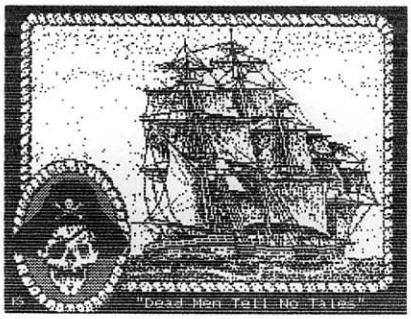
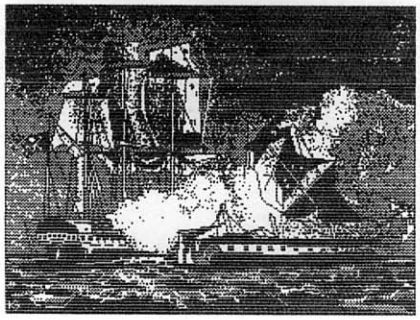
PROGRAM BITEN 92-4





ISSN 0281-1146

P-GRAM+ KORTET

Jag har upptäckt ytterligare två små problem med mitt P-GRAM+ kort.

När jag sparar Mini Memory modulen till disk så kommer >A5 som markerar skrivbart RAM att skrivas till filen för RAM 09 och inte till GROM 3 filen som programladdaren sedan förväntar sig. Du kan dock själv skriva >A5 på rätt ställe med P-GRAM:s egen minnes-editor och sedan spara till skiva och ladda igen.

Jag har inte lyckats få GRAM Packer att fungera trots att jag använder P-PACKER filen. Det verkar som om anropet av flexskiva återställer skrivskyddet i P-GRAM+ beroende på mitt DIJIT AVPC-kort. ■

HORIZON RAM-CHIPS

Horizon 4000 RAM-disk har nu 512 kbytes statiska RAM-chips så att ett kort rymmer 8 Mbytes. Bud Mills Services säljer följande RAM-chips:

32 kbytes (256 kbit)	9 dollar
128 kbytes (1 Mbit)	30 dollar
512 kbytes (4 Mbit)	110 dollar

Färdigbyggda RAM-diskar kostar:

256 kbytes	215 dollar
384 kbytes	245 dollar
512 kbytes	275 dollar
1 Mbytes	385 dollar ■

MYARC 9640 GENEVE

Myarc har upphört att existera som firma. Enligt Micropendium juni och sept 92 kan Myarc-kort repareras av: Cecure Electronics, 7759 So. Scepter Dr. 7, FRANKLIN, WI 53132-2201, USA. Telefon (414) 529-2173.

Rättigheterna till MDOS, ABASIC, PSYSTEM m.m. för Geneve har (Micropendium sept 1992) köpts av: Beery Miller, P.O. Box 752465, MEMPHIS, TN 38175, USA

Källkoden till dessa program finns tillgänglig för de som minst betalt 25 dollar. Det finns sammanlagt 2200 tillverkade Geneve. Den slutgiltiga versionen av MDOS kommer att sändas direkt till alla registrerade Geneve-ägare. ■

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
(Endast om tidningens innehåll)

PROGRAMMING MUSIC - PART 4

by Jim Peterson, Tigercub, USA

The first three parts of this series were written and published some time ago, so I had better review.

In Part 1, I showed you this one-line routine to set up a musical scale.

```
100 DIM N(36):: F=110 :: FOR
 J=1 TO 36 :: N(J)=INT(F*1.0
59463094^(J-1)+.5):: NEXT J
:: N(0)=40000 ::GOTO 110
101 D,T,A,B,C,V1,V2,V3,J,X,V
102 CALL SOUND
103 !@P-
```

That sets up a scale of three octaves beginning with A. If you decide to change the music to a higher key, just change the 110 to 117, 123, 131, 139, 147, 156, 165, 175, 185, 196, 208 or 220. In fact, for some music you will have to change it, if the program crashes with a BAD VALUE error message.

If you have programmed the music with high notes, you can lower the key by changing 110 to 104, 98, 92, 87, 82, 78, 73, 69 or 65. Again, if you try to go too low you will get that BAD VALUE message.

I have given N(0) a value of 40000, which creates a tone too high to be heard. This can be used to silence a note, but it can also cause a crash when used with some of the following routines. If you are programming three voices and want to play a single note, the easiest way is to give all three notes the same number, such as A,B,C=10. If you need a silent rest, play all the notes at an inaudible volume by V1,V2,V3=30 and then, after the GOSUB, restore their original volume by V1= (whatever is in line 110) and the same for V2 and V3.

Lines 101-103 are a pre-scan routine to start the music playing sooner. There will still be a few seconds delay while that array is set up in line 100. You can perhaps shorten that delay slightly by changing the 36 to the highest note number you have used in programming your piece.

However, Bruce Harrison wrote for me an assembly link which eliminates any delay; this also makes it possible to change key while the music is playing. I won't list the source code here, because everyone is afraid to key in source code anyway, but it is available on my TI-PD disk #1143 and will also be on a tutorial disk on this type of music programming.

Part 2 of this series contained a listing of a program to easily give you the numbers you would need in order to key in a particular piece of sheet music. If you don't have that, you can just take a piece a paper and list the scale A Bf B C C# D Ef etc., on through as many as you will need, and then number them consecutively. For the length of the notes, give the shortest note a value of 1 unless it also appears as a dotted note, in which case it must be 2, and then number the others according to their relative length - for example, 2 for a quarter note, 3 for a dotted quarter, 4 for a half note, 8 for a whole note.

Part 2 showed you how to key in single-note music, and Part 3 showed how to do 3-part harmony. To recap briefly -

First, save yourself a lot of work by identifying any groups of notes in the sheet music that are repeated two or more times. Mark them off wherever they appear. Key them in first, starting with line number 500; at the end, put RETURN. If you find another such series, label it 600 and do the same; you may find several such series. Just stay below line number 1000, which is reserved for mergeable routines. Then, while you are programming the music and come to such a series of notes, just put in GOSUB 500 or whatever.

Start keying in your music in line 120; line 110 is reserved for a line

to be merged in. To key in the music, just give T the number for the length of the first note, and give A, B and C the numbers for the melody and first and second part harmony. Then GOSUB 1000. For instance, T=1 :: A=23 :: B=18 :: C=12 :: GOSUB 1000 .

And for each succeeding note, give a new value to whatever changes; if T is still 1 and B and C are still the same, all you need is, for instance, A=19 :: GOSUB 1000.

Merge in one of the following routines, put in a line 999 STOP, and after every several notes enter RUN and listen to what you have done so far, to catch any errors while it is still easier to find them.

You can merge in any of the following routines to create many different musical effects. The D in line 110 controls the tempo of the music; change it as you wish. V1, V2 and V3 are the volume (loudness) of the three voices; adjust them as you like.

Key this in and save it by SAVE DSK1.PLAY1,MERGE

```
110 D=500 :: V1=1 :: V2=5 ::
V3=7
CALL SOUND(D*T,N(A),V,N(B),V
,N(C),V):: RETURN
```

That plays simple 3-part music, all at the same volume, which may sound rather harsh to your years. Try changing the second V to V2 and the 3rd one to V3. Save that as PLAY2.

For a bass accompaniment in the 3rd voice, change that to

```
CALL SOUND(D*T,N(A),V1,N(B),
V2,N(C)*3.75,40,-4,V3)
```

For a bass melody with accompaniment, change the A to C, V1 to V3, C to A and V3 to V1.

For the melody in two voices two octaves apart, change the C back to A and the V3 back to V1. Are you beginning to see how many different effects can be created by making changes in just this one line? Save any ones you like in merge format with a different name for each.

Perhaps those bass notes sound too deep. Try changing the 3.75 in any of those routines to 7.5 . Better yet, change it to X and add :: X=3.75 to line 110. Then you can switch back and forth in your music by simply X=7.5 or X=3.75. Getting interesting, no?

Music played in that way has a strong throbbing beat, so try this method -

```
110 D=4 :: V1=1 :: V2=5 :: V
3=7
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),V1,N(B),V2,
N(C),V3):: NEXT J :: RETURN
```

I'll be referring back to this one as the negative duration method. Again, you can change the tempo by changing the value of D, but sometimes not as exactly as I would like. With this method, you will find that a series of the same note runs together into a single long note. To avoid this, use different harmony notes each time, or different volumes for V2 and V3.

There's no law that says the harmony has to be lower than the melody, so try changing N(B) to N(B)*2 or even N(B)*4 or do the same with N(C), or both. Or, use *X, add X=1 to line 110, and then in the middle of your music program you can switch by X=2 or X=4 (don't try 3!)

For a vibrato effect, we alternate a note with the same note multiplied by 1.01 -

```
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),V1,N(B),V2,
N(C),V3):: CALL SOUND(-4250,
N(A)*1.01,V1,N(B),V2,N(C),V3
):: NEXT J :: RETURN
```

For vibrato in the harmony rather than the melody, multiply N(C) or N(B), or both, by 1.01 instead - or multiply all three.

For a stronger vibrato, change the 1.01 to 1.02 or even 1.03. Of course, you can also multiply the harmony notes in both CALL SOUNDs by 2 or 4, as above. Or for a "chop"

effect, multiply them in one CALL SOUND but not the other. The possibilities are almost endless!

For a tremolo, we alternate the volume rather than the frequency. Add X=3 to line 110 and use this routine -

```
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),V1,N(B),V2,
N(C),V3):: CALL SOUND(-4250,
N(A),V1+X,N(B),V2,N(C),V3)::
 NEXT J :: RETURN
```

You can vary the value of X as much as you want (V3+X can't total more than 30) for any amount of tremolo from a flutter to a wobble or a stutter, and you can put the +X after V1 or V2 or all three. You can even change it in the middle of your music, by X= whatever you want.

And you can multiply any or all by 1.01 for different combinations of vibrato and tremolo.

To enhance a note, play it twice in the CALL SOUND but multiply one of its voices by 1.01 -

```
110 D=4 :: V1=1 :: V2=5 :: V
3=7
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),V1,N(A)*1.0
1,V1,N(B),V2):: NEXT J :: RE
TURN
```

Of course, with this trick you can only have 2-part harmony, but you can choose to enhance the harmony rather than the melody.

Now, try combining the enhanced note with the vibrato and/or tremolo, for many more effects. For enriched vibrato, use N(A),V1,N(A)*1.01,V1 in the first CALL SOUND and N(A)*1.01, V1,N(A)*1.02, V1 in the second.

The bass notes do not go well with this method because interrupting them through a loop introduces a rattle, but the baritone works well and gives a unique reedy sound. To do this, place the note you want in the 3rd position, multiply it by 7.5, give it a volume of 30, and add the -4 noise at whatever volume you want. You can also combine this with other effects, for instance with vibrato

```
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),V1,N(B),V2,
N(C)*7.5,30,-4,V3)
1010 CALL SOUND(-4250,N(A)*1
.01,V1,N(B),V2,N(C)*7.5,30,-
4,V3):: NEXT J :: RETURN
```

Now for the real fun - the "piano" effects that we get by decreasing the volume gradually. This is the basic routine -

```
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),J+V1,N(B),J
+V2,N(C),J+V3):: NEXT J :: R
ETURN
```

Of course, with all of these you must also have that line 110 to define the duration and volume.

If you want a little more percussion in your piano, try this -

```
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),J*1.5,N(B),
J*1.5,N(C),J*1.5):: NEXT J :
: CALL SOUND(-4250,N(A),15,N
(B),15,N(C),15):: RETURN
```

And, of course, all those tricks we learned above - vibrato, tremolo, bari- tone, enhanced, high harmony, chop - can also be used with piano. This will give you the vibrato -

```
1000 FOR J=1 TO T*D :: CALL
SOUND(-4250,N(A),J+V1,N(B),J
+V2,N(C),J+V3):: CALL SOUND(
-4250,N(A)*1.01,J+V1,N(B),J+
V2,N(C),J+V3):: NEXT J :: RE
TURN
```

And an increasing tremolo -

```
1000 FOR J=1 TO T*D :: V=J*2
 :: CALL SOUND(-4250,N(A),J,
N(B),J,N(C),J):: CALL SOUND(
-4250,N(A),V,N(B),V,N(C),V):
: NEXT J :: RETURN
```

And just one more, the "reverse piano" with an increasing volume -

```
1000 FOR J=T*D TO 1 STEP -1
:: CALL SOUND(-4250,N(A),J+V
1,N(B),J+V2,N(C),J+V3):: CAL
L SOUND(-4250,N(A),J+V1,N(B)
,J+V2,N(C),J+V3):: NEXT J ::
 RETURN
```

By the time you get through exploring all the possible combinations of those, you should have a hundred ways of making music. Save each one you like, complete with line 110, in merge format, so you can try them all with each piece of music you create.

I had intended this to be the last part of this series, but I still haven't told you about autochording, so there will have to be one more. ■

TIPS FROM TIGERCUB #63

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

****

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 492 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 with Supplement #1, listing all titles and authors, is available for $1 which is deductible from the first purchase.

