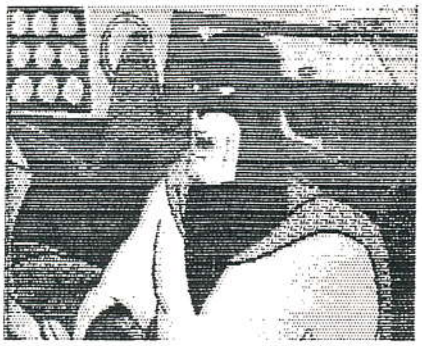
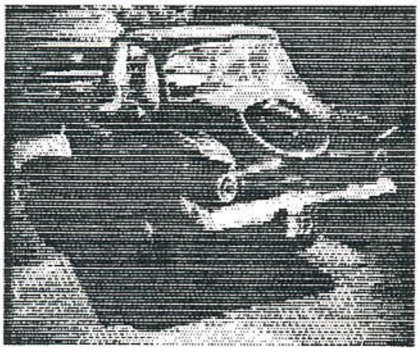
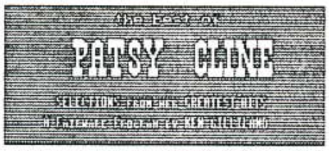
PROGRAM BITEN 94-3






the best of
PATSY CLINE
SELECTIONS from her GREATEST HITS
A Fairware Program by KEN GILLILAND

ISSN 0281-1146

REDAKTÖREN

Flex Cable Interface mellan min TI-99/4A och expansionsbox slutade fungera i juni. Jag lyckades få tag i en kabel från Jim Lesher, 722 Huntley, Dallas, TX 75214, USA. Den kostade 50 dollar. I mitten av augusti så kunde jag börja använda datorn igen och sammanställa detta nummer av Programbiten 94-3.

De två övre bilderna på framsidan har digitaliserats av Niklas Johansson i Mjölby. De är lagrade i My-art-format med 393 respektive 451 sektorer.

Bud Mills har snart ett nytt SCSI-kontrollkort klart för 170 dollar plus porto. Det fungerar enligt SCSI-2 som är ganska ny så det går inte att använda begagnade hårddiskar enligt SCSI-1. SCSI-2 hårddiskar finns nog inte i mindre storlek än 150 Mbytes. Själva DSR(PROM) är inte riktigt klart. SCSI-kortet kan användas med TI-99/4A eller Geneve. Geneve måste dock ha en speciell version av MDOS. SCSI-kortet kan samexistera med alla andra kontrollkort i samma box (inkl. HFDC).

TI-Club Göttingen inbjuder till 9:e internationella TI-mötet den 14-16 oktober 1994 i Göttingen (Rosdorf). Lokal: Kirchl. Gemeindeshaus Rosdorf, Kirchgasse 2, D-37124 ROSDORF, Tyskland. Kontakta i förväg för registrering och information:
Jörg Kirstan, Mengershäuser Weg 5, D-37124 ROSDORF, Tyskland (telefon 0551/781153).

Funnelweb spindel hittades i Visby på Gotland i mars 1994. Enligt tidningsartikeln var det en "trattminör" från Australien som rankas som den tredje mest giftiga spindeln i världen. Funnel = Tratt så jag gissar att det är en Funnelweb-spindel utan att vara helt säker. Tony McGoverns Funnelweb Editor 5.01 har dock med säkerhet använts för rak högermarginal i PB 94-3.

RäTTELSE:
Programmet MERGE TWO COLUMNS XB i PB 94-2 sid 26, programrad 260 ska vara:

```
260 IF ASC(A$)>127 THEN 330
```

■

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

Föreningens adress:
Föreningen Programbiten
c/o Schibler
Wahlbergsgatan 9 NB
S-121 38 JOHANNESHOV, Sverige

Postgiro 19 83 00-6
Medlemsavgiften för 1994 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:-
90-93 (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/hårdvara

PUTTING IT ALL TOGETHER No.10

by Jim Peterson, Tigercub, USA

The hardest part of learning to program is not in learning what the various commands do - it is in learning how to put them all together to do what you want them to do!

Key in this simple routine, and run it, to see what it does. Then read the explanation of each line and see how they do what they do.

```
100 CALL CLEAR
110 INPUT "KNOWN VALUE? ":C
120 X=1
130 GOSUB 180
140 IF A<C THEN Y=X :: X=X*2
 :: GOSUB 180 :: GOTO 140 EL
SE 160
150 IF A>C THEN Y=X :: X=X/2
 :: GOSUB 180 :: GOTO 150
160 Z=(ABS(X-Y))/2 :: Y=X ::
 IF A<C THEN X=X+Z ELSE X=X-
Z
170 GOSUB 180 :: GOTO 160
180 A=X^X/X
190 PRINT X;:: IF A=C OR A=B
 THEN STOP ELSE B=A :: RETUR
N
```

Equations such as the one in line 180 are very difficult to solve mathematically, but the computer can solve them quickly by systematic trial and error. You can substitute any other equation of one unknown value, using A for the known value and X for the unknown.

Line 110 gets the known value in C. Line 120 gives X a starting value of 1. Actually we could start with any value except 0, but we must give X some value or the computer will take it to be zero.

Execution jumps to line 180 to try this value in the equation. Line 190 prints the value of X, just so we can see what is going on, and then checks to see if the value obtained for A is the value we specified for C; in this case the problem is solved and execution stops. We will explain the B later. If not solved, execution returns to 130 and then to 140.

If A<C, meaning A is less than C, the value we received is too small, so X must be too small. In this case we save the value of X in Y, then double the value of X, GOSUB to the equation and check in 190 whether we have solved it. If not, we return to 140, jump back to the beginning of 140, see if A is still smaller than C and, if so, repeat the process again. Eventually the doubling of X will cause A to be more than C, if we do not hit it exactly in 190, and the ELSE 160 jumps us to that line.

However, if the first value we obtained for A was more than C, execution would drop right through line 140 to line 150. Here we would do exactly the same thing except that we would halve the value of X each time until A became less than C.

In either case, we end up in line 160 for the final step. Now here is why we were saving the previous value of X in Y each time. We subtract Y, the previous value, from X, the current value. Either one may be larger than the other, so we may get a negative value. ABS changes the negative to a positive. We divide the result by 2, and give that value to Z. If A is less than C we add Z to X, otherwise we subtract it. Then we GOSUB to 180 repeatedly until 190 finds that we have found the answer. Each time around the difference between Y and X will be half of what it was the previous time, and the result of the equation will alternate between A>C and A<C until it finally centers on the exact value.

Now, the reason for the B in line 190. The true value of X will usually run out to more decimal places than even our 16-bit TI-99/4A can handle and, since the exact full value cannot be reached, the program would go on forever. So, we save the value of A in B each time, and check each time to see if the next value received for A is that same value. If so, we have reached the limit of computer accuracy, so we stop. The same number will probably appear on screen several times at the end, because the screen shows only 10 digits but the computer continues calculating to 13 or 14 digits. ■

Programbiten behöver program och artiklar! Sänd in vad du har till redaktören.

BASIC OCH XB TIPS - 3

av Jan Alexandersson

Jag skall försöka att ta upp några frågor som kan vara av intresse för dig som har TI-99/4A med kassettbandspelare och använder BASIC.

FÖRDELAR MED TI-99/4A BASIC

Till att börja med vill jag påstå att BASIC till TI-99/4A inte är så dålig som många tror. Du kan få automatisk radnumrering med NUM och omnumrering av rader med RES. ALPHA LOCK låser endast bokstavstangenterna men inte siffertangenterna. Datorn räknar faktiskt med 14 decimaler trots att inte alla syns på skärmen med PRINT. Du kan använda tiopotenser upp till 127 trots att endast upp till 99 syns på skärmen med PRINT. Du har även möjlighet att använda långa variabelnamn med upp till 15 tecken där även ÅÄÖ@_ kan användas.

PRINT MED 14 SIFFROR

Jag har gjort ett litet kort BASIC-program som visar att datorn verkligen använder 14 siffror. Alla tal lagras i ett format som kallas RADIX 100 med 8 bytes per tal. Sju bytes används för att lagra 14 siffror som hundrapotenser (0-99 per byte) och en byte används för att sätta ut decimalpunkten (64 hundrapotenser = 128 tiopotenser) samt markera +/- före talet och +/- före potensen.

```
100 TAL=SIN(1)
110 PRINT STR$(TAL)
120 GOSUB 150
130 PRINT TAL$
140 END
150 REM ALLA DECIMALER
160 REM  TAL=.1 TILL <1E9
170 REM JAN ALEXANDERSSON
180 REM 1988-04-02
190 DEF LOG10(X)=LOG(X)/LOG(
10)
200 POTENS=INT(LOG10(TAL))
210 TAL1=INT(TAL*10^(7-POTEN
S))
220 TAL2=INT((TAL-TAL1/(10^(
7-POTENS)))*10^(14-POTENS))
230 TAL$=STR$(TAL1)&STR$(TAL
2)
240 TAL$=SEG$(TAL$,1,POTENS+
1)&"."&SEG$(TAL$,POTENS+2,14
-POTENS-1)
250 RETURN
```

Du kan prova att byta TAL=SIN(1) mot TAL=SIN(1)*10/10 och observera skillnaden. Du ser att datorn använder 13 eller 14 siffror beroende på om det är udda eller jämn tiopotens. Det finns inget behov att skriva ut så många siffror i ett verkligt program men det känns bra att datorn räknar med så stor noggranhet så att man slipper avrundningsfel.

I praktiken har du nog behov att kapa bort decimaler. Du kan göra så här med avrundning av sista decimalen:

```
100 TAL=4*ATN(1)
110 TAL=INT(TAL*1000+.5)/1000
120 PRINT TAL
```

Om du har Extended Basic kan du prova antalet decimaler och avrundning med PRINT USING.

TIOPOTENSER UPP TILL 127

Jag har även gjort ett kort BASIC-program för att visa att TI-99/4A klarar mycket stora potenser vilket inte är så vanligt bland hemdatorer. XB-ägare kan även prova med PRINT USING.

```
100 TAL=1.1234E123
110 PRINT STR$(TAL)
120 GOSUB 150
130 PRINT TAL$
140 END
150 REM ALLA POTENSER
160 REM JAN ALEXANDERSSON
170 REM 1988-04-02
180 DEF LOG10(X)=LOG(X)/LOG(
10)
190 POTENS=INT(LOG10(ABS(TAL
)))
200 DECIMALER=TAL/10^POTENS
210 TAL$=STR$(DECIMALER)&"E"
&STR$(POTENS)
220 RETURN
```

EDIT (ändra programrad)

När du har ett program som du vill ändra så måste du editera berörd programrad. I BASIC har du kanske lärt dig att skriva EDIT 160 om du vill ändra rad 160. Eftersom detta inte finns i Extended Basic bör du redan från början vänja dig vid att skriva 160 FCTN X (pil ner) eller 160 FCTN E (pil upp). När du ändrat raden färdigt så fortsätt till nästa rad med "pil ner" eller "pil upp". Du kommer ur editeringsläget genom att trycka på ENTER. Varning skriv inte radnummer ENTER om du inte vill sudda denna rad.

Det är även möjligt att editera med NUM så du kan skriva NUM 160. För säkerhets skull bör du göra RES först eftersom NUM endast visar var 10:e rad. Mellanliggande rader visas ej. Du kan inte heller göra ERASE av en hel rad om NUM används.

Om du någon gång får problem med att knappa in en lång programrad från en tidning så att datorn vägrar att ta emot flera tecken så gör så här. Avsluta raden var som helst men se till att du inte får SYNTAX ERROR. Om det är en textsträng så avsluta med " eller sätt ut parenteser som behövs. När du fått en riktig rad tryck sedan på ENTER och tag sedan tillbaka raden för editering. Du upptäcker nu att du kan skriva en mycket längre programrad.

KASSETTBANDSPELARE

Den bandspelare du använder bör ha diskantkontroll och diskanten skall vridas på maximalt. Volymen bör vridas på relativt mycket men inte maximalt vid avspelning. Om du får felmeddelande "NO DATA FOUND" bör du öka volymen något och om du får felmeddelande "ERROR IN DATA" så minska volymen något.

Din kassettspelare bör även ha fjärrkontroll (den svarta sladden med den tunnare pluggen). Det finns ingen standard på denna kontroll så vissa kassettspelare måste ha en inverteringsplugg mellan kassettkabeln och kassettspelaren. Fjärrkontrollen är nödvändig vid filhantering med kassett. Att spara och ladda program går dock bra utan fjärrkontroll även om det kanske är något bekvämare.

De äldre TI-99/4A har möjlighet till både CS1 och CS2 medan nyare endast klarar CS1. Denna CS2 kan dock bara användas för att lagra med SAVE eller PRINT #. Dessutom har TI tagit bort CS2 från nya kassettkablar. Det är nog inte säkert att alla TI-99/4A-ägare har den optimala sladden eftersom dessa två ändringar gjorts oberoende av varandra.

Som du redan vet används SAVE och OLD för att spara respektive ladda program. Om du skulle ångra dig efter att du tryckt på ENTER så tryck på E för EXIT så avbryter datorn det du hade börjat med. Observera att detta fungerar endast när bandet står still.

På liknande sätt kan du även jämföra två bandinspelningar med varandra och ta reda på om programmen är lika. Ladda först in med OLD CS1 från första bandet och skriv sedan SAVE CS1 (ENTER). Tryck sedan på C för CHECK och ladda in från det andra bandet. Datorn kommer nu att jämföra de två versionerna av programmet men kan inte visa vad som skiljer utan endast att något skiljer.

MER ATT LÄSA

NN 83-2: Att använda grafik

PB 84-3: BASIC programmering

PB 84-4: Tips och tricks

PB 85-3: Kommandon i Basic och XB

PB 85-4: CALL KEY

COMPUTE! Regena: Guide to the TI-99/4A (358 sidor)

COMPUTE!: TI Games (211 sidor)

COMPUTE!: Creating Arcade Games (200 sidor)

Davis: Programs for TI Home Computer (126 sidor)

Alla dessa uppräknade böcker är mycket bra skrivna och innehåller i huvudsak program i BASIC.

