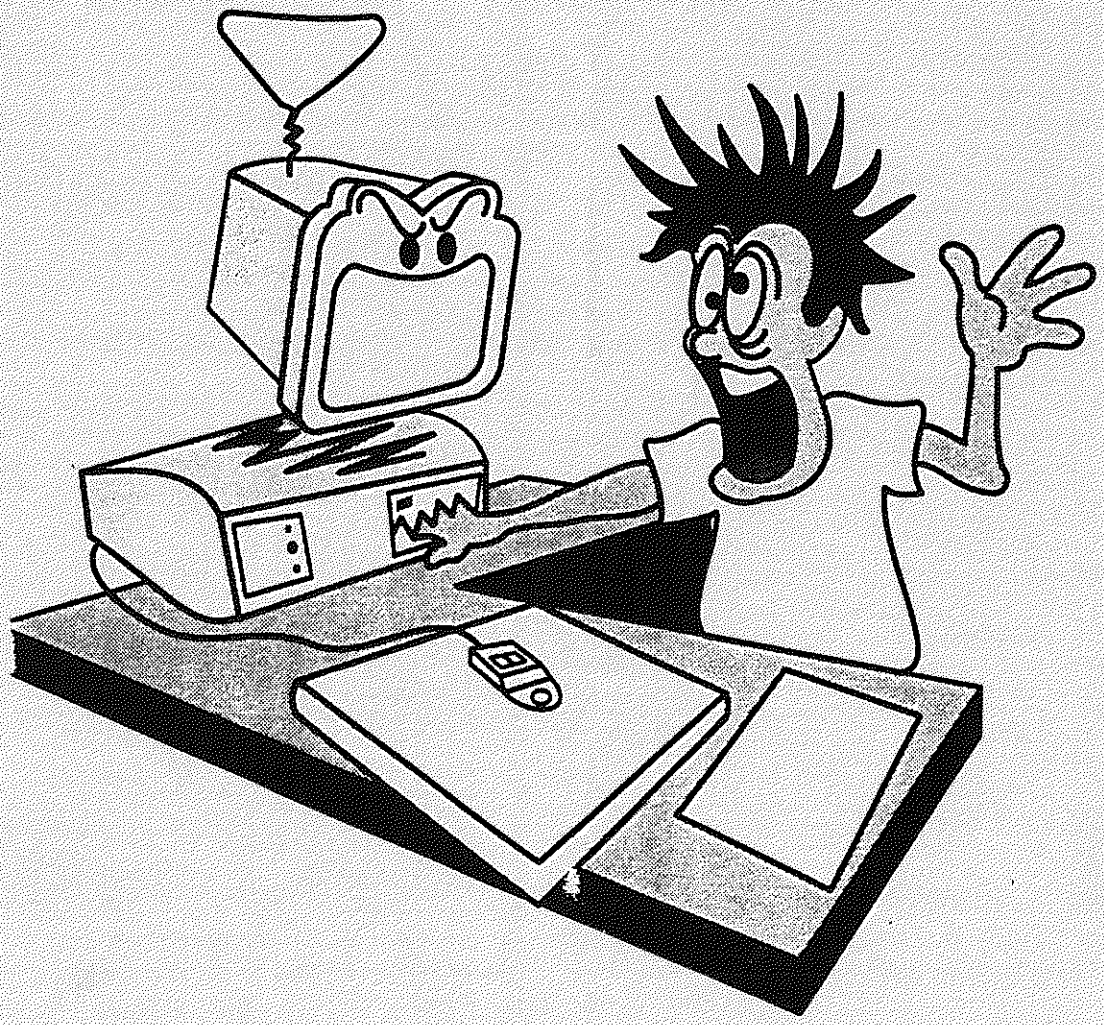
TIsHUG NEWS DIGEST

Focusing on the TI99/4A Home Computer

Volume 14, Number 3 April, 1995

PRINT POST Approved - PP244099/00016



Sydney, New South Wales, Australia $3

TIsHUG (Australia) Ltd.
A.C.N. 003 374 383

TIsHUG News Digest

All correspondence to:
C/o 3 Storey St.
Ryde 2112 Australia

TIsHUG Sydney Meeting

The April Meeting will start at 2.0 pm on the 1st April 1995 at Meadowbank Primary School, Thistle Street, Meadowbank.

Printed by
Kwik Kopy West Ryde

TIsHUG News Digest ISSN 0819-1984

I N D E X

I B M I N D E X

TIsHUG SHOP.

with Percy Harrison.

Due to heavy commitments throughout March I am obliged to get my two TND News Digest articles to the Editor by the March meeting so am not able to report on the attendance and proceedings of that meeting.