Several articles have been published on the subject of using Funlweb as a simple fixed-field data base. Sometimes you might want to rearrange the sequence of fields in such a file. This mini-program will quickly change the position of any field in a D/V80 file.

```
100 DISPLAY AT(3,8)ERASE ALL
:"FIELDSWITCHER":"":"    by
Jim Peterson":"":" To chang
e sequence of     fields in
a DV80 fixed fieldfile creat
ed by Funlweb or  other mean
s"
110 DISPLAY AT(23,6):"PRESS
ANY KEY" :: DISPLAY AT(23,6)
:"press any key" :: CALL KEY
(0,K,S):: IF S=0 THEN 110 EL
SE CALL CLEAR
120 DISPLAY AT(8,1):"FILENAM
E? DSK" :: ACCEPT AT(8,14):F
$
130 OPEN #1:"DSK"&F$,INPUT
140 DISPLAY AT(12,1):"MOVE F
IELD STARTING AT WHAT POSITI
ON?" :: ACCEPT AT(13,11)VALI
DATE(DIGIT):N
150 DISPLAY AT(15,1):"LENGTH
 OF FIELD?" :: ACCEPT AT(15,
18)VALIDATE(DIGIT)BEEP:L
160 DISPLAY AT(17,1):"TO WHA
T POSITION?" :: ACCEPT AT(17
,19)VALIDATE(DIGIT)BEEP:T
170 IF T>N+L-1 OR T<N THEN 1
90
180 CALL SOUND(400,110,0,-4,
0):: DISPLAY AT(23,1)BEEP:"C
ANNOT MOVE FIELD WITHIN ITSO
WN PARAMETERS!" :: GOTO 140
190 DISPLAY AT(19,1):"OUTPUT
 FILENAME? DSK" :: ACCEPT AT
(19,21)BEEP:OF$
200 OPEN #2:"DSK"&OF$,OUTPUT
210 LINPUT #1:M$ :: M$=M$&RP
T$(" ",80-LEN(M$)):: IF T<N
THEN M$=SEG$(M$,1,T-1)&SEG$(
M$,N,L)&SEG$(M$,T+1,N-T)&SEG
$(M$,N+L+1,255)
220 IF T>N THEN M$=SEG$(M$,1
,N-1)&SEG$(M$,N+L,T-N-L)&SEG
$(M$,N,L)&SEG$(M$,T+1,255)
```

```
230 PRINT #2:M$ :: IF EOF(1)
<>1 THEN 210 ELSE CLOSE #1 :
: CLOSE #2
240 DISPLAY AT(12,1)ERASE AL
L:"ANOTHER? Y/N" :: ACCEPT A
T(12,14)VALIDATE("YN")SIZE(1
)BEEP:Q$ :: IF Q$="Y" THEN 1
20 ELSE CALL CLEAR :: STOP
```

And this one will make it easy to completely rearrange the sequence of any number of fields.

```
100 DISPLAY AT(3,9)ERASE ALL
:"REARRANGER":"":"     by Ji
m Peterson"
110 DISPLAY AT(7,1):" To re
arrange the sequence of fiel
ds in a DV80 file of fixed f
ields created by     Funlweb
 or otherwise."
120 DISPLAY AT(24,7):"PRESS
ANY KEY" :: DISPLAY AT(24,7)
:"press any key" :: CALL KEY
(0,K,@):: IF @=0 THEN 120
130 DIM L(20),S(20),F$(20)::
 CALL CLEAR
140 DISPLAY AT(8,1):"INPUT F
ILENAME?":"":"DSK" :: ACCEPT
 AT(10,4)BEEP:I$ :: OPEN #1:
"DSK"&I$,INPUT
150 DISPLAY AT(12,1):"OUTPUT
 FILENAME?":"":"DSK" :: ACCE
PT AT(14,4)BEEP:O$:OPEN #1:"
DSK"&O$,OUTPUT
160 DISPLAY AT(16,1):"HOW MA
NY FIELDS?" :: ACCEPT AT(16,
18)VALIDATE(DIGIT)SIZE(2):F
:: CALL CLEAR
170 FOR J=1 TO F :: DISPLAY
AT(12,1):"FIELD #";J;"LENGTH
?" :: ACCEPT AT(12,20)VALIDA
TE(DIGIT)BEEP:L(J):: NEXT J
:: FOR J=1 TO F
180 DISPLAY AT(12,1):"IN FIE
LD #";J:"":"PLACE FIELD #" :
: ACCEPT AT(14,15)VALIDATE(D
IGIT)BEEP:S(J)
190 IF S(J)<1 OR S(J)>F THEN
 CALL SOUND(300,110,0,-4,0):
: GOTO 180
200 IF POS(E$,CHR$(S(J)),1)=
0 THEN E$=E$&CHR$(S(J)):: GO
TO 220
210 CALL SOUND(300,110,0,-4,
0):: DISPLAY AT(16,1):"FIELD
 #";S(J);"HAS ALREADY  BEEN
PLACED!" :: GOTO 180
220 NEXT J
230 LINPUT #1:M$ :: M$=M$&RP
T$(" ",80-LEN(M$)):: P=1 ::
FOR J=1 TO F
240 F$(J)=SEG$(M$,P,L(J))::
P=P+L(J):: NEXT J
250 FOR J=1 TO F :: N$=N$&F$
(S(J)):: NEXT J :: PRINT #2:
N$ :: N$=""
260 IF EOF(1)<>1 THEN 230 EL
SE CLOSE #1 :: CLOSE #2 :: S
TOP
```

If you need to use either of those programs on files with a record length other than 80, just add VARIABLE (or FIXED) and the record length to all the file opening statements, and change that 80 in line 210 or 230.

This subprogram, in which X=28 for a 28-column screen or whatever width you want, will reformat a string of almost any length to print on screen without breaking words, and will return in L the number of lines required to print it, which can be used to space DISPLAY AT statements.

```
31993 SUB FORMAT(X,M$,L):: Y
=X
31994 IF LEN(M$)<Y+1 THEN 31
996 ELSE IF LEN(M$)<Y+X+1 AN
D SEG$(M$,Y,1)=" " THEN 3199
6 ELSE IF LEN(M$)<Y+X+1 AND
SEG$(M$,Y+1,1)=" " THEN 3199
6 ELSE P=Y-1
31995 IF P<1 THEN 31996 ELSE
 IF SEG$(M$,P,1)=" " THEN M$
=SEG$(M$,1,P)&RPT$(" ",Y-P)&
SEG$(M$,P+1,255):: Y=Y+X ::
GOTO 31994 ELSE P=P-1 :: GOT
O 31995
31996 L=INT(LEN(M$)/X)-(LEN(
M$)/X<>INT(LEN(M$)/X)):: SUB
END
```

The following little program, plus the magic of Funlweb, should be all the mailing list program that most people would need for home use. Just use Funlweb to create a file with name on the first line, address on the second line, city and state on the third - or use 4 or even 5 lines for the address if you need to, but the 6th line must either be

blank or contain selection codes. These codes can be anything you want, such as C for everyone you want to send a Christmas card to, or B11 to send a birthday card in November, or whatever.

You can put as many codes as you want to on that line, separated or strung together but be sure not to use a code that is part of another code - for instance, if you use B11 for those November birthdays, don't use B or 1 or B1 or 11 for something else.

Then continue with the next address in another block of six lines. Just be sure that the line number of the line just above the first address line is always a multiple of six.

```
100 DISPLAY AT(12,1)ERASE AL
L:"Filename? DSK" :: ACCEPT
AT(12,14)BEEP:F$ :: OPEN #1:
"DSK"&F$,INPUT :: OPEN #2:"P
IO"
110 DISPLAY AT(14,1):"Print
addresses with code -":"":"(
to print all addresses,    j
ust press Enter)"
120 ACCEPT AT(15,1)BEEP:X$
130 LINPUT #1:A$ :: LINPUT #
1:B$ :: LINPUT #1:C$ :: LINP
UT #1:D$ :: LINPUT #1:E$ ::
LINPUT #1:F$
140 IF POS(F$,X$,1)<>0 OR X$
="" THEN PRINT #2:A$:B$:C$:D
$:"":""
150 IF EOF(1)<>1 THEN 130 EL
SE CLOSE #1
```

In Tips #62 I reported on the weird behavior of the CALL LOAD(-31961,149), when used to clear all defaults and search for a LOAD file on DSK1. I have since found that if you put this CALL at the beginning of a program, it will not execute until an END or STOP is reached - but if you break the program with FCTN 4, it will not be in memory!

I stated that after this CALL LOAD was executed, any number taken to the power of 0 (which should be a value of 1) acquired a value of 220.5727273. I was led astray by the INT in the the formula in which I first found this puzzle. Actually it is 220.57000101, which prints to the screen in the peculiar format F0.57000101.

If a number between 1 and 9 is added to that, it is printed as 1< followed by the number being added, followed by the decimal part. For a number between 10 and 19, the < is changed to = and between 20 and 29 it becomes > (note the ASCII sequence); from 30 to 35 it becomes ? but from 36 to 99 99 the decimal portion is preceded by 0 to 63 respectively. 100 is 2<0.570001 and the pattern continues.

Although these are not valid representations of numbers, they are treated as such. Run a program to give N the power of 2^0, then break the program and experiment in immediate mode.

PRINT N gives that strange F0.57000101. PRINT N+1, or whatever, gives values represented in the format described above. PRINT N*1 will give the true numeric value 220.57000101 but multiplying by some other values gave me results in the odd format, as did dividing.

Peter Walker pointed out to me that trying to subtract from N within a program resulted in printing a value followed by a crash reporting a SYNTAX ERROR (in the line which had just been executed!) followed by a jump to a non-existent line zero!

N-1 should be 219.57.. of course, but in immediate mode PRINT N-1 results in 63.57000101. In the format in which added values are printed, this would be 319.57000101 but the 63.. is actually a decimal value, as can be proved by PRINT CHR$(INT(N-1))! When I tried to get a zero value by PRINT N-64.57000101, the

computer blew its mind.
Does anyone know what is going on here?

An IBM program called DOC-SMASH, which sells for about $35, will read a D/V80 file and output it to a printer in full carriage-width lines of elite condensed subscript thereby getting up to 216 lines per page. Bud Wright wrote a TI version, with assembly links, to let us do the same thing for free. His version wouldn't work on my trusty old Gemini 10X, which does not support condensed elite, so I wrote this mini-program which is not as fast as Bud's, but does the job.

```
100 DISPLAY AT(3,9)ERASE ALL
:"TEXTSMASHER":"":"For the G
emini 10X printer, to print
D/V80 text in linesof 136 ch
aracters closely spaced, i
n subscript."
110 DISPLAY AT(20,1):"Press
Enter to end input" :: DIM F
$(20)
120 F=F+1 :: DISPLAY AT(12,1
):"FILE #";F:"DSK" :: ACCEPT
 AT(13,4)BEEP:F$(F):: IF F$(
F)<>"" THEN 120
130 OPEN #2:"PIO",VARIABLE 2
55 :: PRINT #2:CHR$(27)&CHR$
(83)&CHR$(1);
140 PRINT #2:CHR$(15)&CHR$(2
7)&CHR$(51)&CHR$(12);:: LN=1
36
150 FOR J=1 TO F-1 :: OPEN #
1:"DSK"&F$(J),INPUT
160 LINPUT #1:M$
170 IF LEN(T$)>0 THEN M$=T$&
" "&M$ :: T$=""
180 IF LEN(M$)<LN+1 AND POS(
M$,CHR$(13),1)<>0 THEN PRINT
 #2:M$ :: GOSUB 260 :: M$=""
 :: GOTO 230
190 IF LEN(M$)=LN THEN PRINT
 #2:M$ :: GOSUB 260 :: M$=""
 :: GOTO 230
200 IF LEN(M$)<LN AND EOF(1)
<>1 THEN LINPUT #1:X$ :: M$=
M$&" "&X$ :: GOTO 170 ELSE I
F LEN(M$)<136 THEN PRINT #2:
M$ :: GOSUB 260 :: GOTO 240
210 P=LN
220 IF SEG$(M$,P,1)=" " THEN
 T$=SEG$(M$,P+1,255):: M$=SE
G$(M$,1,P):: PRINT #2:M$ ::
GOSUB 260 :: M$="" :: GOTO 2
30 ELSE P=P-1 :: GOTO 220
230 IF LEN(T$)<LN+1 AND POS(
T$,CHR$(13),1)<>0 THEN PRINT
 #2:T$ :: GOSUB 260 :: T$=""
240 IF EOF(1)<>1 THEN 160
250 CLOSE #1 :: NEXT J :: ST
OP
260 X=X+1 :: IF X<121 THEN R
ETURN
270 X=0 :: FOR K=1 TO 8 :: P
RINT #2 :: NEXT K :: RETURN
```

For that to work properly, your paragraphs must end in carriage returns, and so must the title line, etc. If such is not the case, try Bill Wood's method - load the file into Funlweb, enter RS for Replace String, then /. /.X/ but instead of X type CTRL U SHIFT M. At the first prompt, enter A for All. If your text has any paragraphs ending in ? or !, get your cursor back to the beginning, change that first period to ? or !, and do it again. You might also need to manually add carriage returns to titles, etc. Just type CTRL U, then SHIFT M wherever a CR is needed.