BASIC-SKOLA PÅ KASSETT

Det finns tre kassettband utgivna av TI som vardera innehåller 7-10 långa

BASIC-program som har undervisande text blandat med exempel som man provar direkt utan besväret att själv knappa in det:

PHT 6007 Teach Yourself Basic
PHT 6019 Teach Yourself Extended Basic
PHT 6067 Begining Basic Tutor

■

KATALOG AV SKIVA

av Jan Alexandersson

Detta program listar skivkatalogen till skärmen eller till skrivare för alla typer av diskkontrollkort. Det visar även de speciella filtyper som finns i Myarc HFDC t.ex. sub-dir. Godtycklig path kan listas. Även programfilernas längd i bytes visas med Myarc-kontrollkort. Katalogen visas först på skärmen och om du sedan trycker på P så skrivs den ut på skrivare via PIO-porten. Om annan tangent trycks kan du sedan välja ny katalog.

```
100 REM CATALOG V2
110 REM JAN ALEXANDERSSON
120 REM 1988-12-25
130 DIM A$(127),A(127),J(127
),K(127)
140 N=1
150 P=2
160 DATA D/F,D/V,I/F,I/V,PGM
,DIR,EMU,DIS/FIX,DIS/VAR,INT
/FIX,INT/VAR,PROGRAM,DIRECT,
EMULATE
170 FOR H=0 TO N
180 FOR I=N TO 7
190 READ T$(H,I)
200 T$(H,I)=SEG$(T$(H,I)&" "
,N,3+H*4)
210 NEXT I
220 NEXT H
230 CALL CLEAR
240 CALL CHAR(123,"000028003
8447C44")
250 PRINT
260 INPUT "DIR ":D$
270 IF SEG$(D$,LEN(D$),N)=".
" THEN 290
280 D$=D$&"."
290 OPEN #P:D$,INPUT ,RELATI
VE,INTERNAL
300 FOR R=0 TO 127
310 INPUT #P:A$(R),A(R),J(R)
,K(R)
320 IF LEN(A$(R))=0 THEN 340
330 NEXT R
340 R=R-N
350 CLOSE #P
360 GOSUB 390
370 GOSUB 510
380 GOTO 250
390 IF A(0)THEN 410
400 D$=A$(0)
410 PRINT #F:D$:"FIL=";STR$(
R);" LED=";STR$(K(0));" ANV=
";STR$(J(0)-K(0)):"filnamn
  sekt typ";TAB(22+4*F);"län
gd p": :
420 FOR I=N TO R
430 PRINT #F:A$(I);TAB(15-LE
N(STR$(J(I))));J(I);
440 PRINT #F:TAB(17);T$(F,AB
S(A(I)));SEG$("      "&STR$(
K(I)),LEN(STR$(K(I))),7);" "
;CHR$(32-10*(A(I)<0))
450 CALL KEY(3,T,S)
455 IF I=63 THEN 640
460 IF S=0 THEN 490
480 GOSUB 510
490 NEXT I
500 RETURN
510 CALL KEY(3,T,S)
520 IF S<N THEN 510
530 IF T=80 THEN 560
540 IF T=83 THEN 250
550 RETURN
560 F=N
570 OPEN #F:"PIO"
580 PRINT #F:CHR$(27);"1";CH
R$(10);CHR$(27);"R";CHR$(5);
CHR$(10);CHR$(10);CHR$(10);C
HR$(14);
590 GOSUB 390
600 PRINT #F
610 CLOSE #F
620 F=0
630 RETURN
640 PRINT #F:CHR$(12)
650 GOTO 490
```

■

LAGRA MINNE PÅ KASSETT

av Anders Persson

För beskrivning av hur programmet fungerar se PB 85-2.05. Den kompletta programlistningen har dock inte publicerats tidigare.

```
***********************************
*                                 *
* SAVE MEMORY ON CASSETTE ROUTINE *
*                                 *
*                                 *
*                                 *
***********************************

       DEF  CSAVE
       REF  VMBW,VSBW,VSBR,VMBR
       REF  VDPWD
       REF  KSCAN,VWTR,SOUND
       REF  GRMRA,GRMWA

STATUS EQU  >837C
KEYBRD EQU  >8375
KEYNUM EQU  >8374
GPLWS  EQU  >83E0

IOWRIT EQU  >1346
IOREAD EQU  >142E
IOVRFY EQU  >1426

XMLADR EQU  >2002
VDPBUF EQU  >1000
USERWS EQU  >20BA

ENTER  BYTE 13
CURSOR BYTE 30
MOTOR1 EQU  22     MOTOR CONTROL CS1
LBIT   DATA >0036  MAG TAPE LOAD BIT
SBIT   DATA >0032  MAG TAPE SAVE BIT
SET    DATA >2000
START  DATA >0000 START ADDR.TO SAVE
STOP   DATA >0000 STOP ADDR. TO SAVE
HEXSTA DATA >0000  VALIDATE HEX AT
*                  INPUT FLAG
SAVARG DATA >0000  ARGUMENT CLUSTER
*                FOR I/O: BYTE COUNT
       DATA VDPBUF BUFFER ADDRESS

TEXT1  TEXT '** MEMORY ON CASSETTE '
       TEXT 'UTILITY **'
TEXT2  TEXT 'START ADDRESS:'
TEXT3  TEXT 'STOP  ADDRESS:'
TEXT4  TEXT '* REWIND CASSETTE TAPE'
TEXT5  TEXT '* PRESS CASSETTE RECORD'
TEXT6  TEXT '* RECORDING'
TEXT7  TEXT '* PRESS CASSETTE STOP'
TEXT8  TEXT 'THEN PRESS ENTER'
TEXT9  TEXT 'PROGRAM NAME:'
```

```
* LJUD DATA
SDATA  BYTE >03,>8B,>06,>90
AVDATA BYTE >03,>8B,>06,>9F
BUFFER BSS  40
NAME   BSS  10
************************************
* SETS TEXT MODE, CLEARS SCREEN ETC.
*

INIT   LI   R0,>0717
       BLWP @VWTR
       LI   R0,>01D0
       BLWP @VWTR
       LI   R0,>D000
       MOVB R0,@>83D4
CLS    CLR  R0             CLEAR SCREEN
       LI   R1,>2000
       LI   R2,959
       BLWP @VSBW
CLS1   MOVB R1,@VDPWD
       DEC  R2
       JNE  CLS1
       RT
*-----------------------------------
* SCROLLS ONE LINE
*

SCROLL LI   R0,40
       LI   R1,BUFFER
       LI   R2,40
SCR1   BLWP @VMBR
       AI   R0,-40
       BLWP @VMBW
       AI   R0,80
       CI   R0,1000
       JNE  SCR1
       LI   R0,919
       LI   R1,>2000
       LI   R2,40
       LI   R3,BUFFER
       BLWP @VSBW
       JMP  SCR3
SCR2   MOVB R1,@VDPWD
SCR3   MOVB R1,*R3+
       DEC  R2
       JNE  SCR2
       RT
*-----------------------------------
* SCANS KEYBOARD
*

INKEY  BLWP @KSCAN
       MOVB @STATUS,R1
       COC  @SET,R1
       RT
*-----------------------------------
* READS A VALUE
```

```
*
* R0: VDP ADDRESS
* R5: CHARACTER COUNT
*

INPUT  MOV  R11,R9
       CLR  @KEYNUM
       CLR  R4
READ   CLR  R3
       CLR  R1
       BLWP @VSBR
       MOV  R1,R2
       MOVB @CURSOR,R1
       BLWP @VSBW
*
GET    BL   @INKEY
       JEQ  WRITE
       INC  R3
       CI   R3,400
       JNE  GET
       CLR  R3
       BLWP @VSBR
       CB   R1,@CURSOR
       JEQ  FLASH1
       MOVB @CURSOR,R1
       JMP  FLASH2
FLASH1 MOVB R2,R1
FLASH2 BLWP @VSBW
       JMP  GET
*
WRITE  MOVB @KEYBRD,R1
       CI   R1,>0D00
       JEQ  ENT
       CI   R1,>0800
       JEQ  LEFT
       CI   R1,>0900
       JEQ  RIGHT
       CI   R1,>0F00
       JEQ  BACK
       CI   R1,>0500
       JEQ  QUIT
       MOV  @HEXSTA,@HEXSTA
*           VALIDATE HEXADECIMAL?
       JEQ  WRITE1
       CI   R1,>3000
       JL   GET
       CI   R1,>4100
       JL   TEST1
       CI   R1,>4600
       JH   GET
       JMP  WRITE1
TEST1  CI   R1,>3900
       JH   GET
WRITE1 BLWP @VSBW
       C    R4,R5
       JEQ  READ
       INC  R0
       INC  R4
       JMP  READ
*
LEFT   MOV  R2,R1
       BLWP @VSBW
```

```
       CI   R4,0
       JEQ  READ
       DEC  R0
       DEC  R4
       JMP  READ
*
RIGHT  MOV  R2,R1
       JMP  WRITE1
*
ENT    MOV  R2,R1
       BLWP @VSBW
       B    *R9
*
BACK   CLR  @STATUS  RETURN CALLER
       LI   R0,>01E0 GRAPHICS MODE
       BLWP @VWTR
       SLA  R0,8
       MOVB R0,@>83D4
       B    *R10
*      BACK TO CALLING PROGRAM
*
QUIT   LIMI 2
       BLWP @>0000
*---------------------------------
* WRITES 'THEN PRESS ENTER'
* AND WAITS FOR ENTER

PRESS  MOV  R11,R9
       BL   @SCROLL
       LI   R0,922
       LI   R1,TEXT8
       LI   R2,16
       BLWP @VMBW
PRESS1 BL   @INKEY
       JNE  PRESS1
       MOVB @KEYBRD,R1
       CB   R1,@ENTER
       JNE  PRESS1
       BL   @SCROLL
       BL   @SCROLL
       B    *R9
*---------------------------------
* ISSUES A BEEP
*

BEEP   LI   R1,SDATA
       LI   R2,4
BEEP1  MOVB *R1+,@SOUND
       DEC  R2
       JNE  BEEP1
       LI   R1,>4000
BEELO  DEC  R1
       JNE  BEELO
       LI   R1,AVDATA
       LI   R2,4
BEEP2  MOVB *R1+,@SOUND
       DEC  R2
       JNE  BEEP2
       RT
*---------------------------------
* CONVERTS ENTERED VALUE
* TO HEXADECIMAL
```

```
*

HEXIN  CLR  R2
HEXI4  CLR  R1
       BLWP @VSBR
       CI   R1,>2000
       JEQ  HEXI1
       SLA  R2,4
       SRL  R1,8
       AI   R1,-48
       CI   R1,9
       JH   HEXI2
       JMP  HEXI3
HEXI2  AI   R1,-7
HEXI3  A    R1,R2
       INC  R0
       JMP  HEXI4
HEXI1  RT
*-------------------------------------
*
* CASSETTE SAVE PROCEDURE
* CALLS THE CASSETTE WRITE PART OF
* THE I/O STATEMENT SERVICE ROUTINE
* IN THE * GPL INTERPRETER.
*
* THIS ROUTINE SHOULD WAIT A LITTLE
* BEFORE IT STARTS WRITING TO THE
* CASSETTE,
* AND SHOULD PERHAPS NOT SHUT DOWN
* THE CASSETTE SO SOON AFTER
* FINISHED OUTPUT.
*

SAVE
       SBO  MOTOR1       START MOTOR
       MOV  @STOP,@SAVARG
       S    @START,@SAVARG
*           CALCULATE BYTE COUNT
       LI   R0,VDPBUF
       MOV  @START,R1
       MOV  @SAVARG,R2
       BLWP @VMBW
*      MOVE DATA TO VDP BUFFER

       LI   R0,SAVARG
       MOV  R0,@GPLWS+2  POINTER TO
*           BUFFER DESCRIPTION BLOCK

       MOVB @GRMRA,R0    RESET GROM
*      ADDRESS TWO STEPS. REEXECUTE
*      THE XML THAT
       SWPB R0           CALLED THIS
*PROGRAM, BUT LET IT START AT SAVE1.
       MOVB @GRMRA,R0
       SWPB R0
       AI   R0,-3
       MOVB R0,@GRMWA
       SWPB R0
       MOVB R0,@GRMWA

       MOV  @XMLADR,R3   MODIFY XML
*      TABLE ENTRY TO FORCE CALL TO
```

```
*      MY SAVE1 LBL
       LI   R0,SAVE1
       MOV  R0,@XMLADR
       LWPI GPLWS
       B    @IOWRIT
*      WRITE TO CASSETTE

SAVE1  LWPI USERWS
       MOV  R3,@XMLADR
       SBZ  MOTOR1
       B    *R11

*-------------------------------------
CSAVE  MOV  R11,R10
       BL   @INIT
       CLR  R12  CLEAR CRU BASE ADDR
RUN1   LI   R0,3
       LI   R1,TEXT1      * UTILITY
       LI   R2,32
       BLWP @VMBW
       CLR  @HEXSTA
       LI   R0,120
       LI   R1,TEXT9  * PROGRAM NAME
       LI   R2,13
       BLWP @VMBW
       LI   R5,9       NUMBER OF CHAR
       LI   R0,133     LOCATION
       BL   @INPUT
       LI   R0,133
       LI   R1,NAME
       LI   R2,10
       BLWP @VMBR         READ NAME
       SETO @HEXSTA
       LI   R0,200
       LI   R1,TEXT2   START ADDRESS
       LI   R2,14
       BLWP @VMBW
       LI   R0,240
       LI   R1,TEXT3   STOP ADDRESS
       LI   R2,14
       BLWP @VMBW
RUN2   LI   R0,214
       LI   R5,3
       BL   @INPUT  READ STARTADDR.
       LI   R0,214
       BL   @HEXIN
       MOV  R2,@START
       LI   R0,254
       LI   R5,3
       BL   @INPUT   READ STOP ADDR.
       LI   R0,254
       BL   @HEXIN    CONVERT TO HEX
       MOV  R2,@STOP
       C    @START,@STOP
       JH   RUN2          START>STOP?
       SBO  MOTOR1
       LI   R0,920
       LI   R1,TEXT4      REWIND
       LI   R2,22
       BLWP @VMBW
       BL   @BEEP
       BL   @PRESS    WAIT FOR ENTER
```

```
SBZ  MOTOR1    CASS.MOTOR OFF
LI   R0,920
LI   R1,TEXT5     RECORD
LI   R2,23
BLWP @VMBW
BL   @BEEP
BL   @PRESS    WAIT FOR ENTER
BL   @BEEP
LI   R0,920
LI   R1,TEXT6     RECORDING
LI   R2,11
BLWP @VMBW
BL   @SCROLL
BL   @SCROLL
BL   @SAVE
LI   R0,920
LI   R1,TEXT7      STOP
LI   R2,21
BLWP @VMBW
BL   @BEEP
BL   @PRESS    WAIT FOR ENTER
BL   @CLS
B    @RUN1
END                               ■
```

"PAGE-PRO" REVIEW

by Charles Good, Lima Ohio User Group, USA

A picture is worth a thousand words. The pages that contain this article have been entitely composed using PAGE PRO and its associated utilities so that you an see for yourself what the printed output of this fine software looks like. The different fonts you see on these pages come with the basic PAGE PRO package. PAGE PRO is exactly what Asgard Software claims it to be, a page making utility that allows you to mix text and pictures on the same page with total on screen "what you see is what you get" capabiltiy.

The text of this article was typed in the normal way with Funnelweb's editor using any margins I desired and leaving groups of 8 blank lines in the body of the text for later insertion of graphics by PAGE PRO. This DV80 file was then processed with a modified version of PAGE PRO's COLUMNIZER utility to prepare the text for use by PAGE PRO. Contrary to the printed instructions that come with PAGE PRO, it is not necessary to save the text file with PF. Also, contrary to the PAGE PRO documentation, processing with COLUMNIZER did not take nearly as long as the stated 15-45 minutes per page. Other than inserting " " for each desired indented space and making sure that each paragraph ended in a carriage return symbol (a little cr), it was not necessary to do anything special to the original DV80 file prior to processing with COLUMNIZER.

