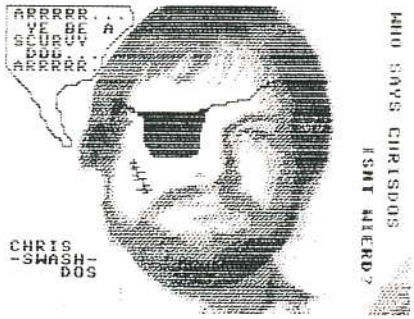
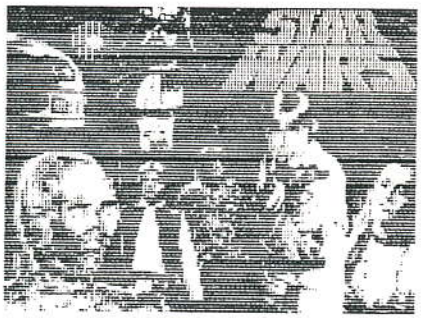
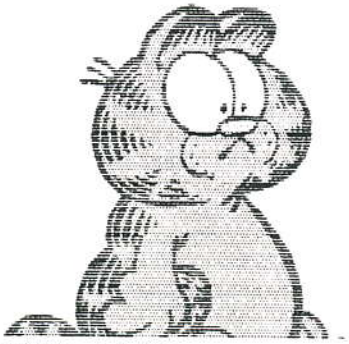
PROGRAM BITEN 94-2







ISSN 0281-1146

REDAKTÖREN

Det är nu dags att fundera på föreningens framtid. Jag har själv kommit fram till att jag inte kan fortsätta som redaktör nästa år. Om det finns någon medlem som kan och vill fortsätta nästa år som redaktör så bör du snarast kontakta styrelsen. Du måste även kunna göra det trovärdigt att du kan göra jobbet självständigt, eftersom det annars är oansvarigt mot medlemmarna att fortsätta utan en fungerande redaktör. Det är hittills 27 medlemmar som betalt avgiften för 1994.

Jim Peterson som skötte Tigercub dog den 12 januari 1994 i en ålder av 70 år. Tigercub kommer inte att fortsätta enligt Charles Good. ■

PASCAL – ARTIKLAR

UCSD PASCAL

NN 83-2.12	99:ans många språk
PB 84-1.04	Sortering
PB 84-2.05	Sortering med procedur i assembler
PB 85-1.26	Upprop Pascal
PB 86-4.05	Intressse(grupp) för Pascal?
PB 87-3.06	Tips för Pascal
PB 88-3.15	RAM-disk, Myarc-disk och tillbehör till p-system
PB 89-3.08	UCSD Pascal Databas
PB 90-1.16	Pascal MODCOL & BOOT

TURBO-PASC'99

PB 87-3.18	Turbo Pasc'99 och dietprogram
PB 88-4.02	Turbo Pasc'99 från L.L.Conner Enterprice
PB 91-4.02	Turbo Pasc'99 Tutor

BITS,BYTES&PIXELS (Lima Ohio UG)

Jan 94	Charles Good: P-CODE Card for Pascal
Mar 94	Anders Persson: Pascal with the TI 99 ■

HJÄLP MIG MED 99:A

Jag kommer att flytta utomlands och kommer att bo mycket trångt. Frågan är hur jag skall göra med min TI-99/4A. Henry Nilsen, Hagagatan 13, 652 20 KARLSTAD (endast dator+XB) ■

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

ÅRSREDOVISNING FÖR ÅR 1993

FÖRENINGEN PROGRAMBITEN
ÅRSBERÄTTELSE 1994-03-05
STYRELSENS ÅRSREDOVISNING

Verksamheten 1993 har bestått av utgivning av fyra nummer av tidningen och tre protokollförda sammanträden.

BALANSRÄKNING

INGÅENDE BALANS 1993

TILLGÅNGAR		SKULDER	
POSTGIRO	9733.74	LEV.SKULDER	00.00
VARULAGER	00.00	FÖRBET.MEDL.AVG.	1560.00
SKATTEFORDRINGAR	00.00	EGET KAPITAL	8173.74
SUMMA	9733.74	SUMMA	9733.74

UTGÅENDE BALANS 1993

TILLGÅNGAR		SKULDER	
POSTGIRO	6159.24	LEVERANTÖRSSKULDER	00.00
VARULAGER	00.00	FÖRBET.MEDL.AVG.	120.00
SKATTEFORDR.	00.00	EGET KAPITAL	6039.24
SUMMA	6159.24	SUMMA	6159.24

RESULTATRÄKNING

INTÄKTER		KOSTNADER	
MEDLEMSAVGIFTER	4180.00	TRYCKKOSTNADER	3487.00
PROGRAMFÖRSÄLJN.	115.00	POSTGIROAVGIFT	48.00
		FÖRTÄRING/MÖTEN	415.00
ÅRETS FÖRLUST	2257.60	PORTO/KUVERT	2485.60
		SKATT	117.00
SUMMA	6552.60	SUMMA	6552.60

STOCKHOLM SOM OVAN
CLAES SCHIBLER JOHN HANSSEN ■

SPEECH LOAD

(repris från PB 86-4.14)

```
100 CALL INIT
110 S=-27648
120 FOR I=1 TO 1000
130 NEXT I
140 PRINT "START ..."
150 FOR X=1 TO 20
160 REM CALL SAY("#THAT IS I
NCORRECT#")
165 CALL LOAD(S,70,"",S,65,"
",S,72,"",S,70,"",S,64,"",S,
80)
170 FOR T=1 TO 30
180 PRINT ">";
190 NEXT T
200 NEXT X
210 PRINT "STOPP ..."
```

■

SWEDLOW TI BITS * 34-35 *

by Jim Swedlow, USA

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

FEST WEST 91 WRAP UP REPORT

Fest West 91 can now be added to the record book as another in a series of highly successful Fest Wests. Attendance exceeded expectations at over 250. Everyone seemed to have a great time. Most major TI software and hardware dealers and authors were represented along with TI owners from across the country.

The Fest, which was hosted by the User Group of Orange County (UGOC) and Pomona User Group, was held at the Ramada Maingate, just across the street from Disneyland. Included in the Fest Guide was a map of all the attractions, restaurants and other facilities that were within walking distance. This was very helpful to visitors from out-of-town.

The Fest honored the tenth anniversary of the TI with banners, balloons and a special retrospective written by Bill Gaskill.

Software and Hardware Dealers:
There were representatives from 9640 News, Asgard, Bill Gaskill, Bud Mills Services, Comprodine, Genial Computerware, JP Software, Ken Hamai Hardware, LA Marketplace, MS Express, Notung Software, Pomona User Group, Rave 99, Regina, Southwest 99ers, T & J Software, TAPE, Tex-Comp, TI-Tax, and UGOC.

There was a wealth of items to purchase and many happy 4A owners walked out with new merchandise or with something from the overflowing consignment table.

Major Winners:
There were three different types of drawings at Fest West 91. Hourly drawings included items kindly contributed by the dealers present. The winners were too numerous to name. Two major winners, however, deserve special mention:

o Ted Whomsley won the free night at the Ramada Maingate.

o Mary Phillips from the Ozarks User Group in Missouri won a fully assembled and tested Horizon RAMdisk, that the Fest West committee purchased from Bud Mills Services. Mary was overjoyed at her good fortune.

o H. R. Jeffery won the door prize; an Asgard Mouse.

The Best of TI:
To honor the tenth anniversary of the TI, everyone who came was asked to vote for the "Best of TI". Ballots were collected on noon Sunday and the winners announced at the Fest. They were:

o HARDWARE: Bud Mills Services and RAVE 99 tied as the best sources of hardware.

o PUBLICATION: Far and away, the clear winner of the best TI publication was MICROpendium.

o WRITER: Regina was picked as the best TI writer of all time. Honorable mention went to Barry Traver and Beery Miller.

o SOFTWARE: There was no winner in the software category because so many fine items were mentioned.

During the Fest, Club 99 of Covina, California presented Jerry Price of Tex-Comp with a plaque to recognize his service to the TI community over the years. Jerry was surprised and touched.

Speakers:
Many fine luminaries in the TI community spoke at the Fest. They included:

o Ken Hamai on Disk Drives

o Berry Miller on 9640 Programming

o Ken Gilliland of Notung Software

o Bud Mills on RAM disks and new offerings from Bud Mills Services and OPA

o Regina on Programming in BASIC

o Bill Gaskill on TI Base

o Bill Chavanne on Multiplan and TI-Tax

o Barry Traver on Programming

o John McDevitt on new items from RAVE 99

o Rodger Merritt on graphics and new items from Comprodine

The speaker sessions were well attended and received.

Many Thank You's:
There are so many people and organizations that helped make Fest West 91 successful that the list could go on and on. A few, however, deserve special mention:

o The Riverside User Group (RUG), Southern California Computer Group (SCCG) and the Pomona User Group, all of California, helped some TI notables attend by partially defraying their expenses.

o All of the dealers who kindly contributed merchandise and discounts for the drawings.

o MICROpendium send 100 catalogs for distribution to those present.

o Southwest 99ers ran the registration process.

o Southern California Computer Group provided a major assistance in running the consignment table.

o Cris Van Allen created the giant Fest poster, made the vendor banners, designed the official Fest West 91 Tee Shirt, was our official photographer and much more.

o TM Direct, the newest TI vendor, sent us catalogs and items for the drawings.

o Special thanks also to Gloria Anders, Stan Corbin, George Dearmin, Eugene Gibson, Daniel Hatheway, Steve Luest, Howard McDonald, Erwin Metz, Bill Mooney, Earl Raguse, Janice Shafer, Shirley Swedlow, and everyone else who helped out.

FEST WEST 91 Committee:
A recap of Fest West 91 cannot be complete without mention of the Organizing Committee who spent a year bringing this event from concept to reality.

o Siles Bazerman coordinated the Speakers and the Friday night get together.

o Gene Bohot took care of promos, printing and keeping us on track.

o Bill Harms was in charge of user group relations, the tenth anniversary celebration and running the front table.

o Bill Nelson did the outstanding graphics, coordinated with the hotel and hosted the uncounted planning meetings.

o Jerry Rash served as the treasurer and organized volunteers and loaner systems.

o Jim Swedlow coordinated with vendors, served as secretary and announced all of the drawings.

Truly, Fest West 91 lived up to its slogan as the HAPPIEST FEST ON EARTH!

FEST WEST 91 - BEHIND THE SCENES

Now that Fest West 91 is over, I would like to take a little space and talk about what worked, what didn't and why.

WHAT WORKED: The success of Fest West 91 is due to the hard work of the Planing Committee. Six of us worked for a year to plan and execute Fest West 91 and that effort was what made the difference. How we proceeded that may be of use to future committees:

o We put as few items to vote as possible. Decisions were made by consensus. If we did not agree, we

let the item mature a bit before addressing it. Along the same lines, we had no chairman or president of the planning committee.

o Our choice of place and time was a big plus. Our event was on a three day weekend (President's Day). We picked a hotel run by a national chain that was just across from Disneyland. Hence our slogan, "the happiest fest on earth".

o There were a lot of phone calls to vendors, speakers and other user groups. We also made a concerted effort to visit local user groups frequently. This helped increase attendance and vendor participation.

o Our secretary kept and produced minutes of every meeting. This helped us keep track of our decisions and commitments. Part of the minutes was a list of items to be completed by the next meeting. Of equal use was an agenda ready before every meeting.

o We also started a time line which tracked, month by month, items that needed to be done. If, in June, we decided to discuss something in December, it was added to the time line so that it would not get lost.

o We sold all of our vendor tables a month before the fest. Two things contributed to this. First, we asked for a $10 deposit to hold the table, reducing the initial investment. Second, when table sales started to move, we wrote to all vendors a second time and warned them that we expected all tables to sell before the event.

o We were frugal. As we started planning, we heard rumors that two TI Fares had limited success and we became concerned that the support might not be there for ours. As a result, we kept expenditures to an absolute minimum and avoided a number of "nice but not necessary" items. Although our Fest was well attended, our frugal fiscal policy paid off. Although I do not have the final numbers in front of me, it appears that we broke even, which was our goal.

o Seed money was not a problem. The two sponsoring User Groups (UGOC and Pomona) lent us money. This, combined with a sizeable loan from one member, got us going. To generate early ticket sales, we asked the hotel for a free room. Tickets sold before January 1st were put in a drawing for that room. This generated some 60 tickets sales well in advance, which gave us needed operating capital.

o We had help. Southwest 99ers (Arizona) graciously agreed to run the registration process and San Diego (California) helped with the consignment table. The members of the sponsoring User Groups volunteered to help us run the other items that must be staffed to operate a Fest.

o We told vendors from the very beginning that we would appreciate one item per vendor for the drawings. We reminded them of this when we confirmed their reservation. During the fest, we asked them for their contribution. Asking without any qualifications was quite successful as every vendor contributed.

o When we did our initial (and subsequent) mailing to vendors, we asked them about power and equipment needs. When we confirmed their reservation we told them what we understood they needed. This led to a few phone calls but we knew what we needed before the Fest.

o We spend a lot of time developing a Fest Book that included a walking map of all the local businesses. This proved to be helpful for our out of town visitors.

o We sent confirming letters to our speakers regarding day, time and subject. As a result, we had a speaker schedule two weeks before Fest West 91 and only had to make two minor changes.