Without having printers to test it on, I think the program can be modified for the SG-10 by changing line 140 to

```
140 PRINT #2:CHR$(27)&"B"&CH
R$(4)&CHR$(27)&CHR$(51)&CHR$
(12);:: LN=160
```

And for old Epson-type printers which don't support elite condensed by

```
140 PRINT #2:CHR$(27)&CHR$(7
7)&CHR$(27)&CHR$(51)&CHR$(18
);:: LN=132
```

And new Epson compatibles by

```
140 PRINT #2:CHR$(27)&CHR$(7
7)&CHR$(15)&CHR$(27)&CHR$(51
)&CHR$(18);:: LN=160
```

You might also have to change that 8 to 12 in line 270 - my old Gemini seems to think that 11*12=128.

COMPLETELY out of memory,
Jim Peterson ■

FAST EXTENDED BASIC!
88/05 - STAMPDBASE

The purpose: a tutorial on the many options you can use with the DISPLAY AT and ACCEPT AT statements. The solution: a screen-input oriented data base. The challenge: a program small enough to fill just over a page. The result: STAMPBASE.

The drawbacks: the SORT function will be added next month. The data base can hold only a limited number of records: 75 (or 125 with memory expansion). And if you want to have different filenames, you will have to add a SUB to ask for it (lines 180 and 360). But that should be no problem once you have studied STAMPDBASE!

I choose a stamp data base because it uses every kind of ACCEPT AT options. But since I have kept the ACCEPT and DISPLAY functions separate in user-defined subs, it is very easy to modify the program to do an address book, or any other kind of small data base that you might need.

The program is very fast because it is kept simple: the data is all in memory; as it is kept in a text file (DIS/VAR 80), it prints fast; saving it in the same format is not the best way (an INT/FIX format would be faster), but here we can use the same coding for both; as a bonus, you can use your file with TI-WRITER. And all the functions (see lines 260 and 270) are available at all times from the main screen.

The beginning of the program should by now be easy to understand; in line 130, "@" is redesigned as an "x" and "%" as a "y", because "x" and "y" are themselves redesigned in line 140, which defines characters to draw a stamp on the screen (line 160). We encounter our first ACCEPT sub in line 170: CALL AS; go to line 980 for the explanation: Tex will accept at row R=11, column C=25, with a size S=1, and validate only the string "YN" (it will not accept anything else); the result will be returned in the string A$. This is the normal way to use ACCEPT with the VALIDATE(string) option; and normally we would check IF A$="Y" or A$="N", but here we have to use the ASC value of A$ because, in this program, all ACCEPT subs have the statement "GOSUB 880". This sub pads each string input with spaces (depending on its length) and a delimitator "¦", so that all our records are built the same way. This will become clear when we get into entering and using the data proper.

The SIZE, which tells Tex how many characters to accept, is made negative in the sub; this allows you to have a default value for the ACCEPT: the space alloted to the entry will not be erased, so whatever is on the screen will be kept or can be modified; here, our default is "Y"; in line 850 (QUIT), built the same way, the default will be "N". This program uses only negative SIZEs, so that you can easily REDO your data entries.

The data for each stamp consists of eight ACCEPTable fields: Country, Date, Scott catalogue and "picture" numbers, Face Value (printed on the stamp), Color, whether you have a New or Used copy, and finally the Catalogue Value, which hopefully will go up over the years. The white "¦" delimitators in the screen display are markers to show you the extent of the size of each field: yes, you can type CZECHOSLOVAKIA, but not DOMINICAN REPUBLIC.

At present, there is only one "default" value put in, for the New/Used, presuming the bulk of your collection is used stamps; if you collect only from one country, you can easily add its name to line 200: replace "C ¦" with "C¦CANADA" for example. The variable F in line 220 is a flag to tell Tex if you have loaded a file or not.

Lines 230 and 240 call user-defined subs for the almost complete range of ACCEPT AT options; refer to lines 940-980 and their trailing remarks to understand what they do. Note the difference between DIGIT and NUMERIC: the catalogue numbers are only digits, while the values (face and catalogue) can be 50 (cents) or 2.00 (dollars). Better, type the program and try each one: even if you never collected stamps, you will quickly learn what kind of data is expected, because if you press the wrong key, Tex will beep and do nothing. This is the beauty of the VALIDATE option: get only what you need, and no need for IFs to check the errors! In all these statements, the parameters are passed in the same order: row, column, maximum size, return string; only CALL/SUB AS has one more, the validated string. As I told you, very easy to modify for your own needs, and the CALLS look much neater than a bunch of ACCEPT AT(R,C)VALIDATE(WHATEVER) SIZE(-S)!

All the (already padded) data is gathered in one string in line 250, together with the padded record number. The Menu is then displayed, with its inevitable CALL KEY. There are 10 options; that would need a lot of IF K=, with no easy checking for out of range keys, since there is no real range. So we check the position of the key value in a string (line 290), a much much more elegant and shorter way. Lines 300 to 390 are the action lines, documented by trailing remarks.

In the ADD line (300), change the maximum T to 125 if you have the memory expansion (and don't forget to re-DIM S$(125) in line 120). When you DELETE a record (line 310), all you do is erase its content; the screen will show "~~ del ~~" as the country name, and you can always reuse the record with REDO. As they say in the DBASE III PLUS manual, these records are only "marked for deletion". Deleting a record in the middle of a file takes a lot of time; to save it, we will actually delete these records each time a SORT will be done: this is called "packing" a file.

The same coding is used for printing or saving to disk, since both are D/V 80 files; these files are the default for Tex, so no need to qualify the statement with "VARIABLE,OUTPUT,etc". The SP$ (spaces) will make a nice margin on paper only. And only the printed version will have a proper header (line 370; watch for the spaces and "¦s"). Since both GET a record and DELETE it need a record number to start with ("0" is the escape), we use the same sub to get it. In the sub itself (line 890), we encounter a new ACCEPT AT: when you accept a numeric value, you cannot validate anything of course, so we cannot use any of our CALLs here, only specify the row and column. Note the SIZE though: it will take only 2 digits, without erasing the remainder of the screen line (change the size to 3 if you have the memory expansion).

The sub in lines 900-910 displays a record on the screen; again, I used a user-defined sub; for typing easiness, we read S$(N), our complete record, into A$; we pass the following parameters: row, column, string, position of segment to read from it, and number of characters to read. The SUB DR is in line 990. Now you understand the importance of padding each field entry with trailing spaces! All records being built alike, it is an easy task for Tex to reconstruct each field for screen display (the "¦s" in the records are not really needed, but when you print the data base, you get instant lines to separate each column). We could have used as many arrays as there are fields, but... what a job to write the program, let alone to type it and modify it!

Two finals notes on DISPLAY AT: in line 920, the SIZE of 3 will not disrupt the rest of the line; depending on the length of STR$(T), we will get "2~~" or "34~" or "123"; the "~s" are part of the stamp design. And, Henri, DISPLAY AT(R,C): "" (not used in this program) will simply erase the rest of a line by displaying nothing.

NOTE: see 88/06/TXT for an explanation of lines 390-530 (the SORT function), 350-356 (a GET/search

function) and 850-855, a routine to add the total value of your collection.

MG EPROM till Corcomps Diskkontrollkort säljes av Bud Mills Services för 35 dollar + 5 dollar porto. ■

```
100 REM ** STAMP DBASE ** /
L. Dorais / April 1988
110 REM
120 ON WARNING NEXT :: DIM S
$(75)
130 CALL CLEAR :: CALL COLOR
(12,16,1):: ER$=RPT$(" ",140
):: CALL CHARPAT(120,A$,121,
D$):: CALL CHAR(64,A$,37,D$)
140 CALL CHAR(125,"000000FF"
,120,"0000000000183CFFFF3C18
0000000000010103070703010180
80C0E0E0C08080")! stamp desi
gn
150 GOTO 160 :: F,K,N,P,S,T,
TN,X,Y,Z,COL$,COU$,DA$,FV$,N
U$,PI$,SN$,SP$,SR$,VA$ :: CA
LL KEY :: !@P-
160 A$="z~~~~{" :: DISPLAY A
T(1,2):"xxxx":A$;TAB(15);"ST
AMP":A$:A$;TAB(13);"DATA BAS
E":A$:A$:" yyyy" ! graphic a
nd title
170 DISPLAY AT(11,1):"LOAD A
N EXISTING FILE?  Y" :: CALL
 AS(11,25,1,A$,"YN")
180 IF ASC(A$)=78 THEN N,T=1
 :: GOTO 200 ELSE F=1 :: OPE
N #1:"DSK1.STAMPDATA"
190 X=X+1 :: INPUT #1:S$(X):
: T=T+1 :: IF EOF(1)THEN CLO
SE #1 :: GOTO 200 ELSE 190 !
 read a DV80 file
200 GOSUB 920 :: DISPLAY AT(
2,10):"Record # " :: DISPLAY
 AT(4,10):"C ¦";TAB(28);"¦"
:: DISPLAY AT(6,10):"D ¦
   ¦"
210 DISPLAY AT(9,1):"Scott>
# ¦    ¦   Pic ¦    ¦": :"Fa
ce Value  ¦    ¦": :"     Co
lor  ¦";TAB(28);"¦"
220 DISPLAY AT(15,1):"  Used
/New  ¦U¦": :"Cat. Value  ¦
   ¦": :RPT$("¦",28):: IF F=
1 THEN F=0 :: N=1 :: GOSUB 9
00 :: GOTO 260
230 DISPLAY AT(2,19):STR$(N)
:: CALL AU(4,13,15,COU$):: C
ALL AN(6,13,7,DA$):: CALL AD
(9,11,4,SN$):: CALL A(9,24,4
,PI$)! accept data
240 CALL AN(11,14,4,FV$):: C
ALL AU(13,14,14,COL$):: CALL
 AS(15,14,1,NU$,"UN"):: CALL
 AN(17,14,4,VA$)
250 S$(N)=SEG$(STR$(N)&"  ",
1,3)&"¦"&COU$&DA$&SN$&PI$&FV
$&COL$&NU$&VA$
260 DISPLAY AT(20,1)BEEP:" [
A]dd a record   [F]orward":"
 [D]el a record   [B]ackward
":" [R]edo record    [P]rint
 DB"
270 DISPLAY AT(23,1):" [G]et
 a record   [S]ave DB":" [*]
sort a field  [Q]uit"
280 CALL KEY(3,K,S):: IF S=0
 THEN 280
290 P=POS("ADRG*FBPSQ",CHR$(
K),1):: IF P=0 THEN 280 ELSE
 ON P GOTO 300,310,320,350,3
90,330,340,360,360,850
300 GOSUB 870 :: IF T<75 THE
N T=T+1 :: N=T :: GOTO 200 E
LSE 260 ! add record
310 A$="DEL" :: GOSUB 890 ::
 IF P THEN N=P :: S$(N)=SEG$
(S$(N),1,4)&"~~ del ~~" :: G
OSUB 900 :: GOTO 260 ELSE 26
0 ! delete
320 GOSUB 870 :: GOTO 230 !
redo record
330 IF N=T THEN 260 ELSE N=N
+1 :: GOSUB 900 :: GOTO 260
! forwrd
340 IF N=1 THEN 260 ELSE N=N
-1 :: GOSUB 900 :: GOTO 260
! bckwrd
350 GOSUB 870 :: DISPLAY AT(
20,2):"Press:  D": :" [R] to
 get a RECORD NUMBER":" [D]
to get DATA"
351 CALL AS(20,10,1,A$,"RD")
:: IF ASC(A$)=82 THEN 356
352 GOSUB 870 :: DISPLAY AT(
21,1):"DATA TO GET: " :: ACC
EPT AT(21,14):D$ :: Z=1
353 FOR X=Z TO T :: IF POS(S
$(X),D$,5)=0 THEN 355 ELSE N
=X :: GOSUB 900 :: DISPLAY A
T(23,1)BEEP:"THIS RECORD? Y"
354 CALL AS(23,14,1,A$,"YN")
:: IF ASC(A$)=89 THEN 260 EL
SE DISPLAY AT(23,1):"" :: Z=
X+1 :: GOTO 353
355 NEXT X :: DISPLAY AT(23,
9)BEEP:"NOT FOUND !" :: FOR
X=1 TO 300 :: NEXT X :: GOTO
 260
356 A$="GET" :: GOSUB 890 ::
 IF P THEN N=P :: GOSUB 900
:: GOTO 260 ELSE 260 ! get r
ecord by no.
```

```
360 IF P=8 THEN D$="PIO" ::
SP$="      " ELSE D$="DSK1.ST
AMPDATA" :: SP$="" ! print/s
ave data base
370 OPEN #1:D$ :: IF P=8 THE
N PRINT #1:SP$&"Rec¦    Coun
try    ¦ date  ¦Sc# ¦Pic ¦Fa
ce¦    Color     ¦S¦Val.¦":S
P$&RPT$("-",65)
380 FOR X=1 TO T :: PRINT #1
:SP$&S$(X):: NEXT X :: CLOSE
 #1 :: GOTO 260
390 TN=N :: N=0 :: S$(0)="SR
T" :: GOSUB 900 ! sort on on
e field only
400 DISPLAY AT(20,1):" Enter
 ""***s"" in the field     %
ou want to SORT on.":"":" To
 return to Menu,  enter
nothing ever%where."
410 CALL A(4,13,15,COU$):: C
ALL A(6,13,7,DA$):: CALL A(1
3,14,14,COL$)! get field to
sort
420 S$(0)="0  ¦"&COU$&DA$&RP
T$("    ¦",3)&COL$&" ¦   ¦"
430 P=POS(S$(0),"*",1):: IF
P=0 THEN N=TN :: GOSUB 900 :
: GOTO 260 ELSE GOSUB 870 ::
 DISPLAY AT(22,9):"SORTING..
."
440 S=POS(S$(0),"¦",P):: S=S
-P :: D$=RPT$("~",65):: K=1
450 K=2*K :: IF K<=T THEN 45
0
460 K=INT(K/2):: IF K=0 THEN
 510
470 FOR X=1 TO T-K :: Y=X ::
 IF POS(S$(Y),"~",5)<>0 THEN
 S$(Y)=D$
480 Z=Y+K :: IF SEG$(S$(Y),P
,S)<=SEG$(S$(Z),P,S)THEN 500
490 A$=S$(Y):: S$(Y)=S$(Z)::
 S$(Z)=A$ :: Y=Y-K :: IF Y>0
 THEN 480
500 NEXT X :: GOTO 460
510 FOR X=1 TO T :: IF POS(S
$(X),"~",5)<>0 THEN T=X-1 ::
 GOSUB 920 :: GOTO 530 ! sto
p at first del. = pack
520 S$(X)=SEG$(STR$(X)&"  ",
1,3)&SEG$(S$(X),4,65):: NEXT
 X ! re-number each record
530 N=1 :: GOSUB 900 :: GOTO
 260
850 GOSUB 870 :: DISPLAY AT(
21,8):"QUIT (Y/N)?  N": :"[E
nter ""T"" to get total cat.
value before quitting pgm.]"
851 CALL AS(21,21,1,A$,"YNT"
):: IF ASC(A$)=78 THEN 260 E
LSE IF ASC(A$)=89 THEN END
852 GOSUB 870 :: Z=0 :: DISP
LAY AT(21,3):"TOTAL VALUE: $
"
853 FOR X=1 TO T :: A$=SEG$(
S$(X),61,4):: Y=VAL(A$):: IF
 POS(A$,".",1)<>0 THEN Y=Y*1
00
854 Z=Z+Y :: DISPLAY AT(21,1
7):Z/100 :: NEXT X :: DISPLA
Y AT(24,3):"Press an% ke% to
 quit..."
855 CALL KEY(0,K,S):: IF S=0
 THEN 855 ELSE STOP
860 REM ** subs **
870 DISPLAY AT(20,1):ER$ ::
RETURN ! erase
880 A$=A$&RPT$(" ",S-LEN(A$)
)&"¦" :: RETURN ! pad: space
s + ¦
890 GOSUB 870 :: DISPLAY AT(
21,3):A$&" RECORD #     (or
0)" :: ACCEPT AT(21,16)SIZE(
2)BEEP:P :: IF P<0 OR P>T TH
EN 890 ELSE RETURN
900 A$=S$(N):: CALL DR(2,19,
A$,1,3):: CALL DR(4,13,A$,5,
15):: CALL DR(6,13,A$,21,7):
: CALL DR(9,11,A$,29,4)! dis
play a record
910 CALL DR(9,24,A$,34,4)::
CALL DR(11,14,A$,39,4):: CAL
L DR(13,14,A$,44,14):: CALL
DR(15,14,A$,59,1):: CALL DR(
17,14,A$,61,4):: RETURN
920 DISPLAY AT(2,2)SIZE(3):S
TR$(T)&"~~" :: RETURN ! show
 total
930 !@P+
940 SUB A(R,C,S,A$):: ACCEPT
 AT(R,C)SIZE(-S):A$ :: GOSUB
 880 :: SUBEND ! any charact
er
950 SUB AU(R,C,S,A$):: ACCEP
T AT(R,C)VALIDATE(UALPHA)SIZ
E(-S):A$ :: GOSUB 880 :: SUB
END ! uppercase only
960 SUB AN(R,C,S,A$):: ACCEP
T AT(R,C)VALIDATE(NUMERIC)SI
ZE(-S):A$ :: GOSUB 880 :: SU
BEND ! 0-9 & ".,+-E"
970 SUB AD(R,C,S,A$):: ACCEP
T AT(R,C)VALIDATE(DIGIT)SIZE
(-S):A$ :: GOSUB 880 :: SUBE
ND ! 0-9 only
980 SUB AS(R,C,S,A$,B$):: AC
CEPT AT(R,C)VALIDATE(B$)SIZE
(-S):A$ :: GOSUB 880 :: SUBE
ND ! only char. in string B$
990 SUB DR(R,C,A$,P,S):: DIS
PLAY AT(R,C)SIZE(S):SEG$(A$,
P,S):: SUBEND ! display a se
gment of string
```

