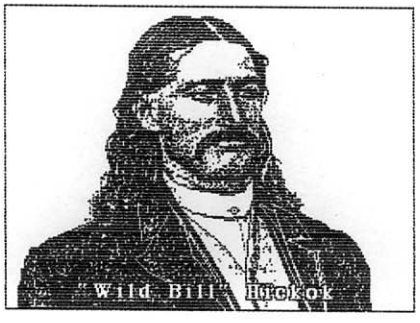
PROGRAM BITEN 92-3


George Armstrong Custer


Geronimo


"Wild Bill" Hickok

INNEHÅLL

ISSN 0281-1146

REDAKTÖREN

Jag har hoppat över Fast Extended Basic 88/02 eftersom den handlar om adresslappar på franska och inte har några nya kommandon. Programmet QUICKLABEL har ASCII-koder över 127 som ej kan listas. Vid programkörningen behövs en 7-bitars skrivare som skriver ut CHR$(142) som CHR$(14). En 8-bitars skrivare tolkar detta som bokstaven Ä.

TI*MES har en ny adress för medlemmar (GBP 15 per år): TI99/4A USERS GROUP(UK), c/o Mr. A.Bryce, 51 Dumbuie Avenue, Silverton, Dumbarton, Scotland, G82 2JH .

NOTUNG SOFTWARE har följande nytt: (se även PB 92-1 sid 16)

TI Casino Supplement Disk	USD 5
Disk of the Old West	USD 15
DINOSAURS(alla 3 paket)	USD 25■

```
100 REM ALPHA TO OMEGA
110 REM TI*MES 9 p.10
120 REM by Stephen Shaw
130 CALL CLEAR
140 FOR I=1 TO 24
150 READ A$
160 CALL CHAR(100+I,A$)
170 PRINT CHR$(100+I);" ";
180 IF I/8<>INT(I/8)THEN 200
190 PRINT : :
200 NEXT I
210 GOTO 210
220 DATA 0000344848484834
230 DATA 0038445858447840
240 DATA 0000645810101010
250 DATA 0038201028484830
260 DATA 00001C201020201C
270 DATA 0020182040201020
280 DATA 0058242424240404
290 DATA 0030484878484830
300 DATA 0000101010101008
310 DATA 0000485060504848
320 DATA 0000201010282844
330 DATA 0000484848784440
340 DATA 0000482828283020
350 DATA 0030403040201020
360 DATA 0000384444444438
370 DATA 00007C2828282828
380 DATA 0030484870404020
390 DATA 0000344848484830
400 DATA 00003C5010101010
410 DATA 0000444444444830
420 DATA 0010385454543810
430 DATA 000064181010284C
440 DATA 0010929292541010
450 DATA 0000444454545428 ■
```

Redaktör: Jan Alexandersson
Medlemsregister: Claes Schibler
Tryckning av tidning: Åke Olsson
Programbankir: Börje Häll

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

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

Datainspektionens licens-nr 82100488

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

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

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

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

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

Artiklar sändes till redaktören:
Jan Alexandersson
Springarvägen 5, 3tr
142 61 TRÅNGSUND
Tel. 08-771 0569
För år 1992 planeras fyra nummer.

CALL SOUND FÖR OLIKA LJUD

```
10 REM *********
20 REM * SOUND *
30 REM *********
40 REM BY AL KANADA
50 REM 99'ER 82-06
55 CALL CLEAR
60 PRINT "  CALL SOUND": : :
 :"1 DOOR CHIMES": :"2 SIREN
": :"3 BEE": :"4 SHORTWAWE R
ECEIVER": :"5 RADIO TELEPRIN
TER"
70 PRINT :"6 FOOTSTEPS": :"7
 SWORD FIGHT": :"8 ENGINE":
:"9 END"
80 CALL KEY(3,KEY,STA)
90 IF (KEY<49)+(KEY>57)THEN
80
95 ON KEY-48 GOTO 110,180,29
0,350,430,540,660,760,880
100 REM DOOR CHIMES
110 FOR A=0 TO 30 STEP 5
120 CALL SOUND(-99,698,A,192
4,A)
130 NEXT A
140 FOR A=0 TO 30 STEP 5
150 CALL SOUND(-99,554,A,152
7,A)
160 NEXT A
165 GOTO 80
170 REM SIREN
180 N=1
190 FOR F=700 TO 900 STEP 5
200 CALL SOUND(-99,F,0)
210 NEXT F
220 FOR F=900 TO 700 STEP -8
230 CALL SOUND(-99,F,0)
240 NEXT F
250 N=N+1
260 IF N=4 THEN 270 ELSE 190
270 GOTO 80
280 REM BEE
290 N=1
300 CALL SOUND(-99,RND*8+110
,RND*10)
310 N=N+1
320 IF N=75 THEN 330 ELSE 30
0
330 GOTO 80
340 REM SHORTWAVE RECEIVER
350 N=1
360 F=RND*15000+110
370 A=RND*30
380 CALL SOUND(-99,111,30,11
1,30,F,A,-8,30-A)
390 N=N+1
400 IF N=100 THEN 410 ELSE 3
60
410 GOTO 80
420 REM RADIO TELEPRINTER
430 N=1
440 CALL SOUND(22,2975,0)
450 FOR D=1 TO 5
460 S=850*INT(RND*2)
470 CALL SOUND(22,2125+S,0)
480 NEXT D
490 CALL SOUND(31,2125,0)
500 N=N+1
510 IF N=30 THEN 520 ELSE 44
0
520 GOTO 80
530 REM FOOTSTEPS
540 N=1
550 X=INT(RND*5)
560 IF X=2 THEN 620
570 CALL SOUND(5,-3,5)
580 CALL SOUND(30,-7,20)
590 CALL SOUND(500,-7,30)
600 N=N+1
610 IF N=30 THEN 640 ELSE 55
0
620 CALL SOUND(60,-7,20)
630 GOTO 590
640 GOTO 80
650 REM SWORD FIGHT
660 N=1
670 FOR A=0 TO 30 STEP 15
680 CALL SOUND(-99,1000,A,32
50,A,6750,A)
690 NEXT A
700 FOR D=1 TO RND*200
710 NEXT D
720 N=N+1
730 IF N=30 THEN 740 ELSE 67
0
740 GOTO 80
750 REM ENGINE
760 FOR N=1 TO 8
770 CALL SOUND(60,220,8,-5,0
)
780 CALL SOUND(60,220,8,-5,5
)
790 NEXT N
800 CALL SOUND(80,220,8,-5,0
)
810 FOR F=1000 TO 5000 STEP
20
820 CALL SOUND(-99,111,30,11
1,30,F,30,-8,0)
830 NEXT F
840 FOR F=4000 TO 800 STEP -
50
850 CALL SOUND(-99,111,30,11
1,30,F,30,-8,0)
860 NEXT F
870 GOTO 80
880 END
```

■

SWEDLOW TI BITS * 20-21 *

by Jim Swedlow, USA

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

HOW YOUR TI SAVES TEXT FILES

In our TI world, most text is saved in Display Variable 80 (DV80) files. This is what the TI Writer Editor uses. What is DV80?

Display means that the file is saved using ASCII characters. If you use a disk sector editor to look at a DV80 file, you will see that the text looks just about the same as it was written. Internal files are not always as easy to read (but that is another column).

A file is made up of records. In a DV80 file, each record contains one line of text as it appears in the Editor. Variable means that each record is only as long as the text line.

Consider these two lines of text:

```
TI
99/4A
```

When this is saved on disk, there will be two records in the file. The first record is "TI" and the second is "99/4A". Each record is preceded by the number of characters in the record in Hex. Hex FF is used to mark the end of the file. The file would look like this:

```
Hex  02 54 49 05 39 39 2F 34 41 FF
ASC     T  I    9  9  /  4  A
```

In a Display Fixed 80 (DF80) file, each record still contains one text line but is exactly 80 characters long. Your 4A pads each record by adding the required number of spaces to the end of each text line.

WHY YOU NEED TO KNOW HOW THE REST OF THE WORLD SAVES TEXT FILES

As a loyal TI user you may not think that you need to know how others (MS-DOS, CPM, etc) save text files. If you do any work with modems, however, you do.

The reason is that you may down load a text file and find that it is Display Fixed 128 (DF128). Why? There is a standard protocol in the TI world for transfering files using XMODEM. It was designed by Paul Charlton (creator of FAST TERM). The first record is NOT the first line of text. Instead, it is the disk header sector (which describes the file in a manner that can be read by the disk controller).

If the first record is not the header, however, your modem program (TELCO, FAST TERM, MASS TRANSFER, etc) assumes that you are talking to a non TI system and will save the file as DF128.

The reason, then, that you need to know how other systems save text files is that you may get one.

HOW THE REST OF THE WORLD SAVES TEXT FILES

The short answer is DF128. But there is more. Unlike a DF80 file, there is no padding. Instead, all of the text lines are run together. The end of each text line is marked with a carriage return <CR or CHR$(13)> and a line feed <LF or CHR$(10)>.

One record may have one, two or more text lines, each ending with a CR and LF. If there is not enough room left in a record for the next text line, enough of the line is added to the record to bring it to 128 characters. The rest of the text line starts the next record - followed by a CR and LF. The end of the file is marked with CHR$(26), which in the IBM and CPM worlds is <CTRL Z>.

Remember our sample text?

```
TI
```

99/4A

Since it is well under 128 characters, the file will only contain one record:

```
54 49 0D 0A 39 39 2F 34 41 0D 0A 1A
 T  I        9  9  /  4  A
```

Hex 0D 0A is a CR and LF. Hex 1A is the end of file marker CHR$(26).

CONVERTING FILES

There are a number of programs that convert files from DF128 to DV80 or from DV80 to DF128. Some of the assembly ones are quite fast. There is a program in this issue called CONVERT. It does those conversions and two others.

A little background. Sometimes you may look at a file and notice that there are no CR's. If you reformat such a document, everything will be jumbled into one big paragraph. TI Writer stops reformatting when it hits a CR. FUNNELWEB stops when it hits a CR or a blank line. Either way, the document is a mess.

CONVERT, when converting a DF128 file to DV80, can add a CR to blank lines, to the end of paragraphs and to lines that start with a period (Formatter commands). This takes a little longer but it makes the file much easier to edit. Also, CONVERT can add CR's to DV80 files that lack them.

```
100 ! CONVERT
110 ! Version 1.0
120 ! 09 Aug 88
130 ! By Jim Swedlow
140 ! Based on XPREP
      By Carl Walters
150 !
160 DISPLAY ERASE ALL :: CAL
L SCREEN(5):: FOR A=0 TO 14
:: CALL COLOR(A,16,1):: NEXT
 A
170 FOR A=1 TO 4 :: READ T$(
A):: NEXT A
180 N$=CHR$(13)&CHR$(10):: Z
$=CHR$(26):: C$=CHR$(13):: G
OTO 300
190 DATA DF128 -> DV80  add
CR's,DF128 -> DV80  no CR's,
DV80  -> DV80  add CR's,DV80
  -> DF128
200 CALL KEY :: Q$,S,P,K,I$,
W$ :: !@P-
210 !
220 ! STRING CHECK SUB
230 !
240 P=1 :: IF I$=" " OR I$="
" THEN I$="" :: RETURN ELSE
IF ASC(I$)=46 THEN RETURN EL
SE P=0 :: RETURN
250 !
260 ! CLOSE FILES AND END
270 !
280 CLOSE #1 :: CLOSE #2 ::
DISPLAY AT(19,1)BEEP:"DONE"
290 FOR P=1 TO 100 :: NEXT P
300 !
310 ! TITLE SCREEN
320 !
330 DISPLAY AT(5,5):"CONVERT
  Version 1.0": : : : : : :
:"Press For"
340 FOR S=1 TO 4 :: DISPLAY
AT(14+S,1):STR$(S);" ";T$(S)
:: NEXT S :: DISPLAY AT(19,1
)BEEP:"5 End Program"
350 !
360 ! PICK FUNCTION
370 !
380 CALL KEY(0,K,S):: IF K<4
49 OR K>53 THEN 380 ELSE K=K-
48 :: IF K=5 THEN DISPLAY ER
ASE ALL :: STOP
390 DISPLAY AT(13,1):T$(K):
:"Input File:   DSK": :"Outp
ut File:  DSK": : : :
400 ACCEPT AT(15,18)BEEP:I$
410 ACCEPT AT(17,18)BEEP:W$
420 !
430 ! OPEN FILES & INIT
440 !
450 DISPLAY AT(19,1):"Workin
g . . . ."
460 IF K>2 THEN OPEN #1:"DSK
"&I$,INPUT ELSE OPEN #1:"DSK
"&I$,INPUT ,FIXED 128
470 IF K=4 THEN OPEN #2:"DSK
"&W$,OUTPUT,FIXED 128 ELSE O
PEN #2:"DSK"&W$,OUTPUT
480 A=1 :: W$="" :: ON K GOT
O 720,570,490,650
490 !
500 ! DV80 -> DV80 ADD CR's
510 !
520 LINPUT #1:I$ :: GOSUB 21
0 :: IF EOF(1)THEN 550
530 IF A THEN IF P THEN PRIN
T #2:I$;C$ :: GOTO 520 ELSE
Q$=I$ :: A=0 :: GOTO 520
540 IF P THEN PRINT #2:Q$;C$
:I$;C$ :: A=1 :: GOTO 520 EL
```

```
SE PRINT #2:Q$ :: Q$=I$ :: G
OTO 520
550 IF A=0 THEN IF P THEN PR
INT #2:Q$;C$ ELSE PRINT #2:Q
$
560 PRINT #2:I$;C$:C$ :: GOT
O 250
570 !
580 ! DF128 -> DV80 NO CR'S
590 !
600 LINPUT #1:I$ :: W$=W$&I$
 :: K=1 :: S=LEN(W$)
610 IF SEG$(W$,K,1)=Z$ THEN
250 ELSE IF K>S THEN IF EOF(
1)THEN 250 ELSE W$="" :: GOT
O 600
620 P=POS(W$,N$,K):: IF P TH
EN PRINT #2:SEG$(W$,K,P-K)::
 K=P+2 :: GOTO 610
630 P=POS(W$,Z$,K):: IF P TH
EN PRINT #2:SEG$(W$,K,P-K)::
 GOTO 250
640 W$=SEG$(W$,K,255):: IF E
OF(1)THEN PRINT #2:W$ :: GOT
O 250 ELSE 600
650 !
660 ! DV80 -> DF128
670 !
680 LINPUT #1:I$ :: IF ASC(I
$)=128 THEN I$=" "
690 W$=W$&I$&N$ :: P=LEN(W$)
700 IF P>128 THEN PRINT #2:S
EG$(W$,1,128):: W$=SEG$(W$,1
29,255)
710 IF EOF(1)THEN PRINT #2:W
$&Z$ :: GOTO 250 ELSE 680
720 !
730 ! DF128 -> DV80 ADD CR'S
740 !
750 LINPUT #1:I$ :: W$=W$&I$
 :: K=1 :: S=LEN(W$)
760 IF SEG$(W$,K,1)=Z$ THEN
820 ELSE IF K>S THEN IF EOF(
1)THEN 820 ELSE W$="" :: GOT
O 750
770 P=POS(W$,N$,K):: IF P TH
EN I$=SEG$(W$,K,P-K):: K=P+2
 ELSE 800
780 GOSUB 210 :: IF A THEN I
F P THEN PRINT #2:I$;C$ :: G
OTO 760 ELSE Q$=I$ :: A=0 ::
 GOTO 760
790 IF P THEN PRINT #2:Q$;C$
:I$;C$ :: A=1 :: GOTO 760 EL
SE PRINT #2:Q$ :: Q$=I$ :: G
OTO 760
800 P=POS(W$,Z$,K):: IF P TH
EN I$=SEG$(W$,K,P-K):: GOTO
820
810 W$=SEG$(W$,K,255):: IF E
OF(1)THEN I$=W$ ELSE 750
820 IF A=0 THEN GOSUB 210 ::
 IF P THEN PRINT #2:Q$;C$ EL
SE PRINT #2:Q$
830 PRINT #2:I$;C$:C$ :: GOT
O 250
```

