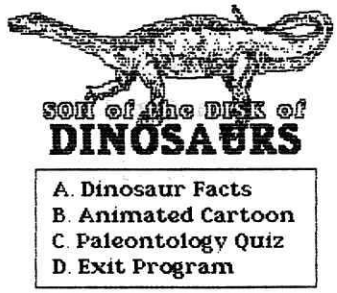
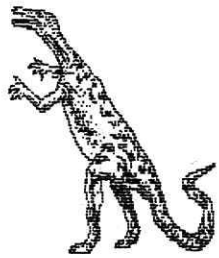
PROGRAM BITEN 91-4





ISSN 0281-1146

DATABAS FÖR CS1

Byt ut följande rader i programmet på sid 21-23 om du använder kassett CS1 (mata in "ZZZ" som sista post):

```
734 OPEN #1:"CS1",INPUT ,
    INTERNAL,FIXED 64
742 IF SEG$(R1$(IR),1,3)<>"ZZZ"
    THEN 738
746 IR=IR-1
752 OPEN #2:"CS1",OUTPUT,
    INTERNAL,FIXED 64
```

■

TURBO-PASC99 TUTOR

(Micropendium apr91p.31 & jun91p.6)

Det finns ett undervisande programpaket för Turbo-Pasc'99. Detta består av två vändbara skivor med Pascal-program och en utökad manual. Pris 12 dollar (räkna med c:a 15 dollar till Europa). Adress: Dan O'Quinn, Route 4 Box 565, Walterboro, SC 29488, USA. ■

GENIAL TRAVelER DISKAZINE

Det finns en tidning för 99:an som ges ut av Barry Traver i form av en flexskiva per nummer. Den består av ett stort antal program och artiklar. En årsprenumeration om sex nummer kostar 36 dollar (se Micropendium feb 1990 sid 38). Det är ej känt om den kostar extra till Europa. Adressen är: Genial Computerware, 835 Green Valley Dr., Philadelphia, PA 19128, USA. ■

NOTUNG SOFTWARE

Framsidan av PB 91-4 visar bilder från "Son of the Disk of Dinosaurs" av Ken Gilliland utgiven av Notung Software. Du kan beställa följande program från Notung Software (porto USD 2.50 per order till Europa tillkommer):

The Baba Brewery Beer Labels	USD	7
Certificate 99 Companion Plus	USD	7
Filmlib for TI-BASE	USD	7
Fonts and Borders I	USD	7
Fonts and Borders II	USD	7
Fonts and Borders III	USD	8
The Ring Companion	USD	8
Son of the Disk of Dinosaurs	USD	12
Star Trek:The Next Generation Calender	USD	10
Casino	USD	15

Notung Software, 7647 McGroarty Street, TUJUNGA, CA 91042, USA. ■

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

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

Postgiro 19 83 00-6
Medlemsavgiften för 1991 är 120:-

Datainspektionens licensnummer:
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. För lösblad (kopieras av annonsören) som skickas med tidningen gäller 200 kr per blad. Föreningen förbehåller sig rätten att avböja annonser som ej hör ihop med föreningens verksamhet eller ej på ett seriöst sätt gäller försäljning av originalexemplar av program.

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 årgång 84-89(styck)	50:-
1990	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.

```
100 ! BASSNOTES 1/3/85
110 CALL CLEAR
120 PRINT "PLAY AND PRINT FR
EQUENCY FOR 2 SUB-OCTIVES OF
 BASS NOTES": : :
130 PRINT " NOTE         FREQ
UENCY": :
140 DEF R(X)=INT(X+.5)
150 F=1654
160 FOR J=1 TO 25
170 READ N$
180 PRINT N$;TAB(8);"=";TAB(
15);INT(F/15)
190 CALL SOUND(500,30000,30,
3000,30,F,30,-4,0)
200 F=F/1.059463094
210 IF J<>12 THEN 230
220 RESTORE
230 NEXT J
240 DATA A,A FLAT,G,F#,F,E,E
 FLAT,D,C#,C,B,B FLAT,A
```

```
100 !BELLTONES
110 !BY E RAGUSE 5 8 88
120 CALL CLEAR
130 DISPLAY AT(12,6):"I PLAY
 BELL TONES"
140 PRINT "    ENTER NUMBER
1 TO 9  "
150 PRINT "     ANY OTHER TO
 QUIT"
160 CALL KEY(3,X,S):: IF S=0
 THEN 160
170 X=X-48 :: IF X<1 OR X>9
THEN END
180 CALL BELL(X)
190 GOTO 100
200 SUB BELL(N)
210 FOR I=1 TO N
220 FOR V=0 TO 25 STEP 2 ::
CALL SOUND(-999,1047,V,784,V
,587,V):: NEXT V
230 FOR V=26 TO 30 :: CALL S
OUND(-999,1047,V,784,V,587,V
):: NEXT V
240 NEXT I
250 SUBEND
```

```
100 !LOWSOUNDS
110 !By Earl Raguse
120 !Early Learning Fun
130 CALL CLEAR
140 PRINT "THIS IS A DISPLAY
 OF THE    COMPUTER'S ABILIT
Y TO PLAY  NOTES LOWER IN FR
EQUENCY    THAN THE SPECIFIE
D 110 HERTZ."
150 FOR I=1500 TO 110 STEP -
20
160 DISPLAY AT(12,10):"F= ";
INT(I/15)
170 CALL SOUND(500,22000,30,
22000,30,I,30,-4,0)
180 NEXT I
190 END
```

```
100 ! NOISE
110 ! BY E RAGUSE
120 ! 4 5 87
130 CALL CLEAR
140 DISPLAY AT(12,12):"NOISE
"
150 DISPLAY AT(14,1):"Plays
the 8 kinds of noise  availa
ble.  Input the third tone f
requency for the -4   and -8
 noises."
160 DISPLAY AT(20,7):"FREQUE
NCY? 400"
170 ACCEPT AT(20,18)VALIDATE
(NUMERIC)SIZE(-4):FREQ
180 FOR N=-1 TO -8 STEP -1
190 CALL SOUND(2000,110,30,1
10,30,FREQ,30,N,0)
200 DISPLAY AT(12,6)ERASE AL
L:"NOISE NUMBER ";N
210 IF (N=-4 OR N=-8)THEN 22
0 ELSE 230
220 DISPLAY AT(14,8):"FREQUE
NCY ";INT(FREQ/15)
230 NEXT N
240 FOR T=1 TO 500 :: NEXT T
250 DISPLAY AT(12,9)ERASE AL
L:"NICE HUH?  AGAIN, PRESS <
A>": : : :"           ELSE":
260 DISPLAY AT(22,7):"PRESS
ANY KEY"
270 CALL KEY(0,K,S):: IF S=0
 THEN 270 ELSE IF CHR$(K)="A
" THEN 100
280 END
```

```
100 !SOUNDTEST
110 CALL CLEAR :: DISPLAY AT
(6,8):"LOW SOUND TEST"
120 V=8
130 FOR I=1 TO 5
140 CALL SOUND(500,659,V,784
,V,981,30,-4,0)
150 CALL SOUND(500,440,V,523
,V,1310,30,-4,0)
160 CALL SOUND(500,494,V,587
,V,1470,30,-4,0)
170 CALL SOUND(500,523,V,659
,V,8250,30,-4,0)
180 FOR J=1 TO 4
190 READ N
200 CALL SOUND(150,110,30,11
0,30,N,30,-4,0)
210 NEXT J
220 RESTORE
230 NEXT I
240 DATA 1470,1310,1236,1101
250 END
```

■

SWEDLOW TI BITS * 5 - 7 *

by Jim Swedlow, USA

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

DISPLAY VARIABLE 80 FILES * MULTIPLAN AND TI WRITER

The DV80 file is TI's workhorse. TI Writer uses this format. If you open a file without specifing a type <OPEN #1:"DSK1.MYFILE">, it will be DV80. Assembly language source code files are DV80. This month we will cover some interesting aspects of these files as they are used by TI Writer and Multiplan.

First, you can save a Multiplan spreadsheet as a DV80 file on disk. Then, later, you can use that DV80 file for printing or for merging into a TI Writer file. You choose Print and then File. You must be careful to use a different file name than the one you used to save your spreadsheet as, unlike Transfer Save, Multiplan does not warn you if you are about to overwrite an existing file.

Just as when printing on a printer, you can control the margins and page format with Print Margin. One of the items that Print Margin let you set is Print Width. If you set this to a number greater than 80, Multiplan will write a DV80 file wherein each record is longer than 80 characters.

Should you attempt to read such a file with a BASIC program, your system will produce a strange error code and lock up. Apparently the folks at TI thought that a DV80 file couldn't have a record longer than 80 characters so their error handling language does not consider that possibility.

TI Writer, however, will read this illegal file. It will only input the first 80 characters in each record but it is just about the only way to access the file (another is a disk sector editor).

Incidently, TI Writer is very forgiving when reading files. More than once I have used TI Writer on a file with a glitch that prevented me from reading it. First I loaded the file into the Text Editor and then I saved the file back to disk. This process can remove a glitch.

QUOTES OF THE MONTH

For those who like this kind of a book, this is the kind of a book they will like.
---A book review by A. Lincoln

Knowledge comes but wisdom lingers.
---Tennyson

TWO TI WRITER TIPS

The Formatter makes sure that you have two spaces after each period. This can cause such strange things as:

Mr. Smith
1023 N. Fargo Street

These extra spaces jump off the page to the reader as simply wrong. The easiest way I have found to solve this is to use the ^ sign to control the spacing. Mr.^Smith will print with just one space as will 1023 N.^Fargo Street.

The other tip concerns writing a on-disk note using the Editor. You might be writing some documentation or a disk-letter. If you save your final version to disk using Save File, the last record will contain the margin and tab information and will cause the final print line to have strange characters. The solution is to save your final version using Print File. Just enter the Disk File name and your on disk file will have only your text.

WORD OF THE MONTH

AVATAR - the descent of a deity to

earth in an incarnate form; the incarnation of a god; any embodiment or manifestation of an abstract quality, attitude, concept or principle in a person.

"[His] piano teacher . . . was reportedly an avatar of the romantic giants of the 19th century."

A TI RESOURCE

Looking for a program to do something but you can't find one that meets you needs? The Amnion Helpline Free Access Library is one of the largest public domain collections for the TI.

FORMATTING DISK TEXT FILES

This month we will explore further into using TI Writer and disk files as output. Two simple utility programs accompany this article.

First, a bit about what the Text Formatter does. If you include the command ".FI;AD", the Formatter will right justify your text (so both the right and left columns are straight lines). When you save a file to disk from the Editor, however, you have a "ragged right" (or not right justified). If you want right justification on disk (and to use the other features of the Formatter), all you do is specify a disk file name as the Print Devicename in the Formatter.

There is a small hitch. Each and every line in the disk file will end in a line feed <CHR$(10)>. Then if you print that file without adding ".LF" to the printer name, your text will be double spaced. It will even be stranger if you use underlining and bold face.

The reason is that the Formatter expects to output to a printer. Since line feed and carriage return are about the only two universal printer command codes, the folks who wrote TI Writer had to come up with a way to do bold face and underline using only those two commands.

Here is what they did. Most printers will advance the print one line when they receive a line feed and return the print head to the left column when fed a carriage return.

To underline a word, print the line, execute a carriage return (so that the print head goes back to the beginning of the same line) and print underline characters (FCTN U) under the word to be underlined. Then send line feed and a carriage return and start the next line. Bold face is similar except that TI Writer prints the bold face word four times.

You add ".LF" to the printer name in the Formatter so that TI Writer can control when line feeds are sent. All of this is fine for a printer but not for a disk file.

If you are going to save your formatted text to disk, first do NOT use either bold face or underline. After you have run it thru the Formatter, you must load the formatted file into the Editor and then save it back to disk. Why? Well, if a line has 80 characters, the Formatter will add an LF to the end making it 81 characters long. Then when a basic program attempts to read that line, it will lock your system up. By loading and saving thru the editor, all lines are trimmed if they are over 80 characters long. Be sure and use Print File to save the file so that the Editor will not add the tabs (see last month's column).

Then use the program LF STRIPPER (elsewhere in this issue) to strip the line feeds from the ends of the lines.

QUOTES OF THE MONTH

"The new phenomena [atomic energy] would also lead to the construction of bombs. . . . A single bomb of this type, carried by boat and exploded in a port, might very well destroy the whole port, together with some of the surrounding territory. However, such bombs might very well prove to be too heavy for transportation by air."

---Albert Einstein (1879-1955)
from an August, 1939 letter
to President Roosevelt

"Computers are charting a new course in human history from the age of the muscle to the age of the mind."
---Author unknown

CARRIAGE RETURNS

Sometimes when you load a text file into the Text Editor there are no carriage returns at the end of the paragraphs. This can cause some serious problems. With TI Writer, if you Reformat or Replace String, you will find all of your paragraphs merged into one huge one (FUNEL-WRITER won't do this).