I had the opportunity of buying an XT computer for my grandchildren at what appeared to be a very good price. Fortunately, it was suggested that I take the system home and try it out before I paid for it. It was reported to have a 5.25 floppy drive and a small hard drive in the set-up but on examining the equipment we discovered that the hard drive cover, complete with red LED, was just a facade as, when the cover was removed, the hard drive compartment was full of emptyness.

This we could have lived with but then we noticed that some bright person had inserted two 5.25 floppies in the one slot and our immediate thought was that the heads would have been pushed out of alignment thus rendering the drive useless so we decided to remove the two disks and then run the unit to check the drive.

Before powering up the system, we removed the cover from the case, checked that everything was connected up correctly, and then turned the power on. You guessed it, there was a "ppffsst" sound and then nothing. The system had thrown out the circuit breaker in my mains power supply box to the house as well as the 5 amp fuse in the computers power supply unit. Another unusual thing that we noticed was that the glass fuse was actually soldered into the power supply's printed circuit board, not a recommended practice by reputable board manufacturers.

Cyril managed to salvage a fuse holder out of an old 'clapped-out' board that he had and soldered it into the power supply board so that the fuse could be readily replaced without the necessity of using a soldering iron. As we only had access to a 3.2 amp fuse, this was inserted into the board and once again the power was turned on. This time there was a mighty bang and then nothing. The glass fuse had completely disintegrated and a ceramic capacitor had the top half blown off. At this stage we decided that the power supply unit had to go. We replaced it with one from a system that Cyril was putting together so that we could test the rest of the equipment such as the mother board, video card, disk control card, printer card, floppy drive etc.. On firing up, the screen indicated that there was a memory problem so we bypassed this and the system booted up straight into BASIC. No matter what we did we could not get it to boot up into DOS nor could we get it to access the floppy drive so, rather than waste any more time, we decided that the unit was not worth buying and put it all back together to return it to the original owner. Consequently, I am still looking for a cheap computer that will work reasonably well.

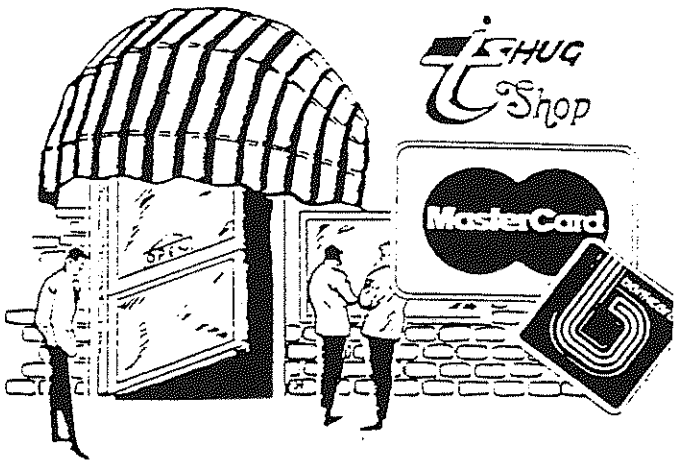
A lot is to be said for the old faithful TI computer, one of which I have been using for about 14 years. It is a complete system with a Corcomp Controller, 32K memory, PE Box fitted with two half height 5.25 double sided double density drives, a Tripletek card and three RAM cards. Except for the occasional crash with one of the RAM cards, which is fixed in about three minutes flat, I have not had any other problems.

The outstanding feature of the TI is that it is so easy to repair if anything does go wrong. I have fixed up many of the dead computers returned to the club under our replacement scheme and only on a few occasions have I had to send the odd one down to Geoff to fix as I do not have the sophisticated repair equipment that he has, nor do I have as much electronic experience.

Having said all of this, I must confess that I do have an IBM compatible computer that I use for specific tasks which are more laborious to perform on the TI. However, I still do all of the club work, including the shop sales and stock records on the good old TI which, at the moment, is being used for more hours per month than the PC.

We have now installed a double speed CD ROM and 16 Bit Sound Card in the club's PC so that we can take advantage of all those demonstration programs that are available on CD's and start demonstrating them at our meetings.

What has all of the above got to do with the shop you may well ask? Well to tell you the truth exactly nothing except for a change I thought that a little bit of waffling on would break the monotony. In any case I am sure that most of you are well aware of what is available from the shop and, if you don't, all you have to do is tell me what you require and if I haven't got it in stock I will endeavour to get it in for you.



```
**********************
*   SAVE/RECALL      *
*  By Todd Kaplan.   *
**********************
*  Notes on Usage    *
**********************
```

(1) The purpose of the above utility is to allow assembly language subprograms to load in a lot faster than the usual way of CALL LOADs with the slow BASIC Loader. This also has the beneficial side-effect of taking up less room on disk, and allowing both XBASIC and a/l programs to reside in the same program.