NOTE TO OTHER USER GROUPS

I have now written 20 XB Columns and 20 in the TI BITS series. From time to time, other user groups have published some of my work in their news letters. If anyone wants a complete set, please send me two (2) DSSD disks or four (4) SSSD disks, a return mailer and return postage. My address is 7301 Kirby Way, Stanton, CA 90680, USA.

OCTOBER 28, 1983

This month marks the fifth anniversary of Black Friday, the day TI dumped the 4A into the unmapped territory of orphanhood.

Who would have believed it? Five years later and we are still getting better software every year. Consider this very incomplete list:

1988	1983
FUNNELWEB	TI Writer
TELCO	TE II
DISK MANAGER 1000	Disk Manager II
TI ARTIST	Video Graphics
TI BASE & PR BASE	Personal Record Keeping
JIFFY FLIER, CLASS & CERTIFICATE 99	Nothing
ARCHIVER	Nothing

Another indicator is TI Bulletin Board Systems (BBS). I recently saw a list of 165 BBS that support the TI. Not bad!

There are problems. Membership in User Groups is dropping. Some groups are in trouble. New members are harder to find. The migration from TI to other systems continues. Some who have given us first rate material are heard from less often.

We are, however, stronger today than anyone expected. The sixth anniversary should see us still around and still a hale and hardy orphanage.

NEWT ARMSTRONG PRINTER RIBBON TRICK

Within days of putting a new ribbon into your printer, the type seems to fade from black to grey.

Newt has a trick that helps. When he buys a new ribbon, he only uses it for final copies. He uses the old ribbon for drafts and such. That way he gets a lot more use from a ribbon. I keep my new one in a zip-lock baggie.

ARCHIVING FILES

The standard program in the TI world for archiving files is ARCHIVER. Archiving files is vital for transferring programs over phone lines.

Many programs contain more than one file. For example, TELCO comes with 32 files. Suppose you wanted to down load it from Compuserve or a BBS. Can you imagine how long it would take to transfer this program, one file at a time? It would be a pain in the neck (not to mention the check book). Also, if you missed a file, you might not be able to use the program.

Think of going on a vacation. You could transfer all of your clothing, item by item to your car, to the plane, etc. Or you could pack them in a suit case and then unpack them when you arrived.

ARCHIVER packs all of the files into one big file. You transfer only that file. Then you use ARCHIVER to unpack the files so you can use the program.

ARCHIVER also compresses the packed file to make it smaller (like sitting on your suit case). The compressed, archived file reduces transfer time and costs.

ARCHIVER 3.01

Barry Boone has rewritten ARCHIVER from the ground up. There are several nice improvements.

You can uncompress and unpack with just one menu choice. In earlier versions, you had to uncompress the file and then, in a second step, unpack it.

Another enhancement is that compressed files are smaller. This saves transfer costs, which is a plus.

The program is a snap to use. There is only one menu, and each choice is self evident. If you are using an older version, upgrading is easy (just replace the file). You will have no problems using the new version.

Now for the bad news. Barry says that this will be his LAST program for the TI unless there is a response to this FAIRWARE offering. If you are one of those who use ARCHIVER but have not sent him a contribution, do this NOW:

Get out your checkbook and write a check.

Write a note saying that you use and like ARCHIVER.

Send your note and your check to:
Barry Boone
PO Box 1233
Sand Springs, OK 74063, USA

Do it today. You won't be sorry.

A WORD TO THE WISE * OR *
IF THE SHOE FITS . . .

One airs clean laundry, not dirty stuff.

GETTING THE BUGS OUT

Did you ever wonder why software and hardware glitches are called bugs? I have this from two sources, so it might even be true.

In the early days, computers were made of relays and vacuum tubes. The first computers took up a whole room, required an extraordinary amount of electricity and were, at best, balky. A big, big computer had a whopping 4K of memory. It could not do a quarter of what a 4A does.

These early computers were problem prone. Tubes would blow. The relays would hang up. Heat was always a worry. The relays would also trap insects. Every now and then, you see, they had to stop and get the bugs out. That's where "debugging" came from.

Enjoy. ■

TIPS FROM TIGERCUB #62

Tigercub Software
156 Collingwood Ave.
Columbus, OH 43213

Dec. 1990

My stock of Tigercub Software catalogs is depleted and it would not pay me to reprint it. Therefore I have released all copyrighted Tigercub programs, except the Nuts & Bolts Disks, for free distribution providing that no price or copying fee is charged. All of my Tigercub programs have been added to my TI-PD library and are cataloged, by category, in TI-PD catalog #4.

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

My TI-PD library now consists of 452 disks of fairware (by author's permission only) and public domain, all arranged by category and as full as possible, provided with loaders by full program name rather than filename, Basic programs converted to XBasic, etc. The price is just $1.50 per disk(!), post paid if at least eight are ordered. TI-PD catalog #4 listing all titles and authors, is available for $1 which is deductible from the first purchase.

According to Charles Good, running a program containing CALL SAY on a beige console without the speech synthesizer attached will cause a lockup.

On a black and silver console, there is no lockup but program execution can be greatly delayed. To avoid that, CALL PEEK(-28672,@) at the beginning of the program and add IF @=96 before each CALL SAY (remember that, IF causes program execution to skip to next program line if not true!), or IF @<>96 THEN to skip over the CALL SAYs.

In Tips #60 I presented a routine to find the lowest power of 7 which contains six 7s in sequence. My version took 24 minutes to find the answer on my TI-99/4A. Several users tried this on a Geneve. The NUTI News of the Nittany UG, Oct 1990 reports that on a 9640 (MDOS 0.97H) with TI XBasic loaded through GPL (speed 5) it ran in 11 min. 33.86 seconds, and with MYARC Advanced Basic V2.99A loaded through GPL it ran in 4 min. 58.62 seconds!

Now, from the TI*MES of England, here is a method using a level of math beyond my comprehension that will solve the problem on an ordinary TI in 6 minutes and 17 seconds!

```
100 ! FASTER WAY John Seager
110 CALL CLEAR :: DIM ELEM(2
6):: ELEM(0)=7 :: POWER,SS=0
 :: DISPLAY AT(1,1):"7 TO TH
E POWER OF"
120 ELM=SS :: SS,CARRY=0 ::
POWER=POWER+1
130 DIS$=STR$(ELEM(ELM)):: F
OR I=ELM-1 TO 0 STEP -1 :: D
IS$=DIS$&RPT$("0",10-LEN(STR
$(ELEM(I))))&STR$(ELEM(I)):: 
 NEXT I
140 DISPLAY AT(1,19):STR$(PO
WER);"=": : :DIS$
```

```
150 FOR I=6 TO LEN(DIS$)STEP
 6 :: IF SEG$(DIS$,I,1)<>"7"
 THEN 190
160 FOR J=I-5 TO I :: IF SEG
$(DIS$,J,6)<>"777777" THEN 1
80 ELSE DISPLAY AT(24,1):"AN
Y KEY TO CONTINUE"
170 CALL KEY(0,K,S):: IF S=0
 THEN 170 :: DISPLAY AT(24,1
):: J=I
180 NEXT J
190 NEXT I
200 ELEM(SS)=ELEM(SS)*7+CARR
Y :: IF ELEM(SS+1)=0 AND ELE
M(SS)<1.E+10 THEN 120
210 CARRY=INT(ELEM(SS)/1.E+1
0):: ELEM(SS)=ELEM(SS)-CARRY
*1.E+10
220 SS=SS+1 :: GOTO 200
```

And if you think that is fast, the Autumn '90 edition of TI*MES contains a Mini-memory program to solve the program in 2 SECONDS! And an assembly version that will search to the 10,000 power and find 52 strings of six 7's in an hour and a half!

Here's a puzzler for you. Can you figure out why that 1000-microsecond CALL SOUND is cut short?

```
100 CALL CLEAR
110 DISPLAY AT(12,1):"Filena
me? DSK" :: ACCEPT AT(12,14)
BEEP:F$
120 ON ERROR 130 :: OPEN #1:
"DSK"&F$ :: STOP
130 GOSUB 140 :: RETURN 110
140 CALL SOUND(1000,110,0,-4
,0):: DISPLAY AT(24,1):"CAN'
T OPEN FILE" :: RETURN
```

I recently programmed a diskfull of gospel songs, and in each one I used this formula to set up an array containing the frequencies for 3 octaves:

```
DIM N(36) :: F=110 :: FOR J
=1 TO 36 :: N(J)=INT(F*1.059
463094^(J-1)+.5):: NEXT J
```

At the end of each selection I put CALL INIT :: CALL LOAD(-31961,149) I don't remember where I learned that one, but it clears the screen, sets all colors and characters to default, deletes sprites, and looks for a LOAD program on DSK1.

The LOAD program has a routine to play each song one after another, but one song crashed with a BAD VALUE error even though it had previously been OK. I found that this was the only song that actually used N(1). The value should have been 110 but it had somehow changed to 24263 which the program line multiplied by 2, therefore out of range.

I found that the routine was correctly giving N(1) a value of 110 the first time but after the CALL LOAD it always had the 24263 value. Substituting other values for 110, I found that any value was being multiplied by 220.5727273, rounded off.

Further experimentation revealed that the problem was being caused by the ^ (exponentiation sign, shift 6 on your keyboard, in case someone prints this through the Formatter!). So I wrote this little routine to experiment with:

```
100 FOR J=1 TO 10 :: PRINT
2^J   :: NEXT J :: CALL INIT
 :: CALL LOAD(-31961,149)
```

I saved that as DSK1.TEST and then wrote another one 100 RUN "DSK1.TEST", saved that as DSK1.LOAD, and then entered RUN "DSK1.TEST".

It printed out the proper values time after time, so I changed the 2^J to read 2^(J-1). The first time around, the first value was 1 as it should be - the computer will consider any number to the power of 0 to have a value of 1. But, the next time around, the first value was F0.57000101 !

That was not even a valid numerical representation, so I changed the formula to 2^(J-1)*2, expecting it to crash. Instead, it gave me a value of 441.140002 !

Further experimentation showed that 2^(J-1)+1 gave

a value shown as 1<1.570001.

Changing the +1 to +10 gave 1=0.570001 and to +100 gave 2<0.570001 !

So, poking a value of 149 into -31961 will cause any number taken to the power of zero to have a value of 220.5727273, which will be represented on screen in some apparently undocumented format - it's not even radix 100. I wonder if the fellows who built this computer could explain that!

ATTENTION all newsletter editors! If you print the above through the Formatter PLEASE transliterate the caret sign!

