PROGRAM BITEN 94-1

ISSN 0281-1146

REDAKTÖREN

Det har nu gått tio år sedan TI slutade med hemdatorn TI-99/4A. Nedläggningen meddelades fredagen den 28 oktober 1983, av TI-ägare även kallade den svarta fredagen.

Jag skaffade min 99:a den 15 januari 1983 och expansionsbox den 18 oktober 1983 strax före nedläggningen. Sedan dess har jag skaffat ytterligare tillbehör och monterat en ny tystare fläkt i expansionsboxen. Den är nu full med följande kort (slot nummer):
1 - Flex interface
2 - Myarc RS232/PIO med DIJIT EPROM
3 - Horizon 4000, 512 kB + 32 kB EM
4 - DIJIT AVPC 80 kol 9938, 192 kB
5 - Myarc HFDC med EPROM H11
6 - TI Diskkontrollkort
7 - P-GRAM+ 192 kbytes med klocka
8 - Horizon RAM-disk 192 kbytes

Jag har en tom PC/XT-Box med 160 W (5V-9A, 12V-5A) för de skivenheter som inte får plats i den vanliga expansionsboxen. Totalt har jag sex platser för halvhöjds skivenheter. TI-kontrollkortet är anslutet till två stycken Fujitsu DS/SD 180 kbytes (DSK1-2). Myarc HFDC är anslutet till Seagate ST-125 hårddisk 20 Mbytes (WDS1), TEAC DS/DD 360 kbytes (DSK5) och två Mitsubishi DS/QD 720 kbytes 5,25 tum (DSK7-8).

Jag använder en Philips CM8833 färgmonitor (50/60Hz, 0.42mm, 12 MHz) och en NEC Pinwriter P6 skrivare (24 pinnar). Det finns även TI Joystick par och WICO Joystick samt Speech Synthesizer. Jag har en Commodore 1352 Amiga buss-mus och ett modem med 300 V.21, 75/1200 V.23, 300 BELL103.

Kom ihåg ÅRSMÖTET Lördagen
5 mars 1994 kl.13 hos
Kent Edgardh (08-777 2944)
Albatrossvägen 46, Haninge
(se dagordning i PB 93-4)

Asgard Software har ny adress och ny ägare (endast program på flexskiva): Harry Thomas Brashear, 2753 Main St., NEWFANE, NY 14108, USA. Den tidigare adressen i Woodbridge gäller endast hårdvara och moduler.■

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

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

Postgiro 19 83 00-6
Medlemsavgiften för 1994 är 120:-

Datainspektionens licens-nr 82100488

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

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

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

Användartips med Mini Memory	20:-
Nittinian T-tröja	40:-
99er mag. 12/82, 1-5,7-9/83(st)	40:-
Nittinian årgång 1983	50:-
Programbiten 84-89 (per årgång)	50:-
90-93 (per årgång)	80:-
TI-Forth manual	100:-
Hel diskett ur programbanken(st)	30:-

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

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

PUTTING IT ALL TOGETHER No. 9

by Jim Peterson, Tigercub, USA

The hard part of learning to program is not in learning what the various commands do - it is learning how to put them together to do what you want them to do! Key in this little program and run it to see what it does, then study the explanation of how it does it.

```
1 !STRAIGHT-LINE CALCULATOR
   TINYGRAM by Jim Peterson
   Accepts input such as
   6+6-11*2+3/4
2 T,F=0 :: C$="+-*/" :: ACCE
PT AT(12,1)ERASE ALL VALIDAT
E(NUMERIC,C$):F$ :: L=LEN(F$
):: FOR J=1 TO L :: X$=SEG$(
F$,J,1):: P=POS(C$,X$,1):: I
F P=0 THEN 5
3 IF F=0 THEN T=VAL(SEG$(F$,
1,J-1)):: F=1 :: A=J+1 :: P2
=P :: GOTO 5
4 V=VAL(SEG$(F$,A,J-A)):: A=
J+1 :: GOSUB 7 :: P2=P
5 NEXT J :: V=VAL(SEG$(F$,A,
255)):: GOSUB 7 :: DISPLAY A
T(12,L+1):"=";STR$(T)
6 DISPLAY AT(24,1):"PRESS AN
Y KEY" :: CALL KEY(0,K,S)::
IF S=0 THEN 6 ELSE 2
7 IF P2=1 THEN T=T+V ELSE IF
 P2=2 THEN T=T-V ELSE IF P=3
 THEN T=T*V ELSE T=T/V
8 RETURN
```

The calculations are done from left to right, not in the mathematical hierarchy of multiplication and division first.

The variables T and F are reset to 0 because program execution returns here. A string of math symbols is placed in C$. The calculation is accepted into F$, using ERASE ALL to clear the screen; the VALIDATE will accept only numeric characters (numerals and decimal point) and the symbols assigned to C$. L measures the length of the string. The J loop examines the characters in the string, from the first to the last, extracting one character at a time into X$. POS checks whether that character is the 1st, 2nd, 3rd or 4th character of the C$ "+-*/" and places that value in P, or a 0 if it does not match any of them. In this case, X$ was a numeric character so execution jumps to NEXT J to continue the loop.

Otherwise, the first math symbol in the string has been found. F (a flag variable) still equals 0 so VAL converts the part of F$ from the first character up to the math symbol into its numeric form, in T. The flag F is set to 1 so that line 3 will be skipped over from now on. The position of the first character after the math symbol (the beginning of the next number) is saved in A and the value of P (the position of the math symbol in C$) is saved in P2. The loop continues, finding the digits of the next number, until another math symbol is found. F does not equal 0 so execution jumps to line 4. The segment of F$ starting from the position saved in A, to J-A (the character preceding the current math symbol) is converted to numeric by VAL and placed in V. The position to start looking for the next number is again saved in A. The GOSUB jumps to line 7. Depending on the position in C$ ("+-*/"), saved in P2, of the previously found math symbol, the value of this second number, saved in V, is added to, subtracted from, multiplied by or divided into the previous number saved in T, and the new value is saved in T. Execution then RETURNs to the last statement in line 4, to save the value of P (the location in C$ of the current, not yet used, math symbol) in P2, and the loop continues.

When the loop is completed, in line 5, the value of the final numeric characters is determined, the GOSUB again uses the value saved in P2 to determine the final calculation, and the result is printed out. Since the original input was in row 12, column 1, and the length of the input was saved in L, L+1 places the "=" directly after it, and converting the value T into a string by using STR$ causes it to print directly thereafter without an intervening space.

If S (status) in the CALL KEY is 0, it means that no key was pressed, so the line is repeated; otherwise, execution goes back for another input. ■

Tidigare avsnitt av "Putting it all together" har publicerats i Programbiten: Putting No.5 se PB 90-5.28 och Putting No.8 se PB 92-1.03. ■

SWEDLOW TI BITS * 32-33 *

by Jim Swedlow, USA

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

I had occasion to use two features of TI Writer that were interesting, so I though I would pass them along.

INCORPORATING INSTANCES

This will be very brief, because Bill Nelson, Graphs Guru Extrodinare, did most of the work. He designed two instances, which, together, formed a letterhead. Bill then used Rodger Merritt's PICTURE IT to convert them into a TI Writer compatible file.

The file is filled with transliterate (.TL) commands. It accesses the graphics mode of your Star Gemini or Epson compatible printer. I had two problems.

First, the file would not work with FUNNELWEB. The computer locked up before printing the first line. A quick call to Bill and I learned the fix - go back and use real TI Writer. Haven't used TI Writer in ages but I found the cartridge and disk, changed the printer name to PIO.CR and every thing worked fine - except for the second problem.

I wanted to use the letterhead with TI Writer's mail merge capability. But I couldn't. The PIO.CR printer name in the Formatter is fine for the converted instances, but not for a text file.

Bill explained a solution that involved saving the file to disk and then printing it from there. I opted for a simpler option - two pass printing.

I ran my paper through the first time to print the letterhead (using PIO.CR) and then a second time to print the text (using PIO.LF). It worked.

MAIL MERGE

When would you use mail merge?

o You want to write that annual Christmas letter, but, instead of photocopying, you want to personalize each one.

o You have a business and you want to send individualized letters to your customers (". . . yes, right there in Garden Grove, you can use . . .").

o You are planning a conference, a party or a Fest and you want to send individualized invitations.

What to do? You could write your letter and then change key information for each person. Better yet, you could use mail merge and let your computer do the work.

Let's try a Christmas letter. Perhaps it might run like this:

Dear ___,

It has been an exciting year for us. Junior was accepted to Yale, Mary had straight A's and Bud stole his fifteenth truck.

How are things with you ___? Write soon and let us know.

Love,

We have two variables. The first is the name (Annie, Aunt Susie, Grandpa, etc.). The second is family members (Bueford; Uncle Sam and little Quincy; or, perhaps, no one).

Here are the steps you must take to use mail merge.

1. Write your letter. Where you want to have a variable, identify it with the Alternative Input symbol *n*. The "n" between the asterisks stands for the field number (*1*, *2*, *3*, etc.). Your mail merge ready letter looks like this:

```
.FI;AD
Dear *1*,

It has been an exciting year for us.
Junior was accepted to Yale, Mary
had straight A's and Bud stole his
fifteenth truck.

How are things with you*2*? Write
soon and let us know.

Love,
```

Note that there is no spaces between "you", *2* and "?". If Grandpa lives alone, you will want it to read "How are things with you?" Whereas for Annie, you would say "How are things with you and Bueford?".

Remember to include the Fill and Adjust commands (.FI;AD) and to save your file before going on to the next step.

2. Create your value file. Here is a sample:

```
1 Grandpa
2
*
1 Annie
2 ^and Bueford
*
1 Aunt Susie
2 , Uncle Sam and little Quincy
*
```

Each line must end with a carriage return. There must be one space between the field numbers and the beginning of the text. There must be one space and then a carriage return after each asterisk.

If the field is empty (as in the second field in the first letter) nothing will print. To make an empty field, type the field number, one space and then a carriage return.

Note the required space (^ or SHIFT 6) before "and Bueford". Because there is no spaces between "you", *2* and "?", we have to tell the Formatter to insert the leading space. Otherwise, it will print "How are things with youand Bueford?".

Save this file using another name. For example, if you called your letter DSK2.LETTER and your could call your value file DSK2.NAMES (original, huh?).

3. You are now ready to merge your two files into individualized Christmas letters. Load the Formatter and use DSK2.LETTERS for the input file name. When you get to this question:

```
USE MAILING LIST? N
```

type Y and press ENTER. When the Formatter asks you:

```
MAILING LIST NAME:
```

type in your file name (in our example, DSK2.NAMES) and press ENTER.

TI Writer will print a letter for each entry in your value file (in our case, three letters).

There are more tricks with mail merge but this will get you started.

TI SURVIVAL NET

Do you subscribe to any TI related publications? If not, you should. They are excellent sources for information on new products, updates, support and trends in the TI world.

This publication is TI-only:

MICROpendium
PO Box 1343
Round Rock, TX 7868

The only major computer magazine that still supports the TI is:

Vulcan's Computer Monthly
PO Box 7062
Atlanta, GA 30357-0062
Cost: $15.95 a year

Vulcan's carries Barry Traver's column. If you subscribe to Vulcan's, tell them that you are a TI 99/4A owner and proud of it!

One more to complete the list, Barry's diskazine, Genial TRAVelER.

Every so often you get a disk (sometimes two!) from Barry that is filled with wonderful stuff.

Genial Computerware
835 Green Valley Drive
Philadelphia, PA 19128
Cost: $36 for six issues

The Bottom Line: If you do not subscribe to one or more of these, you are making two fatal errors. First, you are depriving yourself of some of the best available TI support. Second, you are speeding the demise of the TI survival net.

The people who produce these fine publications are not getting rich. They do it because they love the TI. If you don't let them know that you are still a TI'er with your check book, sooner or later they will move onto greener fields.

BTW: These rates are as of October, 1990 and are, as they say, subject to change without reason or notice.

A NEWT'ism
Seen on the BBS and attributed to Cost Compute:

The only difference between a guru and an expert is that the guru reads the manuals.

MORE FROM THE BBS

Kevin McAlleavey of Selkirk, New York, left this message on the BBS under the title of "I WANT TO WHINE!"

(sniff, whimper) . . . After all these years, I remain a loyal TI-ONLY user and am getting VERY depressed watching the last lifeblood resources for my darling die off. Yes, I am familiar and fluent in MS DOS owing to my being forced to deal with the critter at my job site. After playing with a 386, and having toyed with a briefly operational 9640, yes, it's slow as molasses in a bitter New York winter, but I just can't bring myself to separate myself from my little black and silver "mamita espanol". I have heard much whining about the humble yet "mas fuerte" 99/4A being a dinosaur, but this darling and I have always been lovers and I cannot live without "her". My apologies for my whining session (sniff), but I don't wanna see everyone abandon this delight.

Whine on, Kevin!

QUESTION

What have you done recently to prolong support for the 4A? What will you do next? Think about it.

Enjoy! ■

DSKU REFUSES TO BOOT FW

by Charles Good, Lima Ohio UG, USA

DSKU v4.2 which was distributed by the Lima User Group as part of its Funnelweb v4.40 and earlier Funnelweb v4.31 distribuiton. There is an item on the main DSKU menu that says "Load FW". It usually doesn't work. The reason is that DSKU searches the drive you specify for a file named UTIL1, which is what the main Funnelweb file used to be called. The main Funnelweb file is now called FW.

