EAR 99'ERS

East Anglia Region 99'ers User's Group

VOLUME 2 – ISSUE 1 – MAY '88

Great Britain:

Scott Copeland
EAR 99'ers User's Group
13 Elm Walk
Lakenheath, Suffolk
England IP27 9QR
Tel: 063881-3457

United States:

SSgt D. S. Copeland
% EAR 99'ers User's Group
1979CS/LGG
PCS Box 5927
APO NY 09179-5379
Tel: 011-44-63881-3457

President † Scott Copeland † 063881-3457
Vice-President † Robert Wordsworth † 0603-38832
Secretary/Treasurer † Jo Ann Copeland † 063881-3457
Publication's Officer † Bryan Cloud † 0473-464996

Contents:

This newsletter is not copyrighted and may be reprinted in whole or in part. PLEASE give credit where credit is due, and notate the Club and the Author. Any user assumes sole liability for use and please note any information by authors' does not guarantee it to be error-free.

Now, turn the page for an EAR-ful of information.....

MINUTES OF THE MEETING
16 April 1988
by: JoAnn Copeland, Sec/Treas

The meeting was called to order at 2:45 PM by the President, with 14 members present. The minutes of the meeting for the previous month and treasury report for the previous month were read and approved.

The first item on the agenda was the One Year Old Birthday Party for EAR 99'ers. It was approved to have the party and a figure (not to exceed) $100.00 was approved for supplying items for the party.

The next item was approving the Publication's Officer position. It was approved to include this position along with those of President, Vice-President and Secretary/Treasurer. From this point on, exchange newsletters will be given to the Publication's Officer who will Catalog them and issue the catalog for publication in the newsletter. If any member is looking for a particular article or column, they will now have access to it via the Publication's Officer. If all goes as planned, Bryan Cloud will be heading this position.

A motion was brought up by Bryan Cloud and approved to provide for a donation to the Great Ormond Street Hospital in the amount of £10.00. This hospital specifically provides for treatment of children and illnesses associated with children.

Prentice-Hall should have copies of the book several members are looking for. TI-99/4A Home Computer Assembly Language Program (by Ira McComic) can be located at: Prentice-Hall, 66 Wood Lane, Hemel Hempstead, Herts, England, HP2 4RG (tel - 0442~231555 or 0442~212771 or Telex 82445). (ISBN # 0-915381-56-7). Thanks to Eddy Carter for tracking this down, and thanks to Mike Brick and Mark Playle who live close enough to check it out for us. Scott has also sent the company a Telex requesting the quantity available and cost.

The Modem Account is up to $107.22 (÷ exchange rate for British equivalent). DONATIONS are still accepted for this account! It was agreed to raise the account up to $200.00 before a motion was made to buy the Modem for the group. At that time, inquiries will be made to seek a V21/23 Modem at the least expensive cost.

Dis-Tel is available to those Looking for second-hand electrical equipment, or a list of Bulletin Boards. They have also been known to carry disk drives at £15.00 each. Dis-Tel runs on 300/300 baud. Dial 01-679-1888 for more information.

The meeting was adjourned at 3:37 PM.

Demonstrations were given on: Prestel/MicroNet; Legends; and several DisAssembler programs available through the Library. MIKE BRICK was the winner of the Mystery Disk of the Month - we found he already had the item prepared and it was agreed he could pick another choice from the Library.

We said a farewell to MARK and CHRIS ZIEGLER. This was the last meeting they could attend as they will be PCS'ing to the States (North Dakota) the first of May. We wish you's guys luck! Keep in touch! Try getting in touch with the following User Groups in your area and tell them EAR 99'ers recommended you! >>>

MAD HUGgers
509 Reeves Drive
Grand Forks, ND 58205

Minnesota & Dakota Home UG
509 Reeves Drive
Grand Forks, ND 58201

Well, the street address seems to be the same although the Zip (Post) Code is different, but that's what my list of User Groups says! Give it a shot and let us know what you find out! Mark and Chris renewed their subscription with EAR 99'ers (THANKS), however we seem to have lost Joe Quigley as a member due to his PCS move, and Derek Duddy due to his work schedule. We Thank everyone who has decided to stay with our membership! We have a really great TI Family!

PARTY!

BRING THE FAMILY!

Day :	SUNDAY
Date :	22 MAY 1988
Time :	1:00 PM
Place:	13 ELM WALK
Bring:	DESERT Dish and BYOB
RSVP :	BY 16 MAY 1988

The Main Dish ingredients (Bar-B-Que) will be provided for by EAR 99'ers (hamburgers, hot-dogs, chicken, etc.), along with Soft Drinks, Utensils, etc. A Vegetarian meal will be provided. PLEASE provide a Desert, and if required, BYOB (Bring Your Own Bottle). Please RSVP with # of Adults and # of Children that will be attending. RSVP to Scott or JoAnn Copeland (063881-3457). THANKS! (Please note if you decide to BYOB: Your car keys will be gratiously accepted at the front door!)

NEWS AND MORE NEWS

We received a post-card from ASGARD SOFTWARE. It seems that registered owners of Legends can return the original disks with $3.00 to receive the update - Version 1.1 of Legends. Yes, I sent in my disks and check already! I'll never see Scott again, except for the back of his head at the computer, that is. What do you mean, that's an improvement?

We welcome WEST PENN 99'ers CLUB to the Exchange List. JOHN WILLFORTH was kind enough not only to accept our exchange offer, but sent a years worth of back-issues to boot, along with a nice Graphics Program Disk (see library updates).

News on FUNLWEB from TONY McGOVERN: Work on Version 4.1 is underway with no projected release date yet. We may see a new CONFIG file. The FINAL RELEASE at this time for Version 4.0 is dated JANUARY 22 1988 (our Library has December 22 '88). Seems a couple of bugs were fixed by Tony found without notice by users. (He sure must look hard for them! - I haven't found any). You should see an entirely new appearance in detail coding. We look forward to seeing it! Thanks for the update Tony!

We welcome aboard new members: DAVID FORD and DAVID LEIGH. We met David and David at Bloxwich. Surely, we had a grand ol' time (well, it sounds cute if you say it with an Irish accent!).

We've added over 12 new disks to the Library, plus some Jo had in her collection, so be sure to read through the Catalog Listing for the updates!

VOTING RESULTS

The results are in! You can send Sympathy Cards to the following 'winners' now. (In my case, don't applaud - just send money)...

President ~~ Scott Copeland
Vice-President ~~ Robert Wordsworth
Secretary/Treasurer ~~ Jo Ann Copeland
Publication's Officer ~~ Bryan Cloud

Congratulations to the new officers! And a big THANK YOU to those volunteering their services! We all hope we can help each and every member in 1988-1989!

BLOXWICH

Well, GORDON PITT did it again, and it seemed to be a really good showing at Bloxwich! As usual, Jo did so much talking she didn't make the full rounds and can't give too much information on what happened or who was there (maybe I should grab Scott and have him do the review?) - too many groans, better not!

I did have the occasion to enjoy speaking in person with those members who we usually only have the opportunity of speaking with on the telephone. Good to see you EDDY CARTER, ELAINE (Mike and Claire)! DEREK HAYWARD, you really are still alive and kicking! IAN JAMES, good to see you again! BILL MORAN, as good-looking as ever! How's the bumper after the car accident NEIL? Thank goodness it was only a bumper injured, and not yourself! (Some members will go to any length to make Bloxwich!) We missed the members that couldn't make it due to other obligations (work - yick!).

I also got the chance to chat twice with PETER BROOKS (all snazzed up and looking gorgeous). I had a nice time talking with RICHARD BLANDEN and MARTIN ROSS (used merchandise, I beg to differ Martin!).

Gordon had some sandwich and pizza provisions and everyone seemed to enjoy that! Of course, the tea and coffee kept us going! We saw the MYARC GENEVE up and running, the Cortex User Group was there, along with lots of other set-ups and, as I stated before, I was so busy talking I didn't get around to meet everyone or see what was happening. We had a 'table' set up for EAR 99'ers and I pretty much stayed by that. For those who couldn't make it you missed a good time and I hope we have another Bloxwich again soon (but how I doooo hate the trip!). Perhaps one of our members who attended Bloxwich could write an article on what was there? Please??? (Isn't it pitiful to see the Editor beg?)

IN THIS ISSUE

This issue contains flow charts for Funnelweb Version 4.0, gratiously supplied by TI READER, CHARLOTTE TI-99/4A USER GROUP (March 88). As they state: (1) The charts have been in several newsletters and have been reformatted using boxes to help clarify and further organize the contents (by TI Readers' Editor - THANKS!). (2) They were kind enough to reproduce this on white paper for easy photo-copying by other groups (nice!) and (3) why not remove this and keep it for reference near your computer? (In my case because I don't understand any of it!). Our thanks to the work Charlotte UG put into this to make it more concise and organized!

Also included you'll find some Tutorials I had on disk regarding Disk Sector Bit Map and Disk Talk. Hope someone finds it useful! ROBERT WORDSWORTH continues to amaze us with the time he puts into his Mini-Memory articles for the Newsletter, and as usual, you'll see an Adventure Column and For Big & Little Kids column. (I say that because I'm just a big kid - not only in heart but in mind also - no remarks from the peanut gallery, thank you!).

Thanks for the Post Card JOHN - Hope you enjoyed Budapest! (Pretty soon my wall will be covered with post cards from all over the world!) Isn't it nice to be remembered?

Scott bought me (us?) a Compact Disc Player and I've been sitting at the computer typing along to Oohwa-Oohwa Boop Boop Ditty, How about the boy from New York City? (How many of us remember the "Purple People Eater" and "Along the Colonel Jackson down the mighty Mississip? We took a little bacon and we took a little beans...) Or ~ Ooh Eeh Ooh Ah Ah, Ching Chang Walla Walla Bing Bang? Shows you what happens when you're the Editor for too long a time...