The other program this month, CR ADDER, will add carriage returns at the end all paragraphs and to all blank lines. It also adds a carriage return to the end of lines that start with a period as they are probably Text Formatter commands.

A note about this program. One thing I had to resolve was how to add a carriage return to a line that was already 80 characters long. After a bit of experimenting, I came up with this (assuming that A$ is the line and C$ is CHR$(13), the carriage return):

```
PRINT #2:A$;C$
```

Just as in printing to a printer, the semi-colon will ensure that the on disk file is properly set up.

However, I could have used this code:

```
PRINT #2:A$&C$
```

This works because the disk controller automatically breaks strings that are longer than the specified record length into record length pieces.

WORD OF THE MONTH

BLATHERSKITE: A person given to voluble, blustery, empty talk; a talkative, foolish person.

"Educators accuse politiciains of being blatherskites, compromisers and opportunists. In turn, politicians see educators as stuffy, sanctimonious prigs who are out of touch with reality."

A TI-WRITER TIP

If you want more than one word in a row to be bold face you can start the first word with the "at" sign (SHIFT 2) and then connect the following words with the exponentiation or circumflex or required space (SHIFT 6). This is fine except that the Text Formatter will see all of them as one huge word and you may have some strange spacing if you are right justifing.

Another way is to place the "at" sign before each word you want in bold face. This also works with underlining if you do not want spaces between words underlined.

```
*****************************
* programlistning för       *
* CR ADDER och LF STRIPPER  *
* se PB 89-1 sid 16-17      *
*****************************
```

IF THEN IN XB

A number of the XB columns discussed alternatives to IF THEN. Here is another. Suppose that A$ depends on the value of I. You might use this code:

```
IF I=1 THEN A$="FRED" ELSE A$="PAUL"
```

A simpler way is to use the SEG$ function:

```
A$=SEG$("PAULFRED",1-4*(I=1),4)
```

Will this work if the two variables have different lengths? Yes. Remember that SEG$ does not produce an error if the length of the new string (the last number) is longer than the source string. If our two names are "PAUL" and "SAM", this works:

```
A$=SEG$("PAULSAM",1-4*(I=1),4)
```

QUOTES OF THE MONTH

"I shall never believe that God plays dice with the world."
---Albert Einstein (1879-1955)

"Genius is one percent inspiration and ninetynine percent perspiration."
---Thomas A. Edison (1847-1931)
Attributed

A SAD TALE

A TI owner in New Jersey wrote that a thunder storm had destroyed his 4A and expansion system. I asked him if he used a surge suppressor. He responded:

"I am and always have been a strong advocate of the surge protector.

"All my equipment was plugged into a surge suppressor containing an on/off switch. The switch was in the off position. The surge protector was plugged into a wall outlet which is switch controlled. This switch was also in the off position.

"In my opinion, the air was highly charged with static electricity. Via some strange method of conduction, the charge dissipated to the chips. I have never heard of such occurrences. While discussing it with a few neighbors, I learned of VCR's and TV's being damaged during that same storm, none of which were in use at the time.

"How do you protect against this? My friends in the electronic field tell me that you can't. We discussed possibilities such as a "grounded" station but, since most computers have a plastic case, this is not practical (unless you want to open the case and attach a ground lead to the mother board or another central ground).

"When considering the odds against this, it is not worth the effort. Had I known that Murphy's Law would select me, I would have tried it. Too late now."

WORD OF THE MONTH

CURMUDGEON - an irascible, churlish person; a surly, ill-mannered, bad-tempered fellow; a cantankerous person

"He [Howard Hughes] had been a genius and curmudgeon. Red-baiter and environmentalist, wastrel and philanthropist."
---T. Mathews, "The Secret World of Howard Hughes," Newsweek, April 19, 1973

ERROR TRAPPING AFTER RUN

The following will NOT work:

```
100 ON ERROR 200
110 RUN "DSK1.TEST"

. . . program continues

200 PRINT "TEST PROGRAM NOT FOUND"
210 PRINT "INSERT DISK IN DRIVE ONE"
220 PRINT "AND PRESS ANY KEY"
230 CALL KEY(0,K,S) ::
    IF S<1 THEN 230
240 RETURN 100
```

If you run this program without a program called TEST in drive one, lines 200 thru 220 will print their message and then your program will fail in line 230 with 'SYNTAX ERROR IN 230'.

Why? As near as I can tell, when your TI executes the RUN part of RUN "DSK1.TEST", it clears memory including the variable table (think of it as 'un-pre-scanning' the program). The program and the line number table, however, remain.

When your TI tries to execute line 230, it looks for the variable table to find out, where the values for S and K are stored. Finding none, the CPU decides that an error condition exists and ends the program.

What to do? One way is to add a disk directory reading routine to find out if the desired program is on the disk. This will significantly increase the time it takes to load the program, however.

Another way is to use RUN. This will recreate the variable table.

ASSEMBLER I PROGRAMFORMAT

av Jan Alexandersson

När du assemblerar källkoden får du ditt program i INT/FIX 80 format. Det finns möjlighet att omvandla denna INT/FIX 80 till PROGRAM. Ett program i PROGRAM-format tar mindre plats på flexskivan och laddas snabbare. Det finns tre olika program som kan hjälpa till med omvandlingen från INT/FIX 80 till PROGRAM:

- SAVE som finns på skivorna till EA-modulen.

- FSAVE som följer med Funnelweb.

- LINKER av R.A.Green (finns i programbanken).

SAVE UTILITY TILL EA-MODULEN

Detta program finns beskrivet i manualen på sid 420. Använd som exempel objektskoden till BUBBLE-programmet BUBBLE/O som finns i PB 91-2. Du följer nedanstående punkter för att göra omvandlingen:

1. DEF SFIRST,SLAST,SLOAD först i källkoden.

2. Början av programmet skall ha label SFIRST och SLOAD.

3. Början måste ha en körbar instruktion som kan ordnas med JMP (kort hopp) eller B (långt hopp) till den verkliga startpunkten.

4. Slutet måste ha label SLAST före END.

5. Välj LOAD and RUN.

6. Ladda ditt objektskodsprogram t.ex. DSK2.BUBBLE/O.

7. Ladda DSK2.SAVE.

8. Tryck ENTER för att fortsätta.

9. Startadress SAVE.

10. Följ instruktionerna på skärmen och spara som t.ex. DSK2.BUBBLE.

11. Om programmet är längre än >2000 bytes, kommer SAVE att dela upp i flera delar på vardera 8 kbytes. Sista tecknet stegas en position för varje block (t.ex. BUBBLE, BUBBLF, BUBBLG o.s.v).

12. Välj EA-menyn nr 5 RUN PROGRAM av DSK2.BUBBLE.

13. Du kan starta samma PROGRAM från många olika PROGRAM-laddare t.ex. Funnelweb, TI-Writer och Editor/ Assembler.

FSAVE UTILITY TILL FUNNELWEB

Du kan använda FSAVE i Funnelweb på samma sätt som EA SAVE. Filen heter FSAVE och startadressen SAVE. Se även Tony McGoverns beskrivning i FWDOC/UTIL. Gör omvandling till PROGRAM med FSAVE och spara med filnamnet BUBBLE/FWB.

Använd en sektoreditor (t.ex. FW Disk Review) för att titta på den första datasektorn för BUBBLE/FWB och jämför med vad som finns med EA SAVE ovan. De tre första orden blir:

EA SAVE >0000 0300 A000

Since you can specify the line number where program execution starts, you can control what happens. This works:

```
200 RUN 210
210 PRINT "TEST PROGRAM NOT FOUND"
220 PRINT "INSERT DISK IN DRIVE ONE"
230 PRINT "AND PRESS ANY KEY"
240 CALL KEY(0,K,S) ::
    IF S<1 THEN 240
250 GOTO 100
```

What about pre-scan? Even though you specified that program execution started at line 210, the entire program is pre-scanned. You must be careful, however, to return to a point in the program that will assure that all necessary variable initalization is completed.

Enjoy. ■

```
FSAVE      >0000 02FA A000
```

Första ordet berättar att någon ytterligare fil inte ska laddas. Andra ordet är programmets längd i bytes. EA SAVE räknar med de inledande 6 byten i längden medan FSAVE endast visar den riktiga längden som motsvarar det RAM-minne som behövs. Det tredje ordet visar var i RAM-minnet som programmet ska laddas, nämligen >A000 i det höga (24 kbytes) expansionsminnet.

RAG LINKER

Innan du börjar använda LINKER bör du konfigurera den för dina egna behov genom att använda programmet LNKINST. Jag ändrade bl.a. "Default Options" till ingenting (tryck ENTER) så att skrivaren inte blir aktiverad i onödan och "Return" till BLWP (mitt system med AVPC och HFDC fungerar inte med RT). LINKER är inte beroende av att SFIRST, SLOAD och SLAST finns i programmet. Den förutsätter dock att början av programmet har en körbar instruktion som finns i vårt exempel med BUBBLE/O. Nu är det dags att starta LINKER. Välj CONTROL DSK2.BUBBLE/O, Tryck ENTER för PRINTER och LIBRARY. Välj OUTPUT DSK2.BUBBLE/RAG och tryck ENTER för OPTIONS.

LINKER har även en annan användningsmöjlighet om du har objektskoden till ett program som du inte själv har skrivit. Du kan trots detta omvandla DIS/FIX 80 till PROGRAM genom att peka ut startadressen och genom att lösa ut REF till de BLWP-rutiner som EA-laddaren klarar av. Om programmet är välskrivet så att det initierar alla VDP-register och VDP-RAM så brukar detta räcka för att detta nya PROGRAM ska kunna köras oberoende av vilken modul som används. Som ett dåligt exempel använder vi här BUBBLE-programmet i PB 91-2 sid 12 högra spalten som har REF VMBW o.s.v. Objektskoden antas ha sparats som BUBBLE/OEA. Skriv in följande korta kontrollfil som DIS/VAR 80 och spara den som BUBBLE/L1:

```
* SKAPAR BUBBLE/R1+R2
LOAD DSK*.BUBBLE/OEA
LIBRARY DSK1.RAGLIB
ENTRY BUBBLE
```

Observera att biblioteket endast laddar in de REF som verkligen används. Välj CONTROL DSK2.BUBBLE/L1, ENTER för LIBRARY och OUTPUT DSK2.BUBBLE/R1. LINKER skapar nu två sammanhörande filer BUBBLE/R1 och BUBBLE/R2. Programmet kan köras som BUBBLE/R1 med EA eller FW.

Du kan ordna så att det endast blir en PROGRAM-fil med följande kontrollfil(körbar med EA eller FW) som sparas som BUBBLE/LA:

```
* SKAPAR BUBBLE/RA
BLOCK 1,>A004,>5FFC
LOAD DSK*.BUBBLE/OEA
LIBRARY DSK1.RAGLIB
PATCH >A000,>0460
PATCH >A002,BUBBLE
ENTRY >A000
```

Välj CONTROL DSK2.BUBBLE/LA och OUTPUT DSK2.BUBBLE/RA.

För att få BUBBLE/OEA körbar med godtycklig modul måste vi först köra INIT-programmet i PB 91-2 med följande källkod som sparas under namnet INIT/S (och assembleras till INIT/O). källkoden till INIT måste även finnas:

```
        DEF  START
        REF  BUBBLE
        REF  VMBW,VSBW,VWTR,GPLLNK
START   LWPI REG
        BL   @INIT
        B    @BUBBLE
REG     BSS  >20
        COPY "DSK2.INIT"
        END
```

Skriv in följande kontrollfil BUBBLE/LB för RAG LINKER och omvandla till PROGRAM-fil BUBBLE/RB:

```
* SKAPAR BUBBLE/RB
LOAD DSK*.INIT/O
LOAD DSK*.BUBBLE/OEA
LIBRARY DSK1.RAGLIB
```

Denna fil BUBBLE/RB kan nu köras med godtycklig modul. Om du själv har skrivit programmet behöver du naturligtvis inte krångla till det på detta sätt. Skriv istället ditt program från början som BUBBLE/S i PB 91-2. ■

RAG LINKER VERSION 3

by R.A. Green, Canada

INTRODUCTION

The TI 99/4A LINKER is a tool for building assembler language memory image programs from tagged object. It makes this process simple and straight forward. LINKER's main features are:

1. tagged object modules may be compressed and/or uncompressed,

2. tagged object modules may be absolute or relocatable,

3. a library search can be done to resolve REFs,

4. a listing can be produced containing object information, memory maps and REF/DEF cross-references,

5. the memory image program can be built to load anywhere in the TI 99/4A's 64K address space.

The LINKER has three modes of operation depending upon the type of input in its control file.

1. Automatic. A single tagged object file is processed, a single library file is searched if necessary and a memory image program is produced. The memory image program will load as if the tagged object had been loaded by the E/A loader, excluding the E/A routines in low memory. That is, the object is loaded first in the 24K high memory block, then into the 8K low memory block.