WHAT DIDN'T: Of course, not everything went off as we expected. It has been said that "planning is priceless - plans are useless but planning is priceless". Murphy's law struck us:

o We purchased commemorative Fest West 91 ball point pens to sell. They did not move. Memorabilia was not a big item.

o Vendor tables were $25 and included two free passes per vendor (not per table). I think we could have charged more per table. We did not charge for loaner systems and that was as it should have been.

o We had some misunderstandings with a few User Groups about this and that. We made a dedicated effort to keep this to a minimum. Sore feelings would hurt attendance and our income. More importantly, one key element of the 4A survival net is the user groups. We must work together if we are to continue.

o We had reserved a room at a local coffee shop for a no host Saturday night dinner. We forgot to reconfirm the reservation Saturday (it got a bit busy) and they lost the reservation. It worked out OK but there was a bit of confusion at first.

PARTING NOTE: Perhaps the key element to our success was our committee. Six of us spent a year working on the Fest, planning, discussing, working, working and working. It was a lot of hard work. Work continued through the end of the Fest keeping our participation as 4A owners to a minimum. On the other hand, while we are all glad to be done with it, I think that I can safely say that none of us would have missed it for anything. Well, almost anything!

Enjoy. ■

PC99 – A FIRST REPORT

by Charles Good, Lima Ohio User Group, USA (Feb 1994)

A growing number of 99/4A users also own IBM compatible computers. Such owners known to me include myself, Dave Szippl and some other local members of the Lima User Group, Bruce Harrison, Jim Peterson, Irwin Hott (runs the C.O.N.N.I. clearing house BBS), Jim Krych (of Asgard Peripherals), and Barry Traver. There are certainly many others. An obvious question for those who own both types of computers is, "Wouldn't it be nice if I could run some of my neat 99/4A software on my IBM compatible?" And for even those 99/4A owners who do not have their own personal IBM clone, "Wouldn't it be neat to run my favorite 99/4A software on my computer at work?"

I am writing this report using the 40 column Funnelweb v5 text editor running on my 386DX/40 IBM compatible computer. It will be printed right justified in condensed print with double strike in the appropriate places using the Funnelweb (TI Writer) formatter also running on my 386DX/40 system. PC99 is the magic that allows me to use this 99/4A software on my IBM clone. For years there has been talk of IBM emulation for TI users, maybe a new type of computer with two separate CPUs that would allow 99/4A and IBM software to run on the same machine. Something like this has now arrived! It doesn't use multiple CPUs and in fact it isn't even a new computer. The whole thing comes on a single 3.5 or 5.25 inch high density IBM compatible disk.

PC99 is a complete software emulation of an expanded 99/4A system designed to run from an IBM clone. Minimum requirements are a 386 or fancier (486, Pentium) processor, 640K memory, VGA graphics, and a hard drive. Disk size can be specified at the time of purchase. PC99 programs the clone's microprocessor to behave exactly as if it were an "all TI" expanded 99/4A system. This means the /4A console, 32K memory expansion, RS232/PIO interface, and TI disk controller with three DSSD drives on line. You can also purchase a version that behaves exactly like a 99/4 (without the

"A"). All you do is install the software onto your PC's hard drive. Once installed, just enter C: PC99 PC99 from the C:> prompt and your IBM clone will act EXACTLY like a 99/4A. The TI color bar title screen will appear on the VGA monitor, and when you "PUSH ANY KEY TO CONTINUE" you will get your choice of TI BASIC and any other TI cartridge (such as Extended Basic) that you chose to "plug in" to your PC99 computer. If you have a joystick or two connected to your clone's game ports, the joysticks will behave like TI joysticks when running TI software. Parallel and serial ports on your clone are addressed as "PIO" and "RS232" just like the real 99/4A when running TI software.

When you purchase PC99 you get the Tombstone City and Extended Basic "cartridges" as part of the initial purchase price. Cartridges and TI "disks" are files on your PC's PC99 subdirectory. The PC files that emulate TI "disks" do in fact behave exactly as if they were disks on a 99/4A system. When running PC99 you can OLD, SAVE, RUN, OPEN, CLOSE, and catalog the TI files that are on these PC99 "disks". Cartridges that run on PC99 are based on Gram disk files created on a real 99/4A system with a GramKracker, PGram, or similar Gram emulating device. If you don't have a gram device, the PC99 authors will sell you in PC99 format a legal copy of any TI cartridge ever produced. TI gets a royalty from such sales.

You always have access with PC99 to 3 DSSD "disks" and to any software on these disks. Converting actual physical TI disks (with all their Gram files, BASIC XBASIC EA3 and EA5 software, and their data files) to a PC99 "disk" file requires either cabling the IBM clone and 99/4A systems together via their serial ports or using the commercial software PC Transfer on a TI system with a double density floppy controller. The process is fairly straight forward for converting module gram files, and a bit more complicated for other types of TI software. But it can be done with a little time and careful reading of instructions. So far I have converted a complete Funnelweb system, and about a dozen modules. Once the conversion is done you can save your IBM "cartridge" and "TI disk" files to IBM floppies for archival purposes. The conversion only has to be done once. Utilities are provided to convert individual "TI" files or whole "TI" disks both ways, from the TI to the clone and from the clone to an actual 99/4A. I suspect PC99 owners will trade a lot of these TI "disk" and "cartridge" IBM compatible files.

Changing cartridges is easy! Just exit PC99 and from DOS run a little file called "LC" which means Load-Cartridge. Enter the file name followed by the name of the cartridge, such as "LC extended basic". Then the next time you start PC99, Extended Basic will be item #2 in the menu that follows the color bar title screen. Of course you need to have the PC99 extended basic gram files on your clone's hard drive to do this. One thing that takes some getting used to about "LC" is that cartridge names are case sensitive. In the above example, entering "LC Extended Basic" would not work because of the capital E and B. Most of the time DOS is not case sensitive, and LC runs out of DOS so you might think that case isn't important. It is!

There are some things you don't get with PC99. Because it only emulates official TI peripherals you only have three floppy drives available and these are only single density. PC99 does not emulate the double density 4 drive capacity of CorComp or other third party floppy controllers. 80 column software for the 99/4A and Geneve specific software will not run with PC99. You don't get to use ramdisks (you really don't need them), clock cards, or gram emulators with PC99. The current release of PC99 has no speech synthesis and only has one sound channel, because a typical clone has only one sound channel to its PC speaker. The next release is expected to support a PC sound card such as a "sound blaster". This will allow emulation of TI speech synthesis and full "3 channel plus noise generator" 99/4A sound.

Some additional observations about PC99, not necessarily in any order:

SPEED- PC99 is either slow or very slow, depending on the speed of the PCs microprocessor. Typing in BASIC listings, or responding to INPUT statements from within a running BASIC program can be done at normal typing speed. Almost everything else is slow, particularly sprites and other graphic intensive applications. I am told that the Funnelweb text editor works at near normal speed on a 486/DX2 66 machine. It should run even faster with a Pentium processor. On my 386/DX 40, the machine I am using to write this article Funnelweb's text editor is SLOW, too slow to really be practical. I have to wait after each typed letter for the letter to appear on screen before typing the next letter. There seem to be no speed problems printing through the Funnelweb formatter using PC99. I have used PC99 on a 386SX 20 machine at work with acceptable but slow results. This is just about the slowest microprocessor in the 386 line.

GAMES- I have been able to achieve some fantastic scores on TI game cartridges, fantastic for me anyway. These cartridges run slow enough that I can anticipate what is going to happen. Keyboard response or joystick response (or maybe just my response time) from PC99 seems more responsive than on an actual 99/4A. I now have no trouble eating all the dots in car wars and advancing outside the village to kill all the bad guys in tombstone city, things I can't do on a real 99/4A.

MATHEMATICAL ACCURACY- We pride ourselves on the 10 digit display and 13-14 digit accuracy of mathematical calculations done on a 99/4A. This is better than what you get using the various BASICs that run on IBM clones. On a 99/4A 1/3 x 3 = 1, whereas on many PCs it will equal 0.9999999 beacuse of rounding errors. PC99 is every bit as mathematically accurate as a 99/4A. This means that a 386 or 486 CPU on an IBM machine is NOT less accurate than a 9900 CPU on a 99/4A. The apparent lack of mathematical accuracy in PC BASICs is because of the way these BASICs are originally written in the PC's assembly language, not because of any deficiencies in 386 or 486 CPUs.

THE 360K DOUBLE DENSITY DISK PROBLEM - Transferring disks and files from a 99/4A to PC99 requires either cabling the computers together or using PC Transfer. PC Transfer allows you to put an IBM formatted 360K 5.25 inch double density disk into a double sided double density drive of a 99/4A system that has a double density controller. You can then, using only the 99/4A computer, transfer files from 99/4A formatted disks to 360K IBM formatted disks. The problem is that most modern IBM clones with 386 or fancier processors have only "high density" floppy drives. These drives can be used to read low capacity "double density" floppies but cannot be reliably used to format a "double density" floppy to only 360K. The DOS instruction book says this can be done, but it really can't. You either have to have access to an older IBM clone (such as my Tandy 100HX) with a "double density" 5.25 inch floppy drive or you need to puchase preformatted 360K "double density" 5.25 inch floppies. For most people it is probably most convenient to purchase preformatted 360K disks. They are available at Radio Shack and some discount department stores.

THE PC99 DEBUGGER- This is a superior assembly language tool that lets you set break points, move in single steps through software, etc. Since the PC99 debugger runs outside of the 99/4A environment, it allows access and analysis to ALL memory areas. No debugging tool running on a 99/4A can do this.

In conclusion: if you have a modern IBM compatible computer and you want to run all your favorite 99/4A software off your new machine, than PC99 is the only way to go. It seems to work perfectly, although PC99 is sometimes a bit slow. Cost for the current release is $120 from CaDD Electronics, 81 Prescott Rd., Raymond NH 03077, USA. 603-895-0119■

EXTENDED BASIC - SPRITES

by Andy Frueh, Lima Ohio User Group, USA

Sprites are handy little things. If you are interested in graphics or games at all, I can recommend buying the Extended BASIC cartridge simply for sprites. For those who don't know, sprites are characters that can be set in real motion. Instead of moving character by character (8 pixels) they can move 1 pixel at a time, giving very smooth movement.

XB has several built-in CALLs that relate to sprites. A few of the familiar CALLs such as CHAR and COLOR have been expanded so that the controls either the shape or colors of sprites. I'll get into that in a minute.

The first, and most essential, statement for sprite programming is the CALL SPRITE. It literally calls a sprite into being. The syntax for it is as follows: CALL SPRITE(sprite #,ASCII value,color,row,column,row speed,col speed) where the sprite # is a number between 1 and 28 used to label the sprite. The ASCII value can be pre-defined or defined by the user. It must be between 32 and 143. If you use double sized sprites (see below) than the character value you input is the first of a set of 4, the next three having consecutive values. For example, if you used 32 as your value, a double size sprite also uses 33, 34, and 35.

The sprite color can be any color between 1 and 16. Note that a color of 1, or one that matches the screen color will make the sprite invisible. Next, you specify the row. This is a dot (pixel) value between 1 and 192. It can go up to 256, but this is off of the screen. The column can be between 1 and 256. The sprite's position is where the upper leftmost part of the sprite will appear, rather than the center of the sprite as you might think. Keep this in mind if you need to place sprites exactly.

The next two numbers are optional. The row and column velocity can be between -128 and 127. Negative numbers go to the left, and positives to the right, with numbers being close to zero the slowest. A row and column speed of 0 stops the sprite.

A little while ago, I mentioned the use of COLOR and CHAR in sprites. If CHAR is used to define a character before it is turned into a sprite, that sprite assumes the defined shape. Thus, a sprite can be any shape. To use the COLOR subprogram, use CALL COLOR(sprite #,foreground). You don't define the background because with sprites, the background is always 1.

Next we have the CALL COINC routine. This is handy for chase-type games because it detects when sprites are touching. It comes in three forms. CALL COINC(sprite,sprite,tolerance, variable) lets you know when two sprites have met. The two sprites are named by their number. The tolerance is given in dots. For example, if the tolerance was 5, the sprites would "be touching" when they were 5 dots apart at any angle. This is preceded set to 1 or 0 in games, to enhance the precision of the judging of the game. The last part is the variable which returns different values depending on the sprite's coincidence.

A second form of COINC is CALL COINC (sprite,row,column,tol,var). This form reports whether or not a sprite is at or (depending on the tolerance) near a certain location. The final form is CALL COINC(ALL,var). The reports on all the sprites in use. It determines a coincidence if one or more of the dots that make the sprites are overlapping. The accuracy of this statement depends on many things. For example, a sprite that moves very quickly may not always register a coincidence every time one occurs. Also, the routine checks only when it is called, so if a sprite has a coincidence condition while the program is doing something else, it may not

be detected. Programmers beware!

Another statement we have to use is the CALL DELSPRITE. It erases certain sprites, or all of them. The syntax is CALL DELSPRITE(sprite/ALL). Note that with all routines that call for a sprite number, you can provide a numeral and perform some operation on it. For example, sprite #3-X is an acceptable sprite number, as long as the operation doesn't make the sprite number exceed the values between 1 and 28. Also, sprite numbers must be preceeded with a # sign. Anyway, either place a sprite number, list of numbers, or the word ALL after a DELSPRITE to completely erase that sprite. After it is erased, it must be recreated with a CALL SPRITE.

