2,TA$:TA$=TA$+CHR$(&HD):GOSUB 370
1460 WEND:CLOSE2
1470 IF I1 OR I2 THEN TA$="$$EOF":GOSUB 370
1480 LOCATE 5,23:PRINT SPC(13)
1490 GOTO 160

```

プログラム List その4

```

1500 ' LIST #LI _____
1510 IF LEN(TA$)=4 THEN VS=0:VL=VT-1:GOTO 1530
1520 VS=VAL(MID$(TA$,4))-1:IF VT>VS+22 THEN VL=VS+22 ELSE VL=VT-1
1530 PRINT:LOCATE 0,22
1540 FOR I=VS TO VL
1550 COLOR 4:PRINT I+1;CHR$(224);:COLOR 7:PRINT T$(I)
1560 IF F7 THEN LPRINT I+1;':';T$(I)
1570 NEXT I
1580 LOCATE 20,23:PRINT SPC(14);:GOTO 160

1590 ' output line printer #LP _____
1600 FOR I=0 TO VT-1
1610 LPRINT I+1;CHR$(224);T$(I)
1620 NEXT I
1630 PRINT:GOTO 160

1640 ' text clear #ER _____
1650 VT=0:LOCATE 51,23:PRINT VT;':ERASE T$:DIM T$(C2)
1660 GOTO 160

1670 ' PF 6 SUB DATA SAVE #SA _____
1680 KEY ON:LOCATE 0,22:INPUT '** TEXT / SAVE ** unit, File name, saved file is
new(1) or old(0)';UNIT$,NA$,I
1690 IF I THEN OPEN UNIT$+':'+NA$ FOR OUTPUT AS #2 ELSE OPEN UNIT$+':'+ NA$ FOR
APPEND AS #2
1700 FOR I=0 TO VT-1
1710 PRINT #2,T$(I):IF F7 THEN LPRINT T$(I)
1720 NEXT I:CLOSE 2
1730 GOTO 160

1740 ' f.6 or #LO disk file --->text _____
1750 KEY ON:LOCATE 0,22:INPUT '** file -> text ** unit, file name, data are all(
1) or part(0)';UNIT$,NA$,I
1760 IF I=0 THEN INPUT 'input start record, end record';I1,I2 ELSE I1=0:I2=0
1770 I=0:OPEN UNIT$+':'+NA$ FOR INPUT AS #2
1780 WHILE NOT EOF(2)
1790 LINE INPUT #2,A$:I=I+1:IF I1<=I THEN T$(VT)=A$:VT=VT+1:IF F7 THEN LPRINT A$
1800 IF I2=I THEN 1820
1810 WEND
1820 CLOSE 2
1830 LOCATE 51,23:PRINT VT;':GOTO 160

1840 ' file dump f.8 or #DU _____
1850 KEY ON:LOCATE 0,22:INPUT '** file dump ** input unit, file name';UNIT$,NA$
1860 OPEN UNIT$+':'+NA$ FOR INPUT AS #2
1870 I=0:WHILE NOT(EOF(2))
1880 I=I+1:LINE INPUT #2,A$:COLOR 4:PRINT I;CHR$(224);:COLOR 7:PRINT A$
1890 IF F7 THEN LPRINT I;CHR$(224);A$
1900 WEND
1910 PRINT '** data end **':CLOSE 2:GOTO 160