Have a good look through the Library Catalog!
KEEP THOSE LIBRARY ORDERS COMING IN!

Well, how about we get into the Newsletter → → → → →
(and escape all this insanity?)

>>> E.N.D. <<<

◆ ◆ DISK TALK ◆ ◆

by: Credit Unknown (Sorry!)

This is an article on TI-DOS and the layout of the TI Disk. To get the most out of these articles, you should have access to a program that allows you to read a disk sector-by-sector. TI also released a similar program called DISKUTILITY.

◆ The Mysterious Sector #0 ◆

By now, quite a few of us have in our possession some type of "disk-fixing" program that allows us to go deep into the caverns of even the most protected disk we own. However, even with this valuable tool, few of us really know what to look for when we read those mysterious sectors >0 to >21.

This article is intended to give a short description of one of the most important sectors on the TI disk: Sector >0 and its Disk Bit Map.

◆ TI DISK CAPACITY ◆

Before jumping into the hard stuff, let's first get a look at the capacity of the TI disk. A Single-Sided, Single-Density TI disk is capable of holding 92160 bytes. This can be broken down as follows:

92160	Bytes per side
360	Sectors per side
40	Tracks per side
9	Sectors per track
256	Bytes per sector

As we all know, a disk initialized with TI's Disk Manager is not capable of utilizing all 360 sectors for programs and files. A number of the first sectors (>0 to >21) are reserved for the operating system. These sectors hold the information that allows TI BASIC Assembly Language to locate and retrieve data from the disk.

FORTH, however, uses a different disk operating system. It is for this reason that you cannot always copy a FORTH disk with the Disk Manager module. Although this fact may seem to be an annoyance every time you want to back up your FORTH disk, it is also a valuable lesson: You don't HAVE to use TI's DOS - you can invent your OWN. That is, you can if you know what you're doing! For now let's find out how the TI DOS works.

◆ PHYSICAL LAYOUT ◆

The TI disk is divided into blocks called Allocatable Units (AUs). On the present TI system one AU is equal to one sector of 256 bytes. The maximum number of AUs per disk is 4096 (for a Double-Sided, Double-Density format). The AUs are numbered starting from 0.

AU #0 contains the Volume Information BLock (VIB). This AU contains vital information on the disk itself including:

- * Disk Name
- * Number of AUs per disk
- * Number of Sectors per track
- * Number of sides
- * Allocation Bit Map

AU #1 contains an alphabetical index of all the files on the disk. It is used to quickly access any file or program requested.

AU #2-359 contain File Descriptor Blocks and Data Blocks. A File Descriptor Block is similar to the VIB except that it refers to a specific file. More on these in another article.

* THE VIB AND THE BIT MAP *

Below is a diagram of AU #0, The Volume Information Block:

```
     +------------------------+
   0 !                        !   1
     =    D I S K   N A M E   =
   8 !                        !   9
     +------------------------+
  10 !   TOTAL NUMBER OF AUs  !  11
     +------------------------+
  12 !#SECT/TRACK !    "D"    !  13
     +------------------------+
  14 !    "S"     !    "K"    !  15
     +------------------------+
  16 ! PROTECTION !#TRACKS/SIDE!  17
     +------------------------+
  18 ! # OF SIDES !  DENSITY  !  19
     +------------------------+
  20 !                        !  21
     = RESERVED FOR FUTURE USE =
  54 !                        !  55
     +------------------------+
  56 !                        !  57
     =   ALLOCATION BIT MAP   =
 254 !                        ! 256
     +------------------------+
```

Byte 0-9 contain the disk name. The name can be any combination of ten ASCII characters except for the space or period (".") or the null character (ASCII 0). If the name is less than 10 characters, spaces are filled to the right.

Byte 10-11 gives the total number of allocation units on the disk.

Byte 12 indicates the number of sectors per track.

Byte 13-15 contain the ASCII characters "DSK". The TI Disk Manager checks to see if these three letters are present. If they are not, the disk is assumed to be uninitialized.

Byte 16 contains the ASCII code for "P" (>50) if the disk is protected. If it is not, this byte contains a space character (>20).

Byte 17 indicates the number of tracks per side.

Byte 18 shows how many sides have been formatted.

Byte 19 indicates the density of the disk.

Byte 20-55 are reserved for future use. In the current version of TI-DOS, they are set to zero.

Byte 56-255 contain the allocation bit map. This 200 byte map can keep track of up to 1600 256-byte records, or around 400K - enough to be able to handle double-sided, double-density formatting. Each bit represents a sector on the disk. If a sector is in use, the bit is set to one. If the sector is not currently in use, the bit is set to zero.

SO NOW WHAT? Now that you have some idea what kind of information is SUPPOSED to be on Sector >0, get out a new disk and start experimenting. Initialize the disk and then go in and take a look around with your "disk-fixing" program.

Inspect each of the locations shown in the diagram. Now add a file or program. Notice how the Bit Map is updated. Set the PROTECT byte (#16) and try to copy the disk. Remove the "DSK" in bytes 13-15. Can you still load the file? Can you copy the disk? The best way to learn is to EXPERIMENT.

◆ THE FILE DESCRIPTOR BLOCK ◆

Each file stored on disk is referenced by a File Descriptor Record (FDR). It is this FDR that tells TI-DOS what sector the file is stored at, whether it is a program or a data file, and whether the file is stored in one block or in several non-contiguous blocks. The FDR's are located on tracks 2-34 (>22) and are entered in the order they are created (not alphabetically!).

TI-DOS uses sector 1 as an alphabetical index of all the file names currently on disk. Sector 1 is the File Descriptor Index Record (FDIR). The index consists of sector numbers. Each number refers to the FDR for that file. When a new file is created, the FDRs are scanned, sorted and then their sector numbers are reprinted onto sector 1 in the NEW alphabetical order. This indexing helps to speed up file access and cut down on wasted disk space.

* WHAT DOES IT LOOK LIKE? *

Below is a diagram of the FDR, for the first program directory located at Sector 2. As you can see, it looks much like the VIB. We will describe each of the sections later in the article.

```
     +++++++++++++++++++++++++++++
   0 +                           +   1
     ~      F I L E  N A M E     ~
   8 +                           +   9
     +---------------------------+
  10 +         RESERVED          +  11
     +---------------------------+
  12 +  FILE TYPE  !RECS PER SECT+  13
     +---------------------------+
  14 +    # OF SECTORS USED      +  15
     +---------------------------+
  16 +  EOF OFFSET ! REC LENGTH  +  17
     +---------------------------+
  18 +    # OF RECORDS USED      +  19
     +---------------------------+
  20 +                           +  21
     ~         RESERVED          ~
  26 +                           +  27
     +---------------------------+
  28 +                           +  29
     ~    DATA BLOCK POINTERS    ~
 254 +                           + 255
     +++++++++++++++++++++++++++++
```

* FDR DESCRIPTION *

Bytes 0-9 contain the filename (up to ten ASCII characters - padded if necessary)

Bytes 10-11 are reserved for future expansion.

Byte 12 contains the file type flag. The bits are set according to the file attributes in TI-BASIC (Internal, Display, Fixed, Variable, etc) and can be interpreted as follows:

BIT	MEANING
0	0=Data file 1=Program file
1	0=DISPLAY format 1=INTERNAL format
2	RESERVED
3	0=Unprotected file 1=Protected file
4-6	RESERVED
7	0=Fixed length recs 1=Variable length recs

For example, if byte 12 contained >07 (b00000101) then you'd know that the file was a protected program file. If byte 12 contained >80 (b10000000) you'd know it was a Display/Variable data file.

Byte 13 contains the number of records per sector. For example, if the file was a DIS/VAR 80 file then byte 13 would contain >03 (3=240). Note that TI-DOS automatically takes care of any "blocking factors" that may be needed. TI-DOS accesses the disk in 256 byte blocks. This means it does not "split" any records between sectors. In other words, any record more then 128 bytes long takes up an entire sector for storage! Keep that in mind next time you plan your data files!

Byte 14-15 contain the number of sectors used by the file.

Byte 16 contains the EOF Offset for the last sector in the file. Since the DOS accesses in 256 byte blocks, this value is used to locate the last byte in the file. This prevents reading past the end of the file. This is only used for variable length data files and for program files.

Byte 17 contains the record length. If the file is 80 bytes long, byte 17 contains >50. If the file is variable in length, this value is the maximum length allowed.

Bytes 18-19 contain the number of records allocated for the file. This is either the number of records presently on file or the number of records the file was initially "opened for" in the TI-BASIC OPEN statement. If the file is VARIABLE-type, this value is the same as the value in bytes 14-15, but in REVERSE ORDER!

Bytes 20-27 are RESERVED and set to 0.

Bytes 28-255 contain the data pointers. When the file must be "broken up" due to its size, a reference to the next record of the file is entered in the pointer area. This tells TI-DOS where on the disk to find the next block of records for this file. Each data chain pointer consists of two THREE byte entries. The first entry contains the sector number of the START of the new data block. The second entry contains the "EOF offset" of THAT block (not necessarily the EOF of the FILE!). To make matters worse, the three bytes are stored in a rather awkward manner. See the diagram following:

```
        Start Sector: ¦S3¦S2¦S1¦

        Block "EOF" : ¦B3¦B2¦B1¦
```

Note that the bytes are stored in "reverse" order or right to left.

Now the two sets of three bytes are stored in a 6 byte segment as follows:

```
                 v  v
        ¦S2¦S1¦B1¦S3¦B3¦B2¦
```

Note the location of bytes S3 and B1!

As each new block is created, a six-byte entry is added to the data chain pointer area. The pointer area can handle up to 76 different blocks for the same file.

