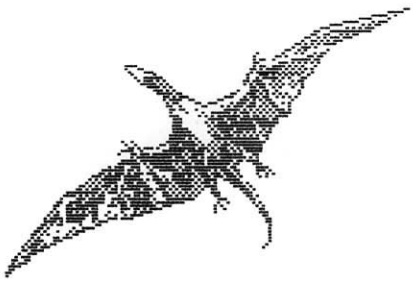
PROGRAM BITEN 93-1



INNEHÅLL

ISSN 0281-1146

KALLELSE ÅRSMÖTE

Medlemmar i föreningen Programbiten kallas härmed till Årsmöte Lördagen 6 mars 1993 kl.13: hos Kent Edgardh, Albatrossvägen 76, 1 tr.ned, Haninge

Förslag till dagordning:

1. Mötet öppnas

2. Val för mötet av ordförande, sekreterare och två justeringsmän

3. Mötets behörighet
   - utlysning
   - röstberättigade närvarande

4. Beslut om dagordning

5. Styrelsens årsredovisning för 1992. Verksamhetsberättelse och kassarapport delas ut på mötet.

6. Revisorernas berättelse för 1992

7. Om ansvarsfrihet för styrelsen för 1992

8. Val av styrelse för 1993. Årsmötet väljer enligt stadgarna en styrelse bestående av ordförande och kassör, vilka samtliga utses till befattning på årsmötet, samt därtill tre och högst sju övriga ledamöter.

9. Val av revisorer för 1993 (två revisorer och en suppleant).

10. Val av valberedning (två ledamöter varav en sammankallande).

11. Om årsavgift 1993.

12. Årsmötets avslutas. ■

MYARC MOVE DSK-FIL

av Jan Alexandersson

Myarc DM V med HFDC fungerar inte med MOVE av en fil från ROOT till SUBDIR eller SUBDIR till SUBDIR inom samma flexskiva. Filen flyttas ordentligt men sektor 0 får inte de nya sektorerna markerade som upptagna medan de gamla sektorerna frigöres på rätt sätt. Om du försöker att lagra ytterligare filer så kommer den flyttade filen att skrivas över. Motsvarande flyttning mellan olika SUBDIR på samma hårddisk fungerar dock utan problem. ■

Redaktör: Jan Alexandersson
Medlemsregister: Claes Schibler
Programbankir: Börje Häll

Föreningens adress:
Föreningen Programbiten
c/o Schibler
Wahlbergsgatan 9 NB
S-121 46 JOHANNESHOV, Sverige

Postgiro 19 83 00-6
Medlemsavgiften för 1992 är 120:-

Datainspektionens licens-nr 82100488

Annonser, insatta av enskild medlem (ej företag), som gäller försäljning av moduler eller andra tillbehör i enstaka exemplar är gratis.

Övriga annonser kostar 200 kr för hel sida. Föreningen förbehåller sig rätten att avböja annonser.

Föreningens tillbehörsförsäljning:
Följande tillbehör finns att köpa genom att motsvarande belopp insätts på postgiro 19 83 00-6 (porto ingår)

Användartips med Mini Memory	20:-
Nittinian T-tröja	40:-
99er mag. 12/82, 1-5,7-9/83(st)	40:-
Nittinian årgång 1983	50:-
Programbiten 84-89 (per årgång)	50:-
90-91 (per årgång)	80:-
TI-Forth manual	100:-
Hel diskett ur programbanken(st)	30:-

Enstaka program 5:- st + startkostnad 15 kr per skiva eller kassett (1 program=20kr, 3 program=30 kr). Se listor i PB89-3 och PB90-4.

Artiklar sändes till redaktören:
Jan Alexandersson
Springarvägen 5, 3tr
142 61 TRÅNGSUND
Tel. 08-771 0569
(Ring eller skriv till mig om du har frågor om program eller hårdvara)

FUNNELWEB 80 EDITOR V 5.00

by Tony McGovern, Australia (Dec/15/92)

The latest development in the Funnelweb system for the TI-99/4a computer is an extensive rewrite of the system editor. For better or for worse it remains as compatible with the original TI-Writer and E/A editors as can be managed, but now incorporates multilingual features previously available only in an incompatible form in the European version of TI-Writer. A new mode allows use on screen of the full PC character graphics set as supported by most modern printers. All functions use only the base 128Kb VDP memory of the 9938 video processor so that users of standard Geneve 9640 machines are not left out, and no Geneve-hostile code is knowingly included. This 80-column version is dual mode and will serve as a superior 40-column editor as well. An RGB monitor of adequate resolution is essential for serious 80-column work. Amiga compatible monitors are suitable. As in all previous versions it does not depend on any given manufacturer's specific hardware extensions and may be run on a minimal system (80 column needed of course) as long as it can load Funnelweb, the original TI-Writer or E/A modules not being necessary. In particular it is compatible with the Extended Basic module.

Those not already familiar with Funnelweb or TI-Writer should consult TI's original manual (which remains TI copyright material and may still be available from TI in the US at final clearance prices), or else most User Groups have tutorial material or experienced users willing to assist.

The editor is identified internally as Vn 5.00 but loads with no problems from Vn 4.40 of Funnelweb. At this stage in the history of the TI-99/4a there is no longer any grand plan to issue a full update of Funnelweb, but as individual parts are updated they will be so identified.

The language and character files sets are not all complete. This reflects both my limited language abilities and the level of interest in Funnelweb from those parts. German and Swedish are complete, and French largely so. Italian and Dutch are incomplete, and Spanish has not even been considered.

CHARUTIL is a utility for converting char files to source form, or trimming null sectors. It is included because I found existing public domain char-set tools quite unsatisfactory.

‹HD› -- for HardDisk pathname instructs SD to read the directory using the catalog pseudo-file for the pathname, running an assembly version of the standard Basic disk catalog program. The pathname is presented for editing first (also good for Horizon DSK A-Z).

Entering 0 for the disk number reads the Internal, Relative 38 catalog pseudo-file for the pathname as configured or as last entered by ‹HD› from the command line. A directory so obtained does not indicate fractured files. File protection is indicated, but nothing can be done about it, as only DSR file level operations are available in this mode. Marking, deleting, and viewing function normally.

```
***  EDITOR PrintFile  ***
C PIO      Clean print strips hi-bit
           and control chars
L PIO      Includes line numbers
P PIO      Sends printer start codes
Q PIO      Sends printer stop codes
P Q PIO    Sends both, as configured
F DSK1.OB  Saves as DF/80 to DSK1.OB
A DSK1.F   Adds to end of DV/80 file
M DSK1.F   Writes to DSK1.F as DF/128
           with MS-DOS format
U DSK1.F   As DF/128 in UNIX format
```

Du kan få en kopia av FW ED 5.00 om du sänder en DS/SD eller två SS/SD skivor till redaktören och ett frankerat svarskuvert. Observera att du även måste ha FW 4.40 och 80-kolumnskort med 9938 eller 9958.■