1920 ' program end #EN _____
1930 KEY OFF:KEY (1) OFF:CLOSE:CONSOLE 0,25:CLS.
1940 LOCATE 20,15:PRINT '_____ JOB END _____'.
1950 CLOSE:END

```

### (3) プログラム List の説明

変数 T\$(C2) : テキストを入力するディメンション。C2はその大きさ。

VT : テキストに入力された行数。ただし、260行の memory on にした時に入力されるテキストの1行は、ホストから送信した1論理レコードでなく、Line にたまたま存在したコードである。通常は端末の速度の方がホストの送信速度よりも早いため、1論理レコードがいくつもに切斷されてテキストに入力されてしまうので、この場合の利用価値は少ない。これを避ける方法は2点ある。1つには、CRコードがLineに入力するまで待ちつづけて所定の変数に取り込む Line input #1, A\$を使用する方法である。この文が使用できれば非常に都合が良いが、残念ながら使えない。次の方法はLineより取り出した文字列を加工する。この方法もBubcom80にとっては難しい。

VR : 受信されたレコード数。上記の所で述べた理由により、論理レコード数と一致しない。

60行 on com gosub 260 は割込み先の定義。on Key gosub 510, …… は f. Key の押された時の分岐先を定義。

150-210行 メインループ。Key-in ルーチンは gosub 290-350行である。リターンコードまでを1レコードとして、先頭の文字が“¥”でなければ(190行)、Line-out ルーチン(gosub 370-400)に分岐する。

220-270行 Line-in ルーチン。230行の com off は必要である。

Lineから入力する方法は ① input \$(n, #1)  
② Line input #1, A\$  
③ input #1, A\$ の3点ある。

③は“，”までのデータをA\$に読み込む。ただしこのTSSでは使用できない。ところで受信データは下図のようになっている。

	6×DEL	LF	0~4×DEL	データ	CR	6×DEL	
--	-------	----	---------	-----	----	-------	--

データは、先頭にDELコード群、末尾にCRコードを付加しているのが通常である。そして次のレコードまでにはわずかなインターバルがある。したがって Line input 文を使用すれば、受信データと歩調がとれ十分その速さに追従でき、後の処理も容易になる。ところが、残念な事に最終レコードのみCRコードをホストが送出していないのでこの文も使用できない。

410-690行 各ファンクションの設定。F1, F2, F3, F7の値が1の時はいずれも on の状態になる。f.4はESC(ape)処理。570行の goto 565 は飛び先が存在しないのでエラーが発生する。つまり故意にエラーを発生させ(ERR=8, 750行を見よ)、ある状態から抜けだそうという訳である。

### 3. 最 後 に

1200 BPS の通信速度は 300 BPS のものと比べ、すこぶる使用感が良く、画面を見つめて待つという動作がなくなった。これからますます TSS の拡大とあいまって 1200BPS に対する要望も強くなると思う。

本プログラムを作るにあたって気がついたことだが、端末側のパーソナルコンピューターはまだまだ非力であるにもかかわらず、ホストからのデータは選択の余地もなしに一方的に送られてきて、これを受ける端末側が自分のものにデータを加工しなければならないことである。例えば DEL コードは 1 レコードに 6 から 12 個と数もきままに付加されて送出されるが、TSS の初期に使われてきた CRT なしのミニプリンター形式のものにはそれは必要欠くべからずのものだが、CRT 表示を主体としたパーソナルコンピューターにはまったく不必要なものでホストとのコミュニケーションに障害さえもたらす。これらを処理する事は端末側にとって非常に負担で余裕のあるコミュニケーションができない。そこで、ホストの大型計算機にとってその能力は余るものがあるはずであるから、もう少し端末の面倒を見てもらう事を期待したい。

#### 〔参考文献〕

- 1) 有澤 博: 「マイコンの新しい使い方」 bit Vol.14, No.2, p. 111, 1981.
- 2) 紺野義仁, 森川 修: 「TSSによるPC-8001を使ったデータ処理(1)」  
東京大学大型計算機センターニュース Vol.14, No.4, p. 61, 1982.
- 3) 紺野義仁, 森川 修: 「TSSによるPC-8001を使ったデータ処理(2)」  
東京大学大型計算機センターニュース Vol.14, No.5, p. 123, 1982.
- 4) 酒井 聰, 小岩昌宏: 「マイコンと大型計算機をつなぐ」 SENAC  
SENAC Vol.14, No.2, p. 22, 1981.
- 5) 西本史雄, 石田晴久: 「マイコン PC 8001 と IF-800 を TSS 端末にするための BASIC プログラム」 東京大学大型計算機センターニュース Vol.13, No.4, p. 25, 1981.
- 6) 藤井 博: 「パーソナル・コンピュータ PC 8801 を用いたインテリジェント・ターミナル」  
大阪大学大型計算機センターニュース Vol.12, No.2, p. 83, 1982.
- 7) ACOS シリーズ77 「レベル0 手順通信制御仕様書」 日本電気 1979年3月