♦ LETTING IT SINK IN ♦

Now get out your disk-reading program and start looking at the FDRs. What are the file attributes? Experiment with changing the protection bit, or the record length and see what happens. Can you figure out a way to read a program file like a data file? See if your data files have any data chain pointers. If so, follow them down and find the next starting sector.

After you get comfortable with the information contained in the FDRs you will be ready to do a little file-handling of your own including recovering already deleted files, and restoring "blown directories." For a different approach, read the Disk Format article by Cecil Crowder (in this issue).

>>> E.N.D. <<<

ADVENTUREMANIA INFOCOMITIS

ZORK III: THE DUNGEON MASTER

(C) 1982 INFOCOM, INC.
by: Contradiction-In-Terms

> OBJECTS	> POINTS
Getting Amulet	1
Getting Key	1
Getting Sword	1
Getting to Cliff Ledge	1
Getting Ring	1
Getting Book	1
Stabbing Figure	1

Total Accumulation	7

Things to Collect: Ring, Amulet, Staff, Book, Cloak.

◆ 1◆ When first starting this adventure, take the lamp with you, but you'll have to wait on the sword until later. Don't spend wasted hours trying to get something you can't! Check the LOS (Land of Shadow) later for a surprise.

◆ 2◆ Several things must be done before the earthquake (or else you won't finish!). Discover the Lake and surrounding areas before anything else.

◆ 3◆ The Scenic Vista holds more than it shows. Examine the Indicator and the Table - this holds your clue. Go to Room 8 first then to the Damp Passage. Remember, you'll need light in both rooms! A lamp and torch are provided, and you may have to leave more than one item behind for use later.

◆ 4◆ After the Earthquake another exit is provided for you. Your answer is in the Technology Room. You have experienced Time Travel, haven't you? An appropriate year must be dialed and then press the button. The only thing is, the right machine must be pushed into another room first! (One hint - the year is >after 750.)

◆ 5◆ Carrying items with you on your trip will be useless. Hiding them might work if you find the right spot. Examine the machine and especially the seat. Listen to the guards - you'll know when it's safe. Remember the right year to return to!

◆ 6◆ Flathead Ocean provides another Scenic View. Remember Zork I and "Hello Sailor"? Might help here again! Use the vial you gained (after you go inside the mirror).

◆ 7◆ The Wizened Man in the Damp Passage is frail and weak. Maybe some food might help here? You'll find this at the Cliff, and try taking a trip down while you're at it.

◆ 8◆ Don't fight the Thief for the treasure in the Chest. Take what is offered and use the chest later past the Button Room.

◆ 9◆ You still need a Cloak. Don't fight the Hooded Figure to the Death or you've lost your chance! Look for the prompts saying the Figure is badly hurt. See what happens when you try to remove the hood (but SAVE before the fight as you're likely to be hit when you least expect it!). Random Sequence?

◆10◆ The Royal Puzzle Room displays your patience (or lack of patience!). It will take you ◆53◆ moves to do the maze correctly, and DON'T waste the book where it doesn't belong! You need it yourself. Push the walls and come back to the beginning with the ladder in place for an exit.

◆11◆ If you want to see if anyone is home, why not try knocking on the door? A prompt of >Say "Frotz Ozmoo" might help also.

◆12◆ The Mirror will move if you pressed the button in the Button Room. You did have something blocking the beam, didn't you? Go inside the Mirror Room and examine the Mahogany'and Pine Panels. Lifting the grip might help, too. Push the appropriate panel to indicate South then try pushing the Pine panel. Have your exit yet?

◆13◆ You can't get past the guards like you are, but your vial may help here. Watch the amount of moves you take!

◆14◆ Have the Dungeon Master follow you to the Parapet. From here you'll give him instructions. Enter the Prison Cell and instruct him from there. Try having him turn the Dial and Push the Button. The right sequence gets you where you're going. The #'s 1 and 4 work well (if you know what you're doing!). You may have to go in and out of the cell. If you try to unlock the Bronze Door with the key and it doesn't work, have the Dungeon Master turn the Dial and Push the Button again. Then try opening the Bronze Door again.

If you got this far you have entered the TREASURY OF ZORK! You'll find Chests containing Precious Jewels, Mountains of Zorkmids, Rare Paintings, Ancient Statuary, and Beguiling Curios. Your ending statements reads:

> For a moment you are relieved, safe in the knowledge that you have at last completed your quest in ZORK. You begin to feel the vast powers and lore at your command and thirst for an opportunity to use them. <

Your potential is 7 of a possible 7, in ??? moves.

-- End of Session --

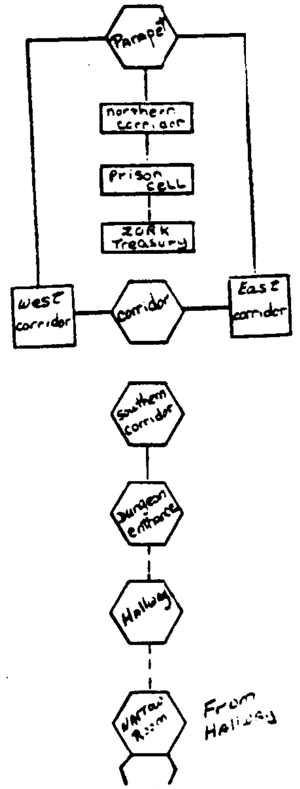
Good Luck (and if you need the Maze Moves let me know!). I couldn't have completed the Maze without my hubby's help so I can't take credit for it (but I do have it scripted out!). HAPPY ADVENTURING!!

>>> E.N.D. <<<

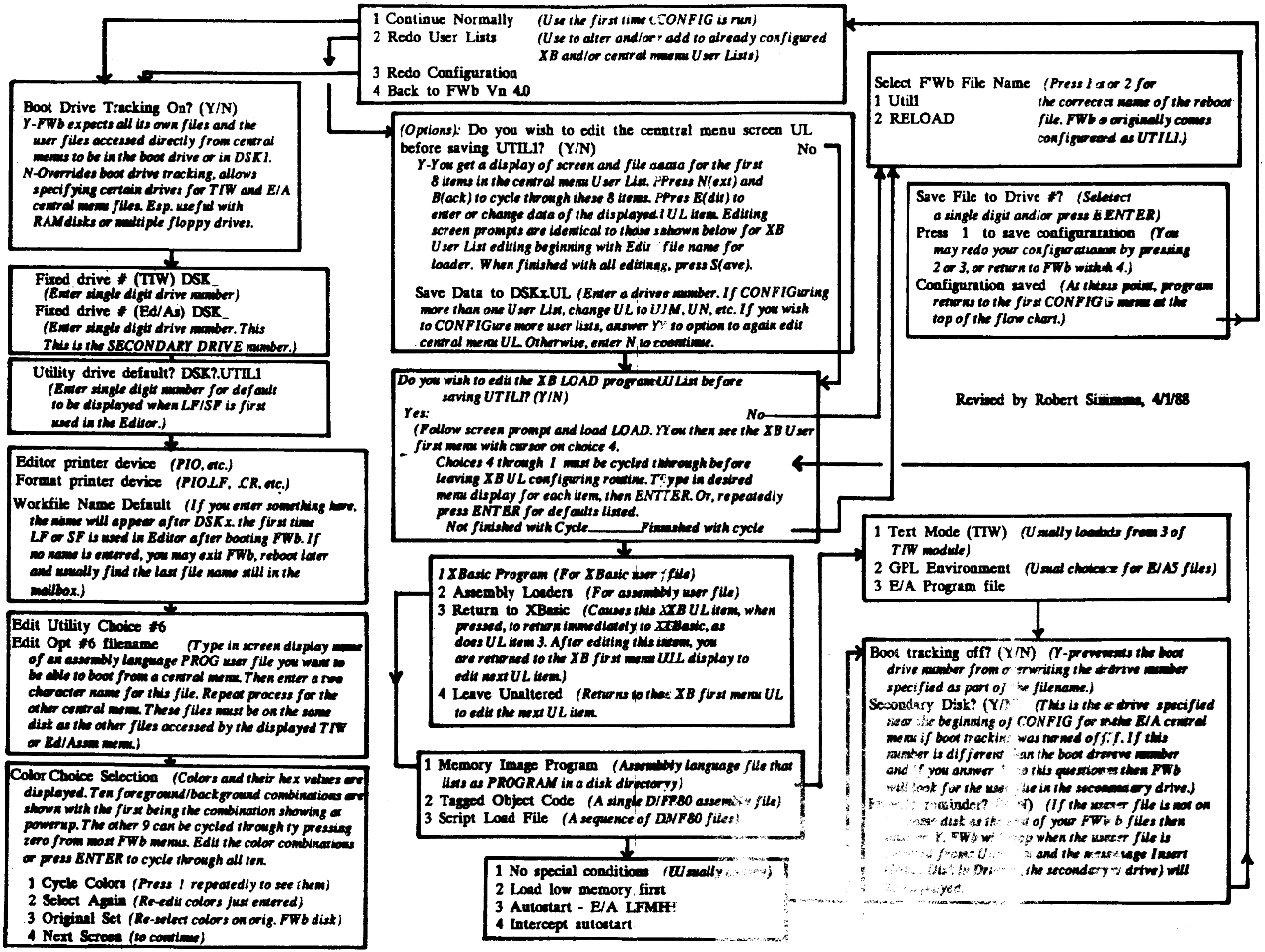
Royal Puzzle

mm	mm	mm	mm	mm	mm	mm	mm	mm	mm	mm
mm	••	SS	??	••	SS	??				
mm	••	••	mm	••	SS ladder	??				
mm	SS	••	••	••	mm	??				
mm	SS ladder	••	••	SS	??	??				
mm	••	••	SS	??	??	??				
mm	mm	mm	??	??	??	??				
mm	??	??	??	mm	mm	??				
mm										
mm										
mm										

mm = Marble Wall
SS = Sandstone Wall
?? = Unknown (blocked passage)
•• = Area you can move



FUNNELWEB 4.0 CONFIGURE PROGRAM FLOW CHART



1 Continue Normally (*Use the first time CONFIG is run*)
2 Redo User Lists (*Use to alter and/or add to already configured XB and/or central menu User Lists*)
3 Redo Configuration
4 Back to FWb Vn 4.0

Boot Drive Tracking On? (Y/N)
Y-FWb expects all its own files and the user files accessed directly from central menus to be in the boot drive or in DSK1.
N-Overrides boot drive tracking, allows specifying certain drives for TIW and E/A central menu files. Esp. useful with RAMdisks or multiple floppy drives.