■

SWEDLOW TI BITS * 22-23 *

by Jim Swedlow, USA

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

JIFFYFLIER

JIFFYFLIER is a new release from Roger Merritt, one of TI's graphics gurus. It is an elegant program that does its job quickly and simply. In other words, I liked it.

JIFFYFLIER makes one page fliers. It requires 32K, one disk drive, XB and an Epson compatible printer. It supports the color capabilities of the Star NX1000 Rainbow. Roger points out that you will also need "lots of paper to enjoy the output".

The flier can have a border, a graphic within the flier, one line of large text and up to 40 lines of small text. You have a choice of font for the small text. The large text line can be regular or inverted characters.

The distribution disk includes lots of borders and small text fonts. There are 55 graphics for inside the flier. You can also use other CSGD "/GR" graphics.

Choices are very simple - use the <SPACE BAR> to change something and <ENTER> to accept it as displayed. The flier is shown on your screen half a page at a time. Once you are done, you can save your flier to disk and edit it with TI Writer.

The documentation is straight forward. It is only three pages long but that is all that is needed as the program is simple to operate.

Printing is very fast due to a new screen dump Assembly routine by Robby Robinson.

At $11, JIFFYFLIER is an outstanding value. If you want a copy, send your order to (bank notes or money order, add $3 extra for postage, Ed.Note):

Roger Merritt
1949 Evergreen Avenue
Fullerton, CA 92635, USA

Be sure and mention where you heard about JIFFYFLIER.

HAVE YOU HEARD?

You can attach a 3 1/2 inch drive to a 4A? You are still limited to 40 tracks, however.

WORDCOUNT

This short program appeared in the PUNN Wordplay. It counts the number of words in a DV80 file.

```
 100 ! WORDCOUNT
 110 DISPLAY AT(12,1)ERASE
       ALL:"File: DSK" ::
     ACCEPT AT(12,10):F$ ::
     OPEN #1:"DSK"&F$,INPUT
 120 A=1 ::
     LINPUT #1:A$ ::
     IF ASC(A$)=46 THEN 140
 130 X=POS(A$," ",A) ::
     IF X=0 THEN 140 ::
     IF X=A THEN A=X+1 ::
     GOTO 130 ELSE
     F=1 :: C=C+1 ::
     A=X+1 :: GOTO 130
 140 C=C+F :: F=0 ::
     IF EOF(1)<>1 THEN 120 ELSE
     CLOSE #1 ::
     DISPLAY AT(14,1):"Has about";
       C;"Words"
```

This nifty program does its job but I though it could be improved a bit.

In line 110, F$ is used only for the file name. Later on, A$ is used for the line contents. Since we don't need to know the value of F$ and A$ at the same time, I changed F$ to A$. Each additional variable slightly slows program execution.

Line 140 has this code:

```
IF EOF(1)<>1 THEN 120 ELSE
CLOSE . . .
```

I changed it to:

```
IF EOF(1) THEN CLOSE . . .
ELSE 120
```

This eliminated the step of comparing EOF(1) to the number one. This works because EOF(1) returns 0 (or FALSE) if it is not the end of the file, 1 (or TRUE) if it is and -1 (also TRUE) if the disk is full. In an IF test, zero is FALSE and everything else is TRUE.

Line 120 could cause the program to bomb. ASC(A$) causes a fatal error when A$ is a null string. To prevent this, I added some code:

```
120 A=1 ::
    LINPUT #1:A$ ::
    IF A$="" THEN 140 ELSE
    IF ASC(A$)=46 OR ASC(A$)>127
      THEN 140
```

The test for Formatter commands (lines that begin with a period) is now performed after we have excluded null strings. Remember that ASC(A$) returns the ASCII value of the first character in A$.

This also eliminates lines that contain Editor tab and margin information (these lines start with a character that has an ASC value higher than 127).

In line 130, IF X=0 THEN 140 ELSE can be simplified. The same thing can be accomplished with IF X THEN. When X=0 execution will automatically transfer to line 140 (the next line). Also, it did not seem necessary to duplicate A=X+1. The line changed to:

```
130 A=X+1 ::
    X=POS(A$," ",A) ::
    IF X THEN IF X=A THEN 130 ELSE
    F=1 :: C=C+1 ::
    GOTO 130
```

The beginning of line 120 must be changed from A=1 to X=0.

The flag <F> in lines 130 and 140 bothered me. The program sets F as 1 every time it counts a word. The flag is used to make sure that the last word in the line is counted. I noticed that you could also tell if the last word needed to be counted if X=0 and A is less than LEN(A$).

Note that (A<LEN(A$)) returns 0 (or FALSE) if A is equal to or greater than LEN(A$) and -1 (or TRUE) if A is less than LEN(A$). The test and the addition can be combined into one formula:

```
140 C=C-(A<LEN(A$)) ::
    IF EOF(1) . . .
```

Line 130 changed to:

```
130 A=X+1 ::
    X=POS(A$," ",A) ::
    IF X THEN IF X=A THEN 130 ELSE
    C=C+1 :: GOTO 130
```

This suggested another change to line 130. IF X=A . . . could also be incorporated into a formula:

```
130 A=X+1  ::
    X=POS(A$," ",A) ::
    IF X THEN C=C-(X<>A) ::
    GOTO 130
```

To make sure that we don't test for a word at the end of a line when execution moves from line 120 to line 140, I moved C=C-(A<LEN(A$)) to the end of line 130.

The program is now leaner and faster:

```
100 ! WORDCOUNT
110 DISPLAY AT(12,1)ERASE
      ALL:"File: DSK" ::
    ACCEPT AT(12,10):A$ ::
    OPEN #1:"DSK"&A$,INPUT
120 X=0 ::
    LINPUT #1:A$ ::
    IF A$="" THEN 140 ELSE
    IF ASC(A$)=46 OR ASC(A$)>127
      THEN 140
130 A=X+1 ::
    X=POS(A$," ",A) ::
    IF X THEN C=C-(X<>A) ::
    GOTO 130 ELSE C=C-(A<LEN(A$))
140 IF EOF(1) THEN CLOSE #1 ::
    DISPLAY AT(14,1):"Has about";
      C;"Words" ELSE 120
```

A DIRTY DOZEN

Being a compendium of things small and not so small that are worth a word or two.

DOCS, Part 1

On disk documentation is a good way to tell you how to use a program. There are a number of ways to print them. The program may print the docs. You may have to print them through the Formatter. Other times there is a special program just to print the docs.

It would be nice if there was a README file that told you how to print the docs.

TI WRITER, Part 1

Sometimes, when you display a disk text (DV80) file on your screen, the last line looks like hieroglyphics. When you save a TI Writer file using SaveFile, the very last record contains the tab and margin settings. The characters are outside the ASCII 32 to 127 range, so they show on your screen as strange graphics.

When you save a file with PrintFile, the tabs and margins are not saved as TI Writer thinks that you are going to send your file to a printer. PrintFile, however, will accept any legal devise name (DSKn.FILENAME, PIO, RS232, etc).

If you are saving a file you will edit again or will run through the Formatter, use SaveFile. If you are saving a file some one else will view on disk, use PrintFile.

FAIRWARE, Part 1

People who use Fairware but don't send the author a contribution earn a place in the dirty dozen.

NEWSLETTERS, Part 1

If you ever read through all the various TI news letters that TI User Groups publish you will be awed by the vast array of information that they purvey. From original software to hardware fixes to reviews to commentary, the range is immense.

Why is this in the dirty dozen? Because you probably have never looked at all of these wonderful resources.

THIS AND THAT, Part 1

Another category in the dirty dozen is presentations at User Group meetings that are so complex, so techy that no one understands what is being said. Drives members away.