2. Semi-automatic. A control file is processed which names one or more tagged object files to be processed and names one or more libraries to be searched. The resulting memory image program will load as if the tagged object had been loaded by the E/A loader, excluding the E/A routines in low memory. That is, the object is loaded first in the 24K high memory block, then into the 8K low memory block.

3. Complete Programmer Control. A control file is processed which names one or more tagged object files to be processed, names one or more libraries to be searched and which designates the memory layout for the resulting memory image program.

DISK CONTENTS

The distribution disk is a single sided single density disk containing the following files.

LINKER	The linker program in memory image form.
LINKES	The second segment of LINKER
LNKDOC	The documentation to be printed by the TI WRITER formatter.
LNKDOC1	Continuation of the doc.
LNKDOC2	Continuation of the doc.
LNKDOC3	Continuation of the doc.
LNKINST	Installation program
LOAD	XB-loader for LINKER
LOADINST	XB-loader for LNKINST
RAGLIB	A library of routines similar to those provided by the E/A, MM or XB.

FAIRWARE

This package is being made available via the Fairware concept. If you like the package and are using it, send a donation to:

R. A. Green
1032 Chantenay Drive
Gloucester, Ont. Canada
K1C 2K9

And, at the same time, distribute complete copies of the package to your friends. If you have any suggestions for improvements or have found any bugs please forward them to the above address.

MEMORY IMAGE PROGRAMS

Memory image programs are just that. They consist of the exact contents of memory written to a disk. A

memory image program may be split into two or more segments depending upon the length of the program. Each of the segments is written to a separate disk file. Option 3 of TI WRITER will load a segment with a maximum size of >2000 while Option 5 of Editor/Assembler will load a segment with maximum size of >2400 bytes. LINKER produces segments with a maximum length of >1FFA. The name of the first segment is specified and the names for all following segments is derived by adding 1 to the last byte of the name of the previous segment. Each memory image file has a 3 word (or 6 bytes) header that indicates to a loader where in memory to load the program. The header is:

WORD 1: Flag word which is either >FFFF to indicate that the program consists of at least one more segment after this one; or is >0000 which indicates that this is the last segment of the program.

WORD 2: Length of code in this segment in bytes. Note that the length of the segment on disk is 6 bytes more to include the 6 bytes of header information.

WORD 3: Address at which this segment is to be loaded.

NOTE that the length of the memory image program may not be the same as the length of the tagged object program. This can be caused by a work area that has no code or data loaded into it occurring at the end of the program. LINKER will not waste disk space writing out an area that you have not filled with code or data. In fact, if your program contains areas inside the program that are unused and which exceed 518 bytes in size LINKER will break the program at that point and create two separate segments. The 518 bytes is the overhead for creating another file -- 1 sector for the disk catalog, 1 sector minimum for a file and 6 bytes of header information per file.

NOTE: The flag word (WORD 1) as defined above is the TI standard definition. This standard has been extended (mainly by Millers Graphics for their GRAM KRACKER). This extended flag use is supported by LINKER via the FLAG statement. This means that LINKER can be used to convert GPL object code to memory image for loading by the GRAM KRACKER.

The extended flag field is defined as follows:

Byte 1 -
>FF This is not the last segment of the program.
>00 This is the last segment of the program.

Byte 2 -
>FF Segment is to be loaded into CPU RAM.
>00 Segment is to be loaded into CPU RAM. Note that >00 and >FF are the same for compatibility to the TI standard.
>01 Load into GRAM 0 at >0000.
>02 Load into GRAM 1 at >2000.
>03 Load into GRAM 2 at >4000.
>04 Load into GRAM 3 at >6000.
>05 Load into GRAM 4 at >8000.
>06 Load into GRAM 5 at >A000.
>07 Load into GRAM 6 at >C000.
>08 Load into GRAM 7 at >E000.
>09 Load into ROM BANK 1 at >6000.
>0A Load into ROM BANK 2 at >6000.

SUBROUTINE LIBRARY

On the LINKER distribution disk is a LINKER library with name "RAGLIB" containing the following routines:

DSRLNK Device Service Routine Link
GPLLNK GPL Subroutine Link
KSCAN Keyboard Scan
VMBR VDP Multi Byte Read
VMBW VDP Multi Byte Write
VSBR VDP Single Byte Read
VSBW VDP Single Byte Write
VWTR VDP Write To Register

Which perform the same functions as the routines described in the Editor/Assembler or Mini Memory manuals. The routines are all separate so that when the library is searched by LINKER only those used will be loaded. These routines (in particular GPLLNK) will work no matter which cartridge is used to load the memory image program. They are all relocatable object modules,

BEGINNER ASSEMBLER - 4
FILE HANDLING

by Mack McCormick, USA

File specifications describing a file's record length, length format, data and file format, and mode of operation are accumulated in a block of memory known as the Peripheral Access Block (PAB). The writing, reading and updating of file data is handled by resident routines called Device Service Routines (DSR's). For every file you must open, read or write, and close it.

You must first define the file characteristics. The actual number of bytes in the PAB are variable. PAB byte zero instructs the DSR which operation you wish to perform. (e.g. open, read, write, close). Here are the available opcodes for PAB byte zero:
You cannot use opcode >08 (scratch) with disks. The number one byte defines the files open mode, record type, type of data, and sequential or random. Bits 0,1, and 2 are used to report error conditions as they occur. Here are the available opcodes for PAB byte one:

Bytes 2,3 (1 word) contain the address in VDP RAM to be used as a buffer as the data is read or written. Byte 4 defines the logical record length in bytes. For variable length records this value is the maximum length. The largest value is >FF (255). Byte 5 defines the number of bytes to be written for a write operation, or the actual number of bytes to be read for a read operation. For fixed length records PAB byte 4 and 5 should be set equal when writing and will always be equal when reading. For variable length records PAB byte 5 can be tested to determine the actual record length on a read and can be dynamically changed for each write. PAB byte 5 can never exceed byte 4. PAB bytes 6 and 7 are only used for relative records. This word contains the relative record number to access. The MSB is ignored so the range of values is 0 to 32,767. Byte 8 is only used for files to be stored on cassette tape. Set byte 8 to >60 (screen offset) for cassette access. Byte 9 is the file descriptor length. Byte 10 i the file description. Here's an example PAB:

PABs are coded in your program and then placed in VDP RAM with VMBW. The first free address in VDP RAM for PABs is usually >F80. VDP RAM free space extends through >37D6. Lines 5,6,10,21-24 in our program place the PAB in VDP RAM. Actual access is obtained by the DSR routine. REF DSRLNK must be included in your program. The pointer need by DSRLNK is the address of the file descriptor byte. This value must be placed at >8356. Line 54 and 55. DSRLNK is then envoked with a BLWP instruction. Lines 56 and 57. When errors occur the equal bit of the status byte is set.

---

and are not necessarily loaded into low memory as they are by E/A.

Note also, that the information about the DSR routine left in memory by the E/A and MM DSRLNK routines is not left by the library DSRLNK.

ERRORS CORRECTED V3 (October 1988)

1. A zero length line caused LINKER to crash.
2. The error code from DSRLNK was being returned in the wrong position in R0.

NEW FEATURES V3

1. The FLAG statement allows you to change the flag word on the memory image program.
2. An "install" program allows changing the LINKER defaults without using a disk patcher.
3. The printer file can now be directed to disk. ■

You must design the layout of each file.

Recommend you read the following references in your editor assembler manual:

Section 16.2.2 page 251
Section 16.2.4 page 262
Section 16.5 page 270 through
Section 16.5.4 page 271

EDITORIAL

The only way to become a proficient programmer is to write programs. For extra practice try converting this BASIC program to assembler. You have had all the programming pieces in previous tutorials, you just have to put them together. All you lack is confidence.

```
100  CALL CLEAR
200  OPEN #1:"PIO",SEQUENTIAL,
DISPLAY, OUTPUT, VARIABLE 80
300  REM SUBSTITUTE YOUR PRINTER
NAME FOR "PIO"
400  FOR I=1 TO 10
500  PRINT "YOUR NAME"
600  PRINT #1:"YOUR NAME"
700  NEXT I
800  CLOSE #1
900  END
```

```
****************************************
*  THIS ROUTINE DEMOS FILE ACCESS      *
*   BY RALPH MOLESWORTH                *
****************************************

       DEF  START
       REF  DSRLNK,VSBW,VMBW,VSBR,VMBR
STATUS EQU  >837C
POINTR EQU  >8356         DSR POINTER ADDRESS
BUFADR EQU  >1000         VDP RAM ADDR FOR REC BUFFER
PABADR EQU  >F80          VDP PAM ADDR FOR PAB
READ   BYTE >02           "READ" OP-CODE
CLOSE  BYTE >01           "CLOSE" OP-CODE
EOF    DATA 0             EOF FLAG
PAB    DATA >0014,BUFADR,>5000,>0000,>000A  PAB DATA
       TEXT 'DSK1.FILE1'
ERRMSG TEXT 'I/O ERROR='  DSR ERROR MESSAGE
CPUBUF BSS  80            CPU RAM REC BUFFER ADDR
FNAME  EQU  CPUBUF        FIRST NAME ADDR
LNAME  EQU  CPUBUF+14     LAST NAME ADDR
LEN    BSS  2             ACTUAL RECORD LEN WORKSPACE
RETURN BSS  2             SAVE RTN ADDR AREA
WR     BSS  >20           WORKSPACE REGISTERS
START  MOV  R11,@RETURN   SAVE RETURN ADDR
       LWPI WR
       LI   R0,PABADR     VDP RAM ADDR FOR PAB
       LI   R1,PAB        CPU RAM ADDR FOR PAB DATA
       LI   R2,20         LENGTH OF DATA
       BLWP @VMBW         WRITE PAB TO VDP RAM
       BL   @DSR          OPEN THE FILE
       MOVB @READ,R1      LOAD READ OPCODE INTO R1
       LI   R0,PABADR     LOAD PAB ADDR INTO R0
       BLWP @VSBW
       CLR  R4            R4 IS RECORD COUNTER
READF  BL   @DSR          PERFORM DSR ROUTINE
       MOV  @EOF,@EOF     CHECK FOR EOF
       JNE  EOJ
       INC  R4            ADD 1 TO REC COUNT
       CI   R4,3          CHECK FOR THIRD REC
       JNE  READF         IF NOT THIRD, READ AGAIN
       LI   R0,PABADR+5   ADDRESS OF THE CHARACTER COUNT
       BLWP @VSBR         READ COUNT INTO LEFT BYTE R1
```

```
       SRL  R1,8
       MOV  R1,R2          MOVE VALUE TO R2
       MOV  R1,@LEN        SAVE VALUE IN LEN
       LI   R0,BUFADR      VDP RAM REC BUFFER ADDR
       LI   R1,CPUBUF      CPU RAM ADDR FOR REC
       BLWP @VMBR
       LI   R0,290         PUT UP FIRST NAME
       LI   R1,FNAME
       LI   R2,14
       BLWP @VMBW
       LI   R0,305         PUT UP LAST NAME
       LI   R1,LNAME
       MOV  @LEN,R2        MOV RECORD LEN TO R2
       AI   R2,-14         SUBTRACT LEN OF FIRST NAME
       BLWP @VMBW
       JMP  EOJ            GOTO END OF JOB
DSR    LI   R6,PABADR+9    LOAD R6 WITH DESCRIPTOR LEN
       MOV  R6,@POINTR     MOV ADDR TO POINTER
       BLWP @DSRLNK
       DATA 8              DATA NEEDED BY DSR LINK
       JEQ  DSRERR         CHECK FOR ERROR
       RT
DSRERR INC  @EOF           SET EOF INDICATOR
       LI   R0,PABADR+1    ADDRESS OF PAB BYTE 1
       BLWP @VSBR
       SRL  R1,13          SHIFT HIGH ORDER 3 BITS TO LOW
       CI   R1,5           CHECK FOR EOF VAL=5
       JNE  IOERR          IF NOT EOF THEN OTHER ERROR
       RT
IOERR  AI   R1,>30         MASK ERROR CODE
       SLA  R1,8           SWAP BYTES
       LI   R0,299         PUT UP ERROR CODE
       BLWP @VSBW
       LI   R0,288         DISPLAY ERROR MSG
       LI   R1,ERRMSG
       LI   R2,10
       BLWP @VMBW
EOJ    MOV  @EOF,@EOF      IF EOF REACHED, DSR WILL
       JNE  NOCLOS         CLOSE FILE
       MOVB @CLOSE,R1      MOVE CLOSE OP-CODE TO R1
       LI   R0,PABADR      LOAD PAB ADDR
       BLWP @VSBW          WRITE CLOSE OP-CODE TO PAB 0
       BL   @DSR           CLOSE FILE
NOCLOS DECT @RETURN        ALTER RETURN ADDR
       MOV  @RETURN,R11
       RT
       END
```

```
*********************************************
*         BASIC PROGRAM TO                  *
*           CREATE FILES                    *
*100 CALL CLEAR                             *
*110 OPEN #2:"DSK1.FILE1",OUTPUT,VARIBLE 80 *
*120 INPUT "ENTER AN E WHEN DONE":X$        *
*130 IF X$="E" THEN 190                     *
*140 INPUT "FIRST NAME?":FN$                *
*150 IF LEN(FN$)>14 THEN 140                *
*160 INPUT "LAST NAME?":LN$                 *
*170 PRINT #2:FN$,LN$                       *
*180 GOTO 120                               *
*190 CLOSE #2                               *
*200 END                                    *
*********************************************
```