Fixed drive # (TIW) DSK_
(Enter single digit drive number)
Fixed drive # (Ed/As) DSK_
(Enter single digit drive number. This This is the SECONDARY DRIVE number.)

Utility drive default? DSK?.UTIL1
(Enter single digit number for default to be displayed when LF/SF is first used in the Editor.)

Editor printer device (*PIO, etc.*)
Format printer device (*PIO.LF, .CR, etc.*)
Workfile Name Default (*If you enter something here, the name will appear after DSKx. the first time LF or SF is used in Editor after booting FWb. If no name is entered, you may exit FWb, reboot later and usually find the last file name still in the mailbox.*)

Edit Utility Choice #6
Edit Opt #6 filename (*Type in screen display name of an assembly language PROG user file you want to be able to boot from a central menu. Then enter a two character name for this file. Repeat process for the other central menu. These files must be on the same disk as the other files accessed by the displayed TIW or Ed/Assm menu.*)

Color Choice Selection (*Colors and their hex values are displayed. Ten foreground/background combinations are shown with the first being the combination showing at powerup. The other 9 can be cycled through ty pressing zero from most FWb menus. Edit the color combinations or press ENTER to cycle through all ten.*)
1 Cycle Colors (*Press 1 repeatedly to see them*)
2 Select Again (*Re-edit colors just entered*)
3 Original Set (*Re-select colors on orig. FWb disk*)
4 Next Screen (*to continue*)

(Options): Do you wish to edit the ceentral menu screen UL before saving UTIL1? (Y/N) No
Y-You get a display of screen and file aama for the first 8 items in the central menu User List. PPress N(ext) and B(ack) to cycle through these 8 items. PPres E(dit) to enter or change data of the displayed.l UL item. Editing screen prompts are identical to those sshown below for XB User List editing beginning with Edit file name for loader. When finished with all editinng, press S(ave).

Save Data to DSKx.UL (*Enter a drivee number. If CONFIGuring more than one User List, change UL to UJM, UN, etc. If you wish to CONFIGure more user lists, answer Y to option to again edit central menu UL. Otherwise, enter N to coontinue.*)

Do you wish to edit the XB LOAD programUJ List before saving UTIL1? (Y/N)
Yes: *No*
(Follow screen prompt and load LOAD. YYou then see the XB User first menu with cursor on choice 4.
Choices 4 through 1 must be cycled tthrough before leaving XB UL configuring routine. TType in desired menu display for each item, then ENTTER. Or, repeatedly press ENTER for defaults listed.
Not finished with Cycle________Finnished with cycle

1 XBasic Program (For XBasic user file)
2 Assembly Loaders (*For assembly user file*)
3 Return to XBasic (*Causes this XXB UL item, when pressed, to return immediately to XXBasic, as does UL item 3. After editing this iteem, you are returned to the XB first menu UUL display to edit next UL item.*)
4 Leave Unaltered (*Returns to thee XB first menu UL to edit the next UL item.*)

1 Memory Image Program (*Assembly language file that lists as PROGRAM in a disk directoryy*)
2 Tagged Object Code (*A single D/FF80 assembly file*)
3 Script Load File (*A sequence of DD/F80 files*)

1 No special conditions (*Usually ...*)
2 Load low memory first
3 Autostart - E/A LFMH
4 Intercept autostart

Select FWb File Name (*Press 1 aor 2 for the correcct name of the reboot file. FWb a originally comes configuured as UTIL1.*)
1 Util1
2 RELOAD

Save File to Drive #? (*Selelect a single digit and/or press EENTER*)
Press 1 to save configuraration (*You may redo your configuraation by pressing 2 or 3, or return to FWb withh 4.*)
Configuration saved (*At thiiss point, program returns to the first CONFIGG menu at the top of the flow chart.*)

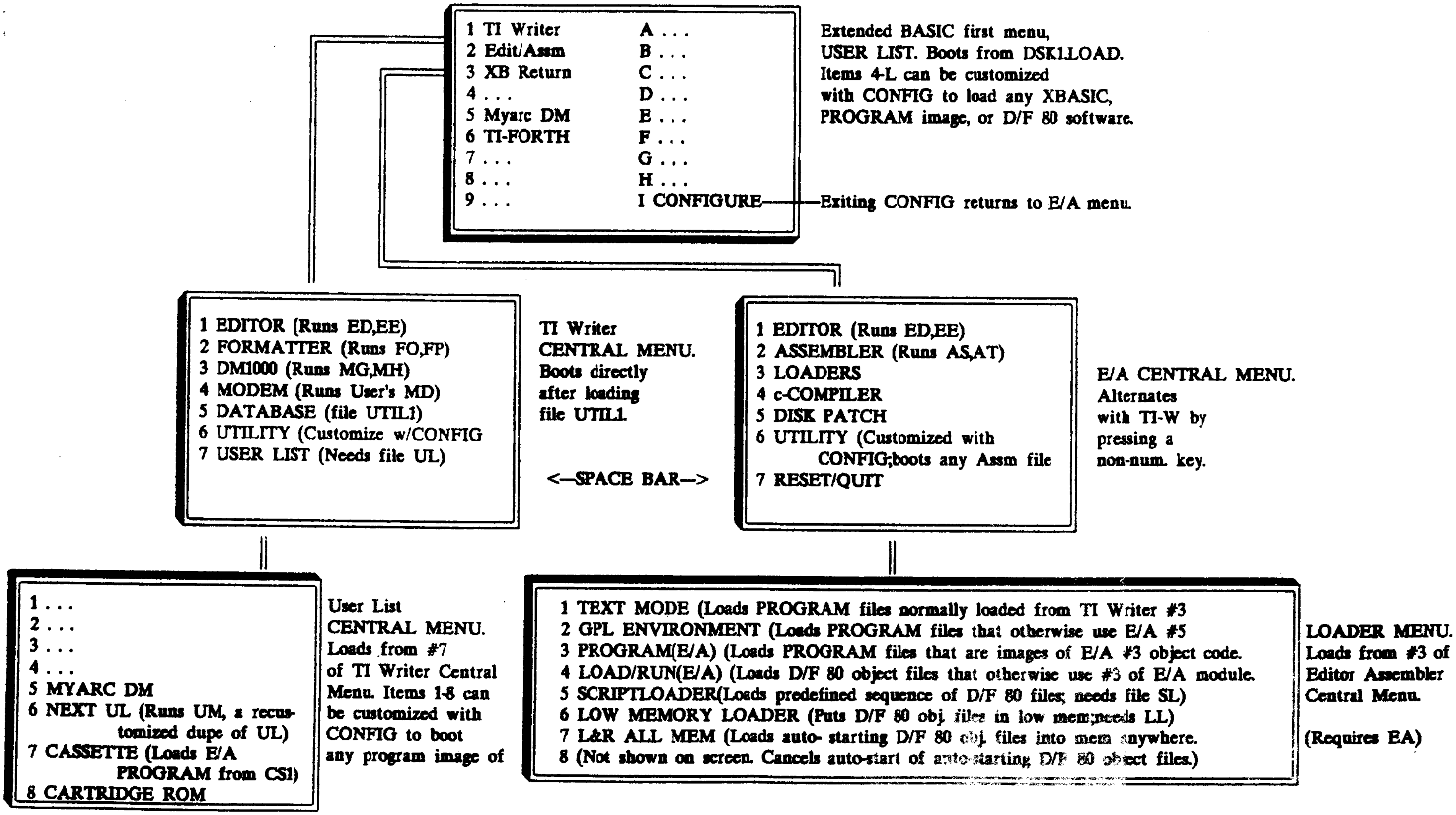
Revised by Robert Simmons, 4/1/88

1 Text Mode (TIW) (*Usually loadds from 3 of TIW module*)
2 GPL Environment (*Usual choice for E/A5 files*)
3 E/A Program file

Boot tracking off? (Y/N) (*Y-preveents the boot drive number from overwriting the ddrive number specified as part of the filename.*)
Secondary Disk? (Y/N) (*This is the drive specified near the beginning of CONFIG for tthe E/A central menu if boot tracking was turned off. If this number is different than the boot drreve number and if you answer Y to this questionn then FWb will look for the user file in the seconnnary drive.*)
[illegible] reminder? (Y/N) (*If the useer file is not on the same disk as the rest of your FWb b files then answer Y. FWb will stop when the useer file is [illegible] from U[illegible] and the messaage Insert [illegible] Disk in Drive (the secondaryy drive) will [illegible]yed.*)

FUNNELWEB 4.0 MENU FLOW CHART

FCTN-7 (AID) from any of these menus gives a disk directory.



1 TI Writer
2 Edit/Assm
3 XB Return
4 . . .
5 Myarc DM
6 TI-FORTH
7 . . .
8 . . .
9 . . .
A . . .
B . . .
C . . .
D . . .
E . . .
F . . .
G . . .
H . . .
I CONFIGURE

Extended BASIC first menu, USER LIST. Boots from DSK1LOAD. Items 4-L can be customized with CONFIG to load any XBASIC, PROGRAM image, or D/F 80 software.