FAIRWARE, Part 2

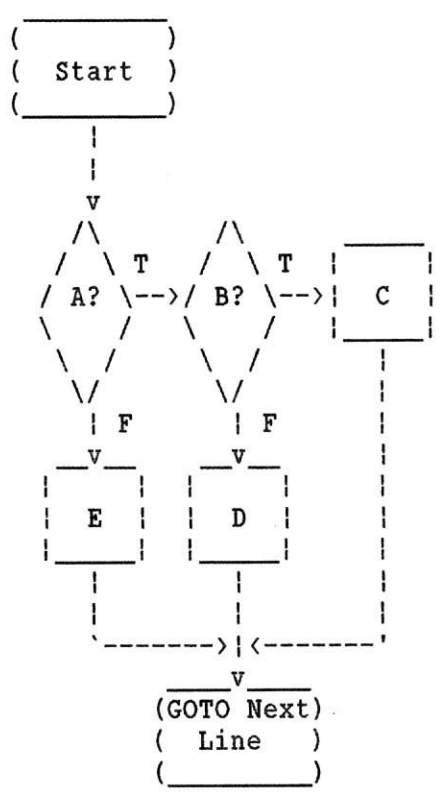
Right next to users who do not contribute are Fairware authors who receive our contributions but fail to acknowledge them. It is good manners to say thanks. Just taking the money discourages future support.

THIS AND THAT, Part 2

An IF THEN problem that many of us have is figuring out how to match IF's and ELSE's.

Reading from the start of the line, each ELSE matches the last unmatched IF. Consider the following:
IF A THEN IF B THEN C ELSE D ELSE E

This can be displayed as a flow chart:



DOCS, Part 2

It would also help if programmers had a novice user or two read their docs to make sure they tell you how to use the program.

NEWSLETTERS, Part 2

If you do get around to reading through old newsletters you will see articles that run something like this:

"Your officers are getting sick and tired of doing everything ourselves. If some of you don't start helping, things just aren't going to get done".

Folks, this does not solve the problem. Broad band appeals in newsletter articles almost always result in nothing. The only way to get folks to help is to ask them, one to one.

THIS AND THAT, Part 3

The PE Box fuses get special attention in the dirty dozen. TI put a 1.25 amp slow blow fuse outside the PE Box (the one you can get to). Inside the box (actually inside the main transformer) they put a one amp regular fuse. Guess which fuse goes when you get a short?

The following description comes from Newt Armstrong. COM stands for the common tap of the primary side of the power supply transformer.

```
:.....................................:
:The primary has taps for 100, 120, 220:
:and 240 volt input.  The FAN is before:
:the INTernal FUSE between the 120V tap:
:       240---+ and COM.   The 120V tap:
:             ( is the center  of  the:
:       220---+ primary.              :
:             ( .....................:
: EXT         ( One  fix  is  to  rewire:
:  F          ( the COM  and  FAN leads:
:  U          ( to the 240V tap and re-:
+--S--+-120---+ gain transformer.  Taps:
|  E  |       ( for  100V and 220V will:
|     | 100---+ not work, but  they are:
AC    |       ( not used anyhow.      :
IN   FAN   INT( .....................:
|     |     F ( The other fix is to dig:
|     |     U ( out  the fuse carefully:
+-----+-COM-S-+ and replace it.  Do not:
:           E ( take  the   transformer:
:             apart.  Strip  off  the:
:insulation.  Be very, very careful.  :
:.....................................:
```

Remember that fuses usually don't blow by themselves - often there is another problem. You can tell if the internal fuse is blown - the PE Box fan works but nothing else does.

TI WRITER, Part 2

When FIll and ADjust are on, TI Writer always adds two spaces after a period. This is good for a sentence but bad for names and abbreviations. Mr. Jones looks wrong.

There is an easy fix. Use the circumflex (^) as a required space. If you type this in the Editor:
Mr.^Jones

You will get this from the Formatter:
Mr. Jones

Makes a better final document.

CLOSING SHOT

One last item to fill this dirty dozen. The thing in the TI World that bothers me the most is when folks loose sight of our prime objective. The 4A will have strong support as long as we hang together and keep our eye on the ball. For us, the center issue is survival.

Enjoy.■

MUS FÖR RS232-PORT

En mus kan användas istället för en joystick i ritprogram. Den finns som bussmus eller seriemus. En buss-mus kan endast anslutas direkt till videoprocessorn V9938. En serie-mus anslutes till RS232 och fungerar med TI-99/4A som har disk, 32 kbytes EM och RS232. Asgard Software, P.O.Box 10306, Rockville, MD 20849, USA säljer en sådan mus för USD 40 + USD 7 porto. Horizon (Bud Mills Services) har en liknande mus för USD 40 + USD 5 porto. Båda innehåller drivrutiner för TI-Artist, YAPP och XB.■

HORIZON RAM-DISK FÖR 99/4A

av Jan Alexandersson

Horizon RAM-disk är ett stort RAM-minne på ett kort som placeras i expansionsboxen. Du kan anropa den precis som om det var en vanlig flexskiveenhet. RAM-disken kan lagra program eller datafiler, men kan även anropas med en sektoreditor (t.ex. Disk Patch eller Disk Review) och även användas för att lagra FORTH-skärmar. En vanlig disk manager (t.ex. DM 1000) kan användas för att kopiera filer till och från RAM-disken. Sektor 0 och 1 ser ut precis som på en vanlig flexskiva. Du kan köpa kortet som byggsats eller färdigbyggt. Det är möjligt att ha flera Horizon-kort i boxen samtidigt bara det finns lediga kortplatser och CRU-nummer.

CRU-ADRESSER

Horizon kan använda åtta olika CRU-adresser >1000 - >1700. Du måste välja ett unikt nummer som inte används av andra kort i din expansionsbox vilket normalt utesluter >1100 (diskkontrollkort) och >1300 (RS232/PIO). Det vanligaste är nog att välja >1000 eftersom du då har full valfrihet att välja DSK-nummer 1-9. På CRU >1200 och uppåt kan endast DSK 4-9 användas (DSK5-9 med Myarc och Corcomp diskkontrollkort). TI:s originalkontrollkort förutsätter alltid att det är det sista kortet med DSK-nummer så att sökning av högre CRU förhindras. Många äldre program kan då ej komma åt Horizon på höga CRU-nummer. Alla nya program (Funnelweb, DM1000 m.fl.) har en intelligent sökning så att CRU >1100 avsökes sist. Jag tror att man brukar börja med >1200 upp till >1F00 och sedan avsluta med >1000 och >1100. Som nybörjare bör du således alltid använda CRU >1000 tills du förstått hur det hänger ihop. Själv har jag Horizon på CRU >1600 utan att ha problem med de program jag använder. SAVE och OLD av BASIC-program fungerar bra oberoende av CRU-nummer.

BATTERIER

Horizon använder statiska RAM-chip och laddningsbara nickel-kadmium-batterier. När expansionsboxen är på så kommer batterierna att laddas. Detta betyder att hela operativ-systemet och alla lagrade filer finns kvar nästa gång du sätter igång datorn. Du kan till och med plocka ut kortet ur boxen och låta det ligga på hyllan några månader utan att något försvinner. Tänk alltid på att vänta 2 minuter efter det du stängt av boxen innan du stoppar i eller plockar ut kortet.

Batterierna till min 192 kbytes RAM-disk (från nov 1986) slutade fungera i sept 1992 så att jag blev tvungen att köpa nya. ELFA säljer NiCd-batterier med IEC-storlek R03 som är samma som de amerikanska AAA som beskrivs i manualen. De kostade 25 kr per styck + moms och porto, totalt 119 kr för alla tre. Batterierna höll således knappt 6 år vilket troligen är vad man normalt kan räkna med.

Du kan även välja litium-batterier eller alkaliska batterier men då måste du lägga in en extra diod som förhindrar uppladdning.

HISTORIA

Horizon började säljas i feb 1986. Originalversionen hade 64 kbit (8 kbytes) statiska RAM-chip som måste monteras i dubbla lager ovanpå varandra för att få maximi-storleken 192 kbytes (24 chips).

Under 1987 började Elektronik-Service i Tyskland sälja Horizon-kort med 392 kbytes. Man använde 32 kbytes chips och ett litet dotterkort för att ordna kodningen. Ett känt serienummer är 1216. Se foto i PB 88-4.

HRD+ kom i feb 1988 med 32 kbytes chips som gav möjlighet till Horizon med 384 kbytes (12 chips) utan att några RAM-chips behövde staplas på

varandra. Högsta serienummer 1900.

Horizon 2000 kom i sept 1988 och behöver ej ha kontrollchip (U2) staplade. Serienummer 2000-2200.

Horizon 3000 kom i feb 1989 och kan användas med 32, 128 eller 512 kbytes chips. Till att börja med användes dock endast 32 kbytes chips av ekonomiska skäl. Först i apr 1990 började 128 kbytes RAM-chips bli ekonomiska så att Horizon kunde tillverkas med 1536 kbytes (12 chips). Ändringarna #0 - #2 ingår som standard. Serienummer 3000-3399.

Horizon 3000B har även ändring #3 inbyggt. Serienummer 3400-.

Horizon 4000 kom i okt 1992 och har Rambo inbyggt som standard. Den använder nu statiska 512 kbytes RAM så att upp till 8 Mbytes får plats.

OPERATIVSYSTEM (ROS)

Det ursprungliga operativsystemet för RAM-disken tog 8 kbytes samtidigt som 4 kbytes reserverades för CALL och teckenset. En 192 kbytes RAM-disk kunde då användas för 720 sektorer.

ROS (Ramdisk Operating System) version 6 har tidigare beskrivits i PB 87-4. Den har flera allvarliga buggar som gör att den ej bör användas.

ROS 7.1 och ROS 7.3 har beskrivits i PB 88-1 och PB 88-2. Dessa fungerar helt tillfredsställande. Endast 8 kbytes har reserverats för operativsystemet så att en 192 kbytes Horizon kan använda 736 sektorer.

ROS 8.1 som kom i apr 1990 använder inte VDP-RAM som diskbuffert utan istället användes RAM på Horizonkortet. Det är möjligt att nu dela upp kortet i 10 olika DSK-nummer 1-9 och A-Z. Även om du har CALL FILES (1) så kan du ha 16 öppna filer på RAM-disken. SCRATCH RECORD ger dig möjlighet att ta bort en post mitt i en öppen datafil. ROS 8 säljs av Bud Mills för 10 dollar.

OPA säljer ett modifierat operativsystem version 9 i en PROM.

AUTO-START

Det är möjligt att definiera en av de filer du sparar på RAM-disken så att den startar automatiskt varje gång datorn slås på eller efter reset. Istället för att visa färgbalkarna startas filen. Jag har Funnelwebs FW-fil som sådan självstart.

NYA CALL OCH DSR

Det finns flera nya CALL som kan anropas från BASIC-program men endast från kommando-mode i Extended Basic. Det finns CALL AO, AF, WO, WF & DN.

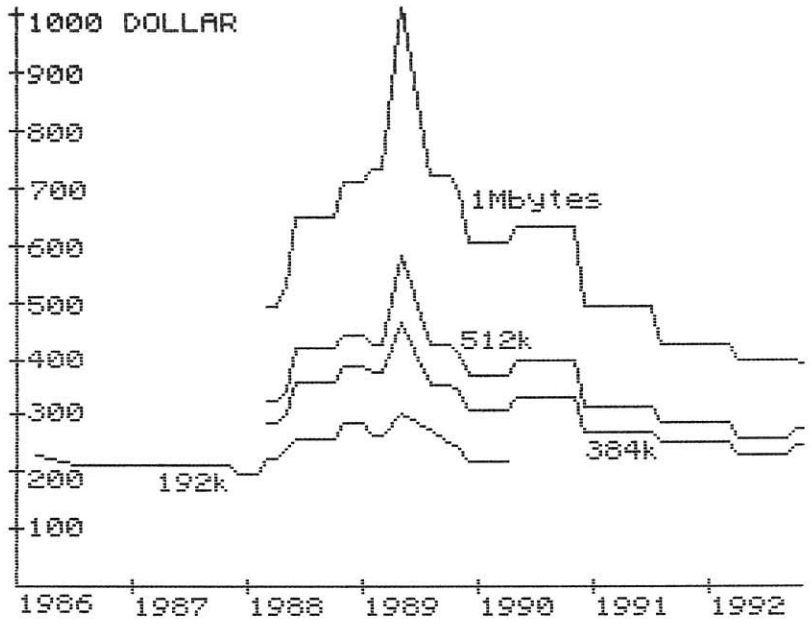
Du kan även låta upp till nio filer som finns på RAM-disken laddas med CALL. På detta sätt kan Funnelweb laddas med CALL FW och Disk Manager 1000 med CALL DM. Dessutom kan samma namn laddas som DSR så att programmet laddas med t.ex. DELETE "FW", OLD FW, OPEN #1:"FW". Du kan även använda dessa DSR direkt i ett annat program som har möjlighet att ladda eller spara filer. I TI-Writer Save File kan du skriva FW utan något DSK före.

MENU

Det finns ett menyprogram skrivet av J.Johnson som kan placeras på RAM-disken med auto-start och CALL. Det har en meny på 21 program som du själv enkelt kan ändra till just dina populäraste program. Det finns även ett antal funktioner som brukar finnas i en disk manager som skivkatalog, visa fil och köra program. Se även PB 89-4 som beskriver Miami BOOT och MENU.