(2) This utility as is will only work with RELOCATABLE a/l programs. The reason for this restriction is that absolute code does not change any memory pointer, thus making it kind of hard for the utility to locate where the program is in memory.

(3) This utility also saves only the amount of memory that is necessary, that is, none of the utilities like VMBW are saved, because XB loads those in when you type in CALL INIT .

Instructions for Use:

Part 1: Saving the a/l code with your XB program.

(1) Enter XB.

(2) Type NEW

(3) Type CALL INIT

(4) Load in ALSAVE by typing CALL LOAD("DSK1.ALSAVE") (or any appropriate drive).

(5) After ALSAVE is loaded, load in your a/l routines in the same manner, for example,
CALL LOAD("DSK1.A/L-CODE1")
CALL LOAD("DSK1.A/L-CODS2")
or _whatever_ filenames may be appropriate in your case.

(6) When all a/l routines are loaded in, type CALL LINK("SAVE")
This will move the a/l routines from low memory (XB a/l space) to high memory (XB program space) and adjust the necessary pointers to tell XB so that the routines won't be over-written.

(7) Type in 100 REM
as the first statement.
(8) Type in MERGE DSK1.ALLOADM

(9) Finally save the program with SAVE DSK1.PROGRAM
or _whatever_ filename you want for the resultant XBAL program.

Part 2: Using the program created with XBALSAVE.

After the above program is saved, you may add another program after it only by loading in the above program and MERGEing another XB program with it. The entire combination can be saved.

The contents of the ALLOADM program are as follows:

```
10 CALL INIT :: CALL LOAD(8196,63,248) :: CALL LOAD(16376,65,32,32,32,32,32, 255,48):: CALL LINK("A")
```

Here is what each part does:

CALL INIT - Initializes the XB a/l pointers and loads in utilities.

CALL LOAD(8196,63,248) - Loads the address >2004 (LFAL) with the value of >3FF8. This tells XB that we have a program name in the name table.

CALL LOAD(16376,65,32,...,32,255,48) - This puts the name 'A ' into the name table with an entry address of >FF30 (start of recall program) so that we can access the recall program with a CALL LINK("A") .

CALL LINK("A") - Executes the recall program.

The recall program - CALL LINK("A") - does what the save program does, except in reverse. It moves the a/l program from high memory (where it was saved) back to its original position so that it can be run. After the CALL LINK("A"), all of the other CALL LINKs that were accessable at the time they were saved are now fully accessable again. They work exactly as if they were just loaded with a CALL LOAD, except you didn't have to wait for the slow XB loader.

Some additional notes:

(1) As SAVE moves data into high memory, this also limits the size of your XB program. To regain the original amount of programming space, you must either type NEW or RUN another XB program. The above utility could just be used by itself solely as a load program, and then RUN the main program.

(2) As mentioned above, this will only work with relocatable code. The code also has to reside in the normal a/l space for use with XB, namely >2000- >3FFF. If the code is relocatable, you will not be able to load it outside of that range anyway!

(3) The above is almost exactly like my old XBALSAVE, except the method of running the recall program has been changed to make it more reliable. It used to load the Interrupt Service Routine (ISR) # hook at >83C4 with the starting address of the recall program, but that only worked at random.

- Todd Kaplan

DRAWING UP COLUMNS IN TI-WRITER

By Loren West (TIsHUG)

ve you ever wanted to draw columns in TI-Writer, well can show you how I used Transliterations to do the b. You can also produce street maps.

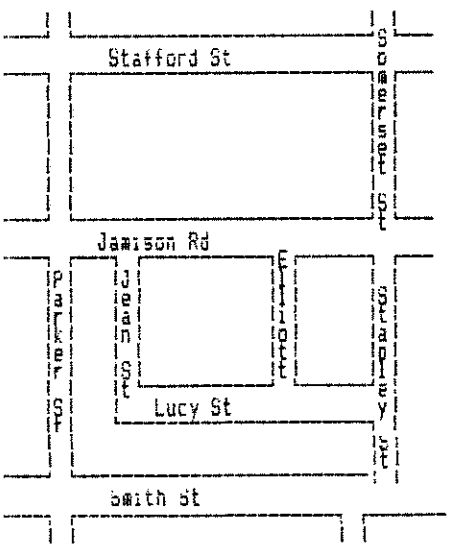
These columns below are example only.

PAYMENT LIST

NAME	CLASS	DATE	AMOUNT $	SPARE	NO
JOHN	4C	12/6	10		11
DAVE	4C	12/6	10		12
ALLAN	5A	13/6	15		13
JOHN	4C	14/6	20		14
JOHN	4C	15/6	5		15
JOHN	4C	16/6	2		16
SUE	2A	16/6	30		17

STREET MAP

Previousaly being an Occupational Health and Safty ommittee member I was put in the position to write to ie local council for problems. Re: The local roads, i: Overhanging trees, blind corners etc. Being able to oduce a map and text through tne same programme made : look more professional and acceotable, leaving less iom for misinterpretation.