TIPS FROM TIGERCUB #43

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213

If you have as much trouble as I do, trying to get the strip labels lined up in the printer, you'll like this one -

```
100 DISPLAY AT(4,7)ERASE ALL
:"TIGERCUB LABELER": : : :"
This label maker will allow"
:"you to specify different":
"printer codes for each line
"
110 DISPLAY AT(11,1):"of a 5
-line label.": : :" You may
stop the program":"while lab
els are printing":"by pressi
ng any key, turn"
120 DISPLAY AT(17,1):"off th
e printer to adjust":"the la
bels, turn it back on,":"and
 press any key to con-":"tin
ue printing."
130 DISPLAY AT(23,1):"Printe
r designation?":"PIO" :: ACC
EPT AT(24,1)SIZE(-28)BEEP:PR
$ :: OPEN #1:PR$ :: P$,E$,DS
$,CEN$="Y" :: DW$,I$,SS$,U$=
"N" :: P=1
140 CALL CHAR(95,"FF")
150 FOR J=1 TO 5 :: CALL KEY
(3,K,S)
160 DISPLAY AT(2,1)ERASE ALL
:"Line #";J;" - PRINT? "&P$
:: CALL QUERY(2,20,P$):: IF
P$="N" THEN L$(J)="" :: GOTO
 360
170 IF J>1 THEN DISPLAY AT(4
,1):"Change codes? N" :: CAL
L QUERY(4,15,Q$):: IF Q$="N"
 THEN 300
180 DISPLAY AT(4,1):"Print p
itch? ";P:" (1)pica":" (2)el
ite":" (3)condensed" :: ACCE
PT AT(4,15)SIZE(-1)VALIDATE(
"123"):P
190 CI=(P=1)*-10+(P=2)*-12+(
P=3)*-17 :: L$(J)=CHR$(27)&"
B"&CHR$(P):: DISPLAY AT(5,1)
:"":"":""
200 DISPLAY AT(6,1):"Double
width? "&DW$ :: CALL QUERY(6
,15,DW$):: IF DW$="Y" THEN C
I=CI/2 :: L$(J)=L$(J)&CHR$(1
4)ELSE L$(J)=L$(J)&CHR$(20)
210 DISPLAY AT(8,1):"Italics
? "&I$ :: CALL QUERY(8,10,I$
):: IF I$="Y" THEN L$(J)=L$(
J)&CHR$(27)&"4" ELSE L$(J)=L
$(J)&CHR$(27)&"5"
220 DISPLAY AT(10,1):"Supers
cript? "&SS$ :: CALL QUERY(1
0,14,SS$):: IF SS$="Y" THEN
L$(J)=L$(J)&CHR$(27)&CHR$(83
)&CHR$(0)ELSE L$(J)=L$(J)&CH
R$(27)&CHR$(84)
230 IF SS$="Y" THEN 250
240 DISPLAY AT(12,1):"Double
-strike? "&DS$ :: CALL QUERY
(12,16,DS$):: IF DS$="Y" THE
N L$(J)=L$(J)&CHR$(27)&"G" E
LSE L$(J)=L$(J)&CHR$(27)&"H"
250 IF P<>1 OR SS$="Y" THEN
270 :: DISPLAY AT(14,1):"Emp
hasized? "&E$ :: CALL QUERY(
14,13,E$)
260 IF E$="Y" THEN L$(J)=L$(
J)&CHR$(27)&"E" ELSE L$(J)=L
$(J)&CHR$(27)&"F"
270 DISPLAY AT(16,1):"Underl
ine? "&U$ :: CALL QUERY(16,1
2,U$)
280 IF U$="N" THEN L$(J)=L$(
J)&CHR$(27)&CHR$(45)&CHR$(0)
290 DISPLAY AT(18,1):"Center
 text? Y" :: CALL QUERY(18,1
4,CEN$)
300 DISPLAY AT(18,1):"Type l
ine";J;". Enter each":"scree
n line, enter again":"when d
one." :: DISPLAY AT(22,1):RP
T$("_",INT(CI*3.5)):: R=21 :
: CALL KEY(5,K,S)
310 ACCEPT AT(R,1):M$ :: IF
M$="" THEN 320 :: A$=A$&M$ :
: R=R+1 :: GOTO 310
320 IF LEN(A$)>INT(CI*3.5)TH
EN DISPLAY AT(16,1):"LINE TO
O LONG!" :: CALL SOUND(300,1
10,0,-4,0):: A$="" :: R=21 :
: GOTO 310
330 L=LEN(A$):: IF U$="Y" TH
EN A$=CHR$(27)&CHR$(45)&CHR$
(1)&A$&CHR$(27)&CHR$(45)&CHR
$(0)
340 IF CEN$="Y" THEN A$=RPT$
(" ",(INT(CI*3.5)-L)/2)&A$
350 L$(J)=L$(J)&A$ :: A$=""
360 NEXT J
370 DISPLAY AT(12,1)ERASE AL
L:"Print how many?" :: ACCEP
T AT(12,17):N
380 FOR J=1 TO N :: FOR K=1
TO 6 :: PRINT #1:L$(K):: NEX
```

```
T K
390 CALL KEY(0,K,S):: IF S=0
 THEN 410 ELSE CLOSE #1
400 CALL KEY(0,K1,S1):: IF S
1<1 THEN 400 ELSE OPEN #1:PR
$
410 NEXT J
420 DISPLAY AT(12,8)ERASE AL
L:"Another?" :: CALL QUERY(1
2,17,Q$):: IF Q$="N" THEN ST
OP ELSE 150
430 SUB QUERY(R,C,Q$):: ACCE
PT AT(R,C)SIZE(-1)VALIDATE("
YN")BEEP:Q$ :: SUBEND
```

More peculiarities of the TI computer -

```
90 CALL CLEAR :: PRINT TAB(7
);"SPRITE PUZZLE #1":"
 from Tigercub"
100 PRINT "A non-existent sp
rite can be":"created by CAL
L MOTION.": :"It apparently
starts in"
110 PRINT "dot-row 1, dot-co
lumn 1, and":"has color 1, b
ut its pattern":"is not that
 of any ASCII!"
120 !by Jim Peterson
130 FOR CH=0 TO 255 :: PRINT
 CHR$(CH);:: NEXT CH
135 PRINT "CALL MOTION(#1,5,
5):: CALL COLOR(#1,16):: CAL
L MAGNIFY(4)"
140 CALL MOTION(#1,5,5):: CA
LL COLOR(#1,16):: CALL MAGNI
FY(4)
150 GOTO 150
```

And another -

```
100 DISPLAY AT(3,5)ERASE ALL
:"SPRITE PUZZLE #2": :"
 from Tigercub"
110 DISPLAY AT(7,1):"Non-exi
stent sprites can be":"creat
ed by CALL COLOR.": :"Their
existence can be con-"
120 DISPLAY AT(11,1):"firmed
 by CALL COINC, but":"CALL P
OSITION reports that":"they
have no position!"
130 CALL COLOR(#1,16):: CALL
 COLOR(#2,16)
140 CALL COINC(#1,#2,1,X)::
DISPLAY AT(15,1):"COINC #1,#
2=";X :: CALL POSITION(#1,X,
Y)
150 CALL POSITION(#1,X,Y)::
DISPLAY AT(17,1):"POSITION #
1=";X;Y
160 CALL POSITION(#2,X,Y)::
DISPLAY AT(19,1):"POSITION #
2=";X;Y
170 IF FLAG=1 THEN 140 :: FL
AG=1
180 DISPLAY AT(21,1):"PRESS
ANY KEY"
190 CALL KEY(0,K,S):: IF S=0
 THEN DISPLAY AT(21,1):"pres
s any key" :: GOTO 180
200 DISPLAY AT(21,1):"Until
they're set in motion!"
210 CALL MOTION(#1,5,5):: CA
LL MOTION(#2,-5,-5):: GOTO 1
50
```

If you have the Terminal Emulator II, Speech Synthesizer, and a pre-schooler in the house, this will help him to grasp the idea of spelling as well as letter recognition and keyboard familiarization-

```
100 REM  PRE-SPELLER BY JIM
PETERSON
110 REM  TI BASIC WITH TERMI
NAL EMULATOR II AND SPEECH S
YNTHESIZER
120 CALL CLEAR
130 DIM M$(100),S$(100)
140 OPEN #1:"SPEECH",OUTPUT
150 PRINT "         PRE-SPELL
ER":::::
160 PRINT "TYPE WORDS TO PRA
CTICE":"TYPE `END' WHEN FIN
ISHED"
170 X=X+1
180 INPUT M$(X)
190 IF M$(X)="END" THEN 380
200 PRINT #1:M$(X)
210 PRINT "PRONUNCIATION OK?
 (Y/N)"
220 CALL KEY(3,K,S)
230 IF S<1 THEN 220
240 IF K=78 THEN 280
250 IF K<>89 THEN 220
260 S$(X)=M$(X)
270 GOTO 170
280 PRINT "TRY SPELLING PHON
ETICALLY"
290 INPUT S$(X)
300 PRINT #1:S$(X)
310 PRINT "PRONUNCIATION OK?
 (Y/N)"
320 CALL KEY(3,K,S)
330 IF S<1 THEN 320
340 IF K=89 THEN 170
350 IF K<>78 THEN 320
360 PRINT "TRY AGAIN"
370 GOTO 290
380 CALL CLEAR
390 FOR J=1 TO X-1
400 PRINT #1:"CAN YOU SPELL
```

```
THIS?"
410 FOR A=1 TO LEN(M$(J))
420 CALL HCHAR(12,8+A,ASC(SE
G$(M$(J),A,1)))
430 NEXT A
440 FOR B=1 TO LEN(M$(J))
450 CALL KEY(3,K,S)
460 IF (S<1)+(K=32)THEN 450
470 IF K=ASC(SEG$(M$(J),B,1)
)THEN 500
480 GOSUB 640
490 GOTO 450
500 C$=C$&CHR$(K)
510 CALL HCHAR(14,8+B,K)
520 NEXT B
530 IF C$<>M$(J)THEN 640
540 PRINT #1:S$(J)
550 FOR D=1 TO 500
560 NEXT D
570 PRINT #1:"VEREE GOOD"
580 FOR D=1 TO 500
590 NEXT D
600 C$=""
610 CALL HCHAR(12,1,32,100)
620 NEXT J
630 GOTO 390
640 PRINT #1:"NO THAT IS NOT
 RIGHT"
650 PRINT #1:"TRY AGAIN"
660 RETURN
```

And, a simple little game that is a bit different than any I've seen -

```
100 !FORMATION by Jim Peters
on - use the S and D keys
110 CALL CLEAR :: CALL CHAR(
100,"381010FEFE383810103838F
EFE10103838"):: CALL SCREEN(
5):: CALL MAGNIFY(2):: RANDO
MIZE
120 V,W,P=0 :: FOR J=1 TO 7
:: CALL SPRITE(#J,100,7,1,25
0*RND+1,10,4):: FOR D=1 TO 1
00 :: NEXT D :: NEXT J :: CA
LL SPRITE(#11,101,16,160,128
)
130 CALL KEY(3,K,S):: W=W+1
:: IF W=150 THEN 170 ELSE IF
 W=300 THEN 180 ELSE IF K=68
 THEN V=V+2+(V>125)*2 ELSE I
F K=83 THEN V=V-2-(V<-125)*2
140 IF P=0 THEN CALL MOTION(
#11,0,V)ELSE IF P=1 THEN CAL
L MOTION(#11,0,V,#12,0,V)ELS
E CALL MOTION(#11,0,V,#12,0,
V,#13,0,V)
150 CALL COINC(ALL,A):: IF A
=0 THEN 130
160 CALL SOUND(1000,-4,0)::
H=MAX(H,W):: DISPLAY AT(23,1
):"SCORE";W:"HIGH SCORE";H :
: CALL DELSPRITE(ALL):: GOTO
 120
170 P=1 :: CALL POSITION(#11
,R,C):: CALL SPRITE(#12,101,
16,160,C-40-(C<40)*256):: GO
TO 140
180 P=2 :: CALL POSITION(#11
,R,C):: CALL SPRITE(#13,101,
16,160,C+40+(C>216)*256):: G
OTO 140
```

If you can't figure out where all the money goes, this may be an eye-opener -

```
100 DISPLAY ERASE ALL AT(3,5
):"THE COST OF CREDIT" ! by
Jim Peterson
110 S,T,X=0 :: DISPLAY AT(8,
1):"AMOUNT OF PURCHASE?" ::
ACCEPT AT(8,21):A :: B,T=A :
: DISPLAY AT(10,1):"CREDIT C
ARD INTEREST RATE?" :: ACCEP
T AT(11,1):R
120 DISPLAY AT(13,1):"SAVING
S ACCOUNT INT. RATE?" :: ACC
EPT AT(14,1):SR
130 X=X+1 :: I=B*R/100/12 ::
 B=B+I :: T=T+I :: P=B/10 ::
 B=B-P :: S=S+P+S*SR/100/12
:: IF S<A THEN 130
140 D$="$"&STR$(INT((T-A+S-A
+.5)*100)/100)
150 DISPLAY AT(17,1):"If you
 had saved the amount":"of y
our minimum 10% of the":"bal
ance credit card payment":"e
ach month for";X;"months,"
160 DISPLAY AT(21,1):"and us
ed it to pay cash, you":"wou
ld have saved ";D$ :: GOTO 1
10
```

And this is one of the handiest routines I've seen in a long time -

```
10 !TURNS ALL NUMERALS AND P
UNCTUATION WHITE! BY HARRY W
ILHELM IN TWIN TIers UG NEWS
LETTER
20 !TURN IT OFF BY CALL LOAD
(-31804,0)::TURN IT ON BY CA
LL LOAD(-31804,63)
100 CALL INIT
110 CALL LOAD(16128,2,224,38
,0,2,0,8,17,2,1,63,36,2,2,0,
3,4,32,32,36,2,224,131,192,3
,128)
120 CALL LOAD(16164,240,240,
240)
130 CALL LOAD(-31804,63)
```

Memory full Jim Peterson ■

FAST EXTENDED BASIC!
87/10 - HELLO, PSCANFORM

A longer program this month, but as useless as the one last month; believe me, you will not impress anyone, its purpose is totaly educational. And the lesson is loaded! So much in fact that it will give me something to write about next month...