GLOSFÖRHÖR - MED XB

```
100 ! **********************
110 ! *     GLOSFÖRHÖR       *
120 ! *          By          *
130 ! *  JONAS FJELLSTEDT    *
140 ! *  tel. 0762/12780     *
150 ! **********************
160 ! VERSION 2.3.1985.32kb
170 CALL CLEAR
180 MAXANTAL=200
190 DIM OVER$(200)
200 DIM GLOS$(200)
210 DIM RAN(200)
220 GLOS$(200)="slut"
230 CALL CHAR(48,"0038444C54
644438")
240 CALL CHAR(64,"04087C4078
40407C")
250 CALL CHAR(79,"0038444444
444438")
260 CALL CHAR(91,"280038447C
444444")
270 CALL CHAR(92,"2800384444
444438")
280 CALL CHAR(93,"100038447C
444444")
290 CALL CHAR(123,"002800384
47C4444")
300 CALL CHAR(124,"002800384
4444438")
310 CALL CHAR(125,"001000384
47C4444")
320 CALL CHAR(111,"000000384
4444438")
330 CALL CHAR(96,"0004087C40
78407C")
340 CALL CHAR(94,"2800444444
444438")
350 CALL CHAR(126,"002800444
4444438")
360 CALL SCREEN(9)
370 FOR SET=1 TO 12
380 CALL COLOR(SET,2,12)
390 NEXT SET
400 DISPLAY AT(12,6):"*** GL
OSFÖRHÖR ***"
410 FOR DELAY=1 TO 400 :: NE
XT DELAY
420 CALL CLEAR
430 ROW=5 :: COL=4
440 DISPLAY AT(ROW,COL):"VÄL
J!"
450 DISPLAY AT(ROW+2,COL):"L
)agring(på kassett)"
460 DISPLAY AT(ROW+4,COL):"H
)ämtning(från kassett)"
470 DISPLAY AT(ROW+6,COL):"I
)nmatning(av glosor)"
480 DISPLAY AT(ROW+8,COL):"G
)losförhör"
490 DISPLAY AT(ROW+10,COL):"
F)ormatering(av glosor)"
500 DISPLAY AT(ROW+12,COL):"
R)adering(av allt)"
510 DISPLAY AT(ROW+14,COL):"
S)lut"
520 CALL KEY(0,K,S):: IF S=0
 THEN 520
530 CALL SOUND(10,700,1)
540 IF K=ASC("L")OR K=ASC("l
")THEN 620
550 IF K=ASC("H")OR K=ASC("h
")THEN 800
560 IF K=ASC("I")OR K=ASC("i
")THEN 1070
570 IF K=ASC("G")OR K=ASC("g
")THEN 1300
580 IF K=ASC("S")OR K=ASC("s
")THEN CALL CLEAR :: STOP
590 IF K=ASC("F")OR K=ASC("f
")THEN 1760
600 IF K=ASC("R")OR K=ASC("r
")THEN GOSUB 2500
610 GOTO 520
620 REM *** LAGRING ***
630 IF GGR<2 THEN 2640
640 DISPLAY AT(11,6)ERASE AL
L:"1. ""CS1"""
650 DISPLAY AT(13,6):"2. ""C
S2"""
660 FOR S=1 TO 100 :: NEXT S
 :: CALL KEY(0,K,S):: IF S=0
 THEN 660
670 IF K=ASC("1")THEN NAME$=
"CS1" :: GOTO 710
680 IF K=ASC("2")THEN NAME$=
"CS2" :: GOTO 710
690 GOTO 660
700 CALL CLEAR
710 OPEN #1:NAME$,SEQUENTIAL
,INTERNAL,OUTPUT,FIXED 56
720 FOR X=1 TO GGR
730 RP=24-LEN(OVER$(X))
740 UTFYLL$=RPT$(" ",RP)
750 ORD$=OVER$(X)&UTFYLL$&GL
OS$(X)
760 PRINT #1:ORD$
770 NEXT X
780 CLOSE #1
790 GOTO 420
800 REM *** HÄMTNING ***
810 IF GGR=MAXANTAL-1 THEN C
ALL MEMFULL ELSE 830
820 GOTO 420
830 DISPLAY AT(12,2)ERASE AL
L:"Ska minnet tömmas (J/N)?"
840 CALL KEY(0,K,S):: IF S=0
 THEN 840
850 IF K=ASC("j")OR K=ASC("J
```

```
")THEN 880
860 IF K=ASC("N")OR K=ASC("n
")THEN CALL CLEAR :: GGR=GGR
-1 :: GOTO 930
870 GOTO 830
880 FOR X=1 TO GGR
890 GLOS$(X)=""
900 OVER$(X)=""
910 NEXT X
920 GGR=0
930 DISPLAY AT(23,6)ERASE AL
L:"*** Hämtning ***"
940 OPEN #1:"CS1",SEQUENTIAL
,INTERNAL,INPUT ,FIXED 56
950 GGR=GGR+1
960 IF GGR=MAXANTAL THEN 970
 ELSE 990
970 CALL MEMFULL
980 GOTO 1040
990 INPUT #1:ORD$
1000 OVER$(GGR)=SEG$(ORD$,1,
24)
1010 GLOS$(GGR)=SEG$(ORD$,25
,LEN(ORD$)-24)
1020 IF GLOS$(GGR)="slut" OR
 GLOS$(GGR)="SLUT" THEN 1040
1030 GOTO 950
1040 CLOSE #1
1050 GOTO 420
1060 GLOS$(X)=""
1070 REM *** INMATNING ***
1080 IF GGR=MAXANTAL-1 THEN
CALL MEMFULL ELSE 1100
1090 GOTO 420
1100 CALL CLEAR
1110 DISPLAY AT(4,6):"*** In
matning ***"
1120 DISPLAY AT(10,1):"Skriv
 ""slut"" när du är klar"
1130 DISPLAY AT(23,6):"Tryck
 på en knapp"
1140 CALL KEY(0,K,S):: IF S=
0 THEN 1140
1150 IF GGR THEN GGR=GGR-1
1160 CALL CLEAR
1170 GGR=GGR+1
1180 IF GGR=MAXANTAL-1 THEN
CALL MEMFULL ELSE 1200
1190 GOTO 420
1200 DISPLAY AT(5,8):"Inmatn
ing"
1210 DISPLAY AT(8,5):"Glosan
:"
1220 ACCEPT AT(10,3)SIZE(-24
):GLOS$(GGR)
1230 IF GLOS$(GGR)="slut" OR
 GLOS$(GGR)="SLUT" THEN 420
1240 DISPLAY AT(13,5):"Övers
ättningen:"
1250 ACCEPT AT(15,3)SIZE(-24
):OVER$(GGR)
1260 DISPLAY AT(19,5):"Stämm
er det (J/N)?J"
1270 ACCEPT AT(19,23)VALIDAT
E("njNJ")SIZE(-1):A$
1280 IF A$="j" OR A$="J" THE
N 1160
1290 GOTO 1200
1300 REM *** GLOSFÖRHÖR ***
1310 IF GGR<2 THEN 2640
1320 PO=0
1330 CALL CLEAR
1340 DISPLAY AT(12,6):"Var g
od vänta!"
1350 RANDOMIZE
1360 FOR X=1 TO GGR-1
1370 DISPLAY AT(23,3):"Initi
ering av glosa: ";X
1380 RAN(X)=INT(RND*(GGR-1))
+1
1390 FOR Y=1 TO X-1
1400 IF RAN(Y)=RAN(X)THEN 13
80
1410 NEXT Y
1420 NEXT X
1430 CALL CLEAR
1440 FOR X=1 TO GGR-1
1450 DISPLAY AT(4,5):"*** Gl
osförhör ***"
1460 DISPLAY AT(9,1):"Översä
tt:"
1470 DISPLAY AT(11,5):OVER$(
RAN(X))
1480 ACCEPT AT(14,1)SIZE(24)
:O$
1490 IF O$=GLOS$(RAN(X))THEN
 PO=PO+1 ELSE 1580
1500 DISPLAY AT(23,8):"Din p
oÄng är";PO
1510 NEXT X
1520 CALL CLEAR
1530 DISPLAY AT(11,6):"Din p
oäng blev: ";PO
1540 DISPLAY AT(13,6):"av ";
GGR-1;" möjliga."
1550 IF PO=GGR-1 THEN CALL M
ELGLAD ELSE CALL MELLED
1560 FOR DELAY=1 TO 750 :: N
EXT DELAY
1570 GOTO 420
1580 CALL CLEAR
1590 FOR Z=1 TO 5
1600 CALL SOUND(10,660,1)
1610 DISPLAY AT(12,1):"*** F
EL!!! FEL!!! FEL!!! ***"
1620 FOR DELAY=1 TO 10 :: NE
XT DELAY
1630 CALL SOUND(10,440,1)
1640 DISPLAY AT(12,1):""
1650 FOR DELAY=1 TO 10 :: NE
XT DELAY
1660 NEXT Z
1670 CALL CLEAR
1680 DISPLAY AT(11,4):"Det r
```

```
ätta svaret är:"
1690 DISPLAY AT(13,4):GLOS$(
RAN(X))
1700 FOR DELAY=1 TO 500 :: N
EXT DELAY
1710 CALL CLEAR
1720 DISPLAY AT(10,3):"Skriv
 ordet för:": :OVER$(RAN(X))
1730 ACCEPT AT(14,3):O$
1740 IF O$<>GLOS$(RAN(X))THE
N 1580
1750 GOTO 1500
1760 REM ** FORMATERING **
1770 IF GGR<2 THEN 2640
1780 RAD=7 :: KOL=6
1790 CALL CLEAR
1800 DISPLAY AT(RAD,KOL):"VÄ
LJ!"
1810 DISPLAY AT(RAD+3,KOL):"
L)ista glosorna"
1820 DISPLAY AT(RAD+5,KOL):"
R)adera en glosa"
1830 DISPLAY AT(RAD+7,KOL):"
J)ustera en glosa"
1840 DISPLAY AT(RAD+9,KOL):"
H)uvudmenyn"
1850 CALL KEY(0,K,S):: IF S=
0 THEN 1850
1860 IF K=ASC("L")OR K=ASC("
l")THEN 1910
1870 IF K=ASC("R")OR K=ASC("
r")THEN 2090
1880 IF K=ASC("J")OR K=ASC("
j")THEN 2290
1890 IF K=ASC("H")OR K=ASC("
h")THEN 420
1900 GOTO 1850
1910 REM *** LISTNING ***
1920 CALL CLEAR
1930 AA=0
1940 FOR X=1 TO GGR-1
1950 AA=AA+3
1960 DISPLAY AT(AA,1):USING
"###":X
1970 DISPLAY AT(AA,5):GLOS$(
X)
1980 DISPLAY AT(AA+1,5):OVER
$(X)
1990 IF AA=21 THEN 2000 ELSE
 2050
2000 DISPLAY AT(24,6):"Tryck
 på en knapp"
2010 CALL KEY(0,K,S):: IF S=
0 THEN 2010
2020 IF GLOS$(X+1)="slut" OR
 GLOS$(X+1)="SLUT" THEN 2050
2030 CALL CLEAR
2040 AA=0
2050 NEXT X
2060 DISPLAY AT(24,6):"Tryck
 på en knapp"
2070 CALL KEY(0,K,S):: IF S=
0 THEN 2070
2080 GOTO 1770
2090 REM *** RADERING ***
2100 REM  AV ENSTAKA GLOSA
2110 CALL CLEAR
2120 DISPLAY AT(11,1):"Skriv
 nummret på glosan:"
2130 ACCEPT AT(13,3)VALIDATE
("1234567890")SIZE(3):RAD
2140 IF RAD>GGR-1 THEN 2130
2150 FOR X=1 TO GGR-1
2160 IF GLOS$(X)=GLOS$(RAD)T
HEN 2230
2170 NEXT X
2180 DISPLAY AT(17,3):GLOS$(
RAD)
2190 FOR X=5 TO 30
2200 CALL HCHAR(17,X,32)
2210 NEXT X
2220 GOTO 1770
2230 FOR Y=X TO GGR
2240 GLOS$(Y)=GLOS$(Y+1)
2250 OVER$(Y)=OVER$(Y+1)
2260 NEXT Y
2270 GGR=GGR-1
2280 IF GGR<2 THEN 420 ELSE
1770
2290 REM *** ÄNDRING ***
2300 CALL CLEAR
2310 DISPLAY AT(4,7):"*** Än
dring ***"
2320 DISPLAY AT(7,1):"Skriv
nummret pÅ glosan:"
2330 ACCEPT AT(9,3)SIZE(2)VA
LIDATE("1234567890"):ANDR
2340 IF ANDR>GGR-1 THEN 2330
2350 DISPLAY AT(11,1):"Ändra
 glosan:"
2360 DISPLAY AT(13,3):GLOS$(
ANDR)
2370 ACCEPT AT(13,3)SIZE(-24
):NYGLOS$
2380 DISPLAY AT(15,1):"Ändra
 översättningen"
2390 DISPLAY AT(17,3):OVER$(
ANDR)
2400 ACCEPT AT(17,3)SIZE(-24
):NYOVER$
2410 FOR X=1 TO GGR-1
2420 IF GLOS$(X)=GLOS$(ANDR)
THEN 2450
2430 NEXT X
2440 GOTO 1770
2450 GLOS$(X)=NYGLOS$
2460 OVER$(X)=NYOVER$
2470 NYGLOS$=""
2480 NYOVER$=""
2490 GOTO 1770
2500 REM *** RADERING ***
2510 IF GGR<2 THEN 2640
2520 DISPLAY AT(24,3):"Vill
du radera (J/N)?"
```

```
2530 CALL KEY(0,K,S):: IF S=
0 THEN 2530
2540 IF K=ASC("N")OR K=ASC("
n")THEN 420
2550 IF K=ASC("J")OR K=ASC("
j")THEN 2570
2560 GOTO 2530
2570 REM
2580 FOR X=1 TO GGR
2590 GLOS$(X)=""
2600 OVER$(X)=""
2610 NEXT X
2620 GGR=0
2630 GOTO 420
2640 REM ** MINNET TOMT **
2650 FOR X=1 TO 10
2660 CALL SOUND(10,220,1)
2670 DISPLAY AT(23,2):"** IN
GA GLOSOR INMATADE **"
2680 FOR S=1 TO 10 :: NEXT S
2690 DISPLAY AT(23,1):""
2700 FOR S=1 TO 10 :: NEXT S
2710 NEXT X
2720 GOTO 520
2730 SUB MELGLAD
2740 DISPLAY AT(21,5):"Toppe
n! Lycka till!"
2750 FOR Z=1 TO 7
2760 CALL SOUND(10,440,1)
2770 CALL SOUND(10,660,1)
2780 NEXT Z
2790 SUBEND
2800 SUB MELLED
2810 DISPLAY AT(21,6):"Bättr
e kan du!!!"
2820 DISPLAY AT(23,8):"Pröva
 igen!!"
2830 FOR Z=1 TO 7
2840 CALL SOUND(10,110,1)
2850 CALL SOUND(10,220,1)
2860 NEXT Z
2870 SUBEND
2880 SUB MEMFULL
2890 CALL CLEAR :: GGR=MAXAN
TAL
2900 FOR Z=1 TO 5
2910 CALL SOUND(10,220,1)
2920 DISPLAY AT(11,1):"MINNE
SKAPACITETEN ÄR UTTÖMD."
2930 FOR D=1 TO 50 :: NEXT D
2940 DISPLAY AT(11,1):""
2950 CALL SOUND(10,440,1)
2960 DISPLAY AT(13,2):"INGA
FLER GLOSOR FÅR PLATS"
2970 FOR D=1 TO 50 :: NEXT D
2980 DISPLAY AT(13,1):""
2990 NEXT Z
3000 OVER$(MAXANTAL)="SLUT"
3010 SUBEND                  ■
```

SAMLINGSSKIVA PROGBIT-92

(30 kr till postgiro 19 83 00-6) Used=359 Free=1 Filecount=49

Filename	Size	Type	Rec	P
ALPHAOMEGA	5	Program	BX	
AMORTERING	5	Program	BX	
BATMAN	9	Program	BX	
BLKVDP	3	Dis/Var	80	
BOKSTAV	6	Program	BX	
BUG-AL-1/O	3	Dis/Fix	80	
CALENDAR-2	6	Program	BX	
CLOCK-TIME	3	Program	BX	
CONVERT	11	Program	BX	
CRAYON/O	10	Dis/Fix	80	
DAYBETWEEN	7	Program	BX	
DAYCOUNT	8	Program	BX	
DISPLAY	2	Program	BX	
DRAW	7	Dis/Var	80	
EASTERCALC	4	Program	BX	
FILECOMPAR	3	Program	BX	
G1-PRG1/O	6	Dis/Fix	80	
G1-PRG2/O	5	Dis/Fix	80	
G1-PRG3/O	6	Dis/Fix	80	
G1-TAB-38	3	Dis/Var	80	
G1-TAB-4A	3	Dis/Var	80	
G138	4	Dis/Var	80	
G14A	9	Dis/Var	80	
G2-PRG1/O	6	Dis/Fix	80	
G2-PRG2/O	6	Dis/Fix	80	
G2-PRG3/O	8	Dis/Fix	80	
G2-TAB-38T	4	Dis/Var	80	
G2-TAB-4AG	3	Dis/Var	80	
G2-TAB-4AT	3	Dis/Var	80	
G238T	4	Dis/Var	80	
G24AG	7	Dis/Var	80	
G24AT	10	Dis/Var	80	
INVADER	17	Program	BX	
MAILING	27	Program	BX	
MC-TUT7/O	5	Dis/Fix	80	
QUICKBILL	9	Program	BX	
RÄNTEBERÄK	18	Program	BX	
SOUND	9	Program	BX	
SPRINGSONG	13	Program	BX	
STAMPDBASE	20	Program	BX	
T1-PRG1/O	6	Dis/Fix	80	
T1-TAB-4A	2	Dis/Var	80	
T14A	6	Dis/Var	80	
T2	8	Dis/Var	80	
T2-PRG1/O	6	Dis/Fix	80	
T2-TAB-38	3	Dis/Var	80	
TERMITE	15	Program	BX	
TSTKEY	6	Dis/Var	80	
XBCAT	8	Program	BX	

■

BASIC OCH XB TIPS - 1

av Jan Alexandersson

VARIABELNAMN

Det är tillåtet att använda @, _, Å, Ä och Ö i variabelnamn:

```
@        ASCII-kod 64
] = A    ASCII-kod 93
[ = Ä    ASCII-kod 91
\ = Ö    ASCII-kod 92
_        ASCII-kod 95
```

Följande program fungerar:

```
100 ]=3
110 [=4
120 \=]*[
130 PRINT \
```

Det är tillåtet att använda samma namn som underprogram och som variabel. Prova följande:

```
100 SCREEN=14
110 CALL SCREEN(SCREEN)
120 CALL KEY(3,KEY,STA)
130 IF KEY<>74 THEN 120
```

RADLÄNGDER

Maximal radlängd är normalt 112 i Basic och 140 i Extended Basic. Dessa kan förlängas genom upprepad editering av raden. Gränsen sätts ytterst av Crunch Buffer som är 158 tecken för både Basic och Extended Basic. Gör så här i Basic:

```
100 PRINT "AAAA...A"    (100 st A)
(ENTER)           (112 tecken/rad)
100 (FCTN X) pil nedåt
(editera in flera A)
100 PRINT "AAAA...A"    (128 st A)
(ENTER)           (140 tecken/rad)
100 (FCTN X) pil nedåt
(editera in flera A)
100 PRINT "AAAA...A"    (155 st A)
(ENTER)        (rad med 167 tecken)
```

Om du försöker med 156 st A får du felmeddelandet "LINE TOO LONG".

I Extended Basic är det något enklare eftersom du redan från början får in 140 tecken per rad. Fortsätt sedan att editera som i Basic-exemplet. Du bör dock aldrig gå utöver 140 tecken i Extended Basic eftersom REDO-bufferten endast klarar 140 tecken.