This one requires the TEII module and the Speech Synthsizer. Want to make the computer so mad it will fuss and fume and cuss and mutter? Run this program and answer the prompt with 1.

```
100 CALL CLEAR
110 OPEN #1:"SPEECH",OUTPUT
120 INPUT X
130 PRINT #1:"//"&STR$(X)&"
"&STR$(X*3.17)
140 PRINT #1:"THIS IS THE SE
CRET METHOD OF MAKING THE CO
MPUTER SPEAK IN A WHISPER"
150 GOTO 120
```

Want to make it whisper to you? Answer the prompt with 0 or -10.

Why did I get an INPUT ERROR when the strings in this routine got too long?

```
100 CALL CLEAR :: X=1
110 X=X*2 :: A$=RPT$("A",X)
: B$=RPT$("B",X):: C$=RPT$("
C",X):: D$=RPT$("D",X):: PRI
NT A$:B$:C$:D$
120 OPEN #1:"DSK1.TEST",VARI
ABLE 254,OUTPUT :: PRINT #1:
A$:B$:C$:D$ :: CLOSE #1
130 OPEN #1:"DSK1.TEST",INPU
T :: INPUT #1:A$,B$,C$,D$ ::
 PRINT A$:B$:C$:D$ :: CLOSE
#1 :: GOTO 110
```

Thanks to Irwin Hott for the answer to that one. I don't think it's in the books anywhere, but the TI won't input multiple records in a single INPUT if the total number of bytes is too high - less than 154 for two records to less than 144 for six records.

I still think computers should be fun, so here is a quickie for the kids, or for the kid in you -

```
100 PRINT TAB(9);"QUICK DRAW
": : : :" How good a gunslin
ger are":"you?": :" Can you
outdraw":"Deadeye Joe?": :
110 PRINT " Watch the countd
own from 1":"to 10.": :" Wai
t for the gun....": :" Then
hit any key FAST!! - ": :" -
 and HOLD IT DOWN": :
120 PRINT " I got down to 20
 once - can":"you beat that?
": :" Press any key to start
"
130 CALL KEY(0,K,ST):: IF ST
=0 THEN 130
140 CALL CLEAR :: S@=300 ::
CALL CHAR(58,"009F9191919191
9F"):: CALL CHAR(42,"0000FCF
E171F0707")
150 CALL KEY(0,K,ST):: IF ST
=-1 THEN 150
160 CALL CLEAR :: FOR M=1 TO
 10 :: CALL HCHAR(12,16,M+48
):: FOR N=1 TO 100
170 NEXT N :: CALL KEY(0,F,X
):: IF F=70 THEN 330
180 NEXT M :: CALL CLEAR ::
FOR J=1 TO 500
190 NEXT J :: IF F=70 THEN 3
30
200 CALL KEY(0,K,ST):: IF ST
<>0 THEN 330
210 CALL HCHAR(12,16,42):: F
OR D=1 TO S@
220 NEXT D :: CALL KEY(0,Z,X
):: IF X=0 THEN 240
230 GOTO 270
240 CALL CLEAR :: PRINT :: P
RINT "YOU'RE DEAD!"
250 FOR D=1 TO 200
260 NEXT D :: GOTO 160
270 PRINT "OUCH!" :: IF S@<5
1 THEN 290
280 S@=S@-50 :: GOTO 320
290 IF S@<31 THEN 310
300 S@=S@-5 :: GOTO 320
310 S@=S@-1
320 PRINT S@ :: GOTO 250
330 PRINT "YOU CHEATED!" ::
```

```
GOTO 150
```

I always wondered about those recipe programs. Does the cook lug the computer out to the kitchen to read the screen, or use a printer to make a hardcopy of a file that was keyed in from a hardcopy in the first place?

Anyway, some of those programs do convert quantities for different servings, so here is a little program to do that. It provides input and output in fractions instead of decimals, because that is the way recipes are written.

```
100 DISPLAY AT(3,6)ERASE ALL
:"RECIPE CONVERTER"
110 DISPLAY AT(6,1):"Enter f
ractional quantities separat
ed by a space from  whole q
uantities."
120 DISPLAY AT(9,1):"For ins
tance, to enter threeand one
-half, type 3 1/2"
130 DISPLAY AT(12,1):"Result
s will be rounded to  the ne
arest 8th."
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
150 DISPLAY AT(12,1)ERASE AL
L:"TURN PRINTER ON!"
160 OPEN #1:"PIO" :: PRINT #
1:CHR$(27);"@" :: CALL CLEAR
170 DISPLAY AT(5,1):"Name of
 recipe?" :: ACCEPT AT(7,1):
M$ :: PRINT #1:M$:"":""
180 DISPLAY AT(3,1)ERASE ALL
:"Recipe is for how many
  servings?" :: ACCEPT AT(4,
11)VALIDATE(DIGIT)BEEP:R
190 DISPLAY AT(6,1):"You wan
t to cook how many  serving
s?" :: ACCEPT AT(7,11)VALIDA
TE(NUMERIC):S :: X=S/R
200 DISPLAY AT(10,1):"Name o
f ingredient? (just  enter
if finished)" :: ACCEPT AT(1
3,1)BEEP:A$ :: IF A$="" THEN
 STOP
210 DISPLAY AT(15,1):"Unit o
f measure?" :: ACCEPT AT(17,
1)BEEP:M$
220 ON ERROR 310 :: DISPLAY
AT(19,1):"Quantity in recipe
?" :: ACCEPT AT(21,1)BEEP:AX
$ :: A=VAL(AX$)
230 Q=X*A :: J=INT(Q):: P=Q-
J :: IF P=0 THEN X$=STR$(J):
: Y$="" :: GOTO 290
240 IF J=0 AND P<=.0625 THEN
 X$="" :: Y$="less than 1/16
" :: GOTO 290 ELSE IF P<=.06
25 THEN X$=STR$(J):: Y$="" :
: GOTO 290
250 IF P>.9375 THEN X$=STR$(
J+1):: Y$="" :: GOTO 290
260 DATA .8125,7/8,.6875,3/4
,.5625,5/8,.4375,1/2,.3125,3
/8,.1875,1/4,.0625,1/8
270 RESTORE 260
280 READ M,N$ :: IF P>M THEN
 Y$=N$ :: X$=STR$(J)ELSE 280
290 IF J<1 THEN X$=""
300 PRINT #1:A$&" "&X$&" "&Y
$&" "&M$ :: GOTO 200
310 P=POS(AX$," ",1):: Q=POS
(AX$,"/",1):: IF Q=0 THEN 34
0
320 ON ERROR 340 :: IF P=0 T
HEN A=0 ELSE A=VAL(SEG$(AX$,
1,P-1))
330 B=VAL(SEG$(AX$,P+1,Q-1-P
)):: C=VAL(SEG$(AX$,Q+1,255)
):: A=A+B/C :: RETURN 230
340 DISPLAY AT(24,1):"OOPS!
TRY AGAIN" :: CALL SOUND(1,1
10,0,-4,0):: RETURN 220
```

And here is an oldie - a utility to get the bugs out of your programs.

```
100 ! MOSQUITO #2 by Jim Pet
erson from a PEEK by Crag Mi
ller
110 CALL CLEAR :: CALL SPRIT
E(#1,42,2,100,100)
115 DISPLAY AT(22,1):"Don't
let the mosquito get":"out o
f the TV!":"Press any key -Q
UICK!"
120 RANDOMIZE :: CALL PEEK(-
31808,A,B):: CALL MOTION(#1,
A-128,B-128):: CALL KEY(0,K,
S):: IF S=0 THEN 120
130 CALL CLEAR :: CALL COLOR
(1,2,8):: CALL SCREEN(2):: C
ALL CHAR(32,"FF888888FF88888
8"):: GOTO 120
```

Long live the TI-99/4A!

Jim Peterson

The Tigercub ■

GRAPHIC2 FÖR ASSEMBLER INITIERAD FÖR GRAFIK

av Jan Alexandersson

I den tidigare artikeln om Graphic2 beskrev jag hur VDP-register användes och hur text kan skrivas på skärmen. Läs mera om detta i PB 92-2.

För att kunna använda grafik så måste du initiera skärmtabellen med byte >00 till >FF i nummerföljd. Detta måste upprepas tre gånger eftersom skärmen är delad i tre delar. Varje teckenruta med 8x8 punkter på skärmen får en byte i SCREEN TABLE som pekar ut var man ska titta efter punkter och färger i PATTERN TABLE och i COLOR TABLE.

Varje sådan teckenruta (8x8 pkt) på skärmen har 8 bytes i PATTERN TABLE och 8 bytes i COLOR TABLE. Varje punktrad i teckenrutan tilldelas en egen byte i respektive tabell:

tecken	0	1	2	3
byte	0	8	16	24
byte	1	9	17	25
byte	2	10	18	26
byte	3	11	19	27
byte	4	12	20	28
byte	5	13	21	29
byte	6	14	22	30
byte	7	15	23	31

I PATTERN TABLE användes de 8 bitarna i respektive byte för att sätta de åtta vågräta punkterna till (=1) eller från (=0).

I COLOR TABLE användes motsvarande byte för förgrundsfärg (de fyra mest signifikanta bitarna) och bakgrundsfärg (de fyra minst signifikanta bitarna). Förgrundsfärgen tilldelas de 1-satta punkterna i PATTERN TABLE medan de 0-satta punkterna får bakgrundsfärgen.

DEMO-programmet visar hur du kan rita på skärmen med hjälp av piltangenterna (ALPHA LOCK nedtryckt):

E, S, D, X	för normala riktningar
W, R, Z, C	för diagonalerna
1	ändra skärmfärg
2	ändra punktfärg
K	rensa skärmen
Q	avsluta (QUIT)

Programmet läser alltid befintlig COLOR BYTE och ändrar endast förgrundsfärgen så att bakgrundsfärgen lämnas oförändrad. Detta har glömts bort i TI-Artist och andra ritprogram som jag har sett. TI-Artist ändrar alltid både för- och bakgrundsfärg. Detta tvingar TI-Artistanvändare att normalt ha samma bakgrundsfärg över hela skärmen (vanligen transparent). Om du ritar med piltangenterna på den randiga startskärmen så kan du se att detta fungerar riktigt i mitt program. Försök att rita motsvarande med TI-Artist (det går inte).

```
**********************************
* Graphic2 DEMO PROGRAM 3 for 99/4A
* by Jan Alexandersson
* 1992-04-17
* some ideas taken from
* Peter M.L.Lottrup: COMPUTE!'s
* Beginner's Guide to
* Assembly Language on the TI-99/4A
* R3 = pixel column
* R4 = pixel row
* R5 = screen colour
* R6 = pixel foreground colour
* R12= pattern byte and colour byte
* R13= bit offset in pattern byte
**********************************

        DEF  START

        REF  VWTR,VSBW,VSBR,VMBW
        REF  KSCAN,GPLLNK,GPLWS

KEY     EQU  >8375

WS      BSS  >20
RTN     DATA 0

        COPY "DSK2.G2-TAB-4AG"
        COPY "DSK2.G24AG"
        COPY "DSK2.DRAW"
        COPY "DSK2.TSTKEY"
        COPY "DSK2.BLKVDP"      PB91-6
        COPY "DSK2.G1-TAB-4A"   PB91-6
        COPY "DSK2.G14A"        PB91-6
```

```
       EVEN

START  MOV  R11,@RTN  save return
       LWPI WS
       BL   @G24AG    graphic2/graph
       BL   @DEFALT

****** make every second column red
       LI   R0,COLTA2
       LI   R1,>0600  red colour
       LI   R2,COLEN2
REDLP  LI   R7,8
REDLP2 BLWP @VSBW
       INC  R0
       DEC  R7
       JNE  REDLP2
       AI   R0,8
       AI   R2,-16
       JNE  REDLP

****** main program loop
LOOP   BL   @DRAW     draw pixel
       LI   R2,6000   drawing speed
DELAY  DEC  R2
       JNE  DELAY
       BLWP @KSCAN    key scan
       CLR  R1
       MOVB @KEY,R1
       SWPB R1
       BL   @TSTKEY   test key
       BL   @CHECK    test limits
       JMP  LOOP

****** load default values
DEFALT LI   R3,256/2  initial column
       LI   R4,192/2  initial row
       LI   R5,>0001  black screen
       LI   R6,>A000  yellow pixel
       RT

****** restore graphic1
QUIT   BL   @G14A
       CLR  R1
       MOVB R1,@STATUS
       LWPI GPLWS
       MOV  @RTN,R11
       RT               return

       END


********************************
* G2-TAB-4AG tables for 99/4A
* in Graphic2 mode with graphics
********************************

PART   EQU  3          3 parts of scr
SCREE2 EQU  6
SCREN2 EQU  24*32
PATTE2 EQU  0
PATEN2 EQU  >800*PART
COLOR2 EQU  >80
COLEN2 EQU  >800*PART
ATTRI2 EQU  >36
SCRCO2 EQU  1          black screen
COLCH2 EQU  >0400      blue background

SCRTA2 EQU  SCREE2*>400
PATTA2 EQU  PATTE2*>800
COLTA2 EQU  COLOR2*>40
ATTAD2 EQU  ATTRI2*>80


**********************
* G24AG initialize
* Graphic2 for 99/4A
* for showing graphics
* by Jan Alexandersson
* 1992-04-18
**********************

*FAC    EQU  >834A      in G14A
*STATUS EQU  >837C      in G14A
*KEYBRD EQU  >8374      in G14A

G24AG  MOV  R11,R10    save return
       LI   R0,>01A0   screen off
       BLWP @VWTR
       LI   R0,>0002   graphic2
       BLWP @VWTR
****** screen table
       LI   R0,>0200+SCREE2
       BLWP @VWTR
****** colour table
       LI   R0,>0300+COLOR2+>7F
       BLWP @VWTR
****** pattern table
       LI   R0,>0400+PATTE2+>03
       BLWP @VWTR
****** sprite attribute table
       LI   R0,>0500+ATTRI2
       BLWP @VWTR
****** screen colour
       LI   R0,>0700+SCRCO2
       BLWP @VWTR
****** standard keybord = 0
       CLR  R0
       MOVB R0,@KEYBRD

****** clear color table
       BL   @BLKVDP
       DATA COLTA2,COLCH2,COLEN2

****** clear pattern table
       BL   @BLKVDP
       DATA PATTA2,0,PATEN2

****** D0 removes standing sprites
       LI   R0,ATTAD2
       LI   R1,>D000
       BLWP @VSBW

****** screen table for graphics
       LI   R2,PART
```

```
       LI   R0,SCRTA2
       CLR  R1
SCRLP  BLWP @VSBW
       INC  R0
       AI   R1,>0100
       JNE  SCRLP
       DEC  R2
       JNE  SCRLP

       LI   R0,>01E0  screen on
       BLWP @VWTR
       SWPB R0
       MOVB R0,@>83D4 for KSCAN

       B    *R10      return


***********************************
* Calculate pattern and colour byte
* See E/A manual page 336
***********************************
DRAW   MOV  R4,R12
       SLA  R12,5
       SOC  R4,R12
       ANDI R12,>FF07
       MOV  R3,R13
       ANDI R13,7
       A    R3,R12
       S    R13,R12
****** set pattern bit
       MOV  R12,R0
       AI   R0,PATTA2
       BLWP @VSBR     read old byte
       LI   R2,>8000  first value
       MOV  R13,R7    bit offset
       JEQ  DRAW2
DRAW1  SRL  R2,1      new bit value
       DEC  R7
       JNE  DRAW1
DRAW2  SOC  R2,R1     add new/old
       BLWP @VSBW
****** set colour bits
       MOV  R12,R0
       AI   R0,COLTA2
       BLWP @VSBR     read old byte
       ANDI R1,>0F00  extract backg
       SOC  R6,R1     add forground
       BLWP @VSBW
       RT

****** screen colour
SC     INC  R5
       ANDI R5,>000F
       MOV  R5,R0
       AI   R0,>0700
       BLWP @VWTR
       LI   R2,20000
SCDEL  DEC  R2
       JNE  SCDEL
       RT

****** pixel colour
LC     AI   R6,>1000
       LI   R2,20000
LCDEL  DEC  R2
       JNE  LCDEL
       RT

****** check screen limits
CHECK  CI   R3,32*8
       JLT  CK1
       LI   R3,32*8-1
CK1    CI   R3,0
       JGT  CK2
       LI   R3,0
CK2    CI   R4,24*8
       JLT  CK3
       LI   R4,24*8-1
CK3    CI   R4,0
       JGT  CK4
       LI   R4,0
CK4    RT


*************************
* Test key
* EXDS normal keys
* RWZC diagonal keys
* 1 screen colour
* 2 pixel colour
* K Clear Screen
* Q Quit
*************************

TSTKEY MOV  R11,R9
       CI   R1,69        E key
       JNE  K88
       DEC  R4
K88    CI   R1,88        X key
       JNE  K68
       INC  R4
K68    CI   R1,68        D key
       JNE  K83
       INC  R3
K83    CI   R1,83        S key
       JNE  K82
       DEC  R3
K82    CI   R1,82        R key
       JNE  K87
       DEC  R4
       INC  R3
K87    CI   R1,87        W key
       JNE  K90
       DEC  R4
       DEC  R3
K90    CI   R1,90        Z key
       JNE  K67
       INC  R4
       DEC  R3
K67    CI   R1,67        C key
       JNE  K75
       INC  R4
       INC  R3
K75    CI   R1,75        K key
```