So first study and run HELLO, then read all about our subject today: PRE-SCANNING!

When you ran the program, did you notice that it started right away, instead of the long waiting you would normally expect? Why? Each time you ask Tex to run a program for you, it first reads it all to check for errors and, more important, to set aside memory space for your variables, arrays, data, etc. Actually, Tex needs only the first reference to each of those, so the creators of XB have provided you with a mean to put this pre-scanning off and on at will; in other words, you can identify the sections of your program that will be included in the pre-scanning and those that need not be.

As the "Extended Basic Product Information" booklet (inserted into the XB manual) says, "careful planning is required"; and it then sets up five rules, all dutifully followed in the above program:

1. "Enter your first DATA statement within the pre-scan". In order to keep the pre-scanning to the minimum, and leave the data lines where I wanted them, I invented a dummy DATA (line 140), that I then read (but do not use further) in line 170. You could also frame your first data line by the pre-scan statements: here, we could add a line "195 !@P+", on, and "205 !@P-", off.

2. "Include the first use of each variable and/or array (also the OPTION BASE if used)". But what happens if you do not use all your variables at the beginning of the program? Well, XB provides you with the possibility to put them all in one line or two at the beginning; but you MUST precede that line with a GOTO to the next program line for it to execute properly. In our example, all the variables that are not used before the computer encounters the pre-scan line (which is line 160) are thus declared in line 120; since the variables T and V, and the NOTE() array, were used in line 110, they are not repeated in line 120.

3. "Include the first reference to each CALL statement of any subprogram", whether they be TI's or the ones you created yourself, like CALL WAIT. This is done in lines 120 and 130. You will notice that I have not declared the CALL KEY statement in line 400, since it will be pre-scanned with the SUB (see 5).

4. "Include all DEF statements for user-defined functions", as I did in line 150; the definition itself is used in line 280.

5. "Include all SUB statements and SUBEND statements in the pre-scan". That of course applies to the user-defined subs; since as a rule they have to be put at the very end of your program, the easiest thing to do is to put the pre-scan off statement on a line before your sub, as I have done in line 380. Since we learned last month that the variables in our own subs are totally independent from the ones in the body of the program (I could have used A and B instead of K and S), Tex needs to know them if you want it to save memory space for them too; the CALL KEY statement will also be pre-scanned here.

Now you know what is a PRE-SCAN; the best way to implement it is first to make sure that your program works, then gather all your variables and

CALL statements. A long task, so I have written PSCANFORM; it will print a convenient form that you can use for your gathering of information. Note line 110, another way to put the "pre-scan off" statement in your program.

What happens if you forget to include a variable or a statement in you pre-scan? Well, Tex will unpolitely stop and warn you with a "* SYNTAX ERROR IN nn". Just go back to your pre-scanning line(s), and include it.

As for the extra goodies, they concern simpler ways to use the graphic CALL statements in XB; I will leave that for next month, and we will use the same program, HELLO, so please keep it.

```
100 REM ** HELLO ** L. Dorai
s/sept 87
110 OPTION BASE 1 :: DIM NOT
E(5):: T=-200 :: V=0
120 GOTO 170 :: A,A$,B,B$,C,
DUM$,I :: CALL WAIT :: CALL
CLEAR :: CALL SOUND :: CALL
SCREEN :: CALL COLOR :: CALL
 CHAR
130 CALL HCHAR :: CALL VCHAR
 :: CALL CHARPAT :: CALL CHA
RSET
140 DATA DUMMY
150 DEF D(X)=-(3*X)
160 !@P-
170 READ DUM$ :: CALL CHARPA
T(63,A$,87,B$):: CALL CHAR(3
6,A$,35,B$)
180 FOR I=1 TO 5 :: READ NOT
E(I):: NEXT I
190 DATA 1047,1175,1319,1397
,1568
200 DISPLAY AT(9,3)ERASE ALL
:"#ant to have A surprise$"
210 CALL WAIT :: CALL SCREEN
(16)
220 REM ** define char. and
play music **
230 A$="FEFEFEFEFEFEFEFE"
240 CALL SOUND(T,NOTE(1),V):
: CALL CHAR(56,A$,57,"3F3F3F
3F3F3F3F3F8080808080808080FE
FEFEFEFEFFFFFF")
250 CALL SOUND(T,NOTE(2),V):
: CALL CHAR(64,"0000000000FF
FFFF3F3F3F3F3FFFFFFFFFFFFFFF
FEFEFEFEFFFFFFFF00000000")
260 CALL SOUND(T,NOTE(3),V):
: CALL CHAR(63,"FFFFFFFF3F3F
3F3F")
270 CALL SOUND(T,NOTE(4),V):
: CALL CHAR(72,A$)
280 CALL SOUND(D(T),NOTE(5),
V):: CALL CHAR(85,A$,86,"FEF
EFEFEFE00000000387CFEFE7C380
0")
290 CALL CLEAR :: CALL COLOR
(4,1,16,5,1,16,6,1,16,7,1,16
)
300 REM ** draw **
310 CALL VCHAR(7,11,56,6)::
CALL VCHAR(7,13,57,6):: CALL
 VCHAR(7,14,58,6)
320 FOR I=1 TO 6 :: READ A,B
,C :: CALL HCHAR(A,B,C):: NE
XT I :: CALL COLOR(4,9,16,5,
9,16)
330 DATA 9,11,59,9,12,64,9,1
3,65,10,11,66,10,12,67,10,13
,63
340 CALL VCHAR(7,16,72,6)::
CALL COLOR(6,5,16)
350 CALL VCHAR(7,19,85,4)::
CALL HCHAR(11,19,86):: CALL
HCHAR(12,19,87):: CALL COLOR
(7,12,16)
360 CALL WAIT :: CALL CLEAR
:: CALL CHARSET
370 DISPLAY AT(9,3):"NOW..."
: :"  THAT WAS AN EASY ONE?"
 :: CALL WAIT :: STOP
380 !@P+
390 SUB WAIT :: DISPLAY AT(2
4,6)BEEP:"press a key please
"
400 CALL KEY(0,K,S):: IF S=0
 THEN 400 ELSE DISPLAY AT(22
,6):""
410 SUBEND
```

```
80 REM ** PSCANFORM **
   prints an easy-to-fill
   form to list pre-scanned
   variables and CALLs
90 REM
100 L$=RPT$("_",34):: LL$=L$
&L$&"_____" :: S$="  " :: V$
=" VARIABLES"
110 CALL CLEAR :: GOTO 120 :
: PGM$,X :: !@P-
120 OPEN #1:"PIO" ! or your
printer
130 PRINT "PRE-SCANNING"&V$:
 :"FOR PROGRAM (title): ": :
```

```
 :: INPUT ">":PGM$
140 PRINT #1:CHR$(27)&"@":"
PRE-SCAN FOR: "&CHR$(14)&PGM
$: :" will be on line _____
_": : :
150 PRINT #1:TAB(12);"NUMERI
C"&V$;TAB(52);"STRING"&V$:S$
&LL$
160 FOR X=65 TO 90 :: PRINT
#1:S$&L$&"| "&CHR$(X)&" |"&L
$ :: NEXT X
170 PRINT #1:"":"":"   CALL
STATEMENTS:":"" :: FOR X=1 T
O 4 :: PRINT #1:S$&LL$ :: NE
XT X
180 CLOSE #1 :: END
```

■

HFDC 40 MBYTES HARD DISK

by Don Jones, Chicago Times, USA

I have completely, entirely run out of disk space on my first 40 meg hard disk. A solution was therefore required. And what, pray tell, do you think that my solution was??? Why, to get another 40 meg hard disk of course!!! Yes, that's exactly what I did, but you might like to know how I went about doing it. I first began by placing the following message up on Delphi information services:

HELP!!! I was about to purchase another Miniscribe 3053 hard disk, similar to the one which I currently have, as I like it a great deal. I called QUICK Electronics, where I purchased my first one, to find out what they are going for. I was told that the model was discontinued. I was then told that Miniscribe has gone out of business. I need some advice from other hard disk users. To be specific, I need to know what other 40 meg. hard disks are compatible with our Myarc HFDC. I know that the 40 meg. Seagates will work SOMETIMES, but sometimes they WON'T. I purchased one from QUICK, but I had to return it and have it replaced with my Miniscribe which has never given me a bit of trouble. PLEASE, I need to know what is functional with the Myarc HFDC BEFORE I go out and buy one. Thank you in advance. kdj

I then received the following reply:

Don, I use the Seagate ST251-0 on mine with no problems. The 251-1 will not work. The new Mitsubishi 40 meg works good, and it is a voice coil drive. I found one in computer shopper for $259.00, Roy

I then replied with the following message:

Roy, Thanks a bunch for the lead. I deeply appreciate your advice. I will get a copy of a recent edition of Computer Shopper, and I will look for the best price. BTW, do you have any experience with the Mitsubishi or do you personally know of anyone who has??? What is the model number of the Mitsubishi drive that you are speaking of??? I have had very good results with QUICK ELECTRONICS when I first purchased my Miniscribe. I am really sorry that they went out of business, as I have never had a lick of trouble with my current hard disk. Thanks for your time and your advice. kdj

Here is the final message which I received from Roy Camp:

Don, yes I know of a friend that purchased one, (and) he is in love with it. I think the speed is 28ms and it will work MFM or RLL, which is unusal. The model is MR535, Mitsubishi. Fast micro has it for 269.00, but I have found it for less. If you get one, let me know how it does. Roy

Here is my final message on the subject to Roy Camp:

Roy, just yesterday, I received my Mitsubishi MR 535B 40 meg. hard disk. I hope to have it installed by next week. More on it later. Thanks for the advice. BTW, I got it for $259.00 as it has been discontinued. kdj

In a community such as ours, it's good to know what others are using with success. It pays to check stuff out FIRST!!! ■