The above is the final product. The procedure that I used was:

1. Made a TI-Writer file called COL/S (column/start).

```
TL 33:242     ! : ¬
TL 60:247     < : ¿
TL 62:246     > : ¡
TL 39:240     ' : ¦
TL 91:248     [ : ±
TL 92:243     \ : ¶
TL 93:244     ] : ÷
TL 95:241     _ : ¯
TL 123:250    { : °
TL 125:245    } : ¡
TL 43:249     + : ¹
TL 126:027.049
```

Also making another file called COL/F (column/finish) to put everything back to normal.

```
TL 33:33
TL 60:60
TL 62:62
TL 39:39
TL 91:91
TL 92:92
TL 93:93
TL 95:95
TL 123:123
TL 125:125
TL 43:43
TL 126:126
```

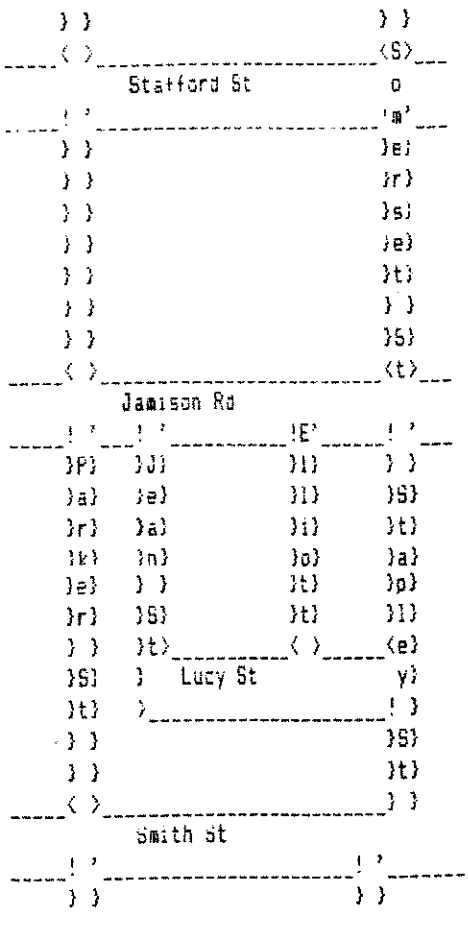
2.Following the tables above I made the columns in TI-Writer as you see below.

The codes I used were for a Star Gemini - 10X printer.

The first 128 characters in the 8-bit character set are (ASCII) they are usually the same from one printer to the next, BUT the characters from 129 to 255 may not be the same, you will have to check with your printer manual, The code from this printer is different to the clubs Hewlet Packard, HP DeskJet printer.

Below is the unformated project typed in through TI-Writer.

```
,________\________\________\________\________\___!
} NAME   } CLASS  } DATE   }AMOUNT $ } SPARE  }NO }
]________[________[________[________[________[___+
} JOHN   } 4C     } 12/6   } 10      }        }11 }
} DAVE   } 4C     } 12/6   } 10      }        }12 }
} ALLAN  } 5A     } 13/6   } 15      }        }13 }
} JOHN   } 4C     } 14/6   } 20      }        }14 }
} JOHN   } 4C     } 15/6   } 5       }        }15 }
} JOHN   } 4C     } 16/6   } 2       }        }16 }
} SUE    } 2A     } 16/6   } 30      }        }17 }
}        }        }        }         }        }   }
>________[________[________[________[________[___(
```



END OF ARTICLE

ARCHIVING.

By TOM ARNOLD.

Archiving, what the hell is that? A little initating maybe? Well you are going to have to live with it whether you like it or not. However, it really is not all that difficult. To begin with, archiving is the squeezing together of files so that they can be transmitted faster or fit on a disk more compactly. There are many good reasons for doing this. The most important is when using bulletin board services. Time costs money, especially if you are hooked up to The Source or Compuserve. By compacting files we can transfer more data in less time and so save money. (Not applicable to TEXPAC which supports DIS/VAR 80 files only).

Another use is for storage on disks. If you send disks to friends by mail you will be able to get more on your disk, thus saving postage. As I'm sure you will come across archived files I thought you would like an explanation of how they work.

The first Archiving program that I saw was Barry Traver's Archiver which was written in Extended Basic, slow but effective. He soon came up with an assembley language version which was much faster. Several other versions are around but the best and most recent is Barry Boone's Archiver II V2.3. If you do not have this one get it from our library. It has several features the others do not have, including a Cataloger option. Archived File Catalog and Return to Funnelweb. Most importantly it archives in two stages so that you can make your files smaller still.