Hur kan man få in 167 tecken i en buffert på 158 tecken? Jo, det görs en översättning(kompilering) till en mer kompakt kod. Alla Basic-kommandon lagras med en byte "token". Prova följande i Extended Basic:

```
100 GOTO :: GOTO :: GOTO :: GOTO ::
    GOTO o.s.v.
```

Genom upprepade editeringar kan man klämma in 78 st GOTO och 77 st ::. En programrad täcker hela skärmen!

Med Basic måste editeringen upprepas för varje rad om 28 tecken medan det i Extended Basic räcker med var annan rad, d.v.s. efter 56 tecken. Vid DATA-satser kan ej fulla rader användas.

SIZE (MINNESUTRYMME)

Efter det att ett program har körts eller efter det att beräkningar gjorts direkt från tangentbordet (utan radnummer) finns det minnesutrymme ockuperat av numeriska variabler och strängvariabler m.m. Om du endast vill veta hur långt programmet är för att beräkna hur mycket plats det tar på kassettband eller flexskiva måste variabelminnet tömmas. Detta görs alltid vid editering. Ett lämpligt kommando att använda är:

```
1 (ENTER)
SIZE
```

Observera att även om du har expansionsminne måste du tömma det statiska variabelminnet före SIZE. Följande gäller nämligen:

PROGRAM SPACE = program + numeriska variabler + pekare till strängvariabler

STACK SPACE = strängvariabler + återhoppsadresser

För att alltid kunna använda 1 (ENTER) på detta sätt bör radnummer 1 inte användas i programmet. Även SAVE-kommandot rensar variabelminnena.

Det tillgängliga minnesutrymmet i TI-99/4A för program varierar beroende på vilken modul och vilken kringutrustning som används. Följande tabell visar läget uttryckt i bytes:

	program	stack
Basic	14536	-
Basic + diskkontroll	12448	-
Basic + DSK + FILES(1)	13480	-
Extended Basic	13928	-
XB + 32 kbytes EM	(24488)	13928
XB + diskkontroll	11840	-
XB + EM + DSK	24488	11840
XB+EM + DSK + FILES(1)	24488	12876
Terminal Emulator II	14024	-
TE II + diskkontroll	11936	-
SAVE MINIMEM	4088	-

För XB + EM bestämmer stackutrymmet hur långa program som kan sparas på kassett. Detta problem med kassettlagring finns även tillsammans med flexskiveenhet. Du kan inte göra reservkopior av långa program på kassett. Om du använder flexskiva går det bra att ha 24488 bytes långa program som lagras i INT/VAR 254 format. Program i Extended Basic kan vara ännu längre om du delar upp det i olika programmoduler som laddar in varandra med RUN "DSK1.NAMN".

Som du ser finns det ytterligare 8 kbytes i expansionsminnet vid XB + EM. Denna minnesdel används endast för assembler-program. Grund-Basic kan inte använda expansionsminne även om det är anslutet till maskinen. Tillsammans med Editor/Assembler eller Mini Memory modulerna kan man dock använda expansionsminnet för assemblerprogram som kan anropas från Basic.

I vanlig Basic måste du använda ett knep för att ta reda på programmets storlek eftersom SIZE-kommandot saknas:

```
1 M=M+8
2 GOSUB 1
```

Detta placeras först i programmet som sedan körs tills det stannar p.g.a. MEMORY FULL. PRINT M ger då hur mycket ledigt minne som finns. Eftersom programmet körs kommer alla variabler att tilldelas minnesutrymme vid "pre-scan" som alltid görs innan programmet löper.

Ett annat sätt att testa minnet utan att köra programmet är att använda DIM A(X) i kommando-mod. Testa olika värden på X tills du får MEMORY FULL. Kom ihåg att editera med t.ex. 1 (ENTER) innan DIM A(X) så att inga andra variabler ockuperar minnet.

Ett tredje sätt är att använda Mini Memory eller Editor/Assembler i Basic:

```
CALL PEEK(-31974,A,B)
PRINT 256*A+B-1817
```

I Extended Basic utan expansionsminne kan man göra på samma sätt om man istället skriver PRINT 256*A+B-2413.

Du kan testa innehållet i stackminnet med följande program (ändra konstanten till 1817 för Basic med MM eller EA):

```
100 CALL CLEAR
110 CALL PEEK(-31974,A,B)
120 PRINT A*256+B-2413
130 A$="AAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA"
140 GOTO 110
```

När tillgängligt minnesutrymme närmar sig 0 kommer datorn att göra "garbage collection", d.v.s. rensa överblivet minne så att det kan användas på nytt. Du får då en liten extra paus som märks bäst i Basic (c:a 1 sekund).

LADDA LÅNGA PROGRAM FRÅN KASSETT

Om du har ett mycket långt program på kassett som ska föras över till flexskiva kan det bli problem. Första åtgärden är att skriva CALL FILES(1) följt av NEW. Om detta inte räcker för Extended Basic måste du korta ned programmet och lagra tillbaka på kassettband. Tag bort alla onödiga REM och skriv flera PRINT-satser på samma rad med kolon (:). Om detta inte hjälper måste programmet delas i två delar och sedan smältas ihop med MERGE.

Även om du fått in programmet på flexskivan återstår ett problem. Du är alltid tvungen att skriva CALL

FILES(1) varje gång du använder programmet eftersom det lagras i PROGRAM-format. Alla långa XB-program bör göras om till INT/VAR 254-format. Gör så här:

```
CALL FILES(1)
NEW
OLD CS1
SAVE DSK1.NAMN,MERGE
CALL FILES(3)
NEW
MERGE DSK1.NAMN
SAVE DSK1.NAMN
```

Det finns en liten skillnad mellan SAVE och OLD med CALL FILES(3). Maximal längd för SAVE är 11840 bytes som väntat men OLD klarar 11872 bytes med PROGRAM-format.

TRACE

Om man använder TRACE för att söka fel, kommer alla CALL CLEAR man passerar att sudda ut de radnummer som TRACE tidigare har skrivit ut på skärmen. Med hjälp av Extended Basic kan man tillfälligt skriva till ett nytt underprogram i slutet av listningen som ändrar datorns inbyggda underprogram CLEAR:

```
100 CALL CLEAR
9000 SUB CLEAR
9010 SUBEND
```

På liknande sätt kan man definiera om alla inbyggda CALL så att de gör något annat än vad TI har tänkt.

INMATNING AV TEXTSTRÄNGAR

Det gäller att välja rätt kommando för inmatning av text. I Basic finns det bara INPUT att välja på men i Extended Basic finns stor valfrihet. Det är dock stor skillnad på maximalt antal tecken:

INPUT i Basic	111-112
INPUT i XB (eller LINPUT)	138-139
ACCEPT	255
ACCEPT AT	28

Det är faktiskt möjligt att få mer än 28 tecken med ACCEPT AT. Du måste använda en variabel med flera fält t.ex. A$(0). I ACCEPT-satsen måste du då skriva A$(1-1) och använda VALIDATE för alla tecken som ska kunna matas in:

```
100 CALL CLEAR
110 ACCEPT AT(10,1)VALIDATE(
UALPHA,NUMERIC):A$(1-1)
120 DISPLAY AT(15,1):A$(0)
130 GOTO 110
```

Ett problem med denna rutin är att kolumn 31, 32, 1 och 2 kommer att användas vilket inte ser snyggt ut. Dessutom fungerar (INS) endast för ett tecken åt gången vilket skiljer sig från det normala.

INPUT

Du bör inte använda CTRL-, A B C i INPUT-satsernas ledtexter i Extended Basic. CTRL-, A B C motsvarar ASCII-koderna 128-131. Om någon av dessa används vid numeriska variabler och du trycker på en bokstav borde man få WARNING och en ny chans att skriva rätt. Istället får man WARNING följt av SYNTAX ERROR och programmet stoppar. Övriga ASCII-koder fungerar bra. I vanlig Basic finns inga problem med ASCII koderna 128-131. Du bör dock undvika alla ASCII-koder över 127 i programlistningar eftersom dessa ej kan fås ut på skrivare och således EJ publiceras i Programbiten.

PRINT

Det är även möjligt att använda flera kolon (:) i rad i PRINT-satser i Extended Basic. Tänk bara på att använda mellanslag mellan närliggande kolon vid editering. Efter det du tryckt på (ENTER) gör datorn en översättning till intern kod. Då lagras : för radmatning med ett unikt värde ASCII-kod 181 (som ej används i textsträngar) och dubbelkolon :: med ett eget värde ASCII-kod 130.

Datorn lagrar alltid dessa kolon (181) på samma sätt i både Basic och Extended Basic. Det är endast vid editering som det ser olika ut. Du kan alltid flytta program mellan Basic och Extended Basic när det gäller PRINT-satser.

TAB

I normala PRINT-satser kan fler än

28 tecken användas så att flera rader kan skrivas med samma kommando t.ex.:

```
PRINT
"   ABCDEFGHIJKLMNOPQRSTUVXYZ123".
```

Om man istället använder TAB kommer det inte att fungera om texten skall skrivas på flera rader t.ex.:

```
PRINT
TAB(4);"ABCDEFGHIJKLMNOPQRSTUVXYZ123"
```

I detta fall skrivs texten ut längst till vänster.

DATA och READ

Man bör alltid lägga till en extra DATA-post på den sista raden med DATA. Denna extra post ska aldrig läsas med READ men kommer att snabba upp programmet. När datorn läser den sista posten kommer den annars att stanna upp en kort stund (någon sekund). Det kan vara lämpligt att använda något ovanligt t.ex. @. Se följande exempel:

```
1900 DATA ADAM,23,10,BERTIL,24,10,@
```

OPEN

Om du skriver ett program som ska läsa en DIS/VAR 80-fil bör du inte i onödan öppna den som UPDATE eftersom programmet undersöker om filen är skrivskyddad. Även om du inte skriver något efter OPEN så är UPDATE ett "default"-värde. Programmet kommer att krascha om du öppnar en skrivskyddad fil med UPDATE även om du bara läser filen i programmet. Följande rad är således olämplig: 100 OPEN #1:FIL$ och bör ändras till 100 OPEN #1:FIL$,INPUT.

Det är också viktigt att du stänger en öppen fil innan du byter skiva. Du kan annars råka ut för att den stängs oavsiktligt och detta innebär att delar av skivan blir förstörd. Följande kommandon stänger filer och skriver på den skiva som råkar sitta i just då: CLOSE, SAVE, OLD, BYE, editering och 1 (ENTER).

AND, OR, XOR, NOT

Dessa kommandon saknas i Basic men kan enkelt omskrivas:

```
IF A=8 AND B=6 THEN ...      XB
IF (A=8)*(B=6) THEN ...      Basic

IF A=8 OR B=6  THEN ...      XB
IF (A=8)+(B=6) THEN ...      Basic

IF A=8 XOR B=6 THEN ...      XB
IF (A=8)+(B=6)=-1 THEN ...   Basic

IF NOT A=8 THEN ...          XB
IF A<>8    THEN ...          Basic
```

NOT kan även skrivas som:

```
IF A-8 THEN ...
```

MATEMATISKA FUNKTIONER

DEF av ARCSIN och ARCCOS i beskrivningen för Extended Basic fungerar inte om X=1 eller X=0. Använd istället följande:

```
DEF ARCSIN(X)=
2*ATN(X/(1+SQR(1-X*X)))

DEF ARCCOS(X)=2*ATN(1)-ARCSIN(X)
```

Dessa definitioner fungerar i både Basic och Extended Basic. Om du endast använder Extended Basic kan man byte ut 2*ATN(1) mot PI/2. Om endast ARCCOS används kan definitionen för ARCSIN bakas in i ARCCOS.

På motsvarande sätt kan man definiera PI i Basic:

```
DEF PI=4*ATN(1)
```

Man kan sedan använda PI i beräkningar som om PI fanns från början.

Andra användbara funktioner är 10-logaritm och sinus för grader:

```
DEF LOG10(X)=LOG(X)/LOG(10)

DEF SINGRAD(X)=SIN(X*ATN(1)/45)
```

■

MODULER TI-99/4A

Lägg till följande moduler till listan i PB 91-5 sid 23:

```
PHM 3017 TERMINAL EMULATOR I
PHM 3019 DISK MANAGER I
PHM 3125 E.T.
PHM 3207 CROSSFIRE
```

■

HORIZON 4000 RAM-DISK

av Jan Alexandersson

Horizon 4000 RAM-disk har nu börjat levereras från Bud Mills Services. Kortet är helt omgjort jämfört med tidigare kort eftersom det sticker ut en bra bit bakåt ungefär som Myarc HFDC. På den utstickande delen finns laddningsbara batterier monterade samt en "disable switch".

Man har utnyttjat den återstående ytan av kortet mycket effektivt så att det får plats 16 st RAM-chips (32, 128 eller 512 kbytes) för data/sektorer, 1 st 8 kbytes för operativsystem och 1 st 32 kbytes för expansionsminne (måste beställas extra). Du kan inte blanda olika data-chip utan alla 16 måste ha samma storlek. Följande statiska RAM-chips används:

128 kbytes	Hitachi	HM628128LP-10
32 kbytes	NEC	D43256AC-10L
8 kbytes	Hitachi	HM6264ALP-12

Alla chips finns i hållare medan motstånd, dioder och kondensatorer är fastlödda. Dessa hållare för chipen finns monterade från början även om du inte beställer alla RAM-chips. Det finns således plats för 8 Mbytes men av detta kan endast 4 Mbytes användas som RAM-diskar medan resterande minne måste användas som bankswitchade 8 kbytes. Eventuellt kanske Phoenix som finns inbyggt kan ordna det dubbla antalet DSK (vet inte hur det fungerar). Även om du använder 128 kbytes RAM-chips så får det plats 2 Mbytes. Den tidigare Horizon 3000 hade endast 12 st RAM-chips som tillät max 1,5 Mbytes.