The DISTANCE subprogram is next in line. It tells us how far a sprite is from a certain location or another sprite. To use it, pick one of two forms. CALL DISTANCE(sprite, sprite,variable) to tell the distance between two sprites. The variable will return a value that equals the distance in dots. It is found in the following way. First, the distance between the dot-rows of the sprites (or the sprite and the location) is found and squared. Next, the square of the dot-columns is found. These two values are added together. If the value is higher than 32767 (what IS it with this number?) than 32767 is returned. The distance between one sprite and the other, or the location, is square root of the number returned by the variable. The other form is CALLS DISTANCE (sprite,row,col, var).

CALL LOCATE is the subprogram that lets us change the location of any sprite without having to actually move it, or recreate it. The syntax is CALL LOCATE(sprite,row,col). As soon as this statement is executed, it locates the sprite with the number you gave, and where it's at. It then makes it skip to the new spot.

The next, and most interesting, of the sprite controllers is the MAGNIFY subprogram. It has a simple syntax of CALLS MAGNIFY(size). All sprites are affected by it. Mag factors are between 1 and 4. A size of 1 is the standard size. A factor of 2 causes the sprites to be simply magnified in size. Factor 3 causes sprites to be double size, but unmagnified. The sprite is defined by 4 characters, not one. The first character is the upper left. The next goes in the lower left, then upper right. The final character is in the lower right corner. A factor of 4 is similar to 3, except that the sprites are magnified as well as doubled up.

Another interesting command is the MOTION command. It lets you change the speed an/or direction of any sprite. The syntax is CALL MOTION (sprite,row velocity,col velocity). The values may be from -128 to 127, with the rules being the same as those with the CALL SPRITE command.

A seldom-used, but nonetheless useful command is PATTERN. It lets you alter the character pattern of any sprite instantly. Type in CALL PATTERN(sprite e,value) where the sprite is the number of the one to change, and the value is the new ASCII code for the sprite.

Yet another command is POSITION. It returns two values for any sprite's current row and col. CALL POSITION (sprite,row,col).

Something I should mention is that with most of the sprite command subprograms, you can repeat the data over and over, so that one command will control several sprites. For example, you could use CALL LOCATE (sprite 1,row,col,sprite 2,row,col, ...)and so forth. You can repeat with the COLOR, DELSPRITE, LOCATE, MOTION, PATTERN, POSITION, and SPRITE commands.

I hope this clears some of the confusion with using sprites. The XB manual does a poor job, because it references other parts constantly, and repeats itself when you don't want it to. The examples are worth looking at, though, and to keep this article short, I didn't inlcude any example programs. However, I hope I cut through some of the stuff TI through at us in the manual. Note that the section on MAGNIFY is

wrong, and that they mixed up magnification factors 2 and 3. There may be other manual errors, but none that I could find out about (by the way, the XB manual WAS laced with errors. Some of us have an addendum about these problems. TI may still let you have a copy.). ■

FW XB FORTH LOADER

by Tony McGovern, Australia

This program allows the standard TI FORTH disk to be loaded by TI XB. It works only with the XB and E/A modules, and its primary use would be from the Funnelweb XB User List.■

FUNNELWEB "EVRAM" EDITORS

described by Charles Good, Lima Ohio User Group, USA (Feb 1994)

Tony McGovern has released "the first of several new versions of the Funnelweb v5.0 editor". This 80 column editor uses a fully expanded 9938/9958 video processor with 192K ram to permit a 64K text buffer. This is enormous by any TI standard, even larger than the text buffer of MYWORD running on a Geneve. In ShowDirectory you get a display of % free and number of text lines free. This new editor will handle over 3500 text lines.

Although I don't write large blocks of assembly code, I have found some uses for the extra text buffer.
1- Text files (such as CYC files) ported over via PC TRANSFER from an MS-DOS computer are often too large for the Funnelweb text editor. I can load them into the 64K program editor and them break them down conveniently into TI Writer sized segments.
2- When I create disk listings of Lima Library software I do this by printing DSKU disk reports to a disk file. The resulting DV80 disk file is often too large to fit into a normal text editor for editing, but easily fits into the new Funnelweb 64K editor.

Tony McGovern has also written a utility which converts PROGRAM files into a text file of the ASCII equivalent. The result is similar to the ASCII display of a sector editor. These ASCII text files can then be transferred to a PC disk using PC TRANSFER and uploaded to a BBS. Recipients of these ASCII text files can convert them back to PROGRAM files with the same utility. The ASCII-HEX conversion utility is an EA5 program. Run it, then type in the name of a file. If the file is a PROGRAM file you will get ASCII DV80 output. If the file is an ASCII file you will get output as a PROGRAM file. This is useful if your TI is not connected to the internet, but you know someone with a PC that is connected to the internet. You can then send and recieve TI PROGRAM files using MS-DOS computers, and convert them to/from TI disks using PC TRANSFER.

The 64K v5 editor is sharware. The ASCII-HEX conversion utility is public domain. They have been uploaded to GEnie and are also available to anyone who sends $1 to the Lima User Group, P.O. Box 647, Venedocia OH 45894.

I also have from Tony McGovern a preliminary version of the 80 column Editor that has TWO 64K buffers in which you can put text for editing. What used to be the V(iew) buffer has been made into a second edit buffer. This means two gigantic text edit buffers simultaneously in memory. You can put different text into memory in each of the two text buffers for editing and you can quickly switch from one buffer to the other. You can also cut and paste between the two buffers. FANTASTIC! Tony says "Keep this one for yourself- it is not in shape yet- it is not a secret either."

(FW EVRAM Editor Vn 5.00b för 192 kbytes VDP-RAM kan fås från redaktören genom att sända skiva och frankerat svarskuvert. Endast filerna ED/EE/EF utan manual. Måste laddas från Funnelweb 4.40) ■

FROM BASIC TO ASSEMBLY - 9

by Bob August, Bug News, USA

Last month we said we were going to show you how to put some color in your life. So this month we have the 40 column text program with a white screen with dark blue text. We did this by adding two lines of code:

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

This loads a hex 4F into VDP register seven. The 4 is the text color of dark blue and the "F" is the screen color of white. Remember that in assembly the color table starts with a zero instead of one like basic. Just subtract a one from all your basic colors and you have the correct number.

In basic we would enter the following:

```
100 CALL SCREEN(16)
110 FOR C=1 TO 12
120 CALL COLOR(C,5,1)
130 NEXT C
```

This is easy in text mode but a little more complicated in graphics mode. We will show you how to do it in graphics mode next month. Play with the program and change the colors. The more you change a program the more you learn.

HAPPY ASSEMBLING!

```
****************************************
* BASIC TO ASSEMBLY   Lesson Number 9  *
****************************************
*
       DEF  START                    Entry point of program
       REF  VSBW,VMBW,KSCAN,VWTR     Utilities used in program
*
WRKSP  BSS  32                       Workspace buffer
SAV11  BSS  2                        Save return address buffer
*
MESAGE TEXT 'This is in text mode and we have forty  '   Message in
       TEXT '                                        '
       TEXT 'columns on a white screen with the text '   40 columns
       TEXT '                                        '
       TEXT 'in dark blue.'
QUIT   TEXT 'Press the Enter key to quit'                Prompt to quit
*
       EVEN                  Make sure we start on even byte
*
* Start of program
*
START  MOV  R11,@SAV11       Save return address
       LWPI WRKSP            Load our workspace
       BL   @CLEAR           GOSUB CLEAR to clear the screen
*
* Set screen and text color
*
       LI   R0,>074F         Load VDP Register 7 with >4F
       BLWP @VWTR            ( Sets screen white, Text blue )
*
* Put into text form
*
       LI   R0,>01F0         Load VDP R1 with >F0
       BLWP @VWTR            Write if to VDP register
       SWPB R0               Switch left and right bytes
       MOVB R0,@>83D4        Store result in >83D4 for key scan
*
```

```
       BL   @DISPLY          GOSUB to DISPLY routine
       DATA 240,MESAGE,173   Screen location, Text, Length
*
       BL   @DISPLY          GOSUB to DISPLY
       DATA 726,QUIT,27      Screen location, Text, Length
*
*      Call Key routine
*
       CLR  @>8374           Clear to zero for CALL KEY(0,K,S)
       CLR  @>837C           Clear status to zero
       LI   R4,>2000                                  (Ed.change)
KLOOP  BLWP @KSCAN           CALL KEY(0,K,S)
       CB   @>837C,R4        Check status for key press    (Ed.change)
       JNE  KLOOP            If S=0 THEN KLOOP             (Ed.change)
       MOV  @>8375,R0        Move key press to register zero
       CI   R0,>0D           Compare to 13 or Enter key
       JNE  KLOOP            If not enter key, GOTO KLOOP
       CLR  @>837C           Clear status to zero
       MOV  @SAV11,R11       Put return address in register 11
       BLWP @0               Quit ( FCTN = )
*
* Clear screen routine
*
CLEAR  LI   R0,0             Load R0 with zero ( Row 1, Column 1 )
       CLR  R1               Clear Register one
       LI   R2,1             Load R2 with one ( Length byte )
CLOOP  BLWP @VSBW            Write a space to the screen
       INC  R0               Add one to R0
       CI   R0,959           Compare R0 to 959 ( 24 X 40, 0 to 959 )
       JLE  CLOOP            Jump to CLOOP if less then 959
       RT
*
* Display at routine
*
DISPLY MOV  *R11+,R0         Put screen location in R0
       MOV  *R11+,R1         Put Text in R1
       MOV  *R11+,R2         Put length of text in R2
       BLWP @VMBW            Write it to the screen
       RT                    Return to calling area
*
* End program with auto start
       END  START                                              ■
```

NEWS AND VIEWS #2 – Jan93

by Jim Peterson, Tigercub, USA

When I wrote an article entitled "I Like Brain Games", I neglected to mention three of the best - the Sliding Block Puzzles series programmed by Norman Rokke and sold by MS Express Software. In their Winter 1992 catalog (P.O. Box 498, Richmond OH 43944), Series I contains 3 puzzles, Series II contains 5, and Series III contains 13, and each Series sells for $7.95 + $1 S&H. Since they are infernally difficult, you may be glad to hear that you can also obtain the solutions to each series, at $7.95 + $1 S&H each.

The article mentioned that no one had put out a diskfull of cryptograms. Since then, Larry Tippett released as fairware a diskfull of humorous sayings, with a program to encode them as cryptograms and dump them to a printer, to be solved the old-fashioned way on paper with pencil and plenty of eraser. The sayings are rather too short, therefore quite difficult.

This inspired me to dig out and upgrade a cryptogram program I had written several years ago, which enables one to solve cryptograms on screen, with several help options. I needed a source of phrases of suitable length and the handiest source was the Bible, so I released a diskfull of 100 Bible Cryptograms. Since I followed the convention of quoting the book and verse number at the end of each verse, those who are familiar with the Bible find them quite easy to solve. Therefore I created a second diskfull of Song Cryptograms, using verses from songs.

Since then, Bruce Harrison has released a diskfull of cryptograms called Code Breaker which is rather bare-bones in appearance but lightning-fast in execution, as it is written entirely in assembly. It also offers several levels of difficulty including a super-difficult level in which the letters are grouped into series of five regardless of spacing or word length. It is still available from Harrison Software (5705 40th Place, Hyatsville MD 20781) for $8 ppd.

The same article mentioned that no one had yet programmed the TI to play an intelligent game of checkers. Asgard Software has now released Classic Checkers, which still does not play at all intelligently. It does offer the option of playing against an intelligent human, if you can find one who would rather play on a monitor screen than a checkerboard. The instructions say that it can be played with the Asgard mouse, joystick or keyboard. I do not have a mouse, and my joystick is broken; from the keyboard it does not allow diagonal moves although all checker moves are diagonal.

Asgard Software is also offering TI-Pei, a solitaire version of Mahjongg, programmed by Bill Reiss. I have not seen it. Both are available from Asgard Software at $14.95 each.

In a couple of articles recently I mentioned, based on what I had been told, that the TI-99/4A could not drive a 24-pin printer unless the printer had a 9-pin emulation, in which case only 9 pins would be used. Several people have written to tell me I was wrong about that, and even sent printouts to prove it.

Bruce Harrison has written a fast assembly routine to convert screen fonts to download fonts, and download them to his NX-1000 printer. He sent it to me, to try out on my NX-1020R. It put my printer off-line so thoroughly that the on-line command wouldn't even work - the printer had to be turned off and back on. So I loaned him my manual. He discovered that the NX-1020 in IBM mode uses somewhat different codes than the NX-1000 to download characters - and in standard mode uses ENTIRELY different codes. Wouldn't you think they could have standardized by now?

Woody Wilson notified me of a bug in my Printall V1.8, which will crash it if you attempt to print a DV 254 file you have previously printed to disk. In fact, it should crash the program in any case because I used a single variable which had been dimensioned and prescanned as a subscripted variable, but for some reason it didn't. Anyway, please fix -

```
300 LINPUT #2:M$(J):: PRINT #1:M$(J)&
CHR$(10):: IF EOF(2)<>1 THEN 300
310 RESTORE #2 :: NEXT J :: CLOSE #2 ::
GOTO 250
```

Also, if you are going to use download characters, remove the PRINT #1:X$&"@" in line 240.