DATABAS FÖR PHT OCH PHM

(om du använder kassett så se ändringen på sid 2)

```
10 REM 99/4A PROGRAM BASIC
12 REM COMPUTE! 83-05 DSK
14 REM BY JEFFREY S.YOHAY
15 REM REV JAN ALEXANDERSSON
16 CALL CHAR(93,"00100038447
C4444")
18 CALL CHAR(91,"00440038447
C4444")
20 CALL CHAR(92,"0044007C444
4447C")
40 DIM R1$(300)
50 CC$="DA DT DS PA PT AD CH
 DE SR SN ST LO SA "
52 CALL CLEAR
54 PRINT TAB(7);"99/4A PROGR
AM":TAB(10);"DATABAS"
56 PRINT : :"DISPLAY";TAB(10
);"<DT> TITEL":TAB(10);"<DS>
 S\KA"
60 PRINT : :"[NDRA    <AD> A
DERA":TAB(10);"<CH> [NDRA":T
AB(10);"<DE> DELETE"
62 PRINT : :"SORTERA";TAB(10
);"<SR> NUMMER":TAB(10);"<SN
> NAMN":TAB(10);"<ST> TYP"
64 PRINT : :"DATAFIL  <LO> L
OAD":TAB(10);"<SA> SAVE": :T
AB(10);"<QU> QUIT PROGRAM":
:
70 INPUT C$
80 C$=SEG$(C$&"  ",1,2)
82 IF C$="QU" THEN 800
84 P=POS(CC$,C$,1)
85 IF P=0 THEN 52
86 IF (P-1)/3<>INT((P-1)/3)T
HEN 52
87 P=INT(P/3)+1
88 ON P GOSUB 240,290,340,40
0,430,450,820,560,600,600,60
0,730,750
89 GOTO 52
120 R$=R1$(IO)
122 TY$=SEG$(R$,8,16)
130 RETURN
144 TN$=SEG$(R$,24,2)
185 RETURN
190 CALL CLEAR
192 PRINT X1$;" RECORDS": :"
NUMMER ATT ";X2$: :
200 INPUT T$
201 T$=SEG$(T$&"       ",1,7
)
203 FOR IO=1 TO IR
204 IF T$=SEG$(R1$(IO),1,7)T
HEN 212
205 NEXT IO
207 PRINT : :"** NO SUCH REC
ORD **"
208 PRINT :"PRESS <ENTER>":
209 CALL KEY(0,K,S)
210 IF K<>13 THEN 209
212 RETURN
240 REM RESERV DA
250 RETURN
290 IP=0
291 MP=INT(IR/10)
292 IF IR/10<>INT(IR/10)THEN
 299
294 MP=MP-1
299 CALL CLEAR
300 PRINT "NR ";IP*10+1;"-";
IP*10+10: :
301 FOR IL=1 TO 10
302 IO=IP*10+IL
303 IF IO>IR THEN 310
304 GOSUB 120
305 GOSUB 350
306 NEXT IL
310 GOSUB 360
320 CALL KEY(0,K1,S1)
321 IF K1=77 THEN 365
322 K=K1
323 IF (K<>76)*(K<>78)+(((K=
76)+(K=78))*((IP=MP)+(MP<=0)
))THEN 320
325 IF K<>78 THEN 328
326 IP=IP+1
327 GOTO 299
328 IF K<>76 THEN 299
329 IP=MP
330 GOTO 299
340 X1$="SEARCH"
341 X2$="SEARCH FOR"
342 GOSUB 190
343 R$=R1$(IO)
344 PRINT : :SEG$(R$,1,7);"
";SEG$(R$,8,16);" ";SEG$(R$,
24,2): :
345 PRINT : :"PRESS ENTER"
346 CALL KEY(3,KEY,STA)
347 IF STA<1 THEN 346
348 RETURN
350 PRINT SEG$(R$,1,7);" ";S
EG$(R$,8,16);" ";SEG$(R$,24,
2): :
355 RETURN
360 PRINT :"<N>EXT   <L>AST
  <M>ENU";
365 RETURN
400 REM
430 RETURN
450 X$=""
451 GOSUB 530
452 RP=1
453 Q$="NR  "
454 L=7
455 GOSUB 540
456 Q$="NAMN"
```

```
457 L=16
458 GOSUB 540
459 Q$="TYP "
460 L=2
461 GOSUB 540
466 PRINT
512 IF IR<>0 THEN 515
513 I1=1
514 GOTO 525
515 FOR I1=1 TO IR
516 IF SEG$(R1$(I1),1,7)>=SE
G$(R$,1,7)THEN 520
517 NEXT I1
518 GOTO 525
520 FOR I2=IR TO I1 STEP -1
521 R1$(I2+1)=R1$(I2)
522 NEXT I2
525 R1$(I1)=R$
526 IR=IR+1
529 RETURN
530 CALL CLEAR
532 PRINT "     ** ADD RECORD
 **": : :
533 R$=""
534 RETURN
540 PRINT Q$;
541 INPUT A$
542 IF LEN(A$)<=L THEN 546
543 A$=SEG$(A$,1,L)
544 GOTO 550
546 FOR II=LEN(A$)+1 TO L
548 A$=A$&" "
549 NEXT II
550 R$=R$&A$
551 RP=RP+L
552 PRINT
554 RETURN
560 X1$="DELETE"
561 X2$="DELETE"
562 GOSUB 190
563 IF IO>IR THEN 572
565 PRINT :"DELETING RECORD
..."
567 FOR I=IO TO IR-1
568 R1$(I)=R1$(I+1)
569 NEXT I
570 IR=IR-1
572 RETURN
600 IF C$<>"SR" THEN 603
601 C=1
602 GOTO 610
603 IF C$<>"ST" THEN 606
604 C=2
605 GOTO 610
606 IF C$<>"SN" THEN 610
607 C=3
610 CALL CLEAR
611 PRINT "... SORTING RECOR
DS ...": : :
613 N=IR
620 N=INT(N/2)
622 IF N=0 THEN 699
624 I3=IR-N
626 I2=1
630 I1=I2
633 ON C GOTO 640,658,680
640 I4=I1+N
641 IF SEG$(R1$(I1),1,7)<=SE
G$(R1$(I4),1,7)THEN 650
642 T$=R1$(I1)
643 R1$(I1)=R1$(I4)
644 R1$(I4)=T$
645 I1=I1-N
646 IF I1>=1 THEN 640
650 I2=I2+1
655 IF I2>I3 THEN 620 ELSE 6
30
658 I4=I1+N
659 S1$=SEG$(R1$(I1),24,2)&S
EG$(R1$(I1),1,7)
660 S2$=SEG$(R1$(I4),24,2)&S
EG$(R1$(I4),1,7)
661 IF S1$<=S2$ THEN 650
663 T$=R1$(I1)
664 R1$(I1)=R1$(I4)
665 R1$(I4)=T$
666 I1=I1-N
667 IF I1>=1 THEN 658 ELSE 6
50
680 I4=I1+N
681 IF SEG$(R1$(I1),8,16)<=S
EG$(R1$(I4),8,16)THEN 650
682 T$=R1$(I1)
683 R1$(I1)=R1$(I4)
684 R1$(I4)=T$
685 I1=I1-N
687 IF I1>=1 THEN 680 ELSE 6
50
699 RETURN
730 X1$="LOAD"
732 GOSUB 760
734 OPEN #1:"DSK1.DATAFIL2",
INPUT ,INTERNAL,FIXED 32
736 IR=0
738 IR=IR+1
740 INPUT #1:R1$(IR)
742 IF EOF(1)<>0 THEN 747 EL
SE 738
747 CLOSE #1
748 RETURN
750 X1$="SAVE"
751 GOSUB 760
752 OPEN #2:"DSK1.DATAFIL2",
OUTPUT,INTERNAL,FIXED 32
754 FOR I=1 TO IR
755 PRINT #2:R1$(I)
756 NEXT I
758 CLOSE #2
759 RETURN
760 CALL CLEAR
762 PRINT "     ** ";X1$;" DA
TA FILE **": : : :
764 RETURN
790 B$=""
```

```
792 FOR B=1 TO B1
794 B$=B$&" "
796 NEXT B
798 RETURN
800 CALL CLEAR
810 END
820 X1$="[NDRA"
830 X2$="[NDRA"
840 GOSUB 190
850 IF IO<=IR THEN 860
855 RETURN
860 CALL CLEAR
870 PRINT "[NDRING AV:": :"1
 NUMMER": :"2 NAMN": :"3 TYP
": :"4 INGEN [NDRING": :
880 INPUT P
890 IF P<1 THEN 880
900 IF P>4 THEN 880
910 IF P=4 THEN 855
920 ON P GOTO 930,970,1010
930 PRINT : :"TIDIGARE NUMME
R: ";SEG$(R1$(IO),1,7): :
940 INPUT "NYTT NUMMER    :
":T$
950 R1$(IO)=SEG$(T$&"
",1,7)&SEG$(R1$(IO),8,18)
960 RETURN
970 PRINT : :"TID. NAMN: ";S
EG$(R1$(IO),8,16): :
980 INPUT "NYTT NAMN: ":T$
990 R1$(IO)=SEG$(R1$(IO),1,7
)&SEG$(T$&"                "
,1,16)&SEG$(R1$(IO),24,2)
1000 RETURN
1010 PRINT : :"TIDIGARE TYP:
 ";SEG$(R1$(IO),24,2): :
1020 INPUT "NY TYP:       ":
T$
1030 R1$(IO)=SEG$(R1$(IO),1,
23)&SEG$(T$&"  ",1,2)
1040 RETURN
```

■

```
PHT6002 TI-TREK          SP
PHT6003 PERS.FINANC.AIDS HA
PHT6004 PROGRAM.AIDS I   SK
PHT6006 MATH.ROUTINE LIB TE
PHT6007 TEACH YOUR.BASIC UV
PHT6008 ELECTRICAL E.LIB TE
PHT6009 MUSIC SK.TRAINER MU
PHT6010 MYSTERY MELODY   SP
PHT6011 COMP.MUSIC BOX   MU
PHT6013 GRAPHING PACK    GR
PHT6015 OLDIES GOOD. I   SP
PHT6016 STRUCTURAL E.LIB TE
PHT6017 OLDIES GOOD. II  SP
PHT6018 MARKET SIMULAT.  UV
PHT6019 TEACH EXT. BASIC UV
PHT6025 SAT. NIGHT BINGO SP
PHT6026 BRIDGE BIDD. I   UV
PHT6031 SPEAK & MATH     UB
PHT6037 DRAW POKER       SP
PHT6038 BUISNESS AIDS LI HA
PHT6039 BRIDGE BIDD. II  UV
PHT6041 BRIDGE BIDD. III UV
PHT6042 SPELL WRITER     UB
PHT6043 PIRATE ADVENTURE SP
PHT6044 AC CIRCUIT ANALY TE
PHT6046 ADVENTURE LAND   SP
PHT6047 MISSION IMPOSS.  SP
PHT6048 VODOO CASTLE     SP
PHT6049 THE COUNT        SP
PHT6050 STRANGE ODESSEY  SP
PHT6051 MYSTERY FUN HOUS SP
PHT6052 PYRAMID OF DOOM  SP
PHT6053 GHOST TOWN       SP
PHT6054 SAVAGE ISLAND    SP
PHT6056 GOLDEN VOYAGE    SP
PHT6067 BEGINNERS BASIC  UV
PHT6070 TI LOGO SAMPLER  SK
PHT6071 LINE ASSEMBLER   SK
PHT6101 ENTRAPMENT       SP ■
```

ASGARD EGI 80 KOL

Asgard Extended Graphics Interface (EGI) som är ett 80-kolumnskort med videoprocessorn V9938 för TI-99/4A beräknas börja levereras maj 1991. Se beskrivningen i PB 91-1 sid 32.

Det säljs nu både som byggsats och färdigt kort:

BASIC KIT, pris USD 95. Består av kretskort, V9938-chip, mekaniskt ytterhölje, standard EPROM-DSR, kretsschema, dokumentation, stycklista över komponenter som måste köpas separat.

COMPLETE KIT, pris USD 160. Består av Basic kit samt alla komponenter inklusive IC och RAM-chip exklusive spänningsomvandlare. Fem flexskivor med program medföljer.

READY-TO-GO EGI, pris USD 250. Består av ett färdigbyggt kort inklusive allt som medföljer Complete Kit. Spänningsomvandlare (ta reda på om den passar till 220 V före köp) och YAPP ritprogram medföljer.

Porto tillkommer i alla tre fall med USD 8 för flygpost till Europa. För Master Card och Visa tillkommer 7 %. Du kan även betala med postgirots utlandsblankett som kostar 35 kr per utförd transaktion. Adress: Asgard Peripherals, P.O. Box 10697, Rockville, MD 20849, USA. ■

FUNNELWEB UTILITY FSAVE

by Tony McGovern, Australia

The E/A SAVE utility loads as absolute code in low memory. FUNNELWEB is compatible with SAVE, but does take up its own 6K share of high memory, so the FSAVE utility has been prepared to allow SAVEing of object files loaded by FUNNELWEB, including into low memory. Refer to the E/A manual for general information.

FSAVE loads as absolute code overlaying the FUNNELWEB (UL) system area. The start and first executable instruction in your own code should be DEFed with SFIRST and the last address DEFed by SLAST. Select entry point SAVE and enter the filename to which your program is to be SAVEd in E/A compatible memory image format.

If the Loader has placed files so that SFIRST is in hi-mem and SLAST is in low memory, FSAVE will SAVE high memory from SFIRST to the FFAHM indicated by the Loader at UTLTAB+2 and then proceed to SAVE low memory from >2676 (above the E/A utilities) to the FFALM. The utilities are not included so that the files will remain compatible with FUNNELWEB if reloaded under a different module.

When used with Low-Loaded (Opt 6) files, FSAVE saves its first module from low-mem from SFIRST to the top of lo-mem, nominally >3FFA (at UTLTAB+4), and then from hi-mem from >A000 to SLAST. If SFIRST and SLAST both point to the same segment the SAVE is normal. The MBSAVE entry adjusts the hi-mem start to >A050 above the Mailbox. Use E/A SAVE for addresses in the >6000 to >8000 cartridge space.