Du kan endast ha ett expansionsminne 32 kbytes så du måste ta bort TI:s originalkort för 32 kbytes om Horizon har 32 kbytes EM monterat. TI:s originalkort använder 16 st dynamiska 2 kbytes chips som nu ersätts av ett enda chip. Naturligtvis får högst ett Horizon-kort ha expansionsminne.

Läs min artikel i PB 92-4 om Horizon RAM-disk. I fortsättningen beskriver jag endast nyheter för Horizon 4000.

ROS 8.14B OPERATIVSYSTEM 01/20/92

ROS 8.14B kan användas för alla versioner av Horizon RAM-disk med 8, 32, 128 eller 512 kbytes chips.

Programmet CFG (Configure) delar upp kortet i olika logiska RAM-diskar och RAM-minne för bankswitchning på adress >6000 ->7FFF (om RAMBO finns). När du startar CFG så kommer det upp en lista över alla kort med egen DSR:

CARD ADDR.	DEFINITION OF CARD SOFTWARE	TOTAL SIZE	RAM-DISK	RAMBO RAM
>1000	DRIVE CONTROL			
>1100	DRIVE CONTROL			
>1300	RS232 and PIO			
>1400	UNDEFINABLE			
>1500	RAMBO/HORIZON	512k	490k	16k
>1600	8k*8 HORIZON	186k	184k	0k
>1700	PGRAM CARD			

CRU >1400 har i detta fall ett DIJIT AVPC- och CRU >1000 ett Myarc HFDC-kort. Det behövs 10 kbytes för operativsystemet om Horizon har RAMBO, vilket alltid finns hos Horizon 4000 som standard.

Vid konfigureringen av en 512 kbytes Horizon anges att 2040 sektorer finns tillgängliga. Dessa har sedan (i exemplet ovan) delats upp i två RAM-diskar på 1600 respektive 360 sektorer. Resten blir automatiskt RAMBO-RAM.

Du kan ladda in ditt eget tecken-set i RAM-disken med en vanlig CHARA-fil från CFG-programmet. Detta används sedan vid start av program.

QUIT från CFG-programmet fungerar inte alltid p.g.a. mitt DIJIT AVPC.

FILER PÅ RAM-DISK

Varje Horizon kort kan ha upp till 10 st olika DSK-enheter som anropas med det nummer du själv väljer vid konfigureringen. Du kan välja nummer 1-9 och A-Z. Storleken på varje DSK kan väljas fritt (i steg om 8 sektorer) vid konfigureringen

men ingen DSK kan vara större än 1600 sektorer (400 kbytes). Varje DSK kan ha upp till 127 filer.

Horizon RAM-disk sparar alltid filerna på lägsta möjliga sektornummer. Om du sparar en fil kommer den t.ex. att lagras med filhuvud på sektor 2 och data på sektor 3-9. Du som använder FORTH bör tänka på detta. Ett vanligt diskkontrollkort för flexskiva kommer istället att lagra filhuvud på sektor 2 och data på sektor 34-40 eftersom sektor 2-33 har reserverats för filhuvud. DM 1000 lagrar den första filens data på sektor 34-40 även på RAM-disk (skillnad mot FW DR och Basic SAVE).

Även med CALL FILES(1) kan du ha 16 öppna filer eftersom diskbufferten finns i RAM-disken. I princip skulle du kunna ha CALL FILES(0) och frigöra ännu mer VDP-RAM till programkörningen. Du kan inte skriva detta direkt men du kan poka in det med CALL INIT följt av CALL LOAD (-31888,63,255) och NEW. Myarcs diskkontrollkort och HFDC använder på liknande sätt RAM i själva kortet så CALL FILES(0) passar även där.

OPEN av en redan existerande fil kan enligt manualen göras utan att man anger attribut som INT/FIX 192. Denna finess fungerar inte (I/O Error 02) på mitt system. Det enda som går att hoppa över är postlängden 192, men INTERNAL och FIXED måste skrivas ut.

SCRATCH RECORD ger möjlighet till att ta bort en post i en öppen datafil men endast för DIS/FIX eller INT/FIX. Sektorerna frigörs inte p.g.a. detta så filen kan behöva läsas in sin helhet och skrivas tillbaka någon gång då och då.

I assembler kan du själv bestämma om en fil ska läsas via VDP-RAM (opcode >02) eller CPU-RAM (opcode >42).

Det finns ytterligare tre nya opcode för assembler som laddar program:
opcode >A : assembler PROGRAM-fil
opcode >B : XB PROGRAM, INT/VAR 254
opcode >C : cartridge i PGRAM-format

DISK MANAGER

Det medföljer en modifierad DM 1000 version 3.5B som fungerar för DSK 1-9 och A-Z. Av någon underlig anledning försöker DM 1000 ladda in något från DSK7, troligen en CHARA-fil. QUIT från denna version av DM 1000 fungerar inte på mitt system utan det låser sig med konstiga tecken på hela skärmen, beror nog på DIJIT AVPC. Du kan även använda Funnelweb DISK REVIEW för DSK 1-9 men den klarar inte A-Z.

Du kan använda en vanlig Disk Manager för att spara och ta bort filer från Horizon RAM-disk. Formateringen måste dock göras med CFG-programmet och inte med Disk Manager. CFG-programmets formatering motsvarar i stort sett SWEEP DISK eftersom endast sektor 0 och 1 skrivs.

CALL och DSR(Device Service Routine)

CALL och DSR av upp till nio olika program på varje Horizon-kort kan ladda in Assemblerprogram, Basic-program eller Cartridge-program (för körning i P-GRAM). Med två Horizon-kort kan sammanlagt 18 olika CALL av program finnas.

Det finns även en DSR som kan ladda andra program som inte finns med bland ovanstående nio förvalda med t.ex. DELETE "LD.4.NAMN" eller OLD LD.4.NAMN om programmet NAMN ska laddas från DSK4. Detta går mycket snabbt eftersom man inte går omvägen över VDP-RAM vid laddningen (mycket bra för XB-program). Även i detta fall kan Assembler, Basic och Cartridge laddas.

MENU V.7.39, 03/04/90, OPA

Denna version som medföljer klarar nu DSK 1-9 och A-Z. Valet av C för Cartridge fungerar ej för mig eftersom datorn låser sig, troligen p.g.a. DIJIT AVPC. Jag använder Funnelweb som auto-start istället för MENU.

RAMBO

RAMBO ger möjlighet till att koppla in 8 kbytes bankar på CPU-adress

>6000 - >7FFF. Du kan ha hur många som helst men naturligtvis kan endast en vara aktivt inkopplad åt gången. Det går mycket snabbt att skifta mellan bankarna.

RAMBO medger även att du laddar in en egen "User-DSR" på högst 1 kbytes. Du kan skriva egna program som aktiveras vid power-up, interrupt eller CALL och DSR. Om DSR hoppar tillbaka till Basic/XB bör endast DELETE eller CALL användas. RUN, OLD och OPEN ger nämligen ett felmeddelande. En fil med "User-DSR" kan enkelt laddas från CFG-programmet på samma sätt som ROS. Det följer med en demo-fil som kan laddas in som User-DSR. Interrupt-DSR fungerar inte på mitt system, troligen beroende på DIJIT AVPC. ■

DU ÄR HÄNGD - GISSA ORD

```
100 REM ***********
110 REM * HANGMAN *
120 REM ***********
130 REM HEMDATOR 84-01 REV
132 REM NITTINIAN 84-04
135 REM EJ XB
140 REM Y.THORFVE&R.EVERETT
145 RANDOMIZE
150 CALL CLEAR
160 DATA 43,FF7E3C18,102,181
8181818181818
170 DATA 115,FFFFFFEFE7E3E1E
,116,FFFFFFF7E7C78707,117,80
40201008040201,118,010204081
020408
180 DATA 119,99997E7E1818181
8,120,1818181818181818,121,8
0C06030180C0607,122,0103060C
183060E
190 DATA 128,C0C0E060783E0F0
3,129,030307061E7CF0C,130,00
0000000000FFFF,137,E0E3E3E0E
0E0E0E
200 DATA 138,008181000000181
8,139,07C7C70707070707,140,0
000007E,141,000081423C,142,0
0003C4281
210 DATA 144,000103070F1F3F7
F,145,0080C0E0F0F8FCFE,146,7
F7F7F3F1F0F0701,147,FEFEFEFC
F8F0E080,148,000103070E0E
220 DATA 149,FFFFFFFFDBDBDBD
B,150,8080C0E060602,151,DBDB
18,152,0080C0E0707,153,01010
307060604
225 DATA 91,00440038447C4444
,92,0044007C4444447C,93,0010
0038447C4444
230 A$="0103070F1F3F7FFF"
240 B$="FFFEFCF8F0E0C08"
250 C$="FFFFFFFFFFFFFFFF"
260 E$="FF7F3F1F0F070301"
270 F$="80C0E0F0F8FCFEFF"
280 CALL CHAR(40,E$)
290 CALL CHAR(41,B$)
300 CALL CHAR(97,C$)
310 CALL CHAR(98,A$)
320 CALL CHAR(99,F$)
330 CALL CHAR(100,B$)
340 CALL CHAR(101,E$)
350 CALL CHAR(104,A$)
360 CALL CHAR(105,B$)
370 CALL CHAR(106,C$)
380 CALL CHAR(107,E$)
390 CALL CHAR(108,F$)
400 CALL CHAR(112,A$)
410 CALL CHAR(113,C$)
420 CALL CHAR(114,F$)
430 CALL CHAR(131,C$)
440 CALL CHAR(136,"")
450 FOR I=1 TO 32
460 READ NR,TEC$
470 CALL CHAR(NR,TEC$)
480 NEXT I
490 CALL SCREEN(16)
500 CALL COLOR(9,2,1)
510 CALL COLOR(10,14,1)
520 CALL COLOR(11,1,1)
530 CALL COLOR(12,11,1)
540 CALL COLOR(13,11,14)
550 CALL COLOR(14,2,10)
560 CALL COLOR(15,10,1)
570 CALL COLOR(16,9,1)
580 CALL COLOR(2,1,1)
585 GOSUB 5000
590 GOTO 1840
600 REM GALGE
610 PRINT TAB(19);"aaaaaaaaa
a"
620 PRINT "jjjjjjjjjj
  x eac   a"
630 PRINT "jDITT ORDj
  x  eac  a"
640 PRINT "jjjjjjjjjj
  x   eac a"
650 PRINT TAB(20);"x    eaca
"
660 PRINT TAB(20);"x     eaa
"
670 PRINT TAB(24);"pqrea"
680 PRINT TAB(24);"sqt a"
690 PRINT TAB(24);"qqq a"
```

```
700 PRINT "GISSA EN
     pqqqra"
710 PRINT TAB(23);"u(q)va"
720 PRINT "BOKSTAV!
      u+v a"
730 PRINT TAB(25);"w  a"
740 FOR I=1 TO 8
750 IF I=6 THEN 790
760 PRINT TAB(28);"a"
770 NEXT I
780 GOTO 810
790 PRINT "DU HAR PROVAT:";
800 GOTO 760
810 PRINT "aaaaaaaaaaaaaaaaj
jjjjjjaaaaa"
820 RETURN
830 REM HUVUD
840 CALL HCHAR(8,21,98)
850 CALL HCHAR(8,22,97)
860 CALL HCHAR(8,23,99)
870 CALL HCHAR(9,21,137)
880 CALL HCHAR(9,22,138)
890 CALL HCHAR(9,23,139)
900 CALL HCHAR(10,21,146)
910 CALL HCHAR(10,22,140)
920 CALL HCHAR(10,23,147)
930 CALL HCHAR(11,22,136)
940 RETURN
950 REM KROPP
960 CALL HCHAR(11,21,122)
970 CALL HCHAR(11,23,121)
980 CALL HCHAR(12,21,128)
990 CALL HCHAR(12,22,130)
1000 CALL HCHAR(12,23,129)
1010 FOR I=13 TO 15
1020 CALL HCHAR(I,21,106,3)
1030 NEXT I
1040 RETURN
1050 REM HOGER ARM
1060 CALL HCHAR(12,24,108)
1070 K=24
1080 FOR I=13 TO 15
1090 CALL HCHAR(I,K,107)
1100 CALL HCHAR(I,K+1,108)
1110 K=K+1
1120 NEXT I
1130 RETURN
1140 REM VANSTER ARM
1150 CALL HCHAR(12,20,104)
1160 K=19
1170 FOR I=13 TO 15
1180 CALL HCHAR(I,K,104)
1190 CALL HCHAR(I,K+1,105)
1200 K=K-1
1210 NEXT I
1220 RETURN
1230 REM HOGER BEN
1240 CALL HCHAR(16,21,131,3)
1250 CALL VCHAR(17,23,131,5)
1260 RETURN
1270 REM VANSTER BEN
1280 CALL VCHAR(17,21,131,5)
1290 RETURN
1300 REM HOGER HAND
1310 CALL HCHAR(16,26,148)
1320 CALL HCHAR(16,27,149)
1330 CALL HCHAR(16,28,150)
1340 CALL HCHAR(17,27,151)
1350 RETURN
1360 REM VANSTER HAND
1370 CALL HCHAR(16,16,153)
1380 CALL HCHAR(16,17,149)
1390 CALL HCHAR(16,18,152)
1400 CALL HCHAR(17,17,151)
1410 RETURN
1420 REM HOGER FOT
1430 CALL HCHAR(22,23,136)
1440 CALL HCHAR(22,24,145)
1450 CALL COLOR(11,2,1)
1460 CALL COLOR(2,2,1)
1470 RETURN
1480 REM VANSTER FOT
1490 CALL HCHAR(22,20,144)
1500 CALL HCHAR(22,21,136)
1510 RETURN
1520 REM RATT ORD
1530 CALL COLOR(2,1,1)
1540 CALL COLOR(11,1,1)
1545 IF LEN(W$)<2 THEN 1560
1550 CALL HCHAR(12,21,106,3)
1560 CALL HCHAR(11,21,32)
1570 CALL HCHAR(11,23,32)
1580 FOR I=7 TO 3 STEP -1
1590 CALL HCHAR(I,22,32)
1600 NEXT I
1605 IF LEN(W$)<1 THEN 1620
1610 CALL HCHAR(10,22,141)
1620 FOR D=1 TO 200
1630 NEXT D
1640 GOTO 1950
1650 REM FEL ORD
1660 CALL HCHAR(10,22,142)
1670 FOR I=1 TO 19
1680 CALL HCHAR(I,1,97,32)
1690 CALL VCHAR(I+1,16,102,3
)
1700 CALL VCHAR(I+3,16,97,2)
1710 NEXT I
1720 CALL HCHAR(23,19,32,7)
1730 CALL HCHAR(20,1,97,32)
1740 CALL HCHAR(23,20,144)
1750 CALL HCHAR(23,21,136)
1760 CALL HCHAR(21,1,97,32)
1770 CALL HCHAR(23,23,136)
1780 CALL HCHAR(23,24,145)
1790 CALL SCREEN(2)
1800 CALL HCHAR(22,1,97,32)
1810 CALL HCHAR(23,1,97,32)
1820 CALL HCHAR(24,1,97,32)
1825 FOR D=1 TO 200
1826 NEXT D
1827 CALL CLEAR
1830 RETURN
1840 REM STYRPROGRAM
```

```
1841 DIM FF$(50)
1843 RESTORE 3000
1844 FOR I=1 TO 50
1845 READ FF$(I)
1846 IF FF$(I)="*" THEN 1848
1847 NEXT I
1848 LIMIT=I-1
1849 CALL CLEAR
1850 MISS=0
1851 CC$=FF$(INT(LIMIT*RND)+
1)
1860 GOSUB 610
1861 CALL HCHAR(7,3,95,LEN(C
C$))
1863 GGR=0
1864 W$=""
1865 U=0
1870 CALL KEY(3,KEY,S)
1880 IF (KEY<65)+(KEY>93)THE
N 1870
1881 FOR I=1 TO LEN(CC$)
1882 IF CHR$(KEY)=SEG$(CC$,I
,1)THEN 3980
1883 NEXT I
1884 IF GGR=LEN(CC$)THEN 152
0
1885 IF U=1 THEN 1865
1887 IF POS(W$,CHR$(KEY),1)T
HEN 1920
1888 W$=W$&CHR$(KEY)
1889 CALL HCHAR(21,3+MISS,KE
Y)
1900 MISS=MISS+1
1910 ON MISS GOSUB 840,960,1
060,1150,1240,1280,1310,1370
,1430,1490
1920 IF MISS<10 THEN 1865
1930 GOSUB 1660
1950 CALL SCREEN(16)
1960 PRINT "SPELA IGEN <J N>
?";
1965 CALL KEY(3,KEY,S)
1966 IF S=0 THEN 1965
1970 IF KEY<>74 THEN 2010
1980 CALL COLOR(2,1,1)
1990 CALL COLOR(11,1,1)
2000 GOTO 1849
2010 CALL CLEAR
2015 END
3000 DATA ADAM,BERTIL,CESAR,
DAVID,ERIK,FILIP,GUSTAV,HELG
E,IVAR,JOHAN
3010 DATA KALLE,LUDVIG,MARTI
N,NIKLAS,OLOF,PETTER,QUINTUS
,RUDOLF,SIGURD,TORE
3020 DATA URBAN,VIKTOR,XERXE
S,YNGVE,ZÄTA,ÅKE,ÄRLIG,ÖSTEN
,*,@
3980 CALL GCHAR(7,2+I,TEST)
3990 IF KEY=TEST THEN 4020
4000 GGR=GGR+1
4010 CALL HCHAR(7,2+I,KEY)
4020 U=1
4030 GOTO 1883
5000 PRINT TAB(10);"DU ÄR HÄ
NGD": : : : :TAB(7);"TRYCK
 PÅ TANGENT": :
5010 CALL KEY(3,KEY,S)
5020 @=RND
5030 IF S=0 THEN 5010
5040 CALL SOUND(99,440,0)
5050 RETURN
```