It is easy to modify DSKU to boot file FW every time you ask DSKU to "Load FW". Here's how. Use Funnelweb's DISK REVIEW or other sector editor to search the third DSKU file (named either DW or DSKW) for the ASCII text "UTIL1". You will find "DSK1.UTIL1" Change the UTIL1 to "FW" and put blank spaces over the IL1. Then change the screen display to Hex (CTRL/H if you are using Funnelweb's DISK REVIEW) and move the cursor backwards (to the left) to the first appearence of "0A". This is at byte >DD in my file DW. Change the 0A to 07 and write these changes back to disk (CTRL/W and then CTRL/A if you are using Funnelweb's DISK REVIEW). This change shortens the length of text the computer expects to find, since DSK1.FW is shorter than DSK1.UTIL1. DSKU will now properly boot Funnelweb when you select "Load FW" from DSKU's main menu. ■

THE 99/4 HOME COMPUTER

by Charles Good, Lima Ohio User Group, USA (Feb 1991)

TI began shipping the 99/4 (copyright 1979 on the color bar title screen) in October 1979. It cost $1150 bundled with a 13 inch color monitor (FORTUNE, December 3, 1979, p.54). Initially you had to take the monitor and could not purchase the 99/4 separately, and most purchasers had to pay close to full price. Bundling was necessary because the 99/4 console passed but TIs TV modulator initially failed to pass FCC lab tests for noninterference with radio and TV broadcast reception. The modulator emitted too much RF radiation (BUSINESS WEEK, March 19, 1979, p.37). However, at that time the FCC did not regulate RF radiation from computing devices not hooked directly to TVs. So TI got around the FCC regulations by offering to the public a "complete package". It wasn't until January 1, 1981 that the FCC began testing ALL computers likely to be used in a home environment for TV/radio broadcast interference (POPULAR COMPUTING, November 1981, p.6). TI eventually came up with a TV modulator that would pass FCC tests and on November 28, 1980 began selling the console and monitor separately. The console's list price was $650 (BUSINESS WEEK, December 8, 1980, p.28). This was in one respect actually a price increase, because the separate prices of the console and monitor were $250 more than their previous bundled price.

TI never published any sales data for the 99/4, but an independent market research firm estimated that TI would sell 25000 between its introduction and the end of 1980 (FORTUNE, June 16, 1980, p.139). During the summer of 1981 TI quietly introduced the 99/4A with a list price of $525. By the time production of the 99/4A ceased in late 1983 or early 1984 the store price for a brand new 99/4A was $50, and over 1 million, perhaps several million 99/4As had been sold.

DIFFERENCES BETWEEN 99/4 AND 99/4A

The most obvious differences are the keyboard, the lack of lower case letters on the "4", and the "4"s EQUATION CALCULATOR. Most "4"s have an earphone jack on the front for private listening, but mine doesn't. I will discuss most of these obvious differences in detail. The 4A gets its "A" from the fact that it has a 9918A video processor, whereas the 99/4 has a 9918 video processor. The 9918A has bit map mode (Graphic2), which is not found on the 9918 processor. This means that any software that uses bit map mode will not run on the 99/4. Other differences between the 99/4 and 99/4A (such as the "4"s lack of an XOP assembly directive) are referenced in the index of the Editor/Assembler manual (p.456) under the heading "Computer differences".

In general, all software written for the "4" will run on the 4A. Some complicated routines on the 4A were required to achieve this compatibility. The "4" has 256 bytes more free memory in TI BASIC than the 4A, so some BASIC software written on a "4" may not work on an unexpanded 4A. Lots of assembly or GPL software written for the 4A will NOT work on the "4", and there is no easy way to upgrade a "4" to a 4A. The Mini Memory module and its line by line assembler, and the E/A module and its editor and assembler work OK on the "4". A partial list of "won't work on the 99/4" software includes TI-Writer, Multiplan, Funnelweb v4.x, the LINES program that comes with the Mini Memory module, all the Milton Bradley game modules that were created to accompany the MBX system, Word Invasion, Parsec, Story Machine, Alpiner, Dragon Mix, and Word Radar. Most of these modules and the LINES program are probably incompatible because they use bit map mode. There are probably other reasons for the incompatibility of Multiplan, TI-Writer, and Funnelweb. Even the non-editor parts of Funnelweb won't

work on the "4". When you boot Funnelweb into the "4" using the extended basic module, the title screen shows blanks where there should be lower case letters. You can then go to Funnelweb's extended basic user list, but here the "4" locks up. You can't boot any software from the XB user list.

THE KLUDGY 99/4 KEYBOARD

After playing around with my "4" for a couple of months, I am forced to agree with the statement made in an accompanying FORTUNE magazine article. The 99/4 is a real dog, mainly because of its keyboard.

There are 41 "chicklet" style keys, each slightly contoured and shaped like a narrow rectangle. The 4A keyboard has 48 keys. Although each 99/4 key depresses separately, the keys are not what experienced users would call "full travel" There is no tactile response, no click, before the keys suddenly bottom out at the end of their downward travel. Non-alphanumeric keys include one (and only one) SHIFT, an ENTER, a SPACE bar, and a SPACE key immediately to the left of the "A" key. Alpha keys always produce upper case letters, so the SHIFT key is not used as often as it is on the 99/4A. There is no ALPHA LOCK, FCTN, or CTRL keys on the "4". The "4"s SPACE key and bar do exactly the same thing, leave a blank space. I can see no reason at all for this space KEY, in addition to the normally positioned space bar. There are ASCII characters built into the 99/4 console that are not implemented on its limited keyboard, yet there is this stupid extra space key.

Touch typing on the 99/4 is difficult. The keys are spread apart the same distance as on the familiar 99/4A keyboard, so it is possible to get all your fingers at once onto the keys. But the small vertical size of the keys and their lack of tactile feel makes touch typing difficult. The small size and minimal contour of the "4"s keys makes it difficult for a touch typist to find by feel and seat his or her fingers in the center of the desired keys as the fingers move blindly around the keyboard. The fully contoured much larger keys of the 4A (larger because there is less space between keys) makes touch typing much easier. A special problem to experienced touch typists is the lack of any key to the right of the "L". This means there is no "home" key for the little finger of the right hand to touch, and this will drive most touch typists crazy. Frequently, when I try to type on my "4" I end up accidently moving my fingers over one key to the left on the home key row so that all ten fingers have something to touch. My left hand pinky finger is then on the useless SPACE key instead of on the "A" where it should be. Then I type rtow fevfw. TI recognized this problem. The only application software written for the 99/A that is likely to require touch typing, the Terminal Emulator II, has a keyboard overlay with a raised area creating a fake key for the right hand's little finger.

TI provided a series of overlays specifically for use with the 99/4 and not usable with the 4A. Some overlays were packaged with the "4" and others were available with specific command modules. Because of the narrow vertical size of each key there is enough room between rows of keys on the "4" to display a text prompt immediately above ANY key, not just above the numeric keys as is the case with the 4A. The overlays have text prompts for special keypresses, and cover the entire "4" keyboard, with the keys sticking up through holes in the overlay. Special keypress usually involve using the SHIFT key in combination with a letter key. One overlay packaged with the "4" shows the editing keys used in BASIC. SHIFT/Q=quit. SHIFT/W=begin. SHIFT/ESDX= arrows. SHIFT/R=redo. SHIFT/T=erase. SHIFT/A=aid. SHIFT/F=delete. SHIFT/G=insert. SHIFT/Z=back. SHIFT/C=clear. SHIFT/V=proceed. There is nothing intuitive about some of these keypresses (why not SHIFT/B instead of /Z for back), so the overlay is really needed. Another overlay packaged with the "4" shows the split keyboard keys that can be used with

some games to simulate the 8 positions of joysticks #1 and #2. In addition to the overlays packaged with the computer, I have seen overlays designed for use with the following command modules: Terminal emulator I, Terminal emulator II, Video graphs (PHM3005), and Video Chess. There may be other overlays I havn't seen.

One of the reasons I give the 99/4 my "real dog" rating is the uncontrollable multiple repeat of the keys on my "4"s keyboard. This makes it almost impossible to do any useful typing, touch or hunt and peck, on my "4". Autorepeat of all keys at rate of 12 characters per second after a 1 second delay is listed as a NEW feature of the 99/4A (99ER MAGAZINE, Vol 1 #2, July/ August 1981, p.48). Autorepeat is NOT described in TI literature as a feature of the "4". On my "4" any of the keys are likely to repeat INSTANTLY. When you depress a "4" key, the keypress registers in the memory of the computer at a point about 1/2 way down the travel of the key. There is no tactile response that this has occured. The only thing your finger feels during a keypress is the sudden stop when the key bottoms out. If the key hovers in this "1/2 way down" region you get mmmultiiiplle displays of theee keeey on the scrrreennn. Try as I might, I can't seem to avoid this. My "4"s keyboard is very sensitive. Other experienced 4A users who have tried my "4" all have the same problem. Having to use backspace (SHIFT/S) and delete (SHIFT/F) after every 6-10 keystrokes gets old really fast. It has been suggested to me that this problem may be related to the ageing of my "4". The condition may not have existed when my "4" was built. One collector of TI computer products told me, "I had a 99/4 that did that. I got rid of it and replaced it with a 99/4 that still works fine."

ONLY UPPER CASE LETTERS

No keypress on the "4" keyboard will give ASCII codes 97-122, the lower case letters. Everything you type is in upper case, and this means you only use the SHIFT key in routine typing to shift the numeric keys and display !@@#$%^&&*(). The 99/4 uses a 5x6 pixel grid to display upper case letters. The 99/4A uses a 5x7 grid to display both upper case and lower case text. If you load into the "4" BASIC software written on a 4A that includes lower case text, the program seems to work OK, but no lowercase letters are displayed on screen.

THE EQUATION CALCULATOR

When you PRESS ANY KEY TO CONTINUE from the color bar powerup screen of the "4", you get a menu with three choices. Press 1 for TI BASIC, 2 for EQUATION CALCULATOR, 3 for TITLE OF COMMAND MODULE. (The rest of this description, see the full article in BITS,BYTES&PIXELS Feb 1991).

CONCLUDING REMARKS

When it was released in 1979 the 99/4 was the only consumer device that could really be called a "Home Computer". It was the first to utilize cartridge software. Its speech synthesis was, and still is, unequaled. It was easy to use, easy to program in BASIC, and it was powerful. Its high price was probably the major reason for its initially limited sales. Its rotten keyboard didn't help either. I'm sure glad we now have the 99/4A. The 4A is much superior to the "4".■

TRANSLITTERERA MED EXTENDED BASIC

av Jan Alexandersson

Du kan byta t.ex. 64 mot 144 med CALL TL(TEXT$,64,144) i texten.

```
9000 SUB TL(TEXT$,CH1,CH2)
9001 ! Transliterates CH1 to
 CH2 in TEXT$
9010 CH$=CHR$(CH1)
9020 CH2$=CHR$(CH2)
9030 IF POS(TEXT$,CH$,1)=0 T
HEN 9060
9040 TEXT$=SEG$(TEXT$,1,POS(
TEXT$,CH$,1)-1)&CH2$&SEG$(TE
XT$,POS(TEXT$,CH$,1)+1,80)
9050 GOTO 9030
9060 SUBEND
```

■

FROM BASIC TO ASSEMBLY - 7

by Bob August, Bug News, USA

Last month we said we would show you an easier way to put the window on the screen and use a lot less code. We still stayed with ASCII 49 through ASCII 56 to make our window. However, This month we loaded 64 bytes into Register two instead of 8 bytes at a time. We still start at the pattern table memory location >0988 which is the location for the start of ASCII 49. We also used text to put our window on the screen along with our text. This means we only write it to the screen one time instead of nine times. We could save more code by not using the BL (GOSUB) to put our window on the screen. For a little practice, re-write the program clearing the screen without a Branch Link and writing to the screen without a Branch Link. Go back to lesson number one if you need help.

The Extended Basic version of our window program is listed below:

```
100 ! Lesson Number 7
110 GOSUB 240
120 CALL CHAR(49,"FFFFC0C0CF
CFCCCC")! 1 = LEFT TOP CORNE
R
130 CALL CHAR(50,"FFFF0000FF
FF0000")! 2 = TOP LINE
140 CALL CHAR(51,"FFFF0303F3
F33333")! 3 = RIGHT TOP CORN
ER
150 CALL CHAR(52,"CCCCCCCCCC
CCCCCC")! 4 = LEFT SIDE
160 CALL CHAR(53,"3333333333
333333")! 5 = RIGHT SIDE
170 CALL CHAR(54,"CCCCCFCFC0
C0FFFF")! 6 = LEFT BOTTOM CO
RNER
180 CALL CHAR(55,"0000FFFF00
00FFFF")! 7 = BOTTOM LINE
190 CALL CHAR(56,"3333F3F303
03FFFF")! 8 = RIGHT BOTTOM C
ORNER
200 GOSUB 260
210 CALL KEY(0,K,S):: IF S=0
 THEN 210
220 IF K<>13 THEN 210
230 STOP
240 CALL CLEAR
250 RETURN
260 DISPLAY AT(8,1):"1222222
22222222222222223":"4
                     5":"4
 PRESS ENTER KEY TO QUIT  5"
270 DISPLAY AT(11,1):"4
                    5":"677
77777777777777777777778"
280 RETURN
290 END
```

Both the above program and the assembly program will display a window in the middle of the screen with the message in the center of the window the same as last month. (Lesson 8 see PB 91-6 p.28)

HAPPY ASSEMBLING!

```
****************************************
* BASIC TO ASSEMBLY   Lesson Number 7  *
****************************************
*
       DEF  START             Entry point of program
       REF  VSBW,VMBW,KSCAN   Utilities used in program
*
WRKSP  BSS  32                Workspace buffer
SAV11  BSS  2                 Save return address buffer
*                                    ASCII Number:
WINDOW DATA >FFFF,>C0C0,>CFCF,>CCCC  1 = Left top corner
       DATA >FFFF,>0000,>FFFF,>0000  2 = Top line of window
       DATA >FFFF,>0303,>F3F3,>3333  3 = Right top corner
       DATA >CCCC,>CCCC,>CCCC,>CCCC  4 = Left side of window
       DATA >3333,>3333,>3333,>3333  5 = Right side of window
       DATA >CCCC,>CFCF,>C0C0,>FFFF  6 = Left bottom corner
       DATA >0000,>FFFF,>0000,>FFFF  7 = Bottom line of window
       DATA >3333,>F3F3,>0303,>FFFF  8 = Right bottom corner
```

```
*
MESAGE TEXT '  12222222222222222222222223  '
       TEXT '  4                          5  '
       TEXT '  4 PRESS ENTER KEY TO QUIT  5  '
       TEXT '  4                          5  '
       TEXT '  67777777777777777777777778  '
*
       EVEN              Make sure we start on even byte
*
* Start of program
*
START  MOV  R11,@SAV11   Save return address
       LWPI WRKSP        Load the workspace
       BL   @CLEAR       GOSUB CLEAR to Clear the screen
*
* Put window INTO MEMORY
*
       LI   R0,>0988     Load pattern table starting at one
       LI   R1,WINDOW    Load window data
       LI   R2,64        8 lines of data times 8 bytes per line
       BLWP @VMBW        Write it to VDP
*
* Put message and window on the screen
*
       BL   @DISPLY         Gosub to display routine
       DATA 224,MESAGE,160  Screen location, Text, length
*
* Call key routine
*
       CLR  @>8374       Clear to zero for CALL KEY(0,K,S)
       CLR  @>837C       Clear status to zero
       LI   R4,>2000                          (Ed.change)
KLOOP  BLWP @KSCAN       CALL KEY(0,K,S)
       CB   @>837C,R4    Check for key press  (Ed.change)
       JNE  KLOOP        IF S=0 THEN KLOOP    (Ed.change)
       MOV  @>8375,R0    Move Key press to register zero
       CI   R0,>0D       Compare to 13 or enter key
       JNE  KLOOP        If not enter key, goto KLOOP
       CLR  @>837C       Clear status to zero
       MOV  @SAV11,R11   Put return address in register 11
       BLWP @0           Quit (FCTN =)
*
* Clear screen routine
*
CLEAR  LI   R1,>2000     Load Register one with space
       CLR  R0           Clear Register zero to zero
CLOOP  BLWP @VSBW        Write blank space to screen
       INC  R0           Add one to register zero
       CI   R0,767       Compare contents to 767
       JLE  CLOOP        If less then 767, goto CLOOP
       RT                Return to next line of calling area
*
* Display at routine
*
DISPLY MOV  *R11+,R0     Put screen location into Register zero
       MOV  *R11+,R1     Put message into Register one
       MOV  *R11+,R2     Put length into Register two
       BLWP @VMBW        Write it to the screen
       RT                Return to next line of calling area
*
* End program with auto start
       END  START                                          ■
```

I LIKE BRAIN GAMES!

by Jim Peterson, Tigercub, USA

I don't much care for those fast-action arcade type games - the dodge-the-pacman, climb-the-ladder, shoot-the-alien type of thing. My grey-haired reflexes are too slow, and my 8-year old grandson can play rings around me.