FAST EXTENDED BASIC!
88/04 - QUICKBILL

NOTE: the program is an UPDATED version; changes were made to add a TAX function, lines 185, 330-340, 415, 460-480, and more IMAGE statements. These docs have been updated accordingly.

PRINT and DISPLAY are two of the most often used statements in XB, and their syntax is easily understood. But wouldn't it be nice to be able to tell the computer exactly how to place the variables on the line, without having to fiddle with the concatenation of strings, or trailing spaces for numeric variables? How about aligning a whole column of text and a whole column of strings?

Like bigger Basics (Microsoft's and GW-BASIC for example), TI's Extended Basic uses a statement called PRINT USING (plus of course DISPLAY USING). In the GW-BASIC manual, there are three pages of different ways to format your line, and you must tell the computer if your value is a string or a number; if a number, you must tell if it is positive or negative when you format the string! Quite complicated... The wise XB guru has provided us with only one way to do it: define the line by using any constant character, and the string or numeric values by replacing them with the maximum number of "#"s that you think you will need. Any character which is not an octothorpe (a.k.a. "#"), including the space and the comma, tells Tex to separate the values. And no need to specify if the value is a string or not, Tex will know and react accordingly; it will even learn and display the proper sign of your numeric value, therefore you can use the same format for about anything. Better, the period will be used for decimals if the value is numeric, or will be "forgotten" if the value is a string!

The PRINT USING statement normally expects the formatting string to follow (in ""s), like in PRINT USING "####":A$. But if you want to use the same format many times, you can use a very unique statement, called "IMAGE". It allows you to define any formatting string on a separate line, without quotes. Great idea isn't it? The only drawback is that if your value is bigger (i.e. has more characters) than the format defined, you get asterisks. Some problem, but not bad if you consider that if the same thing happens to you in GW-BASIC, subsequent formatting can be destroyed.

This short program is a quick review of some of the possibilities; the results at the right are printed exactly as they would appear on the screen:

```
       JNE   K81
       BL    @G24AG
       BL    @DEFALT
K81    CI    R1,81        Q key
       JNE   K49
       B     @QUIT
K49    CI    R1,49        1 key
       JNE   K50
       BL    @SC
K50    CI    R1,50        2 key
       JNE   TSTRTN
       BL    @LC
TSTRTN B     *R9                    ■
```

```
100 IMAGE ###/###.##                    |  R
110 PRINT USING 100:100,3               |  E    100/  3.00
120 PRINT USING 100:5.23,-76.45         |  S      5/-76.45
130 PRINT USING 100:2000,9999           |  U    ***/******
140 PRINT USING 100:"TOT",258.50        |  L    TOT/258.50
150 PRINT USING 100:"ME","THERE"        |  T    ME /THERE
160 PRINT USING 100:"HOUSE","MAILMAN"   |  S    ***/******
```

You will quickly note that the slash (a constant character; any other would have the same effect) separates the two values; numeric values are printed with decimals if there is a period, but only the integer part is printed if there is none. The values which are too big are replaced by asterisks. Another feature, very useful but hard to spot here: the string values are left-justified, the numeric values are right-justified. Great to print lists of words and numbers! How about QUICKBILL, a short program to print a bill?

This crude program (but good things can always be improved) is divided in three modules. The Input Data part is very simple; I decided to use the INPUT and LINPUT (line input, which allows the comma) statements for simplicity. The data entry for the address of the receiver is a straightforward string input, with no error catching; the TAX input is also a simple number variable input (enter "0" if you don't need to calculate the tax). The bill part is a bit more sophisticated, because it is very easy to mix the data, and enter a number when asked for a string. If you enter a number for "Item Name", the program checks if its ASCII value (that of the first character of the string) is lower than 65, i.e. "A". The error checking for the INPUT statements in line 230, which expects numeric values, is controlled by the ON WARNING NEXT in line 110. Without it, your program would crash with a not so nice "* WARNING INPUT ERROR IN 230"... With it, it simply re-executes the statement and will keep doing it until you understand.

Line 240 calculates the total cost for each item in the bill, and tabulates the running total. Please note that no tax is added here: this will be done only later, on the total. You can enter a maximum of 15 items in the bill; if you have less than that, simply press enter at the next "Item Name" prompt.

The second part of the program displays our bill on the screen, by using the DISPLAY AT USING statement. Please note carefully the syntax of each possibility, both for DISPLAY and PRINT, as this is the most complicated part of the statement, and is different when you print to an output device other than the screen. The XB manual is incomplete in that area, and I found the correct syntax after a lot of trial and error.

The IMAGES we will use are conveniently placed at the beginning of each section of the program, but they could be anywhere, since they are referenced by their line number. For the screen output, we use two images: line 270 to display the client's name and address, plus a vertical bar (a constant character), and line 280 to design the bill proper.

Line 290 uses the image in line 270. Since all the octothorpes are continuous, that means we can use only one variable. True? No... as I discovered when I wrote the program, you can re-use the same image for any number of variables; each use of the image will take a new line: easier formatting when you need to mix values and constant characters! When, on the other hand, we format our display to print more than one variable in the same formatted line, everything will fit exactly where you want it: look at line 320, which prints three variables using the image in line 280, and also provides for the dollar sign. In all cases where you format one than more values, they MUST follow the format string or reference to an image, and they MUST be separated by commas.

Since the format in lines 335 and 340 is very short, I decided not to use an external image; in that case, the formatting string must be between quotation marks, but again it can include any constant character, here "TX" and the dollar sign respectively. Please note the CALL KEY in line 360: defining the keyboard as "3" allows us to enter either upper or lower case, Tex will always read it as upper case!

Our bill looks fine on the screen, but how do we send it to our client? Well, we print it, and again the IMAGE statement will help us to

easily format a nice and professional-looking document. Since the printer is wider than the screen, we can include more values in our bill proper, and some formats can be longer.

The Image in line 390 expects four values (see line 450) and also provides the dollar signs plus some vertical bars. As for the longer values, try to bill your client for a few ANTIESTABLISHMENTS, the screen will show asterisks, but the printed bill will be OK; but don't sell him any ARCHEOLOGICAL ARTIFACTS: he will see only stars, and will rightly refuse to pay for them!

Line 275 Image is used to print the total in line 470; instead of including a long line of spaces in the IMAGE statement (in that case, we would need quotations marks to keep those spaces in), we print them in the previous statement, followed by a semi-colon. Why use an image for such a simple case? Well, you guessed it: whatever the value of the TOT variable, it will always be placed at the same spot on the page, and always aligned with the list of prices in the main portion of the bill. The same goes of course for the tax; if you told Tex "no tax", i.e. TAX=0, it will skip line 470 and print only the total.

I hope that you followed me through my picture book, and that you will use the very useful IMAGE statement, or at least DISPLAY USING.

```
100 REM ** QUICKBILL ** L. D
orais/March 1988
110 CALL CLEAR :: LI$=RPT$("
_",28)
120 CALL CHAR(95,"000000FFFF
",124,"1818181818181818")
130 DIM N(15),IT$(15),P(15),
TP(15)
140 GOTO 160 :: K,S,TAX,TOT,
X,AD$,CI$,LP$,NA$,PC$,SP$ ::
 CALL KEY :: !@P-
150 REM == enter data ==
160 ON WARNING NEXT :: PRINT
 "       QUICK BILL MAKER":
: :
170 LINPUT "Bill to: ":NA$ :
: LINPUT "Address: ":AD$
180 LINPUT "City/Pr: ":CI$ :
: LINPUT "P. Code: ":PC$
185 PRINT :: INPUT "TAX (in
%, or 0): ":TAX
190 PRINT :"Now, enter the d
ata for":"each item; enter n
othing":"when you are finish
ed.": :
200 FOR X=1 TO 15
210 PRINT X;:: LINPUT "Item
Name: ":IT$(X)
220 IF IT$(X)="" THEN 290 EL
SE IF ASC(IT$(X))<65 THEN 21
0
230 INPUT "   How many? ":N
(X):: INPUT "   Cost Each? "
:P(X)
240 TP(X)=P*X)*N(X):: TOT=TO
T+TP(X)
250 PRINT :: NEXT X
260 REM == screen display ==
270 IMAGE ################ |
275 IMAGE $#####.##
280 IMAGE ### ##############
# $####.##
290 CALL CLEAR :: DISPLAY AT
(1,1):USING 270:NA$,AD$,CI$,
PC$
300 DISPLAY AT(5,1):LI$
310 FOR X=1 TO 15 :: IF N(X)
=0 THEN 330
320 DISPLAY AT(6+X,1):USING
280:N(X),IT$(X),TP(X):: NEXT
 X
330 DISPLAY AT(1,20):USING 2
75:TOT :: IF TAX=0 THEN 350
335 TAX=TOT*.08 :: TAX=INT(T
AX/.01+.5)*.01 :: DISPLAY AT
(2,20):USING "TX####.##":TAX
340 DISPLAY AT(3,20):RPT$("_
",9):: DISPLAY AT(4,20):USIN
G "$#####.##":TOT+TAX
350 DISPLAY AT(23,1)BEEP:LI$
:"PRESS  <P>rint <R>edo <Q>u
it"
360 CALL KEY(3,K,S):: IF S=0
 OR K<80 OR K>82 THEN 360
370 IF K=81 THEN END ELSE IF
 K=82 THEN RUN ! easy way ou
t...
380 REM == print bill ==
390 IMAGE ###   ############
###### | $####.## | $####.##
400 IMAGE t o t a l :    $##
###.##
405 IMAGE t a x :    $#####.
##
410 OPEN #1:"PIO" :: PRINT #
1:TAB(35);"* B I L L *": : :
415 SP$=RPT$(" ",15):: LP$=R
PT$(" ",52)&"_________"
```

```
420 PRINT #1:RPT$(" ",11)&"T
O: "&NA$:SP$&AD$:SP$&CI$:SP$
&PC$: : : :
430 PRINT #1:SP$&"NUM   ITEM
 NAME                UNIT
TOT. COST":SP$&RPT$("_",46):
""
440 FOR X=1 TO 15 :: IF N(X)
=0 THEN 460
450 PRINT #1:SP$;:: PRINT #1
,USING 390:N(X),IT$(X),P(X),
TP(X):: NEXT X
460 PRINT #1:LP$:"" :: IF TA
X=0 THEN 475
470 PRINT #1:RPT$(" ",52);::
 PRINT #1,USING 275:TOT :: P
RINT #1:RPT$(" ",41);:: PRIN
T #1,USING 405:TAX :: PRINT
#1:LP$
475 PRINT #1:RPT$(" ",37);::
 PRINT #1,USING 400:TOT+TAX
:: PRINT #1:LP$
480 PRINT #1:CHR$(12):: CLOS
E #1 :: GOTO 350
```

PROGRAMMING MUSIC – PART 3

by Jim Peterson, Tigercub, USA

In Part 1 of this series, I showed you the simple routine to set up a musical scale, and showed you how easy it was to merge in various routines to create different effects in single-note music. In Part 2 I showed you how to key in single-note melodies from sheet music. Now, we will get into 3-part harmony.