Exiting CONFIG returns to E/A menu.

1 EDITOR (Runs ED,EE)
2 FORMATTER (Runs FO,FP)
3 DM1000 (Runs MG,MH)
4 MODEM (Runs User's MD)
5 DATABASE (file UTIL1)
6 UTILITY (Customize w/CONFIG
7 USER LIST (Needs file UL)

TI Writer CENTRAL MENU. Boots directly after loading file UTIL1.

<—SPACE BAR—>

1 EDITOR (Runs ED,EE)
2 ASSEMBLER (Runs AS,AT)
3 LOADERS
4 c-COMPILER
5 DISK PATCH
6 UTILITY (Customized with CONFIG;boots any Assm file
7 RESET/QUIT

E/A CENTRAL MENU. Alternates with TI-W by pressing a non-num. key.

1 . . .
2 . . .
3 . . .
4 . . .
5 MYARC DM
6 NEXT UL (Runs UM, a recustomized dupe of UL)
7 CASSETTE (Loads E/A PROGRAM from CS1)
8 CARTRIDGE ROM

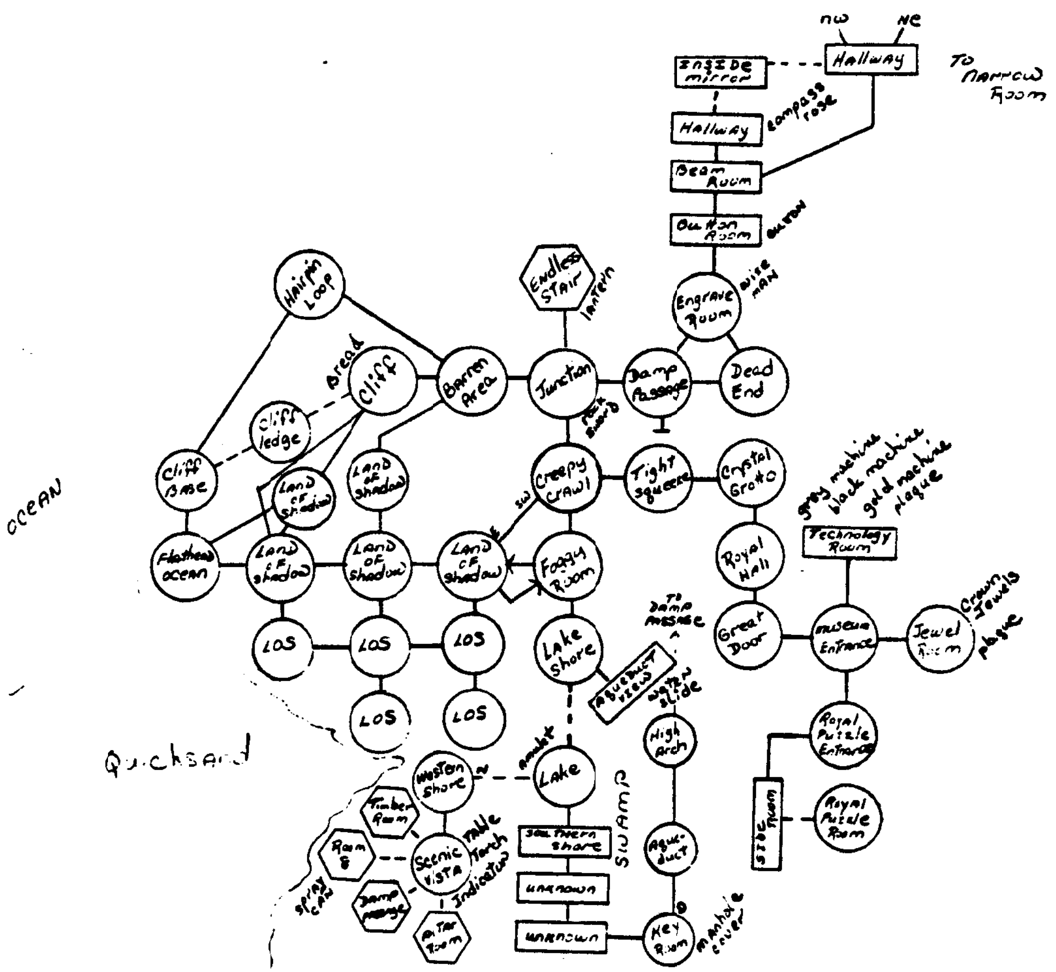
User List CENTRAL MENU. Loads from #7 of TI Writer Central Menu. Items 1-8 can be customized with CONFIG to boot any program image of

1 TEXT MODE (Loads PROGRAM files normally loaded from TI Writer #3
2 GPL ENVIRONMENT (Loads PROGRAM files that otherwise use E/A #5
3 PROGRAM(E/A) (Loads PROGRAM files that are images of E/A #3 object code.
4 LOAD/RUN(E/A) (Loads D/F 80 object files that otherwise use #3 of E/A module.
5 SCRIPTLOADER(Loads predefined sequence of D/F 80 files; needs file SL)
6 LOW MEMORY LOADER (Puts D/F 80 obj. files in low mem;needs LL)
7 L&R ALL MEM (Loads auto- starting D/F 80 obj. files into mem anywhere.
8 (Not shown on screen. Cancels auto-start of auto-starting D/F 80 object files.)

LOADER MENU. Loads from #3 of Editor Assembler Central Menu.

(Requires EA)

Reformatted by Robert Simms from versions printed in the Lima, Ohio Users Group newsletter.


nw
NE
Hallway
To Narrow Room
Inside mirror
Hallway
compass rose
Beam Room
Button Room
button
Engrave Room
wise man
Dead End
Damp Passage
Endless Stair
lantern
Junction
rock sword
Hairpin Loop
Barren Area
Cliff
bread
Cliff ledge
Cliff Base
Land of Shadow
Land of Shadow
Flathead ocean
Land of shadow
Land of Shadow
Land of shadow
LOS
LOS
LOS
LOS
LOS
Creepy Crawl
sw
Tight squeeze
Crystal Grotto
grey machine
black machine
gold machine
plaque
Technology Room
Royal Hall
Foggy Room
Lake Shore
To Damp Passage
Aqueduct view
water slide
Great Door
museum Entrance
Jewel Room
Crown Jewels
plaque
Royal Puzzle Entrance
Side Room
Royal Puzzle Room
High Arch
Aqueduct
Key Room
manhole cover
amulet
Lake
SWAMP
Southern Shore
unknown
unknown
Weston Shore
Timber Room
Room 8
Scenic Vista
table
Torch
Indicator
Damp passage
Altar Room
spray can
Quicksand
OCEAN

MINI-MEMORY
Part VIII

by:
ROBERT WORDSWORTH

We finished last month with a routine which was able to scroll the screen up or down. This month we'll look at scrolling the screen sideways.

In the vertical scrolling routine, we read each screen line from VDP RAM into a thirty-two byte "buffer" in our program and then, by manipulating the contents of register 0, wrote the line back to a different place in VDP RAM. With horizontal scrolling we need a slightly different approach. Each line is read into a buffer in CPU RAM, as before. It is, however, written back to the same place in VDP RAM that it came from, but "shifted" leftwards by one byte, for leftwards scrolling. But we don't actually need to move the line about at all. You'll remember that with the VDP Multiple Byte Write and Read routines, register 1 addresses the buffer in CPU RAM. If we increment the contents of register 1 by one, so that it is pointing one byte further to the right we have, in effect, shifted the line itself one byte to the left. What we have done is "shift" the buffer one byte to the right simply by incrementing its address register rather than actually shifting the contents of the buffer one byte to the left, which would have involved a loop of MOVB instructions. You will often find this technique of manipulating the contents of an address register rather than shifting the addressed data itself in Assembly Language programming. It's one reason why we can write code that is so much more "efficient" and faster to execute than it would be if written in a high-level language such as BASIC.

We also have to take care of the right-most byte of the line. If we want the newly-vacated space at the right-hand end of our "shifted" line to be space filled, then we simply move a space to the appropriate byte in the buffer. If we want a "wrap-round" effect, we move the left-hand byte of the buffer instead.

In the coding below this is done at the instruction whose address is >7E22: MOVB *1+,@BU+32.

The source operand in this instruction is in the "Workspace Register Indirect Auto-increment" addressing mode, one of the addressing modes that can be used with a "general address", as discussed in a previous article. In the other addressing modes mentioned there were Workspace Register Addressing and Workspace Register Indirect Addressing. There are two other addressing modes that can be used with general addresses, Symbolic Memory and Indexed Memory. The destination operand in this instruction is an example of the Symbolic Memory Addressing mode.

A Symbolic Memory address simply specifies the address in CPU RAM of an operand, without involving the use of a register. It is always preceded by an "at" sign (@). The address may be specified by a symbol as in this example. Here the symbol has been modified by a displacement of +32 which the Assembler will add to the value which "BU" represents. The address may also be shown as a hexadecimal or decimal number, again preceded by an "@". We have already used this form of Symbolic Memory address in the BLWP instructions which call utility routines such as VMBRead.

As we are using Workspace Register Indirect auto-increment addressing for the source operand, we can both move the first byte of the buffer to the end and increment register 1 in just one instruction. Although we do not wish to make use of it here, both source and destination operands in Format I (two general address) instructions can be in Symbolic Memory mode. This can cut out a lot of the loading and storing of registers necessary with other CPUs.

Note the way in which we handle the loop counter for the twenty-four lines on the screen. With BASIC's "FOR ... NEXT" construction in mind, we might be tempted to initialize the counter to one, and at the end of the loop add one to the counter then test if it was greater than twenty-four, looping back if not. The instructions for this would be:

```
     LI   9,1
  NR BLWP @>6030
     .
     .  (rest of loop)
     .
     .
     INC  9
     CI   9,24     Compare Immediate instruction:
                   compares register 9 with 24.
     JGT  NR       Loop back if reg 9 not > 24.
```

If we initialize the counter to twenty-four, however, and use the DECrement instruction, we avoid the need for a separate compare instruction, since DEC will reset the "arithmetic greater than bit" of the Status Register to zero if the result is zero. The JGT instruction will only cause a branch to take place if this Status Register bit is set to one.