And I HATE those adventure games that do nothing but print out responses that "I don't know how to do that" or "you can't go thataway". Sounds too much like the SYNTAX ERROR or BAD VALUE messages that I get when I'm trying to write a program!

But I do like brain games! - the ones that challenge me to exercise the grey cells under my grey hair, and give me plenty of time to do so. I also enjoy programming that type of game - although they have certainly proven to be the least popular of anything I have ever done.

The world's premier brain game, of course, is chess. I can't comment much on that, because I don't know the game - other than the wild Japanese version, where every piece that reaches enemy territory can be promoted and every captured piece can be placed back on the board as your own. I wish that someone would program that game!

Anyway, Western-style chess is available as an old Texas Instruments module and as a public domain program translated by Swiridenko from a version written for some other computer. From reviews, I understand that neither offers much of a challenge to an expert, but that either one is a worthy opponent for an average player.

There are also a couple of TI computer games based on chess. The Queen Board Game, public domain by D. Decker, is a real challenge. Hexapawn is an early computer classic from Ahl's days; the computer starts out by knowing nothing but learns from its mistakes and, after a few games, becomes unbeatable!

The blue-collar, redneck equivelant of chess is checkers. Several versions have been written for the TI, all apparently from scratch. Their programmers deserve credit for tackling a complex subject, but any of their games can be easily beaten by a beginner.

The favorite game of most of Africa, and dating back to 2000 B.C. in the Middle East, is Mancala, also known as Awari or Mawari in other African languages. It is commonly played with pebbles placed in holes dug in the dirt, or gouged out of a slab of wood. Several public domain versions exist, but the best game by far is the assembly version called Mancala, copyrighted in 1982 by Aldebaran and finally released recently by Triton.

Othello is an American board game, based on ancient Oriental games, in which the object is to capture territory by placing markers at both ends of a row. Its weakness is that the player who goes first is at a distinct disadvantage. Several public domain versions have been released for the TI, all quite slow. Dean Clevaland's was the first. I like the version by Rick Mirus, which has a

black board. Nguyen Long in France wrote the version which is most difficult to beat but it is also very slow, presumably because the computer researches each move one step farther.

Go or Gomoku is a simpler game in which the object is to get 5 markers in a row before your opponent blocks you. There are several public domain versions, but the best by far is Links by Curtis Alan Provance, a unique variant with many features not found elsewhere.
A variant of this game, popular as a toy several years ago but a really challenging brain game, involves stacking chips to get four in a row either vertically or diagonally. One of the best versions was written in the Netherlands.

Tic-Tac-Toe is a child's game, too simple to be called a brain game, but there are 3-dimensional versions, by various authors, which are much more challenging. I have been planning, for years, to write a version in which, if the first player gets 3 in a row and the second player can counter with 3 in a row, the game continues.

The 15 Puzzle was originally a pocket game, consisting of fifteen tiles numbered 1 to 15, randomly arranged in a 4x4 grid, movable but locked within the grid by a frame. The challenge was to slide the tiles around until the numbers were in sequence. The promoter sold hundreds of thousands by offering a large reward to anyone who could solve the puzzle - but his version was impossible to solve! Some public domain computer versions are also impossible, because the programmer has assumed that any random arrangement of numbers was possible. The Texas Instruments version, sold in the early days on cassette, correctly started out with a properly sequenced grid in memory and then scrambled it by a series of random moves. My version did the same, and also offered the option of having two players take turns solving the same puzzle.

Many puzzle games are based on determining, by a series of educated guesses, the sequence in which the computer has randomly arranged colored squares or what have you. These are most frequently called Master Mind, and the most ambitious was written in assembly, occupying 322 disk sectors (!), by J-L. Bazanegue in France.

Peg Jump was an old favorite board game in which holes on a board, in the form of a cross, were filled with pegs. The object was to jump pegs over each other, removing jumped pegs as in checkers, until only one peg remained in the center hole. Texas Instruments sold a good version of this on cassette; Regena wrote another fine version. Many many years ago I owned one of these puzzles which was accompanied by a little booklet showing about 50 "end games." If I could find that booklet again, it would be fun to program these end games into the TI or Regena versions.

Games of the "fox and geese" type require moving, or blocking moves, along certain pathways. The best of these in the TI world, and very difficult to beat, are Giants and Dwarfs by Barry Traver and Quintus by Sam Pincus.

Another type requires placing geometric figures within a

specified area. This is the basis for the L-Game, originally published in Ahl's Creative Computing by Bill Gardner. I have never been able to beat it. Of the same type, but much less difficult, is my Mechanical Aptitude Test, based on the "broken block" problems of S.A.T. tests and other IQ tests.

Many brain games are based on a mathematical theorem or a mathematical progression. These are almost impossible to win until you have puzzled out the secret, and too easy to win thereafter. An example is Pick Up Sticks, in which you and the computer take turns picking up 1 to 3 sticks from a pile of random size, with the player who gets the last stick being the loser. In my version, after the user has lost several games, the computer changes the rules to specify that whoever gets the last stick is the winner - but the computer can still win every time! My Can of Worms lets the user make up all the rules, but he still loses - and Nimbo, based on the Fibonacci series of numbers, is almost impossible to win without knowing the secret.

Other mathematical puzzles depend on logical thinking. Barry Traver wrote a series of three, based on the number 31, which appeared in a recent Genial Traveler. I have written several, mostly as "tinygrams" or short programs in my Tips From The Tigercub. Regena recently published in Micropendium her ingenious Magic Boxes which has several skil levels ranging from fairly easy to extremely difficult.

Most card games played against the computer, such as Twenty-One or Blackjack, are based on pure luck rather than skill or brainwork. There are also various poker games, but I doubt that anyone has yet programmed on the TI - perhaps not on any computer? - the true odds on a poker hand. Arcade Action Software has released a cribbage game which has been reviewed highly, but I have not seen it - nor do I know how to play the game.

Most solitaire card games are based on pure luck. Quality 99's QS-Solitaire is a beautifully programmed solitaire game in assembly, but it is the standard Klondike game in which no real skill is involved. However, Walt Howe's Chainlink Solitaire is my favorite of all the brain games ever programmed for the TI-99/4A. In this version of solitaire, all cards are visible, so an intelligent choice of moves is available - and an option is available to replay the hand by a different method, if the first try ends in failure. The later fairware versions of this program, with assembly links, are very fast. The commercial version, with ribbons of cards streaming between piles, is something to be seen!

Regena published in Micropendium a Poker Solitaire game which also lends itself to some intelligent playing.

Word games are still another category which requires some brainwork - although I would consider the mental exercise to be minimal in the popular wordsearch puzzles, the object of which is to find each of a list of words within a grid of letters. Texas Instruments had this on a cassette. My version offers a somewhat more challenging option, to find words of a specified category which are not listed.

Cryptograms are perhaps the most challenging of word games, but as far as I know, no one has programmed a diskfull of them for the TI-99/4A. The simplest word game is Scramble, in which the letters of a word have been

reassembled into a random arrangement. I wrote one of those, as did everyone else, but I also wrote a more challenging version called Scrambulation, in which each word of a sentence is scrambled and, optionally, the sequence of words is also rearranged. I also wrote Squinch, which Jack Sughrue described as a fiendish game - two words with their letters randomly intermingled into one. And I wrote Bazoo, in which you must find a word by guessing 5 letters at a time, and Changeroo in which you must change one word into another by changing a letter at a time, making a valid new word each time.

However, I believe that the most unique word game ever written for the TI is Karl Romstedt's Superjot, into which he has programmed every 3-letter word in the English language. You and the computer each select a word, and try to guess each other's word - the computer wins more often than not!

Memory games also qualify as brain games, I believe. The most popular is Concentration, originally based on remembering the locations of pairs of cards in a deck scattered face down. Computer versions, such as my Match A Patch, normally use graphics patterns rather than cards.

Other memory games are based on remembering a sequence of numbers or colors, etc. - these are the Simon games. The most viciously difficult of these is one that I wrote several years ago called Nervous Breakdown - it challenges you to simultaneously remember the sequence of three flashing colors, the highest and lowest of three numbers, and the highest and lowest of three tones!

Maze games, if played intelligently rather than by guesswork, are also brain games. The best of these are the "hallways" type which graphically depict your progress through the maze in 3-dimensional graphics.

And there are many other types of brain games - the many versions of the Towers of Hanoi; coin switching puzzles and coin weighing puzzles and liquid measuring puzzles; the classic Nim, and the other classic computer puzzles such as Black Box, Explosion, and others. And I hardly know where to classify some that I have written, such as Reverso and Bassackwards - and Preachers, Lawyers and Salesman.

Most of these I have mentioned are in the public domain, not even fairware. So, if you are tired of trying to zap the invading aliens, if the text adventure has brought you back to the starting point for the umpteenth time, why not try doing something intelligent for a change? ■

```
70 REM PRATOR-PROGRAM 1
80 REM av Lars-Erik Svahn
90 REM PROGRAMBITEN 84-04.14
95 REM XB, MM, EA, SPEECH
100 CALL INIT
110 HEX$="123456789ABCDEF"
120 SPWR=-27648
130 LOAD=64
140 SPEAK=80
150 RESET=112
160 REM HEX KOD I STR-VAR
170 KOD$="06696"
180 REM LADDAR VSM-ADRESSEN
190 FOR I=5 TO 1 STEP -1
200 CALL LOAD(SPWR,LOAD+POS(
HEX$,SEG$(KOD$,I,1),1))
210 NEXT I
220 CALL LOAD(SPWR,SPEAK)
230 REM PAUS
240 FOR I=1 TO 380
245 REM loop 110 ger TEXAS
250 NEXT I
260 CALL LOAD(SPWR,RESET)
270 END
```

■

PROGRAMS WRITE PROGRAMS -4

by Jim Peterson, Tigercub, USA

Well, if you have tried your hand at any MERGE format program writing, you have already discovered that it is slow work, and you need to cram more onto a line than will fit. When a little CALL HCHAR(24,12,32,5) turned into CHR$(157)&CHR$(200)&CHR$(5)&"HCHAR"&CHR$(183)&CHR$(200)&CHR$(2)&"24"&CHR$(179)&CHR$(200)&CHR$(2)&"12"&CHR$(179)&CHR$(200)&CHR$(2)&"32"&CHR$(179)&CHR$(200)&CHR$(1)&"5"&CHR$(182) you gave up? There is an easier way! Using DEF can make the job so simple that you might decide to do all your programming in MERGE format - well no, it's not quite that easy.

The DEF does slow up program execution time considerably, especially when DEFs call each other, but we can tolerate that here.

For instance, that complicated mess of parentheses to squish a line number can be written just once as DEF LINES$(X)=CHR$(INT(X/256))&CHR$(X-256*INT(X-256)) and then, whenever you need a line number, just write LINE$(100) or whatever.

The flag token and counting of characters and all for an unquoted string can be DEF'd as U$(X$)=CHR$(200)&CHR$(LEN(X$))&X$ Then, to write "HELLO" just write U$("HELLO") and let the computer do the work. For a numeric value in the unquoted string, use UN$(X)=CHR$(200)&CHR$(LEN(STR$(X)))&STR$(X), and then 999 becomes UN$(999).

CALL HCHAR can be DEF HCHAR$=CHR$(157) for CALL and, since one DEF can call another, U$("HCHAR") and, since it is always followed by an opening parentheses, CHR$(183) - but wait, let's define that open parentheses as OP$=CHR$(183).
Now DEF HCHAR$=CHR$(157)&U$("HCHAR")&OP$, and you can use HCHAR$ for CALL HCHAR(.

Let's also DEF the comma with DEF C$=CHR$(179) and the closing parentheses with DEF CP$=CHR$(182). Now that long HCHAR that had you discouraged can be abbreviated to CHAR$&UN$(24)&C$&UN$(12)&C$&UN$(32)&C$&UN$(5)&CP$.