The best way to find out how this program works is to take a disk with half a dozen files and programs on it and archive it. Follow these steps:

1) Load up the Archiver II V2.2 with the E/A module (or FunnelWeb) and select #5 - DSK1.ARC1.

2) Select the option PACK FILES and then name the Disk Drive you wish to pack. Answer "yes" to "packing all files". Send the output to another fresh disk in drive #2. Make sure the disk is blank so that you have room for all the files. Give the packed filed a logical name. Ususally at this stage the files sre given the name as such: NAME_ARC the _ARC indicating that the files have been packed. The packed files will be in the form DISPLAY FIXED 128.

3) Now select the COMPRESS option. Name the output file as such: NAME_ARC the _ARC indicating compressed files. When you come across thes file name you will know you will have to do a two step proceedure to unpack your files. Compressed files have the structure INTERNAL FIXED 128. Answer "All: to pack all the files. The file names will automatically be written to your disk. The compress option only asks for the respective disk drives, make sure you have enough space on your disks.

4) Now that you have packed the files, try unpacking them. you should land up with the same files you started with.

Handy hint: The intermediate files (the packed files) are not really needed and are used only temporarily. I used the RAM Disk to write to so that I saved time and don't have to worry about changing disks. For example, if unpacking I decompress files to the Ram Disk from disk drive #1. Then I unpack from the Ram Disk to disk drive #2.

The space saving using this double packing is quite significant but varies with the types of files you are unpacking. Program files do not shrink as well as text files. One example I tried went from 352 sectors to 233 where another of 66 only went to 54 sectors (it was two program files).

So next time you come across a file name with the suffix _ARC or _ARK then get out that Archiver and don 't forget to send Barry Boon a donation.
His address is:

Barry Boone
Box 1233
Sands Springs,OK
U.S.A. 74063

OTTAWA TI USERS GROUP 5th.MAR.88.

Retyped by John Ryan of TIsHUG for Texpac BBS.

END OF ARTICLE

HINTS FROM THE MEETING 4/2/95

by Ross Mudie, 4th February 1995.

At the TISHUG meetings, a number of the members help others with problems and ways to do things with their computers. A lot of this help given freely from one member to another can benefit a number of other members if the time was taken to write it down. How about a few other members doing this as well? It will add a lot of very good material to the TND if everyone does so.

Here are some of the things that I was involved in at today's meeting:

Q1. How do I get the GWACCEPT program which was on page 20 of the Jan/Feb 95 TND off the BBS and use it in program?

A1. GWACCEPT on the BBS is in a TEXT FILE format. The simplest method is to log on to the TEXPAC BBS with an PC and modem and down load the text file. Save the file on the IBM disk with the name GWACCEPT.BAS. If there is a problem in the terminal program providing the file name extension .BAS the just save it as GWACCEPT and change the filename to GWACCEPT.BAS later.

Once you are ready to use GWACCEPT in a program, it can be included in either of two ways.

a) Use GWBASIC "MERGE" to include the subprogram in an existing program.
b) Load the file into GWBASIC using F3 in the usual way for loading a GWBASIC program and then build the rest of your GWBASIC program around it.

Q2. How can I interconnect a TI computer RS232 and a PC Serial port to transfer text files as suggested in page 7 of the Jan/Feb 95 TND?

A2. Use a cable set up as follows between the computers. This is just like a null modem with hardware flow control.

```
          PC SERIAL PORT.      TI99/4A RS232.
Protective Earth.       1->--<--1 Protective Earth.
Transmit data. out Tx   2-->----2 Receive data.  Rx  (in)
Receive data.  in  Rx   3---<---3 Transmit data. Tx  (out)
Clear to Send in  CTS   5---<---6 DSR (+12V via 1.8 Kohm)
Data Term Rdy out DTR  20-->---20 Data Term Rdy DTR  (in)
Signal Earth            7->--<--7 Signal earth.
Data Set Ready  DSR In  6-<-+     4, 6 and 8 of the PC are
                            |
Ready To Send RTS Out   4->-|     connected together.
Data Carr Detect DCD In 8-<-+
```

Q3. In the above cable is the "Protective Earth" (inter connecting pins 1) required as well as the "Signal Earth" which interconnects pins 7?

A3. The protective earth should be used as well as the signal earth. It is used for a similar purpose as the earth wire of the 3 core mains power cord. It should interconnect the metal cases of the computers. The signal earth interconnects the earth planes of the serial ports of the two computers to complete the serial serial circuit.

Q4. The interconnection of the transmit and receive data lines are fair enough but what are the connections PC-5 to TI-6 and PC-20 to TI-20 for?