The MEMSAV entry point allows direct entry of hex address limits for the memory block to be SAVEd. The second entry is the address of the last word (inclusive) to be saved. MEMSAV ignores SFIRST and SLAST but these must have been DEFed, perhaps by a dummy object file, for correct LOAD/RUN operation.

FSAVE indicates the actual length of the memory block saved in each file in the second word of the header block, to a maximum of >1FFA in each file. The TI E/A SAVE utility, amongst its other little foibles, adds a further 6 bytes to this count, but the program file loader in the E/A module believes the byte count in the header. In normal usage the extra 6 bytes, falsely indicated by E/A SAVE, as read in from VDP to CPU RAM do not cause any problems. FSAVE files will of course not cause any problems unless perhaps a loader incompatible with E/A is used.

File FWSAVE for cassette or long file saves has been removed from the package as no comments were ever received concerning its use. A revised version which works with Vn 4.3x exists and is available on request.■

RANDOMIZE XB LOAD

Bruce Harrison beskriver i Micropendium dec 1990 sid 44 att RANDOMIZE inte fungerar efter XB DSK1.LOAD. Om Extended Basic självstartar genom att leta efter programmet LOAD på skivenhet DSK1 så kommer RANDOMIZE inte ge ett slumpartat tal. Lösningen är att stoppa in en extra rad i LOAD-programmet:

```
CALL INIT::CALL PEEK(-31880,I,J)::
CALL LOAD(-31808,I,J)
```

Detta kommer att hämta ett värde från nedräknaren för skärmsläckning som sedan ges som frö till random. Denna lösning kommer ursprungligen från Harry Wilhelm. ■

```
90 REM RAM TILL SKÄRM (PB 84-5.13)
95 REM ANVÄND INTE POS 1
96 REM EL. POS 28 PÅ SKÄRM
100 CALL CLEAR
110 CALL SCREEN(5)
120 CALL VCHAR(1,31,1,96)
130 FOR SET=1 TO 12
140 CALL COLOR(SET,2,16)
150 NEXT SET
160 GOTO 160
```
■

"Of course you can have this dance, provided, of course that you have...

* 16 all-new Dinosaurs
* Educational Information on the life of Dinosaurs
* A Dinosaur Quiz Game
* An Animated Cartoon

A two-disk set for only $12

Fonts & Borders
VOLUME I Just $7 plus P&H
7 Artist Fonts & over 15 Borders

Typewriter2 Baskerville2
Baskerville1
3D2 Ctype4

Also Available: Fonts & Borders II at $7 +P&H, Fonts & Borders III at $8 +P&H, or get all 3 Fonts & Borders Volumes for $20 +P&H!

Tired of saving the universe and slaying dragons with your joystick? Want to go to a place where the action is always hot and your credit is always good? Well then, NOTUNG Software has just the place for you... TI CASINO.

TI Casino is a collection of eight Casino games, all interlinked, so you can play any of the games with the same money. Each game is virtually joystick operated with no confusing keypresses. At the end of your gambling session, take your winnings to the cashier and get a printed check!

ACEY DEUCEY ♠ BLACKJACK ♠ BACCARAT ♠ CRAP TABLES
KENO ♠ DRAW POKER ♠ ROULETTE ♠ SLOT MACHINES

Blackjack includes Doubling down, Splits & Insurance, there's unlimited betting in Roulette and on the Crap Tables, and above all, be sure to dine at the Green Parrot Restaurant while playing Keno!

$15 through NOTUNG Software

Specify SSSD or DSSD Version, Joysticks, 32k & Ext.Basic Req'd, Printer Opt'l

And yes, we are proud to admit the fact that this entire Advertisement was composed & printed using a TI99/4a & 9640.

THE RING COMPANION

EXPLORE the fiery world of Richard Wagner's "Der Ring des Nibelungen" through 14 TI-Artist format pictures of Dwarves, Giants, Heroes and Gods. There's also onscreen text of all aspects of the Ring and an informative and muscial look at the Leitmotifs (or musical words) used in the Musical dramas. The two-disk package sells for just $8 +P&H. A must if you have Ken Gilliland's Wagner fairware disks.

ALSO available from NOTUNG SOFTWARE:

FILMLIB for TI-BASE	$7
CERTIFICATE 99 COMPANION PLUS	$7
STAR TREK NEXT GENERATION CALENDAR (JULY 90-JUNE 91)	$10

SHIPPING: Please remember to specify Disk Format and include $1 for the first piece of software and 50 cents for each addt'l piece. Ask for our FREE illustrated catalog.

NOTUNG SOFTWARE

Ken Gilliland and Ray Kazmer
7647 McGroarty Street
Tujunga, California 91042

TI-PD PUBLIC DOMAIN AND FAIRWARE

500 DISKS just $1.50 EACH! And orders for 8 or more disks are postpaid.

Thousands of programs selected from the best from the U.S., Canada, Australia, England, Germany, Holland and Belgium. FAIRWARE IS OFFERED BY AUTHOR'S WRITTEN PERMISSION ONLY. Disks as full as possible, arranged by exact category, BASIC programs converted to XBASIC, assembly programs with XBASIC loader, disks with autoloader by full program name.

Send $1.00 (deductible from first order) for 13-page catalog listing all programs and authors. Catalog also available on disk.

TIGERCUB SOFTWARE, 156 Collingwood Ave., Whitehall, OH 43213. v8/7

PASCAL COMPLETE SYSTEM	$149
TI RS232 CARD used	$115
FULL TI PE/BOX—RS232—32K—DRIVE	$299
EMPTY TI PE/BOX (used)	$110
18" TI PE/BOX EXTENSION CABLE	$25
EDITOR/ASSEMBLER PACKG (new)	$12
SPEECH SYNTHESIZER used	$35
PARALLEL PRINTER CABLE 6'	$19
PE-BOX TECH TRAINING MANUAL	$20
TI ORIGINAL COLOR MONITOR	$149
SERVICE MANUAL (CONSOLE/P BOX)	$25
4A FACTORY REPAIR MANUAL	$20
DISK CONTROLLER & 32K CARDS ea	$95
P-BOX "CARD" REPAIR MANUALS (ea)	$10
TI COLOR MONITOR SERVICE MANUAL	$15
SCHEMATICS/CARDS/CONSOL/PBOX ea	$5
CC40 (TI hand held computer)	$55
CC40 PERIPHERAL USER MANUALS	$10

USED TI99/4A, Hardware, Software, Books and Parts. Call or write for complete free list. S&H $3 min

JIM LESHER, 722 HUNTLEY
DALLAS, TEXAS 75214, 214 821 9274 8/5

HORIZON COMPUTER

HORIZON BARE BOARD, Manual + ROS8.14 $45
Zero K Kit=ALL parts, less Memory $105
128k Memory chips $45 each, 32k chips $8
128k Kit=$150 or $180 Built
256k Kit=$195 $225 Built
384k Kit=$240 $270 Built
512k Kit = $385 $415 Built
One Meg Kit=$465 $495 Built
1.5 Meg Kit=$645 $675 Built
ADD A RAMBO Mod for $45
256/800 PHOENIX Kit=$495 or $525 Built

RAMDISK

P-GRAM kit 72k = $150 or $180 Built
P-GRAM+ kit 192k= $230 $260 Built
CLOCK for P-GRAM's =$20
KITS Include ALL PARTS Needed

MEMory EXpansion for the GENEVE 9640
MEMEX 504k $245
MEMEX 504k+GENMOD $345
MEMEX 1008k+GENMOD $395
MEMEX 1512k+GENMOD $445
MEMEX 2016k+GENMOD $495
The GENMOD is ADDED to YOUR GENEVE 9640 card.

GENMOD allows the 9640 to address all 2 MEG on the MEMEX card at ZERO wait

Up old 180k to 256k w/instructions=$40
32/16 Console Mem Mod w/Supercart =$40
Ohio Residents add 6% sales tax
Ship OverSeas ADD $7 Surface or $10 AIR
Prices may change if MEMORY costs go up
Shipping FREE Within U.S. and Canada
AmEX + MasterCard + Visa ADD %10
Call 419-385-5946 or Send your order to
Bud Mills Services
166 Dartmouth Dr. Please include
Toledo Oh 43614 your PHONE #

Call TI-COMM BBS on 419 385 7484
for current prices or information
300 Baud,7bit,e / 1200,8,n

P Code Manual and Disk

The Boston Computer Society TI99/4A User Group is proud to announce the publication of a Pcode Manual for all you users of the Pcode card on your TI 99/4A. The manual is a compilation of the newsletter articles Ron Williams has written about using the P code language. It is 32 pages long (8.5 x 11), has a pretty orange cover, and is printed on 3 hole stock for your convenient use in a 3 Ring binder. The manual cost is $5.

Ron also makes a disk available that contains many of the programs in the manual, and many that are not in the manual. The cost of this disk is $3 + Shipping & Handling.

Make your check for $5 or $4 or $9 payable to:

Boston Computer Society, TI99/4A User Group, 1 Kendall Square, Cambridge, MA 02139-1562 (new address).

"BOOT PROGRAM"

Copyrighted by the Miami Users Group Feb. 1989. Not available from any other source or Mail Order Co.
Latest up-to-date version by the original author, John Johnson.
"BOOT" is in assembly language and uses the Horizon RAM Disk "MENU" program without a RAM Disk.
You will be able to:

1. View a file
2. Catalog a disk
3. Run a program, XB, EA or cartridges
4. Access any program on any drive
5. Even run XB programs over 42 sectors
6. Print any file or disk catalog

Send $7.95 to:
MIAMI USERS GROUP
YVETTE MCKENZIE
6775 TAMIAMI CANAL ROAD
MIAMI, FL 33126

8/2

The Chicago TI-99/4A Users' Group Presents:

THE CTIUG'S E.O.G.

(The Chicago TI-99/4A Users' Group's Encyclopedia of Graphics for the TI and the 9640 Home Computers)

Volumes #1 and #2

The CTIUG's E.O.G. is a desk top publishing project which will include a cataloging of ALL of the commercially distributed graphics in the 4A/9640 community.

The first volume, which received a "4 star rating" in the December 1990 edition of "MICROpendium Magazine," is a comprehensive compendium of ALL of the commercially distributed fonts which are available in the 4A/9640 community. Its 156 pages depict over 500 discrete fonts, and each is pictured in its entirety. A complete index allows the cross referencing and comparison of all fonts depicted. The pages are printed on single sides in order to facilitate the comparison of the various fonts.

Volume #2 is a compendium of ALL of the TIPS ("TI Print Shop") graphics. TIPS is clearly the largest single source of graphics for the 4A and
9640 computers, and Volume #2 provides a handy reference to these graphics and, as does Volume #1, it facilitates the performance of desk top publishing and graphics operations. (In order to keep our E.O.G. current, updates of additional material will accompany all subsequent releases of the E.O.G. These supplements will also be available separately.)

The cost of each volume is $10.00 (plus $3.00 for postage & handling). Please add an additional $6.00 for each manual which is to be shipped overseas.

Other CTIUG publications available include the CTIUG's "Hardware Manual" and the "TI Writer Supplement" and its accompanying companion disk included. Both were given a "4 star rating" and "A" evaluations by "MICROpendium Magazine" reviewers. They are available for $6.00 apiece (plus $2.00 for postage & handling). Please add an additional $4.00 for each manual which is to be shipped overseas.

Order both Volume #1 & Volume #2 of the CTIUG's E.O.G. for $23.00 (postage paid), or purchase all 4 publications for the special low price of $35.00 (postage paid). All publications are shipped 1st class.

Send all orders to:

Chicago TI Users' Group
P.O. Box 578341,
Chicago, IL 60657.
For more information, call our "hot-line" at
(708)869-4304.

```
THE CHICAGO TI-99/4A USERS' GROUP WOULD
LIKE TO SUPPLY YOU WITH MORE INFORMATION
ABOUT THE TI-99/4A SO YOU CAN DO ANYTHING
AND EVERYTHING THAT YOU HAVE EVER IMAGINED.
        MEMBERSHIP APPLICATION
DATE_____ PHONE # (___)_____
NAME _____
ADDRESS _____
CITY, STATE, ZIP _____
[] DISKETTE        [] CASSETTE TAPE
MEMBERSHIP: $21/USA,CANADA   $24/OVERSEAS
MAIL TO: CHICAGO-AREA TI-99/4A USERS'GROUP
         JAMES BROOKS, P.O.BOX 578341
         CHICAGO, ILLINOIS 60657
HOT LINE:  708-869-4304
```

TI-BASE USER NEWSLETTER

AT LAST! AN INFORMATION RESOURCE FOR OWNERS OF THE TI-BASE DATA BASE MANAGER! INCLUDES SIX BI-MONTHLY ISSUES OF AT LEAST TWELVE PAGES PER ISSUE, PLUS A GUARANTEE OF AT LEAST ONE DISK OF PROGRAMS DURING THE YEAR. FILLED WITH THE KIND OF INFORMATION NEEDED TO MAKE TI-BASE WORK FOR YOU. YOUR SUBSCRIPTIONS TO VOLUME 2 ARE BEING ACCEPTED NOW. VOLUME 1, WITH OVER 80 PAGES OF TI-BASE TUTORIALS, TRICKS, TIPS AND PROGRAMS, IS AVAILABLE NOW, FOR ONLY $22. A DISK OF VOLUME 1 PROGRAMS IS INCLUDED.