I have written a program of 162 of these DEFs, and another program to print out a handy look-up chart of them. It would take 4 pages to print them, so if you want them just ask me for a copy. ■

SAMLINGSSKIVA PROGBIT-93

(pris 30 kr inkl. porto till postgiro 19 83 00-6)

PROGBIT-93 Sectors Used = 360 Free = 0 Filecount 30

Filename	Size	Type	Rec	P
ALPHABLAST	10	Program	BX	
BUG-AL-2/O	5	Dis/Fix	80	
BUG-AL-3/O	5	Dis/Fix	80	
BUG-AL-4/O	8	Dis/Fix	80	
BUG-AL-5/O	4	Dis/Fix	80	
EDITCHAR	14	Program	BX	
EKONOMI	26	Program	BX	
GLOSOR	30	Program	BX	
GOBLIN	11	Program	BX	
GRAPH-BITS	3	Program	BX	
HANGMAN	25	Program	BX	
HORIZONSXB	26	Program	BX	
HORIZON_P	25	Program		
INVENTORY	11	Program	BX	
KEYS/B	4	Program	BX	
KEYS/XB	4	Program	BX	
LOTTO	22	Program	BX	
MAZE	6	Program	BX	
MOSAIC	24	Program	BX	
MULTICAT	13	Program	BX	
PENSION	5	Program	BX	
PG-CLK1	4	Program	BX	
PG-CLK2	3	Program	BX	
PG-CLK3/O	5	Dis/Fix	80	
PG-CLK3/XB	3	Program	BX	
PRINTORD	4	Program	BX	
STATISTICS	19	Program	BX	
SUPER-XB/X	11	Program	BX	
TYPING	8	Program	BX	
XLATE	20*	Program	BX	

■

FUNNELWEB PROGRAM LOADER#1

by Charles Good, Lima Ohio User Group, USA

When booting assembly language PROGRAM software from Funnelweb you usually use #2 or failing that #3 from the LOADERS or DISK REVIEW menus. Loader #2 is basically identical to #5 from the EA module and boots the character set that is in the console GROMs. This is the character set that has those horrible lower case letters that look like small upper case letters and does not have any screen display for ASCII 1-31 (the TI Writer CTRL/U control characters).

Loader #1 is almost identical to #2 except that with loader #1 the first Funnelweb character set is loaded in memory (file C1 in Funnelweb v4.2, file CHARA1 in earlier versions). This is a much better looking character set. Any assembly PROGRAM that can be loaded from Loader #2 will also probably work with Loader #1. Using Loader #1 gives much better looking text with software that does not have its own built in character set.

Try this to see the difference between Loaders #1 and #2. Examine the text that is displayed within DM1000 as it comes already configured in Funnelweb. Then run CONFIGURE and reconfigure DM1000 in the TI Writer side MENU. Change DM1000 from a #2 (GPL Pgm) to a #1 (TIW Pgm) load. Reboot Funnelweb, bring up DM1000, and notice the difference in the appearance of the text. On my system the DM1000 screen colors are also changed, but I like the new screen colors.

Sometimes there is an advantage in not having ASCII 1-31 visible on the screen. The visible control characters create lots of screen clutter. DISK REVIEW of Funnelweb v4.2, both 40 and 80 column versions, comes preconfigured as a loader #1, and you can't change this with CONFIGURE. If you want to get rid of control characters in text VIEWed with DISK REVIEW, you can use CONFIGURE to configure DISK REVIEW (file DR) into a second place in the central menus as a Loader #2 (GPL Pgm) option. Text VIEWed from this alternate DISK REVIEW central menu selection will be free of control characters. ■

PROGRAMBITEN MICROPENDIUM

av Jan Alexandersson

Micropendium i USA har publicerat flera program och artiklar skrivna av (eller om) medlemmar i Programbiten:

Oct 85 p52	NUMTALK, Anders Persson
Dec 85 p08	More on NUMTALK, PB 94-1
Dec 85 p52	Lots of sound from the keyboard, Piano PB 84-2
May 87 p26	Adress Programbiten
Mar 88 p10	A Checksum program that checks, Lars Thomasson, PB 87-4
Oct 88 p20	Swedish user offers PRK BASIC CALLs, PB 88-3
Aug 89 p38	PRK VAR 132, PB 89-3
Apr 91 p27	Quad-density disks and disk manager performance, PB 90-1
Jul 92 p20	Newsletter clearinghouse
Aug 92 p29	CRU addresses, PB 90-6
Jun 93 p29	Comparing sector editors PB 89-4
Oct 93 p26	ALEX/FORMA, PB 93-3 ■

SPRITEDEMO

(from Bug News, USA)

```
100 ! SAVE DSK1.SPRITEDEMO
110 CALL CLEAR :: CALL CHAR(
96,"3C7EFFFFFFFF7E3C")
120 J=-1
130 CALL SCREEN(10)
140 CALL SOUND(-2000,330,3)
150 FOR L=1 TO 28 :: CALL SP
RITE(#L,96,5,L*4+20,60,0,L):
: NEXT L
160 CALL SOUND(-1000,300,3)
170 FOR L=1 TO 28 :: CALL MO
TION(#L,0,L*J):: NEXT L
180 CALL SOUND(-50,440,1)::
J=J*-1
190 Q=Q+1 :: IF Q>5 THEN 200
 ELSE 160
200 CALL DELSPRITE(ALL)
210 END
```

■

NEWS AND VIEWS #1 - Dec92

by Jim Peterson, Tigercub, USA

Mike Wright's "The Cyc" is now available. It is an encyclopedia of knowledge regarding the TI-99/4A and its accessories.

The alphabetical list of material has been drawn from the TI-99/4A Software Directory, 99/4 International Users Group catalog, 99'er Magazine, Texas Instruments Home Computer News, Computer Shopper, Enthusiast 99 Magazine, and various other sources. Mike thinks this is about 40% complete, and plans to add material from the Smart Programmer, MICROpendium, Mini Mag 99, Ryte Data Newsletter, and User Group publications.

In other words, it consists of material from sources that went out of existence several years ago. Since it does not yet include MICROpendium, or the vast amount of material published in user group newsletters during the past 9 years, I doubt that it is even 10% complete.

The appendices consist of indexes to some of the above (including MICROpendium up to Vol. 2 No. 8), etc. and apparently list only a small fraction of the software that has been written for the TI.

The Cyc requires an IBM PC or compatible capable of running WordPerfect 5.1 for DOS or Windows. It is available from CaDD Electronics, 81 Prescott Road, Raymond NH 03077, for $20 including S&H, on your choice of 5.25 360k, 5.25 1.2Mb, 3.5" 720K or 3.5" 1.44Mb diskettes. The price includes one upgrade as more material is added.

Stage 0 of PC99 is now available from the same source for $49, or for $40 to the 130 people who responded to the MICROpendium article. Stages 1 through 4 will each be the same price, if they are ever developed.

PC99 is software which allows TI-99/4A programs to be run on an IBM PC. Stage 0 doesn't do much, and does that too slowly to be practical. The developers are making no promises that any further stages will be completed; they want 1000 TI'ers to show an interest in buying it, and so far have only 130. They also admit that it will only run TI programs on the PC slower than they run on the TI, until a new faster generation of PCs becomes available.

Although PC99 uses software rather than hardware to emulate the TI-99/4A, it will require the Soundblaster card to emulate the TI's speech and music, and will presumably require some specialized hardware to emulate the TI's sprites, if that is ever accomplished.

Seems to me that TI programs with 28-column or 40-column text are going to look strange on a PC's 80-column screen, unless there is a way for a programmer to go in and modify them.

Wouldn't it be more practical to write software that could translate TI XBasic programs into PC Quick Basic? Or even translate TI machine language programs into PC machine language?

An encyclopedia of TI information, that requires a PC running WordPerfect; and software to run TI programs on a PC - is this really the beginning of the end?

In the meantime, Bud Mills is selling his new SCSI ("Scuzzy") hard and floppy disk controller card, although the DSR needed to use it has not been finished. And Asgard Software is selling their new Memory Card, which supports from 128K to 512K of RAM when running programs designed to make use of the card, if any such are ever written. And Barry Boone has completed the buyout of MSDOS, so Geneve owners may finally have an operating system for their computer-on-a--card in an out-of-production P-box, if a programmer can be found to finish it.

All of which has caused me to decide to give the TI world an opportunity to invest in my Mongolian gold mining venture. I haven't actually bought the mine yet, but I will as soon as I get a thousand investors. After that, we will start digging for gold as soon as the mining equipment is designed and built. I want to be totally honest, however, so I warn you that I may drop the project at any moment and leave you high and dry. In the meantime, don't expect me to answer phone calls or letters or keep you posted on the status of your investment.

Now, what devoted TI'er could resist an offer like that?

Gary Bowser of OPA has released an open letter to the TI world to refute rumors that OPA has never made any of the products they offer, have never shipped anything by mail, etc. Actually, the only rumor I had heard was that OPA was apparently out of business because they never answered mail or phone calls.

Gary makes the point that the TI world is such a close-knit community that having one dissatisfied customer reduces the total amount of orders, and that he needs a steady and increasing amount of orders in order to support himself and support future development. That is all very true - but the rumors would never have started, and the customers would never have been dissatisfied, if he would just spend a few pennies and a few minutes of his time to answer every inquiry promptly, to notify customers of any delays and offer refunds if they are unwilling to wait. And he might get some orders if he would take out some ads in MICROpendium to let the TI world know what he has to offer. Messages posted on GENIE are not an effective method of advertising, and not an acceptable method of replying to customers.

While on that subject, TI'ers are quick to complain about poor service from vendors, but have you ever heard one praise a vendor for good service? Bruce Harrison of Harrison Software will spend hours and hours making his software compatible with a customer's system, but you'll never know about it unless you are that customer. Jerry Price has sometimes been accused of poor business ethics, but have you ever heard a complaint about the speed and quality of Tex-Comp's service, in all their years of doing business? There are other long-established vendors whom no one ever complains about, and no one ever praises. If I may blow my own horn just a bit, in the past 9 years 99% of Tigercub orders have been shipped the day they were received or the next mailing day, and complaints have been handled just as promptly

But... time to get off the soapbox. ■

MY GENEALOGY PROGRAM

by Jim Peterson, Tigercub, USA

Some 20 years ago, my late brother researched our family ancestry and gave me a copy of his work. I was not too much interested. It consisted of charts branching backwards in time, showing parents, grandparents, etc., much like a Biblical recitation of "and Jonah begat Abraham and Abraham begat Noah", etc., etc., except that in modern genealogy the mother who actually bears the child is at least given second billing.

But last year a gift of some old family photos and a visit to some graveyards kindled my interest. However, I wanted to do more than just trace that forking family tree backwards. I wanted to know who my grandfather's cousins were, and who their children and grandchildren were.

I was told that there was no really good genealogy program for the TI-99/A. I obtained a sample of a family group sheet, one of the standard tools used by genealogists, and began recording data on it. I soon filled a disk with D-V80 files of those, which printed out to a very thick file of pages with a lot of wasted space.

I thought of trying to write a genealogy program, but wasn't sure what I wanted. About that time, I had an amazing piece of good luck - I was put in contact with a distant relative in Sweden who had researched the family history back into the 1700s and beyond!

He sent me a 3.5 disk containing his genealogy program for the PC, and his files on 1400 family members. Since I do not own a PC and never intend to, I ran to Chuck Grimes for help. He accessed the program's options and printed out for me a list of all 1400 names, a cross-reference list of all children, and two cross-reference lists of marriages, plus several of those family tree charts.

About 1000 of those 1400 names were of the Swedish researcher's father's relatives and his wife's relatives, which were of no interest to me, so I went to work to extract the 400 who were actually my blood relatives. After about

a week of checking one list against another, back and forth, I was not too impressed with the program.

So, again I thought about writing a genealogy program. I was not interested in being able to sort data seventeen ways from Sunday, and I did not care about printing out those bare-bones family trees, but I wanted to be able to easily find a person by name, and find a complete record of parents, spouse, children, biographical data, and sources of data.

Such a program would be difficult to write - and unnecessary. I realized that the best program for my purpose would be no program at all. The magic of Funnelweb and the efficiency of the TI disk controller was all that I needed.

I booted up Funnelweb, went into the Editor, set the tab at 39, and typed -

[1] JAMES WARREN PETERSON is the son of [2]› NORTH EDWIN PETERSON and [3]› LINNIE LEONA STEVENS. He was born 20/8 1923 in Pelican Rapids, Otter Tail County, Minnesota. He was married 7/7 1956 in Tokyo, Japan to [4]› MIDORI IMAI. Their children are [5]› MARIANN MIEKO and [6]› ALAN EDWIN.

And that was followed with some biographical data. I saved it to disk, with SF to preserve the tab setting, as filename 001.

The › after an index number means that a file exists under that number, with information about the person. So, I typed up a similar file about my father and saved it as 002; and so on. Padding the number with 0's causes the disk controller to catalog filenames from 001 to 999 in numerical sequence.

Now, if I need to add to a file, I just load it into Funnelweb and go to work. Since it is in 40-column format, it is easy to edit on-screen.

The TI disk controller can only handle 127 files on a disk, but many of my 400 names are those of children listed in their parents's file without enough data to require a file of their own. When I do run over the 127 limit, it is easy to use an additional disk. If I get more information about such a child, I will just add a › after his number, and set up a new record for him.

What about a printout? I could easily create an .IF file listing all those filenames in numeric sequence, and print them all through the Formatter, using dot commands to change them to 80-column width. I enclosed the index numbers in brackets so Icould easily .TL to double--strike, emphasize or underline them.

However, I like 40-character 2-column text, so I wrote a little program to catalog drives 1 and 2 and print all the files in sequence in two columns.

Now, how about finding records? I booted up Funnelweb again, set the tabs at 5, 35, 50, 55, 60 and 65 and began entering names in index number sequence by index number, first name, last name, file number, father's index number, mother's index number and spouse's index number.

The resulting file was too big for a simple sorting routine to handle, so I tried using Peter Hoddie's fairware program SORT EXPERIMENT, sorting on the last name field with a secondary sort on the first name. I thought that it did a perfect job, until I found that many names were missing. The documentation for SORT EXPERIMENT says it will handle up to 1000 records or 24k, whichever comes first. It fails to mention that after reading in 24k of data it will begin to sort, without warning you that it did not read the complete file!

So I went to Dennis Faherty's TI-SORT, sold by Inscebot. The documentation for that program is very neatly printed but difficult for me to understand. I finally figured it out, and produced an index in alphabetic sequence. I plan to update it with Funnelweb, inserting lines in the proper place, so I will not have to sort it again.

I now have a text-format genealogy which I can easily and quickly update. I can print copies of the index and text to send to relatives who do not have a computer, and the printouts will be very easy for them to understand. If any of them do have a computer and a genealogy program, it will be very easy for them to copy the data.

So once again, the best program is the simplest program that will do the job, and the simplest of all programs is no program at all. ■

TI SINGS (KIND OF)

by Andy Frueh, Lima Ohio User Group, USA (Jan 1993)

First of all, you'll need Terminal Emulator 2, and a Speech Synthesizer. Most of you have heard the TI speak something, and it can be programmed to speak different voices digitally (which reminds me, does anyone know if the TI voice digitizer is still available? I have information on this, and it sounds great), but did you know that the TI can sing?

To clear up a few points on TI speech, you may wonder why the Synthesizer doesn't say every phrase in Extended BASIC or the Speech Editor that it's supposed to. To do this, put a # sign in front of the phrase.