I have previously mentioned a serious bug in J. Peter Hoddie's Sort Experiment. The documentation states that it will sort up to 24k of records, but fails to mention that when it has loaded 24k it will go into the sort - without warning you that it could not load the complete file!

Dolores Werths is preparing to send out the first disk of material of the MIDI SIG. If you are interested in Midi Master 99 and haven't yet sent her your dollar, be sure to do so! the address is 5705 40th Place, Hyattsville MD 20781. While you're at it, enclose a stamped self-addressed envelope for the Harrison Software catalog, and then order some of her superb MIDI disks.

If you are interested in genealogy, do you know about the NGS/CIG BBS? It is devoted strictly to genealogical research, no games or chit-chat or dirty jokes or whatever, but you can post messages requesting info about anyone you are researching and your message will be ECHO'd to many boards and read by many people with similar interests. The number is (703) 528-2612 and it's free.■

JUST A LITTLE BIT HELPS

by Jim Peterson, Tigercub, USA

I'm not going to say that everybody should learn to program. Most folks can get along just fine by using the programs written by others.

But, every once in a while you need to do something that would be so easy for the computer to do, and so hard to do without a computer - but no one has written a program to do it. That is when just a little bit of programming knowledge can be a lifesaver!

In a previous article, I described my genealogy program, and the index for it I had created in Funnelweb by setting the tabs at 1 for the person's index number, 5 for the first name, 36 for the surname, 56 for the paragraph number, 61 for the father's index number and 66 for the mother's, and 71 for the spouse.

I have been busily adding names to that index until I am now up to well over 900, and I realize that I have goofed!

I set that second tab to allow space at left for a 1- to 3-digit index number and a space, but I will soon be going past 999. I need an extra space there. I allowed more than enough spaces for the names, so I can take some of those away. While I'm at it, I would like to allow space for second and third spouse numbers at the right, and I want to keep the total spaces well short of 80 so I can print the index in two columns of elite condensed.

Am I going to retype 900+ lines, and undoubtedly make some errors while doing so? No way! Never do anything that you can make the computer do for you.

All this requires is knowing how to open one file to read from and another to write to, read a record from the one file, manipulate it a little, and write it to the other file.

The file part is extremely simple, just OPEN #1:"DSK1.OLDFILE",INPUT and OPEN #2:"DSK2.NEWFILE",OUTPUT - or whatever drives and filenames I choose. The computer takes it for granted that I'm using DV80 files, so I don't have to tell it. Even the INPUT and OUTPUT could be omitted, but it's best not to. If you don't tell it otherwise, the computer will allow you to accidentally write to the file you meant to read from, and that is disastrous!

The reading and writing is equally simple - 110 LINPUT #1:M$ :: PRINT #2:M$ :: IF EOF(1)<>1 THEN 110 ELSE CLOSE #1 :: CLOSE #2

Always use LINPUT when reading a text file, because INPUT will only read the record up to the first comma it finds.

EOF is set to zero until the last record in the file has been read, when it is set to 1, so IF EOF(1)<>1 means "if the end of the file opened as #1 has not been reached" - in which case, go back to the same line number and read another record, but if you have read everything, close the files - and especially close the one you have written to, or you will have wasted your time.

But, I wanted to change the lengths of the fields in each record before I move it to the new file. For that, I need to know only how to use two commands - SEG$ and &.

SEG$ extracts a part of a string - a string, in this case, is the record I get with that LINPUT, which is one line of up to 80 characters that I keyed in with Funnelweb. SEG$(M$,1,4) will give me 4 characters starting with the first character of M$; SEG$(M$,5,25) will give me 25 characters starting with the 5th. And & will put those two together into one string. I want to keep that first 4-character field but expand it to 5 characters, so SEG$(M$,1,4)&" " . The next two fields, starting at 5 and 36 and consisting of 31 and 20 characters, I want to cut to 20 and 16 characters, so - SEG$(M$,1,4)&" "&SEG$(M$,5,20)&SEG$(M$,36,16). And I go on to add a space to each of the next three fields, and the whole program looks like this -

```
100 OPEN #1:"DSK1.INDEX1",IN
PUT :: OPEN #2:"DSK1.NEWFILE
",OUTPUT
110 LINPUT #1:M$ :: PRINT #2
:SEG$(M$,1,4)&" "&SEG$(M$,5,
20)&SEG$(M$,36,16)&SEG$(M$,5
6,5)&" "&SEG$(M$,61,5)&" "&S
EG$(M$,66,5)&" "&SEG$(M$,71,
3)
120 IF EOF(1)<>1 THEN 110 EL
SE CLOSE #1 :: CLOSE #2
```

And in a few minutes, that program does the job, saving many hours of work! (If you print this through the Formatter, transliterate the & !!) ■

THE ORIGINAL TI SYSTEM

antiques decribed by Charles Good, Lima Ohio User Group, USA (Jan 1991)

The "original" TI Home Computer system, released to the public in 1979 and 1980, consisted of the 99/4 computer (without the "A") and a series of stand alone peripherals that plug directly into the side of the 99/4 (or 99/4A) console, or into the side of the previous peripheral (hence the unoffical descriptive term "freight train peripheral"). Each of these freight train peripherals except the speech synthesizer has a base that measures 17x26cm (a bit larger than 6.5x10 inches), a separate power supply rated at 0.2A (23 watts) at 115 volts, and its own separate power cord. I recently purchased a "4" (just to play with) and was later given many of the freight train peripherals. After using these devices for awhile I realize how fortunate we are to have the "4A"and its peripheral expansion box.

Components of the "original" TI home computer system are listed below, together with their official TI part numbers and some prices mostly quoted from an ad by CBM INC of Lexington KY published on page 12 of the first edition (May/June 1981) of 99er Magazine. These CBM INC prices are probably below TI's official list price. These peripherals are not the same as those designed to fit in the PE Box. PE box peripherals all have "PHA12xx" part numbers and are described in official TI publications as "cards".

--TI 99/4 console, (PHC004C): $499
--RF (TV) modulator; in 1980 this was an extra cost item,(PHA2100):$41
--Solid State Speech Synthesizer; the same one most of us still use, (PHP1500): $122
--32K RAM memory expanion, (PHP2200): $325
--RS232 Acessories Peripheral, (PHP1700): $183
--Solid State Thermal Printer, (PHP1900): $325
--Disk Drive Controller, (PHP1800): $243
--Disk Memory Drive, (PHP1850): $399
--PCode Peripheral, (PHP 2400): $400
--Video Controller, (PHP2300): $700

Prices of the last two items are official list prices quoted from TI's suggested retail price list dated June - December 1982 (1049705-1).

You can connect a maximum of three periphrals in series to the right side of the computer. If present, the speech synthesizer has to be first and the 32K second. A "typical" freight train minimum expansion system (99/4 with modulator, 32K, thermal printer, controller and one drive) would be almost four feet wide and cost $1832. Bringing the system up to the maximum of three SSSD drives and buying all the other freight train peripherals would bring the cost up to a total of $4035. Wow! And you can only have simultaneous use of 3 peripherals.

In this article I will describe what I know about these freight train peripherals. I have hands on experience with the Thermal Printer, 32K, and Disk Controller. I will not discuss the Speech Synthesizer since the 1979/80 device is the same one we are all familiar with. In a separate article I will describe my experiences with the 99/4.

--32K EXPANSION MEMORY: This functions exactly like the equivalent PE box card. These days you can, for about $10, buy a 32K RAM chip that measures about 1x3cm and draws very little current. It amazes me that TI's original 32K was so bulky and required a 23 watt power supply. A 12 inch black and white TV only draws 29 watts. But I guess if you compare a 1955 room sized UNIVAC computer in memory, watts of power consumption, and bulk, the vintage 1979 TI 32K looks pretty good.

--RS232: This stand alone box offeres only one RS232 port and no parallel port. The PE Box RS232 card allows connection of TWO RS232 (serial) devices (with a special Y cable) AND one paralle device all to

the same card. The PE Box card is obviously superior to the stand alone peripheral.

--DISK DRIVE CONTROLLER: This device used the original DISK MANAGER module (the DMI), and can control up to three SSSD stand alone drives. Double sided is not available with the freight train disk controller. The main difference between the DMI and DMII modules is that the "I" has no provision for double sided disk initialization. A TI stand alone drive plugs directly into the back of the freight train Disk Controller without the need for any special adapter cable other than the cable that comes with the stand alone drive. Other drives plug into the cable of DSK1 using a small adapter board. A special cable that comes with the PE box controller card is needed to plug a stand alone TI drive to the back of the controller card for use as DSK2 or DSK3. An interesting feature of the freight train controller in combination with the DMI module is that they do not recognize the "whole disk protected" byte >10 of sector zero. With the TI PE box controller and the DMII module, if this byte is set for a value of >50 you cannot copy the disk with the DMII.

--THERMAL PRINTER: This is printer device "TP", and was sold to TI users at a time when cheap dot matrix or daisy wheel printers cost $600+. The 1982 list price for the 99/4A's official dot matrix printer was $750. The TP uses 3.5 inch thermal paper, prints 32 characters per line, and like all thermal printers is both quiet and slow. 3.5 inch thermal paper rolls are a non standard size these days. TP users either have to purchase 10 year old official TI paper from one of the few TI dealers that stock this item, or use a paper cutter to trim 8.5 inch FAX paper rolls down to 3.5 inches. Such 8.5 inch wide FAX rolls are commonly available these days from many stores including SEARS, KMART, and WALMART. On the title page of the TP manual it says that the TP "prints a copy of a TI BASIC program or the screen displays from certain Command Modules." And that is about it! A few modules, such as MUSIC MAKER, allow screen dumps with the TP. You can specify output to the TP with the DMI, DMII, PRK, Statistics, LOGO2, and maybe a few other modules. You can't use the TP with TI Writer, Funnelweb, the EA module, or Microsoft Multiplan. From BASIC you can LIST a program to the TP, a common application. You can also OPEN a file to the TP using any of these file attributes: SEQUENTIAL or RELATIVE, DISPLAY or INTERNAL, OUTPUT or APPEND, FIXED or VARIABLE. I can't immagine what use RELATIVE, INTERNAL, or APPEND have in OPENing a printer file. When opened in INTERNAL, the printer prints a meaningless graphic of the internal representation of each ASCII character. The maximum length of a VARIABLE TP attribute is 32. All printed characters of the TP's built in character set are on a 5x7 dot grid. The TP has a unique graphic set for ASCII 0-31 and the usual alpha/numeric characters for ASCII 32-127. Each printed dot of a character is printed only once and individual dots can be seen with the naked eye. There is no way to make extra dense high quality characters. Emphasized, double strike, and "NLQ" is not available. The user can also, using an 8x8 dot grid, redefine ASCII chars 32-159 in BASIC using CALL CHAR, and then directly print any of these redefined chars to the TP with the appropriate keyboard keypress as in PRINT #1:"{" where { has been redefined, or with PRINT #1:CHR$(xxx). This is a neat trick! It is much harder to print redefined characters with other kinds of dot matrix printers.

--VIDEO CONTROLLER: A photograph and brief description of this peripheral appears as part of an article on page 53 of Volume 1, No. 4 of 99er Magazine (Nov/Dec 1981). The photograph shows a box identical to that of the stand alone 32K or disk controller, with a cable coming out of the right side where the "pass through" expansion bus is found on other stand alone peripherals. The article describes the video controller as allowing "computer controlled interactive video with VCR's and Video Disk Players", whatever that means. As evidenced by the

videos we created from the formal preentations at the 1990 Lima MUG Conference, it is possible without this device to mix human voice, computer audio and video outout, and video camera footage on the same video tape. Such mixing of various audio and video sources was done by us manually however, not under computer control. An extra cost cable ($99.95 for each of the three available cables in the June - December 1982 TI price list) is needed to interface the video controller to a Sony or Panasonic VCR or a Pioneer video disk player. I really don't understand the need for computer control of a video disk player. If I remember correctly, 1980 video disks resembled phonograph records in that you could only PLAY them from the beginning, not record onto them.

--P CODE PERIPHERAL: My June - December 1982 TI price list states that this device is "available only until replaced by peripheral card", with such a card "available in second quarter 1982." The freight train P Code peripheral is apparently exactly equivalent to the PE Box P Code card.

There you have it folks, the original TI Home Computer expansion "system". Now you know why the expansion port on the 99/4 and 99/4A is on the SIDE of the console, rather than on the back where it should have been placed. You can only use three of these freight train peripherals at once, and they take up huge amounts of desktop space. Arn't you glad we now have the PE Box!

I want to acknowledge the generous gift of Mr. E.T. Breer of the St. Louis Missouri User Group who gave me several of the freight train peripherals described in this review. ■

A MATTER OF MEMORY

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

The following is a refresher dealing with one of the basic components of the computer, its memory. Memory is used to store programs, which are instructions and useful information needed to get a particular job done. The ammount of memory your computer has is measured in bytes. Each letter or number occupies on byte of memory. It is impractical to measure computer memory in plain bytes, since there are thousands required. For that reason, we measure it in one-thousand, or one K, blocks. 1K is equal to 1024 bytes, so a 32K computer would have 32768 bytes of RAM.