$20

TIMELINE 99

IF YOU ENJOYED THE ORPHAN CHRONICLES, THEN TIMELINE 99 IS FOR YOU. IT COVERS THE SIGNIFICANT EVENTS IN THE LIFE OF THE TI-99/4A AND 9640 AND THE COMMUNITY THAT HAS SUPPORTED THOSE COMPUTERS FROM THE EARLY DAYS, TO THE PRESENT. YOU WON'T FIND A MORE DETAILED CHRONICLE OF "THINGS TI" ANYWHERE. CHAPTERS INCLUDE;

- -THE BEGINNING,
- -THE MIDDLE,
- -THE END,
- -FAIRS AND FESTS
- -THE IUG
- -PUBLICATIONS
- -BOOKS
- -SOFTWARE
- -PERSONALITIES
- -TIME LINE

and more...

$18

MEMBERSHIP MANAGER &

NEWSLETTER EXCHANGE

THIS THREE-DISK PACKAGE IS DESIGNED FOR COMPUTER USER GROUPS THAT WANT TO UPGRADE THEIR MEMBERSHIP ROSTERS & NEWSLETTER EXCHANGE SYSTEM TO A TI-BASE ENVIRONMENT. BOTH APPLICATIONS PROVIDE TRUE RELATIONAL DATA BASE MANAGEMENT AND AUTOMATION CAPABILITIES TO YOUR TI OR 9640 SYSTEM RUNNING TI-BASE v3.0 OR HIGHER. PRICE INCLUDES ALL S/H CHARGES.

$25

PRODUCTIVITY+

PERSONAL AUDITOR: The most complete personal finance management package written for the 99/4A or Geneve. A must for users who need to know where the hard-earned dollars go every month. XB,32K disk.

$23

MICROdex 99: A completely independent data base that is dedicated specifically to the indexing of books, journals, magazines and newsletters. An excellent example of TI-99 relative file programming for the aspiring Extended Basic programmer. Many features like sorting, subfiles, indexing, querying etc are included. Fills a SS/SD disk.

$10

JUNCTION SOFTWORKS
2310 CYPRESS COURT
GRAND JUNCTION, COLO 81506

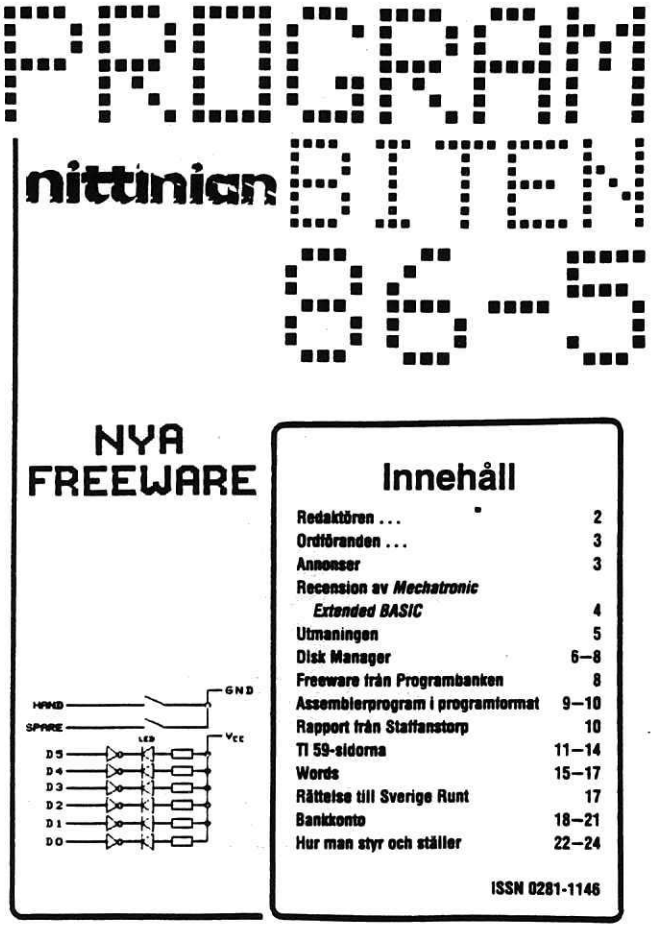
This advertisement created with PAGE PRO 99 by Ed Johnson.

Innehåll

ISSN 0281-1146

Innehåll

ISSN 0281-1146



Innehåll

ISSN 0281-1146

Du kan beställa äldre årgångar av Nittinian/Programbiten genom att betala in 50 kr per årgång på föreningens postgiro 19 83 00-6. PB 90 kostar 80 kr eftersom den består av 6 nummer. PB86-1 levereras kopierad på nitade lösa blad på samma sätt som PB89 och framåt.

NN 83-1, -2, -3, -4	50:-
PB 84-1, -2, -3, -4, -5	50:-
PB 85-1,1.5, -2, -3, -4	50:-
PB 86-1, -2, -3, -4, -5	50:-
PB 87-1, -2, -3, -4	50:-
PB 88-1, -2, -3, -4	50:-
PB 89-1, -2, -3, -4	50:-
PB 90-1, -2, -3, -4, -5, -6	80:-

Du kan även beställa samlingskivor med program från varje årgång som kostar 30 kr per skiva och årgång.

Porto ingår i alla priser.

PROGRAM nittinian BITEN 87-1

GRAM KRACKER

GENEVE

Innehåll

ISSN 0281-1146

PROGRAM nittinian BITEN 87-2

Teckensmeden

Texteditor i C99

42 SPELET

Innehåll

ISSN 0281-1146

PROGRAM nittinian BITEN 87-3

MATRIKEL NUMMER

ALFA SLANG

SOUTH PACIFIC

Innehåll

ISSN 0281-1146

PROGRAM nittinian BITEN 87-4

Checklist

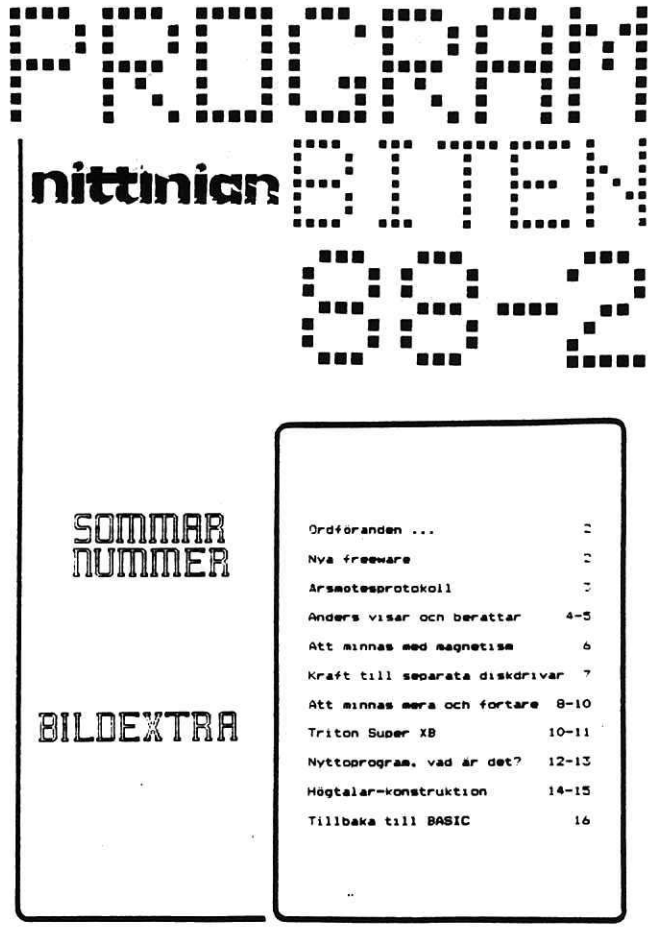
BONGO SOFT. PRESENTS PLUS

Mer om RAM-diskar

Innehåll

ISSN 0281-1146

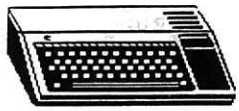

PROGRAM
nittinian BITEN
88-1
Inför årsmötet
GENEVE 9640
Innehåll
Redaktören 2
Kallelse till årsmötet 3
McGoverns XB-skola 4
Superbug II 5
Menu version 7.1 6
Star-recension 7—8
Myarcs diskkontrollkort 8
Geneve 9—14
TI-74 BASIC 15
EDP — en recension 16
Variabelöverföring 16—17
Myarc Extended BASIC 18—20
ISSN 0281-1146

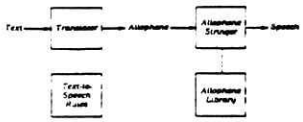

PROGRAM
nittinian BITEN
88-2
SOMMAR NUMMER
BILDEXTRA
Ordföranden ... 2
Nya freeware 2
Årsmötesprotokoll 3
Anders visar och berättar 4-5
Att minnas med magnetism 6
Kraft till separata diskdrivar 7
Att minnas mera och fortare 8-10
Triton Super XB 10-11
Nyttoprogram, vad är det? 12-13
Högtalar-konstruktion 14-15
Tillbaka till BASIC 16


PROGRAM
nittinian BITEN
88-3
Nya kort
MATRIKEL NUMMER
PRK på djupet
Innehåll
Ordföranden 3
Rapid Copy 4
PRK-BASIC 5–8
Nya kort 8
Tips och tricks 9–10
Bokrecension 10
Matrikeln 1988 11–14
Utmaningen 15–17
Programbanken 18–19
Hårddiskkort 19
Dubbelspelet 20–21
Satellitbanor 22
Hårdvarumanual 22
Tips from the Tigercub 23–24
ISSN 0281-1146

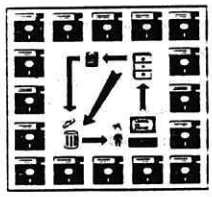

PROGRAM
nittinian BITEN
88-4
80-kolumnskort
TI-BASE
Myarc HFDC
Innehåll
Redaktören ... 2
TI-BASE 3
Myarc HFDC-kort 4–6
Katalog från L.L. Conner 6
DIJIT AVPC-kort 7–8
Archiver III 8
Statistics BASIC 9–11
Horizon RAM-disk 12–13
Bud Mills services 13
"TEX" ordbehandlare 14–15
RS232-interface 16
ISSN 0281-1146

PROGRAM BITEN 89-1





TI-BASE



FOR ALL YOUR DATA NEEDS

ISSN 0281-1146

PROGRAM BITEN 89-2

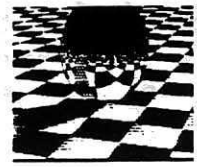

```
FW=4=13
FIL=37 LED=15 ANV=705
filnamn     sekt typ          längd
-READ-ME      22 DIS/VAR         80
AS            33 PROGRAM       8192
AT            22 PROGRAM       5304
C99PFI:O       2 DIS/FIX         80
CF            32 PROGRAM       7850
CG            25 PROGRAM       6060
CHARA1         5 PROGRAM       1024
CHARA2         5 PROGRAM       1024
CP             4 PROGRAM        545
CTSRAM         9 DIS/FIX         80
DP            18 PROGRAM       4138
DS             4 PROGRAM        542
EA             9 PROGRAM       1860
ED            33 PROGRAM       8192
EE            19 PROGRAM       4442
FMSAVE         6 DIS/FIX         80
FO            33 PROGRAM       8192
FP            15 PROGRAM       3566
FWDOC/EASM    33 DIS/VAR         80
FWDOC/LOAD    38 DIS/VAR         80
FWDOC/REPT    55 DIS/VAR         80
FWDOC/TIWR    25 DIS/VAR         80
FWDOC/UTIL    63 DIS/VAR         80
FWSAVE         6 DIS/FIX         80
LDFW           9 DIS/FIX         80
LH            16 PROGRAM       3836
LL            10 PROGRAM       2064
LOAD          31 PROGRAM       7661
MG            33 PROGRAM       8192
MH            20 PROGRAM       4856
QD            11 PROGRAM       2512
SAVIT          2 DIS/VAR         80
SL            10 PROGRAM       2240
SYSCON         6 PROGRAM       1198
UL             4 PROGRAM        542
UTIL1         33 PROGRAM       8152
XB4THLD        2 PROGRAM        203
(delad på 2 skivor i progr-bank)
```

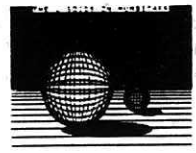

ISSN 0281-1146

PROGRAM BITEN 89-3

ISSN 0281-1146

PROGRAM BITEN 89-4





ISSN 0281-1146