You may also wonder why some units have a flip open lid. Well, originally the speech units were designed to say only what was in the XB and Speech cartridges. However, there were supposed to be some small modules made that fit into these lids. If you have such a unit, open the lid and look inside. You should see a hole a little more than half way in. This is were these modules were to be plugged. Well, the TE2 modules was introduced before the modules. Since TE2 offered an unlimited vocabulary, this project was disbanded.

Reread the TE2 manual (pp 33-42). You should go over the section that deals with using the _, >, and ^ symbols. These affect where the accents come in various words.

To change the actual pitches and slopes (the length of time the voice raises or lowers to its peak pitch), PRINT to the synthesizer using "//xx yyy" where xx is a value from 0 to 63. 0 is a ghostly whisper, 1 is the highest pitch, and 63 is the lowest. yyy is a number from 0 to 255. The TE2 manual says that best results are achieved when you multiply the pitch by 3.2 and use that new value as your slope. The defaults are 43 and 128.

You can use changes in pitch, slope, and infliction symbols to have the computer "sing" notes. I have hear such examples, but do not have a list of what pitches, ect. are what notes. If anyone out there has such a list, please send it to me so I can publish it here. However, like I said, I have heard this "singing", and it isn't the greatest. Oh, the notes are accurate, but to me, singing and EMOTION go hand in hand. The computer can't replace the interpretation that singers do. But it is interesting. ■

THE CONSOLE AS CALCULATOR

by Charles Good, Lima Ohio User Group, USA

This idea comes under the "why on earth didn't I think of this before" category. Have you ever wanted a really easy way to use the 99/4A as a calculator? Perhaps you can't remember where you put your pocket calculator but you always know there the good old TI is located. The best way to use the console as a calculator is from command mode! Type PRINT followed by a string of calculations, and then press ENTER. Thats all there is to it! This method allows very rapid data entry and rapid calculation of complex problems. You might use this method to balance your checkbook by entering a series of numbers such as:

```
PRINT
1254-56.25-452-6.95-77.89-36-45.80
+45+6.32-99.99
530.44
```

You can also do complex calculations such as:

```
PRINT (56*9)/(2.5*6.52)+96-(8/.2+65)
-30.797546
```

Try it! All you need is a console. Just turn it on and enter BASIC. ■

TIPS FROM THE TIGERCUB #69

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

****

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

My TI-PD library now has almost 600 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 #5 and the latest supplement is available for $1 which is deductible from the first order.

In Tips #68 I published my solution to Dr. Ecker's challenge to alternately assign X the value of A and B without using IF...THEN or any outside help. Computer Monthly has arrived again and his solution is better than mine. Try it with any two numbers -

```
100 A=2.765 :: B=-10
110 X=A+B-X :: PRINT X :: GO
TO 110
```

There has been controversy for years as to whether the TI's psuedorandom number generator is truly random. Dr. Ecker's "Computer Fun & Learning" column in Computer Monthly had a question - if you randomly generate numbers between 0 and 9, how often will you get the same number twice in succession? Three times in succession? And etc. Since there are 10 numbers to choose from, it seems to me you would get 2 in a row 10% of the time, 3 in a row 1% of the time, 4 in a row .1%...etc. I wrote this to prove it -

```
100 RANDOMIZE
110 C=C+1 :: X=INT(RND*10)::
 PRINT X;:: IF X=F THEN FL=F
L+1 :: CL(FL)=CL(FL)+1 :: PR
INT "":FL;"=";CL(FL):"C=";C:
"%=";CL(FL)/C :: GOTO 110 EL
SE FL=0 :: F=X :: GOTO 110
```

After 10,000 tries, I had 2 in a row 8.75% of the time and 3 in a row .83% and 4 in a row .07% . Does that prove anything? I don't know.

(Dr. Ecker points out that those percentages could not ever quite add up to 100%!)

Here is another of my XBasic programs to write assembly source code -

```
100 DISPLAY AT(2,1)ERASE ALL
:"ASSEMBLY HELP SCREEN WRITE
R":"":" This program will wr
ite the":"source code for an
 assembly":"routine which ca
n be linked"
110 DISPLAY AT(7,1):"from Ex
tended Basic to dis-":"play
any one of several help":"sc
reens at any designated":"ke
y press or input at any":"po
int in a program."
120 DISPLAY AT(12,1):" The o
riginal source code,":"autho
r unknown, was improved":"by
 Karl Romstedt and further":
"modified by Bruce Harrison.
"
130 DISPLAY AT(20,1):"How ma
ny help screens?" :: ACCEPT
AT(20,24)SIZE(1)VALIDATE(DIG
IT)BEEP:N
140 FOR J=1 TO N :: H$=H$&"H
ELP"&STR$(J)&"," :: NEXT J :
: H$="      DEF   "&SEG$(H$,
1,LEN(H$)-1)
150 DATA VMBW   EQU  >2024,V
MBR   EQU  >202C,KSCAN  EQU
 >201C,STATUS EQU  >837C
160 OPEN #1:"DSK1.HELP/S",OU
TPUT :: PRINT #1:H$ :: FOR J
=1 TO 4 :: READ M$ :: PRINT
#1:M$ :: NEXT J
170 FOR J=1 TO N :: H$="HELP
"&STR$(J):: PRINT #1:H$&"  L
WPI WS":"       LI   R13,HEL
PS"&STR$(J)
180 IF J<N THEN PRINT #1:"
     JMP  SAVSCR"
190 NEXT J :: H$=RPT$(" ",7)
200 PRINT #1:"SAVSCR CLR  R0
":H$&"LI   R1,SAVIT":H$&"LI
  R2,768":H$&"BLWP @VMBR":H$
&"LI   R9,NEWSCR":H$&"MOV  R
9,R1":H$&"MOV  R2,R4"
210 PRINT #1:H$&"LI   R3,>60
00":"ADDOFF MOVB *R13+,*R9":
H$&"AB   R3,*R9+":H$&"DEC  R
4":H$&"JNE  ADDOFF":H$&"BLWP
 @VMBW"
220 PRINT #1:"KEYLOO BLWP @K
SCAN":H$&"BLWP @KSCAN":H$&"C
B   @ANYKEY,@STATUS":H$&"JNE
  KEYLOO"
230 PRINT #1:"REPL   LI   R1
,SAVIT":H$&"BLWP @VMBW":"RET
N   LWPI >83E0":H$&"B    @>6
A"
240 PRINT #1:"WS     BSS  32
":"SAVIT  BSS  768":"NEWSCR
BSS  768":"ANYKEY BYTE >20":
H$&"EVEN"
250 DISPLAY AT(3,1)ERASE ALL
:" Enter data just as you":"
want it to appear, in 24":"l
ines. Press Enter for blank"
:"lines."
260 FOR J=1 TO N :: DISPLAY
AT(12,1):"Ready for screen #
"&STR$(J):"":"Press any key"
270 CALL KEY(0,K,S):: IF S=0
 THEN 270 ELSE CALL CLEAR
280 ACCEPT AT(1,0):M$ :: PRI
NT #1:"HELPS"&STR$(J)&" TEXT
 ' "&M$&RPT$(" ",30-LEN(M$))
&" '"
290 FOR K=2 TO 24 :: ACCEPT
AT(K,0):M$ :: PRINT #1:H$&"T
EXT ' "&M$&RPT$(" ",30-LEN(M
$))&" '"
300 NEXT K :: NEXT J :: PRIN
T #1:H$&"END"
310 DISPLAY AT(3,1)ERASE ALL
:" Source code has been writ
-":"ten to DSK1 as HELP/S. T
o":"assemble, insert Editor/
":"Assembler module."
320 DISPLAY AT(7,1):"Insert
Assembler disk in   drive 1
.":"Select 2 ASSEMBLER":"Loa
d Assembler? Y":"Source file
name DSK2.HELP/S"
330 DISPLAY AT(12,1):"Object
 file name? DSK2.HELP/O":"Li
st file name? Press Enter":"
Options? R"
340 DISPLAY AT(15,1):"Load t
he resulting object":"file i
nto your program by":"CALL I
NIT ::":"CALL LOAD(""DSK1.HE
LP/O"") or,"
350 DISPLAY AT(19,1):"much b
```

```
etter, imbed it with":"ALSAV
E or SYSTEX."
360 DISPLAY AT(21,1):"Access
 the screens in your progra
m by":" CALL LINK(""HELP1"")
":"CALL LINK(""HELP2""), etc
."
370 CALL KEY(0,K,S):: IF S=0
 THEN 370 ELSE CALL CLEAR
```

For instance, at any point in a program where keyboard input is required and user may not know what to do - ACCEPT AT(24,1):M$ :: IF M$="HELP" THEN CALL LINK("HELP1") and the first help screen will pop up to give instructions. Press any key and the previous screen reappears.

This time I am borrowing heavily from the TI*MES news letter of England, which has also borrowed from the REC newsletter.

This one is useless, but is a remarkable example of compact complex programming. It shows that there is an algorithm for everything. See if you can figure out how it works -

```
100 CALL CLEAR :: FOR A=1 TO
 2 :: FOR B=1 TO 4 :: X=2-AB
S(SGN(B-3)):: FOR C=1 TO X :
: PRINT CHR$(84-7*A+5*B-8*X)
;:: NEXT C :: NEXT B :: PRIN
T CHR$(A+31):: NEXT A
```

Another useless one that is easier to figure out -

```
100 DISPLAY AT(1,1)ERASE ALL
:"NUMBER OF MONTH(1-12)"
110 ACCEPT AT(2,12)SIZE(2)VA
LIDATE(DIGIT):A :: IF A<1 OR
 A>12 THEN 110
120 DISPLAY AT(3,1):A;"x 4="
;A*4 :: A=A*4
130 DISPLAY AT(4,1):A;"+13="
;A+13 :: A=A+13
140 DISPLAY AT(5,1):A;"x 25=
";A*25 :: A=A*25
150 DISPLAY AT(6,1):A;"-200=
";A-200 :: A=A-200
160 DISPLAY AT(8,1):"Input d
ate (1-31):" :: ACCEPT AT(8,
19)SIZE(2)VALIDATE(DIGIT):B
:: IF B<1 OR B>31 THEN 160
170 DISPLAY AT(10,1):A;"+";B
;"=";A+B :: A=A+B
180 DISPLAY AT(11,1):A;"x 2=
";A*2 :: A=A*2
190 DISPLAY AT(12,1):A;"-40=
";A-40 :: A=A-40
200 DISPLAY AT(13,1):A;"x 50
=";A*50 :: A=A*50
210 DISPLAY AT(15,1):"Input
last two digits of    year e
g 91:"
220 ACCEPT AT(16,16)SIZE(2)V
ALIDATE(DIGIT):B
230 DISPLAY AT(18,1):A;"+";B
;"=";A+B :: A=A+B
240 DISPLAY AT(19,1):A;"-105
00=";A-10500 :: A=A-10500
250 DISPLAY AT(24,1):"ANY KE
Y FOR ANOTHER"
260 CALL KEY(5,A,B)
270 IF B<1 THEN 260
280 RUN
290 END
```

One for the little ones - change the string to anything you want.

```
1 REM SILLY PROG BY S SHAW
  MARCH 1991
2 ! did you see COMPUTER WAR
S-the film? It is said that
the star, who was required t
o type fast into a computer
3 ! could not type, so a pro
gram just like this one was
used to give a good effect!
4 ! now adjust it how you wi
sh and show your friends how
 fast you can type
5 ! at end of text string pr
ogram will just stop with th
is listing but can be modifi
ed to do anything you wish!
6 !
100 A$="This is how a non-ty
pist canproduce information
on      screen quickly,witho
ut      "
110 A$=A$&"having to look at
 what keys are being bashed!
 Just bash keys and watch ho
w perfect  text appears no m
atter what you press."
120 CALL CLEAR :: PRINT A$:
: : : : :
130 CALL KEY(5,A,B):: IF B<1
 THEN 130
140 C=C+1 :: PRINT SEG$(A$,C
,1);:: IF C=LEN(A$)THEN 160
150 GOTO 130
160 GOTO 160
```

And a very fast routine to find prime numbers -

```
100 ! FIRST 100 PRIMES
    -QUICKLY-
110 ! Dr H B Phillps
  from THE REC NEWSLETTER
 March 1988  Vol 3 #2
120 DIM P(300),X(12)
130 A=0 :: B=1 :: D=0.5 :: E
=180
140 M=100 :: L=3 :: F=0
150 ! increase M for more- a
lso increase DIMs.
160 PRINT 2;:: C=B :: IF M=B
 THEN END
170 L=INT((M/C)*L+F):: N=L+L
+B
180 FOR I=B TO INT((SQR(N)-B
)*D):: PP=P(I)
190 IF PP=B THEN 230
200 IF PP=A THEN PP=I+I+B ::
 PRINT PP;:: P(I)=PP :: C=C+
B :: IF C=M THEN END
210 IF X(I)=A THEN X(I)=(PP*
PP-B)*D
220 FOR J=X(I)TO L STEP PP :
: P(J)=B :: NEXT J :: X(I)=J
230 NEXT I :: IF F=0 THEN S=
I
240 FOR I=S TO L
250 IF P(I)=A THEN PP=I+I+B
:: PRINT PP;:: P(I)=PP :: C=
C+B :: IF C=M THEN END
260 NEXT I :: F=(M-C)*L/E ::
 S=L+B
270 GOTO 170
```

And a demonstration of how the INTERRUPT routine works independently of whatever else the computer is doing -

```
100 REM interrupt demo
110 REM
120 REM MACHINE LANGUAGE
130 REM ROUTINE LOADED AT
140 REM >2600 XB OR E/A WITH
 32K
150 REM >7200 MINI MEM NO 32
K
160 REM
170 CALL INIT
180 XM=9728
190 MM=29184
200 LAD=XM
210 REM TEST XB OR MM?
220 CALL LOAD(XM,170)
230 CALL PEEK(XM,X)
240 IF X=170 THEN 270
250 REM NO 32K MUST BE MM
260 LAD=MM
270 A=LAD
280 REM LOAD M/C
290 CALL CLEAR
```

```
300 FOR D=540 TO 630 STEP 10
310 CHECK=0
320 FOR N=1 TO 10
330 READ X
340 CALL LOAD(A,X)
350 CHECK=CHECK+X
360 A=A+1
370 NEXT N
380 READ X
390 IF CHECK<>X THEN 490
400 NEXT D
410 REM POKE INTERRUPT
420 REM ROUTINE ADDRESS
430 REM INTO >83C4
440 CALL LOAD(-31804,LAD/256
)
450 REM JUST IDLE AWAY TIME
460 FOR N=1 TO 9940
470 NEXT N
480 STOP
490 PRINT "ERROR IN DATA STA
TEMENT ";D
500 STOP
510 REM EACH DATA STATEMENT
520 REM HAS 10 DATA BYTES
530 REM PLUS A CHECK SUM
540 DATA 192,236,000,092,004
,194,005,131,002,131,987
550 DATA 000,060,026,003,004
,195,006,236,000,094,624
560 DATA 203,003,000,092,060
,172,000,090,006,002,628
570 DATA 017,015,019,010,006
,002,019,004,002,000,94
580 DATA 002,039,010,083,016
,002,002,000,002,086,242
590 DATA 096,003,016,007,002
,000,000,119,010,083,336
600 DATA 016,002,002,000,000
,072,160,003,002,096,353
610 DATA 064,000,006,192,215
,192,006,192,215,192,1274
620 DATA 016,000,216,044,000
,094,140,000,004,091,605
630 DATA 000,015,000,000,138
,128,000,000,000,000,281
640 END
```

Run that, then press FCTN 4. Enter LIST. Enter NEW. To stop it, enter BYE.