A4. These wires implement wired or hardware handshaking. The PC CTS must be a positive voltage to allow the PC to send data on pin 2, Tx data. Because not all TI programs don't use pin 8 for a handshaking line, the restive 12 volt positive will permanently enable the CTS. If the program in use on the TI end of the communications uses pin 8 for hardware handshaking then pin 8 can be used in preference to pin 6, otherwise it is suggested that pin 6 is used to permanently enable the PC CTS.

PC-20 to TI-20 allows the PC to instruct the TI when it is ready to receive. When TI-20 is a negative voltage the TI will not send, but when it is a positive voltage, the TI will send data out on pin 3.

At the meeting, one member had a problem with their 80 column card. The monitor screen was not locked, it was a bit like the horizontal hold control was incorrectly set. Some text from the screen was however visible after leaving the initial screen of the computer. The system worked when first built and then failed when the monitor was reassembled after modification of the synchronising circuit.

The fault was found to be a dry joint in the monitor where the sync circuit was modified. The dry joint occured where the capacitor was soldered on the PC board in the monitor.

END OF ARTICLE

SAVE/RECALL
SOURCE CODE
By Todd Kaplan

```
* Note, User may wish to change the name of 'SAVE' as it
might be one of the user's assembly CALL LINKs.
      DEF  SAVE

M24F4  EQU  >24F4
START  EQU  >FF30        Start of Recall Program
       AORG START

* START OF RECALL PROGRAM
       LIMI 0            DISABLE INTERRUPTS
       MOV  R11,@SAVRTN  SAVE RETURN ADDRESS
       LWPI MYREG        GET WORKSPACE

* RESTORE UTILITY DATA
       LI   R0,M2000     Get Address of Storage Block
       LI   R1,>2000     Get Place to Move To
       LI   R2,4         4 Words to Copy
       BL   @MOVE        Move it!

* RESTORE REF/DEF TABLE
       MOV  @RSTART,R0   GET START OF REF/DEF BUFFER
       MOV  @>2004,R1    GET START OF REF/DEF TABLE
       MOV  @RLEN,R2     GET WORD LENGTH
       BL   @MOVE        MOVE IT

* RESTORE PROGRAM
       MOV  @PSTART,R0   GET START OF PROGRAM BUFFER
       LI   R1,M24F4     GET START OF PROGRAM
       MOV  @PLEN,R2     GET PROGRAM WORD COUNT
       BL   @MOVE        MOVE THE JUNK

* RETURN TO XBASIC
RETURN
       SB   @>837C,@>837C  CLEAR GPL STATUS BYTE
       LWPI >83E0        GET GPL WORKSPACE
       MOV  @SAVRTN,R11  RESTORE RETURN ADDRESS
       RT                RETURN TO CALLING PROGRAM

********
* MOVE * Move CPU TO CPU
********
* R0 = From: Address *MUST BE EVEN ADDRESS*
* R1 = To: Address * MUST BE EVEN ADDRESS
* R2 = # OF WORDS TO MOVE
* BL   @MOVE
MOVE
       MOV  *R0+,*R1+    Move a word from *R0 to *R1
       DEC  R2           Decrement Pointer
       JNE  MOVE
       RT                Return

* STORAGE

M2000  BSS  2            XML VECTOR
M2002  BSS  2            First Free Address in Low Mem
M2004  BSS  2            Last Free Addres in Low Mem
                         (START OF REF/DEF TABLE)
M2006  DATA >AA55        VALIDATION CODE
RLEN   BSS  2            STORAGE FOR REF/DEF TABLE
                         LENGTH
RSTART BSS  2            START OF BUFFER FOR REF/DEF
                         TABLE
PLEN   BSS  2            STORAGE FOR PROGRAM LENGTH
PSTART BSS  2            START OF BUFFER FOR PROGRAM

* END OF RECALL PROGRAM

********
* SAVE * SAVE IT!
********

       AORG >A010        PUT IN >A000 SO THIS IS NOT
                         SAVED WITH RECALL!
MYREG  BSS  32           Work Space
SAVRTN BSS  2            Return Address Storage
SAVE
       LIMI 0            CLEAR INTERRUPTS
       MOV  R11,@SAVRTN  SAVE RETURN ADDRESS
       LWPI MYREG        GET WORKSPACE

* SAVE UTILITY POINTERS (>2000->2004)
       LI   R0,>2000
       LI   R1,M2000
       LI   R2,3         3 WORDS
       BL   @MOVE

* SAVE REF/DEF TABLE

* FIGURE OUT LENGTH OF REF/DEF TABLE (IN BYTES)
       LI   R2,>4000     GET END OF REF/DEF TABLE
       MOV  @>2004,R0    GET START OF REF/DEF TABLE
       S    R0,R2        SUBTRACT START OF REF/DEF
                         TABLE  FROM END TO GET LENGTH
* FIGURE OUT START OF BUFFER FOR REF/DEF TABLE
       LI   R1,START     GET ADDRESS OF START OF THIS
                         PROGRAM
       S    R2,R1        SUBTRACT LENGTH OF REF/DEF
                         TABLE TO GET START OF BUFFER
```

```
    MOV  R1,@RSTART   STORE START OF REF/DEF TABLE
    SRL  R2,1         DIVIDE BYTE COUNT BY 2 TO GET
                      WORD COUNT FOR MOVE
    MOV  R2,@RLEN     STORE REF/DEF TABLE LENGTH
                      (IN WORDS)

)PY REF/DEF TABLE TO BUFFER
) = START OF REF/DEF TABLE
1 = START OF BUFFER FOR REF/DEF TABLE
2 = WORD LENGTH OF REF/DEF TABLE

    BL   @MOVE        COPY REF/DEF TABLE TO CPU
                      BUFFER

AVE PROGRAM

IGURE OUT PROGRAM LENGTH
    LI   R0,M24F4     GET START OF PROGRAM SPACE
    MOV  @>2002,R2    GET END OF PROGRAM SPACE (FFAL)
    S    R0,R2        SUBTRACT START FROM END TO GET
                      BYTE LENGTH

IGURE OUT ADDRESS OF START OF BUFFER FOR PROGRAM
    MOV  @RSTART,R1   GET START OF BUFFER FOR REF/DEF
                      TABLE
    S    R2,R1        SUBTRACT PROGRAM BYTE LENGTH
                      FROM START OF REF/DEF BUFF

    MOV  R1,@PSTART   STORE START OF PROGRAM BUFFER
    SRL  R2,1         DIVIDE BYTE COUNT BY TWO TO GET
                      WORD COUNT
    MOV  R2,@PLEN     STORE PROGRAM WORD LENGTH
    BL   @MOVE        COPY PROGRAM TO PROGRAM BUFFER

ET UP NEW BOTTOM OF XB
    MOV  @PSTART,R0   GET START OF PROGRAM BUFFER
    AI   R0,->10      LEAVE A LITTLE BIT OF SPACE
                      BETWEEN PROGRAM & XBASIC

ET UP START AND END OF XB LINE NUMBER TABLE
    MOV  R0,@>8330
    MOV  R0,@>8332
    DEC  R0           Back one byte
    MOV  R0,@>8386    Store First Free address in
                      High Memory
    B    @RETURN      RETURN TO XB

    END
```