COLUMNIZER took my original file and split it into PAGE PRO page sized blocks of text, each a separate DV80 file. Each page of test was in 2 right justified columns This was all done automatically by COLUMNIZER. I then booted PAGE PRO itself and imported each DV80 page of text into a PAGE PRO one page at a time. The header box, header text, and page border were then manually added to each page to make the page resemble the normal format of this newsletter. Graphics were then inserted in the body of the text. Everything including border and graphics was visible on screen in a "what you see is what you get" format as each page was composed. Each page was then printed and saved to disk.

COLUMNIZER, as it comes with the PAGE PRO package, normally leaves 2 blank lines at the top and bottom of the page and spreads text out to the extreme right and left edges of the 60 column page. I needed 4 blank lines (not 2) at the top to leave room for the header box shown here, and I needed each line of text indented 1 space on the right and left to leave room for the page border. Since COLUMNIZER is in XBASIC, I was able to modify it to meet these special needs (see below). There is much to be said for having software written in XBASIC so that it can be easily modified by the user. Assembly is fast, but is much harder for the average user to modify. The actual PAGE PRO program is in assembly.

With PAGE PRO you get a 60 column by

66 line page to work with. As you can see on the pages that contain this article, this quite adequately fills a printed page. The screen window is 1/2 of a page width and approximately 1/5 of a page length. You window left/right and up/down with the TI Writer standard F4 F5 and F6 keys. You can also scroll up/down one line at a time with the arrow keys. At any one time only 1/10 of the page is in view on the screen. The current position of the cursor, with respect to the entire page, is indicated at the bottom of the screen with a very clear numerical row and column indicator. There is a Geneve specific version of PAGE PRO, but its screen display looks just like the 99/4A version. The Geneve version does not yield an 80 column display.

You can type directly onto the page, and this is the usual method of text entry. As you type you have full screen cursor control with the arrow keys, but there is no automatic word wrap or right justification. The only way to edit existing text is with single line editing such as is done by inserting within, deleting from, or typing over an existing line of text. Other text lines are not directly affected by such editing. If you "import text" from any DV80 file such as that created by COLUMNIZER, everything else on the page is erased, a distinct disadvantage.

Two text font sizes are available at one time, one twice the size of the other. Each of the several large and small fonts included included with the PAGE PRO package contains a complete set of upper and lower case letters and other keyboard characters. You can also use software in the PAGE PRO package to convert TI Artist or CSGD fonts to PAGE PRO disk format. Although you are not limited to a particular font, you ARE LIMITED to the font sizes and spacing shown on these pages. It is not normally possible with PAGE PRO to use th tight CONDENSED printer pitch (17 characters per inch) that is normally used for text in this newsletter. It is possible to give the appearance of having more than two text fonts on a PAGE PRO page by including a graphic made out of text. Some examples follow:

Unlike importing text, graphics (pictures) can be imported anywhere onto a PAGE PRO page without altering the rest of the page. Graphics are seen on screen exactly as they will appear on the printed page. You position the cursor where the upper left part of the picture will be and enter the disk file name of the graphic. If you don't like what you see you can press CTRL/K to "kill" (erase) the picture. As many as 28 graphics on a page are allowed. Several graphics are part of the PAGE PRO package, including this 99/4A console.

You can convert any TI Artist instance into a PAGE PRO graphic disk file with software provided with the PAGE PRO package. This software automatically converts all instances on a disk into files usable by PAGE PRO. It takes about 2 hours to convert a full SSSD disk of instances, but this only has to be done once. In the future software is promised (probably at extra cost) that will convert other graphic formats (such as TI Artist pictures with file names that end in "_P") into PAGE PRO graphic format. In theory one graphic can be as large as an entire PAGE PRO page. It is possible to type text over a graphic, but this leaves holes in the graphic. If you import graphics with blank areas within the outline of the graphic, the picture will "flow around" existing text without disturbing text that already exists in the graphic's "holes".

A wide variety of lines and shapes can easily be drawn on a PAGE PRO page just like typing text. CTRL/8 puts you in Lines mode. You then put lines and shapes on the page by typing specific upper and lower case letters. A very clear display on the lower half of the screen shows you which keys produce which lines. The shapes are, in effect, a third font. The borders and header box of the pages that contain this article were made in this way. This line creating ability makes PAGE PRO great for drawing borders and making business forms and graphs. Several sample page files that are included with the basic PAGE PRO package, including a blank INVOICE form, are examples of what is possible.

If you are trying to create a vertical design, such as a straight up/down line, it is very inconvenient to have the cursor move one space to the right every time you type a line drawing key. With PAGE PRO you can specify the direction of cursor movement (left, right, up, down) after each character is typed. CTRL/EXSD sets the cursor movement directon until you change this direction. You still have full screen cursor control with the arrow keys.

--DISADVANTAGES OF PAGE PRO-

It is very easy for me with PAGE PRO to create a blank newsletter page template with borders, header box, and header text and save this template to disk. I normally use xeroxes of a professionally printed page template for the cut and paste masters of these newsletter pages. IF I could load my newsletter page template into PAGE PRO and then import some DV80 text created with COLUMNIZER onto this template this would be fantastic! I can't. When text is imported, everything else on the page is erased. The borders and headers on the pages that contain this article had to be individually drawn on each page after importing the text. Since a PAGE PRO graphic can, in theory, be as large as an entire page, it may someday be possible to store page borders as a single large graphic and overlay this graphic onto a page of text. As of now the utility software needed to do this does not exist.

When a PAGE PRO page is printed, the printer uses dot addressable graphic mode to print everything, including text. You can choose single, double, or quadruple density for the printout. As stated earlier you are limited to only the two font sizes shown on these pages. You cannot with PAGE PRO directly access the printer's built in fonts (NLQ, emphasized, condensed, etc). Also you can't insert special printer features such as single word double strike, underline, or italic into the body of the text. Such features are easily available with PLUS! when printing from the TI Writer formatter. In my opinion the PAGE PRO text looks grainy. It just doesn't look as good as text created from the printer's normal fonts. Compare the various pages of this newsletter to see the differences.

PAGE PRO apparently uses every last bit of VDP memory with a 99/4A. If you have an AVPC card as part of you system it is necessary to run the SET99/4A program that comes with the AVPC prior to booting PAGE PRO.

--CONCLUSIONS--

PAGE PRO is quite easy to use. It is great for the creation of single page signs, forms, ads, greeting cards, and stationary. The ability to mix text, pictures, and lines on the same page and the "what you see is what you get" screen display are unequaled in the world of the 99/4A and Geneve. I intend to purchase my review copy. However I do not plan to use PAGE PRO to produce this newsletter. For the newsletter layout I am usually going to stick the with the good old cut and paste technique that gives me the total flexiblity I require. I guess other newsletter editors agree with me in this respect. I have seen favorable reviews of PAGE PRO in other newsletters, but I have not seen an entire issue of any newsletter produced with PAGE PRO.

The COLUMNIZER XB program that comes with PAGE PRO is a very useful utility in and of itself. It is one of the easiest to use text column making programs I have seen, much easier to use than its documentation would suggest.

The basic PAGE PRO package ($24.95 + shipping) comes with several on disk sample pages, graphic files, and alternate fonts, as well as COLUMNIZER. Included also is a well written hard copy manual and detailed tutorial. Extra cost picture and font disks are available.

ASGARD SOFTWARE
Harry Thomas Brashear
2753 Main St.
NEWFANE, NY 14108
USA ■

CARTRIDGE AT >6000

by Mack McCormick, USA

This is a text file discussion of the ROM cartridge port for the TI-99/4A. It represents information I have been able to obtain from various references. Cartridge programs must operate from >6000 to >7FFF. When the computer is RESET or turned on, the power up routine looks for a Header or Control block at location >6000 in the cartridge port. This contol block establishes the linkage into your cartridge program and allows you to have multiple entry points. Here is an example contol block used to provide one entry point;

```
0000 AA01       DATA     >AA01      6000      ID FOR BOOT
0002 0000       DATA     >0000      6002
0004 0000       DATA     >0000      6004
0006 000C       DATA     CHAIN      6006      ADDRESS OF MENU LIST
0008 0000       DATA     >0000      6008
000A 0000       DATA     >0000      600A
000C 0000 CHAIN DATA     >0000      6010      CHAIN POINTER
000E 0020       DATA     SLOAD      6012      ENTRY POINT
0010   0F       BYTE     SLOAD-$-1 6014       LENGTH OF MENU TEXT
0011   54       TEXT     'CARTRIDGE NAME'
0020 0460 SLOAD B        START
0022 092E
```

Let's examine the control block. If the TI operating system finds >AA at >6000 it knows a cartride is plugged in the port. The next byte must be a >01 at location >6001. This informs the operating system that the code in the cartridge is executable machine language. Other codes are used for GROM, but that's another discussion. The data at location >6002 - >6005 is zero. Location >6006 must contain a word pointer to a list which identifies the menu text and associated entry point when that item is selected. This location usually contains a >600C. Locations >6008 - >600B must be zero. The chain list at >600C contains the following:

Bytes 1 & 2 = chain pointer to the next menu list - or 0000 is this is the last list in the chain.

Bytes 3 & 4 = entry point associated with this menu selection.

Byte 5 = length of the menu text.

Bytes 6 - N = Menu Text - this is displayed on main menu. Craig Miller's newsletter has additional information on the power up routine for the computer. Remember all dynamic data must be in RAM usually in the >8300 area. This area is used for registers plus VDP RAM is used for variable storage. Cartridges cannot REFerence any lable or routine outside the cartridge. This means the cartridge program must provide it's own VSBW, VSBR, VMBW, and VMBR routines which are normally loaded from the Editor Assembler cartridge. Examples of what these routines look like may be found in the Tombstone City game or Craig Millers newsletter. Armed with this information, it possible to disassemble code to see how the program works. Hope you find this information useful. See ya around the boards. ■

```
10 !************************
20 !*      ÅLVSTOPPNING     *
40 !* AV F.NILSSON 1988     *
50 !* (repris fr. PB 88-3) *
60 !* XB + EM              *
70 !************************
80 CALL INIT :: CALL CLEAR :
: CALL SCREEN(2)
90 CALL CHAR(40,"FFA5A5A5A5A
5A5FF")
100 FOR I=1 TO 28 :: CALL SP
RITE(#I,40,5,I*8-7,256):: NE
XT I
110 FOR I=1 TO 28 :: CALL MO
TION(#I,0,RND*90+10):: NEXT
I
120 CALL LOAD(-31878,28)
130 CALL KEY(0,K,S):: IF S<1
 THEN 130
140 CALL LOAD(-31878,0)
150 CALL KEY(0,K,S):: IF S<1
 THEN 150 ELSE 120
```

■

PROGRAMS THAT WRITE PROGRAMS, PART 7

by Jim Peterson, Tigercub, USA

In the first five parts of this series, written some years ago, I showed how to use a Basic program to write a program in the form of a tokenized D/V 163 file, which could then be merged in and run as a program.

In part 6 (PB 91-5 p.9) I later showed how to incorporate tokenized code into a Basic program, creating programs to write programs to write programs. Now, here is the final step - programs that rewrite themselves!

This all began due to a weakness of TI Basic - the RUN statement will not recognize a string variable name. You cannot write A$="DSK1. PROGRAM" :: RUN A$ - you can only use the format RUN "DSK1.PROGRAM" . That made it difficult to write a general-purpose menu-loader. In an early issue of the 99'er Magazine Dr. Stefan-Romano, writing under the psuedonym A. Kludge, solved this problem by publishing what was probably the first PEEK and POKE for the TI. He wrote a dummy line - RUN "DSK1.1234567890" as the last line of the program, two PEEKs to find that last line in memory, and a CALL LOAD routine to overwrite that dummy string with the desired DSK and filename. CALL PEEK(-31952,X,Y):: CALL PEEK(X*256+Y-65534,X,Y):: Z=X*256+Y-65534 finds the 5th byte of the last line.

Since the line already contains the two tokens for the line number, the token 169 for RUN and the token 199 for a quoted string, it is only necessary to poke in the new length-of-string token by CALL LOAD(Z,LEN (A$)) and then poke in the string by FOR J=1 TO LEN LEN(D$):: CALL LOAD (Z+J,ASC(SEG$(D$,J,1))) :: NEXT J and finally add the end-of-line token CALL LOAD(Z+J,0).

TI Basic also will not allow line numbers to be represented by variables. You cannot write X=1000 :: GOSUB X. Barry Traver used the above technique to get around this limitation, and I adopted it on my Nuts & Bolts Disk No. 2 to write subprograms for a variable GOSUB, variable GOTO and variable RESTORE. In these cases, the last line of the program is written as a dummy subprogram such as GOSUB 30000 :: SUBEND. Since the line already contains the two tokens for the line number, the token for GOSUB or whatever, the token 201 for a referenced line number, two tokens for the line number, the token for SUBEND and the token 0, it is only necessary to tokenize the line number and poke it in by CALL LOAD (Z,INT(N/256),N-256*(INT(N/256))).

The current challenge was to write a program in which a formula or algorithm could be input from the keyboard and then used to solve problems. The only practical way seemed to be to overwrite the last line of the program with the formula, as a subroutine. A subprogram is not practical because the values of all variables would have to be passed as parameters.