The saving of one instruction may seem of minor importance, but one of the reasons for writing in Assembly Language is to save space and reduce execution time. There may, of course, be occasions where the "INC with CI" method is more appropriate, ie: where we need to access the loop counter value inside the loop, for example.

Here is the full program, as it will appear on the Line-by-line Assembler screen. Note that a line beginning with "*" is a comment line. If you use "OLD", the location counter should initially be set to >7DE8. If it isn't, start with AORG >7DE8 so that you won't overwrite last month's program.

```
XXXX XXXX     AORG >7DEB
7DEB XXXX BU BSS   34          To hold a screen line (32 characters)
             *                 + an extra byte for shifting
7E0A 02E0     LWPI >70B8       Load Workspace pointer:
7E0C 70B8                      program entry point.
7E0E 0200     LI   0,0         Put address of top Left Hand corner.
7E10 0000                      in Reg 1.
7E12 0201     LI   1,BU        Reg 1 addresses buffer to hold line.
7E14 7DEB
7E16 0202     LI   2,32        Number of bytes to read/write.
7E18 0020
7E1A 0209     LI   9,24        Count of lines.
7E1C 0018
7E1E 0420 NR BLWP @>6030       Read line from screen.
7E20 6030
7E22 D831     MOVB *1+,@BU+32 Move first byte to end of line,
7E24 7E08
             *                  incrementing Reg 1 so that line
             *                 "begins" one byte to the right.
             *
7E26 0420     BLWP @>6028      Write line to screen.
7E28 6028
7E2A 0220     AI    0,32       Point to next line on screen.
7E2C 0020
7E2E 0601     DEC   1          Point to beginning of buffer again.
7E30 0609     DEC   9          Count down number of lines
             *                 to be moved.
7E32 15F5     JGT   NR         If not yet zero, loop back
             *                 to process next line.
             *
7E34 045B     B     *11        Return to calling program.
             *
             *
             *
7E34 XXXX     SYM
RESOLVED REFERENCES
MS-7D1C  NX-7D0C  RT-7D1A  M1-7D3E
TL-7D66  BU-7D86  NL-7DC6  BU-7DE8
NR-7E1E 7E34 XXXX    END
0000 UNRESOLVED REFERENCES
```

Save the whole of MiniMemory RAM to cassette with the Easybug S(ave) command, as usual, saving from 7000 to 7FFF.

Up to now, we have always run our programs using Easybug and there is no reason why we shouldn't now. Remember the entry point is at >7E0C. There are, however, two other ways of running machine code programs from MiniMemory: the "RUN" option from the MiniMemory menu and the TI BASIC "CALL LINK" command.

To be able to use either of these, we first need to choose a name for our program and add the name and the program entry point address to a table. The procedure for doing this is fully documented at the end of the Line-by-line Assembler manual, but here goes anyway!

Appendix D of the MiniMemory manual shows the layout of the module's RAM when it's being used for machine-code programs as opposed to TI BASIC files. Two important one-word areas here are the First Free Address in Medium Memory (FFAM) at >701C and Last Free Address in Medium Memory (LFAM) at >701E. "Medium Memory" refers to the MiniMemory's own 4K bytes of RAM. "Low Memory" and "High Memory" are in the 32K memory expansion, if it's connected. When the MiniMemory is initialized, either by Option 3 of the MiniMemory menu or by the TI BASIC "CALL INIT" command, the FFAM is set to >7118 and the LFAM to >8000. The RAM from the FFAM up to but not including the LFAM is the maximum we can possibly hold in the MiniMemory. When we loaded the MiniMemory from cassette, the values at >701C and >701E were set as appropriate. We must now adjust them to reflect the changes we've been making to the contents of the module's RAM.

Firstly we should update the FFAM at >701C to show the address of the first byte free following the program we've just assembled. This is the initial location counter address when we now run "OLD". This should be >7E36. Set the FFAM by entering:

```
        AORG >701C
        DATA >7E36
```

The location counter should now be >701E, the address of the LFAM field. The table of program names and addresses mentioned earlier is held at the top end of RAM and grows "downwards" as more entries are added. The LFAM field points to the last entry added, which will be the one with the lowest RAM address. When the module was initialized there were no entries in this table, which is referred to as the "REF/DEF" table. Therefore the LFAM field held >8000, which is one byte beyond the end of MiniMemory RAM. There are probably three entries in it at the moment, for "NEW", "OLD" and "LINES". Each entry is eight bytes long: a six-byte program name followed by a one-word entry-point address. As there are three entries, the LFAM should be >8000 - decimal 24, or >8000 - >18 = >7FE8. Whatever its value, subtract eight from the contents of the LFAM field and update it with a DATA statement: DATA >7FD8.

Now we must change the location counter to the new LFAM value and enter a new REF/DEF table entry there. (Incidentally, REF and DEF are two Editor/Assembler directives, not available to us, which would save us all this trouble!) So enter:

```
        TEXT 'SIDESC'
        DATA >7E0A
        END
```

>7E0A is our program entry-point address. The name may be any you like, up to six characters. If you choose a name less than six characters long, you must pad it with spaces in the TEXT directive until it is, for example:

```
             ↓↓↓↓ = 4 spaces
        TEXT 'SI    '
```

We are now ready to run the program. The program we've written isn't suitable for the MiniMemory RUN option, since it does not generate its own screen display, it only scrolls something already there, so we will concentrate on running from TI BASIC.

Press "QUIT" (FCTN/7) and request TI BASIC. The machine code is preserved in RAM even though we pressed QUIT. Enter the following program and save it.

```
        NUM
        100 FOR N=1 TO 32
        110 CALL VCHAR(1,N,64+N,24)
        120 NEXT N
        130 CALL LINK("SIDESC")
        140 GOTO 140
```

The first three lines are purely to get something on the screen. "CALL LINK" is the way in which machine-code routines are called from TI BASIC. The name, enclosed within quotes, is the name that we previously added to the REF/DEF table. We must have added a name and entry-point address to the REF/DEF table before we can call a machine-code routine from BASIC. RUN the program. You should see the whole screen move leftwards by one column.

As with the vertical scrolling program, it is slighty more e::citing if we scroll continuously. To do this, return to the Line-by-Line Assembler "OLD" option. We need to replace the B *11 at >7E34 with a JMP back to the beginning of the loop at address >7E0E. We haven't got a label for this and we don't need one, since the address of the instruction, >7E0E, can be used as the operand of the JMP instruction. So enter:

```
        AORG >7E34
        JMP >7E0E
        END
```

Note the operand of the JMP doesn't need a preceding "@", since the address is stored as a displacement inside the instruction. As before, run your TI BASIC program.

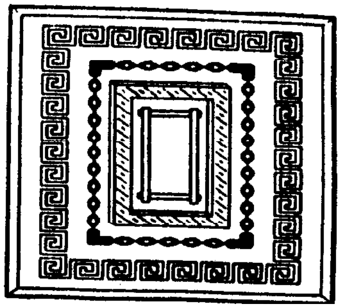
Have fun!

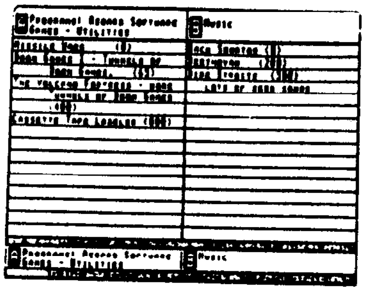
>>> E.N.D. <<<

Artist Borders #1

Putting borders around TI-Artist pictures is difficult at best. You have to be a real-artist to do a professional job. No More! Artist Borders #1 Contains 40 pre-drawn borders ready for use! Quickly and easily create certificates, awards, fancy pictures and more! Requires: TI-Artist or Font Writer II. Disk. $7.95

$7.95 & $.50 S&H




Music
Music

Cassette Labeller

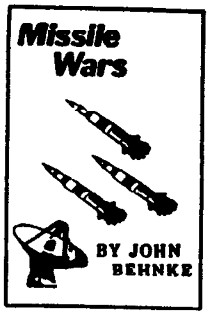
This program isn't a database but it will let you organize your audio and computer cassettes better then you ever dreamed possible! How? It quickly prints an insert for your cassette boxes that details the exact content of your cassette! Easy-to-use and well-documented it's a program you'll use all the time. Requires: Disk, Extended BASIC. $9.95

$9.95 & $.50 S&H

Missile Wars & The Haunted Mine II

Now! Two great arcade games by John Behnke (author of TOD Editor) on one disk! Joystick-bending action will give you hours of game fun. Save the world and explore a haunted mine in the same day. Requires: 32K, Disk, Extended BASIC. $9.95