END OF ARTICLE

DISK ASSEMBLY FILE FORMATS.

DISPLAY FIXED 80 (UNCOMPRESSED) tagged object code ay be loaded by option 3 using E/A, option 1 using Mini emory or using CALL LOAD in TI BASIC with either the E/A or MM modules. It can be ABSOLUTE or RELOCATABLE. The Absolute code must always be loaded at the same place in memory while Relocatable code can be loaded anywhere. If the Tagged Object has references to other files or subroutines they will be resolved by the loader, except in the case of the XB loader. If source code does not contain an AORG directive then the code will be, by default, relocatable.

COMPRESSED TAGGED OBJECT code is like Tagged except that the program data is saved in bytes allowing it to load faster. It contains characters outside the printable ascii range and cannot be modified by the E/A editor. Compressed object files are created by using the C option with the assembler. They are NOT loadable by the X/B loader, however they are loadable by basic under E/A and E/A option 3 etc.

MEMORY IMAGE format is the most compact and the fastest loading of Assembly programs and can be stored on disk or cassette. It is identified as a PROGRAM file in a disk catalog and can be loaded with option 5 using E/A, or option 3 using TI-Writer. Please note that the screen will go blank and must be turned back on by the program itself after loading is complete. Memory Image files are produced using the SAVE utility on the E/A disk "B". Memory Image files like BASIC programs can be accessed from/to any I/O device with a single I/O call. That is why they load so fast. There is a size restriction for MEMORY IMAGE FILES OF 8192 BYTES, (hex 2000), although the E/A the TI/WRITER Modules will load multiple Memory Image files to make a larger program. The loader does this by looking for files after the initial file is loaded whose filename is similar except for the last character which is incremented by one. EXAMPLE: The file GAME is loaded. The loader then looks for GAMF, GAMG etc.,if such files are required due to program size.

MEMORY IMAGE assembly files have a 3 word header followed by the data to be placed in memory as follows:
1) The first word is a 'FLAG'. If it is not 0 (zero) i.e. >FFFF then this file is not the last in a multi-file program. For example, if the flag for GAME is >FFFF then there HAS to be at least a file named GAMF, etc.. 2) This word is the length of the Memory Image file in bytes, including the six byte header. The largest value here is >2000. 3) This word is the CPU RAM address where the file is to be loaded. Execution always begins at the first byte of the first segment loaded.