PRISER PÅ RAMDISK

Horizon 192 kbytes RAM-disk kostade 210 dollar 1986. Sex år senare kan man få en 512 kbytes RAM-disk för 275 dollar. Lägg till 15 dollar till priserna i PB 92-3 för RAM-diskar upp t.o.m. 512 kbytes. 1 Mbytes kostar 385 dollar färdigbyggd. Prisutvecklingen visas i följande kurvor som ritats av ett Super-XB program till en TI-Artist bild. För större RAM-diskar tillkommer 110 dollar för varje ytterligare 512 kB.



EXPANSIONSMINNE 32 KBYTES

Du måste normalt ha kvar det vanliga 32 kbytes minneskortet som CPU-RAM. Det finns även möjlighet att montera detta minne på Horizon RAM-disk. Bud Mills säljer en byggsats för 25 dollar som kan monteras på alla gamla RAM-diskar oberoende av version. Den består av en statisk 32 kbytes minneskrets HM62256-ALP10 och en SN74LS08N samt två dioder och hållare för de två chipen.

Montagebeskrivning finns publicerad i Micropendium mars 89 och maj 89.

RANDOM ACCESS MEMORY BLOCK OPERATOR (RAMBO)

Rambo är ett litet dotterkort som monteras på Horizon RAM-disk. Det ger möjlighet att dela minnet i en del för RAM-disken och i en del för CPU-adresserbart minne som kopplas in i 8 kbytes bankar vid CPU-adress >6000. Se artikel från OPA i detta nummer av Programbiten och Micropendium dec 1989. Pris 45 dollar.

BYGG UT 192 KBYTES TILL 256 KBYTES

Denna ombyggnad gäller endast dig som har den ursprungliga RAM-disken med 8 kbytes RAM-chip och ett sammanlagt minne av 192 kbytes.

Ed Hallet publicerade i R/D Computing Ver 20 en ombyggnad med 8 kbytes chips som måste monteras i ett tredje lager på kortet.

Garry Bishop publicerade i Micropendium sept 89 och nov 89 en mycket bättre ombyggnad med 32 kbytes chips. Denna säljs av Bud Mills för 40 dollar.

ÄNDRING #0: LYSDIOD OCH STRÖMMATNING

RAM-diskar med serienummer upp till #0600 använder ofta endast röda lysdioder (LEDs). Den övre lysdioden bör vara grön eller gul för att säkerställa att U11 blir skyddad vid power-up.

Serienummer upp till #1200 kan ha för högt motstånd i serie med NiCd-batterierna. Detta bör vara totalt 66 ohm fördelat på 33 ohm (orange-orange-svart-guld) på respektive sida om batterierna för att ge tillräcklig laddning.

Exempel på RAM-diskar (övre LED):
192 kbytes #0694 gul LED 33+27 ohm
392 kbytes #1216 grön " 33+40 ohm

ÄNDRING #1: CONSOLE RESET LINE

Denna ändring ersätter en fördröjningskrets vid power-up genom att istället använda konsolens RESET

line. Detta gör också att RAM-disken kan få RESET varje gång konsolen får detta.

ÄNDRING #2: DISABLE SWITCH

En omkopplare placeras på kortets bakkant som gör det möjligt att gömma kortet från datorn vid power-up. Det kan annars inträffa att RAM-diskens operativsystem (ROS) blir delvis förstört på ett sådant sätt att datorn inte kan startas utan att den låser sig. Denna omkopplare är då avstängd vid starten av datorn och kan sedan sättas tillslagen innan du laddar in ny ROS. En byggbeskrivning finns publicerad i Micropendium aug 89. Observera att denna switch INTE användes vid normal start eller avstängning av TI-99/4A utan endast när man behöver kringgå problem med låsning eller liknande.

ÄNDRING #3: SKYDD VID POWER-DOWN

Denna ombyggnad gäller alla kort med 32 eller 128 kbytes minneskretsar. Den ursprungliga RAM-disken med 8 kbytes (totalt 192 kbytes disk) kretsar ska inte byggas om på detta sätt.

Denna bug märks inte med ROS version 7 men kan allvarligt störa ROS version 8. ROS 7 använder VDP-ram som diskbuffert medan ROS 8 istället använder RAM i Horizon-kortet som buffert (ungefär som Myarcs kontrollkort). Denna del av minnet i RAM-disken kommer att utsättas för slumpartade skrivningar vid Power-down. Med ROS 7 har detta minne inte använts medan ROS 8 använder det. Bud Mills säljer en byggsats och beskrivning av denna ändring. Det finns olika beskrivningar för HRD+, Horizon 2000 och Horizon 3000. Alla nyare RAM-diskar med serienummer #3400 och uppåt har denna ändring inbyggd från början, kallas även Horizon 3000B.

ÄNDRING #4: DIOD SKYDDAR POWER-UP

Denna ändring från juni 1992 gäller för nästan alla Horizon RAM-disk och P-GRAM(+). Den övre gröna lysdioden ska skydda minnet från åtkomst under power-up. Denna ska ha 2,3 V tändspänning men kan ibland vara 1,8 V. Ombyggnaden innebär att en signaldiod 1N914 (1,2 V) monteras i serie med lysdioden. Det är ingen fara att montera denna extra diod även om lysdioden är 2,3 V från början.

HORIZON REFERENSER:

PB 86-4 Bygg din egen RAM-disk
PB 87-1 RAM-diskbygget (bild)
PB 87-3 RAM-disk: Horizon 192k
PB 87-4 Menu version 6.4 och 392k RAM-disk
PB 88-1 Menu version 7.1
PB 88-2 Menu version 7.3
PB 88-4 Horizon RAM-disk operativsystem (bild av 392k)
PB 88-4 Bud Mills Services
PB 89-4 Miami BOOT och MENU
PB 90-2 Horizon 3000 RAM-disk

Micropendium:
Nov 85 p.18 Battery-Backed RAM disk
Feb 86 p.25 RAMdisk sells as kit or assembled
Jul 86 p.33 Horizon RAMdisk, review
Aug 87 p.39 Menu ver 6.3, review
Nov 87 p.51 Horizon lower prices
Feb 88 p.33 Horizon history outlined: Bud Mills (HRD+)
Jun 88 p.29 New Mills board reduces soldering (2000)
Jan 89 p.35 HRD 3000 RAMdisk now available
Mar 89 p.46 Optional 32k mod to replace 32k card
May 89 p.45 Correction on Horizon RAMdisk 32k mod
Aug 89 p.46 Disable switch for Horizon RAMdisk
Sep 89 p.30 An easier upgrade for your 192k Horizon
Nov 89 p.41 Addendum to expanding RAMdisk (192k to 256k)
Dec 89 p.41 8k banks accessible on Horizon RAMdisk (Rambo)
May 90 p.35 A Q&A on RAMdisks
Dec 90 p. 7 Rescuing a RAMdisk with corrupted ROS
Jan 91 p.34 Horizon 3000 Board kit with 128k chips
Feb 91 p.38 Horizon 128x8 RAM chips
Sep 91 p.25 Multiple columns for BOOT beyond 24
Apr 92 p.18 OPA: ROS_9 and Rambo
Jul 92 p.27 No more Horizon 3000 board
Oct 92 p.19 Horizon 4000 & ROS8.14b■

HORIZON ROS_9 AND RAMBO

HORIZON ROS_9 SERIES

When John Johnson released ROS_7, it soon became standard for HORIZONs, offering more features than before and was the most compatible problem-free ROS ever; but unforeseen problems and limitations were found as new devices like V9938 upgrades, hard drives, and larger Horizon RAMdisks appeared. A new series became imperative when OPA released RAMBO. A new operating system supporting all HORIZON models compatible with over thirty different devices was not an easy task, and many of the initial users had more problems, but now we are pleased to say with ROS_8.14, users report compatibility with all devices. But now OPA is offering ROS_9 in an Eprom only format which would replace the 8K RAM chip on the Horizon RAMdisk which allows all the features of ROS_8.14 plus allows no chance of corrupted disks and no chance of mixed versions, of ROS as all needed programs such as the ROS, ConFiG, menu and disk manager are contained in the EPROM.

The ROS_9 package includes:

* Complete User Operating Manual, covering CFG, ROS, MENU and RAMBO.

* ConFiGure program allows you to partition memory between 1 to 10 uniquely numbered/lettered RAMdisks & RAMBO space, change fonts, set colors, and setup custom calls.

* Ram-Operating System controls your Horizon to interface and look completely like a floppy drive to your system.

* Power-Up MENU program allows you on power-up and exit from any program to:- catalog disks, view and delete files, access custom calls, and run programs either from the 24-item changeable menu or directly from a catalog.

* OPA's Horizon TeSTer program, is a high-speed and thorough board tester for all HORIZONs. It detects almost all possible hardware failures in memory storage and support control.

C-900: $55.00 Canadian / $45.00 U.S.

NOTE: Those Horizon RAMdisk owners who wish to obtain ROS_8.14 should contact BUD MILLS SERVICES, 166 Dartmouth Dr., Toledo, OH 43614.

R.A.M.B.O. HORIZON UPGRADE KIT

RAMBO is a software and hardware package specially designed for HORIZON RAMdisks in your TI99/4A expansion box. The standard Horizon is designed mainly to be a RAMdisk, with >4000 based RAM (TI99 DSR space), a 6K DSR for ROS_8 and ROS_9, & 2K RAM pages to allow easy reading and writing of sectors. This is fine for RAMdisk operations, but using the Horizon memory for other applications was found by many software developers including OPA to be too difficult. But now, OPA in co-operation with BUD MILLS SERVICES has broken this barrier in the Horizon world by designing a combination of a 1"*1" PCBoard containing a complex PAL, and the ROS_8 and the new ROS_9, all easily installed in any Horizon RAMdisk model. Using ROS_8 or ROS_9, you can easily partition the on-board memory between RAMDISK and PROGRAM space. Combined use of this new PROGRAM space, a function of RAMBO, and ROS_8 or ROS_9, allows new, larger and more powerful programs to use the partitioned Horizon memory in a format better designed for direct Assembly use or execution. RAMBO comes with:-

* Step-By-Step User Installation Guide.

* An earlier version of OPA's RECALLIT 2001, a smaller un-marketed version.

* Two powerful and useful sample demo programs, plus its commented source code, and documentation.

Y-300: $55.00 Can. or $45.00 U.S. (includes ROS_8.14 disk version)

Y-301: $85.00 Can. or $70.00 U.S.
(includes ROS_9 on EPROM)

ORDERING INFORMATION

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

Shipping and Handling:
US/Canada $4.50; Other $7.50 ■

HORIZON FOR MYARC GENEVE

The Horizon Ramdisk with Geneve may be used as:

1. A BOOT DRIVE (up to 256 kbytes) that will allow SYSTEM/SYS to load very quickly, with enough room left to load GPL and some other files. RAMDOS must be on the BOOT DRIVE and run from the autoexec once to patch DOS to know where the Horizon is. Don't forget the ASSIGN statements in the autoexec also. Note that a larger size Ramdisk may be used as a BOOT DRIVE, but only 256 kbytes will be usable.

2. A RAMDRIVE (any size up to 800 kbytes). The Ramdrive will not allow booting since the Ramdrive only responds to a 16 bit address, this is where RAMDOS is important. The first time RAMDOS is run to patch DOS to locate the Horizon on the 8 bit address bus (the Ramdrive is still hidden). RAMDOS must be run again to toggle to the 16 bit address to see the Horizon Ramdrive.

3. A PHOENIX (a single Horizon board with 2 distinct memory blocks that respond as 1 and 2 above). The Phoenix is really two separate ramdisk memory blocks that share a common I/O access. Control of which Block or Drive is thru a counter that simply counts the 8 and 16 bit address string and routes the commands to the proper Drive. 8 bit address = Boot Drive, 16 = Ramdrive.