$9.95 & $.50 S&H

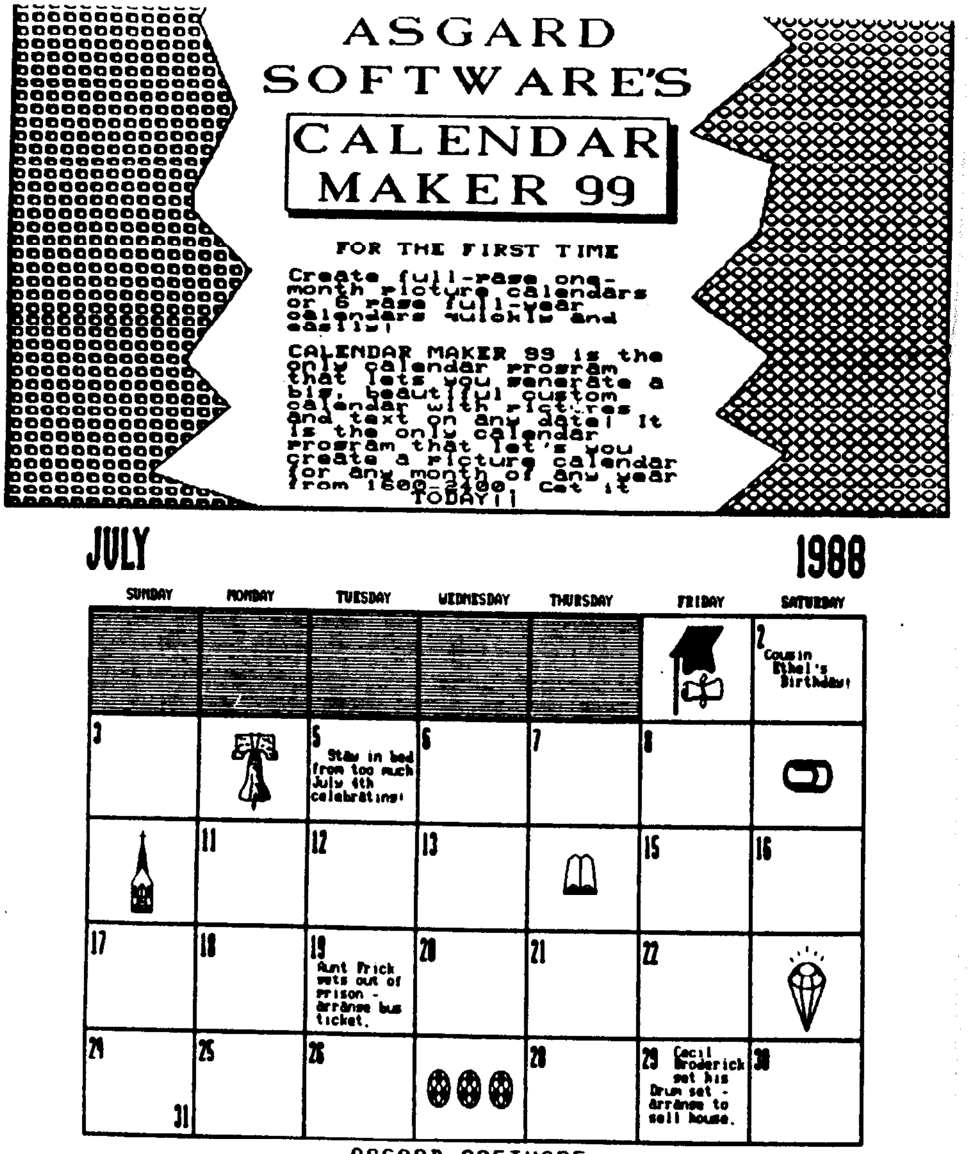


Doom Games I

Cassette users rejoice! This terrific collection 4 great games for the Tunnels of Doom module is now on cassette as well as disk. Only $7.95!

SEND TO;

Asgard Software
P.O. Box 10306
Rockville, MD 20850



ASGARD SOFTWARE'S

CALENDAR MAKER 99

FOR THE FIRST TIME

Create full-page one-month picture calendars or 6 page full-year calendars quickly and easily!

CALENDAR MAKER 99 is the only calendar program that lets you generate a big, beautiful custom calendar with pictures and text on any date! It is the only calendar program that let's you create a picture calendar for any month of any year from 1600-2100. Get it TODAY!!

JULY 1988

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
						2 Cousin Ethel's Birthday!
3		5 Stay in bed from too much July 4th celebrating!	6	7	8	
	11	12	13		15	16
17	18	19 Aunt Prick gets out of prison - arrange bus ticket.	20	21	22	
24 31	25	26		28	29 Cecil Broderick got his Drum set - arrange to sell house.	30

ASGARD SOFTWARE
Box 10306
Rockville, MD 20850

```
          TERMINAL EMULATOR II
              (1) TI BASIC
            >OLD DSK1.COUNT
                 >RUN

          (proofread by jof)
   (available on cassette or disk)
       (instructions in program)

10 REM S. MOORE SHERMAN,TX 79050 4/83
20 RANDOMIZE
30 CALL SCREEN(5)
40 CALL CLEAR
50 FOR CH=100 TO 130
60 READ A$
70 CALL CHAR(CH,A$)
80 NEXT CH
90 FOR CC=1 TO 14
100 CALL COLOR(CC,16,1)
110 NEXT CC
120 OPEN #1:"SPEECH",OUTPUT
130 PRINT #1:"//37"
140 R=1
150 C=2
160 A$="HI. I AM ROCKY ROBOT."
170 B$="THIS PROGRAM IS ABOUT..."
180 C$=""
190 D$=""
200 E$="      LEARNING TO COUNT"
210 F$=""
220 PRINT : : :
230 GOSUB 670
240 CALL CLEAR
250 ANS=INT(RND*14)+1
260 CALL CHAR(99,"081CFF7E7E3C24")
270 FOR I=1 TO ANS
280 PRINT TAB(RND*9+9);CHR$(99)
290 NEXT I
300 GOSUB 990
310 PRINT " HOW MANY STARS ARE THERE? ": : :
320 GOSUB 860
330 GOSUB 960
340 GOSUB 1170
350 GOSUB 960
360 GOSUB 1110
370 GOSUB 960
380 A=0
390 CALL CHAR(101,"000F")
400 CALL KEY(0,K,S)
410 IF S=0 THEN 400
420 CALL SOUND(-100,880,29)
430 PRINT #1:CHR$(K)
440 IF K>47 THEN 470
450 CALL SOUND(200,110,0)
```

```
460 GOTO 400
470 IF K>57 THEN 450
480 A=A+1
490 IF A<>1 THEN 520
500 A$=CHR$(K)
510 IF ANS<=9 THEN 580
520 IF A<2 THEN 400
530 IF A=3 THEN 570
540 A$=A$&CHR$(K)
550 IF ANS<=99 THEN 580
560 IF A<3 THEN 400
570 A$=A$&CHR$(K)
580 IF ANS<>VAL(A$)THEN 620
590 PRINT #1:"THAT IS CORRECT"
600 GOSUB 1200
610 GOTO 240
620 PRINT #1:"THAT IS INCORRECT"
630 GOSUB 1300
640 GOTO 240
650 GOSUB 670
660 END
670 REM
680 CALL CHAR(101,"000F")
690 PRINT : : : : :A$:B$:C$: : :D$: : E$:F$: : :
:
700 GOSUB 860
710 GOSUB 960
720 PRINT #1:"HI. I AM ROCKY RO-BOT"
730 GOSUB 960
740 PRINT #1:"THIS PRO-GRAM IS ABOUT"
750 GOSUB 960
760 PRINT #1:C$
770 GOSUB 960
780 PRINT #1:D$
790 GOSUB 960
800 PRINT #1:"LEARNING TO COUNT"
810 GOSUB 960
820 PRINT #1:F$
830 CALL CHAR(101,"000F")
840 GOSUB 1060
850 RETURN
860 REM
870 FOR RR=18 TO 23
880 FOR CC=29 TO 30
890 PP=RR
900 IF CC<>30 THEN 920
910 PP=RR+6
920 CALL HCHAR(RR,CC+1,82+PP)
930 NEXT CC
940 NEXT RR
950 RETURN
960 CALL CHAR(101,"000F")
970 CALL SOUND(-50,2000,26)
980 CALL SOUND(-50,4000,25)
990 CALL CHAR(101,"0000000F")
1000 RETURN
```

```
1010 DATA 070F0C0C0F3F3F0F,0000000F,030F3F3F3F3F
3F3F,3F3F3F3F073F073F,073F1F0F070301,00000000003
03F3F
1020 DATA FCFEFEFEE2FAE2FA,E2FEFEFCF8F0F0F,FCFF0
777BBBBBBDD,DDDEEEEEEE0EFE,FFFFFEFCF8F0F0F,F0F0F
0F0F0F0F0F
1030 DATA 0000003C7EDBFFFF,00000003070F1F3F,663C
3CFFFFFFFFFF,000000C0E0F0F8FC,0000010303030303,7
BF3E3C181010101
1040 DATA FFFFFFFFFFFFFFFF,DECEC68686868686,FF,E
F0F0F07010101,BFC04,010202040F081012,FF000000FF0
000AA
1050 DATA FF000000FF0F1F9,FC0C0C1CFCF8F83,202540
4A8080FF,005500AA0000FF,202040418183FF,70E0E0C0C
0808
1060 FOR H=1 TO 7
1070 CALL SOUND(80,RND*50+110,25,2000-RND*50,25,
-3,24)
1080 CALL SOUND(80.RND*50+110,25,2000-RND*50,25,
-3,24)
1090 NEXT H
1100 RETURN
1110 PRINT "             ";CHR$(112);"
";CHR$(113);CHR$(114);CHR$(115)
1120 PRINT "          ";CHR$(116);CHR$(117);CHR$
(118);CHR$(119)
1130 PRINT "         ";CHR$(120);CHR$(121);CHR$(
122);CHR$(120);CHR$(120);C HR$(12 0)
1140 PRINT "         ";CHR$(123);CHR$(124);CHR$(
125);CHR$(126)
1150 PRINT "         ";CHR$(127);CHR$(128);CHR$(
129);CHR$(130)
1160 RETURN
1170 REM
1180 PRINT #1:"HOW MANY STARS ARE THERE"
1190 RETURN
1200 REM
1210 CALL SCREEN(3)
1220 FOR I=1 TO 5
1230 CALL SOUND(-100,880,0)
1240 CALL SOUND(-100,880,8)
1250 CALL SOUND(-100,880,16)
1260 CALL SOUND(-100,880,26)
1270 NEXT I
1280 CALL SCREEN(5)
1290 RETURN
1300 REM
1310 CALL SCREEN(9)
1320 FOR I=1 TO 5
1330 CALL SOUND(-100,110,0)
1340 CALL SOUND(-100,110,8)
1350 CALL SOUND(-100,110,16)
1360 CALL SOUND(-100,110,27)
1370 NEXT I
1380 CALL SCREEN(5)
1390 RETURN
```

```
1010 DATA 070F0C0C0F3F3F0F,0000000F,030F3F3F3F3F
3F3F,3F3F3F3F073F073F,073F1F0F070301,00000000003
03F3F
1020 DATA FCFEFEFEE2FAE2FA,E2FEFEFCF8F0F0F,FCFF0
777BBBBBBDD,DDDEEEEEE0EFE,FFFFFEFCF8F0F0F,F0F0F
0F0F0F0F0F
1030 DATA 0000003C7EDBFFFF,000000003070F1F3F,663C
3CFFFFFFFFFFF,000000C0E0F0F8FC,0000010303030303,7
BF3E3C181010101
1040 DATA FFFFFFFFFFFFFFFF,DECEC686868686 86,FF,E
F0F0F07010101,BFC04,010202040F081012,FF000000FF0
000AA
1050 DATA FF000000FF0F1F9,FC0C0C1CFCF8F83,202540
4A8080FF,005500AA0000FF,202040418183FF,70E0E0C0C
0808
1060 FOR H=1 TO 7
1070 CALL SOUND(80,RND*50+110,25,2000-RND*50,25,
-3,24)
1080 CALL SOUND(80,RND*50+110,25,2000-RND*50,25,
-3,24)
1090 NEXT H
1100 RETURN
1110 PRINT "                    ";CHR$(112):"
";CHR$(113);CHR$(114);CHR$(115)
1120 PRINT "                 ";CHR$(116);CHR$(117);CHR$
(118);CHR$(119)
1130 PRINT "                ";CHR$(120);CHR$(121);CHR$(
122);CHR$(120);CHR$(120);C HR$(12 0)
1140 PRINT "                ";CHR$(123);CHR$(124);CHR$(
125);CHR$(126)
1150 PRINT "                ";CHR$(127);CHR$(128);CHR$(
129);CHR$(130)
1160 RETURN
1170 REM
1180 PRINT #1:"HOW MANY STARS ARE THERE"
1190 RETURN
1200 REM
1210 CALL SCREEN(3)
1220 FOR I=1 TO 5
1230 CALL SOUND(-100,880,0)
1240 CALL SOUND(-100,880,8)
1250 CALL SOUND(-100,880,16)
1260 CALL SOUND(-100,880,26)
1270 NEXT I
1280 CALL SCREEN(5)
1290 RETURN
1300 REM
1310 CALL SCREEN(9)
1320 FOR I=1 TO 5
1330 CALL SOUND(-100,110,0)
1340 CALL SOUND(-100,110,8)
1350 CALL SOUND(-100,110,16)
1360 CALL SOUND(-100,110,27)
1370 NEXT I
1380 CALL SCREEN(5)
1390 RETURN
```