But first, there are a few more things I should have told you about reading music. You will often see curved lines arching over two or more notes. If the notes are not all the same, ignore those lines - they call for phrasing which you cannot really accomplish. But, if the line curves over two or three of the same note, you will get a better effect if you add all their duration values together and program them as a single note. For instance, if your chart gives a whole note a value of 8 and a half-note a value of 4, and the music has a curved line over a whole note followed by a half-note, just program one note with a duration of 12.

You may find a heavy black bar at the beginning of a measure, with a colon to its right, and somewhere later in the music will be a heavy bar with a colon at its left. This means that the notes between those bars are to be played through twice - and naturally you will want to save time by programming them in a GOSUB as I showed you in Part 2. It can get more complicated than that, but generally you can follow the lyrics to decipher what to do.

Rather rarely, you may find three notes, usually joined together, with a 3 above them. These are called a triplet, and all three of them are to be played, with the same duration for each, in the length of time it would normally take to play one of them. These can create a problem under any method of music programming. The best method is to divide the duration of the note by three and write individual CALL SOUNDs in your music, rather than a GOSUB to a routine, to handle those notes.

Now, let's get on to 3-part harmony. It is just the same as keying in single note music, except that you must also give frequency values to B and C - and, as before, you have to give those values only when they change.

So, load the SCALE routine from the first lesson, and key in this bit of music to experiment with. Notice that I found three repeating phrases and put them in subroutines in 500, 600 and 700 to make this shorter.

```
110 GOSUB 500 :: T=4 :: A=15
 :: B=11 :: C=9 :: GOSUB 100
0 :: T=8 :: A=18 :: GOSUB 10
00 :: T=2 :: A,B,C=0 :: GOSU
B 1000 :: T=2 :: A=23 :: B=1
8 :: C=15 :: GOSUB 1000 :: G
OSUB 600
120 T=2 :: A=21 :: B=18 :: C
=15 :: GOSUB 1000 :: A=23 ::
 GOSUB 1000 :: T=12 :: A=20
:: B=16 :: C=11 :: GOSUB 100
0
130 T=2 :: A,B,C=0 :: GOSUB
```

```
1000 :: GOSUB 500 :: T=4 ::
A=21 :: B=16 :: C=13 :: GOSU
B 1000 :: T=10 :: A=25 :: GO
SUB 1000
140 T=2 :: A=28 :: GOSUB 100
0 :: GOSUB 600
150 T=2 :: A=27 :: B=23 :: C
=18 :: GOSUB 1000 :: A=30 ::
 GOSUB 1000 :: T=10 :: A=28
:: B=23 :: C=20 :: GOSUB 100
0
160 T=2 :: A,B,C=0 :: GOSUB
1000 :: T=3 :: A=28 :: B=23
:: C=20 :: GOSUB 1000 :: T=1
 :: A=27 :: GOSUB 1000 :: GO
SUB 700
170 T=6 :: A=25 :: B=21 :: C
=9 :: GOSUB 1000 :: T=2 :: A
=23 :: B=18 :: C=15 :: GOSUB
 1000
180 T=10 :: A=20 :: B=16 ::
C=11 :: GOSUB 1000 :: T=2 ::
 A,B,C=0 :: GOSUB 1000
190 T=3 :: A=28 :: B=23 :: C
=20 :: GOSUB 1000 :: T=1 ::
A=27 :: GOSUB 1000 :: GOSUB
700
200 T=4 :: A=25 :: B=21 :: C
=16 :: GOSUB 1000 :: A=21 ::
 B=18 :: C=15 :: GOSUB 1000
210 T=14 :: A=20 :: B=16 ::
C=11 :: GOSUB 1000 :: T=2 ::
 A,B,C=0 :: GOSUB 1000 :: ST
OP
500 T=2 :: A=23 :: B=20 :: C
=16 :: GOSUB 1000 :: A=28 ::
 GOSUB 1000 :: A=27 :: GOSUB
 1000 :: A=28 :: GOSUB 1000
:: A=27 :: GOSUB 1000
510 A=28 :: GOSUB 1000 :: A=
23 :: B=20 :: C=16 :: GOSUB
1000 :: A=20 :: B=16 :: C=11
 :: GOSUB 1000 :: A=16 :: B=
11 :: C=8 :: GOSUB 1000 :: R
ETURN
600 T=2 :: A=27 :: B=23 :: C
=18 :: GOSUB 1000 :: A=23 ::
 B=18 :: C=15 :: GOSUB 1000
:: A=21 :: GOSUB 1000 :: A=2
3 :: GOSUB 1000
610 A=27 :: GOSUB 1000 :: A=
23 :: GOSUB 1000 :: RETURN
700 T=4 :: A=27 :: B=21 :: C
=16 :: GOSUB 1000 :: T=8 ::
A=25 :: GOSUB 1000 :: T=3 ::
 A=27 :: B=23 :: C=18 :: GOS
UB 1000
710 T=1 :: A=21 :: GOSUB 100
0 :: T=4 :: A=25 :: B=21 ::
C=16 :: GOSUB 1000 :: T=8 ::
 A=23 :: B=20 :: C=16 :: GOS
UB 1000
720 T=3 :: A=25 :: B=21 :: C
=16 :: GOSUB 1000 :: T=1 ::
A=23 :: GOSUB 1000 :: T=2 ::
 A=23 :: B=18 :: C=15 :: GOS
UB 1000
730 A=21 :: GOSUB 1000 :: A=
20 :: GOSUB 1000 :: A=21 ::
GOSUB 1000 :: RETURN
```

Save that under the filename ROSES, clear the memory with NEW, and key this in -

```
1000 CALL SOUND(D*T,N(A),V1,
N(B),V2,N(C),V3):: RETURN
```

Save that by SAVE DSK1.PLAIN3,MERGE.

Load ROSES again and merge it in by

MERGE DSK1.PLAIN3 . Add a line -
105 D=200 and RUN it.

Sounds rather raw and harsh, doesn't it? Try changing that line 105 to -
105 D=200 :: V2=5 :: V3=8

Try it again. Sound better? The first time, all 3 voices were being played at the loudest volume. Usually computer music will sound better if the harmony notes are given a lower volume. Experiment and find the volumes you like best. Is the music too slow for you? Just change the value of D. Is it not in your singing key? Just change the value of F in line 100, as I showed you before.

But, does the music still have too strong a beat for your taste? Clear the memory again and key this in -

```
1000 CALL SOUND(-4250,N(A+Z)
,V1,N(B+Z),V2,N(C+Z),V3):: G
OSUB 1010 :: RETURN
1010 FOR W=1 TO T*D :: NEXT
W :: RETURN
```

Save that as NEG3,MERGE because it uses negative duration for 3 voices. Then load ROSES again and merge it in. This time, try line 105 with D=50 and with V2 and V3 as you wish. Sound smoother?

In lines 110, 130, 160, 180 and 210 of ROSES, you will find A,B,C=0 . That makes all three voices silent, because in line 100 N(0) is given a frequency of 40000 which is above

the range of human hearing. This is how I programmed those silent pauses, the "rests" which were written in the music.

On a piano or guitar, the strings continue to vibrate during a rest, so that the sound gradually fades out. However, the electronically generated tones of a computer stop very suddenly. That is why I often add the duration of the rest to the duration of the preceding note, and play it right on through. Some people think that doesn't sound right, so here is another solution. Clear memory again and key this in -

```
2000 FOR W=2 TO 8 STEP 8 ::
CALL SOUND(-999,N(A+Z),V1+W,
N(B+Z),V2+W,N(C+Z),V3+W):: G
OSUB 2010 :: NEXT W :: RETUR
N
2010 FOR Y=1 TO T*D/4 :: NEX
T Y :: RETURN
```

Save that as REST,MERGE. Load ROSES again, merge in SCALE and NEG3 (this will not work well with PLAIN3) and merge in REST. Now go to lines 110, 130, 160, 180 and 210, delete the A,B,C=0 :: and change the GOSUB 1000 after it to GOSUB 2000. Add line 105, run it and see if you like that better.

Anyway, keep it for now because we will use it again.

You will probably want to have the music play through more than once. Just add :: FOR J=1 TO 4 to the end of line 105 (if you want it to play 4 times) and change the end of line 210 to read NEXT J :: STOP .

I said that you could change the key of the music just by changing the value of F in line 100. There is also a way to change it while the music is playing. After the FOR J=1 TO 4 in 105 put ::

```
Z=Z-(J=2)*3-(J=3)*1+(J=4)*4
```

That is somewhat complicated but it just means to play the second time three whole keys higher, the third time one key higher still (I know the *1 is unnecessary!) and drop back 4 keys for the 4th time, so you can take it from there and modify it as you wish. If you want to use that routine with silent rests, change the GOSUB after each rest to 3000 instead of 1000, and add this line -

```
3000 CALL SOUND(-4250,N(A),V
1,N(B),V2,N(C),V3):: GOSUB 1
010 :: RETURN
```

This tune happens to end in a rest, which is unusual. If you key in another tune and it seems to end too abruptly, just after that NEXT J and before the STOP, put in a long duration such as T=12 and a GOSUB 2000 to that REST routine to fade out more slowly.

Now, when you are keying in your own tunes, the notes on your sheet music will usually have two or three of those little eggs on the stem. It is best to use the upper one for A, the next one for B, and the lower one for C; the computer could care less, but you will find it easier to keep track of what you are doing. If there are less than three, just go directly below to the bass clef and find a note there. If you still don't have enough, you can always use 0 to make that voice silent. Or, you can usually just let the previous note continue. If your sheet music has guitar chords - those little square grids with dots on them - above the staff, they will give you some help - if there is no guitar chord above the note you are working on, the chord has not changed and it is safe to use the previous harmony notes.

There are many other CALL SOUND routines you can use for different effects. This is similar to the one that Bill Knecht used for his hymns - I call it VIBRA.

```
105 D=1 :: V1=1 :: V2=5 :: V
3=11
1000 FOR J=1 TO T*D :: CALL
SOUND(-99,N(A),V1,N(B),V2,N(
C),V3):: CALL SOUND(-99,N(A)
*1.01,V1,N(B),V2,N(C),V3)::
NEXT J :: RETURN
```

This one I call WUBBA, for no good reason -

```
105 D=1 :: V1=1 :: V2=5 :: V
3=11
```

```
1000 FOR J=1 TO T*D :: CALL
SOUND(-99,N(A),V1,N(B),V2,N(
C),V3):: CALL SOUND(-99,N(A)
*1.01,V1,N(B),V3,N(C),V2)::
NEXT J :: RETURN
```

And this one I call TREM -

```
105 D=1 :: V1=1 :: V2=5 :: V
3=11
1000 FOR J=1 TO T*D :: CALL
SOUND(-999,N(A),V2,N(B),V2,N
(C)*1.01,V3):: CALL SOUND(-9
99,N(A)*1.01,V1,N(B),V2,N(C)
,V3):: NEXT J :: RETURN
```

I included line 105 in those, to merge in the duration and volumes along with the sound routine. Change the value of D to suit yourself, even in decimal increments such as D=1.5 .

It is easy to play a song repeatedly but with a different effect each time. Merge in VIBRA and change its line number to 1010. You can do this by typing 1000 and FCTN X, Enter, FCTN 8 to bring it back, type over the line number, and Enter. Merge in WUBBA and change it to line 1020 in the same way, then TREM and change it to line 1030.

Add :: FOR R=1 TO 3 to the end of line 105. Put in a new line 1000 -

```
1000 ON R GOSUB 1010,1020,10
30 :: RETURN
```

And change the end of line 210 to NEXT R :: STOP.