■

FROM BASIC TO ASSEMBLY – 2

by Bob August, Bug News, USA

Last month I failed to give you the explanation for two commands:
JNE = Jump if not equal
JLT = Jump if less than equal

Also in last months program we had a delay routine to hold the message on the screen. Robbie gave me a better routine that takes less code:

```
       LI   R1,6    FOR R1=1 TO 6
       CLR  R0      R0=0
DELAY  DEC  R0      FOR R0=1 TO 65535
       JNE  DELAY   NEXT R0
       DEC  R1      R1=R1-1
       JNE  DELAY   NEXT R1
```

Here is what it does. First we load Register one with a six. Then we clear Register zero to zero. At the label DELAY we subtract one from Register zero. Register zero now has Hex FFFF or Decimal 65535. The next line says Jump to DELAY if Register zero is not down to zero. The next line subtracts one from Register one. The last line says Jump to DELAY if Register one does not equal zero.

Now for this months program. We clear the screen with a GOSUB CLEAR, put message number one on the screen along with a prompt for a keyboard entry. If we enter a two we will clear the screen and display message number two. The enter key will end the program. Here are the Basic and XB equivalents of the program:

Basic version.

```
100 REM  Print message one
110 GOSUB 270
120 PRINT TAB(4);"THIS IS ME
SSAGE NUMBER ONE"::::::::::: :
130 GOTO 180
140 REM  Print message two
150 GOSUB 270
160 PRINT TAB(4);"THIS IS ME
SSAGE NUMBER TWO"::::::
170 REM  Print message three
180 PRINT "PRESS: 1 For Mess
age # one":TAB(8);"2 for Mes
sage # two"::TAB(8);"Enter k
ey to quit"
190 REM  Call key routine
200 CALL KEY(0,K,S)
210 IF S=0 THEN 200
220 IF K=49 THEN 110
230 IF K=50 THEN 150
240 IF K=13 THEN 250 ELSE 200
250 STOP
260 REM Clear screen routine
270 CALL CLEAR
280 RETURN
```

Extended Basic version.

```
100 ! Print message one
110 GOSUB 190 :: DISPLAY AT(
8,4):"THIS IS MESSAGE NUMBER
 ONE" :: GOTO 150
120 ! Print message two
130 GOSUB 190 :: DISPLAY AT(
14,4):"THIS IS MESSAGE NUMBE
R TWO"
140 ! Print message three
150 DISPLAY AT(20,1):"PRESS:
 1 For Message # one":TAB(8)
;"2 For Message # two": :TAB
(8);"Enter key to quit"
160 ! Call key routine
170 CALL KEY(0,K,S):: IF S=0
 THEN 170 :: IF K=49 THEN 11
0 ELSE IF K=50 THEN 130 ELSE
 IF K<>13 THEN 170 :: STOP
180 ! Call clear routine
190 CALL CLEAR :: RETURN
```

This month we used some new commands that we didn't use last month.

```
KSCAN = Built in key scan command.
B     = Branch
BL    = Branch and link.
CB    = Compare byte
JEQ   = Jump if equal
MOV   = Move from here to there
RT    = Return
```

In our program we used M1 as the label for message #1, M2 as the label for message #2, M3 as the label for message #3 and CLEAR as the label for the Call Clear routine. We have two sections which have a loop in them so we used KLOOP for the key loop routine and CLOOP for the Clear routine. I also think I forgot to tell you last month that any number with a greater than symbol " > " in front of it means that it is a Hex number not a Decimal number. The Hex number >FFFF is equal to Decimal number 65535. In Assembly you can use either the Hex or Decimal number. It's just easier to use >FFFF and if you look at your program you will only find Hex numbers.

Message number three is four lines long so you have to be sure that lines one two and three are 32 characters long by adding spaces in necessary. The last line does not need to be 32 characters long as long as you put the correct count in Register two.

In the key scan routine we first must clear the status bit. This is the S in CALL KEY(0,K,S). We then scan the keyboard and if no key is pressed the command JNE KLOOP (Editor's change) puts the computer into a loop. In this case JNE means jump if not equal to >20. If a key is presed it drops through to the MOV @>8375,R3. >8375 contains the value of the key pressed. We then compare that value with >31 (or 1), >32 (or 2) and >0D (or 13) and jump if equal to either M1 or M2. The JNE KLOOP kills all the other keys if a 1, 2 or enter key is not pressed. If the enter key is pressed then the program falls through to clear the Status bit and exit the program. You must always clear the Status bit at the start of your key scan routine and upon exiting your key scan routine. You will notice that our clear screen routine is the same as last month except we moved our label clear to the start of the routine and added the CLOOP in place of the CLEAR.

I put these routines in a book and when I need to use them I just go to my book and add them to my program. Next month we will show you how to use the Display in a gosub and how to use CALL KEY(3,K,S).

```
************************************
* BASIC TO ASSEMBLY, Lesson Number 2 *
************************************
*
       DEF  START          Entry point label
       REF  VSBW,VMBW,KSCAN,VWTR  E/A Utilities
*
WRKSP  BSS  >20            Allocate workspace
* Messages 1 2 and 3
MSG1   TEXT 'THIS IS MESSAGE NUMBER ONE'
MSG2   TEXT 'THIS IS MESSAGE NUMBER TWO'
MSG3   TEXT 'PRESS: 1 For Message number one '
       TEXT '       2 For Message number two '
       TEXT '                                '
       TEXT '       Enter key to quit'
*
       EVEN                Make sure its even
*
START  LWPI WRKSP          Load your workspace
* Print message one - DISPLAY AT(8,4)
M1     BL   @CLEAR         GOSUB CLEAR
       LI   R0,227         Row 8 Col 4
       LI   R1,MSG1        Message #1
       LI   R2,26          Length of message
       BLWP @VMBW          Write it to the screen
       B    @M3            GOTO M3
* Print message two - DISPLAY AT(14,4)
M2     BL   @CLEAR         GOSUB CLEAR
       LI   R0,419         Row 14 Col 4
       LI   R1,MSG2        Message #2
       LI   R2,26          Length of the message
       BLWP @VMBW          Write it to the screen
* Print message three - DISPLAY AT(20,1)
M3     LI   R0,608         Row 20 Col 1
       LI   R1,MSG3        Message #3
       LI   R2,120         Length of message
       BLWP @VMBW          Write it to the screen
* Key scan routine - CALL KEY(0,K,S)
       CLR  @>837C         Clear status
       LI   R4,>2000                                  (Ed. change)
KLOOP  BLWP @KSCAN         Scan keyboard routine
       CB   @>837C,R4      Check status for key press (Ed. change)
       JNE  KLOOP          IF S=0 THEN KLOOP          (Ed. change)
       MOV  @>8375,R3      Move Key value to R3
       CI   R3,>31         Is it 49 (ASCII 1)
       JEQ  M1             If K=1 then M1
       CI   R3,>32         is it 50 (ASCII 2)
       JEQ  M2             If K=2 then M2
       CI   R3,>0D         Is it 13 (Enter key)
       JNE  KLOOP          If K<>13 then KLOOP
       CLR  @>837C         Clear status
       BLWP @0             FCTN Quit
* Clear screen routine - CALL CLEAR
CLEAR  LI   R1,>2000       R1=32
       CLR  R0             Make R0 zero
CLOOP  BLWP @VSBW          Put blank space on screen
       INC  R0             R0=R0+1
       CI   R0,767         Is R0=767
       JLE  CLOOP          if < 767 goto cloop
       RT                  Return
* End program with auto start
       END  START                                     ■
```

SWEDLOW TI BITS * 24-25 *

by Jim Swedlow, USA

(This article originally appeared in the User Group of Orange County, California ROM)

EA DISK ERROR CODES

Some programs (like Archiver) display a number when they encounter a disk error (something like "IO ERROR #7"). The numbers by themselves are of little use. Here is what they mean:

0 UNKNOWN DEVICE - Could not find the specified drive.

1 WRITE PROTECTED - The disk is write protected.

2 BAD OPEN ATTRIBUTE - One or more OPEN options were illegal or didn't match the file characteristics.

3 ILLEGAL OPERATION - The book says that this code should not be generated!

4 OUT OF SPACE - The disk is full or you are trying to open more files than are allowed (127).

5 END OF FILE - You are trying to read beyond the end of the file.

6 DEVICE ERROR - The disk is not initialized, the disk is damaged, the disk drive is broken (oh no!), the drive door is open, etc.

7 FILE ERROR - The file doesn't exist or you are trying to read a BASIC file as if it were data.

These are the same as the second digit in the BASIC disk error codes.