The TI uses three main types of memory. These are RAM, ROM, and GROM. ROM means Read Only Memory. As its name implies, it can only be read from. No information can be written to this memory (such as a program) and stored there. The factory places instructions in this memory, where it remains forever. GROM is similar to ROM. Graphics Read Only Memory is more or less a TI invention. There are vague differences between the two, but not enough for me to go into detail.

RAM means Random Access Memory. You can write to it and read to it as often as necessary while using the computer. It is easily changed and thus needed for programming.

There are two types of RAM. One is called "static". It is very stable, and more expensive. Once an instruction is written to it, it will remain in memory until changed or power is shut off.

The TI uses dynamic RAM, as do most other computers. The computer must constantly re-read then re-write what's in the memory. This means the machine has an extra step and runs slower as a consequence. ■

MAGIC CRAYON - MULTICOLOR

```
**********************************
* MAGIC CRAYON
* av John Clulow
* 99'ER 82-06
* MULTI COLOR MODE
* Rita med joystick och
* använd följande tangenter
* C = Change Color
* S = Save Screen
* R = Recall Screen
* E = Erase Screen
* T = Terminate
* Se även PB 92-4.29 om Multicolor
**********************************

       DEF  CRAYON
       REF  VSBW,VMBW,VMBR,VSBR
       REF  VWTR,KSCAN,DSRLNK

*****************
* Definiera LABEL
*****************
SCREEN BSS  >300
PALET  BSS  >600
PATRN  BSS  >600
ROW    BSS  1
COL    BSS  1
       EVEN
CURSOR DATA >8040,>2010,>0804,>0000
       DATA >0000,>0408,>1020,>4080
       DATA >0102,>0408,>1020,>0000
       DATA >0000,>2010,>0804,>0201
ARROW  DATA >0102,>0408,>0000,>0000
       DATA >0000,>0000,>0000,>0000
       DATA >0080,>4020,>0000,>0000
       DATA >0000,>0000,>0000,>0000
ATTRIB DATA >5878,>800F,>D000
ARRATT DATA >6578,>8401
PDATA  DATA >0600,>1000,>0000,>0600
       DATA >000F
       TEXT 'DSK1.CRAYON/SCR'
       EVEN
ZERO   DATA >0000
D32    DATA >0020
D8     DATA >0008
GRAY   DATA >EEEE
MAX    DATA >05FF
COLMAX DATA >0100
LOAD   BYTE >05
BLACK  BYTE >11
ONE    BYTE >01
TWO    BYTE >02
FCOLOR BYTE >10
BCOLOR BYTE >0E
H18    BYTE >12
H14    BYTE >0E
H11    BYTE >0B
H07    BYTE >07
H06    BYTE >06
H05    BYTE >05
H02    BYTE >02
NOKEY  BYTE >FF
       EVEN
PAB    EQU  >0F80
USRWS  EQU  >20BA
PNTR   EQU  >8356
UNIT   EQU  >8374
FIRE   EQU  >8375
JOYSTY EQU  >8376
JOYSTX EQU  >8377
SPRITE EQU  >837A
STATUS EQU  >837C
GPLWS  EQU  >83E0

***************************
* Definiera sprite PATTERN
* för tecken 128 och 132
***************************
CRAYON LWPI USRWS
       LI   R0,>400
       LI   R1,CURSOR
       LI   R2,64
       BLWP @VMBW

*********************************
* Sätt förgrund och bakgrund grå
* i text1 mode så att det som
* finns på skärmen blir osynligt
*********************************
       LI   R0,>01F0
       BLWP @VWTR
       LI   R0,>07EE
       BLWP @VWTR

************************
* Initiera skärmtabellen
* för Multi Color mode
************************
       LI   R0,SCREEN
       LI   R1,6
       CLR  R2
LOOP0  LI   R3,4
LOOP1  LI   R4,>20
       MOVB R2,R5
LOOP2  MOVB R5,*R0+
       AI   R5,>0100
       DEC  R4
       JNE  LOOP2
       DEC  R3
       JNE  LOOP1
       AI   R2,>2000
       DEC  R1
       JNE  LOOP0
       LI   R0,>00
```

```
       LI   R1,SCREEN
       LI   R2,>300
       BLWP @VMBW

********************************
* Initiera skärmen med färgpalett
********************************
       LI   R0,>100
       LI   R1,PALET

LOOP3  MOV  @GRAY,*R1+
       DEC  R0
       JNE  LOOP3
       CLR  R0
       LI   R3,16
LOOP4  LI   R4,2
LOOP5  MOVB @GRAY,*R1+
       MOVB @GRAY,*R1+
       MOVB @BLACK,*R1+
       LI   R5,4
LOOP6  MOVB R0,*R1+
       DEC  R5
       JNE  LOOP6
       MOVB @BLACK,*R1+
       DEC  R4
       JNE  LOOP5
       SWPB R0
       AI   R0,>11
       SWPB R0
       DEC  R3
       JNE  LOOP4
       LI   R0,>300
LOOP7  MOVB @GRAY,*R1+
       DEC  R0
       JNE  LOOP7

*************************
* Initiera teckentabellen
* - transparent
*************************
CLEAR  LI   R0,>300
       LI   R1,PATRN
LOOP8  MOV  @ZERO,*R1+
       DEC  R0
       JNE  LOOP8

***********************
* Ladda teckentabellen
***********************
       LI   R0,>800
       LI   R1,PATRN
       LI   R2,>600
       BLWP @VMBW

*******************************
* Välj Multi Color mode och
* och Sprite med dubbel storlek
*******************************
       LI   R0,>01EA
       BLWP @VWTR
       SWPB R0
       MOVB R0,@>83D4

********************************
* Definiera attribut för sprite #0
********************************
       LI   R0,>300
       LI   R1,ATTRIB
       LI   R2,6
       BLWP @VMBW

********************************
* Definiera antalet aktiva sprites
********************************
       MOVB @ONE,@SPRITE

*****************************
* Initiera markörens färg och
* räknaren för ändring av färg
*****************************
       LI   R3,>0F01
       CLR  R4

*****************************
*****************************
** START FÖR HUVUD-LOOPEN **
*****************************
*****************************

********************************
* Undersök joystick för rörelse,
* FIRE och  tangenter
********************************
CHECK  LIMI 2
       LIMI 0
       LI   R0,1
       BL   @CHECKS
       MOVB @ONE,@UNIT
       BLWP @KSCAN
       CB   @FIRE,@H05
       JEQ  CLEAR
       CB   @FIRE,@H02
       JNE  NEXT1
       B    @SAVE
NEXT1  CB   @FIRE,@H06
       JNE  NEXT2
       B    @RECALL
NEXT2  CB   @FIRE,@H11
       JNE  NEXT3
       LIMI 2
       LWPI GPLWS
       BLWP @0000
NEXT3  CB   @FIRE,@H14
       JNE  NEXT4
       B    @SELECT
NEXT4  CB   @FIRE,@H18
       JNE  SKIP

***********************
* Rutin för att placera
* block på skärmen
***********************
DRAW   LI   R0,>300
       LI   R1,ROW
       LI   R2,2
```

```
        BLWP @VMBR
        CLR  R7
        CLR  R8
        CLR  R2
        MOVB @ROW,R8
        SWPB R8
        AI   R8,9
        C    R8,@COLMAX
        JLT  NOCORR
        S    @COLMAX,R8
NOCORR  DIV  @D32,R7
        SLA  R7,8
        A    R7,R2
        SRL  R8,2
        A    R8,R2
        CLR  R7
        CLR  R8
        MOVB @COL,R8
        SWPB R8
        AI   R8,8
        C    R8,@COLMAX
        JLT  NOCORC
        S    @COLMAX,R8
NOCORC  DIV  @D8,R7
        SLA  R7,3
        A    R7,R2
        MOV  R2,R2
        JLT  SKIP
        C    R2,@MAX
        JGT  SKIP
        LI   R0,>14
        BL   @CHECKS
        CLR  R1
        MOVB @FCOLOR,R1
        SWPB R1
        CLR  R0
        MOVB @PATRN(2),R0
        SRL  R8,2
        JEQ  MARK1
        SRL  R1,4
        SWPB R0
        SRL  R0,4
        SLA  R0,4
        JMP  MARK2
MARK1   SLA  R0,4
        SRL  R0,4
        SWPB R0
MARK2   A    R1,R0
        SWPB R0
        MOVB R0,@PATRN(2)
        LI   R0,>0800
        LI   R1,PATRN
        LI   R2,>600
        BLWP @VMBW
SKIP    CLR  R5
        CLR  R6
        MOVB @JOYSTY,R5
        NEG  R5
        SLA  R5,2
        MOVB @JOYSTX,R6
        SLA  R6,2
        SWPB R6
        MOVB R5,R6
        LI   R1,USRWS+12
        LI   R0,>0780
        LI   R2,2
        BLWP @VMBW
        B    @CHECK

*********************************
* Slutet av programmets huvud-loop*
*********************************

************************
* Rutin för att välja färg
************************
SELECT  LI   R0,>07EE
        BLWP @VWTR
        LI   R0,>800
        LI   R1,PALET
        LI   R2,>600
        BLWP @VMBW
        LI   R0,>300
        LI   R1,ARRATT
        LI   R2,4
        BLWP @VMBW
        BL   @DEBNC
LOOP9   LIMI 2
        LIMI 0
        MOVB @ONE,@UNIT
        BLWP @KSCAN
        CB   @FIRE,@H18
        JEQ  CMARK
        CB   @FIRE,@H14
        JEQ  CSCRN
        CLR  R6
        MOVB @JOYSTX,R6
        SLA  R6,2
        SWPB R6
        LI   R1,USRWS+12
        LI   R0,>0780
        LI   R2,2
        BLWP @VMBW
        JMP  LOOP9
CSCRN   BL   @DOTCOL
        SWPB R1
        MOVB R1,@BCOLOR
        JMP  BACK
CMARK   BL   @DOTCOL
        SLA  R1,12
        MOVB R1,@FCOLOR
BACK    BL   @DEBNC
        CLR  R0
        MOVB @BCOLOR,R0
        SWPB R0
        MOVB @H07,R0
        BLWP @VWTR
        LI   R0,>800
        LI   R1,PATRN
        LI   R2,>600
        BLWP @VMBW
        LI   R0,>300
        LI   R1,ATTRIB
        LI   R2,4
```

```
       BLWP @VMBW
       B    @SKIP

*********************************
* DSR-rutin för att spara skärmens
* teckentabell på disk
*********************************
SAVE   LI   R0,>1000
       LI   R1,PATRN
       LI   R2,>600
       BLWP @VMBW
       LI   R0,PAB
       LI   R1,PDATA
       LI   R2,25
       BLWP @VMBW
       LI   R6,PAB+9
       MOV  R6,@PNTR
       BLWP @DSRLNK
       DATA 8
       B    @CHECK

*********************************
* DSR-rutin för att ladda skärmens
* teckentabell från disk
*********************************
RECALL LI   R0,PAB
       LI   R1,PDATA
       LI   R2,25
       BLWP @VMBW
       LI   R0,PAB
       MOVB @LOAD,R1
       BLWP @VSBW
       LI   R6,PAB+9
       MOV  R6,@PNTR
       BLWP @DSRLNK
       DATA 8
       LI   R0,>1000
       LI   R1,PATRN
       LI   R2,>600
       BLWP @VMBR
       LI   R0,>0800
       LI   R1,PATRN
       LI   R2,>600
       BLWP @VMBW
       B    @CHECK

*****************************
* Subrutin för att periodiskt
* ändra sprite-färgen
*****************************
CHECKS AI   R4,>100
       JEQ  CHANGE
       DEC  R0
       JNE  CHECKS
       JMP  RETURN
CHANGE SWPB R3
       MOV  R3,R1
       LI   R0,>303
       BLWP @VSBW
RETURN RT

*********************
* Debounce subroutine
*********************
DEBNC  MOVB @ONE,@UNIT
       BLWP @KSCAN
       CB   @FIRE,@NOKEY
       JNE  DEBNC
       RT

***************************
* Subrutin för att bestämma
* pilen färg
***************************
DOTCOL CLR  R1
       LI   R0,>301
       BLWP @VSBR
       SWPB R1
       AI   R1,>07
       SRL  R1,4
       RT

***********************
* Slutet med auto-start
***********************
AUTO   END  CRAYON
```

DIGISYNT - SOUND SAMPLER

(C) 1994 Stefano Bonomi, Italy

Digisynt it' s a program that allow you to digitize and syntetize every kind of sound. It make use of the TI 99/4A standard cassette TI 99 input port in order to read the sound you play with any connected sound players device. Then it make an analogical to digital conversion so that you can then play the sound or save it on disk. At this point it's also possible to reproduce what you saved using TI EXTENDED BASIC or TI BASIC (if you have the editor assembler cartridge inserted in your consolle).

For use DIGISYNTH you need a TI 99/4A consolle with 32KB memory expansion, a disk drive and EXTENDED BASIC or EDITOR ASSEMBLER cartridge. For digitize sound you need also a cassette recorder and the standard cable that you received with your consolle in order to connect the EAR

socket to the TI 99. This connection should be done simply like when you want to load a program using a cassette player.