Next time - more different effects, and autochording. ■

MULTIPLAN SYLK FILES – RÄTTELSE

Programmet i PB 90-6 sid 14 saknar följande rader på slutet:

```
480 PRINT #2:T$&RPT$(CHR$(0),128-
    LEN(T $))
490 CLOSE #2
500 END
510 SUB WRITE(T$,T1$)
520 PRINT #2:SEG$(T$,1,128)
530 T1$=SEG$(T$,129,LEN(T$)-128)
540 SUBEND
```
■

DAYS BETWEEN DATES

by Jim Peterson, Tigercub, USA

```
100 DISPLAY AT(2,5)ERASE ALL
:"DAYS BETWEEN DATES":"":"
including leap year days" ::
 M$(1)="From" :: M$(2)="To
" :: R=13
110 DATA 31,28,31,30,31,30,3
1,31,30,31,30,31
120 DIM L(12):: FOR J=1 TO 1
2 :: READ L(J):: NEXT J
130 FOR J=1 TO 2 :: DISPLAY
AT(R-1,1):M$(J):"year      m
onth    day  " :: ACCEPT AT(
R,6)VALIDATE(DIGIT)SIZE(4):Y
(J)
140 ACCEPT AT(R,17)VALIDATE(
DIGIT)SIZE(2):M(J):: IF M(J)
<1 OR M(J)>12 THEN 140
150 ACCEPT AT(R,24)VALIDATE(
DIGIT)SIZE(2):D(J):: IF D(J)
<1 OR D(J)>31 THEN 150
160 CALL LEAP(Y(J),X):: L(2)
=L(2)-X :: IF D(J)>L(M(J))TH
EN 150
170 L(2)=28 :: R=R+3 :: NEXT
 J :: R=13 :: IF Y(1)>Y(2)TH
EN T=Y(1):: Y(1)=Y(2):: Y(2)
=T :: T=M(1):: M(1)=M(2):: M
(2)=T :: T=D(1):: D(1)=D(2):
: D(2)=T
180 IF Y(1)=Y(2)AND M(1)>M(2
)THEN T=M(1):: M(1)=M(2):: M
(2)=T :: T=D(1):: D(1)=D(2):
: D(2)=T
190 L(2)=28 :: IF Y(2)>Y(1)T
HEN 220
200 IF M(1)=M(2)THEN B=ABS(D
(2)-D(1)):: GOTO 260
210 CALL LEAP(Y(1),X):: FOR
J=M(1)+1 TO M(2)-1 :: B=B+L(
J)+X*(M(1)=2):: NEXT J :: B=
B+L(M(1))+X*(M(1)=2)-D(1)+D(
2):: GOTO 260
220 CALL LEAP(Y(1),X):: B=L(
M(1))-D(1)+X*(M(1)=2)
230 FOR J=M(1)+1 TO 12 :: B=
B+L(J)+X*(J=2):: NEXT J
240 FOR J=Y(1)+1 TO Y(2)-1 :
: CALL LEAP(J,X):: B=B+365-X
 :: NEXT J
250 FOR J=1 TO M(2)-1 :: CAL
L LEAP(Y(2),X):: B=B+L(J)+X*
(J=2):: NEXT J :: B=B+D(2)
260 DISPLAY AT(20,1):B;"days
 between" :: B=0 :: GOTO 130
270 SUB LEAP(Y,X):: X=(Y/400
=INT(Y/400)):: IF X=-1 THEN
SUBEXIT ELSE X=(Y/4=INT(Y/4)
):: IF X=0 THEN SUBEXIT ELSE
 X=(Y/100<>INT(Y/100))
280 SUBEND
```
■

MAILING LIST - BASIC

```
1 REM MAILING LIST
2 REM by Doug Hapeman
3 REM COMPUTE 83-07 CS/DS
4 REM REV 2-10,15-19,57,89,4
29,447,497,515
5 CALL CHAR(93,"00382838447C
4444")
6 CALL CHAR(91,"00280038447C
4444")
7 CALL CHAR(92,"0028007C4444
447C")
8 OPTION BASE 1
9 DIM LN$(45),NA$(45),CH$(45
),AD$(45),CP$(45),PC$(45),TP
$(45)
10 CALL CLEAR
11 PRINT " *   99/4A MAILING
 LIST   *": : : : : : : : :
: :"    PRESS ENTER TO BEGIN
"
13 GOSUB 550
21 G$="       PLEASE WAIT...
      WHILE THE PRINTER IS
WORKING"
23 REM MENU
25 CALL CLEAR
27 PRINT "         MAIN INDEX
": : : :
29 PRINT "PRESS   TO": : :
31 PRINT "  1  =  VIEW NAMES
 LIST":"  2  =  SEARCH FOR A
 NAME":"  3  =  ADD NAMES":"
  4  =  CHANGE NAMES"
33 PRINT "  5  =  DELETE NAM
ES":"  6  =  ALPHABETIZIZE L
IST":"  7  =  SAVE DATA FILE
":"  8  =  LOAD DATA FILE"
35 PRINT "  9  =  PRINT LABE
LS/LIST":"  0  =  FINISH SES
SION": : : :
37 GOSUB 550
39 IF (KEY<48)+(KEY>57)THEN
37
43 CALL CLEAR
45 ON KEY-47 GOSUB 525,51,77
,113,185,289,335,427,445,475
47 GOTO 25
49 REM VIEW NAMES LIST
51 T=0
53 FOR I=1 TO N
55 T=T+1
57 PRINT :NA$(I);" ";LN$(I):
CH$(I):AD$(I):PC$(I);" ";CP$
(I):"TEL. ";TP$(I): : :
59 IF T<2 THEN 69
61 PRINT " *PRESS ENTER TO C
ONTINUE*":" *""R"",ENTER FOR
 MAIN INDEX*"
63 GOSUB 550
65 IF KEY=82 THEN 73
67 T=0
69 NEXT I
70 PRINT "       *END OF FIL
E*         *PRESS ENTER TO C
ONTINUE*"
71 GOSUB 550
73 RETURN
75 REM SEARCH NAMES
77 INPUT "LAST NAME? ":Y$
79 FOR I=1 TO N
81 IF LN$(I)<>Y$ THEN 103
83 PRINT : : :"  IS THE PERS
ON:": :"  ";NA$(I):"  ";LN$(
I): :"  (Y/N)?"
85 GOSUB 550
87 IF KEY=78 THEN 103
89 PRINT : : :NA$(I);" ";LN$
(I):CH$(I):AD$(I):PC$(I);" "
;CP$(I):"TEL. ";TP$(I): : :"
    DO YOU WISH TO PRINT
A MAILING LABEL? (Y/N)"
91 GOSUB 550
93 IF KEY<>89 THEN 97
95 GOSUB 495
97 PRINT "SEARCH MORE NAMES?
  (Y/N)"
98 GOSUB 550
99 IF KEY=89 THEN 77
101 GOTO 109
103 NEXT I
105 PRINT : : :"  THE ";Y$:"
  YOU ARE SEARCHING FOR":"
IS NOT IN THIS FILE.": : :
107 GOTO 97
109 RETURN
111 REM ADD NAMES
113 A=N+1
115 FOR I=A TO 45
117 CALL CLEAR
119 PRINT : : : :"ENTER DATA
: #";I;"  (MAX:45)": : :
121 PRINT "  *LAST NAME:"
123 INPUT LN$(I)
125 PRINT :"  *FIRST NAME(S)
:"
127 INPUT NA$(I)
129 PRINT :"  *CHILDREN:":"
  NOTE--DO NOT USE COMMAS!"
131 INPUT CH$(I)
133 PRINT :"  *STREET ADRESS
:"
135 INPUT AD$(I)
137 PRINT :"  *CITY/PROVINCE
:":"   NOTE--DO NOT USE COMM
AS!"
139 INPUT CP$(I)
141 PRINT :"  *POSTAL CODE:"
143 INPUT PC$(I)
```

```
145 PRINT :"  *PHONE:"
147 INPUT TP$(I)
149 V=I
151 REM VERIFY ENTRIES
153 CALL CLEAR
155 PRINT "ENTRY #";V: : :
157 PRINT "YOU ENTERED:": :"
  ";LN$(V);", ";NA$(V):"  ";
CH$(V):"  ";AD$(V):"  ";PC$(
V);
159 PRINT " ";CP$(V):"  TEL.
: ";TP$(V): : : : : : :
161 PRINT "CHANGE ANYTHING?
 (Y/N)"
162 GOSUB 550
163 IF KEY<>89 THEN 171
165 C=N+1
167 CALL CLEAR
169 GOSUB 201
171 PRINT "ADD MORE NAMES?
(Y/N)"
172 GOSUB 550
173 N=N+1
175 IF KEY=78 THEN 181
177 NEXT I
179 PRINT "    *DATA FILE IS
 FULL*      *PRESS ENTER TO
CONTINUE*"
180 GOSUB 550
181 RETURN
183 REM CHANGE DATA
185 PRINT "  LAST NAME OF TH
E PERSON   WHOSE DATA IS TO
BE CHANGED:": : : :
187 INPUT C$
189 CALL CLEAR
191 FOR C=1 TO N+1
193 IF LN$(C)=C$ THEN 195 EL
SE 239
195 PRINT "IS THE PERSON:":"
  ";NA$(C);"  ";LN$(C): :
197 PRINT "  (Y/N)?"
198 GOSUB 550
199 IF KEY=89 THEN 201 ELSE
239
201 PRINT : : : : : :"PRES
S   TO CHANGE": :
203 PRINT "  1  =  LAST NAME
":"  2  =  FIRST NAME(S)":"
 3  =  CHILDREN":"  4  =  ST
REET ADRESS"
205 R=C
207 R$="  *ENTER THE NEW DAT
A:"
209 PRINT "  5  =  CITY/PROV
INCE":"  6  =  POSTAL CODE":
"  7  =  PHONE":"  8  =  NO
CHANGE": : : : : : :
211 GOSUB 550
213 CALL CLEAR
215 IF (KEY<49)+(KEY>56)THEN
 211
219 IF KEY=56 THEN 229
221 ON KEY-48 GOSUB 245,251,
257,263,269,275,281
223 PRINT : :"MORE CHANGES F
OR:":"  ";NA$(R):"  ";LN$(R)
: :
225 PRINT "  (Y/N)?"
226 GOSUB 550
227 IF KEY<>78 THEN 201
229 PRINT : : :"CHANGE DATA
FOR OTHER NAMES?": : :
231 PRINT "  (Y/N)":Z$
232 GOSUB 550
233 CALL CLEAR
235 IF KEY<>78 THEN 185
237 RETURN
239 NEXT C
241 RETURN
243 REM  CHANGE LOOPS
245 PRINT "LAST NAME WAS:":
:LN$(R): : :R$
247 INPUT LN$(R)
249 RETURN
251 PRINT "FIRST NAME(S) WER
E:": :NA$(R): : :R$
253 INPUT NA$(R)
255 RETURN
257 PRINT "CHILDREN WERE:":
:CH$(R): : :R$
259 INPUT CH$(R)
261 RETURN
263 PRINT "ADRESS WAS:": :AD
$(R): : :R$
265 INPUT AD$(R)
267 RETURN
269 PRINT "CITY/PROVINCE WAS
:": :CP$(R): : :R$
271 INPUT CP$(R)
273 RETURN
275 PRINT "POSTAL CODE WAS:"
: :PC$(R): : :R$
277 INPUT PC$(R)
279 RETURN
281 PRINT "PHONE NUMBER WAS:
": :TP$(R): : :R$
283 INPUT TP$(R)
285 RETURN
287 REM DELETE NAMES
289 INPUT "LAST NAME? ":X$
291 FOR I=1 TO N
293 IF LN$(I)<>X$ THEN 325
295 PRINT : : :"IS THE PERSO
N:":"  ";NA$(I):"  ";LN$(I):
 :
297 PRINT "  (Y/N)?"
298 GOSUB 550
299 IF KEY<>89 THEN 325
301 A=I
303 FOR D=A TO N-1
305 LN$(D)=LN$(D+1)
307 NA$(D)=NA$(D+1)
309 CH$(D)=CH$(D+1)
```

```
311 AD$(D)=AD$(D+1)
313 CP$(D)=CP$(D+1)
315 PC$(D)=PC$(D+1)
317 TP$(D)=TP$(D+1)
319 NEXT D
321 N=N-1
323 GOTO 327
325 NEXT I
327 PRINT "MORE DELETIONS? (
Y/N)"
328 GOSUB 550
329 IF KEY=89 THEN 289
331 RETURN
333 REM ALPHABETIZE
335 PRINT "        PLEASE WAI
T...": : :" THE LIST IS BEIN
G ARRANGED": : : : : : : : :
 :
337 B=1
339 B=2*B
341 IF B<=N THEN 339
343 B=INT(B/2)
345 IF B=0 THEN 369
347 FOR Y=1 TO N-B
348 X=Y
349 I=X+B
351 IF LN$(X)=LN$(I)THEN 363
353 IF LN$(X)<LN$(I)THEN 365
355 GOSUB 381
357 X=X-B
359 IF X>0 THEN 349
361 GOTO 365
363 GOSUB 373
365 NEXT Y
367 GOTO 343
369 RETURN
371 REM ORDER FIRST NAMES
373 IF NA$(X)<NA$(I)THEN 377
375 GOSUB 381
377 RETURN
379 REM CHANGE ORDER
381 N$=LN$(X)
383 LN$(X)=LN$(I)
385 LN$(I)=N$
387 N$=NA$(X)
389 NA$(X)=NA$(I)
391 NA$(I)=N$
393 N$=CH$(X)
395 CH$(X)=CH$(I)
397 CH$(I)=N$
399 N$=AD$(X)
401 AD$(X)=AD$(I)
403 AD$(I)=N$
405 N$=CP$(X)
407 CP$(X)=CP$(I)
409 CP$(I)=N$
411 N$=PC$(X)
413 PC$(X)=PC$(I)
415 PC$(I)=N$
417 N$=TP$(X)
419 TP$(X)=TP$(I)
421 TP$(I)=N$
423 RETURN
425 REM SAVE DATA FILE
427 GOSUB 467
429 OPEN #1:L$,INTERNAL,OUTP
UT,FIXED 128
431 PRINT #1:N
433 FOR I=1 TO N
435 PRINT #1:LN$(I),NA$(I),C
H$(I),AD$(I),CP$(I),PC$(I),T
P$(I)
437 NEXT I
439 CLOSE #1
441 RETURN
443 REM LOAD DATA FILE
445 GOSUB 467
447 OPEN #1:L$,INTERNAL,INPU
T ,FIXED 128
449 INPUT #1:N
451 FOR I=1 TO N
453 INPUT #1:LN$(I),NA$(I),C
H$(I),AD$(I),CP$(I),PC$(I),T
P$(I)
455 NEXT I
457 CLOSE #1
459 CALL CLEAR
461 PRINT "  ";L$: :"  THIS
FILE HAS";N;"ENTRIES.": :"
*45 ENTRIES IS MAXIMUM*": :
: : : : : : : :"*PRESS ENTER
 TO CONTINUE*"
463 GOSUB 550
465 RETURN
467 PRINT "     WHAT IS THE
NAME OF":"    YOUR STORAGE D
EVICE?": :"(EXAMPLE: CS1 OR
DSK1.FILE)": : : : : : : : :
 :
469 INPUT L$
471 RETURN
473 REM PRINT LABELS/LIST
475 PRINT "PRESS   TO PRINT"
: : :"  1     MAILING LABELS
": :"  2     MALING LIST": :
 : : : : : :
477 GOSUB 550
479 IF (KEY<49)+(KEY>50)THEN
 477
483 PRINT : : : : : : : : :
: : :G$: : : : : : : : : : :
485 IF KEY=50 THEN 505
487 FOR I=1 TO N
489 GOSUB 495
491 NEXT I
493 RETURN
495 OPEN #2:"PIO"
497 PRINT #2:TAB(5);NA$(I);"
 ";LN$(I):TAB(5);AD$(I):TAB(
5);PC$(I);" ";CP$(I): : :
499 CLOSE #2
501 RETURN
503 REM PRINT MAIL LIST
505 FOR I=1 TO N
```

```
507 GOSUB 513
509 NEXT I
511 RETURN
513 OPEN #2:"PIO"
515 PRINT #2:TAB(5);LN$(I);"
, ";NA$(I);"       ";CH$(I):T
AB(5);AD$(I);"    ";PC$(I);"
";CP$(I);
517 PRINT #2:TAB(58);"TEL. "
;TP$(I): :
519 CLOSE #2
521 RETURN
523 REM FINISH SESSION
525 PRINT "       DO YOU WIS
H TO          TERMINATE THIS
 SESSION?     (Y/N)"
526 GOSUB 550
527 CALL CLEAR
529 IF KEY<>89 THEN 25
531 PRINT "      HAVE A NICE
 DAY!": : : : : : : : :
533 END
540 REM KEY LOOP
550 CALL KEY(3,KEY,STA)
560 IF STA<1 THEN 550
570 RETURN                    ■
```