H U G TIBBS NEWSLETTER

January 12, 1985
by: Cecil Crowder

The following D/V 80 file gives address information needed to repair blown sectors on disk. It is a DISK SECTOR BIT MAP and was first downloaded from Compuserve. I have made minor additions and corrections to the original file. This is not a tutorial so you may not understand the contents at first without help.

TI-99/4A DISK FORMAT

The following is a complete and, to the best of my knowledge, accurate description of the Disk Directory format and file storage allocation used by the TI-99/4(A) computer:

▸▸

SECTOR 0 - VOLUME INFORMATION BLOCK

ADDRESS	CONTENTS
0000-0009	Disk name - up to 10 characters
000A-000B	Total number sectors on disk (>0168=360, >02D0=720, >05A0=1440)
000C	(# of sectors/trk) >09=SD >12=DD
000D-000F	'DSK' (>44534B)
0010	>50 = Disk backup protected, >20 = not protected
0011	# of tracks per side (>28=40, >23=35)
0012-0013	# of sides/density (>0101=SS/SD, >0201=DS/SD, >0102=SS/DD, >0202=DS/DD)
0038-end	Sector allocation bit map.

This is a sector-by-sector bit map of sector use; 1=sector used, 0=sector available. The first byte is for sectors 0 through 7, the second for sectors 8 through F, and so on. Within each byte, the bits correspond to the sectors from RIGHT to LEFT. For example, if byte >0038 contained >CF00 then the first byte equals 1100 1111. This means that sectors 0 through 3 are used, sectors 4 and 5 unused and sectors 6 and 7 used.

Information for sector 168 starts at >0065. Therefore, if your disk is SS/SD, all addresses from >0065 to end should be FFFF if it was formatted by DISK MANAGER and has not been tampered with.

▸▸

SECTOR 1 - DIRECTORY LINK

Each 16-bit word lists the sector number of the File Descriptor Record for an allocated file, in alphabetical order of the file names. The list is terminated by a word containing >0000; therefore, the maximum number of files per disk is 127 [(256/2)-1]. Any addresses past >0000 will not catalog, but will still be accessable. If the first address is >0000, move all addresses four digits to left. (eliminating this false address) then the disk will catalog. If the alphabetical order is corrupted (by a system crash during name change, for instance), the binary search method used to locate files will be effected and files may become unavailable.

▸▸

SECTOR >2 TO >21 - FILE DESCRIPTOR RECORDS

ADDRESS	CONTENTS
0000-0009	File name - up to 10 characters
000C	Filetype: >01=Program(memory-image) >00=DIS/FIX >02=INT/FIX >80=DIS/VAR >82=INT/VAR File deletion protection invoked by Disk Manager 2 will be shown by >08 added to the above.
000D	# of (MAXRECSIZE) records/sector
000E-000F	Number of sectors allocated to the file. (Disk Manager 2 will list one more than this number, thereby including this sector in the sector count)
0010	For memory-image program files and variable-length data files, this contains the number of bytes used in the last disk sector. This is used to determine end-of-file.
0011	MAXRECSIZE of data file. >50=80 >FE=254 ect.
0012-0013	File record count, but with the second byte being the high-order byte of the value.
001C - end	Block Link (see note *)

* Note on file storage:

Files are placed on the disk in first-come / first-served manner. The first file written will start at sector >0022, and each subsequent file will be placed after it. If the first file is deleted, a newer file will be written in the space it occupied. If this space isn't big enough, the file will be 'fractured', and the remainder will be placed in the next available block of sectors. The block link map keeps track of this fracturing. Each block link is 3 bytes long. The value of the 2nd digit of the second byte followed by the 2 digits of the first byte is the address of the first sector of this extent. The value of the 3rd byte followed by the 1st digit of the 2nd byte is the number of additional sectors within this extent. Sectors 2 through >21 are reserved for File Descriptor Records and are allocated for file data only if no other available sectors exist. If more than 32 files are stored on a disk, additional File Descriptor Records will be allocated as needed, one sector at a time, from the general available sector pool.

▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸E▸N▸D▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸▸

```
                         TREASURY REPORT:

MONTHLY BEGINNING BALANCE.................$  431.93+
ASSETS (INCOME):
   Library Tapes and Disks          $  57.48
   Subscriptions Income             $ 230.00
   Miscellaneous Income             $  11.10
      (Jo's TIW Manual plus)
   Modem Account                    $ 108.12
   Joysticks/Cassette Leads         $ (14.05)
   Assets Sub-Total.......................$  392.65+
LIABILITIES (EXPENSES):
   Postage/Stamps                   $  29.81
   Stationery Supplies Expense      $  41.94
      (2 boxes comp/paper, disk mailers,
       envelopes)
   Bank Service Charges (17 Mar)    $   4.00
   Miscellaneous Expense            $  64.81
      (Bloxwich Fees, Donations; Group Meeting
       Refreshments)
   Liabilities Sub-Total..................$  140.56-
ENDING MONTHLY BALANCE....................$  684.02+

                          Petty Cash
                     $ 48.00      £ 9.90
                      Checkbook Balances
              American $ 499.31     British £ 46.05
                      Checks for Deposit
                         2 @ £ 10.00

              Exchange Rate at $1.85/$1.90=£1.00
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                    ESTIMATED EXPENDITURES:

             Estimated Postage Expenses  $ 30.00
               Photocopying Expenses = $ 143.60
Vol 1 Iss 10 Feb 88 ~ 15 @ 80 = 1200 @ .02599 = $ 31.19
Vol 1 Iss 11 Mar 88 ~ 23 @ 80 = 1840 @ .02599 = $ 47.82
Vol 1 Iss 12 Apr 88 ~ 19 @ 75 = 1425 @ .02599 = $ 37.03
             Back Issues Reproduced In January
     V1 I1 May 87 ~ 15 @ 12 = 180 @ .02599 = $ 4.68
     V1 I2 Jun 87 ~ 17 @ 12 = 204 @ .02599 = $ 5.30
     V1 I3 Jul 87 ~ 10 @ 11 = 110 @ .02599 = $ 2.86
     V1 I4 Aug 87 ~ 14 @ 10 = 140 @ .02599 = $ 3.64
     V1 I5 Sep 87 ~ 13 @ 11 = 143 @ .02599 = $ 3.72
     V1 I6 Oct 87 ~ 17 @  9 = 153 @ .02599 = $ 3.98
     V1 I8 Dec 87 ~ 13 @ 10 = 130 @ .02599 - $ 3.38

                      ESTIMATED INCOME:

       Subscription Renewals (April/May 1988) $ 108.00
              Library Income (May 1988)  $ 60.00

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                     Buffer Full . . . E N D
                 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```