THIS PROGRAM IS FAIRWARE:
STEFANO BONOMI
VIA SACCHI 21
37124 VERONA
ITALY

DIGISYNT: FIL=18 LED=53 ANV=305

filnamn	sekt	typ	längd p
BASYNT	9	DIS/FIX	80
BLAST	3	DIS/VAR	80
CAT	16	DIS/VAR	80
COW	11	DIS/VAR	80
DISCO	63	DIS/VAR	80
DOG	16	DIS/VAR	80
DS	15	PROGRAM	3330
EXSYNT	11	DIS/FIX	80
FROG	16	DIS/VAR	80
GAOV	9	DIS/VAR	80
HORSE	17	DIS/VAR	80
KAPOW	17	DIS/VAR	80
LOAD	2	PROGRAM	110
LOADER	10	DIS/FIX	80
NAME	10	DIS/VAR	80
README	44	DIS/VAR	80
RECORD	18	DIS/VAR	80
SUBMARINE	18	PROGRAM	4214

Sänd en skiva och svarskuvert till redaktören för att få en kopia. ■

PROGRAMS WRITE PROGRAMS -5

by Jim Peterson, Tigercub, USA

In addition to writing programs in MERGE format, the same techniques can be used to analyze or modify programs which have been SAVEd in MERGE format. The D/V 163 file editor in Part 2 of this series was an example.

Here is a simple program to remove REM statements -

```
100 DISPLAY AT(3,5)ERASE ALL
:"REM REMOVER": : :"Program
must be SAVEd in":"MERGE for
mat by":"SAVE DSK(filename),
MERGE"
110 DISPLAY AT(12,1):"FILENA
ME? DSK" :: ACCEPT AT(12,14)
:F$ :: DISPLAY AT(14,1):"NEW
 FILENAME? DSK" :: ACCEPT AT
(14,18):NF$
120 OPEN #1:"DSK"&F$,VARIABL
E 163,INPUT :: OPEN #2:"DSK"
&NF$,VARIABLE 163,OUTPUT
130 LINPUT #1:M$ :: A=POS(M$
,CHR$(131),1):: B=POS(M$,CHR
$(154),1):: A=MAX(A,B):: IF
A=3 THEN 150 :: IF A=0 THEN
PRINT #2:M$ :: GOTO 150
140 PRINT #2:SEG$(M$,1,A-1)&
CHR$(0)
150 IF EOF(1)<>1 THEN 130 ::
 CLOSE #1 :: PRINT #2:CHR$(2
55)&CHR$(255):: CLOSE #2
```

The REM statement will begin with either a !, which is CHR$(131), or REM which is CHR$(154). So, line 130 reads in the lines one at a time. A finds the position in the line of ! and B finds the position of REM; one or the other, or both, will not be present and will equal 0. Then MAX finds the larger of A and B, which will be whichever one is present, or 0 if neither.

If ! or REM is in the 3rd position, immediately after the 2-byte line number, we want to delete the line entirely, so we do not reprint it. If A=0 then neither ! nor REM is present, so we reprint the entire line in the new file.

Otherwise, the REM statement is obviously a tail remark, so we reprint to the new file the segment of it starting with the first character and consisting of the number of characters one less than the position of the ! or REM. And, since we have lopped off the end of the line, we do not forget to replace the end-of-line marker CHR$(0).

If we have not reached the end of the file, we go back for the next line. Otherwise, we close the old file, but we remember to add the end-of-file marker to the new file before we close that too. ■

NY ADRESS: Texaments, 701 S.Wicklow, Ste. 506, Stillwater, OK 74074, USA. Telefon (405) 372-0819. ■

ATT GÖRA EN TIDNING
PROGRAM FÖR FORMATERING

av Jan Alexandersson

Jag kommer här att beskriva hur tidningen Programbiten kommer till. Följande olika saker måste göras:

- Samla in artiklar från olika källor: medlemmar, kontakter utomlands, program från äldre tidningar, nyheter från Micropendium.

- Planera innehåll så att det blir en lämplig blandning mellan olika ämnen. Förutom den närmaste utgåvan planeras innehåll översiktligt i två nummer framåt i tiden.

- Redigera innehåll i varje textfil, kontrollera att fakta stämmer, kontrolläsning hittar även stavfel.

- Formatera texten till 36 teckens bredd, rättstavning (med Spellcheck) av svenska artiklar. För engelska artiklar kontrolleras stavning endast om den skrivits av icke engelsktalande författare.

- Avstavning av texten.

- Lista Basic-program med Super Extended Basic med kommandot LIST "DSK2.PROGRAM":28: för att få 28 teckens bredd som det ser ut på skärmen när du knappar in Extended Basic program. Använd sedan programmet 28-80 för att omvandla från DIS/VAR 28 till DIS/VAR 80.

- Dela upp artikeln i spalter med 62 raders längd (sidan med rubrik får spalt med 57 rader). Rubriken sparas separat och och Delete av dess rader. Texten sparas sedan med Save File 1 57 DSK2.TEXT-1 följt av Delete av rad 1-57 o.s.v.

- Tvåspaltig text ordnas med programmet MERGE som omvandlar TEXT-1 och TEXT-2 till en ny fil TEXT-12.

- En komplett sida ordnas genom att lägga till kontrollkoder, rubrik och sidfot. Denna sida kan ses i sin helhet med Funnelweb 80-kolumnseditor. Små korrigeringar kan göras här för t.ex. kontrollkoder i vänstra spalten som kräver ytterligare blanktecken i högra spalten.

- Utskrift av den färdiga sidan görs med RAG-Formatter. Samtliga sidor i Programbiten som har högst 80 teckens bredd skrivs ut på detta sätt.

- Framsidans text inklusive den höga texten "PROGRAM BITEN 94-2" skrivs ut med RAG-Formatter. Sidan backas sedan manuellt i skrivaren och en fil med kontrollkoder som sätter vänstermarginal och startpunkt i höjdled skrivs med Print File. Bilderna skrivs sedan ut en och en med Alexander Hulpkes Smartcopy. Efter varje bild görs fyra manuella Line Feed på skrivaren.

- Sidor som innehåller mer än 80 tecken per rad skrivs ut med programmet COLUMNS.

PROGRAM SOM ANVÄNDS

```
100 ! 28-80 CONVERTER XB
110 ! D/V 28 to D/V 80
120 ! by Jan Alexandersson
130 ! Change the file name i
n the two OPEN statements
140 OPEN #1:"DSK1.PROGRAM",V
ARIABLE 28,INPUT
150 OPEN #2:"DSK2.PROGRAM/L"
,OUTPUT
160 LINPUT #1:A$
170 PRINT #2:A$
180 IF EOF(1)=0 THEN 160
190 CLOSE #1 :: CLOSE #2
```

```
100 ! MERGE TWO COLUMNS XB
110 ! Delete all CR (CHR 13)
120 ! put CHR 20 in front of
 period at start of line
130 ! Delete line which star
ts with CHR >127
140 ! Select 10/12 CHR/INCH
150 ! Jan Alexandersson
160 ! Sweden, 1993-12-31
170 ! Change the file name i
n the three OPEN statements
180 OPEN #1:"DSK1.TEXT-1",IN
PUT
190 OPEN #2:"DSK1.TEXT-2",IN
PUT
200 OPEN #3:"DSK2.TEXT-12",O
UTPUT
210 INPUT "10 OR 12 CHR/INCH
 ":A
220 IF A<>10 AND A<>12 THEN
```

```
210
230 IF EOF(2)=0 THEN LINPUT
#2:B$ ELSE B$=""
240 IF ASC(B$)>127 THEN B$="
"
250 IF EOF(1)=0 THEN LINPUT
#1:A$ ELSE A$=""
260 IF ASC(A$)>127 THEN A$="
"
270 IF ASC(A$)=46 THEN A$=CH
R$(20)&A$ !Period with CH 20
280 CALL CHARFILTER(A$):: CA
LL CHARFILTER(B$)
290 IF A=12 THEN PRINT #3:SE
G$(A$&RPT$(" ",80),1,41)&B$
300 IF A=10 THEN PRINT #3:SE
G$(A$&RPT$(" ",80),1,36)&B$
310 IF EOF(2)=0 THEN 230
320 IF EOF(1)=0 THEN 230
330 CLOSE #1 :: CLOSE #2 ::
CLOSE #3
340 SUB CHARFILTER(TEXT$)
350 ! Deletes CR = CHR 13
360 CH$=CHR$(13)
370 CH=32
380 IF POS(TEXT$,CH$,1)=0 TH
EN 410
390 TEXT$=SEG$(TEXT$,1,POS(T
EXT$,CH$,1)-1)&CHR$(CH)&SEG$
(TEXT$,POS(TEXT$,CH$,1)+1,80
)
400 GOTO 380
410 SUBEND
```

```
100 ! PRINT THREE COLUMNS XB
110 ! Jan Alexandersson
120 ! Sweden, 1993-12-31
130 ! Change the three OPEN
DSK file names
140 OPEN #1:"DSK2.TEXT-1",IN
PUT
150 OPEN #2:"DSK2.TEXT-2",IN
PUT
160 OPEN #3:"DSK2.TEXT-3",IN
PUT
170 OPEN #4:"PIO",VARIABLE 1
50
180 PRINT #4:CHR$(28);"@" !
Reset the printer
190 PRINT #4: : :CHR$(27);"g
";! print mode 15 chr/inch
200 PRINT #4:CHR$(27);"D";CH
R$(14);CHR$(47);CHR$(80);CHR
$(0)! Set the printer TABs
210 LINPUT #1:A1$ :: CALL CH
ARFILTER(A1$)
220 LINPUT #2:A2$ :: CALL CH
ARFILTER(A2$)
230 LINPUT #3:A3$ :: CALL CH
ARFILTER(A3$)
240 PRINT #4:CHR$(9);A1$;CHR
$(9);A2$;CHR$(9);A3$ ! CHR$(
9) will Select the next TAB
250 IF EOF(1)=0 THEN 210
260 CLOSE #1 :: CLOSE #2 ::
CLOSE #3 :: CLOSE #4
9000 SUB CHARFILTER(TEXT$)
9010 ! Deletes CR = CHR 13
9020 CH$=CHR$(13)
9030 CH=32
9040 IF POS(TEXT$,CH$,1)=0 T
HEN 9070
9050 TEXT$=SEG$(TEXT$,1,POS(
TEXT$,CH$,1)-1)&CHR$(CH)&SEG
$(TEXT$,POS(TEXT$,CH$,1)+1,8
0)
9060 GOTO 9040
9070 SUBEND
```

TIPS FROM THE TIGERCUB #70

by Jim Peterson, USA
(died January 12, 1994)

****

I still like to program "brain games". Here is one of the most devilish of all.

```
100 DISPLAY AT(2,3)ERASE ALL
:"THE FORE AND AFT PUZZLE":"
":" Try to get the numbers i
n  the lower half and the
   letters in the upper half
."
110 DISPLAY AT(8,1):" You ca
n move horizontally  or vert
ically to the vacant square
or jump over one     space t
o the vacant square,"
120 DISPLAY AT(12,1):"but nu
mbers can only move   right
and down, letters can only m
ove left and up!" !programme
d by Jim Peterson
130 DISPLAY AT(16,1):" Type
the number or letter  to mov
e or FCTN 8 to start  over o
r FCTN 7 for a demo."
140 DISPLAY AT(20,1):" It ca
n be done in 46 moves but pr
obably not in more    than 4
6 because you will getstuck.
"
150 DISPLAY AT(24,8):"PRESS
ANY KEY" :: DISPLAY AT(24,8)
:"press any key" :: CALL KEY
(0,K,S):: IF S=0 THEN 150
160 CALL CLEAR :: CALL COLOR
(0,16,16,3,16,5,4,16,5,5,16,
7,6,16,7,9,2,2,12,16,16):: C
ALL SCREEN(2)
170 A$=RPT$("~",9):: GOSUB 3
30 :: GOSUB 340 :: V$="12345
678ABCDEFGH"&CHR$(1)&CHR$(6)
180 CALL CALLKEY(24,1,V$,C$)
:: V=ASC(C$)<65 :: IF C$=CHR
$(6)THEN GOSUB 330 :: GOSUB
340 :: GOTO 180
190 IF C$<>CHR$(1)THEN GOSUB
 220 :: GOTO 180
200 GOSUB 330 :: GOSUB 340 :
: FOR W=1 TO 46 :: C$=SEG$("
A87AC63CFG87FBE635F21ABEH87D
354BED21CDG54GH21H",W,1):: V
=ASC(C$)<65 :: GOSUB 220 ::
NEXT W
210 FOR D=1 TO 500 :: NEXT D
 :: GOSUB 330 :: GOSUB 340 :
: GOTO 180
220 FOR J=3 TO 7 :: P=POS(M$
(J),C$,1):: IF P=0 THEN 230
ELSE X=J :: J=7 :: GOTO 240
230 NEXT J
240 IF V=-1 THEN 260 :: T=X-
1 :: GOSUB 290 :: IF F=1 THE
N F=0 :: RETURN ELSE T=X-2 :
: GOSUB 290 :: IF F=1 THEN F
=0 :: RETURN
250 T=P-1 :: GOSUB 310 :: IF
 F=1 THEN F=0 :: RETURN ELSE
 T=P-2 :: GOSUB 310 :: IF F=
1 THEN F=0 :: RETURN ELSE 28
0
260 T=X+1 :: GOSUB 290 :: IF
 F=1 THEN F=0 :: RETURN ELSE
 T=X+2 :: GOSUB 290 :: IF F=
1 THEN F=0 :: RETURN
270 T=P+1 :: GOSUB 310 :: IF
 F=1 THEN F=0 :: RETURN ELSE
 T=P+2 :: GOSUB 310 :: IF F=
1 THEN F=0 :: RETURN
280 CALL SOUND(500,110,0,-4,
0):: RETURN
290 IF SEG$(M$(T),P,1)<>"c"
THEN RETURN
300 M$(T)=SEG$(M$(T),1,P-1)&
C$&SEG$(M$(T),P+1,255):: M$(
X)=SEG$(M$(X),1,P-1)&"c"&SEG
$(M$(X),P+1,255):: GOSUB 340
 :: F=1 :: RETURN
310 IF SEG$(M$(X),T,1)<>"c"
THEN RETURN
320 M$(X)=SEG$(M$(X),1,T-1)&
C$&SEG$(M$(X),T+1,255):: M$(
X)=SEG$(M$(X),1,P-1)&"c"&SEG
$(M$(X),P+1,255):: GOSUB 340
 :: F=1 :: RETURN
330 M$(1),M$(2),M$(8),M$(9)=
A$ :: M$(3)="~~123~~~~" :: M
$(4)="~~456~~~~" :: M$(5)="~
```

```
~78cAB~~" :: M$(6)="~~~~CDE~
~" :: M$(7)="~~~~FGH~~" :: R
ETURN
340 FOR J=8 TO 16 :: DISPLAY
 AT(J,10):M$(J-7):: NEXT J :
: RETURN
350 SUB CALLKEY(R,C,V$,K$)
360 CALL HCHAR(R,C+2,30):: F
OR T=1 TO 3 :: CALL KEY(0,K,
S):: IF S<>0 THEN 390
370 NEXT T :: CALL HCHAR(R,C
+2,20):: FOR T=1 TO 3 :: CAL
L KEY(0,K,S):: IF S<>0 THEN
390
380 NEXT T :: GOTO 360
390 IF POS(V$,CHR$(K),1)=0 T
HEN 360 ELSE K$=CHR$(K)
400 SUBEND
```