This is an oldie, but well worth repeating. You can use it to turn your cassette recorder on and off, to add speech or music from tape to a running program. With the proper hardware, you could write a program to control almost anything from the cassette port. If it doesn't work, reverse the polarity of the remote. Ed Hall wrote this -

```
100 CALL INIT
110 CALL LOAD(16368,79,70,70
,32,32,32,36,252)
120 CALL LOAD(16376,79,78,32
,32,32,32,36,244)
130 CALL LOAD(8194,37,4,63,2
40)
140 CALL LOAD(9460,2,12,0,45
,29,0,4,91,2,12,0,45,30,0,4,
91,203,78)
150 PRINT "PRESS":" P Play":
"S Stop"
160 CALL KEY(3,A,B)
170 IF B<1 THEN 160
180 ON POS("PS",CHR$(A),1)+1
 GOTO 160,190,200)
190 CALL LINK("ON"):: GOTO 1
60
200 CALL LINK("OFF"):: GOTO
160
```

And that is just about - MEMORY FULL!

Jim Peterson ■

DISASSEMBLER — FUNNELWEB

```
100 REM ****************
110 REM * MM / EA / XB *
120 REM * DISASSEMBLER *
130 REM ****************
140 REM
150 REM Revised Sept 18/84
160 REM  Funnelweb Farm
170 REM
180 DIM S(16),Z(5),HX$(15),N
B$(15),SP$(3),W$(3)
190 REM TITLES
200 GOSUB 920
210 FOR I=1 TO 12
220 CALL COLOR(I,7,1)
230 NEXT I
240 PRINT TAB(9);"MM / EA /
XB":;:TAB(9);"DISASSEMBLER":
;:TAB(9);"using  BASIC":;:
250 PRINT :;:TAB(13);"from":
;:TAB(8);"FUNNELWEB FARM":;:
;:;:;:
260 GOSUB 950
270 LOC=24576
280 GOSUB 5090
290 EMX=1+(M=170)*(N=0)-2*(R
ND>.8)
300 LOC=3322
310 GOSUB 5120
320 TO=256*M+N
330 LOC=3324
340 GOSUB 5120
350 T1=256*M+N
360 CALL CHAR(128,"00FE18FE1
8187E")
370 CALL COLOR(13,11,1)
380 REM USEFUL ARRAYS
390 S(0)=32768
400 HD$="0123456789ABCDEF"
410 FOR I=0 TO 15
420 S(I+1)=S(I)/2
430 HX$(I)=SEG$(HD$,I+1,1)
440 NB$(I)=SEG$("00010203101
112132021222330313233",1+2*I
,2)
450 NEXT I
460 SP$(0)=" "
470 SP$(1)=" "
480 SP$(2)="   "
490 SP$(3)="   "
500 REM RESTART ENTRY
510 GOSUB 920
520 PRINT "PRESS 1 Screen":"
——— 2 COLIST Diskfile":"
    3 Printer":"      4 Disk
 File":"      5 Quit":;:;:;:
530 GOSUB 950
540 OFST=0
550 UNCHR=128
560 GOSUB 5020
570 IF K=15 THEN 560
580 K=K-49
590 IF (K<0)+(K>4)THEN 560
600 IF K=4 THEN 2460
610 F=K
620 G=F>1
630 IF F=0 THEN 760
640 CALL CLEAR
650 PRINT "Enter CHAR (singl
e char)":" to be used instea
d of":" non-printing chars":
" in TEXT mode"
660 GOSUB 5020
670 UNCHR=K
680 IF F=2 THEN 730
690 PRINT :;:"Disk.file name
 ?":;:
700 INPUT " ":DEVICE$
710 OPEN #F:DEVICE$,OUTPUT,D
ISPLAY ,VARIABLE 80
720 GOTO 760
730 PRINT :;:"Printer name ?
":;:
740 GOTO 700
750 REM OPTIONS
760 GOSUB 920
770 PRINT "PRESS 1 Disassemb
le Opcode":"——— 2 Display
Hex Data":"      3 Read ASCI
I Text"
780 PRINT "      4 Read with
 offset >60":"      5 BL,BLW
P branches":"      6 REF/DEF
```

```
 Table"
790 PRINT "      7 Restart":
"      8 End":;:;:
800 GOSUB 950
810 GOSUB 5020
820 K=K-48
830 IF (K<1)+(K>8)THEN 810
840 CALL CLEAR
850 IF K<6 THEN 870
860 ON K-5 GOTO 1280,2460,24
60
870 GOSUB 1020
880 ON K GOTO 2050,4330,4480
,890,4660
890 OFST=96
900 GOTO 4480
910 REM SCREEN
920 CALL CLEAR
930 CALL SCREEN(7)
940 RETURN
950 CALL SCREEN(15)
960 RETURN
970 REM JOB DONE
980 PRINT :;:
990 INPUT "Press ENTER to co
ntinue:":CON$
1000 GOTO 760
1010 REM INPUT ADDRESS RANGE
1020 CALL CLEAR
1030 PRINT TAB(8);"ADDRESS R
ANGE":;:;:" From ?":;:
1040 GOSUB 1140
1050 A=DEC
1060 PRINT :" To ?":;:
1070 GOSUB 1140
1080 CALL CLEAR
1090 B=DEC
1100 PRINT " PRESS   FOR
ACTION":" -----   ---   ---
---":;:
1110 PRINT " SPACE
PAUSE":;:"  KEY          CO
NTINUE":;:" BACK          R
ESTART":;:;:;:;:;:;:
1120 RETURN
1130 REM GET ADDRESS
1140 INPUT " (Hex address) >
":A$
1150 A$=SEG$("0000"&A$,LEN(A
$)+1,4)
1160 PS=1
1170 FOR I=1 TO 4
1180 PS=PS*POS(HD$,SEG$(A$,I
,1),1)
1190 NEXT I
1200 IF PS=0 THEN 1140
1210 DEC=0
1220 FOR I=1 TO 4
1230 DEC=DEC+S(4*I-1)*(POS(H
D$,SEG$(A$,I,1),1)-1)
1240 NEXT I
1250 DEC=2*INT(DEC/2)
1260 RETURN
1270 REM AVAILABLE MEMORY
1280 PRINT " AVAILABLE MEMOR
Y":" --------- ------":;:
1290 ON EMX GOTO 1310,1480,1
740
1300 REM EA ADDRESSES
1310 LOC=8228
1320 GOSUB 5090
1330 IF HEX$>"0000" THEN 136
0
1340 PRINT :;:" Not Initiali
zed"
1350 GOTO 980
1360 PRINT "  FSTHI  >";HEX$
1370 LOC=8230
1380 GOSUB 5090
1390 PRINT "  LSTHI  >";HEX$
1400 LOC=8232
1410 GOSUB 5090
1420 PRINT :"  FSTLOW >";HEX
$
1430 LOC=8234
1440 GOSUB 5090
1450 PRINT "  LSTLOW >";HEX$
:;:
1460 GOTO 1870
1470 REM MM ADDRESSES
1480 LOC=28672
1490 GOSUB 5090
1500 IF HEX$="A55A" THEN 153
0
1510 PRINT "  BASIC FILES IN
 MM"
1520 GOTO 980
1530 LOC=28700
1540 GOSUB 5090
1550 PRINT "  FFMM >";HEX$
1560 LOC=28702
1570 GOSUB 5090
1580 PRINT "  LFMM >";HEX$
1590 LOC=28706
1600 GOSUB 5090
1610 PRINT :"  FFHM >";HEX$
1620 LOC=28708
1630 GOSUB 5090
1640 PRINT "  LFHM >";HEX$
1650 LOC=28710
1660 GOSUB 5090
1670 PRINT :"  FFLM >";HEX$
1680 LOC=28712
1690 GOSUB 5090
1700 PRINT "  LFLM >";HEX$:;
:
1710 LOC=28702
1720 GOTO 1870
1730 REM XB ADDRESSES
1740 LOC=-31866
1750 GOSUB 5090
1760 PRINT "  LFHM  >";HEX$
1770 LOC=8194
1780 GOSUB 5090
1790 PRINT :"  FFALM >";HEX$
1800 LOC=8196
1810 GOSUB 5090
1820 PRINT "  LFALM >";HEX$:
;:
1830 IF HEX$>"0000" THEN 187
0
1840 PRINT :;:" Not Initiali
zed"
1850 GOTO 980
1860 REM REF/DEF TABLE
1870 GOSUB 5090
1880 A=256*M+N
1890 B=16383-16384*(EMX=2)
1900 PRINT " TABLE PROGRAM
ENTRY":" ----- -------  ----
-":;:
1910 FOR U=A TO B STEP 8
1920 GOSUB 5060
1930 LOC$=HEX$
1940 CALL PEEK(U,Z(0),Z(1),Z
(2),Z(3),Z(4),Z(5),M,N)
1950 V$=""
1960 FOR R=0 TO 5
1970 V$=V$&CHR$(Z(R))
1980 NEXT R
1990 GOSUB 5140
2000 PRINT " >";LOC$;" ";V$;
TAB(17);">"&HEX$
2010 GOSUB 4940
2020 NEXT U
2030 GOTO 980
2040 REM DISASSEMBLER
2050 PRINT #F:;:"Disassemble
r output":;:
2060 FOR LOC=A TO B STEP 2
2070 L=0
2080 U=LOC
2090 GOSUB 5060
2100 LOC$=HEX$
2103 IF F=0 THEN 2110
2105 PRINT LOC$
2110 GOSUB 5090
2120 V=M*256+N
2130 V$=HEX$
2140 IF (LOC<14)-(LOC>66)*(L
OC<76)+(LOC=3324)-(LOC>T0-2)
*(LOC<T0+28)-(LOC>T1-2)*(LOC
<T1+24)THEN 2220
2150 IF (EMX=3)*(LOC>24590)*
(LOC<24656)THEN 2220
2160 IF (EMX=2)*(LOC>24574)*
(LOC<24666)THEN 2220
2170 NYB$=NB$(INT(M/16))&NB$
(M-16*INT(M/16))&NB$(INT(N/1
6))&NB$(N-16*INT(N/16))
2180 REM FORMAT ?
2190 IF V>8191 THEN 2210
2200 ON -(V<512)-(V<832)-(V<
1024)-(V<2048)-(V<4096)-(V<8
192)GOTO 2630,3010,3100,3470
,3510,2220
```

```
2210 ON 1-(V<11264)-(V<12288
)-(V<14336)-(V<16384)GOTO 25
10,3730,2920,3730,2830
2220 E$="DATA >"&HEX$
2230 GOTO 2300
2240 REM PRINT OPCODE
2250 S$=""
2260 IF LEN(E$)<19 THEN 2300
2270 V=POS(E$,",",3)
2280 S$=SEG$(E$,V+1,LEN(E$)-
V)
2290 E$=SEG$(E$,1,V)
2300 PRINT #F:LOC$;SP$(F);V$
;" ";E$
2310 IF L=0 THEN 2330
2320 ON L GOSUB 2380,2400,24
30
2330 GOSUB 4940
2340 IF K=15 THEN 2460
2350 NEXT LOC
2360 GOTO 980
2370 REM PRINT MULTILINES
2380 PRINT #F:LO$(1);SP$(F);
W$(1);TAB(28-LEN(S$));S$
2390 RETURN
2400 GOSUB 2380