REFERENSER

PB 90-3 Horizon RAM-disk för Geneve

Micropendium:
Jan 88 p.27 Using the Horizon RAMdisk with Geneve
Mar 88 p.32 The Horizon RAMdisk meets the 9640(Phoenix)
Apr 88 p.32 RAMdisk and the 9640 ■

RÄNTEBERÄKNING – BASIC

```
100 CALL CHAR(35,"6666003C42
7E4242")
110 CALL CHAR(64,"1818003C42
7E4242")
120 CALL CHAR(94,"6666003C42
42423C")
130 CALL CLEAR
140 PRINT TAB(7);"R#NTEBER#K
NINGAR"
150 PRINT TAB(7);"==========
======": : : : : : : : : :
160 PRINT : : : : : : : : :
170 PRINT "            Bengt
 Karlsson"
180 PRINT "            (apri
l-maj 1983)"
190 FOR I=1 TO 900
200 NEXT I
210 CALL CLEAR
220 PRINT TAB(11);"M E N Y"
230 PRINT TAB(10);"=========
"
240 PRINT : : :
250 PRINT "1  -  SAMMANSATT
R#NTA": : :
260 PRINT "2  -  SAMMANSATT
R#NTA MED": : :"      @RLIG
INS#TTNING": : :
270 PRINT "3  -  ANNUITET":
: :
280 PRINT "4  -  AVSLUTA!"
290 PRINT : : :
300 PRINT "V#LJ RUBRIK GENOM
 ATT TRYCKAMOTSVARANDE SIFFR
A!"
310 GOTO 330
320 CALL SOUND(100,110,0)
330 CALL KEY(0,KEY,STATUS)
340 IF STATUS=0 THEN 330
350 IF (KEY<49)+(KEY>52)THEN
```

```
 320
360 IF KEY=49 THEN 400
370 IF KEY=50 THEN 760
380 IF KEY=51 THEN 1150
390 IF KEY=52 THEN 1530
400 CALL CLEAR
410 PRINT TAB(6);"SAMMANSATT
 R#NTA"
420 PRINT TAB(6);"==========
======"
430 PRINT : : :
440 PRINT "ANGE STARTKAPITAL
, R#NTESATS": :"(%), SAMT DE
N TID (@R,R#NTE-": :"PERIOD)
 SOM PENGARNA ST@R ": :
450 PRINT "INNE, S@ R#KNAR D
ATORN UT": :"(MED R#NTA P@ R
#NTA) HUR": :"STORT DITT SAM
MANLAGDA": :"KAPITAL BLIR.":
 : : : :
460 PRINT "TRYCK N@GON TANGE
NT!"
470 CALL KEY(0,KEY,STATUS)
480 IF STATUS=0 THEN 470
490 CALL CLEAR
500 PRINT "KAPITAL?": :
510 INPUT K
520 PRINT : : :
530 PRINT "R#NTESATS (%)?":
:
540 INPUT P
550 PRINT : : :
560 PRINT "TID?": :
570 INPUT T
580 PRINT : : :
590 TOT=K*(1+P/100)^T
600 TOT=INT(TOT*100)/100
610 CALL CLEAR
620 PRINT "STARTKAPITAL:";K:
 :"R#NTESATS:";P: :"TID:";T:
 : : : : : :
630 PRINT "SLUTKAPITAL:";TOT
640 PRINT : : : : : :
650 PRINT "VILL DU R#KNA P@
SAMMA      MOMENT IGEN - TRY
CK D@ 1.": :
660 PRINT "VILL DU STARTA FR
@N MENYN   IGEN - TRYCK D@ 2
."
670 CALL HCHAR(11,2,42,31)
680 CALL HCHAR(13,2,42,31)
690 GOTO 710
700 CALL SOUND(100,110,0)
710 CALL KEY(0,KEY,STATUS)
720 IF STATUS=0 THEN 710
730 IF (KEY<49)+(KEY>50)THEN
 700
740 IF KEY=50 THEN 210
750 IF KEY=49 THEN 400
760 CALL CLEAR
770 PRINT TAB(7);"SAMMANSATT
 R#NTA"
780 PRINT TAB(7);"==========
======": :
790 PRINT TAB(5);"MED @RLIG
INS#TTNING"
800 PRINT TAB(5);"==========
=========="
810 PRINT : : : : :
820 PRINT "DATORN R#KNAR UT
BEH@LLNING-": :"EN EFTER ETT
 VISST ANTAL @R,": :
830 PRINT "D@ ETT VISST BELO
PP S#TTS IN": :"I B^RJAN AV
VARJE @R."
840 PRINT : : : : :
850 PRINT TAB(5);"TRYCK N@GO
N TANGENT!"
860 CALL KEY(0,KEY,STATUS)
870 IF STATUS=0 THEN 860
880 CALL CLEAR
890 PRINT "@RLIG INS#TTNING:
": :
900 INPUT K
910 PRINT : : :
920 PRINT "R#NTESATS:": :
930 INPUT P
940 PRINT : : :
950 PRINT "TID:": :
960 INPUT T
970 PRINT : : :
980 TOT=K*((((1+P/100)^T)-1)
/(P/100))
990 TOT=INT(TOT*100)/100
1000 CALL CLEAR
1010 PRINT "@RLIG INS#TTNING
:";K: :"R#NTESATS:";P: :"TID
:";T
1020 PRINT : : : : : :
1030 PRINT "SLUTKAPITAL:";TO
T: : : : : : :
1040 PRINT "SAMMA AVSNITT? -
 TRYCK 1.": :
1050 PRINT "MENYN?          -
 TRYCK 2."
1060 CALL HCHAR(13,2,42,31)
1070 CALL HCHAR(15,2,42,31)
1080 GOTO 1100
1090 CALL SOUND(100,110,0)
1100 CALL KEY(0,KEY,STATUS)
1110 IF STATUS=0 THEN 1100
1120 IF (KEY<49)+(KEY>50)THE
N 1090
1130 IF KEY=49 THEN 760
1140 IF KEY=50 THEN 210
1150 CALL CLEAR
1160 PRINT TAB(10);"ANNUITET
"
1170 PRINT TAB(10);"========
"
1180 PRINT : : : : :
1190 PRINT "ETT L@N @TERBETA
LAS MED LIKASTORA BELOPP UND
ER ETT VISSTANTAL @R (F^RSTA
```

```
 G@NGEN"
1200 PRINT "EFTER 1 @R)."
1210 PRINT "ANGE L@NETS STOR
LEK, R#NTE- SATS OCH ANTAL @
R, S@ BER#K-NAR DATORN DET @
RLIGA BE-"
1220 PRINT "LOPPET SOM SKALL
 @TERBETALAS(ANNUITETEN)."
1230 PRINT : : : : :
1240 PRINT TAB(5);"TRYCK N@G
ON TANGENT!"
1250 CALL KEY(0,K,S)
1260 IF S=0 THEN 1250
1270 CALL CLEAR
1280 PRINT "ANGE L@NETS STOR
LEK!": :
1290 INPUT K
1300 PRINT : : :
1310 PRINT "ANGE R@NTESATS (
I %)!": :
1320 INPUT P
1330 PRINT : : :
1340 PRINT "ANGE @TERBETALNI
NGSTID (@R)!": :
1350 INPUT T
1360 PAR=(1+P/100)^T
1370 TOT=K*((PAR)*P/100)/((P
AR)-1)
1380 TOT=INT(TOT*100)/100
1390 CALL CLEAR
1400 PRINT "L@NETS STORLEK:"
;K: :"R#NTESATS:";P;"%": :"T
ID:";T;"@R": : : : : :
1410 PRINT "DET @RLIGA @TERB
ETALNINS-   BELOPPET (ANNUIT
ETEN) #R:":TOT: : : : : :
1420 PRINT "SAMMA AVSNITT -
TRYCK 1 !": :
1430 PRINT "MENYN?          -
TRYCK 2 !"
1440 CALL HCHAR(12,2,42,31)
1450 CALL HCHAR(16,2,42,31)
1460 GOTO 1480
1470 CALL SOUND(100,110,0)
1480 CALL KEY(0,K,S)
1490 IF S=0 THEN 1480
1500 IF (K<49)+(K>50)THEN 14
70
1510 IF K=49 THEN 1150
1520 IF K=50 THEN 210
1530 CALL CLEAR
1540 CALL SCREEN(2)
1550 PRINT TAB(11);"S L U T"
1560 PRINT TAB(11);"======="
1570 PRINT : : : : : : : : :
 : :
1580 CALL SCREEN(14)
1590 FOR DELAY=1 TO 1000
1600 NEXT DELAY
1610 CALL CLEAR
1620 CALL SCREEN(4)
1630 END                    ■
```

FROM BASIC TO ASSEMBLY - 1

by Bob August, Bug News, USA

I have been attending ROBBIE'S assembly class and thought you might like to learn with me. Assembly requires more programming than Basic but the speed of assembly makes up for the extra time. This month we will clear the screen, Write to the screen, hold it on the screen for 5 seconds and then quit. In Basic and Extended Basic, this is what you would write.

```
IN BASIC
100 CALL CLEAR
110 PRINT TAB(3);"MY FIRST T
RY AT ASSEMBLY"::::::::::::
120 FOR R1=1 TO 1500
130 NEXT R1
140 CALL CLEAR
150 END

IN EXTENDED BASIC.
100 DISPLAY AT(12,3)ERASE AL
L:"MY FIRST TRY AT ASSEMBLY"
110 FOR R1=1 TO 1200 :: NEXT
 R1
120 CALL CLEAR :: END
```

The first thing you will need is an Editor Assembly module or FunnelWeb or some other version of assembly language. The next is understand what the commands mean. Here are the commands we will use in our program and what they mean:

```
DEF  = Define
REF  = Reference
VSBW = Writes a single byte
       to VDP RAM
VMBW = Writes a multiple byte
       to VDP
BSS  = Block starting with Symbol
LWPI = Load Workspace pointer
       immediate
LI   = Load immediate
CLR  = Clear
BLWP = Branch and load workspace
```

pointer
INC = Increment by one
CI = Compare immediate
JLE = Jump if low or equal

Now we can look at our program and see what it is doing. Any time you use an * (asterisk) in your program at the beginning of a line, it is the same as REM. Also after your last instruction you can leave one space and then add a remark like you do in Extended Basic. The format is like this:

```
Label  Do what Where  Remarks
START  LWPI    WKRSP  Load workspace
* This whole line is a remark
```

We used a lot of them to help you understand the program.

The first thing we did was define the name we would use for the entry point of our program. This tells the computer where to start executing your program. You can use any name up to 6 characters in length. We used START.

The next thing we did was Reference what Utilities we would use. These are built into the assembler. We used VSBW and VMBW.

Next we give our message a label and tell the computer it is text. We use the apostrophe instead of Quotes. We could add more messages and give each one a different label and put them on the screen. We will do this next month.

Next we allocate one set of registers as our workspace. You need this to run your program. We could use >20 or 32, it really doesn't make any difference.

Next we use the EVEN command so we will start the program on the next first byte of the program. In this program we really didn't need it, but if our message was 25 characters long instead of 24, we would need it.

We now are at the start of our program where we load our workspace. This will load into >A000 of your 32K memory expansion as we did not use an absolute origin for our workspace. We could have used >20BA or >8300.

We now clear the screen by loading register one with >2000, clearing register zero and write a space character to the screen 767 times. The VSBW command writes the first byte of register one to the screen. To play with this, change the >2000 to >5800 and you will fill the screen with X's. We write this to screen location 0 to 767. In assembly you start with zero not one and we have 32 characters across the screen, not 28. 24 times 32 equals 768.

Next we write our message to the screen by loading the screen location in R0 and label address of our message in R1 and the number of characters to write in R2. We then tell the computer to put it on the screen with the BLWP at VMBR.

The next thing we do is a double FOR-NEXT loop to hold the message on the screen. Without this, you would not see your message as it would flash on and off to fast. To hold it on the screen longer change the CI R1,3 to a higher number. You will notice that we are counting up to 65,535 three time or up to 196,605. This will give you some idea of the speed of assembly.

The last thing we do is go back to our power up screen and quit. If we had not put the START after the END you would have to enter START as your program name. With it here we will auto start.

Now type in your program using the edit portion of your assembly. Save it to disk as DSK1.TRY/S and go to the assembler and assemble. Use DSK1.TRY/S as your source file name, DSK1.TRY/O as your Object file name, your printer name as List file name if you want a listing and R for Options. You must always use R for Options, but you can use C if you want the output in compressed form, L for a listing to your printer, S to include the symbol table in your listing and T if you wish your text to be listed in your printer listing.

Run the program with option #3 of the Editor assembler using DSK1.TRY/O and see the results. This is a very basic program and you should play with it to see what you can do.