I don't think this is very useful, but somebody asked me for it - it converts decimals to fractions.

```
100 CALL CLEAR :: CALL CHAR(
95,"000000FF")
110 DISPLAY AT(12,1):"Decima
l?" :: ACCEPT AT(12,10):D ::
 T=1
120 IF INT(D)<>D THEN D=D*10
 :: T=T*10 :: DISPLAY AT(14,
1):D :: DISPLAY AT(16,1):T :
: GOTO 120
130 DISPLAY AT(14,1):D :: DI
SPLAY AT(15,2):RPT$("_",LEN(
STR$(T))):: DISPLAY AT(16,1)
:T
140 FOR J=2 TO 5 STEP 3
150 IF D/J=INT(D/J)AND T/J=I
NT(T/J)THEN D=D/J :: T=T/J :
: DISPLAY AT(14,1):D :: DISP
LAY AT(16,1):T :: GOTO 150
160 NEXT J :: GOTO 110
```

Several years ago, John Hamilton wrote a program you could use to key in a program with TI-Writer, then merge it in, delete the "!" after each line number, and run it as a program. Its only problem was with lines of over 80 characters. Since then, better programs have been written - XLATE and TEXTLOADER - which do not require deleting anything but they still have some trouble with long lines and with missing spaces. This little version overcomes those faults but you do have to delete the "!".

Try keying in a program into the Funlweb Editor, be sure to put a carriage return at the end of each program line. When finished, check each program line which has wrapped around to two lines. If the first character in that second line should be preceded by a space, insert a space as its first character. Then save the file with the PF option and run this little program. Enter NEW, merge in the output file by MERGE DSKn.filename, go through it with FCTN X and FCTN 1 deleting the "!" after each line number, and it should run as a program.

```
100 DISPLAY AT(12,1)ERASE AL
L:"Input file? DSK":"":"Outp
ut file? DSK"
110 ACCEPT AT(12,16):A$ :: A
CCEPT AT(14,17):B$
120 OPEN #1:"DSK"&A$,INPUT :
: OPEN #2:"DSK"&B$,VARIABLE
163,OUTPUT
130 LINPUT #1:M$
140 IF POS(M$,CHR$(13),1)=0
THEN LINPUT #1:M2$ :: M$=M$&
M2$ :: GOTO 140 ELSE M$=SEG$
(M$,1,LEN(M$)-1)
150 X=POS(M$," ",1):: Y=VAL(
SEG$(M$,1,X-1))
160 PRINT #2:CHR$(INT(Y/256)
)&CHR$(Y-256*INT(Y/256))&"!"
&SEG$(M$,X+1,255)&CHR$(0)
170 IF EOF(1)<>1 THEN 130 EL
SE CLOSE #1 :: PRINT #2:CHR$
(255)&CHR$(255):: CLOSE #2
```

I had a question from a friend who wanted to key in some pieces of information in Funnelweb and then sort them. Trouble was, the data tended to be more than 80 characters long. Therefore it was saved as two or more separate records, which a sort scrambled into garbage.

So, how do you create and sort long records of varying length? The easiest way is to let the disk drive controller do it for you. Just type whatever you want, as long as you want, then save it as a separate file, using the first several letters of the text as the filename. Don't include any spaces or periods, of course. If you are using numbers as filenames, pad them with leading zeros to all the same length such as 001 to 999 or 0001 to 1000.

The drive controller will sort those files alphabetically, and this little program will print them in that sequence -

```
100 CALL CLEAR :: DIM F$(127
):: OPEN #1:"DSK1.",INPUT ,R
ELATIVE,INTERNAL :: INPUT #1
:D$,A,B,C
110 INPUT #1:M$,A,B,C :: IF
A=2 AND C=80 THEN X=X+1 :: F
$(X)=M$
120 IF LEN(M$)<>0 THEN 110 E
LSE CLOSE #1 :: OPEN #2:"PIO
"
130 FOR J=1 TO X :: OPEN #1:
"DSK1."&F$(J),INPUT
140 LINPUT #1:M$ :: IF ASC(M
$)<127 THEN PRINT #2:M$
150 IF EOF(1)<>1 THEN 140 EL
SE CLOSE #1
160 NEXT J :: STOP
```

This method is limited by the fact that you can only put 127 files on a disk, but if you have more than one drive you can have 127 on each one, and use this program -

```
100 DISPLAY AT(12,1)ERASE AL
L:"How many drives?" :: ACCE
PT AT(12,18)SIZE(1)VALIDATE(
NUMERIC):D :: DIM F$(510)
110 FOR J=1 TO D :: OPEN #1:
"DSK"&STR$(J)&".",INPUT ,REL
ATIVE,INTERNAL :: INPUT #1:D
$,A,B,C
120 INPUT #1:M$,A,B,C :: IF
A=2 AND C=80 THEN X=X+1 :: F
$(X)=M$&"*"&STR$(J)
130 IF LEN(M$)<>0 THEN 120
140 CLOSE #1 :: NEXT J :: CA
LL LONGSHELL(X,F$()):: OPEN
#2:"PIO"
150 FOR J=1 TO X :: W=POS(F$
(J),"*",1)
160 OPEN #1:"DSK"&SEG$(F$(J)
,W+1,1)&"."&SEG$(F$(J),1,W-1
)
```

```
170 LINPUT #1:M$ :: IF ASC(M
$)<127 THEN PRINT #2:M$
180 IF EOF(1)<>1 THEN 170
190 PRINT #2:"" :: CLOSE #1
:: NEXT J
200 SUB LONGSHELL(N,N$())
210 D=N
220 D=INT(D/3)+1 :: FOR I=1
TO N-D :: IF N$(I)<=N$(I+D)T
HEN 250 :: T$=N$(I+D):: J=I
230 N$(J+D)=N$(J):: J=J-D ::
 IF J<1 THEN 240 :: IF T$<N$
(J)THEN 230
240 N$(J+D)=T$
250 NEXT I
260 IF D>1 THEN 220
270 SUBEND
```

A recent article in a news letter reminded me of something I knew long ago but had forgotten. If you have been entering a lot of data into a disk file and the program crashes, all is not lost. Just enter CLOSE #1 in command mode and your data will be saved. If you get a FILE ERROR message, just try CLOSE #2 and so on until you hit the right one.

Many user group newsletter editors use a program that puts a code on the address label to indicate when membership expires. Trouble is, no one ever reads their address label!

This quick & dirty little program requires you to prepare your address file in TI-Writer or Funnelweb with name on first line, address on second, city and state on third, the fourth line blank or you can use it for additioal address, number of expiration month on fifth line and year on sixth. Continue with other addresses, making sure you use six lines for each. Such a file is easy to update with TI-Writer. The program will read addresses from that file and print an address label for everyone whose membership has not expired. It will also optionally print a warning label, which you can slap conspicuously on the front page of the newsletter, if the subscription currently expires or expires next month. If you give a grace period for renewal, you can choose to print an address label and a warning label for those who are one month or two months overdue.

```
100 DISPLAY AT(1,4)ERASE ALL
:"REMINDER LABEL PRINTER"
110 DISPLAY AT(3,1):"Address
 file? DSK" :: ACCEPT AT(3,1
8):F$ :: OPEN #1:"DSK"&F$,IN
PUT
120 DISPLAY AT(5,1):"Printer
? PIO" :: ACCEPT AT(5,10)SIZ
E(-20):P$ :: OPEN #2:P$
130 DISPLAY AT(6,1):"Emphasi
zed print? (Y/N)" :: ACCEPT
AT(6,25)VALIDATE("YN")SIZE(1
):E$ :: IF E$="Y" THEN PRINT
 #2:CHR$(27)&"E";
140 DISPLAY AT(7,1):"Doubles
truck print? (Y/N)" :: ACCEP
T AT(7,27)VALIDATE("YN")SIZE
(1):D$ :: IF D$="Y" THEN PRI
NT #2:CHR$(27)&"G";
150 DISPLAY AT(9,1):"Print p
ending expiration notice?
 (Y/N)" :: ACCEPT AT(10,15)S
IZE(1)VALIDATE("YN"):PEND$
160 DISPLAY AT(11,1):"Print
current expiration notice
? (Y/N)" :: ACCEPT AT(12,15)
SIZE(1)VALIDATE("YN"):CUR$
170 DISPLAY AT(13,1):"Print
past expiration notice
? (Y/N)" :: ACCEPT AT(14,15)
SIZE(1)VALIDATE("YN"):PAST$
180 DISPLAY AT(15,1):"Print
two months past expira
tion notice? (Y/N)" :: ACCEP
T AT(16,26)SIZE(1)VALIDATE("
YN"):PAST2$
190 DISPLAY AT(18,1):"Curren
t year?" :: ACCEPT AT(18,15)
:Y :: Y=Y+(Y>99)*1900 :: Y=Y
-92
200 DISPLAY AT(20,1):"Number
 of month?" :: ACCEPT AT(20,
18)VALIDATE(DIGIT):M :: X=M+
Y*12
210 IF EOF(1)=1 THEN 330 ::
LINPUT #1:A$ :: IF ASC(A$)=1
28 THEN 330
220 LINPUT #1:B$ :: LINPUT #
1:C$ :: LINPUT #1:D$ :: INPU
T #1:M,Y :: Y=Y+(Y>99)*1900
:: Y=Y-92 :: M=M+Y*12
230 IF M>=X THEN GOSUB 280
240 IF M=X AND CUR$="Y" THEN
 GOSUB 290 :: GOTO 210
250 IF M=X+1 AND PEND$="Y" T
HEN GOSUB 300 :: GOTO 210
260 IF M=X-1 AND PAST$="Y" T
HEN GOSUB 280 :: GOSUB 310 :
: GOTO 210
270 IF M=X-2 AND PAST2$="Y"
THEN GOSUB 280 :: GOSUB 320
:: GOTO 210 ELSE GOTO 210
280 PRINT #2:A$:B$:C$:D$:"":
"" :: RETURN
290 PRINT #2:A$:"YOUR SUBSCR
IPTION EXPIRES THIS":"MONTH.
 PLEASE RENEW NOW SO YOU":"W
ILL NOT MISS ANY ISSUES":" "
:" " :: RETURN
300 PRINT #2:A$:"YOUR SUBSCR
IPTION EXPIRES NEXT":"MONTH.
 PLEASE RENEW NOW SO YOU":"W
ILL NOT MISS ANY ISSUES":" "
:" " :: RETURN
310 PRINT #2:A$:"YOUR SUBSCR
IPTION EXPIRED LAST":"MONTH.
 PLEASE RENEW NOW SO YOU":"W
ILL NOT MISS ANY ISSUES":" "
:" " :: RETURN
320 PRINT #2:A$:"YOUR SUBSCR
IPTION EXPIRED":"TWO MONTHS
AGO":"THIS WILL BE YOUR LAST
 ISSUE":"UNLESS YOU RENEW PR
OMPTLY":" " :: RETURN
330 CLOSE #1 :: END
```