To accomplish this, it is necessary to convert the entire formula, as input, into tokenized format. This is how it was done -

```
100 DISPLAY AT(3,3)ERASE ALL
:"PROGRAMMABLE CALCULATOR":"
":"      by Jim Peterson" ::
CALL INIT
110 DISPLAY AT(7,1):"  Input
 any mathematical    formula
 in the form of a    valid B
ASIC statement, usingA for t
he value to be calcu-"
120 DISPLAY AT(11,1):"lated
and B thru F for the  values
 to be input.":"  Examples -
 ":"   A=(B-C)^D-7":"   A=B-
C+C*.1-C*.0575":"  A=INT(AB
S(B-C))-PI"
130 DISPLAY AT(19,1):"  To c
hange the formula,    enter
0 for all values."
```

```
140 DISPLAY AT(24,7):"PRESS
ANY KEY" :: DISPLAY AT(24,7)
:"press any key" :: CALL KEY
(0,K,S):: IF S=0 THEN 140 EL
SE CALL HCHAR(7,1,32,18*32)
150 A$="" :: DISPLAY AT(8,1)
:"FORMULA?" :: ACCEPT AT(10,
1):F$ :: ON WARNING NEXT
160 DATA ),182,(,183,=,190,+
,193,-,194,*,195,/,196,^,197
,ABS,203,ATN,204,COS,205,EXP
,206,INT,207,LOG,208
170 DATA SGN,209,SIN,210,SQR
,211,TAN,212,PI,221
180 RESTORE 160 :: FOR J=1 T
O 19 :: READ X$,W
190 P=POS(F$,X$,1):: IF P<>0
 THEN F$=SEG$(F$,1,P-1)&CHR$
(W)&SEG$(F$,P+LEN(X$),255)::
 GOTO 190
200 NEXT J :: J=0
210 IF J=LEN(F$)THEN 240 ::
J=J+1 :: Z$=SEG$(F$,J,1):: I
F POS(".0123456789",Z$,1)=0
THEN A$=A$&Z$ :: GOTO 210
220 N$=N$&Z$ :: Z$="" :: IF
J=LEN(F$)THEN 230 :: J=J+1 :
: Z$=SEG$(F$,J,1):: IF POS("
.0123456789",Z$,1)<>0 THEN 2
20
230 A$=A$&CHR$(200)&CHR$(LEN
(N$))&N$&Z$ :: N$="" :: GOTO
 210
240 A$=A$&CHR$(130)&CHR$(136
)&CHR$(0):: GOSUB 330 :: CAL
L HCHAR(12,1,32,250)
250 W=0 :: IF POS(A$,"B",1)<
>0 THEN DISPLAY AT(12,1):"B=
?" :: ACCEPT AT(12,5):B :: W
=W+B
260 IF POS(A$,"C",1)<>0 THEN
 DISPLAY AT(13,1):"C=?" :: A
CCEPT AT(13,5):C :: W=W+C
270 IF POS(A$,"D",1)<>0 THEN
 DISPLAY AT(14,1):"D=?" :: A
CCEPT AT(14,5):D :: W=W+D
280 IF POS(A$,"E",1)<>0 THEN
 DISPLAY AT(15,1):"E=?" :: A
CCEPT AT(15,5):E :: W=W+E
290 IF POS(A$,"F",1)<>0 THEN
 DISPLAY AT(16,1):"F=?" :: A
CCEPT AT(16,5):F :: W=W+F
300 ON ERROR 310 :: GOTO 320
310 CALL SOUND(400,110,0,-4,
0):: DISPLAY AT(12,1):RPT$("
 ",250):: DISPLAY AT(24,5):"
INVALID FORMULA" :: RETURN 1
50
320 IF W=0 THEN 150 :: GOSUB
 350 :: DISPLAY AT(18,1):"A=
";A :: GOTO 250
330 CALL PEEK(-31952,A,B)::
CALL PEEK(A*256+B-65534,A,B)
:: C=A*256+B-65534
340 FOR J=1 TO LEN(A$):: CAL
L LOAD(C+J-3,ASC(SEG$(A$,J,1
))):: NEXT J :: RETURN
350 !***********************
****************************
```

Line 150 accepts the input of a formula. Lines 160 and 170 are DATA containing every valid statement or symbol that could be used in a math equation, each one followed by its token. Line 180 restores the DATA lines, because they can be used repeatedly, and reads the data two at a time, the statement and its token.

Line 190 searches the formula for the first occurrence of that statement and, if it is found, replaces it with its token. Note that LEN is used to determine how many bytes of the string are being replaced, because statements and symbols vary from 1 to 3 characters in length. This line goes back to itself to check for repeated occurrences of the same statement.

Line 210 goes through the partially tokenized string character by character, looking for any numeric characters. As long as none are found, the characters are built into string A$. If a numeric character is found, line 220 takes over and begins building these into the string Z$ until a non-numeric character is reached. Line 230 then adds this to A$ preceded by the token 200 for an unquoted string and the length token, and goes back to 210 to continue searching.

Line 240 adds the double colon token, the RETURN token, and the end of line token zero to the string A$, and jumps to the routine to overwrite it into the program.

Lines 250 through 290 check whether variable names B through F are present in the formula and, if so, ask for their values.

In line 320, if zero has been input for all values, W=0 so execution goes back for input of a new formula; otherwise the GOSUB goes to line 350 which now contains the formula. The answer is displayed and execution goes back for input of new values. ■

REFORMATTING

by Jim Peterson, Tigercub, USA

With the establishment of the Clearinghouse BBS, newsletter editors will have available more articles on disk, rather than having to xerox them or retype them from other newsletters. This will make it easier for them to reformat articles to their own requirements - but they will have to know how to do so.

I am by no means an expert on this subject, but I will offer a few ideas. It seems to me that the most practical column widths are 28, 40, 60 and 80.

I have always believed that Basic and XBasic program listings should be published in 28-column width, exactly as they appear on the screen. This makes it much easier to key them in accurately, especially when the listing contains strings of blank spaces or long strings of hex codes. For that reason, years ago I wrote a program to reformat listed programs accurately into 28-column width. Nowadays, the Super Extended Basic module will do that much more easily. Since my Tips From The Tigercub newsletters consisted mostly of program listings, I always published them in four columns of 28-character width. However, that is too narrow for primarily text articles, requiring too much hyphenation or creating too many gaps.

The 40-column width is perhaps best of all, because it can be printed in two columns of elite font or three columns of condensed font. The 60-column width can be printed in two columns of condensed font.

The 80-column width is suitable for regular D/V80 files in pica font, but this is somewhat wasteful of space; pica is considerably larger than most printed material. It is possible to write routines to reformat and print D/V80 text in even longer lines, up to 160 characters long in Epson condensed elite, but lines of more than 80 characters are difficult to scan.

Page Pro printing may require other widths, but I know nothing about that; I really do not consider the oversized crowded characters of Page Pro to be suitable for newsletters.

The first step in reformatting should be to separate any program listing portions from the rest of the text, and reformat them separately if at all. For instance, if you load the article THISNTHAT into Funlweb Editor and find that lines 200 to 300 of a 400-line article are a program listing, do this - FCTN 9, SF, 200 300 DSK1.PROGRAM and then FCTN 9, LF, 1 199 DSK1.THISNTHAT and then FCTN 9, LF, 199 301 E DSK1. THISN-THAT and FCTN 9, SF, DSK1. TEXT.

The next step is to make sure that the title and the paragraphs, any line that nothing should be added onto, ends in a carriage return. The carriage return (CR, ASCII 13) in the Funlweb Editor looks like a little square C above and to the left of a little upside-down L.

If the CR's are missing and you have a text file with indented paragraphs and centered headers (i.e., blank spaces before the title, etc.), this tinygram program will add the CR's.

```
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 DISPLAY AT(18,1):"Put bl
ank lines between     paragr
aphs? Y/N" :: ACCEPT AT(19,1
7)SIZE(1)VALIDATE("YNyn"):Q$
150 OPEN #1:"DSK"&IF$,INPUT
:: OPEN #2:"DSK"&OF$,OUTPUT
:: C$=CHR$(13)
160 IF EOF(1)THEN 190 :: LIN
PUT #1:M$ :: IF Q$="Y" OR Q$
="y" THEN 180
170 IF M$="" THEN PRINT #2:C
$:M$;:: GOTO 160 ELSE IF ASC
(M$)<33 THEN PRINT #2:C$:M$;
:: GOTO 160 ELSE PRINT #2:""
:M$;:: GOTO 160
180 IF M$="" OR M$=" " THEN
PRINT #2:C$ :: GOTO 160 ELSE
 IF ASC(M$)<33 THEN PRINT #2
:C$:C$:M$;:: GOTO 160 ELSE P
RINT #2:"":M$;:: GOTO 160
190 PRINT #2:CHR$(13):: CLOS
E #1 :: CLOSE #2
```

Another way to add CR's is to type CTRL U to get the underline cursor, then go through the text with FCTN 4 and FCTN 6 and the arrow keys, typing SHIFT M wherever you need a CR.

Now, here is a tip. FCTN 9, S, 1 to get to the beginning of the file, FCTN 9, RS, / CTRL-U M CTRL-U / CTRL-U M CTRL-U FCTN-W / (slash, carriage return, slash,˜,carriage return,slash) to replace each carriage return with a tilde followed by a carriage return. Why? We'll get to that! Hit Enter, then A for All.

In order to use the Funlweb Formatter to reformat a file, it is necessary to print it back to disk - and when you do that, the Formatter can play some very nasty tricks! Any & symbol in the text will simply disappear! An @ in the text will disappear but cause the word following it to be repeated again and again on following lines. An * followed by two or more digits will disappear, along with the first two digits. A caret sign (shift 6) will disappear. And a period at the beginning of a line will cause the entire line to disappear! (and since this article now contains a & and a @, you've got problems already!)

So, to be on the safe side, get back to the top of your text, go into RS again with /&/\/ (the \ is FCTN Z). If you jump to the end of the file, there were no ampersands, so you won't have to restore them after reformatting. You might do the same thing with the @, just in case. The * bug is very unlikely to occur in a text file, and the ˆ is also unlikely (except in this article!), but you might scan down through the first character of each line to be sure that a decimal number in the text has not placed a period there. If so, the best solution is to insert a 0 before the period.

There is a better way to prevent the Formatter from garbling your text. I have never understood why Texas Instruments used characters that might appear in text as control characters, when other useless characters were available - and I've often wondered why the McGoverns didn't do something about it. But you can, if you have John Birdwell's DSKU. Copy your Funlweb to a new disk, just in case. Boot up DSKU and select 1. File utilities, then 5. Find string. The filename is FO. Select H for hex. At the prompt for a string, type 2A23214026 and for replacement string type 7C2321605C. Select R for replace, then FCTN W, hit Enter twice to accept defaults, and it's done. From now on, if you want to underline a word, use FCTN Z instead of &, if you want to double-strike a word use FCTN C instead of @, and if you want to input a value from a data file use FCTN A instead of * . You will still have to watch out for those periods, and the caret sign - I don't know the fix for those.

Now, to get Funlweb to reformat the text, open a new line 1 with FCTN 8 and type in .LM 0;RM 79;IN 0;FI . The 79 would give you 80-column text; you can substitute any number, 1 less than the actual column width you want (because the computer is counting from a left margin of 0, not 1). If you wanted a pre-set left margin you could change that 0, and adjust the RM figure accordingly. If the text does not have paragraph indentations, and you want to add them, change the 0 after IN to whatever number of spaces you want to indent; you can also increase the amount of indentation that way. And, if you want Funlweb to insert additional blank spaces in the text in order to justify the right margin (line it up evenly), add ;AJ after the FI . Now save the file to disk, then go to the Funlweb Formatter.

Accept that same default filename; instead of the printer option, enter DSK1. and a different filename, accept all the defaults, and the file will be printed back to the disk under that new filename.

Return to the Editor and load that new filename. You will find that your text has been reformatted to the desired width, but every line now ends in a line feed (a little L to the upper left of a little F); your carriage returns have also been converted to line feeds. And there are now three lines at the top containing nothing but a blank followed by a line feed, groups of similar lines throughout the text, probably a long series of such lines at the end, and sometimes a few lines in the text containing a few dashes followed by a line feed. Some of them also contain a form feed! You will have to go through the text with FCTN 4, FCTN 6 and the arrow keys, deleting those lines with FCTN 3.

Now, to get rid of those line feeds - FCTN 9, PF, C DSK1.(and a new filename). It is best to always use a new filename for each step in the process so that if you make a mistake - as you will occasionally! - you do not have to go all the way back to start over.

Load that new file, and you will find that printing to disk with the C option has apparently stripped out all the line feeds. Actually, it changed them to blank ASCII 32's, which could cause problem in multiple-column printing or concantenation. If you want to get rid of them, SF the file to disk, LF it back and PF it to get rid of the tab line.

You might find it easier to just use this handy-dandy routine to delete the line feed lines and line feeds -

```
100 DISPLAY AT(12,1)ERASE AL
L:"Input filename?":"DSK" ::
 ACCEPT AT(13,4):IF$ :: OPEN
 #1:"DSK"&IF$
110 DISPLAY AT(15,1):"Output
 filename?":"DSK" :: ACCEPT
AT(16,4):OF$ :: OPEN #2:"DSK
"&OF$
120 A$=CHR$(32)&CHR$(10):: B
$=CHR$(12)&CHR$(10)
130 IF EOF(1)THEN 150 :: LIN
PUT #1:M$
140 IF M$=A$ OR M$=B$ THEN 1
30 ELSE IF SEG$(M$,LEN(M$),1
)=CHR$(10)THEN PRINT #2:SEG$
(M$,1,LEN(M$)-1):: GOTO 130
ELSE PRINT #2:M$ :: GOTO 130
150 CLOSE #1 :: CLOSE #2
```

But, now the carriage returns you worked so hard to put in have also disappeared! Not to worry. FCTN 9, S, 1, FCTN 9, RS, / FCTN-W / CTRL-U M CTRL-U / , Enter, A for All, and those tildes will magically turn into carriage returns! Occasionally one will appear at the beginning of a blank line instead of at the end of the preceding line, in which case you will have to delete it and put it where it belongs with CTRL-U SHIFT-M. Why did I use FCTN-W, the tilde? Just because I have never seen it used for anything else in a text file (oops! except in this article!) - I could have used FCTN A, Z, C, etc. In fact, if you want to preserve any CTRL-U type printer codes in the text, you could RS them back and forth in the same way. After CTRL-U to get the underline cursor, FCTN-R is the ASCII 27 escape code, SHIFT-2 is 0 and SHIFT-A is 1, etc.

Rather than go through all that, maybe you would rather just throw the Funlweb Formatter away and use my little handy dandy Text Reformatter in good old primitive Extended Basic. It's slow, but it avoids all those extra steps and all those pitfalls. The text must have carriage returns.

```
100 CALL CLEAR :: CALL SCREE
N(5):: FOR SET=0 TO 12 :: CA
LL COLOR(SET,2,16):: NEXT SE
T :: CR$=CHR$(13)
110 DISPLAY AT(2,7):"TEXT RE
FORMATTER":"":"       by Jim
 Peterson"
120 DISPLAY AT(6,1):"Input f
ilename?":"DSK" :: ACCEPT AT
(7,4)BEEP:IF$ :: OPEN #1:"DS
K"&IF$,INPUT
130 DISPLAY AT(8,1):"Output
filename?":"DSK" :: ACCEPT A
T(9,4)BEEP:OF$ :: OPEN #2:"D
SK"&OF$,OUTPUT
```

```
140 DISPLAY AT(11,1):"Presen
t line length?" :: ACCEPT AT
(11,22)SIZE(2)VALIDATE(DIGIT
):LL
150 DISPLAY AT(13,1):"Reform
at to what length?" :: ACCEP
T AT(13,26)SIZE(2)VALIDATE(D
IGIT):R
160 IF R=LL THEN 140 ELSE CA
LL CLEAR
170 IF EOF(1)THEN 290 :: LIN
PUT #1:M$ :: M$=P$&M$ :: P$=
"" :: IF R>LL THEN 230
180 L=LEN(M$)+(POS(M$,CR$,1)
<>0):: IF L<=R AND POS(M$,CR
$,1)<>0 THEN PRINT #2:M$ ::
GOTO 170 ELSE IF L<R THEN P$
=M$&" " :: GOTO 170
190 C$=SEG$(M$,1,R):: CALL L
ASTPOS(C$," ",P)
200 IF P<>0 THEN 210 ELSE PR
INT #2:C$ :: M$=SEG$(M$,R+1,
255):: GOTO 180
210 GOSUB 300 :: GOTO 180
220 GOSUB 310 :: GOTO 180
230 IF POS(M$,CR$,1)<>0 AND
LEN(M$)<=R+1 THEN PRINT #2:M
$ :: GOTO 170
240 IF LEN(M$)<R THEN P$=M$&
" " :: GOTO 170
250 C$=SEG$(M$,1,R):: CALL L
ASTPOS(C$," ",P):: IF P=0 TH
EN PRINT #2:C$ :: M$=SEG$(M$
,R+1,255):: GOTO 230
260 IF P=R THEN PRINT #2:SEG
$(M$,1,P-1):: M$=SEG$(M$,R+1
,255):: GOTO 230
270 GOSUB 300 :: GOTO 230
280 GOSUB 310 :: GOTO 230
290 PRINT #2:P$ :: CLOSE #1
:: CLOSE #2 :: STOP
300 IF SEG$(M$,R+1,1)=" " TH
EN PRINT #2:SEG$(M$,1,R):: M
$=SEG$(M$,R+2,155):: RETURN
310 PRINT #2:SEG$(M$,1,P-1):
: M$=SEG$(M$,P+1,255):: RETU
RN
320 SUB LASTPOS(A$,B$,Y):: X
,Y=0
330 X=POS(A$,B$,X+1):: IF X>
0 THEN Y=X :: GOTO 330
340 SUBEND
```

I have also written a Formatter+ program which will even reformat text which does not have carriage returns, if header lines and paragraphs are idented. It will also optionally allow you to hyphenate any word which breaks after the second character, and optionally right-justifies the text. It is too long to list here, but will be available on the Clearinghouse BBS and in my TI-PD catalog.