2410 PRINT #F:LO$(2);SP$(F);
W$(2)
2420 RETURN
2430 PRINT #F:LO$(3);SP$(F);
W$(3);SP$(F);C$
2440 RETURN
2450 REM ORDERLY FINISH
2460 IF F=0 THEN 2480
2470 CLOSE #F
2480 IF (K=15)+(K=7)THEN 510
2490 CALL CLEAR
2500 STOP
2510 REM FORMAT I
2520 E$=SEG$("SZC SZCBS   SB
  C   CB  A   AB  MOV MOVBSO
C SOCB",1+4*(INT((V-16384)/4
096)),4)
2530 GOSUB 3870
2540 GOSUB 3920
2550 S$=R$
2560 T=VAL(SEG$(NYB$,3,1))
2570 R$=SEG$(NYB$,4,2)
2580 GOSUB 3890
2590 GOSUB 3920
2600 E$=E$&" "&S$&","&R$
2610 GOTO 2250
2620 REM FORMAT II
2630 E$=SEG$("JMPJLTJLEJEQJH
EJGTJNEJNCJOCJNOJL JH JOPSBO
SBZTB ",1+3*INT((V-4096)/256
),3)&" "
2640 DISP$=SEG$(NYB$,5,4)
2650 DIS=0
2660 FOR I=1 TO 4
2670 DIS=DIS+VAL(SEG$(DISP$,
I,1))*S(7+2*I)
2680 NEXT I
2690 IF DIS<128 THEN 2710
2700 DIS=DIS-256
2710 IF SEG$(E$,2,1)="B" THE
N 2810
2720 IF DIS=0 THEN 2790
2730 DS=DIS*2+2
2740 DS$="$"&SEG$("- +",2+SG
N(DS),1)
2750 U=DIS*2+LOC+2
2760 GOSUB 5060
2770 E$=E$&" >"&HEX$&" ["&DS
$&STR$(ABS(DS))&"]"
2780 GOTO 2250
2790 E$="NOP"
2800 GOTO 2250
2810 E$=E$&" "&STR$(DIS)
2820 GOTO 2250
2830 REM  FORMAT III
2840 E$=SEG$("COC CZC XOR ",
1+4*INT((V-8192)/1024),4)
2850 GOSUB 3870
2860 GOSUB 3920
2870 S$=R$
2880 R$=SEG$(NYB$,4,2)
2890 GOSUB 3890
2900 E$=E$&" "&S$&",R"&STR$(
R)
2910 GOTO 2250
2920 REM  FORMAT IV
2930 E$=SEG$("LDCRSTCR",1+4*
INT((V-12288)/1024),4)
2940 R$=SEG$(NYB$,4,2)
2950 GOSUB 3890
2960 C$=STR$(R)
2970 GOSUB 3870
2980 GOSUB 3920
2990 E$=E$&" "&R$&","&C$
3000 GOTO 2250
3010 REM FORMAT V
3020 E$=SEG$("SRASRLSLASRC",
1+3*INT((V-2048)/256),3)&" "
3030 IF V>3071 THEN 2220
3040 GOSUB 3880
3050 V=R
3060 R$=SEG$(NYB$,5,2)
3070 GOSUB 3890
3080 E$=E$&" "&"R"&STR$(V)&"
,"&STR$(R)
3090 GOTO 2250
3100 REM FORMAT VI
3110 E$=SEG$("BLWPB   X   CL
R NEG INV INC INCTDEC DECTBL
  SWPBSETOABS ",1+4*INT((V-1
024)/64),4)
3115 IF V>1919 THEN 2220
3120 GOSUB 3870
3130 GOSUB 3920
3140 IF E$<>"BLWP" THEN 3170
3150 A$=SEG$(R$,5,2)
3160 ON EMX GOSUB 3200,3250,
3300
3170 E$=E$&" "&R$
3180 GOTO 2250
3190 REM E/A UTILITIES
3200 IF (R$<"@>2100")+(R$>"@
>2124")THEN 3230
3210 PS=POS("--__--__04080C1
014181C--002024",A$,1)
3220 GOSUB 3350
3230 RETURN
3240 REM  MM UTILITIES
3250 IF (R$<"@>6018")+(R$>"@
>6050")THEN 3280
3260 PS=POS("4044484C1C20242
82C30345018383C",A$,1)
3270 GOSUB 3350
3280 RETURN
3290 REM XB UTILITIES
3300 IF (R$<"@>2008")+(R$>"@
>2034")THEN 3330
3310 PS=POS("080C1014181C202
4282C3034",A$,1)
3320 GOSUB 3350
3330 RETURN
3340 REM NAME UTILITY
3350 IF PS=0 THEN 3410
3360 B$="NUMASGNUMREFSTRASGS
TRREFXMLLNKKSCAN VSBW  VMBW
 VSBR  VMBR  VWTR  ERR   GPL
LNKDSRLNKLOADER"
3370 B$=SEG$(B$,3*PS-2,6)
3380 R$="@"&B$
3390 IF POS("XMLLNKDSRLNKGPL
LNK",B$,1)=0 THEN 3410
3400 GOSUB 3430
3410 RETURN
3420 REM BLWP DATA
3430 L=2
3440 GOSUB 4070
3450 C$="DATA >"&HEX$
3460 RETURN
3470 REM FORMAT VII
3480 E$=SEG$("IDLERSETRTWPCK
ONCKOFLREX",1+4*INT((V-832)/
32),4)
3485 IF V<>896 THEN 2220
3490 GOTO 2250
3500 IF V<>896 THEN 2220
3510 REM FORMAT VIII
3520 T=INT((V-512)/16)
3525 IF (V<>768)*(V>736)THEN
 2220
3530 V=INT(T/2)
3540 IF V<(T/2)THEN 2220
3550 E$=SEG$("LI  AI  ANDIOR
I CI  STWPSTSTLWPILIMI",1+4*
V,4)
3560 ON 1+V GOTO 3680,3680,3
680,3680,3680,3570,3570,3600
,3640
3570 GOSUB 3880
3580 E$=E$&" R"&STR$(R)
3590 GOTO 2250
```

```
3600 GOSUB 4070
3610 GOSUB 4160
3620 E$=E$&" "&HEX$
3630 GOTO 2250
3640 GOSUB 4070
3650 E$=E$&" "&SEG$(HEX$,4,1
)
3660 W$(2)=""
3670 GOTO 2250
3680 GOSUB 3880
3690 GOSUB 4070
3700 GOSUB 4160
3710 E$=E$&" R"&STR$(R)&","&
HEX$
3720 GOTO 2250
3730 REM FORMAT IX
3740 E$=SEG$("XOP ___ ... MP
Y DIV ",1+4*INT((V-11264)/10
24),4)
3750 GOSUB 3870
3760 GOSUB 3920
3770 S$=R$
3780 R$=SEG$(NYB$,4,2)
3790 GOSUB 3890
3800 IF E$<>"XOP " THEN 3830
3810 D$=STR$(R)
3820 GOTO 3840
3830 D$="R"&STR$(R)
3840 E$=E$&" "&S$&","&D$
3850 GOTO 2250
3860 REM REGISTER #
3870 T=VAL(SEG$(NYB$,6,1))
3880 R$=SEG$(NYB$,7,2)
3890 R=4*VAL(SEG$(R$,1,1))+V
AL(SEG$(R$,2,1))
3900 RETURN
3910 REM T-FIELD
3920 ON 1+T GOTO 3930,3950,3
990,3970
3930 R$="R"&STR$(R)
3940 RETURN
3950 R$="*R"&STR$(R)
3960 RETURN
3970 R$="*R"&STR$(R)&"+"
3980 RETURN
3990 GOSUB 4070
4000 GOSUB 4160
4010 IF R THEN 4040
4020 R$="@"&HEX$
4030 RETURN
4040 R$="@"&HEX$&"(R"&STR$(R
)&")"
4050 RETURN
4060 REM NEXT WORD
4070 LOC=LOC+2
4080 L=L+1
4090 U=LOC
4100 GOSUB 5060
4110 LO$(L)=HEX$
4120 GOSUB 5090
4130 W$(L)=HEX$
4140 RETURN
4150 REM DEFINED ADDRESSES
4160 PD$=""
4170 PS=POS("8300_834A_835C_
836E_837C_83E0_8400_8800_880
2_8C00_8C02_9000_9400_9800_9
802_9C00_9C02",HEX$,1)
4180 IF PS>0 THEN 4220
4190 GOSUB 4280
4200 HEX$=">"&HEX$
4210 RETURN
4220 PD$=SEG$("PAD  FAC  ARG
  STACKGPLSTGPLWSSOUNDVDPRDV
DPSTVDPWDVDPWASPCHRSPCHWGRMR
DGRMRAGRMWDGRMWA",PS,5)
4230 IF HEX$>"835C" THEN 425
0
4240 PD$=SEG$(PD$,1,3)
4250 HEX$=PD$
4260 RETURN
4270 REM LEADING ZEROS
4280 IF (SEG$(HEX$,1,1)>"0")
+(LEN(HEX$)=1)THEN 4310
4290 HEX$=SEG$(HEX$,2,LEN(HE
X$)-1)
4300 GOTO 4280
4310 RETURN
4320 REM DISPLAY DATA
4330 FOR U=A TO B STEP 8-8*G
4340 GOSUB 5060
4350 PRINT #F:HEX$;TAB(5-G*6
);
4360 IF F<2 THEN 4380
4370 PRINT #F:"DATA ";
4380 FOR LOC=U TO U+6-G*8 ST
EP 2
4390 GOSUB 5090
4400 PRINT #F:" >";HEX$;
4410 NEXT LOC
4420 PRINT #F:
4430 GOSUB 4940
4440 IF K=15 THEN 2460
4450 NEXT U
4460 GOTO 980
4470 REM DISPLAY TEXT
4480 FOR U=A TO B STEP 16-G*
40
4490 GOSUB 5060
4500 PRINT #F:TAB(1-G);HEX$;
TAB(6-G*2);"TEXT '";
4510 FOR LOC=U TO U+15-G*40
4520 GOSUB 5090
4530 M=M-OFST
4540 IF (M<127)+(M>31)=-2 TH
EN 4560
4550 M=UNCHR
4560 PRINT #F:CHR$(M);
4570 IF LOC=B THEN 4630
4580 NEXT LOC
4590 PRINT #F:"'"
4600 GOSUB 4940
4610 IF K=15 THEN 2460
4620 NEXT U
4630 PRINT #F:"'"
4640 GOTO 980
4650 REM BL,BLWP TARGETS
4660 PRINT #F:"Locn Inst Trg
t Wksp Pgct":"____ ____ ____
____ ____":;:;:
4670 FOR U=A TO B STEP 2
4680 LOC=U
4690 GOSUB 5090
4700 PS=POS("0420_06A0",HEX$
,1)
4710 IF PS=0 THEN 4890
4720 E$=SEG$("BLWP BL   ",PS
,5)
4730 PC$=""
4740 R$=""
4750 GOSUB 5060
4760 LOC$=HEX$
4770 U=U+2
4780 LOC=U
4790 GOSUB 5090
4800 T$=HEX$
4810 IF PS=6 THEN 4880
4820 LOC=256*M+N
4830 GOSUB 5090
4840 R$=HEX$
4850 LOC=LOC+2
4860 GOSUB 5090
4870 PC$=HEX$
4880 PRINT #F:LOC$;" ";E$;T$
;" ";R$;" ";PC$
4890 GOSUB 4940
4900 IF K=15 THEN 2460
4910 NEXT U
4920 GOTO 980
4930 REM EXIT/HOLD LOOP
4940 CALL KEY(3,K,ST)
4950 IF K<>32 THEN 5000
4960 CALL SCREEN(12)
4970 CALL KEY(3,K,ST)
4980 IF ST<=0 THEN 4970
4990 CALL SCREEN(15)
5000 RETURN
5010 REM KEY LOOP
5020 CALL KEY(3,K,ST)
5030 IF ST=0 THEN 5020
5040 RETURN
5050 REM PEEK/HEX ROUTINE
5060 M=INT(U/256)
5070 N=U-256*M
5080 GOTO 5140
5090 IF LOC<32768 THEN 5120
5100 LOCX=LOC-65536
5110 GOTO 5130
5120 LOCX=LOC
5130 CALL PEEK(LOCX,M,N)
5140 HEX$=HX$(INT(M/16))&HX$
(M-16*INT(M/16))&HX$(INT(N/1
6))&HX$(N-16*INT(N/16))
5150 RETURN
5160 END
```