Memory just about full -

Jim Peterson ■

WORM OF BEMER SPEL I BASIC

av Stephen D.Fultz, USA

Du ska leda ormen genom 11 olika rum (spelnivåer). Varje rum har olika utseende med ökad svårighet för varje rum. Du måste först äta fem svampar i rummet innan du kan fortsätta till nästa rum. Du får inte stöta in i väggen eller i ormens svans. Du har fyra liv från start och får ytterligare ett liv i vart tredje rum. Använd ESDX-tangenterna för att styra ormen.

```
3 REM WORM OF BEMER
4 REM COMPUTE 84-04
5 DIM NN(29),RANK$(12)
```

```
6 GOSUB 11000
10 GOTO 5000
20 FOR I=1 TO LEN(H$)
30 CALL HCHAR(ROW,COL+I,ASC(
SEG$(H$,I,1)))
35 NEXT I
40 RETURN
100 CALL KEY(3,K,ST)
105 IF (K<>68)+(OD=2)THEN 11
0
106 DX=1
107 DY=0
108 DI=1
110 IF (K<>83)+(OD=1)THEN 11
5
111 DX=-1
112 DY=0
113 DI=2
115 IF (K<>69)+(OD=3)THEN 12
0
116 DY=-1
117 DX=0
118 DI=4
120 IF (K<>88)+(OD=4)THEN 14
0
125 DY=1
130 DX=0
135 DI=3
140 CALL HCHAR(YA,XA,136)
145 OD=DI
150 XA=XA+DX
152 YA=YA+DY
154 L=LEN(XA$)
156 XA$=XA$&CHR$(XA)
158 YA$=YA$&CHR$(YA)
160 CALL GCHAR(YA,XA,Z)
162 IF Z<>32 THEN 200
164 CALL HCHAR(YA,XA,128)
166 CALL SOUND(1,622,2)
168 IF L<WO THEN 100
170 CALL HCHAR(ASC(YA$),ASC(
XA$),32)
172 LL=LEN(XA$)-1
174 XA$=SEG$(XA$,2,LL)
176 YA$=SEG$(YA$,2,LL)
180 GOTO 100
200 CALL SOUND(100,311,2)
201 CALL HCHAR(YA,XA,128)
203 GOSUB 6600
205 IF Z<>MUSH THEN 260
210 WO=WO+15+2*LO
212 IF WO<185 THEN 215
214 WO=185
215 RANDOMIZE
216 XX=RND*28+3
218 X=RND*19+4
220 CALL GCHAR(X,XX,H1)
222 IF H1<>32 THEN 216
224 SC=SC+100+LO*7
228 HI=HI-1
230 GOSUB 6600
232 IF HI>0 THEN 245
234 CALL HCHAR(3,17,104)
236 CALL HCHAR(13,2,104)
238 CALL HCHAR(13,31,104)
240 CALL HCHAR(23,17,104)
241 FOR I=1 TO 30 STEP 3
242 CALL SOUND(100,1900,I)
243 NEXT I
244 GOTO 100
245 CALL HCHAR(X,XX,MUSH)
250 GOTO 100
260 IF Z=104 THEN 270
261 IF LI=1 THEN 7500
264 GOSUB 7500
266 GOTO 290
270 CALL HCHAR(YA,XA,136)
272 GOSUB 7000
275 FOR DE=110 TO 880 STEP 3
2
277 PRINT
279 CALL SOUND(1,DE,2)
280 CALL SOUND(-1,DE,2)
281 NEXT DE
282 LO=LO+1
283 IF LO=12 THEN 1200
284 WO=5
285 L1=L1+1
286 IF LO>EX THEN 9100
287 CALL COLOR(14,L1,1)
288 CALL CLEAR
289 GOSUB 1300
290 GOSUB 6600
300 ON LO GOTO 5080,400,500,
550,600,700,800,450,550,1000
,1100,1200
399 GOTO 5080
400 REM 2ND SCREEN
410 CALL HCHAR(13,5,120,24)
420 GOTO 5080
449 REM SCREEN
450 CALL VCHAR(7,15,120,16)
455 CALL HCHAR(9,6,120,22)
460 GOTO 5080
499 REM 4TH SCREEN
500 CALL HCHAR(6,5,120,24)
505 CALL HCHAR(20,5,120,24)
510 GOTO 5080
549 REM 5TH SCREEN
550 CALL HCHAR(7,6,120,22)
555 CALL VCHAR(8,5,120,16)
560 GOTO 5080
599 REM FRAME 6
600 CALL HCHAR(12,3,120,13)
610 CALL HCHAR(12,19,120,12)
620 GOTO 5080
699 REM FRAME 7
700 FOR I=8 TO 18
710 CALL HCHAR(I,7,120,7)
715 CALL HCHAR(I,18,120,8)
720 NEXT I
725 GOTO 5080
799 REM FRAME 8
800 CALL HCHAR(8,3,120,13)
805 CALL HCHAR(14,12,120,19)
810 CALL HCHAR(18,3,120,13)
815 GOTO 5080
999 REM FRAME 9
1000 GOSUB 1400
1015 FOR T=5 TO 21
1020 CALL HCHAR(T,4,32,16)
1025 NEXT T
1030 GOTO 5080
1100 GOSUB 1400
1110 FOR T=5 TO 21
1115 CALL HCHAR(T,4,32,20)
1120 NEXT T
1125 GOTO 400
1199 REM YOU WIN
1200 CALL CLEAR
1205 CALL SCREEN(5)
1206 FOR I=4 TO 8
1207 CALL COLOR(I,2,1)
1208 NEXT I
1210 PRINT TAB(9);"NERM'S HO
ME!": : :TAB(10);"THANK YOU!
": : : : : : : : : :
1275 FOR T=1 TO 3
1280 FOR I=110 TO 880 STEP 3
0
1283 CALL SOUND(1,I,2)
1284 CALL SOUND(-1,I,2)
1285 NEXT I
1286 FOR I=880 TO 110 STEP -
30
1287 CALL SOUND(1,I,2)
1288 CALL SOUND(-1,I,2)
1289 NEXT I
1290 NEXT T
1291 CALL SCREEN(2)
1293 GOTO 7700
1300 CALL CLEAR
1310 PRINT "SCORE";TAB(20);"
ROOM":"MUSHROOMS";TAB(20);"L
IVES": : : : : : : : : : : :
 : : : : : : : : : :
1350 RETURN
1400 FOR T=5 TO 21
1410 CALL HCHAR(T,4,120,26)
1420 NEXT T
1430 RETURN
4999 REM UP THE GAME
5000 GOSUB 10000
5005 MUSH=112
5014 LI=4
5015 SC=0
5020 LO=1
5035 HI=5
5040 WO=5
5045 EX=2
5050 L1=3
5055 GOSUB 5500
5060 CALL CLEAR
5065 CALL SCREEN(2)
5066 FOR I=3 TO 8
5067 CALL COLOR(I,16,1)
```

```
5068 NEXT I
5070 GOSUB 1300
5075 GOSUB 6600
5080 XA$=""
5081 YA$=""
5085 XA=17
5086 YA=18
5091 DX=0
5093 DY=-1
5103 IF HI<6 THEN 5107
5105 HI=5
5107 IF HI>-1 THEN 5110
5109 HI=0
5110 DI=4
5115 FOR I=2 TO 31 STEP 29
5120 CALL VCHAR(3,I,120,21)
5125 NEXT I
5130 FOR I=3 TO 23 STEP 20
5135 CALL HCHAR(I,3,120,28)
5140 NEXT I
5145 CALL HCHAR(24,3,137,28)
5150 IF HI>0 THEN 5174
5155 CALL HCHAR(3,17,104)
5160 CALL HCHAR(12,2,104)
5165 CALL HCHAR(12,31,104)
5167 CALL HCHAR(23,17,104)
5171 GOTO 150
5174 RANDOMIZE
5175 XX=RND*28+3
5178 X=RND*19+4
5180 CALL GCHAR(X,XX,H1)
5185 IF H1<>32 THEN 5174
5190 CALL HCHAR(X,XX,MUSH)
5200 GOTO 150
5500 CALL CLEAR
5505 PRINT TAB(10);"GET READ
Y!": : : : : : : : : : : : :
5525 FOR I=1 TO 14
5530 CALL SOUND(100,NN(I),2)
5535 NEXT I
5540 RETURN
6599 REM PRINT SCORE
6600 H$=STR$(SC)
6603 ROW=1
6604 COL=12
6605 GOSUB 20
6607 H$=STR$(LO)
6608 COL=27
6609 GOSUB 20
6610 H$=STR$(HI)
6611 ROW=2
6620 COL=12
6625 GOSUB 20
6630 H$=STR$(LI)
6635 COL=27
6640 GOSUB 20
6650 RETURN
6999 REM NERM LEAVES
7000 SP=SP-5
7005 GOSUB 6600
7010 HI=5
7015 L=LEN(XA$)
7020 FOR I=1 TO L
7025 CALL SOUND(2,110+I*2,2)
7030 CALL HCHAR(ASC(YA$),ASC
(XA$),32)
7035 LL=LEN(XA$)-1
7040 XA$=SEG$(XA$,2,LL)
7045 YA$=SEG$(YA$,2,LL)
7050 NEXT I
7060 RETURN
7499 REM OOP
7500 CALL CLEAR
7505 PRINT TAB(13);"OOPS": :
 : : : : : : : : : :
7525 LI=LI-1
7547 FOR I=14 TO 24
7549 CALL SOUND(10,I*40,2)
7551 NEXT I
7553 FOR I=1 TO 30
7555 NEXT I
7560 IF LI<1 THEN 7700
7575 GOSUB 1300
7600 RETURN
7699 REM END OF GAME
7700 CALL CLEAR
7704 FOR I=3 TO 8
7705 CALL COLOR(I,16,1)
7706 NEXT I
7710 IF HS>SC THEN 7750
7720 HS=SC
7725 PRINT : : : : :TAB(8);"
NEW HIGH SCORE"
7728 FOR T=110 TO 1760 STEP
50
7729 CALL SOUND(2,T,2)
7730 NEXT T
7750 PRINT : : : : :TAB(7);"
YOUR SCORE: ";SC: :TAB(7);"H
IGH SCORE: ";HS: : : :TAB(5)
;"YOUR NEW RANK IS :": :TAB(
9);RANK$(LO)
7796 FOR I=15 TO 29
7797 CALL SOUND(100,NN(I),2)
7798 NEXT I
7810 PRINT : : : :"(C TO CON
TINUE   Q TO QUIT)": : : : :
7820 CALL KEY(3,K,ST)
7830 IF ST=0 THEN 7820
7840 IF (K<>67)*(K<>81)THEN
7820
7845 IF K=67 THEN 5000
7850 STOP
9099 REM EXTRA LIFE
9100 CALL CLEAR
9110 PRINT TAB(11);"BONUS LI
FE": : : : : : : : : : : : :
9132 FOR I=1 TO 30 STEP 2
9134 CALL SOUND(100,1175,I)
9136 NEXT I
9140 EX=EX+3
9145 LI=LI+1
9150 GOTO 287
10000 CALL CLEAR
10001 FOR T=3 TO 8
10003 CALL COLOR(T,2,1)
10006 NEXT T
10010 CALL COLOR(14,3,1)
10015 CALL SCREEN(15)
10020 PRINT TAB(10);"WELCOME
 TO": : : : :TAB(8);"NERM OF
 BEMER": : : : : : : : : : :
10034 PRINT "USE E,S,D, & X
KEYS TO MOVE": :
10040 CALL HCHAR(21,3,136,4)
10042 CALL HCHAR(21,8,128)
10045 FOR I=1 TO 22
10047 CALL HCHAR(21,6+I,136)
10050 CALL HCHAR(21,7+I,128)
10052 CALL SOUND(10,622,2)
10055 CALL HCHAR(21,2+I,32)
10057 FOR T=1 TO 20
10058 NEXT T
10060 NEXT I
10065 FOR T=1 TO 100
10070 NEXT T
10075 RETURN
10999 REM DEFINE CHARS
11000 FOR I=104 TO 136 STEP
8
11015 READ A$
11020 CALL CHAR(I,A$)
11025 NEXT I
11030 DATA FFFFFFFFFFFFFFFF,
187EFFFF18181818,FF81BDA5A5B
D81FF
11032 DATA 8142243C7E5A3C18,
387CFEFEFEFE7C38
11033 CALL COLOR(10,2,2)
11035 CALL COLOR(11,14,1)
11040 CALL COLOR(12,2,10)
11045 CALL COLOR(13,7,1)
11050 CALL CHAR(137,"FFFFFFF
FFFFFFFFF")
11060 FOR I=1 TO 9
11065 READ RANK$(I)
11070 NEXT I
11075 FOR I=10 TO 12
11080 RANK$(I)="HALL OF FAME
"
11085 NEXT I
11090 DATA ZERO,ROOKIE,NOVIC
E,AVERAGE
11092 DATA MASTER,GRAND MAST
ER,WIZARD,GRAND WIZARD
11094 DATA SUPER STAR
11100 FOR I=1 TO 29
11110 READ NN(I)
11120 NEXT I
11130 DATA 262,349,40000,349
,392,40000,392,440,523,440,5
23,440,349,40000
11135 DATA 349,40000,40000,2
62,247,262,294,294,262,40000
,40000,40000,330,330,349,@
11140 RETURN
```