The next-to-last thing to do when reformatting (but, from my reading of many newsletters, often omitted!) is FCTN 9, S, 1, FCTN 9, RS, /- // (replace a hyphen followed by a blank with a null, to fix hyphenated words that have ended up in the middle of a line). DON'T, repeat DON'T do this until you have restored the carriage returns!! Enter, but this time to do not use A for All; use Y(es) and N(o) to go through the text, deleting unwanted hyphens but leaving those at ends of lines or elsewhere if they belong.

And, last of all, PLEASE, PF rather than SF, to print the file back to disk rather than saving it back, to get rid of that pestiferous tab line!

Now, as to reformatting program listings - preferably, don't! A 28-column listing combined with 40-column text isn't going to waste much space.

Remember that program listings are printed so that people can key them in and run them, and the least mistake in reformatting them will usually result in garbage.

Assembly source code and C99 source code (and probably most any other language) MUST NOT be reformatted! However, anything beyond the 25th character of assembly source code is just a comment, usually preceded by an asterisk, and so is anything after an asterisk in the first column. These do not affect the program. If a comment exceeds your desired line length, it is safe to open a new line below it and retype the comment there, preceded by an asterisk.

Basic and Extended Basic programs can be reformatted, but the method described above is not reliable. If you must reformat them, I think that the following program will do a foolproof job. Again, first you must put carriage returns at the end of each program line, and the only practical way is to get the CTRL-U underline cursor and go through inserting them with SHIFT-M. For those of you who are not programmers, I had better emphasize that I am talking about numbered program lines, not the numbered lines of text that appear in the Editor.

```
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)&CHR$
(13):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
$&CHR$(13):: P=0 :: GOTO 160
220 PRINT #2:SEG$(M$,1,X-P)&
CHR$(13):SEG$(M$,X-P+1,255);
:: P=LEN(SEG$(M$,X-P+1,255))
:: GOTO 160
230 IF EOF(1)THEN 270 :: LIN
PUT #1:M$
240 L=LEN(M$):: IF L+P>X THE
N PRINT #2:SEG$(M$,1,X-P)&CH
R$(13):: M$=SEG$(M$,X-P+1,25
5):: P=0 :: GOTO 240
250 IF M$=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
```

Another way of reformatting a program listing is to use Curtis Alan Provance's remarkable Textloader to convert the listing to program format and then listing it to disk in the desired format, using Super Extended Basic. However, Textloader can introduce some bugs, so be sure to test the program before you list it, and be prepared to fix the bugs.

Now that you have gone to all that work, are you going to print your article through the Funlweb Formatter and perhaps garble it again? Remember what I told you about the &, @, *, ^ and the leading period! Program listings usually contain those characters. If you have not modified your FO file, and you do not replace and transliterate, you will print garbage! ■

FUNNELWEB EDITOR 5.01

av Jan Alexandersson

Tony McGovern har släppt en ny version av sin ordbehandlare Funnelweb Editor version 5.01 (Mar/30/94). Ordbehandlaren finns både i 40-kolumnsversion för TI-99/4A originaldator och 80-kolumnsversion för videoprocessorn 9938/9958. Editorn måste laddas från menyn i Funnelweb 4.40. Fyra olika varianter av editorn finns på den nya flexskivan:

- 40 kolumn, baseline editor, filerna ED och EE

- 40 kolumn, updated/AEH, filerna ED/AEH och EE/AEH

- 80 kolumn, standard 5.00, filerna E8 och E9, för 9938 med 128 kbytes VDP-RAM

- 80 kolumn, EVRAM 5.01 (tidigare 5.00b), filerna ED, EE och EF, för 9938/9958 med 192 kbytes VDP-RAM

Både 40- och 80-kolumnsversionerna kan nu justera jämn högermarginal direkt i editorn med CTRL-R.

80-kol har nu 64 kbytes textbuffert.

Mark/Copy/Paste via "clipboard" gör att man kan kopiera text från andra filer (eller från View Buffer) som sedan klistras in i texten.

Observera att HD (Hard Disk) i editorns meny kan användas för att ta upp katalog av flexskivor med subdirectory på samma sätt som själva hårddisken. Välj DSK5 eller det som passar som path. Flytta sedan markören till önskad sub-dir och tryck på mellanslag. Du kan även backa tillbaka till roten med CTRL-O.

REFERENSER

PB 93-1 Tony McGovern: FW 80 Editor V 5.00

PB 93-2 Charles Good: FW V5 Text Editor

PB 93-3 Jan Alexandersson: FW 40 & 80 kol editor 5.00

PB 94-2 Charles Good: FW EVRAM editors (5.00b och 5.00c)

Micropendium June 94 p.24 Charles Good: FW v.5.01 Editor

```
FW_ED_40
FIL=48 LED=43 ANV=677
filnamn     sekt typ       längd p

-4080COMBO    16 DIS/VAR     80
4PRINTFILE     5 DIS/VAR     80
CHAR*1/ARC    17 INT/FIX    128
CHAR@*/ARC    25 INT/FIX    128
CHAR@1         9 PROGRAM   2046
CHAR@2         9 PROGRAM   2048
CHAR@3         9 PROGRAM   2048
CHAR@4         9 PROGRAM   2046
CHAR@5         9 PROGRAM   2048
CHAR@6         9 PROGRAM   2046
CHAR@7         9 PROGRAM   2048
CHAR@8         9 PROGRAM   2048
CHARA1         7 PROGRAM   1536
CHARB1         7 PROGRAM   1536
CHARC1         7 PROGRAM   1536
CHARD1         7 PROGRAM   1536
CHARE1         7 PROGRAM   1536
CHARF1         7 PROGRAM   1536
CHARG1         7 PROGRAM   1536
CHARUTIL       5 PROGRAM    928
CHRCOAL/S     57 DIS/VAR     80
CON/ED         2 DIS/VAR     80
CONFIG/40     34 DIS/VAR     80
ED            33 PROGRAM   8164
ED/AEH        33 PROGRAM   8118
EE            21 PROGRAM   5114
EE/AEH        32 PROGRAM   7848
F4TXAE         5 PROGRAM    856
F4TXBE         5 PROGRAM    936
F4TXCE         5 PROGRAM    850
F4TXDE         5 PROGRAM    894
F4TXEE         5 PROGRAM    876
F4TXFE         5 PROGRAM    840
FWDOC/ED40    63 DIS/VAR     80
FWDOC/ED41    61 DIS/VAR     80
FWDOC/ED42    58 DIS/VAR     80
HELP4A         5 PROGRAM    960
HELP4B         5 PROGRAM    960
HELP4C         5 PROGRAM    960
HELP4D         5 PROGRAM    960
HELP4E         5 PROGRAM    960
HELP4F         5 PROGRAM    960
HELP4G         5 PROGRAM    960
HELP4H         5 PROGRAM    960
HELP4I         5 PROGRAM    960
HELP4J         5 PROGRAM    960
HELPMAKE40     4 PROGRAM    748
INSTALL/ED     8 PROGRAM   1602
```

```
FW_ED_80
FIL=29 LED=4 ANV=716
filnamn     sekt typ       längd p

8PRINTFILE    10 DIS/VAR     80
CHAR@1         9 PROGRAM   2046
CHAR@6         9 PROGRAM   2046
CHARUTIL       5 PROGRAM    928
CHRCOAL/S     57 DIS/VAR     80
CON/ED         2 DIS/VAR     80
CONFIG/ED     42 DIS/VAR     80
E8            33 PROGRAM   8168
E9            39 PROGRAM   9612
ED            33 PROGRAM   8116
EE            33 PROGRAM   8192
EF            12 PROGRAM   2762
F8TXAE         5 PROGRAM    882
F8TXBE         5 PROGRAM   1018
F8TXCE         5 PROGRAM    872
F8TXDE         5 PROGRAM    926
F8TXEE         5 PROGRAM    882
F8TXFE         5 PROGRAM    884
FWDOC/EV80   308 DIS/VAR     80
HELP00        10 PROGRAM   2080
HELP01        10 PROGRAM   2080
HELP02        10 PROGRAM   2080
HELP03        10 PROGRAM   2080
HELP10        10 PROGRAM   2080
HELP20        10 PROGRAM   2080
HELP30        10 PROGRAM   2080
```

```
HELP40        10 PROGRAM   2080
HELPMAKE80     4 PROGRAM    764
INSTALL/ED     8 PROGRAM   1602
```

Du kan få en kopia av 40- och 80-kolumnsversionen genom att sända flexskivor och frankerat svarskuvert till mig. Som du ser av katalogen så ryms vardera versionen på två SS/SD eller på en DS/SD. Alltihopa ryms på en DS/DD eller fyra SS/SD. ■

FROM BASIC TO ASSEMBLY - 10

by Bob August, Bug News, USA

Last month we showed you how to add color to you text mode assembly program. This month we will add color to your edit mode programs. Remember to add color to the text mode program we added two lines:

```
       LI   R0,>074F
       BLWP @VWTR
```

This time we add:

```
       LI   R0,>074F
       BLWP @VWTR
       LI   R0,>0380
       LI   R1,>4F00
COLSET BLWP,@VSBW
       INC  R0
       CI   R0,>0393
       JNE  COLSET
```

We load register zero with >074F to get the top and bottom screen edges to white. After we write it to the VDP register 7. We then load register zero with >0380 which is the start of the color set table. We load regester one with >4F00 which is blue letters with a white screen. The 4 gives you dark blue and the F gives you the white. We then write it to the screen one byte at a time until we reach >0393 which is color set 16.

In basic we would enter the following:

```
100 CALL SCREEN(16)
110 FOR C=1 TO 16
120 CALL COLOR(C,5,1)
130 NEXT C
```

This is much like the routine to clear the screen. Change the screen and character colors in you program until you have the color routine down in you mind. Play with you programs and you will learn alot more. Now you have two more routines to add to you book and from noe on you will have color in you life.

Until next month

HAPPY ASSEMBLING!

```
*****************************************
* BASIC TO ASSEMBLY   Lesson Number 10  *
*****************************************
*
       DEF  START                  Entry point of program
       REF  VSBW,VMBW,KSCAN,VWTR   Utilities used in program
*
WRKSP  BSS  32                     Workspace buffer
SAV11  BSS  2                      Save return address buffer
*
MESAGE TEXT ' This is in the edit mode with  '   Message in
       TEXT '                                '
       TEXT ' 32 columns on a white screen   '   32 columns
       TEXT '                                '
       TEXT ' with the text in dark blue.    '
QUIT   TEXT 'Press the Enter key to quit'        Prompt to quit
*
       EVEN                 Make sure we start on even byte
*
```

```
* Start of program
*
START  MOV  R11,@SAV11      Save return address
       LWPI WRKSP           Load our workspace
       BL   @CLEAR          GOSUB CLEAR to clear the screen
*
* Set screen and text color
*
       LI   R0,>074F        Load VDP Register 7 with >4F
       BLWP @VWTR           ( Sets screen edge color to white )
       LI   R0,>0380        Start of color sets
       LI   R1,>4F00        Blue letters, White screen
COLSET BLWP @VSBW
       INC  R0              Next color set
       CI   R0,>0393        Color set 16
       JLE  COLSET
*
* Display messages on screen
*
       BL   @DISPLY         GOSUB to DISPLY routine
       DATA 192,MESAGE,160  Screen location, Text, Length
*
       BL   @DISPLY         GOSUB to DISPLY routine
       DATA 578,QUIT,27     Screen location, Text, Length
*
* Call Key routine
*
       CLR  @>8374          Clear to zero for CALL KEY(0,K,S)
       CLR  @>837C          Clear status to zero
       LI   R4,>2000                                 (Ed.change)
KLOOP  BLWP @KSCAN          CALL KEY(0,K,S)
       CB   @>837C,R4       Check status for key press   (Ed.change)
       JNE  KLOOP           If S=0 THEN KLOOP            (Ed.change)
       MOV  @>8375,R0       Move key press to register zero
       CI   R0,>0D          Compare to 13 or Enter key
       JNE  KLOOP           If not enter key, GOTO KLOOP
       CLR  @>837C          Clear status to zero
       MOV  @SAV11,R11      Put return address in register 11
       BLWP @0              Quit ( FCTN = )
*
* Clear screen routine
*
CLEAR  LI   R0,0            Load R0 with zero ( Row 1, Column 1 )
       CLR  R1              Clear Register one
       LI   R2,1            Load R2 with one ( Length byte )
CLOOP  BLWP @VSBW           Write a space to the screen
       INC  R0              Add one to R0
       CI   R0,767          Compare R0 to 959 ( 24 X 40, 0 to 959 )
       JLE  CLOOP           Jump to CLOOP if less then 959
       RT
*
* Display at routine
*
DISPLY MOV  *R11+,R0        Put screen location in R0
       MOV  *R11+,R1        Put Text in R1
       MOV  *R11+,R2        Put length of text in R2
       BLWP @VMBW           Write it to the screen
       RT                   Return to calling area
*
* End program with auto start
*
       END  START
```

TIPS FROM THE TIGERCUB #55

by Jim Peterson, USA

The Tigercub has dipped a cautious paw into the cold dark mysterious waters of asembly, while still keeping a firm grip on trusty old Extended Basic. The result is an XBasic program that writes an assembly program!

The following subprogram, when merged into any program which has reidentified characters, and called after the characters have been reidentified, will write a source code which can be assembled into object code, loaded from XBasic and linked to instantly access the character set.