MERA OM P-GRAM PLUS KORTET

av Jan Alexandersson

Jag har nu själv skaffat ett P-GRAM+ med klocka från Bud Mills Services. Det kostade 230 $ + porto 5 $. Det extra portot till Europa är verkligen 5 dollar eftersom kortet sänds som "SMALL PACKET". Innan du fortsätter att läsa, bör du först läsa recensionen av Charles Good i PB 92-2 (ursprungligen från BITS,BYTES& PIXELS January 1992). Observera att priset har sänkts sedan den artikeln skrevs. Jag kommer i fortsättningen endast att beskriva saker som inte har berörts i den tidigare artikeln.

P-GRAM+ kommer med en Operating Guide på 37 sidor och en Construction Guide på 13 sidor. Du får även fem flexskivor: PG+DSR+, P-G_UTILS, PG+SOURCE, GUMS V1 och MODULES.

P-GRAM+ kan inte användas tillsammans med "Quality Improvement" konsoler som TI tog fram för att hindra användning av andra moduler än TI:s egna. Alla konsoler som är silver- och svart-färgade med copyright 1981 vid färgbalkarna kan användas.

P-GRAM+ är ett kort som placeras i expansionsboxen. Man kan välja 7 olika CRU-adresser: >1000, >1200 - >1700. I normalfallet rekommenderas CRU >1700 eftersom man inte vinner något på ett lågt nummer. Kortet har 192 kbytes RAM uppdelat på 16 kbytes DSR-RAM, 16 kbytes modul-RAM och 4x40 kbytes GRAM. Det finns ett 1 Mbit(128 kbytes) statiskt RAM (NEC D431000ACZ-85L) samt två stycken 256 kbit(32 kbytes) statiska RAM. Ett litium-batteri bevarar minnet när boxen stängs av.

Även om P-GRAM+ är aktiverad så kan du stoppa in en modul i TI-99/4A:s modulport. Kortet kommer då automatiskt att stängas av så att endast modulen kan användes. Genom att skriva CALL PG från BASIC kan du spara den istoppade modulen till flexskiva. Du kan välja att spara GROM-bankarna med 34 eller 26 sektorer. P-GRAM+ använder normalt samma filformat och filnamn som GRAM Kracker (FILNAMN, FILNAMN1, FILNAMN2 o.s.v.). Om du anger ett filnamn på 10 tecken så kommer P-GRAM+ att stega fram sista tecknet (MYCARTFILE, MYCARTFILF, MYCARTFILG o.s.v.) medan GRAM Kracker ersätter det med en etta (MYCARTFILE, MYCARTFIL1, MYCARTFIL2 o.s.v.). P-GRAM+ kan inte ladda in GRAM Krackers filnamn med 10 tecken men kortare filnamn fungerar bra. Du kan alltid ändra filnamn med en disk manager så att det fungerar.

All minneshantering kontrolleras av CRU-bitar så man slipper alla manuella omkopplare som finns i GRAM Kracker. Dessa CRU-bitar är unika för just P-GRAM+ och kan inte användas i andra GRAM utrustningar. Genom att sätta dessa CRU-bitar kan man:

- aktivera DSR
- aktivera GRAM/RAM
- skrivskydda P-GRAM
- aktivera bankswitch av RAM >6000
- aktivera låga delen av RAM >6000
- aktivera höga delen av RAM >6000

EDITOR/ASSEMBLER MODULEN

Det är möjligt att använda modul-RAM tillsammans med Editor/Assembler genom att sätta en speciell flagga: g>6003 ändras till >A5.

Ändring av färgerna i EA-modulen är ej komplett beskrivna i manualen till P-GRAM+. Du måste ändra färgen på tre olika ställen:

- g>652C ger bokstävernas färger, för- och bakgrund i Graphic1.

- g>6537 skärmfärg en kort stund före huvudmenyn och för Editorn.

- g>6B3D skärmfärg för huvudmenyn

Dessa tre adresser har värdet >F5 från början. Jag ändrade till >F4 för att få vit text på mörkblå bakgrund.

Du kan även ändra default-färgerna för LOAD & RUN av assembler-program:

- g>653C ger bokstävernas färger, för- och bakgrund i Graphic1 (normalt >13, svart på ljusgrön).

- g>6547 ger skärmfärgen (normalt >F3 ljusgrön skärm).

ANDRA MODULER MED PROBLEM

Multiplan kan normalt endast användas på GRAM-bank 1. Filen MPINTR innehåller absoluta adresser till GROM. Det medföljer ett patch-program som gör att Multiplan även kan köras på GRAM-bank 2-4, men efter patchningen kan det endast köras från den GRAM-bank som valdes vid din patchning.

Personal Record Keeping och Statistics kan endast köras från GRAM-bank 1. Personal Report Generator visas inte på Review Module Library men den visas av GMENU och kan då köras utan problem från GRAM-bank 1-4. Denna modul använder GROM 3 och 4 men har underligt nog sitt huvud på GROM 4.

MEDFÖLJANDE PROGRAM

GMENU av R.A. Green är ett meny-program som visar alla GRAM-bankarna samtidigt så att man slipper stega genom dessa. Det startas automatiskt vid reset av datorn. Det fungerar bra från alla GRAM-bankar och inte bara bank 1 som det står i manualen.

GRAM Packer medföljer INTE men däremot en P-PACKER fil som frigör P-GRAM+ skrivskydd vilken måste köras före GRAM Packer. DM1000 finns med som ett exempel på en packad fil till GRAM 3 och 4. Det finns även en modifierad CALL-fil eftersom den ursprungliga inte fungerar med RAM-disk. Bud Mills Services säljer GRAM Packer separat för 10 dollar.

BOOT finns med på en av skivorna. Den visar hela tiden klockan på skärmen. Den fungerar ej med G för att se flera GRAM-bankar. Min dator får en felaktig skärm med vita streck. Jag misstänker att problemet beror på mitt DIJIT AVPC 80-kolumnskort eftersom jag tidigare har haft problem även utan P-GRAM+.

Andra program som finns med är Gram User Menu System (GUMS) och Debugging Aid (RAGDBAE) som är fairware från R.A.Green.

MULTIMOD finns även med och består av Disk Manager III (DM2 med DSK1-DSK9 men endast på CRU >1000 och >1100), Editor/Assembler och TI-Writer. De två senare saknar de filer som ska laddas från flexskiva.

MYARC HFDC

Jag har sparat alla filer på min hårddisk under subdirectory WDS1.GRAM. Myarc MDM5 till HFDC fungerar ej tillsammans med P-GRAM+. Det följer med en patch till DM V version 1.27 men den kan inte användes till någon av de andra versioner som jag har. Finns det någon som har patchar till andra versioner t.ex. 1.30? Jag kan dock ladda MDM5 på följande sätt:

- SHIFT/CTRL vid reset startar 80-kolumnskortet i 99/4A mode.

- Välj BASIC

- CALL PGOF stänger av P-GRAM+

- CALL FW för att ladda Funnelweb från Horizon RAM-disk

- ladda WDS1.MYARC.MDM5

P-GRAM+ klocka går på batteri så den fungerar även när expansionsboxen är avstängd. Myarc HFDC har en klocka som saknar batteri så den måste sättas varje gång du startar boxen. Denna klocka används bl.a. för att markera när en fil skapades eller ändrades. Det enklaste är att köra följande lilla BASIC-program när du startar datorn. Programmet läser av P-GRAM+ klocka och skriver detta till HFDC-klockan.

```
100 REM P-GRAM CLOCK TO HFDC
110 REM Jan Alexandersson
120 REM 1992-07-05
130 REM READ P-GRAM CLOCK
140 OPEN #1:"CLOCK"
150 INPUT #1:A$,DATE$,TIME$
160 CLOSE #1
170 MONTH$=SEG$(DATE$,1,2)
180 DAY$=SEG$(DATE$,4,2)
190 YEAR$=SEG$(DATE$,7,2)
200 HOUR$=SEG$(TIME$,1,2)
210 MIN$=SEG$(TIME$,4,2)
220 SEC$=SEG$(TIME$,7,2)
230 REM WRITE HFDC TIME
240 OPEN #2:"TIME",INTERNAL,
FIXED
250 PRINT #2:SEC$,MIN$,HOUR$
,DAY$,MONTH$,YEAR$
260 CLOSE #2
```

HÅRDVARUMODIFIERING

Det finns en ny modifiering av P-GRAM och RAM-DISK från juni 1992. P-GRAM Change #1 (Horizon RAMDISK Change #4) innebär att en diod (1N914) monteras vid en av lysdiod-erna. Kontakta redaktören om du behöver en beskrivning av detta (även andra modifieringar av RAMDISK finns). ■

OPA TIM (V9958) AND SOB

TIM TI-IMAGE-MAKER

TI-IMAGE-MAKER is an internal console expansion board designed to be installed in your TI99/4A with a few simple tools.

TIM upgrades your current TMS9918 video display processor to the latest state-of-the-art and fully compatible 9958 processor. This processor is also compatible with the V9938 used in other upgrades.

OPA used state-of-the-art CAD / CAE designing and top design engineers to bring you the best video upgrade for the TI in the smallest most compatible package possible.

The PCBoard containing the V9958, 192K of VRAM, a special ASIC device, a 25-pin monitor/expansion port for future video devices like digitizers, GENlock, etc. and the analog RGB video driver circuitry, has all been installed on a compacted 4"*3" layout.

TIM comes complete with:

* Step-By-Step User Installation Guide.

* Recommended Analog RGB monitors guide, with detailed pinouts for many different monitors to ease in interfacing your TIM to a RGB monitor. Cable wiring schematics are given in the TIM's manual.

* Two disks packed with updates of the fairware programs which can support V9938/58. Only updates are included as the full packages could well consume over 15 disks. Most the fairware packages are available from BBS's or User Groups.

* Son-of-Board (GROM 0 and 1).

U-558: $179 Canadian or U.S. for TIM

SON OF A BOARD (SOB)

What is SOB? It stands for Son Of A Board, which is an internal plug-in 2" by 2" PCBoard with a 16K EPROM and two ASICs, allowing you to replace GROMs 0 and 1 (the main TI

operating system) in your TI-99/4A console with OPA's new start-up system, allowing ease-of-use of the Review Module Library function plus cataloging and loading of most programs without the need of a module.

For many years, TIers have noticed "REVIEW MODULE LIBRARY" accidentally appearing on module menu screens, with no idea what it was for. SOB makes this feature usable with OPA's knowledge on TEXAS INSTRUMENTS original plans for a GROM LIBRARY PERIPHERAL.