MAGIC NINES

A fun program by Jim Peterson of the TIGERCUB.

```
100 ! MAGIC NINES
      By Jim Peterson
110 CALL CLEAR
120 PRINT ::
    PRINT "ENTER ANY 3 DIGIT NUMBER"
    :"WITH 3 DIFFERENT DIGITS"
130 INPUT N ::
    PRINT ::
    IF N<>INT(N) OR N>999 OR N<100
    THEN 120 ELSE N$=STR$(N)
140 IF SEG$(N$,1,1)=SEG$(N$,2,1) OR
    SEG$(N$,1,1)=SEG$(N$,3,1) OR
    SEG$(N$,2,1)=SEG$(N$,3,1) THEN
    PRINT ">USE 3 DIFFERENT DIGITS<"
    :: GOTO 120
150 N2$,N4$="" ::
    FOR J=1 TO 3 ::
    N2$=SEG$(N$,J,1)&N2$ ::
    NEXT J
160 N2=VAL(N2$) ::
    N3=ABS(N-N2)
170 PRINT N$;" BACKWARDS IS ";N2$ ::
    PRINT
180 N3$=STR$(N3) ::
    IF N3<100 THEN N3$="0"&N3$
190 IF N>N2 THEN PRINT
    N$;" MINUS ";N2$;" IS ";N3$
    ELSE PRINT
    N2$;" MINUS ";N$;" IS ";N3$
200 PRINT ::
    FOR J=1 TO 3 ::
    N4$=SEG$(N3$,J,1)&N4$ ::
    NEXT J
210 PRINT N3$;" BACKWARDS IS ";N4$:
    :N3$;" PLUS  ";N4$;" IS 1089": :
220 PRINT :"I KNEW THAT WOULD BE": :
    "THE ANSWER.  LIST THE": :
    "PROGRAM AND SEE!"
230 !!!!!!!!!!!!!!!!!!!!
240 !   THE ANSWER IS   !
250 !       1089        !
260 !!!!!!!!!!!!!!!!!!!!
```

TI LIVES

The December, 1988 issue of PC Computing (a magazine normally dedicated to MS DOS machines) has an article entitled "Gone But Not Forgotten". There is coverage of most of the orphans (TI, Osborne, Eagle, PCjr, etc). A couple of interesting quotes:

"The 99/4A and PCjr were early experiments in the home computing market. They weren't nearly as fast or powerful as other computers of their time, yet in many homes they continue to fulfill the role that

visionaries once predicted for them: they've evolved from somber, cold pieces of machinery to tools that are useful and fun."

"Fans of the TI 99/4A are legion, and they form a network of users that's as lively as FOG, if less structured. TI user groups number 300 with the Chicago group the largest at nearly 550 members. Other TI organizations are small but spirited. The 14 member north New Jersey group works with other New York area computer groups to sponsor the TI Computer Fest which each year draws about 300 people to attend workshops and hear speakers."

ÄFOG is an international User Group for owners of CPM and MS DOS machinesÄ

TI WRITER MARGINS

For the first item in this column, I used a device called a hanging indent. It looks like this:

The first line is set to the left margin but subsequent lines are indented. It could be called the opposite of normal paragraph indentation.

To do this, you must alter the left margin (.LM) and indentation (.IN) settings. For this column, the initial formatter settings were:

```
.FI;AD;LM 1;RM 40;IN +0
```

This tells the formatter to FIll, ADjust, set the Left Margin at 1, the Right Margin at 40 and NOT to INdent paragraphs.

When I wanted to start the hanging indent, I added this command:

```
.LM +3;IN -3
```

The "+3" after ".LM" tells the Formatter to move the left margin three spaces to the RIGHT. I could have said ".LM 4" but by using the plus sign I don't need to know the previous margin setting. In the same manner, the "-3" INdent tells the formatter to set paragraph indentation three spaces to the LEFT of the left margin.

When I wanted to revert to the original settings, I used this:

```
.LM -3;IN +0
```

The -3 after ‹.LM› moves the left margin three spaces to the LEFT or back where it was before the ‹.LM +3› command. The ‹.IN +0› command cancels the hanging indent (i.e., it tells the Formatter to stop indenting). The + or - in the INdent command is releative to the current left margin.

Notice that I combined a number of formatter commands on the same line. They are separated with semi-colons. This works for most (but not all) formatter commands.

A VIRUS IN TI LAND?

I doubt it.

A computer virus attaches itself to a program and then hides. It moves from program to program, hidden until it starts doing whatever it does.

Some virus programs display humorous and/or obscene messages. Others destroy data. By definition, a virus must exist in an electronic environment where it has programs to move among. This happens on a main frame or a hard disk.

Since our TI's are mainly disk based, there should be no where for the virus to spread.

A real danger, however, is a trojan horse. This little love is a program that is supposed to do one thing but actually does another - like reformat your disk or make the drive jump around.

Some simple precautions will keep you safe. When you get a new program, run it on a disk by itself. Keep all other drives empty. If anything unusual happens, shut your system off.

If you find a bad program and it came from a bulletin board, call the

Sysop IMMEDIATELY. (Sysop is "bulletin boardese" for system operator.)

BOOT TRICK

Want to change the colors on your BOOT screen? This is not in the documentation, but pushing <B> changes the Background color and <F> the Foreground. Use <FCTN 5> to save your changes.

TI v. IBM

No, this is not one of those bash the big blue monster items. Nor is it a lament for the end of the 4A. Rather, some thoughts on reality.

First, IBM compatible computers exist and are widely used. I don't want to get into the IBM v. MAC argument here. The point is that these computers are used and supported in offices and software stores. Our 4A isn't.

Another fact - many TI users have left the 4A and gone to other platforms. These defections have left those of us who continue to use our 4A's gun shy.

Some have suggested that TI user groups form IBM SIG's (Special Interest Groups). This has not been well received and I think wisely so. There is strong support for IBM compatibles. It would be a mistake for us to dilute our energies by diving them.

On the other hand, ignoring the real world would also be a mistake. Some of us use IBM compatibles at work. Others own one. This does not mean the end of the 4A. Our TI continues to be viable. It wasn't, it would have gone the way of Adam and other lesser contenders.

We cannot stick our heads in the sand and pretend that we are alone. We aren't. We can, however, make sure that the TI continues to get the support it deserves. One way is to support those who live in a two computer world.

From time to time, then, you will see some discussion here about working with a TI and another computer. I hope that this will be taken in the spirit that it intended. Not to wean folks away but to keep hands on those 48 keys.

MAGIC FILE MANIPULATOR

One vital necessity when you use more than one operating system is the ability to move files back and forth. If you have this need, MAGIC FILE MANIPULATOR (MAGICFM) is the program for you. You need the following:

- A TI system with disk, RS232 and 32k.

- Extended BASIC.

- A communications (modem) program that performs XMODEM file transfers for your other computer.

- A null modem cable.

The MAGICFM documentation includes pin outs for the null modem cable. Since it only uses three wires, it is easy to make.

You boot up MAGICFM on your TI and your communications program on your other computer. Then, from the other computer keyboard, you can:

- Catalog any disk on your TI.

- Delete, Protect, Unprotect or Rename any file on your TI.

- View any DV80 file.

- Transfer (via XMODEM) any file from your TI to the other computer OR from the other computer to your TI.

One of the things that make this program magic is that the file transfers work at the fantastic speed of 19,200 baud. Files fly over the null modem cable.

If you think you need this program, you do. It solves any number of problems.

Kudos to Ben Hatheway who wrote MAGICFM. Outstanding!

UNSUNG HEROES

The February, 1989 issue of PC Computing contained an article subtitled:

"Sometimes being great isn't good enough. Witness the fate of ten products that deserved better".

Guess what one of the ten was.

"TI 99/4A: Texas Instruments' 16 bit home computer was fast, expandable, cheap and an early victim of 'dinosaur marketing.'"

Enjoy. ■

FAST EXTENDED BASIC! 88/06 - STAMPDBASE (part2)

(c) 1989 Lucie Dorais, Ottawa TI-99/4A Users' Group, Canada

As promised in last issue, here is the SORT module of my STAMP DATA BASE program.

To get to the SORT function, you must press the "*" from the Menu; why, do you ask, did I choose that character? Well, the [S] was already taken by the save function; also, since sorting is a time-consuming operation, better use a key that you would not press by accident!

The first thing we do, in line 390, is to store the value of N (record on screen) into TN, temporary N: we will need it if we make a mistake and want to get out of the sorting input screen. Line 400 displays the instructions for the sort function.

Line 410 uses the CALL A sub to tell Tex which field to sort on; you can sort only one field at a time. We have to use the simpler CALL A because it is the only one that accepts the asterisk as a character... Since all our fields are not useful to sort on, we use only three CALL As: for Country name, Date, or Color; we just repeat the parameters from lines 230-240 of the main program. Just enter "*"s (only one will do as well) in the field to sort on, nothing in the other two; if you put an asterisk in more than one field, the program will sort on the first one only.

Line 420 uses record #0, i.e. S$(0), to build the "sort string", identical to the data ones, so it will be easy for Tex to determine which part of each string to sort. Note that we replace the unused fields by spaces; since three consecutive fields have four characters in size, we use the RPT$ statement. Compare the sort string (Color is the field to sort on) with a real data one:

Rec	Country	date	Sc#	Pic	Face	Color	S	Val.
0						***		
15	CANADA	1952	320	A136	7	BLUE	N	40

We then get the position of the first (or only)"*" in the sorting string; if none, then back to menu, otherwise GO SORT! In line 440, we look for the first delimitator "|" after the asterisk, so that Tex does not kill itself sorting more characters than it should. We then calculate S, size of the string to sort. D$ is a full line of tildes; Tex will replace each deleted record with it; when we sort on any field, the deleted records, being all "~"s (with an ASCII number above any alphabetical character) will all be placed at the end of the list, and the "packing", i.e. the actual deleting, will be much easier to do. K=1 is the beginning of the sort function proper. Note that I re-use the K variable (usually used for KEY), and I also re-used the S

(normally STATUS), since both are not needed in that module of the program: this saves memory, but I admit it can be confusing.

The sorting function used here, which ends at line 500, is called the SHELL SORT, after the man who designed the algorithm. I took it from Regena's "Guide to the TI-99/4A", page 295; she explains how it works, but I admit I do not understand it all, so just have faith, it works! It is a binary sort, that breaks the main list into smaller ones, easier to deal with. It is not instantaneous, since 50 records take about one minute to sort, but much faster that the BUBBLE one.

Now that all our records are sorted, we have two problems: the deleted ones are still there, although now placed at the end of the list; and our record numbers, part of the data strings, are all mixed up. So, lines 510-520 do two things: if the next data string, S$(X), is a deleted one (it contains tildes), get out of the re-numbering function, but do not forget to bring the total T down to the last good data (X-1). If, on the other hand, it is still valid, renumber it sequentially by replacing the first three characters by the correct record number. When all is done, go back to menu and show record number N=1.

Since I still have a lot of room to write, I added another feature to the 88/05 program: when you want to [G]et a record, it is sometimes easier to find data than to remember the correct record number, so I added more lines in the Get function (see lines 350-356).

Sorry, there was no room to add another item to the menu, so Tex will always ask you to choose between Record number or Data, the latter being the default. Again, we need to check the Ascii value because of the automatic addition of the delimitator in the CALL AS sub. If we press "R", we go to line 356.

Lines 352 asks for the data to get; since the CALL A sub adds a "|" at the end of the string, we would have had to strip it, a tedious programming job, so we use a straightforward ACCEPT statement. Starting with record Z=1, Tex will look for our string; when it finds something, it displays that record by giving the value of the X counter to N before exiting the loop. It then asks you if it is the record you want; if not, the value of Z is incrememted by one and given to the X loop counter to continue the search. If it reached the end, that is the NEXT X has run its course, it displays a brief message, then goes back to the menu; note that variable X is reused for the delay counter.

About line 355: the statement "IF P" is just a quick way to say "IF P<>0", very useful in a case like here where there are only two courses to take, whether P is equal to "0" (return to menu) or not (get the record number P).

Do I still have room? How about a module to find out the total value of your collection? Truly, this is NOT the best way to write a program: everything should be planned in advance. Anyway, this is the last column for this year, so please forgive me for all those after thoughts... And, as a summer project, how about reworking the program so that "GET DATA" and "COUNT TOTAL VALUE" can be called directly from the main menu? (lines 850-855).

To count the total value Z, we get that portion of each entry that corresponds to the catalogue value; that is found starting at character 61 of the string, and has a maximum of four characters. We then use the VAL function of Tex to get the numeric value (Y) of the string. Since your values can be entered in more than one way, for ex. 50 cents can be 50 or .50, and one dollar can be 1.00 or 100, the next statement looks for a period (just be consistent when you enter your data). If there is one, multiply by 100.

You will actually see the total growing as each value is added to the total; of course, we have to divide it by ten before showing it. When Tex reaches the last record (T), you can contemplate your fortune until you press a key. ■

PROGRAMMING MUSIC - PART 5

by Jim Peterson, Tigercub, USA

In previous installments I have shown you how to program music by an easy method which requires you to specify a duration or a frequency only when it changes from one note to the next. Now, here is an even easier method - autochording.

With this method, you do not have to key in the accompaniment - you just specify the chord and GOSUB to the proper line to play the type of chord.

Almost all sheet music has guitar chords printed above the upper staff - those little 6x4 grids with black dots on them. And those guitar chords are always labeled with the name of the chord they represent.

The most common chord is a major chord, represented by a letter - A, C or whatever, or a letter followed by a flat or sharp sign. For those, use GOSUB 1000. The second most common chord is the 7th chord, which has the letter followed by a 7, such as C7. For those, GOSUB 1100.

You might come across a minor chord, denoted by a small m after the letter, such as Cm. In that case, GOSUB 1200.

And for a minor 7th, such as Cm7, GOSUB 1300.

There are many more complex chords, but I have not tried to allow for them all in this easy method. If you come to one of them, just try playing on through with the previous chord - it will usually sound alright.