■

AIR DEFENCE
SPEL I BASIC

```
70 REM AIRDEFENCE
80 REM BY T.L.WAHL
90 REM COMPUTE 83-04
100 DIM BLOCK$(2),PLACE(2),B
UILDING(32,2)
110 RANDOMIZE
120 REM BOMB CHAR
130 CALL CHAR(129,"001CBEFFF
FBE1C")
140 REM CROSSHAIR CHAR
150 CALL CHAR(130,"181818FFF
F181818")
160 CALL CLEAR
170 CALL SCREEN(12)
180 FOR J=5 TO 8
190 CALL COLOR(J,5,16)
200 NEXT J
210 FOR J=9 TO 12
220 CALL COLOR(J,2,14)
230 NEXT J
240 T=0
250 P=0
260 Q=0
270 M=0
280 CALL CLEAR
290 PRINT "        AIR DEFEN
CE": : : :
330 PRINT " do you need inst
ructions?": :
350 PRINT "        type Y or
 N": : : : : : : : :
390 CALL KEY(3,Y,STATUS)
400 IF STATUS=0 THEN 390
410 IF Y=78 THEN 750
420 IF Y=89 THEN 520
430 CALL CLEAR
450 PRINT :" you did not pr
ess Y or N.": : : : : : : :
: : : : : :
490 FOR DELAY=1 TO 500
500 NEXT DELAY
510 GOTO 280
520 CALL CLEAR
530 PRINT "   YOU MUST STOP
THE FALLING"
540 PRINT "BOMB BY EXPLODING
 IT IN MID-AIR.": : :
570 PRINT "   -MOVE THE CROS
SHAIR-": :
590 PRINT "  left :HOLD THE
s KEY"
600 PRINT "  right:HOLD THE
d KEY"
610 PRINT "  up   :HOLD THE
e KEY"
620 PRINT "  down :HOLD THE
x KEY"
640 PRINT :"   WHEN THE BOMB
 AND THE"
650 PRINT "CROSSHAIR ARE LIN
ED UP,"
660 PRINT "FIRE BY PRESSING
THE SPACE"
670 PRINT "BAR. THE SOONER Y
OU GET THE"
680 PRINT "BOMB, THE HIGHER
YOUR SCORE.": : : :
720 PRINT "   PRESS any key
TO START"
730 CALL KEY(3,S,STATUS)
740 IF STATUS=0 THEN 730
750 CALL CLEAR
760 CALL COLOR(8,2,1)
770 PRINT "        GOOD LUCK!
!!": : : : : : : : : : :
810 IF R=82 THEN 840
820 GOSUB 2090
830 GOTO 860
840 FOR I=1 TO 250
850 NEXT I
860 CALL CLEAR
870 GOSUB 2300
880 IF T=20 THEN 1860
890 T=T+1
900 CCROSS=16
910 RCROSS=21
920 RBOMB=1
930 CALL SCREEN(6)
940 CBOMB=INT(RND*29)+2
950 H$=STR$(T)
960 ROW=2
970 COL=3
980 GOSUB 2520
990 SCORE=P*Q*10
1000 H$=STR$(SCORE)
1010 ROW=5
1020 GOSUB 2520
1030 FOR I=1 TO 70
1040 NEXT I
1050 FOR I=2 TO 5 STEP 3
1060 CALL HCHAR(I,3,32,6)
1070 NEXT I
1080 OLDRCROSS=RCROSS
1090 OLDCCROSS=CCROSS
1100 CALL KEY(3,A,STATUS)
1110 IF A<>69 THEN 1130
1120 RCROSS=RCROSS-SGN(RCROS
S-1)
1130 IF A<>88 THEN 1150
1140 RCROSS=RCROSS+SGN(22-RC
ROSS)
1150 IF A<>68 THEN 1170
1160 CCROSS=CCROSS+SGN(31-CC
ROSS)
1170 IF A<>83 THEN 1190
1180 CCROSS=CCROSS-SGN(CCROS
S-2)
1190 IF RBOMB=1 THEN 1210
1200 CALL VCHAR(RBOMB-1,CBOM
B,32)
1210 IF (RCROSS=OLDRCROSS)*(
CCROSS=OLDCCROSS)THEN 1230
1220 CALL VCHAR(OLDRCROSS,OL
DCCROSS,32)
1230 CALL VCHAR(RCROSS,CCROS
S,130)
1240 CALL VCHAR(RBOMB,CBOMB,
129)
1250 RBOMB=RBOMB+1
1260 IF RBOMB=23 THEN 1540
1270 IF (RCROSS=RBOMB-1)*(CC
ROSS=CBOMB)THEN 1290
1280 GOTO 1080
1290 CALL KEY(3,B,STATUS)
1300 IF B=32 THEN 1330
1310 GOTO 1080
1320 REM BOMB DESTROYED
1330 RBOMB=RBOMB-1
1340 CALL SCREEN(10)
1350 CALL VCHAR(RBOMB,CBOMB,
32)
1360 CNT=0
1370 C1=92
1380 C2=47
1390 FOR I=-1 TO 1 STEP 2
1400 CALL VCHAR(RBOMB+I,CBOM
B+I,C1)
1410 CALL VCHAR(RBOMB+I,CBOM
B-I,C2)
1420 NEXT I
1430 C1=32
1440 C2=32
1450 IF CNT=1 THEN 1510
1460 CNT=1
1470 FOR VOL=10 TO 30 STEP 5
1480 CALL SOUND(100,-6,VOL)
1490 NEXT VOL
1500 GOTO 1390
1510 P=P+1
1520 Q=Q+(23-RBOMB)
1530 GOTO 880
1540 REM BOMB HITS THE CITY
1550 CALL VCHAR(22,CBOMB,32)
1560 CALL SCREEN(9)
1570 CALL COLOR(12,11,1)
1580 CALL VCHAR(23,CBOMB-1,1
22)
1590 CALL VCHAR(23,CBOMB,32)
1600 CALL VCHAR(23,CBOMB+1,1
23)
1610 CALL VCHAR(24,CBOMB-1,1
24)
1620 CALL VCHAR(24,CBOMB,125
)
1630 CALL VCHAR(24,CBOMB+1,1
26)
1640 FOR I=1 TO 20
1650 NEXT I
1660 CALL COLOR(12,7,1)
1670 CALL SCREEN(12)
1680 FOR I=1 TO 20
1690 NEXT I
1700 CALL SCREEN(7)
1710 FOR VOL=24 TO 1 STEP 4
```

```
1720 CALL SOUND(200,-7,VOL)
1730 NEXT VOL
1740 FOR DVOL=1 TO 24 STEP 4
1750 CALL SOUND(200,-7,DVOL)
1760 NEXT DVOL
1770 FOR J=23 TO 24
1780 FOR I=CBOMB-1 TO CBOMB+
1
1790 CALL VCHAR(J,I,32)
1800 NEXT I
1810 NEXT J
1820 CALL VCHAR(RCROSS,CCROS
S,32)
1830 CALL COLOR(12,2,14)
1840 M=M+1
1850 GOTO 880
1860 CALL CLEAR
1870 CALL SCREEN(4)
1880 CALL COLOR(8,5,16)
1890 PRINT "          GAME OV
ER": : : : :
1930 PRINT "   DESTROYED   "
;P
1950 PRINT :"   MISSED
";M
1970 PRINT :"   TOTAL POINTS
";P*Q*10: : : : :
2010 PRINT "   PRESS r TO PL
AY AGAIN": : :
2040 CALL KEY(3,R,STATUS)
2050 IF STATUS=0 THEN 2040
2060 IF R=82 THEN 160
2070 END
2080 REM READ CITY DATA
2090 FOR ROW=2 TO 1 STEP -1
2100 FOR COL=1 TO 32
2110 READ BUILDING(COL,ROW)
2120 NEXT COL
2130 NEXT ROW
2140 REM CUSTOM CHAR & COLOR
S
2150 CALL CHAR(136,"FFABFFAB
FFABFFFF")
2160 CALL CHAR(128,"003C7EFF
FFFF7E42")
2170 CALL CHAR(131,"42665A66
42427E66")
2180 CALL CHAR(132,"60606060
60606060")
2190 CALL CHAR(133,"607858F8
D8F8D8F8")
2200 CALL CHAR(134,"F8A8F8A8
F8A8F8F8")
2210 CALL CHAR(135,"C3C3FFAB
FFABFFFF")
2220 CALL COLOR(14,7,12)
2230 CALL CHAR(122,"80402010
08040201")
2240 CALL CHAR(123,"01020408
10204080")
2250 CALL CHAR(124,"80E0F8FE
FFFFFFFF")
2260 CALL CHAR(125,"81422418
0081C3E7")
2270 CALL CHAR(126,"01071F7F
FFFFFFFF")
2280 RETURN
2290 REM SET UP CITY
2300 FOR ROW=2 TO 1 STEP -1
2310 FOR COL=1 TO 32
2320 BLOCK$(ROW)=BLOCK$(ROW)
&CHR$(BUILDING(COL,ROW))
2330 NEXT COL
2340 NEXT ROW
2350 FOR ROW=2 TO 1 STEP -1
2360 FOR COL=1 TO 32
2370 PLACE(ROW)=ASC(SEG$(BLO
CK$(ROW),COL,1))
2380 CALL HCHAR(ROW+22,COL,P
LACE(ROW))
2390 NEXT COL
2400 NEXT ROW
2410 RETURN
2420 REM CITY DATA
2430 DATA 136,134,131,135,13
3,136,136,133
2440 DATA 135,136,136,136,13
3,136,136,135
2450 DATA 135,136,136,134,13
3,136,136,136
2460 DATA 135,132,136,32,131
,135,132,135
2470 DATA 134,133,128,32,132
,32,135,32
2480 DATA 32,32,134,132,132,
32,133,32
2490 DATA 32,32,128,32,132,3
2,133,135
2500 DATA 32,132,132,32,128,
32,132,32,@
2510 REM HORIZONTAL # PRINT

2520 FOR I=1 TO LEN(H$)
2530 DIGIT=ASC(SEG$(H$,I,1))
2540 CALL HCHAR(ROW,COL+I,DI
GIT)
2550 NEXT I
2560 RETURN
```

■

MUSIK MED XB

av Lars-Erik Svahn
(Repris från PB 86-2)

```
100 CALL CLEAR
110 PRINT TAB(6);"Inledninge
n till":TAB(8);"Gloriasatsen
": :TAB(6);"'Iste confessor'
": :TAB(7);"av  Palestrina":
 : : : : : :
120 GOSUB 400 !INITIALIZE
130 !
140 CALL TRIO(3/2,D2,G2,B1)
150 CALL TRIO(2,D2,A2,A1)
160 CALL TRIO(2,E2,G2,B1)
170 CALL TRIO(4,E2,G2,C2)
180 CALL TRIO(4,E2,E2,C2)
190 CALL TRIO(2,D2,F@2,A1)
200 CALL TRIO(2,C2,G2,G1)
210 CALL TRIO(2,D2,G2,A1)
220 CALL TRIO(2,D2,F@2,A1)
230 CALL TRIO(2,B1,G2,G1)
240 CALL TRIO(2,D2,G2,G1)
250 CALL TRIO(2,D2,F1,A1)
260 CALL TRIO(2,C2,F1,A1)
270 CALL TRIO(2,D2,F1,A1)
280 CALL TRIO(2,E2,E1,G1)
290 CALL TRIO(2,F2,D1,A1)
300 CALL TRIO(2,F2,D1,B1)
310 CALL TRIO(2,E2,C1,C2)
320 CALL TRIO(2,E2,C3,C2)
330 CALL TRIO(2,F2,A2,C2)
340 CALL TRIO(2,E2,C3,C2)
350 CALL TRIO(1,D2,B2,G1)
360 CALL QUIET
370 STOP
380 !
390 !INITIALIZE
400 A0=110 :: A@0=117
410 B0=123
420 C1=131 :: C@1=139
430 D1=147 :: D@1=156
440 E1=165
450 F1=175 :: F@1=185
460 G1=196 :: G@1=208
470 A1=220 :: A@1=233
480 B1=247
490 C2=262 :: C@2=277
500 D2=294 :: D@2=311
510 E2=330
520 F2=349 :: F@2=370
530 G2=392 :: G@2=415
540 A2=440 :: A@2=466
550 B2=494
560 C3=523 :: C@3=554
570 D3=587 :: D@3=622
580 E3=659
590 F3=698 :: F@3=740
600 G3=784 :: G@3=831
610 A3=880 :: A@3=932
620 B3=988
630 PSE=20000
640 RETURN
650 !
660 SUB TRIO(T,P,H,C)
670 FOR A=0 TO 28 STEP 0.9*T
 :: CALL SOUND(-500,P,A,H,20
,C,16):: NEXT A
680 SUBEND
690 !
700 SUB QUIET
710 CALL SOUND(1,110,29,110,
29,110,29)
720 SUBEND
730 !
740 END
```

■

NUMTALK MED SPEECH

av Anders Persson

CALL SAY("123") uttalar talet one-two-three medan CALL SAY_NUM("123") kommer att säga one-hundred-and-twenty-three. Programmet NUMTALK kan uttala alla tal mellan 0 och 999. NUMTALK klarar även positiva och negativa tal, heltal, decimala tal och 10-potenser som exponent. Tal större än 999 uttalas siffra för siffra beroende på att THOUSAND saknas i ordlistan för Speech Synthesizer. Programlistningen i Micropendium rad 25130 har fel variabel. NRA ska vara NR så att tal på 1000 och uppåt uttalas riktigt.

```
100 ! TEST NUMTALK
110 FOR I=0 TO 110
120 CALL SAY_NUM(I)
130 NEXT I

25000 !NUMTALK, a subprogram
 which allows pronunciation
of numbers correctly in a CA
LL SAY statement
25010 !Can be used in a prog
ram only. Correct format is:
 CALL SAY_NUM(#)
25020 !# Can be any numerica
l data between 0 and 999.
25030 !Keep NUMTALK in MERGE
d format, to be merged with
any program that may be need
ed.
25040 ! Author: Anders Perss
on, Lund, Sweden
25050 SUB SAY_NUM(NR)
25060 IF INITED THEN 25120
25070 DIM TEXT$(33)
25080 RESTORE 25370
25090 FOR I=1 TO 33 :: READ
TEXT$(I):: NEXT I
25100 NUMPOS$="-+.E012345678
9"
25110 INITED=-1
25120 NUM$=STR$(NR)
25130 IF ABS(NR)>=1000 OR AB
S(NR)<10 THEN 25210
25140 NEG=(NR<0)
25150 IF NEG THEN NUM$=SEG$(
NUM$,2,LEN(NUM$)):: NR=ABS(N
R):: CALL SAY(TEXT$(1))
25160 IF NR>=100 THEN GOSUB
25240 !SAY HUNDREDS
25170 ON ERROR 25400
25180 IF VAL(NUM$)>=20 THEN
25300 !SAY TY'S
25190 IF VAL(NUM$)>=10 THEN
25350 !SAY TEENS
25200 !SAY DIGITS
25210 FOR I=1 TO LEN(NUM$)::
 CALL SAY(TEXT$(POS(NUMPOS$,
SEG$(NUM$,I,1),1))):: NEXT I
25220 SUBEXIT
25230 !SAY HUNDREDS
25240 SPEAK$=TEXT$(POS(NUMPO
S$,SEG$(NUM$,1,1),1))&TEXT$(
33)
25250 IF SEG$(NUM$,2,2)<>"00
" THEN SPEAK$=SPEAK$&"+AND"
25260 NUM$=STR$(VAL(SEG$(NUM
$,2,LEN(NUM$)))):: IF NUM$="
0" THEN NUM$=""
25270 CALL SAY(SPEAK$)
25280 RETURN
25290 !SAY TY'S
25300 SPEAK$=TEXT$(VAL(SEG$(
NUM$,1,1))+23)
25310 IF SEG$(NUM$,2,1)<>"0"
 THEN SPEAK$=SPEAK$&"+"&TEXT
$(POS(NUMPOS$,SEG$(NUM$,2,1)
,1))
25320 CALL SAY(SPEAK$):: NUM
$=SEG$(NUM$,3,LEN(NUM$))
25330 GOTO 25210 !TO SAY DIG
ITS
25340 ! SAY TEENS
25350 CALL SAY(TEXT$(INT(VAL
(NUM$))+5)):: NUM$=SEG$(NUM$
,3,LEN(NUM$))
25360 GOTO 25210 ! TO SAY DI
GITS
25370 DATA NEGATIVE,,POINT,E
,ZERO,ONE,TWO,THREE,FOUR,FIV
E,SIX,SEVEN,EIGHT,NINE
25380 DATA TEN,ELEVEN,TWELVE
,THIRTEEN,FOURTEEN,FIFTEEN,S
IX+TEEN,SEVEN+TEEN,EIGHT+TEE
N,NINE+TEEN
25390 DATA TWENTY,THIRTY,FOR
TY,FIFTY,SIXTY,SEVENTY,EIGHT
Y,NINETY,+HUNDRED
25400 RETURN 25410
25410 ON ERROR STOP :: SUBEN
D
```

■

DISPLAY/AT

by B.A. Traver, USA

Programmer: Do you like to program in Extended BASIC but hate to compute the DISPLAY AT statements? Well, now you can use your TI-Writer to compose your screen and let this program write the Extended BASIC program for you (in MERGE format)

The TI-Writer includes many convenient features, such as full screen control of the cursor. The screen you are now viewing made use of those features in its composition. This program converted the TI-Writer file to DISPLAY AT statements.

After you have created a blank TI-Writer file to work with (using option 2), load it into the TI-Writer text editor and prepare your program screen. (Erase any guidelines that enter your workspace while editing in fixed mode; those outside won't hurt anything.)

When you have finished, save your screen to disk, using either the Save File ("SF") or Print File ("PF") option. (This program can handle either.) If you use a different filename, your original blank file will be available again.

Option 3 will do your programming work for you, and you can observe on the screen the progress of the program. The result will be a program on disk in MERGE format, complete with all those DISPLAY AT statements you hate to write.

After entering NEW, you can merge that new file into memory, RESequence it as you think best, and then save it to disk in MERGE format once more for later use in your Extended BASIC program when you want to make use of it.

Enjoy!
(Sänd skiva och frankerat svarkuvert till redaktören för en kopia av programmet)■