How does SOB work? .... The Basic way is via TI's own GROM 0 code as SOB uses the "Review Module Library" feature built into all TI-99/4A consoles. OPA has designed the SOB - Son Of Board hardware combined with software features that allows the ease of use of the RML function by allowing easy module selection via a scrolling pop-up window, and which also allows cataloging and loading Assembly, Object files, Forth, c99 programs via an easy power-up menu. Originally the SOB was built as part of the TIM (TI IMAGE MAKER, V9958 video upgrade board) that allows 80 column for the TI console. SOB allowed us to make TI's operating system fully compatible with a newer V9938/9958 video upgrade devices and to fix some problems the "correct" way. But recently SOB has became a hot-seller for those just wishing to access the cataloging and loading features, as it can load with full pathname access for those hard drive owners out there, and it's a great way to load programs also for those without a RAMdisk with a Horizon MENU.

A friend has SOB in his POP-cart:- What if I plugged in their POP-cart with the "SOB-like" add-in, what happens then? .... Nothing, the SOB in the POP-cart is designed to look for a SOB in the console, and use the internal SOB first!

SOB's pop-up menu will allow the RML function of the TI console to work with John Guion's Multi Mod for the Triton SXB module, PGRAM PLUS by Bud Mills Services, and the GRAMULATOR by CaDD Electronics in allowing you to catalog the the modules they contain.

* 100% compatible with 99/4A operating system.

* Updated to be compatible with V9938/58 devices.

* True lower-case added to all modules/programs.

* 4K MICRO-MANAGER for cataloging floppy, ramdisk, hard drives and from the catalog: run Assembly, Object, Forth,or C99 programs.

T-100: $59.00 Canadian / $49.00 U.S.

ORDERING INFORMATION

If overseas, feel free to use either the U.S. or Canadian prices which are listed. Make sure to include a money order for the proper amount. For Shipping and Handling add US/Canada $4.50; Other $7.50

If you have any questions, or just want more information on our products before ordering, feel free to write or phone us. Telephone 416-960-0925(8am-11pm EST).

OASIS PENSIVE ABACUTORS
432 JARVIS ST. #501-502
TORONTO,CANADA. M4Y-2H3 ■

```
100 ! FILE COMPARE D/V 80 XB
110 ! JAN ALEXANDERSSON
120 ! REV 1989-08-12
130 CALL CLEAR
140 INPUT "FILE 1 ":A$
150 IF A$="" THEN END
160 INPUT "FILE 2 ":B$
170 N=0
180 OPEN #1:A$,INPUT
190 OPEN #2:B$,INPUT
200 LINPUT #1:A$
210 LINPUT #2:B$
220 N=N+1 :: PRINT N;
230 IF A$<>B$ THEN PRINT : :
A$:B$: :
240 IF EOF(1)=0 AND EOF(2)=0
 THEN 200
250 IF EOF(1)=0 THEN PRINT "
FILE 1 HAS MORE"
260 IF EOF(2)=0 THEN PRINT "
FILE 2 HAS MORE"
270 CLOSE #1 :: CLOSE #2
280 PRINT : :
290 GOTO 140
```
■

MINA UPPTÄCKTER I LOGO-SPRÅKET, DEL 2

av Kent Edgardh

5. TILLDELNING

För enkel tilldelning av nya värden för variabler finns två primitiver MAKE och CALL. MAKE har som vi sett 2 stycken input. Det första som skall föregås av " har till uppgift att tilldela variabeln just det namn som kommer efter. Det andra inputet i ordningen är innehållet i variabelns värde. Det som skiljer CALL från MAKE är, att de båda inputen är omkastade till ordningsföljden. Någon annan skillnad har jag inte hittat. MAKE kommer därför att mest likna BASIC:ens LET, som för det mesta utelämnas.

Att tilldela en variabel boolskt värde går till genom att omge värdet med " i båda ändarna. Det tog ett tag med sökande och experimenterande innan jag kom på det. För i manualen står det inte ett ord eller ens en antydning om det. Jag hade på olika sätt försökt tilldela en variabel omväxlande värdena true och false efter bästa fantasi. Så här gick det till. Jag ville undersöka om man kunde få hjälp på traven med interpretatorns felmeddelande. Jag tittade i listan över primitiver och fastnade för den fjärde i ordningen, BOTH. Det som står som condition 1 respektive condition 2 bör av nödvändighet vara ett boolsk värde. Kanske skulle det gå att utnyttja. Jag knappade därför in något som jag visste var fel för att få se om interpretatorns repertoar för felmeddelande skulle vakna:

```
MAKE  "A 3
PRINT BOTH :A :A
```

Det blev napp. Nu fick jag ett mycket elegant felmeddelande.

```
3 WAS GIVEN INSTEAD OF
"TRUE" OR "FALSE"
```

Det såg nästan ut som om LOGO-författaren/konstruktören hade utgått från att det var onödigt att skriva ner det i manualen.

Talvärden som i exemplet ovan, där variabeln A har fått värdet 3 behöver väl ingen ytterligare kommentar. Världens mest förekommande tilldelningssats, som den har blivit utnämnd till, är den där en variabel ökar sitt talvärde med 1. I LOGO skulle den se ut på följande sätt

```
MAKE "X :X + 1
```

Alternativet i språket blir det

```
CALL :X + 1 "X
```

I någon mening har BASIC:s strängar sin motsvarighet till LOGO:s ord. Vid tilldelning skall ett " föregå värdet.

```
MAKE "B1 "ABC
```

Här har naturligtvis variabeln med namnet B1 fått strängvärdet ABC.

Kring listor är det nödvändigt att omsluta listans element med olika hakparenteser i var ända. MAKE har ju bara två stycken input. Hakparenteserna är till för att markera att det verkligen är just precis två input.

```
MAKE "D [1 7 AB ]
```

Givetvis kan en variabel tilldelas en annan variabels värde. Då blir den automatiskt av samma typ och värde som den ursprungliga. Det är helt enligt konceptet att variabeltyper inte deklareras i LOGO. Det ser ut så här

```
MAKE "A2 :A1
```

Tilldelning av nya värden måste även kunna göras vid avbrott i programinterpretatorn med inknappning från tangentbordet. Då tänker jag på jämförelsen med BASIC:s INPUT. I LOGO heter det READLINE. READLINE:s utput är en lista. Det skall observeras att även om man bara skrivit en följd med tecken utan mellanrum på sitt tangentbord blir det en lista med bara ett ensamt element. ■

HORIZON COMPUTER

RAMDISK BARE BOARD, Manual + ROS 8.14 $50
Zero K Kit = above + parts NO MEMORY $110
128k Memory NOW $30 each 32k= $9 each

128k Kit	=	$140	or	$170 Built
256k Kit	=	$170		$200 Built
384k Kit	=	$200		$230 Built
512k Kit	=	$230		$260 Built
1 MEG Kit	=	$370		$400 Built
1.5 M Kit	=	$490		$520 Built

NEW LOWER RAMDISK PRICES NOW

Add a RAMBO Mod $45 (KIT)
256/800 PHOENIX KIT=$400 or $430=Built
Digiport = $40 **Horizon Mouse = $40**

P-GRAM Kit	72k	= $150	or	$180 Built
P-GRAM+ Kit	192k	= $180	or	$210 Built
Clock for P-GRAMS		= $20		U/G 72k to 192k $50

ALL KITS Include ALL PARTS, DOCs + Software

MEMory EXpansion for the GENEVE 9640

MEMEX	504K+	$225
MEMEX	504K+GENMOD	$325
MEMEX	1008K+GENMOD	$365
MEMEX	1512K+GENMOD	$405
MEMEX	2016K+GENMOD	$445

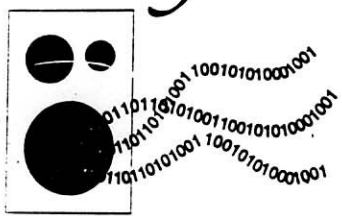
GENMOD is added to YOUR GENEVE Call for INSTALL $

GENMOD allows all 2 meg use at ZERO Wait

ON HOLD > >>>> THE ACCELERATOR >>> ON HOLD
180/256k HRD Mod $40 PUT 32kMEM on HRD $25

Prices will change IF MEMORY COSTS go up
OHIO Residents ADD 6% Sales Tax
FREE Shipping to US & CANADA.Add $5 AIR O/S
Send Order with PHONE # to
BUD MILLS SERVICES
166 Dartmouth Drive
Toledo OH 43614-2911
CALL 419+385-5946 voice or 419+385-7484 BBS
for More Information or Current Pricing

Digi-Port



DIGITAL SOUND PLAYER

For your Texas Instruments 99/4a or Myarc Geneve 9640

Digi-Port is a unique combination of hardware and software to allow your TI 99/4a or Myarc Geneve play TRUE DIGITIZED SOUNDS from a from a MAC, AMIGA, PC or any other digitized sound through your TI, Myarc or Ultracomp PIO port (1 or 2). The package includes an assembly (Both TI and Geneve compatible) sound player that allows you to pick your sound from any directory or disk drive you may have. Digi-Port supports RAMBO compatible memory, such as a Horizon 3000 RAM Disk with RAMBO accessory or a 4a MEMEX. It also supports 32k or 9938/58 VDP memory (up to 192K) on BOTH the Geneve and 99/4a.

Included in each package (SSSD, DSSD, DSDD) are 10 sound disks containing sounds to play, the assembly player, an MDOS player for the Geneve that supports Geneve memory expansion, and BASIC / EXTENDED BASIC call LINKS for playing sounds with RAMBO memory on the 4a.

When combined with Alexander Hulpke's XHI, the XB link provides the FIRST MULTIMEDIA environment available on the 99/4a. Now you can write a SOUND AND SIGHT SPECTACULAR in EXTENDED BASIC and use the true POWER of your 99/4a.

AVAILABLE NOW FROM BUD MILLS SERVICES

WE HARNESS TECHNOLOGY
WESTERN HORIZON TECHNOLOGIES

Introducing:

SCSI

Hard and Floppy disk controller for the TI 99/4a and Myarc Geneve

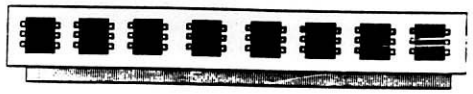
This **ADVANCED** new peripheral for your Texas Instruments 99/4a or Myarc Geneve will **EXPAND** your storage capacity to **HUNDREDS** of megabytes. This high performance disk drive interface allows you to connect up to any combination of 7 SCSI hard and floppy drives with any capacity up the the SCSI limitations. You can connect **FLOPPY** drives, both 3.5 inch and 5.25 inch, with current capacities up to 4 megabytes. Or connect a **WINCHESTER** drive with an astonishing 1.6 **GIGABYTES** of hard disk storage. You can even connect a **CD ROM** player for access to **HUNDREDS** of pictures and sounds!!!

This new peripheral will read and write TI floppies in ALL current formats as well as PC **COMPATIBLE** floppies!! That's right, you get **PC TRANSFER©** **BUILT IN**!!! You can now exchange data **DIRECTLY** with a IBM PC or compatible witout having to convert!

Expand your disk capacity with this new SCSI controller designed by WHT available **SOON** from Bud Mills Services!

4/A MEMEX

Advanced memory peripheral for your TI 99/4a.



Coming soon from WHT and Bud Mills Services is the BRAND NEW 4a Memex RAM expansion peripheral.

This advanced p-box accessory card provides your TI 99/4a system with up to 16 MEGABYTES of PROGRAM SPACE on each card! PLUS you can install up to FOUR cards in one p-box for up to 64 MEGABYTES of memory for programs and data!

Once installed in your expansion box, the 4a Memex allows you to load HUGE programs, sounds (for Digi-Port) and graphics for INSTANTANEOUS access or playback.

The 4a Memex comes with an advanced MEMORY MANAGER BUILT IN to its EPROM based DSR. Other features of the 4a Memex are power up memory test and auto system config and loading. 4a Memex memory can also be used as temporary RAM DISK storage for fast access to frequently used data and program files.

This memory card will provide you with the exspansion you need for the 21st century.

Built using the latest in AMD and SIMM technology, the 4a Memex allows an easy and INEXPENSIVE way to upgrade your computers memory. Each 4a Memex supports INDUSTRY STANDARD SIMM MODULES for memory. SIMM modules can be in 256K x 8 (or 9), 1 MB x 8 (or 9), and 4 MB x 8 (or 9) in cheap 100ns speeds or less. Fours SIMM slots are available for memory configurations of 256k to 4 MB.

This card was built to provide you with the FINAL SOLUTION in memory expansion for your TI 99/4a computer system.

SÄLJES: 2 st TI-99/4A, 1 st expansionsbox som innehåller 32 kbytes minneskort, diskettstation, diskkontrollkort, RS-232 kort. Tillbehör: Speech Synthesizer, Databandspelare med kablar, 2 par TI joystick, joystick-adapter, WICO-joystick, TI-Cartridge expander. Moduler: Personal Record Keeping, Personal Report Generator, Startrek, Amazing, Car Wars, Wumpus, Number Magic, Disk Manager II, Demolision Division, Jawbreaker II, 2 st Indoor Soccer, Tunnels of Doom, TI Extended Basic, Editor/Assembler. Mycket literatur, flera disketter 30-40 st och några kasetter. Albin Rosengren Tel 08-550 30115, Fax samma som telefon. Säljes komplett eller i delar.