To program music in this way, use the scale that I showed you in Part 1, but you will probably have to set the starting frequency considerably higher than 110. Merge in one or the other of the following routines, then program the music just as I showed you before, but only A and B. Give A the number for the melody and B the number for the chord, then GOSUB to the proper line number for that type of chord. If the next note does not have a guitar chord above it, it is the same chord so you do not have to give B a value again, just GOSUB to the same line number.

Now, here is the first routine, to play simple harmony. Let me give you a tip to save you some time. When you are keying in a series of program lines which are all nearly the same, key in the first one, Enter it, then use FCTN 8 to bring it back to the screen. Use the editing keys to change the line number and make other necessary changes, Enter it, use FCTN 8 to bring it back, etc.

```
110 D=3 :: V1=1 :: V2=9 :: V
3=9
1000 X=X+1+(X=4)*4 :: ON X G
OSUB 1010,1020,1030,1040 ::
RETURN
1010 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B),V2,N
(B)/1.585,V3):: NEXT J :: RE
TURN
1020 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B),V2,N
(B)/1.334,V3):: NEXT J :: RE
TURN
1030 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B),V2,N
(B)/2,V3):: NEXT J :: RETURN
1040 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B)/1.58
5,V2,N(B)/1.334,V3):: NEXT J
 :: RETURN
1100 X=X+1+(X=9)*4 :: ON X G
OSUB 1110,1120,1130,1140 ::
RETURN
1110 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B)/1.49
7,V2,N(B)/1.585,V3):: NEXT J
 :: RETURN
1120 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B)/1.49
7,V2,N(B)/1.334,V3):: NEXT J
 :: RETURN
1130 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B)/1.49
7,V2,N(B)/2,V3):: NEXT J ::
RETURN
1140 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B)/1.58
5,V2,N(B)/1.334,V3):: NEXT J
```

```
 :: RETURN
1200 X=X+1+(X=4)*4 :: ON X G
OSUB 1210,1220,1230,1240 ::
RETURN
1210 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B),V2,N
(B)/1.679,V3):: NEXT J :: RE
TURN
1220 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B),V2,N
(B)/1.334,V3):: NEXT J :: RE
TURN
1230 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B),V2,N
(B)/2,V3):: NEXT J :: RETURN
1240 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B)/1.67
9,V2,N(B)/1.334,V3):: NEXT J
 :: RETURN
1300 X=X+1+(X=4)*4 :: ON X G
OSUB 1310,1320,1330,1340 ::
RETURN
1310 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B)/1.49
7,V2,N(B)/1.679,V3):: NEXT J
 :: RETURN
1320 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B)/1.49
7,V2,N(B)/1.334,V3):: NEXT J
 :: RETURN
1330 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B)/1.49
7,V2,N(B)/2,V3):: NEXT J ::
RETURN
1340 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V1,N(B)/1.67
9,V2,N(B)/1.334,V3):: NEXT J
 :: RETURN
```

That routine will play straight 3-part harmony, but I like this one better, although it does not work well with some pieces.

```
110 D=30 :: S=1 :: V1=1 :: V
2=5 :: V3=7
1000 FOR J=1 TO T :: X=X+1+(
X=4)*4 :: ON X GOSUB 1010,10
20,1030,1040 :: GOSUB 2000 :
: NEXT J :: RETURN
1010 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B),V3):: RET
URN
1020 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/1.585,V3)
:: RETURN
1030 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/1.334,V3)
:: RETURN
1040 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/2,V3):: R
ETURN
1100 FOR J=1 TO T :: X=X+1+(
X=4)*4 :: ON X GOSUB 1110,11
20,1130,1140 :: GOSUB 2000 :
: NEXT J :: RETURN
1110 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B),V3):: RET
URN
1120 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/1.585,V3)
:: RETURN
1130 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/1.334,V3)
:: RETURN
1140 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/1.497,V3)
:: RETURN
1200 FOR J=1 TO T :: X=X+1+(
X=4)*4 :: ON X GOSUB 1110,11
20,1130,1140 :: GOSUB 2000 :
: NEXT J :: RETURN
1210 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B),V3):: RET
URN
1220 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/1.679,V3)
:: RETURN
1230 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/1.334,V3)
:: RETURN
1240 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/2,V3):: R
ETURN
1300 FOR J=1 TO T :: X=X+1+(
X=4)*4 :: ON X GOSUB 1110,11
20,1130,1140 :: GOSUB 2000 :
: NEXT J :: RETURN
1310 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B),V3):: RET
URN
1320 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/1.679,V3)
:: RETURN
1330 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/1.334,V3)
:: RETURN
1340 CALL SOUND(-999,N(A),V1
,N(A)*1.01,V1,N(B)/1.497,V3)
:: RETURN
2000 FOR Y=1 TO D :: NEXT Y
:: RETURN
```

Both of those routines cycle through four inversions of the chord, to avoid a monotonous drone.

There are many ways to vary those routines. Just for instance, right after each N(B) put *2 to raise the harmony above the melody. Also try *4. Or alternate *2 and *4. Experiment! Have fun! ■

TIPS FROM TIGERCUB #64

Tigercub Software
156 Collingwood Ave.
Columbus, OH 43213, USA

****

My three Nuts & Bolts disks, each containing 100 or more subprograms, have been reduced to $5.00. I am out of printed documentation so it will be supplied on on disk.

My TI-PD library now has over 500 disks of fair-ware (by author's permission only) and public domain, all arranged by category and as full as possible, provided with loaders by full program name rather than filename, Basic programs converted to XBasic, etc. The price is just $1.50 per disk(!), post paid if at least eight are ordered. TI-PD catalog #5 will probably be printed by the time these Tips appear, and is available for $1 which is deductible from the first order.

Back in the days of David Ahl's Creative Computing magazine, when computers were to expensive for hard-ware hacking and had memory too small to run much of a program, the emphasis was on "recreational computing", and the British TI'ers carry on that tradition. A recent issue of their excellent TI*MES newsletter had this challenge - write a program to set up a circle of any chosen number of objects; starting at one, count them off by 10's, removing every 10th object. What are the numbers of the last two left?

This is my solution. It is not the best one, but it does show how strings can be used to perform math.

```
100 INPUT "NUMBER?":N
110 FOR J=1 TO N :: N$=N$&CH
R$(J):: NEXT J :: IF N<10 TH
EN 140
120 N$=SEG$(N$,11,255)&SEG$(
N$,1,9):: IF LEN(N$)>9 THEN
120
140 FOR J=1 TO 10 :: N$=SEG$
(N$,2,255)&SEG$(N$,1,1):: NE
XT J :: N$=SEG$(N$,1,LEN(N$)
-1):: IF LEN(N$)>2 THEN 140
150 FOR J=1 TO 2 :: PRINT AS
C(SEG$(N$,J,1)):: NEXT J
```

Which reminds me that I forgot to give you the answer to that short CALL SOUND puzzler in Tips #62. A CALL SOUND, even with a pos-itive duration, will be in-terrupted by a BEEP.

Here's a bit of nonsense I worked up from an idea by Tim Brooks. Save this by SAVE DSK1.BUGS,MERGE . Then when you get a chance, load one of your friend's favor-ite programs, add this to it by MERGE DSK1.BUGS, and in the middle of the program somewhere put a line with CALL BUGS .

```
32000 !@P+
32001 SUB BUGS
32002 CALL CLEAR :: CALL CHA
RSET :: CALL DELSPRITE(ALL):
: CALL SOUND(225,220,0):: PR
INT "*ERROR 4 IN LINE 150" :
: PRINT "*BUGS IN PROGRAM"
32003 CALL SCREEN(8):: FOR A
=1 TO 500 :: NEXT A :: A$(1)
="997E3CFF3C7EBD99" :: A$(2)
="DB3CBD7E3CFFBD99" :: X=1 :
: CALL CHAR(96,A$(X))
32004 RANDOMIZE :: CALL MAGN
IFY(2):: FOR T=1 TO 2 :: FOR
 A=1 TO 20 :: X=X+1+(X=2)*2
:: CALL CHAR(96,A$(X)):: FOR
 D=1 TO 20 :: NEXT D
32005 CALL SPRITE(#A,96,2,19
5,RND*240,-5,0):: NEXT A ::
NEXT T :: CALL CLEAR :: CALL
 DELSPRITE(ALL):: SUBEND
```

Here is a puzzle game for you brainy types. I worked it up from a game by Jack Sughrue -

```
100 ! PSYCHO by Jim Peterson
110 CALL CLEAR :: RANDOMIZE
:: CALL SCREEN(2):: FOR S=1
TO 12 :: CALL COLOR(S,2,16):
: NEXT S :: CALL VCHAR(1,31,
31,96):: CALL KEY(3,K,S)
120 RANDOMIZE :: Y$(1)="+" :
: Y$(2)="-" :: Y$(3)="x" ::
Y$(4)="/"
130 CALL VCHAR(1,3,32,672)::
 D$="" :: Y(0),X=INT(10*RND+
5)
140 DISPLAY AT(2,11):"PSYCHO
":" Enter P(lus), (M)inus,
   (T)imes or (D)ivided by"
150 FOR J=1 TO 4 :: Y(J)=INT
(10*RND+5):: Z(J-1)=INT(4*RN
D+1)
160 IF Z(J-1)=1 THEN X=X+Y(J
):: GOTO 180 ELSE IF Z(J-1)=
2 THEN X=X-Y(J):: GOTO 180 E
LSE IF Z(J-1)=3 THEN X=X*Y(J
):: GOTO 180
170 IF X/Y(J)=INT(X/Y(J))THE
N X=X/Y(J)ELSE Z(J-1)=INT(3*
RND+1):: GOTO 160
180 NEXT J :: R=6 :: FOR J=0
 TO 3 :: DISPLAY AT(R,12):Y(
J):: R=R+2 :: NEXT J :: DISP
LAY AT(R,12):Y(4)
190 DISPLAY AT(R+1,12):"____
" :: DISPLAY AT(R+3,12):X
200 FOR J=0 TO 3 :: D$=D$&ST
R$(Y(J))&Y$(Z(J)):: NEXT J :
: D$=D$&STR$(Y(4))&"="&STR$(
X):: FOR J=1 TO 4
210 ACCEPT AT(J*2+5,12)SIZE(
1)VALIDATE("PMTD"):A$ :: IF
A$="" THEN 210
220 ON POS("PMTD",A$,1)GOSUB
 270,280,290,300
230 DISPLAY AT(J*2+4,12):""
:: DISPLAY AT(J*2+6,12):Y(J)
240 NEXT J
250 IF Y(4)=X THEN DISPLAY A
T(19,9):"RIGHT!" :: GOTO 260
 ELSE DISPLAY AT(19,9):"  WR
ONG! OFF BY";ABS(X-Y(4)):: D
ISPLAY AT(21,3):D$
260 DISPLAY AT(23,2):"PLAY A
GAIN? Y/N" :: ACCEPT AT(23,1
8)SIZE(1)VALIDATE("YN"):Q$ :
: IF Q$="N" THEN CALL CLEAR
:: STOP ELSE 130
270 Y(J)=Y(J-1)+Y(J):: RETUR
N
280 Y(J)=Y(J-1)-Y(J):: RETUR
N
290 Y(J)=Y(J-1)*Y(J):: RETUR
N
300 Y(J)=Y(J-1)/Y(J):: RETUR
N
```

Someone uploaded the New Testament books of the Bible to DElphi, probably ported over from IBM files. They included a program to break them into individual verses and another to display them on screen. Neither program worked properly, so I wrote this one to do it right.

```
100 CALL CLEAR :: CALL SCREE
N(16):: FOR J=1 TO 12 :: CAL
L COLOR(J,2,1):: NEXT J :: D
ISPLAY AT(2,8):"BIBLE READER
" !by Jim Peterson
110 DIM I$(127),L$(24)
120 DISPLAY AT(24,1):"DRIVE
#?" :: ACCEPT AT(24,10)VALID
ATE(DIGIT)SIZE(1)BEEP:DN ::
CALL CLEAR :: ON WARNING NEX
T
130 X=0 :: OPEN #1:"DSK"&STR
$(DN)&".",INPUT ,RELATIVE,IN
TERNAL :: INPUT #1:N$,A,A,A
140 INPUT #1:F$,A,B,C :: IF
LEN(F$)=0 THEN 160
150 IF C=80 AND ABS(A)=2 THE
N X=X+1 :: I$(X)=F$ :: DISPL
AY AT(X+(X>23)*23,1-(X>23)*1
3):STR$(X);" ";I$(X):: GOTO
140 ELSE 140
160 DISPLAY AT(23,1):"Read f
ile #" :: ACCEPT AT(23,12)VA
LIDATE(DIGIT):FL :: IF FL<1
OR FL>X THEN 160
170 CLOSE #1 :: OPEN #1:"DSK
"&STR$(DN)&"."&I$(FL),INPUT
:: CALL CLEAR :: DISPLAY AT(
3,1):"Press any key at the b
eep" :: X=0
180 LINPUT #1:M$
190 IF POS(SEG$(M$,1,5),":",
1)=0 THEN 220
200 IF FLAG=0 THEN FLAG=1 ::
 GOTO 220
210 X$=M$ :: GOTO 250
220 IF T$<>"" THEN M$=T$&" "
&M$ :: T$="" :: GOSUB 320 EL
SE GOSUB 320
230 IF LEN(T$)>27 THEN M$=T$
 :: T$="" :: GOSUB 320 :: GO
TO 230
240 IF EOF(1)<>1 THEN 180
250 IF T$<>"" THEN X=X+1 ::
L$(X)=T$ :: T$=""
260 CALL SOUND(1,500,8)
270 CALL KEY(0,K,S):: IF S=0
 THEN 270
280 FOR J=1 TO X :: DISPLAY
AT(9+J,1):L$(J):: NEXT J ::
FOR J=10+X TO 24 :: DISPLAY
```

```
AT(J,1):"" :: NEXT J :: X=0
290 IF X$<>"" THEN M$=X$ ::
X$="" :: GOSUB 320 :: GOTO 2
30
300 IF EOF(1)<>1 THEN 180 EL
SE IF X>0 THEN 250 ELSE CLOS
E #1 :: CALL SOUND(1,500,5)
310 CALL KEY(0,K,S):: IF S=0
 THEN 310 ELSE 100
320 IF LEN(M$)<29 THEN X=X+1
 :: L$(X)=M$ :: RETURN
330 IF SEG$(M$,28,1)=" " THE
N X=X+1 :: L$(X)=SEG$(M$,1,2
8):: T$=SEG$(M$,29,255):: RE
TURN
340 IF SEG$(M$,29,1)=" " THE
N X=X+1 :: L$(X)=SEG$(M$,1,2
8):: T$=SEG$(M$,30,255):: RE
TURN
350 P=27
360 IF SEG$(M$,P,1)=" " THEN
 X=X+1 :: L$(X)=SEG$(M$,1,P-
1):: T$=SEG$(M$,P+1,255):: R
ETURN
370 P=P-1 :: IF P>1 THEN 360
 ELSE X=X+1 :: L$(X)=SEG$(M$
,1,28):: T$=SEG$(M$,29,255):
: RETURN
```