The source code is based on 2FONTS/S by Barry Traver, who gives credit to Mac McCormick, David Migicovsky and Karl Schuneman. (see further in Tips 56 and 58. - Ed.)

```
19000 SUB CHARSUB(HX$())
19001 DISPLAY AT(12,1)ERASE
ALL:"Source code filename?":
"DSK" :: ACCEPT AT(13,4)SIZE
(12)BEEP:F$ :: OPEN #1:"DSK"
&F$,OUTPUT
19002 DISPLAY AT(15,1):"LINK
ABLE program name?" :: ACCEP
T AT(16,1)SIZE(6):P$
19003 DISPLAY AT(18,1):"Rede
fine characters from    ASCI
I      to ASCII"
19004 ACCEPT AT(19,7)VALIDAT
E(DIGIT)SIZE(3):F
19005 ACCEPT AT(19,21)VALIDA
TE(DIGIT)SIZE(3):T
19006 PRINT #1:TAB(8);"DEF";
TAB(13);P$ :: PRINT #1:"VMBW
   EQU  >2024" :: PRINT #1:"
STATUS EQU  >837C"
19007 NB=(T-F+1)*8 :: CALL D
EC_HEX(NB,H$):: A=768+F*8 ::
 CALL DEC_HEX(A,A$)
19008 FOR CH=F TO T :: IF CH
<144 THEN CALL CHARPAT(CH,CH
$)ELSE CH$=HX$(CH)
19009 IF FLAG=0 THEN PRINT #
1:"FONT";:: FLAG=1
19010 FOR J=1 TO 13 STEP 4 :
: M$=M$&">"&SEG$(CH$,J,4)&",
" :: NEXT J :: M$=SEG$(M$,1,
23)&"  *"&CHR$(CH)
19011 PRINT #1:TAB(8);"DATA
"&M$ :: M$="" :: NEXT CH
19012 PRINT #1:P$;TAB(8);"LI
   R1,FONT" :: PRINT #1:TAB(
8);"LI   R0,>"&A$ :: PRINT #
1:TAB(8);"LI   R2,>"&H$
19013 PRINT #1:TAB(8);"BLWP
@VMBW":TAB(8);"CLR  @STATUS"
:TAB(8);"RT":TAB(8);"END" ::
 CLOSE #1
19014 SUBEND
19015 SUB DEC_HEX(D,H$)
19016 X$="0123456789ABCDEF"
:: A=D+65536*(D>32767)
19017 H$=SEG$(X$,(INT(A/4096
)AND 15)+1,1)&SEG$(X$,(INT(A
/256)AND 15)+1,1)&SEG$(X$,(I
NT(A/16)AND 15)+1,1)&SEG$(X$
,(A AND 15)+1,1):: SUBEND
```

Now to try it out. You probably know that CALL CHARSET will restore reidentified characters below ASCII 96 to normal form, but not those above, so let's write a routine to restore those. Clear the memory with NEW, merge in the above, which you should have SAVED with -

SAVE DSK1.CHARSUB,MERGE

by MERGE DSK1.CHARSUB. Add a line -

100 CALL CHARSUB(HX$()) and RUN. Answer the filename prompt with DSK1.OLDLOW/S, the next prompt with OLDLOW and select ASCII 97 to 127.

When done, insert the Editor/Assembler module and its disk Part A. Select Assembler, Y to load assembler, give the source code DSK1.OLDLOW/S, object code DSK1.OLDLOW/O, just press Enter at next prompt, and R for options. You should get 0000 ERRORS.

Now key in this routine to test your program.

```
100 CALL INIT :: CALL LOAD("
DSK1.OLDLOW/O"):: FOR CH=33
TO 126 :: CALL CHAR(CH,"FF81
8181818181FF"):: PRINT CHR$(
CH);:: NEXT CH
101 CALL KEY(0,K,S):: IF S=0
 THEN 101 ELSE CALL CHARSET
102 CALL KEY(0,K,S):: IF S=0
 THEN 102 ELSE CALL LINK("OL
DLOW")
110 GOTO 110
```

Press any key to restore the upper case characters by CALL CHARSET, any key again to use the CALL LINK.

You are now ready to use the routine to copy all kinds of character sets from the programs in your library. You don't have any such programs? Not to worry. You don't have to reidentify characters one by one with one of those graphics editor programs. You can just manipulate the existing hex codes of the normal characters. I have created nearly 50 different character sets by that method!

The space occupied by a character on the screen is really an 8x8 square of 64 tiny dots. Various dots are turned on (colored) and off (transparent) to create a pattern - just the opposite of light bulbs on a scoreboard.

And those on-and-off dots are really the binary numbers which the computer uses. But fortunately the computer lets us use hexadecimal numbers rather than binary. The following will print out a reference chart of decimal to binary to hexadecimal. You can easily convert it to dump to a printer.

```
10 DISPLAY AT(6,1)ERASE ALL:
"DEC BIN HEX"
100 FOR J=0 TO 15 :: CALL DE
C_BIN(J,B$):: CALL DEC_HEX(J
,H$):: DISPLAY AT(J+8,1):J;T
AB(5);B$;TAB(10);SEG$(H$,4,1
):: NEXT J
21020 SUB DEC_BIN(D@,B$):: D
=D@ :: IF D=0 THEN B$="0000"
 :: SUBEXIT
21021 IF D=1 THEN 21022 :: X
=D/2 :: B@$=STR$(ABS(X<>INT(
X)))&B@$ :: D=INT(X):: IF D>
1 THEN 21021
21022 B@$="1"&B@$ :: B$=RPT$
```

```
("0",4-LEN(B@$))&B@$ :: B@$=
"" :: SUBEND
21039 SUB DEC_HEX(D,H$)
21040 X$="0123456789ABCDEF"
:: A=D+65536*(D>32767)
21041 H$=SEG$(X$,(INT(A/4096
)AND 15)+1,1)&SEG$(X$,(INT(A
/256)AND 15)+1,1)&SEG$(X$,(I
NT(A/16)AND 15)+1,1)&SEG$(X$
,(A AND 15)+1,1):: SUBEND
```

And this routine will show you how each letter is formed, by binary 0's (off) and 1's (on), for each key you press. I put it in merge format so you can MERGE it into any program and CALL it to examine the characters.

```
17000 SUB CHARVIEW
17001 !programmed by Jim Pet
erson Feb 1989
17002 DISPLAY AT(1,1)ERASE A
LL:"CHARACTERS IN BINARY & H
EX":;:"Press any key to see
the    binary representation
 of thescreen character and
its hexcode."
17003 DISPLAY AT(8,1):"Press
 Enter to see the char-acter
."
17004 CALL KEY(0,K,S):: IF K
=13 THEN 17005 ELSE IF S=0 O
R K<32 OR K>143 THEN 17004 E
LSE 17007
17005 CALL CHAR(48,"FF"&RPT$
("81",6)&RPT$("FF",9))
17006 CALL KEY(0,K,S):: IF S
<1 THEN 17006 ELSE CALL CHAR
(48,"0038444444444444380010301
010101038"):: GOTO 17004
17007 CALL CHARPAT(K,CH$)
17008 R=12 :: FOR J=1 TO 15
STEP 2
17009 H$=SEG$(CH$,J,1):: CAL
L HEX_BIN(H$,B$)
17010 DISPLAY AT(R,8):B$
17011 H$=SEG$(CH$,J+1,1):: C
ALL HEX_BIN(H$,B$)
17012 DISPLAY AT(R,12):B$ ::
 DISPLAY AT(R,18):SEG$(CH$,J
,2):: R=R+1 :: NEXT J :: DIS
PLAY AT(22,6):CH$ :: GOTO 17
004
17013 SUBEND
17014 SUB HEX_BIN(H$,B$):: H
X$="0123456789ABCDEF" :: BN$
="0000X0001X0010X0011X0100X0
101X0110X0111X1000X1001X1010
X1011X1100X1101X1110X1111"
17015 FOR J=LEN(H$)TO 1 STEP
 -1 :: X$=SEG$(H$,J,1)
17016 X=POS(HX$,X$,1)-1 :: T
$=SEG$(BN$,X*5+1,4)&T$ :: NE
XT J :: B$=T$ :: T$="" :: SU
BEND
```

And to reidentify a character, you just change the numbers and letters in the 16-digit hex code which represents the binary pattern. By writing little routines to switch those digits around, all kinds of things can be done.

For instance, the normal characters always have the top row of dots turned off, to provide spacing between lines of text on the screen. If you want taller characters you will have to double-space the lines, but you can create them by making the numerals and upper case characters consist of the 2nd-7th rows, the 7th row again, and the 8th row - it just happens to work out.

```
18000 SUB HIGHCHAR :: FOR CH
=48 TO 90 :: CALL CHARPAT(CH
,CH$):: CALL CHAR(CH,SEG$(CH
$,3,10)&RPT$(SEG$(CH$,13,2),
2)&SEG$(CH$,15,2)):: NEXT CH
 :: SUBEND
```

I made that a subprogram so you can MERGE it in and use it to modify other character sets.

If we take the hex code apart, 2 digits at a time, and reassemble it backward,

```
100 CALL CLEAR :: FOR CH=33
TO 90 :: CALL CHARPAT(CH,CH$
):: FOR J=1 TO 15 STEP 2 ::
CH2$=SEG$(CH$,J,2)&CH2$ :: N
EXT J :: CALL CHAR(CH,CH2$):
: CH2$="" :: NEXT CH
110 DISPLAY AT(12,1):"?NWOD
EDISPU":"VT EHT DENRUT OHW !
YEH" :: GOTO 110
```

That one was in my first Tips newsletter, years ago, but it is much more effective at assembly speed.

This one shades characters on their left edge by turnon the pixel to the left of the leftmost "on" pixel, if any. Also try it in combination with HIGHCHAR.

```
18001 SUB NEWCHAR3 :: FOR CH
=48 TO 122 :: CALL CHARPAT(C
H,CH$):: FOR J=1 TO 15 STEP
2
18002 CH2$=CH2$&SEG$("0367CD
EF",POS("01234567",SEG$(CH$,
J,1),1),1)&SEG$(CH$,J+1,1)::
 NEXT J :: CALL CHAR(CH,CH2$
):: CH2$="" :: NEXT CH :: SU
BEND
```

This one uses HIGHCHAR to heighten the character and then blanks out three rows. Try following it with NEWCHAR3.

```
18030 SUB NEWCHAR10 :: A$="0
0" :: FOR CH=48 TO 90 :: CAL
L CHARPAT(CH,CH$):: CH$=SEG$
(CH$,3,10)&RPT$(SEG$(CH$,13,
2),2)&SEG$(CH$,15,2)
18031 CH$=SEG$(CH$,1,4)&A$&S
EG$(CH$,7,2)&A$&SEG$(CH$,11,
2)&A$&SEG$(CH$,15,2):: CALL
CHAR(CH,CH$):: NEXT CH :: SU
BEND
```

The next one, which works only on ASCII 97-122, makes tall characters ridiculously elongated above.

```
18050 SUB NEWCHAR20 :: FOR C
H=97 TO 122 :: CALL CHARPAT(
CH,CH$):: CALL CHAR(CH,SEG$(
CH$,7,2)&RPT$(SEG$(CH$,9,2),
4)&SEG$(CH$,11,6)):: NEXT CH
 :: SUBEND
```

This one has the chaacters raised by one line, widened one column at left and two columns at right to make a full 8x8 character which must be double-spaced horizontally and vertically.

```
18090 SUB NEWCHAR27 :: FOR C
H=48 TO 122 :: CALL CHARPAT(
CH,CH$):: CH$=SEG$(CH$,3,10)
&RPT$(SEG$(CH$,13,2),2)&SEG$
(CH$,15,2):: FOR J=1 TO 15 S
TEP 2
18091 CH2$=CH2$&SEG$("014589
CD",POS("01234567",SEG$(CH$,
J,1),1),1)&SEG$("0129",POS("
048C",SEG$(CH$,J+1,1),1),1)
18092 NEXT J :: CALL CHAR(CH
,CH2$):: CH2$="" :: NEXT CH
:: SUBEND
```

Those who have my Nuts & Bolts disks will see how valuable this assembly can be to make instantly available the routines for double height and double width characters, etc., etc. And if you have Todd Kaplan's amazing ALSAVE routine from the Genial Traveler Vol. 1 No. 3, you can imbed them in your XBasic program for fast loading.

And you can merge CHARSUB into any character editor or sprite defining program and, with a bit of modification, use it to convert your creations into fast-loading assembly.

These assembly loads are compatible with my BXB, so you can also load character sets into sets 15 and 16, ASCII 144-159. However, the CHARPAT statement cannot access ASCII above 143, so in this case you must dimension an array in the program you are copying from, as DIM HX$(159), and place the hex codes in the array using the ASCII as the subscript number, such as CALL CHAR(CH+64,CH$) :: HX$(CH+64)=CH$, so that they will be passed to the subprogram. And don't CALL INIT after you have called BXB!

So, now you try creating your own screen fonts!