Here is the Program:

```
***************
* First try at *
*   Assembly   *
***************

* Define name for start of program
*
       DEF  START
*
* Reference Utilities we will use
*
       REF  VSBW,VMBW
*
* Give message a label
*
MSG    TEXT 'MY FIRST TRY '
       TEXT 'AT ASSEMBLY'
*
* Create a workspace for program
*
WRKSP  BSS  32
*
* Make sure we start on an even byte
*
       EVEN
*
* Start of program
*
START  LWPI WRKSP     Load workspace
*
* Clear the screen
*
       LI   R1,>2000  R1=32
       CLR  R0       For R0=1 to 767
CLEAR  BLWP @VSBW     Print CHR$(32)
       INC  R0        R0=R0+1
       CI   R0,767    See if its 767
       JLE  CLEAR     Next R0
*
* Write message to screen
*
       LI   R0,356    Row 12 Col 5
       LI   R1,MSG    Message
       LI   R2,24     Message length
       BLWP @VMBW     Put on screen
*
* Hold it on screen for 5 seconds
*
       LI   R1,0   For R1=1 to 3
FLAG   LI   R0,0   For R0=1 to 65535
DELAY  INC  R0        R0=R0+1
       CI   R0,>FFFF  See if 65535
       JNE  DELAY     Next R0
       INC  R1        R1=R1+1
       CI   R1,3      See if three
       JLT  FLAG      Next R1
*
* End program with auto start
*
       BLWP @0        FCTN Quit
       END  START
```

Until next month this is Bob saying

HAPPY ASSEMBLING! ■

TERMITE – SPEL I BASIC

```
100 REM ***********
110 REM * TERMITE *
120 REM ***********
130 REM BY J.R.DEW
140 REM 99'ER 82-12
170 CALL CLEAR
180 RANDOMIZE
190 PRINT "PRESENTATING.....
....": : : : : : : : : :
210 Z$="h h h h h h h h h h
h h h h "
230 GOSUB 1830
240 FOR X=1 TO 11
250 PRINT Z$: :
260 NEXT X
270 FOR X=1 TO 5
280 GOSUB 1010
290 NEXT X
310 DEF RI(X)=INT(RND*X)+1
320 CALL CHAR(96,"FFFFFFFFFF
FFFFFF")
330 CALL CLEAR
340 CALL COLOR(9,12,1)
350 GOSUB 1250
360 CV=8
370 CH=6
380 D=3
390 FOR A=1 TO 15
400 GOSUB 1330
410 NEXT A
420 D=2
430 GOSUB 1330
440 GOSUB 1330
450 D=4
460 FOR A=1 TO 10
470 GOSUB 1330
480 NEXT A
490 PRINT TAB(10);"TERMITE":
 : :"WANT INSTRUCTIONS (Y/N)
?"
```

```
500 CV=CV-4
510 K$="YN"
520 GOSUB 1730
530 IF K=78 THEN 700
540 CALL CLEAR
550 PRINT TAB(5);"THIS IS A
GAME OF": : :TAB(10);"TERMIT
E": : :"IN WHICH YOU MUST GE
T YOUR  TERMITE THROUGH THE
WOOD"
560 PRINT "WITHOUT HITTING T
HE KNOTS.  YOU MAY USE THE A
RROW KEYS  TO GET FROM THE L
EFT TO THE RIGHT.": :
570 PRINT "PRESS ANY KEY"
580 CALL KEY(3,K,S)
590 IF S=0 THEN 580
600 CALL CLEAR
610 PRINT "WHEN YOU HIT A KN
OT, YOU":"MUST START OVER. I
N THE":"EASY VERSION, I WILL
 SHOW"
620 PRINT "YOU WHERE ALL THE
 KNOTS ARE BEFORE YOU START
OVER. IN":"THE TOUGH VERSION
, YOU HAVE TO USE YOUR HEAD"
630 PRINT : :"PRESS ANY KEY"
: :
640 FOR X=1 TO 4
650 PRINT Z$: :
660 NEXT X
670 CALL KEY(3,K,S)
680 GOSUB 1010
690 IF S=0 THEN 670
700 CALL CLEAR
710 PRINT "SELECT LEVEL OF P
LAY": : : :"    1-EASY": :"
   2-TOUGH": : : : : : : : :
720 K$="12"
730 GOSUB 1730
740 L=D
750 CALL CLEAR
760 GOSUB 1250
770 T=0
780 GOSUB 1050
790 GOSUB 1100
800 K$="EXDS"
810 GOSUB 1730
820 GOSUB 1330
830 IF CH=5 THEN 810
840 IF T>28 THEN 1900
850 CALL HCHAR(1,3,64,T)
860 IF SEG$(M$(CV-5),CH-5,1)
="#" THEN 880
870 GOTO 810
880 FOR X=990 TO 110 STEP -1
10
890 CALL SOUND(200,X,0)
900 CALL HCHAR(CV,CH,104)
910 CALL HCHAR(CV,CH,105)
920 CALL HCHAR(CV,CH,64)
930 NEXT X
940 T=T+1
950 IF L>1 THEN 970
960 GOSUB 1180
970 GOSUB 1250
980 GOSUB 1050
990 GOTO 800
1000 CALL HCHAR(CV,CH,104)
1010 CALL SOUND(50,880,2)
1020 CALL CHAR(104,J$)
1030 CALL CHAR(104,I$)
1040 RETURN
1050 CV=5+RI(9)
1060 CH=5
1070 GOSUB 1830
1080 GOSUB 1000
1090 RETURN
1100 FOR X=1 TO 9
1110 M$(X)="@@@@@@@@@@@@@@@@@
@@@@@"
1120 FOR Y=1 TO 3+L
1130 Z=RI(19)+1
1140 M$(X)=SEG$(M$(X),1,Z-1)
&"#"&SEG$(M$(X),Z+1,21)
1150 NEXT Y
1160 NEXT X
1170 RETURN
1180 FOR Y=2 TO 20
1190 FOR X=1 TO 9
1200 IF SEG$(M$(X),Y,1)="@"
THEN 1220
1210 CALL HCHAR(X+5,Y+5,64)
1220 NEXT X
1230 NEXT Y
1240 RETURN
1250 FOR X=6 TO 26
1260 CALL VCHAR(6,X,96,9)
1270 CALL HCHAR(5,X,59+X)
1280 NEXT X
1290 FOR X=6 TO 14
1300 CALL HCHAR(X,4,43+X)
1310 NEXT X
1320 RETURN
1330 REM MOVE
1340 IF (CV=14)+(D=2)=-2 THE
N 1520
1350 IF (CH=5)+(D<>3)=-2 THE
N 1520
1360 IF (CV=6)+(D=1)=-2 THEN
 1520
1370 CALL HCHAR(CV,CH,32)
1380 ON D GOTO 1390,1420,145
0,1490
1390 CV=CV-1
1400 GOSUB 1770
1410 GOTO 1540
1420 CV=CV+1
1430 GOSUB 1800
1440 GOTO 1540
1450 IF CH=27 THEN 1570
1460 CH=CH+1
1470 GOSUB 1830
1480 GOTO 1540
```

```
1490 CH=CH-1
1500 GOSUB 1860
1510 GOTO 1540
1520 CALL SOUND(200,-2,0)
1530 RETURN
1540 GOSUB 1000
1550 IF CH=26 THEN 1570
1560 RETURN
1570 REM WIN
1580 CALL CLEAR
1590 IF L=2 THEN 1660
1600 PRINT "YOU WON ROUND 1"
: : :"NOW LET'S MAKE IT TOUG
H!"
1610 PRINT : : :"PRESS ANY K
EY"
1620 CALL KEY(3,K,S)
1630 IF S=0 THEN 1620
1640 L=2
1650 GOTO 750
1660 PRINT : : : : :"YOU WIN
!!!": : :"I NOW DECLARE YOU"
:"CHIEF TERMITE!"
1670 END
1680 CALL KEY(3,K,S)
1685 @=RND
1690 IF S THEN 1720
1700 GOSUB 1000
1710 GOTO 1680
1720 RETURN
1730 GOSUB 1680
1740 D=POS(K$,CHR$(K),1)
1750 IF D=0 THEN 1730
1760 RETURN
1770 I$="81818181818181FF"
1780 J$="E7E7E7E7E7E7E7FF"
1790 GOTO 1880
1800 I$="FF81818181818181"
1810 J$="FFE7E7E7E7E7E7E7"
1820 GOTO 1880
1830 I$="FFC0C0C0C0C0C0FF"
1840 J$="FFFFFFC0C0FFFFFF"
1850 GOTO 1880
1860 I$="FF0101010101FF"
1870 J$="FFFFFF0101FFFFFF"
1880 CALL CHAR(104,I$)
1890 RETURN
1900 CALL CLEAR
1910 PRINT "YOU TOOK SO LONG
 THAT YOUR  TERMITE DIED OF
OLD AGE!!!!": : : : : : : :
: : :
```

■

MULTICOLOR FÖR ASSEMBLER

av Jan Alexandersson

MULTICOLOR FÖR TI-99/4A

Multicolor delar upp skärmen i rutor med 48 rader och 64 kolumner. Varje sådan ruta kan fyllas med en av 16 färger helt oberoende av andra rutor. Dessutom kan man ha upp till 32 olika sprites som lägges ovanpå grafikskärmen.

Du måste initiera videoprocessorns 8 olika register (#0-#7) för att kunna använda Multicolor på följande sätt:

```
#0  >00        binärt 0000 0000
#1  >E8        binärt 1110 1000
#2  värdet x >400 = Screen Table
#3  användes ej
#4  värdet x >800 = Pattern Table
#5  värdet x >80  = Sprite Attribute
#6  värdet x >800 = Sprite Pattern
#7  högra hexdelen= Screen Color
```

TABELL 99/4A	REG	BYTES	PLATSER
SCREEN	2	768	16
PATTERN	4	2048	8
SPRITE ATTRIBUTE	5	128	128
SPRITE PATTERN	6	1024	8
TOTALT		3968	4

9938 FÖR 80-KOLUMNSKORT

Med videoprocessorn 9938 ska alla VDP-register #0 - #7 initieras som för TI-99/4A ovan. Dessutom bör följande register initieras:

```
#8  >88  aktiverar musen
#9  >00  ger 60 Hz skärmväxling
#11 Extended Sprite Attribute Table
#14 värdet x >4000 är Base Address
```

PROGRAMMET MAGIC CRAYON

Det var ursprungligen meningen att programmet MAGIC CRAYON från 99'er 82-06 (ett ritprogram för MULTICOLOR MODE) skulle publiceras här men det fick inte plats och kommer troligen inte heller i kommande nummer av Programbiten. Istället kommer "From Basic to Assembly" att fortsätta. Du kan få en kopia av källkoden till MAGIC CRAYON om du skickar en flexskiva med returporto till mig. ■

BATMAN – MUSIK FÖR XB

```
1 REM  BATMAN
2 REM  ÖVERSATT FRÅN IBM PC
3 REM  AV N. JOHANSSON
4 REM  X-BASIC
100 CALL CLEAR :: DISPLAY AT
(9,12):"BATMAN"
110 T=100
120 FOR B=1 TO 4
130 FOR A=1 TO 2
140 CALL SOUND(T,1046.5,2)
150 NEXT A
160 FOR A=1 TO 2
170 CALL SOUND(T,987.77,2)
180 NEXT A
190 FOR A=1 TO 2
200 CALL SOUND(T,933.38,2)
210 NEXT A
220 FOR A=1 TO 2
230 CALL SOUND(T,987.77,2)
240 NEXT A
250 NEXT B
260 FOR B=1 TO 2
270 FOR A=1 TO 2
280 CALL SOUND(T,1396.9,2)
290 NEXT A
300 FOR A=1 TO 2
310 CALL SOUND(T,1318.5,2)
320 NEXT A
330 FOR A=1 TO 2
340 CALL SOUND(T,1246.6,2)
350 NEXT A
360 FOR A=1 TO 2
370 CALL SOUND(T,1318.5,2)
380 NEXT A
390 NEXT B
400 FOR B=1 TO 2
410 FOR A=1 TO 2
420 CALL SOUND(T,1046.5,2)
430 NEXT A
440 FOR A=1 TO 2
450 CALL SOUND(T,987.77,2)
460 NEXT A
470 FOR A=1 TO 2
480 CALL SOUND(T,933.38,2)
490 NEXT A
500 FOR A=1 TO 2
510 CALL SOUND(T,987.77,2)
520 NEXT A
530 NEXT B
540 FOR C=1 TO 2
550 FOR A=1 TO 2
560 CALL SOUND(T,1568,2)
570 NEXT A
580 FOR A=1 TO 2
590 CALL SOUND(T,1482,2)
600 NEXT A
610 FOR A=1 TO 2
620 CALL SOUND(T,1396,2)
630 NEXT A
640 FOR A=1 TO 2
650 CALL SOUND(T,1482,2)
660 NEXT A
670 FOR A=1 TO 2
680 CALL SOUND(T,1396.9,2)
690 NEXT A
700 FOR A=1 TO 2
710 CALL SOUND(T,1318.5,2)
720 NEXT A
730 FOR A=1 TO 2
740 CALL SOUND(T,1246.6,2)
750 NEXT A
760 CALL SOUND(T,1318.5,2)
770 CALL SOUND(T,1396.9,2)
780 FOR B=1 TO 2
790 FOR A=1 TO 2
800 CALL SOUND(T,1046.5,2)
810 NEXT A
820 FOR A=1 TO 2
830 CALL SOUND(T,987.77,2)
840 NEXT A
850 FOR A=1 TO 2
860 CALL SOUND(T,933.38,2)
870 NEXT A
880 FOR A=1 TO 2
890 CALL SOUND(T,987.77,2)
900 NEXT A
910 NEXT B
920 NEXT C
930 FOR A=1 TO 2
940 CALL SOUND(T,1586,2)
950 NEXT A
960 FOR A=1 TO 2
970 CALL SOUND(T,1482,2)
980 NEXT A
990 FOR A=1 TO 2
1000 CALL SOUND(T,1396,2)
1010 NEXT A
1020 FOR A=1 TO 2
1030 CALL SOUND(T,1482,2)
1040 NEXT A
1050 CALL SOUND(T*.9,1568,2)
1060 CALL SOUND(T*.4,2130,3)
1070 CALL SOUND(T,2130,9)
1080 FOR A=1 TO 1001
1090 NEXT A                    ■
```

```
1 ! DISPLAY, Bug News
100 C$="HELLO THERE, HOW ARE
 YOU TODAY"
110 P=14
120 CALL CLEAR
130 C$="      "&C$&"      "
140 FOR Z=1 TO LEN(C$)
150 DISPLAY AT(12,8):SEG$(C$
,Z+1,P)
160 NEXT Z
170 GOTO 140                   ■
```