Files ported over from IBM lack carriage returns, which can be a problem if you want to do any editing. I think this tinygram will do a good job of adding CRs to any text file which has centered headers and indented paragraphs.

```
100 DISPLAY AT(3,4)ERASE ALL
:"CARRIAGE RETURN ADDER":"":
"  This tinygram program wil
ladd carriage returns to any
 text file which has centere
d"
110 DISPLAY AT(8,1):"headers
 and indented para-  graphs.
"
120 DISPLAY AT(12,1):"Input
filename?":"DSK" :: ACCEPT A
T(13,4):IF$
130 DISPLAY AT(15,1):"Output
 filename?":"DSK" :: ACCEPT
AT(16,4):OF$
140 OPEN #1:"DSK"&IF$,INPUT
:: OPEN #2:"DSK"&OF$,OUTPUT
150 LINPUT #1:M$
160 IF M$="" THEN PRINT #2:C
HR$(13):M$;ELSE IF ASC(M$)<3
3 THEN PRINT #2:CHR$(13):M$;
ELSE PRINT #2:"":M$;
170 IF EOF(1)<>1 THEN 150 EL
SE CLOSE #1 :: CLOSE #2
```

Note that the program does all its work in line 160!

When text files are reformatted to a shorter line length, using the Funlweb Formatter, there are often long gaps at the ends of the lines, or between words if Fill and Adjust is used, due to long words which would have been hyphenated if the text had been originally typed in the shorter length. This little program will reformat text (containing carriage returns) to any shorter length and allow you to optionally hyphenate words which do not fit at the end of a line.

```
100 CALL CLEAR :: CALL SCREE
N(5):: FOR SET=0 TO 12 :: CA
LL COLOR(SET,2,16):: NEXT SE
T
110 CALL CLEAR
120 DISPLAY AT(12,1):"Input
filename?":"DSK" :: ACCEPT A
T(13,4)BEEP:IF$ :: OPEN #1:"
DSK"&IF$,INPUT
130 DISPLAY AT(15,1):"Output
 filename?":"DSK" :: ACCEPT
AT(16,4)BEEP:OF$ :: OPEN #2:
"DSK"&OF$,OUTPUT
140 DISPLAY AT(18,1):"Reform
at to what length?" :: ACCEP
T AT(18,26)SIZE(2)VALIDATE(D
IGIT):R
150 IF EOF(1)THEN 270 :: CAL
L CLEAR :: LINPUT #1:M$ :: M
$=P$&M$ :: P$=""
160 L=LEN(M$)+(POS(M$,CHR$(1
3),1)<>0):: IF L<=R AND POS(
M$,CHR$(13),1)<>0 THEN PRINT
 #2:M$ :: GOTO 150 ELSE IF L
<R THEN P$=M$&" " :: GOTO 15
0
170 C$=SEG$(M$,1,R):: CALL L
ASTPOS(C$," ",Y)
180 IF Y<>0 THEN 190 ELSE PR
INT #2:C$ :: M$=SEG$(M$,R+1,
255):: GOTO 160
190 IF R-Y<3 THEN C$=SEG$(M$
,1,Y):: M$=SEG$(M$,Y+1,255):
: PRINT #2:C$ :: GOTO 160
200 X=POS(M$," ",Y+1):: IF X
=0 THEN X=LEN(M$)ELSE IF X=R
+1 THEN PRINT #2:C$ :: M$=SE
G$(M$,Y+2,255):: GOTO 160
```

```
210 DISPLAY AT(2,1):M$ :: DI
SPLAY AT(8,1):SEG$(M$,1,R)
220 DISPLAY AT(12,1):SEG$(M$
,Y+1,R-Y-1)&"-"&SEG$(M$,R,X-
R+1):: Z=R-Y
230 DISPLAY AT(15,1):"Hyphen
ate?" :: ACCEPT AT(15,12)SIZ
E(1)VALIDATE("YNyn"):Q$ :: I
F Q$="N" OR Q$="n" THEN 260
240 ACCEPT AT(18,1)SIZE(Z):H
$ :: IF POS(H$,"-",1)=0 THEN
 240
250 C$=SEG$(C$,1,Y)&H$ :: M$
=SEG$(M$,Y+1+LEN(H$)-1,255):
: PRINT #2:C$ :: GOTO 160
260 PRINT #2:SEG$(C$,1,Y)::
M$=SEG$(M$,Y+1,255):: GOTO 1
60
270 CLOSE #1 :: CLOSE #2 ::
STOP
280 SUB LASTPOS(A$,B$,Y):: X
,Y=0
290 X=POS(A$,B$,X+1):: IF X>
0 THEN Y=X :: GOTO 290
300 SUBEND
```

I really think that all program listings should be published in 28-column format, as my Tips from the Tigercub have always been published, because that is how they appear on screen, making it much easier to key them in accurately. However, if you absolutely MUST reformat them, I think that this program will accurately reformat to/from any length up to 79 PROVIDING that you first put a carriage return at the end of every program line.

```
100 DISPLAY AT(3,6)ERASE ALL
:"PROGRAM RELISTER":"":"  Wi
ll reformat a LISTed    XBas
ic program from any lineleng
th to any other length."
110 DISPLAY AT(8,1):"  Each
program line (not    file li
ne) must end in a    carriag
e return."
120 DISPLAY AT(12,1):"Input
filename?":"DSK" :: ACCEPT A
T(13,4):IF$ :: DISPLAY AT(15
,1):"Output filename?":"DSK"
 :: ACCEPT AT(16,4):OF$
130 DISPLAY AT(18,1):"Presen
t line length?" :: ACCEPT AT
(18,22)SIZE(2)VALIDATE(DIGIT
):A
140 DISPLAY AT(20,1):"Reform
at to what length?" :: ACCEP
T AT(20,26)SIZE(2)VALIDATE(D
IGIT):X :: IF X=A THEN 130
150 OPEN #1:"DSK"&IF$,INPUT
:: OPEN #2:"DSK"&OF$,OUTPUT
:: IF X<A THEN 230
160 IF EOF(1)THEN 270 :: LIN
PUT #1:M$ :: L=LEN(M$):: IF
POS(M$,CHR$(13),1)=0 THEN 18
0
170 IF P+L<X+1 THEN PRINT #2
:M$ :: P=0 :: GOTO 160 ELSE
PRINT #2:SEG$(M$,1,X-P):SEG$
(M$,X-P+1,255):: P=0 :: GOTO
 160
180 IF L<A THEN M$=M$&RPT$("
 ",A-L):: L=A
190 IF P=0 THEN PRINT #2:M$;
:: P=L :: GOTO 160
200 IF P+L<X THEN PRINT #2:M
$;:: P=P+L :: GOTO 160
210 IF P+L=X THEN PRINT #2:M
$ :: P=0 :: GOTO 160
220 PRINT #2:SEG$(M$,1,X-P):
SEG$(M$,X-P+1,255);:: P=LEN(
SEG$(M$,X-P+1,255)):: GOTO 1
60
230 IF EOF(1)THEN 270 :: LIN
PUT #1:M$
240 L=LEN(M$):: IF L+P>X THE
N PRINT #2:SEG$(M$,1,X-P)::
M$=SEG$(M$,X-P+1,255):: P=0
:: GOTO 240
250 IF M$=CHR$(13)THEN 230
260 IF POS(M$,CHR$(13),1)<>0
 THEN PRINT #2:M$ :: P=0 ::
GOTO 230 ELSE PRINT #2:M$;::
 P=LEN(M$):: GOTO 230
270 CLOSE #1 :: CLOSE #2
```

MEMORY FULL

Jim Peterson ■

REPARATIONER AV MYARC-KORT

Reparationer av Myarc-kort görs av Cecure Electronics efter det att Myarc slutat. Firmans nya adress efter flyttning är enligt Micropendium nov 1992:

Cecure Electronics
P.O. Box 132
MUSKEGO, WI 53150
USA

Telefon (414) 679-4343 ■

PLOT AV KURVA FÖR SUPER-XB

(se utritad kurva PB 92-4 sid 20)

```
100 ! REM PLOT FOR SUPER-XB
120 ! by Jan Alexandersson
150 ! CALL FILES(2)
160 ! NEW
170 ! CALL INIT
180 ! CALL DRAWNPLOT
190 CALL LINK("GCLEAR")
200 CALL LINK("MOVE",10,10)
210 CALL LINK("DRAW",10,192)
220 CALL LINK("MOVE",10,10)
230 CALL LINK("DRAW",256,10)
250 CALL READ(X,Y)
260 IF X=0 THEN 320
270 CALL LINK("MOVE",X,Y)
280 CALL READ(X,Y)
290 IF X=0 THEN 320
300 CALL LINK("DRAW",X,Y)
310 GOTO 250
320 REM PLOT CURVE
330 FOR N=1000 TO 4000 STEP 1000
340 CALL RESTORE(N)
350 CALL READ(X,Y)
360 CALL LINK("MOVE",X,Y)
370 CALL READ(X,Y)
380 IF X=0 THEN 410
390 CALL LINK("DRAW",X,Y)
400 GOTO 370
410 NEXT N
420 CALL LINK("SHOW")
500 REM PLOT SCALE
510 DATA 87,00,10
520 DATA 87,00,-10
530 DATA 88,00,10
540 DATA 88,00,-10
550 DATA 89,00,10
560 DATA 89,00,-10
570 DATA 90,00,10
580 DATA 90,00,-10
590 DATA 91,00,10
600 DATA 91,00,-10
610 DATA 92,00,10
620 DATA 92,00,-10
630 DATA 85,11.5,100
640 DATA 86,00.5,100
650 DATA 85,11.5,200
660 DATA 86,00.5,200
670 DATA 85,11.5,300
680 DATA 86,00.5,300
690 DATA 85,11.5,400
700 DATA 86,00.5,400
710 DATA 85,11.5,500
720 DATA 86,00.5,500
730 DATA 85,11.5,600
740 DATA 86,00.5,600
750 DATA 85,11.5,700
760 DATA 86,00.5,700
770 DATA 85,11.5,800
780 DATA 86,00.5,800
790 DATA 85,11.5,900
800 DATA 86,00.5,900
810 DATA 85,11.5,1000
820 DATA 86,00.5,1000
830 DATA 0,0,0,0
1000 REM Horizon 192 kbytes
1010 DATA 86,02,230
1020 DATA 86,06,210
1030 DATA 86,09,210
1040 DATA 86,10,210
1050 DATA 86,11,210
1060 DATA 86,12,210
1070 DATA 87,01,210
1080 DATA 87,02,210
1090 DATA 87,03,210
1100 DATA 87,04,210
1110 DATA 87,05,210
1120 DATA 87,06,210
1130 DATA 87,07,210
1140 DATA 87,08,210
1150 DATA 87,09,210
1160 DATA 87,10,210
1170 DATA 87,11,195
1180 DATA 87,12,195
1190 DATA 88,01,195
1200 DATA 88,02,225
1210 DATA 88,03,225
1220 DATA 88,04,240
1230 DATA 88,05,260
1240 DATA 88,06,260
1250 DATA 88,07,260
1260 DATA 88,08,260
1270 DATA 88,09,260
1280 DATA 88,10,285
1290 DATA 88,11,285
1300 DATA 88,12,285
1310 DATA 89,01,265
1320 DATA 89,02,265
1330 DATA 89,04,300
1340 DATA 89,10,240
1350 DATA 89,11,220
1360 DATA 89,12,220
1370 DATA 90,01,220
1380 DATA 90,02,220
1390 DATA 90,03,220
1400 DATA 0,0,0,0
2000 REM Horizon 384 kbytes
...
3000 REM Horizon 512 kbytes
...
4000 REM Horizon 1 Mbytes
...
10000 SUB READ(X,Y)
10010 XSCALE=36
10020 YSCALE=1/5.6
10030 READ X1,X2,Y
10040 X=10+(X1+X2/12-86)*XSCALE
10050 IF X1=0 THEN X=0
10060 Y=10+Y*YSCALE
10070 SUBEND
```

■