Memory full. ■

PERSONAL RECORD KEEPING I BASIC

av Jan Alexandersson

I PB 85-2 publicerades ett databasprogram skrivet för Basic men med PRK-modulen eller Statistics-modulen. Se även PB 85-4, PB 88-3 och PB 88-4. Välj Basic och sedan CALL FILES(1) och NEW om du har flexskiva. Därfter skriver du CALL P(9000) och NEW. Sedan laddar du in programmet. Den stora fördelen är att du även kan lagra data i programformat på kassettband eller flexskiva. Nedanstående program är kompaktare en det tidigare programmet men fungerar på samma sätt.

```
100 REM FAST-FILE2 RK/ST
110 REM JAN ALEXANDERSSON
120 REM 1988-10-01
130 CALL SCREEN(8)
140 J=1
150 K=2
160 L=3
170 O=4
180 R=5
190 U=6
200 V=9
210 E=V+K
220 M=28
230 N=O*U
240 C=N-J
250 X=V+U
260 Y=M+O
270 CALL CHAR(91,"0028003844
7C4444")
280 CALL CHAR(92,"0028007C44
44447C")
290 CALL CHAR(93,"0038283844
7C4444")
300 REM START
310 CALL CLEAR
320 CALL D(C,J,M,"WHICH FILE
")
330 CALL D(N,J,M,"N(EW) O(LD
) L(OAD)")
340 CALL KEY(L,T,S)
350 IF S<J THEN 340
360 ON J+POS("NOL",CHR$(T),J
)GOTO 340,370,710,650
370 REM NEW
380 GOSUB 1750
390 CALL H(Q,R,Q,Q)
400 CALL H(Q,U,Q,Q)
410 CALL D(R,J,M,"FIELD-NAME
 TYPE WIDTH DEC")
420 CALL D(C,J,M,"1=CHAR 2=I
NT 3=DEC 4=SCI")
430 FOR F=J TO X
440 CALL A(U+F,J,V,T,D$)
450 IF T>J THEN 610
460 CALL H(Q,V,F,D$)
470 CALL A(U+F,X-K,J,T,W,J,O
)
480 IF T=K THEN 470
490 CALL H(Q,V+J,F,W)
500 I=U
510 IF W=O THEN 570
520 I=X+R*(W=K)+O*(W=L)
530 CALL A(U+F,X+L,K,T,D,J,I
)
540 IF T=K THEN 530
550 CALL H(Q,E,F,D)
560 IF W<L THEN 600
570 CALL A(U+F,N,K,T,D,J,I-J
)
580 IF T=K THEN 570
590 CALL H(Q,X-L,F,D)
600 NEXT F
610 F=F-J
620 FT=F
630 GOSUB 1710
640 GOTO 800
650 REM LOAD
660 CALL D(N,J,M," ")
670 CALL A(C,X-L,X,T,F$)
680 IF T>J THEN 710
690 CALL L(F$,T)
700 IF T=Q THEN 670
710 CALL CLEAR
720 CALL H(J,J,Q,N$)
730 CALL D(J,J,M,N$)
740 D$="19"
750 FOR I=O TO K STEP -J
760 CALL H(J,I,Q,D)
770 D$=D$&SEG$("-0",J-(I=O),
K+(D>V)+(I=O))&STR$(D)
780 NEXT I
790 CALL D(J,X+K,M,D$)
800 CALL D(L,J,M,"ITEM=   PA
GE=    BYTE=")
810 GOSUB 1770
820 FOR F=J TO FT
830 CALL H(J,V,F,D$)
840 CALL D(U+F,J,M,D$)
850 CALL H(J,V+J,F,D)
860 CALL D(U+F,M-J,K,SEG$("C
HINDESC",J+(D-J)*K,K))
870 FOR I=J TO K
880 CALL H(J,V+J+I,F,D)
890 CALL HCHAR(U+F,Y-K+I,N+N
+D+O*O*(D=Q)-7*(D>V))
900 NEXT I
910 NEXT F
920 REM MENU
930 CALL D(C,J,M,"F(IELD) A(
DD) D(ISP) S(AVE)")
940 CALL D(N,J,M,"E(XIT)  L(
OAD)")
950 CALL KEY(L,T,S)
960 IF S<J THEN 950
970 ON J+POS("FADSEL",CHR$(T
),J)GOTO 950,980,1060,1250,1
490,1700,300
980 REM FIELD
990 GOSUB 1750
```

```
1000 FOR F=J TO FT
1010 CALL A(U+F,J,V,T,D$)
1020 IF T>J THEN 1050
1030 CALL H(Q,V,F,D$)
1040 NEXT F
1050 GOTO 920
1060 REM ADD
1070 PT=PT+J
1080 P=PT
1090 GOSUB 1710
1100 REM EDIT
1110 GOSUB 1750
1120 CALL D(R,X-K,0,P)
1130 FOR F=J TO FT
1140 CALL H(J,V+J,F,W)
1150 IF W=J THEN 1190
1160 CALL A(U+F,E,X,T,D,F)
1170 CALL G(-K*(T=K),P,F,D)
1180 GOTO 1210
1190 CALL A(U+F,E,X,T,D$)
1200 CALL G(-K*(T=K),P,F,D$)
1210 IF T>K THEN 1230
1220 NEXT F
1230 GOSUB 1770
1240 GOTO 920
1250 REM DISP
1260 GOSUB 1710
1270 CALL D(R,X-K,K,J)
1280 CALL A(R,X-J,L,T,P,J,PT
)
1290 FOR F=J TO FT
1300 CALL H(J,V+J,F,W)
1310 IF W=J THEN 1360
1320 CALL G(J,P,F,T,D)
1330 IF T=J THEN 1400
1340 CALL D(U+F,E,X,STR$(D))
1350 GOTO 1410
1360 CALL G(J,P,F,T,D$)
1370 IF T=J THEN 1400
1380 CALL D(U+F,E,X,D$)
1390 GOTO 1410
1400 CALL D(U+F,E,X," ")
1410 NEXT F
1420 CALL D(C,J,M,"C(HANGE)
M(ENU) N(EXT)")
1430 CALL KEY(L,T,S)
1440 IF S<J THEN 1430
1450 ON J+POS("CMN",CHR$(T),
J)GOTO 1430,1100,920,1460
1460 P=J-P*(P<PT)
1470 CALL D(R,X-K,0,P)
1480 GOTO 1290
1490 REM SAVE
1500 CALL CLEAR
1510 CALL D(N-K,J,M,"NAME "&
N$)
1520 CALL A(N-K,U,V,T,N$)
1530 IF T=K THEN 1520
1540 CALL H(Q,J,Q,N$)
1550 CALL D(C,J,M," DATE 19"
)
1560 CALL A(C,V,K,T,D)
1570 IF T=K THEN 1560
1580 CALL H(Q,0,Q,D)
1590 CALL A(C,X-L,K,T,D)
1600 IF T=K THEN 1590
1610 CALL H(Q,L,Q,D)
1620 CALL A(C,X,K,T,D)
1630 IF T=K THEN 1620
1640 CALL H(Q,K,Q,D)
1650 CALL D(N,J,M,"FILE NAME
 "&F$)
1660 CALL A(N,E,X,T,F$)
1670 CALL S(F$,T)
1680 IF T=Q THEN 1660
1690 GOTO 300
1700 END
1710 CALL D(R,J,M," ")
1720 FOR F=J TO FT
1730 CALL D(U+F,E,X," ")
1740 NEXT F
1750 CALL HCHAR(C,J,Y,Y+Y)
1760 RETURN
1770 CALL H(J,R,Q,FT)
1780 CALL D(L,U,K,STR$(FT))
1790 CALL H(J,U,Q,PT)
1800 CALL D(L,X-J,L,STR$(PT)
)
1810 CALL H(J,7,Q,H)
1820 CALL H(J,8,Q,G)
1830 CALL D(L,C,R,STR$(X*Y+H
+PT*G))
1840 RETURN                 ■
```

PRK-FILER TILL DIS/VAR 80 FIL

Överföring av PRK- eller Statistics-filer till TI-Writer-fil (DIS/VAR 80). Starta med att stoppa in PRK-modulen (eller Statistics) och välj sedan TI-Basic. Om flexskiva används så skriv CALL FILES (1) och därefter NEW. Skriv sedan CALL FILES (10900) och NEW. Sedan kan programmet laddas in med OLD och köras.

```
80 REM FILKONVERT PRK/STA
90 REM PROGRAMBIT 84-03
92 REM REVIDERAD
95 CALL CLEAR
96 INPUT "RK-FIL          ":R
KFIL$
97 PRINT
98 INPUT "DIS/VAR 80-FIL ":D
VFIL$
100 CALL L("DSK1."&RKFIL$,Y)
110 IF Y=0 THEN 310
120 OPEN #1:"DSK1."&DVFIL$,D
ISPLAY ,VARIABLE 80
130 CALL H(1,5,0,F)
140 PRINT "FIELDS";F
150 CALL H(1,6,0,R)
160 PRINT "RECORDS";R
170 FOR I=1 TO R
180 FOR T=1 TO F
190 CALL H(1,10,T,TP)
200 IF TP=1 THEN 240
210 CALL G(1,I,T,Z,RD)
212 RD=-RD*(Z=0)
215 PRINT I;TAB(5);T;TAB(10)
;RD
220 PRINT #1:RD
230 GOTO 270
240 CALL G(1,I,T,Z,RD$)
245 IF Z=0 THEN 250
247 RD$=""
250 PRINT I;TAB(5);T;TAB(10)
;RD$
260 PRINT #1:RD$
270 NEXT T
280 PRINT #1:"  ": : :
290 NEXT I
295 CLOSE #1
300 END
310 PRINT "NOT LOADED"      ■
```

CREATE LINE 0

```
0 ! "FREEWARE" BY BARRY "AL"
 TRAVER, GENIAL COMPUTERWARE
, 835 GREEN VALLEY DRIVE, PH
ILADELPHIA, PA 19128 (215/48
3-1379) — IF YOU LIKE IT, P
LEASE LET ME KNOW !
100 PRINT :"CHGE/TO/0 by Bar
ry Traver": :"   This progr
am will changea single progr
am line saved in merge forma
t to line 0."
110 PRINT :: INPUT "OLD FILE
? ":F1$ :: IF SEG$(F1$,1,3)<
>"DSK" THEN F1$="DSK1."&F1$
120 PRINT :: INPUT "NEW FILE
? ":F2$ :: IF SEG$(F2$,1,3)<
>"DSK" THEN F2$="DSK1."&F2$
130 OPEN #1:F1$,VARIABLE 163
,INPUT
140 OPEN #2:F2$,VARIABLE 163
,OUTPUT
150 LINPUT #1:M$
160 M$=CHR$(0)&CHR$(0)&SEG$(
M$,3,LEN(M$)-2)
170 PRINT #2:M$:CHR$(255)&CH
R$(255)
180 CLOSE #1 :: CLOSE #2
190 PRINT :"   Now type in t
he followingcommands:": :"
NEW": :"  MERGE "&F2$: :"The
n enter LIST to see the  res
ult."                       ■
```

FORCE 1 – SPEL

Manövrera ditt rymdskepp med piltangenterna ESDX och skjut med ENTER.

```
110 REM * FORCE 1 *
130 REM BY W.K. BALTHROP
140 REM 99'ER VERS. 1.5.1XB
160 CALL CLEAR
170 GOSUB 1510
180 DISPLAY AT(2,10):"FORCE
1"
190 DISPLAY AT(4,3):"LEVEL O
F DIFICULTY:"
200 DISPLAY AT(6,5):"1. BEGI
NNER": :"    2. NOVICE": :"
   3. INTERMEDIATE": :"    4
. SEMI PRO": :"    5. PRO"
220 ACCEPT AT(16,5)VALIDATE(
DIGIT)SIZE(1):L1 :: L=L1*4
230 RANDOMIZE
240 CALL CLEAR
250 FOR CO=1 TO 8 :: CALL CO
LOR(CO,16,1):: NEXT CO
260 COU=0 :: D=1 :: DIS=1100
0 :: IF SC>=25 THEN L=L*2
270 CALL CHAR(88,"0102040810
204080",89,"8040201008040201
")
280 CALL CHAR(90,"03070E1C38
70E0C0",91,"C0E070381C0E0703
")
290 CALL CHAR(92,"070F1F3E7C
F8F0E0",93,"E0F0F87C3E1F0F07
")
300 CALL CHAR(94,"03060C1830
60C080",95,"C06030180C060301
")
310 CALL COLOR(8,1,1):: CALL
 SCREEN(2)
320 CALL CHAR(96,"0101010101
010101808080808080808080")
330 CALL CHAR(98,"0000000000
0000FFFF")
340 CALL COLOR(9,16,1)
350 CALL VCHAR(7,12,96,9)::
CALL VCHAR(7,21,97,9)
360 CALL HCHAR(6,13,98,8)::
CALL HCHAR(16,13,99,8)
370 CALL CHAR(33,"FF00000000
000000010101010101010101")
380 CALL VCHAR(12,15,33):: C
ALL VCHAR(12,18,33):: CALL V
CHAR(10,16,34):: CALL VCHAR(
13,16,34)
390 FOR COL=10 TO 12 :: CALL
 COLOR(COL,7,1):: NEXT COL
400 GOSUB 410 :: GOTO 500
410 CALL CHAR(104,"000000080
0000000000000180000000000000
01C")
420 CALL CHAR(107,"0000003C0
0000000000183C000000000000001
C3E")
430 CALL CHAR(110,"00003C7E1
800000000187EFF3C42")
440 CALL CHAR(112,"000C1E7FF
F3F400000000000000000000000000
080C0008000")
450 CALL CHAR(116,"000000061
F7FFFFF3F2040000000000000000
00080E0F0F0C04020")
460 CALL CHAR(120,"000000000
1073FFFFF1F1F306040000000000
00080E0FCFF")
470 CALL CHAR(123,"FFF8F80C0
602")
480 CALL CHAR(124,"02604CD70
0309C01")
490 RETURN
500 CALL COLOR(12,7,1)
510 GOSUB 700 :: GOSUB 850 :
: GOSUB 1390
520 CALL KEY(0,K,S)
530 CALL POSITION(#1,PO1,PO2
)
540 IF K=13 THEN GOSUB 880
550 T=INT(RND*10):: IF T=4 T
HEN DEV=L/10-INT(RND*L/5)::
DEU=L/10-INT(RND*L/5)ELSE DE
V,DEU=0
560 IF K=69 THEN D1=D1+L/5 :
: SA=SA+L/5
570 IF K=88 THEN D1=D1-L/5 :
: SA=SA-L/5
580 IF K=83 THEN D2=D2+L/5 :
: SB=SB+L/5
590 IF K=68 THEN D2=D2-L/5 :
: SB=SB-L/5
600 DIS=DIS-(L*15):: D=11-IN
T(DIS/1000):: IF DIS<200 THE
N GOSUB 1270 :: GOTO 620
610 IF D<9 THEN GOSUB 1230 E
LSE ON D-8 GOSUB 1240,1250,1
260
620 D1=D1+DEV*(D/11):: D2=D2
+DEU*(D/11)
630 CALL MOTION(#1,D1,D2)
640 IF S=0 THEN 660
650 FOR SM=2 TO 15 :: CALL M
OTION(#SM,SA,SB):: NEXT SM
660 CALL SOUND(-100,800,15)
670 TIME=TIME+1 :: IF TIME=1
000 THEN 980
680 DISPLAY AT(1,3):"SCORE:"
;SC,"TIME:";TIME :: GOTO 520
690 CALL CHARSET
700 DISPLAY AT(24,2):CHR$(92
);"                        "
;CHR$(93)
710 DISPLAY AT(23,3):CHR$(92
);"                      ";C
HR$(93)
720 DISPLAY AT(22,4):CHR$(92
);"                    ";CHR
$(93)
730 DISPLAY AT(21,5):CHR$(92
);"                  ";CHR$(
93)
740 DISPLAY AT(20,6):"Z
           ";CHR$(91)
750 DISPLAY AT(19,7):"Z
         ";CHR$(91)
760 DISPLAY AT(18,8):"Z
       ";CHR$(91)
770 DISPLAY AT(17,9):"Z
     ";CHR$(91)
780 DISPLAY AT(16,10)SIZE(1)
:"^" :: DISPLAY AT(16,19)SIZ
E(1):"_"
790 DISPLAY AT(15,11)SIZE(8)
:"^      _"
800 DISPLAY AT(14,12)SIZE(6)
:"X    Y"
810 DISPLAY AT(13,13)SIZE(1)
:"X" :: DISPLAY AT(13,16)SIZ
E(1):"Y"
820 DISPLAY AT(12,14)SIZE(2)
:"XY"
830 CALL HCHAR(11,16,32,2)
840 RETURN
850 IF SP1=0 THEN D1=INT(L-(
RND*L*2)):: D2=INT(L-(RND*L*
2)):: CALL SPRITE(#1,104,7,I
NT(RND*256)+1,INT(RND*256)+1
,D1/(11/D),D2/(11/D))
860 SP1=1 :: D=1 :: DIS=1100
0
870 L=L+1 :: RETURN
880 CALL COLOR(8,7,1,8,1,1)
890 CALL COINC(#1,87,124,D,C
1)
900 CALL SOUND(20,880,2,990,
2,10000,30,-4,2)
910 IF C1=-1 THEN SP1=0 :: C
ALL DELSPRITE(#1):: GOTO 930
920 RETURN
930 SC=SC+1 :: FOR CS=1 TO 5
 :: CALL SCREEN(7):: CALL SC
REEN(2):: NEXT CS
940 CALL SOUND(500,110,2,-4,
2):: CALL HCHAR(12,16,124,2)
:: CALL HCHAR(11,16,124,2)::
 CALL SOUND(1000,110,2,220,2
,330,2,-8,2)
950 CALL SOUND(1,44000,30)::
 GOSUB 820
960 SA,SB=0 :: D=1 :: DIS=11
000 :: L=L+2 :: GOSUB 850
970 RETURN
980 CALL CLEAR :: CALL SOUND
(1000,440,2,550,2,660,2):: C
ALL SOUND(2000,770,2,880,2,9
90,2)
990 CALL DELSPRITE(ALL)
1000 SCO=SCO+SC
1010 IF SCO>=25 AND SCO-SC1>
```

```
+25 THEN TIME=1000 :: SC1=SC
O :: DISPLAY AT(2,3):"BONUS
GAME" :: GOTO 260
1020 CALL CHARSET :: CALL SC
REEN(6):: CALL DELSPRITE(#1,
#2,#3)
1030 IF SCO>=40 THEN 1080
1040 IF SCO>=30 THEN 1100
1050 IF SCO>=20 THEN 1120
1060 IF SCO>=10 THEN 1140
1070 IF SCO>=5 THEN 1160 ELS
E 1180
1080 DISPLAY AT(4,1):"A VERY
 GOOD BATTLE.":"YOUR NAME WI
LL GO DOWN IN":"HISTORY AS O
NE OF THE"
1090 DISPLAY AT(7,1):"GREATE
ST STARSHIP CAPTAINS":"OF YO
UR TIME.":"YOUR SCORE=";SC :
: GOTO 1200
1100 DISPLAY AT(4,1):"YOU AR
E TO BE CONGRATULATED":"ON Y
OUR FINE MISSION. FEW":"PILO
TS HAVE ATTAINED SUCH"
1110 DISPLAY AT(7,1):"SUCCES
S. GOOD LUCK ON FUTURE":"MIS
SIONS.":"YOUR SCORE=";SC ::
GOTO 1200
1120 DISPLAY AT(4,1):"A FAIR
 SHOWING. THE ALIENS":"HAVE
BEEN TURNED BACK AND":"YOUR
HOME WORLD IS SAFE."
1130 DISPLAY AT(7,1):"YOURE
SCORE=";SC :: GOTO 1200
1140 DISPLAY AT(4,1):"YOUR F
LEET WAS BADLY DAMAGED":"IN
THE FIGHT, BUT YOU":"MANAGED
 TO FIGHT OF THE"
1150 DISPLAY AT(7,1):"ALIEN
ATTACK. BETTER LUCK":"NEXT T
IME.":"YOUR SCORE=";SC :: GO
TO 1200
1160 DISPLAY AT(4,1):"YOUR F
LEET HAS BEEN":"DESTROYED. Y
OU ARE THE":"ONLY SURVIVOR."
1170 DISPLAY AT(7,1):"YOUR H
OME PLANET IS SAFE":"AT LEAS
T UNTIL THE NEXT":"ATTACK.":
"YOUR SCORE=";SC :: GOTO 120
0
1180 DISPLAY AT(4,1):"ALL HO
PE IS LOST IN TRYING":"TO SA
VE YOUR PLANET.":"YOUR MISSI
ON HAS FAILED."
1190 DISPLAY AT(7,1):"AND YO
U ARE DISGRACED.":"YOUR SCOR
E=";SC :: GOTO 1200
1200 DISPLAY AT(10,1):"DO YO
U WISH TO PLAY AGAIN":"ENTER
 (Y/N).";
1210 ACCEPT AT(11,14)SIZE(1)
VALIDATE("YN"):ANS$
1220 IF ANS$="N" THEN END EL
SE SC,SCO,SC1,TIME=0 :: CALL
 MAGNIFY(1):: GOTO 230
1230 CALL MAGNIFY(1):: CALL
PATTERN(#1,103+D):: RETURN
1240 CALL PATTERN(#1,112)::
CALL MAGNIFY(3):: RETURN
1250 CALL PATTERN(#1,116)::
RETURN
1260 CALL PATTERN(#1,120)::
RETURN
1270 CALL MAGNIFY(4):: CALL
POSITION(#1,PO1,PO2):: IF PO
1>8 AND PO2>8 THEN CALL LOCA
TE(#1,PO1-8,PO2-8)
1280 IF PO1<110 AND PO1>36 A
ND PO2<148 AND PO2>88 THEN G
OTO 1330
1290 IF D1>10 OR D2>10 THEN
1310
1300 CALL MOTION(#1,D1*5+40*
SIN(D1),D2*5+40*SIN(D2)):: F
OR TD=1 TO 20 :: NEXT TD
1310 CALL DELSPRITE(#1):: SP
1=0 :: DIS=11000 :: D=1 :: F
OR MO=2 TO 15 :: CALL MOTION
(#MO,0,0):: NEXT MO
1320 SA,SB=0 :: RETURN
1330 CALL POSITION(#1,D3,D4)
:: CALL VCHAR(D3/8+3,D4/8+2,
34,21-D3/8):: CRASH=CRASH+1
:: CALL SOUND(1000,110,2,220
,2,10000,30,-4,2)
1340 CALL VCHAR(D3/8+2,D4/8+
2,32,21-D3/8):: GOSUB 1300
1350 CALL SOUND(300,110,2,22
0,2,20000,30,-8,2)
1360 CALL SOUND(500,440,2,66
0,2,3000,30,-4,2)
1370 CALL SOUND(600,110,2,22
0,2,5000,30,-8,2)
1380 CALL SOUND(1000,220,2,3
30,2,1000,30,-8,2):: SA,SB=0
 :: GOTO 980
1390 Z$="81611638C4241211"
1400 CALL COLOR(13,16,1)
1410 Z1$="0000001000000000"
1420 ST=2
1430 CALL CHAR(128,Z1$)
1440 CALL CHAR(129,"00"):: C
ALL CHAR(130,"00"):: CALL CH
AR(131,"00")
1450 FOR ST=2 TO 15
1460 STA1=INT(RND*256)+1 ::
STA2=INT(RND*256)+1
1470 CALL SPRITE(#ST,128,16,
STA1,STA2)
1480 NEXT ST
1490 RETURN
1500 END
1510 DISPLAY AT(11,8):"*****
******"
1520 DISPLAY AT(12,8):"* FOR
CE 1 *"
1530 DISPLAY AT(13,8):"*****
******"
1540 GOSUB 410 :: DISPLAY AT
(21,1):"PRESS ANY KEY TO CON
TINUE"
1550 CALL KEY(0,K,S):: IF S<
>0 THEN CALL SOUND(-1,40000,
30):: CALL CLEAR :: RETURN E
LSE CALL MAGNIFY(1)
1560 T1=INT(RND*192)+1 :: T2
=INT(RND*256)+1 :: CALL SPRI
TE(#1,104,2,T1,T1,INT(RND*7)
-3,INT(RND*7)-3)
1570 D=0
1580 D=D+1 :: IF D<9 THEN GO
SUB 1230 ELSE ON D-8 GOSUB 1
240,1250,1260
1590 CALL SOUND(-4000,600,(1
1-D)*3,400,(D-1)*3)
1600 CALL KEY(0,K,S):: IF S<
>0 THEN CALL DELSPRITE(#1)::
 CALL SOUND(-1,40000,30):: C
ALL CLEAR :: RETURN
1610 IF D<11 THEN 1580 ELSE
CALL DELSPRITE(#1):: GOTO 15
50
```

■

MG LINE TABLE

```
100 CALL CLEAR :: PRINT :"OU
TPUT TO PRINTER? (Y/N) N" ::
 ACCEPT AT(23,26)SIZE(-1)VAL
IDATE("YN"):A$ :: IF A$="Y"
THEN OPEN #1:"PIO" :: P=1
110 CALL CLEAR :: CALL PEEK(
-31952,A,B,C,D):: A=A*256+B-
65536 :: C=C*256+D-65536 ::
PRINT #P:"    PROGRAM INFORM
ATION": :"Line Number Table"
120 PRINT #P: :"Start Adress
 ";A:"End Adress ";C: : :" L
ine    Bytes  Start":" Numb
er  Used    Address":"  ____
_ _____ ______"
130 FOR I=C-3 TO A STEP -4 :
: CALL PEEK(I,B,D,E,F):: B=B
*256+D :: E=E*256+F-65536 ::
 CALL PEEK(E-1,D):: D=D+5
140 PRINT #P,USING "  #####
   ###  ######":B,D,E :: T=T
+D :: CALL KEY(0,D,E):: IF E
 THEN CALL SCREEN(6):: GOSUB
 160
150 NEXT I :: A=(A-C-1)/-4 :
: PRINT #P: : :TAB(8);"Total
 Bytes =";T:"  Number of Li
nes =";A:"Average Bytes/Line
 =";INT(T/A):: STOP
160 CALL KEY(0,D,E):: IF E<1
 THEN 160 ELSE CALL SCREEN(8
):: RETURN
```

■

GOBBLEGOOK

```
100 REM filename: "gobblegoo
k" (formerly "simp")
110 REM  purpose: To generat
e gobbledigook
120 REM   author: jpg & jdr
(after other like efforts) 9
/82 (modified 5/84)
130 REM ────────────────────
────────────────────────────
140 CALL CLEAR :: PRINT "THI
S PROGRAM ILLUSTRATES A  PRA
CTICAL APPLICATION OF THEWOR
D/WRAP AND PAUSEPRINT    SUB
ROUTINE CONTAINED IN IT."
150 PRINT :"SEE PROGRAM LIST
ING FOR MOREINFORMATION." ::
 FOR I=1 TO 2000 :: NEXT I
160 CALL CLEAR :: PRINT "Gob
bledigook Generator": :"Outp
ut to where?": :"  1. Screen
": :"  2. Printer": :"What i
s your choice?";
170 CALL KEY(0,K,S):: IF S=0
 THEN 170 ELSE IF K<49 OR K>
50 THEN 170 ELSE K=K-48 :: P
RINT K: : :: N=K-1
180 IF N THEN INPUT "Output
device? ":O$ :: OPEN #1:O$,O
UTPUT :: PRINT
190 INPUT "How many paragrap
hs? ":P :: PRINT :: INPUT "H
ow many sentences per para-
 graph? ":S
200 RANDOMIZE :: DIM X$(40):
: FOR I=1 TO 40 :: READ X$(I
):: NEXT I :: PRINT :"Press
any key to pause;      press
 any key to resume."
210 FOR I=1 TO P :: PRINT #N
: :"     ";:: FOR J=1 TO S :
: A=INT(RND*10+1):: B=INT(RN
D*10+11):: C=INT(RND*10+21):
: D=INT(RND*10+31)
220 M$=X$(A):: GOSUB 670 ::
M$=X$(B):: GOSUB 670 :: M$=X
$(C):: GOSUB 670 :: M$=X$(D)
:: GOSUB 670 :: PRINT #N:" "
;:: NEXT J :: PRINT #N :: NE
XT I
230 IF N THEN CLOSE #1
240 STOP
250 REM ********************
****************************
260 REM data statements
270 DATA "In particular,","O
n the other hand,"
280 DATA "However,","Similar
ly,","In this regard,"
290 DATA "As a resultant imp
lication,","For example,"
300 DATA "Based on integral
subsystem considerations,"
310 DATA "Thus,","With respe
ct to specific goals,"
320 REM ********************
****************************
330 DATA "a large portion of
 effective information"
340 DATA "a constant flow of
 effective information"
350 DATA "the characterizati
on of specific criteria"
360 DATA "initialization of
critical subsystem developme
nt"
370 DATA "the product config
uration baseline"
380 DATA "the fully integrat
ed test program"
390 DATA "any associated sup
porting element"
400 DATA "the incorporation
of additional mission constr
aints"
410 DATA "the independent fu
nctional principle"
420 DATA "a primary interrel
ationship between system and
/or subsystem technologies"
430 REM ********************
****************************
440 DATA "must utilize and b
e functionally interwoven wi
th"
450 DATA "maximizes the prob
ability of project success a
nd minimizes the cost and ti
me required for"
460 DATA "adds explicit perf
ormance limits to"
470 DATA "necessitates that
urgent consideration be appl
ied to"
480 DATA "requires considera
ble systems analysis and tra
de-off studies to arrive at"
490 DATA "is further compoun
ded, when taking into accoun
t"
500 DATA "presents extremely
 interesting challenges to"
510 DATA "recognizes the imp
ortance of other systems and
 the necessity for"
520 DATA "effects a signific
ant implementation to"
530 DATA "adds overriding pe
rformance constraints to"
540 REM ********************
****************************
550 DATA "the sophisticated
hardware."
560 DATA "the anticipated fi
fth generation equipment."
570 DATA "the subsystem comp
atibility testing environmen
t."
580 DATA "the structural des
ign, based on system enginee
ring concepts."
590 DATA "the preliminary qu
alification limits."
600 DATA "the philosophy of
commonality and standardizat
ion."
610 DATA "the evolution of s
pecifications over a given t
ime period."
620 DATA "the greater flight
-worthiness concept."
630 DATA "any discrete confi
guration mode."
640 DATA "the total system r
ational."
650 STOP
660 REM word/wrap & pausepri
nt subroutine by Barry Trave
r
670 M$=M$&" " :: P1=0
680 P2=POS(M$," ",P1+1):: PR
INT #N:SEG$(M$,P1+1,P2-P1);:
: IF P2=LEN(M$)THEN RETURN
690 P1=P2 :: CALL KEY(0,K,ST
):: IF ST<1 THEN 680
700 CALL KEY(0,K,ST):: IF ST
<1 THEN 700 ELSE 680       ■
```

```
1 !** RAD 1 BLIR RAD 0     **
2 !** RAD 2 BLIR RAD 32767**
10 !************************
20 !* RAD NR 0 & LISTSKYDD *
30 !* (repris fr. PB 88-3) *
40 !* AV F.NILSSON 1988    *
50 !* XB + EM              *
60 !************************
70 CALL CLEAR
80 CALL PEEK(-31952,ST1,ST2)
! HÄMTA STARTADRESS
90 CALL PEEK(-31950,SL1,SL2)
! HÄMTA SLUTADRESS
100 STA=ST1*256+ST2-65536
110 SLU=SL1*256+SL2-65536
120 FOR I=STA TO SLU STEP 4
130 CALL PEEK(I,RAD1,RAD2)!
HÄMTA RADNUMMER I TABELL
140 RAD=RAD1*256+RAD2
150 PRINT RAD
160 IF RAD=1 THEN 190 ! ÄR D
ET RÄTT RAD? JA ->190
170 NEXT I
180 END
190 CALL LOAD(I,0,0)! SKRIV
IN DET NYA RADNUMRET
200 END                    ■
```