Retyped for TEXPAC BBS by John Ryan,
edited by Ross Mudie.

END OF ARTICLE

LEARN TO KNOW YOUR TI

LESSON 25

with Percy Harrison

Today we will take a look at Line Editing which allows you to call up a line from a program and change it.

EDIT 125

You can make any changes you wish, except the line number must remain the same. This restriction makes EDIT mode less useful than it might be.

Editing a line is conceptually more complicated than simply typing it over again. However, you will learn the editing procedure and use it for most line repairing.

The rules for naming variables in TI BASIC are rather free from restrictions compared to some other versions of BASIC. The name can have up to 15 characters. Names must start with a letter character. You cannot use a reserve (or "key") word as a name. There are certain punctuation marks you may not put in a name. In fact, it is better to leave all punctuation out of names (except the "$" at the end of string names).

LESSON 25 LINE EDITING

PRACTICE MOVING THE CURSOR

Remember how to move the cursor left and right?

FCTN and left arrow key (on "S")

FCTN and right arrow key (on "D")

Push the keys once to move one space.

Hold down the keys and the cursor will keep moving.

FIXING A LINE

You learned this before.

To fix a line that you are typing:

1. Use the arrow keys to move the cursor to the error.
2. Fix things up:

   type the correct letter
   or use the FCTN DEL keys to remove (delete) a letter
   or use the FCTN INS keys to stick in (insert) letters.

3. Press ENTER to put the line in memory.

LINE EDITING: CHANGING A LINE THAT IS IN MEMORY

Sometimes you don't find the error until after the line is put in memory.

Sometimes you just change your mind about the line.

Do you have to type the line over? No!

Suppose you want to change line 623. Do these three things:

1. Enter EDIT 623 to put the line on the screen.
2. Move the cursor to the error in the line using the FCTN and the left arrow keys.
3. Fix the line. Press the ENTER key when done.

PRACTICE LINE EDITING

Enter: 623 PRINT "I LIKE SPINACH"
624 CALL CLEAR

Now enter EDIT 623

Press FCTN and the right arrow key to move the cursor over the "L" in "LIKE".

Press the FCTN INS keys. Type DON'T. This fixes the line to read:

623 PRINT "I DON'T LIKE SPINACH"

Press Enter.

Then LIST 623 to see if the line is correct.

VARIABLE NAMES

Variable names can be up to 15 characters long. They must start with a letter and have only letters and numbers in them. (Except, of course, string variables must end in a dollar sign ($)). Actually, certain punctuation characters are allowed inside names, but you will make fewer errors if you do not allow any.

Good names: JOE
TS
WX$
REDCOW$
FATLETTER1
COWBOY

Bad names: 2SIDE starts with a number
X% has punctuation char.
LET is a keyword

The computer will print:

*BAD NAME

if you try to enter a line that has an incorrect name in it.

It is a good idea to make your variable names describe the variable. Example:

CAR1 and CAR2	in a racing game
HOUSE and HOTEL	in a board game (Monopoly)
GUESS$ and COLOR$	in a colour guessing game

On the other hand, short names are quicker to type.

It's only a short lesson this month but a very important one if you do programming. Next month we will take a look at cutting up strings and gluing them back together in any order.

Assignment 25

1. Load one of your old programs from tape or disk and practice EDITing lines.

ANSWERS TO LESSON 24

Assignment Question 24

```
10 REM_____GOSUB AND RETURN
12 CALL CLEAR
20 REM_____THE FIRST ONE
21 GOSUB 200
30 REM_____NEXT ONE
31 GOSUB 300
40 REM_____THE LAST ONE
41 GOSUB 400
50 REM_____AGAIN
51 GOSUB 200
99 END
200 REM
201 REM_____SUBROUTINE 1
202 REM
210 PRINT "LOOK OUT!"
215 PRINT
250 GOSUB 900
299 RETURN
300 REM
301 REM_____SUBROUTINE 2
302 REM
350 PRINT "RED SMOKE IS POURING FROM "
355 PRINT
360 GOSUB 900
399 RETURN
400 REM
401 REM_____LAST ONE
402 REM
450 PRINT "YOUR COMPUTER!"
455 PRINT
460 GOSUB 900
499 RETURN
900 REM
901 REM_____TIMER
902 REM
930 CALL SOUND(300,800,10)
950 FOR T=1 TO 400
951 NEXT T
999 RETURN
```

Bye for now.

END OF ARTICLE

ENHANCED BASIC.

from Stephen Shaw in England.

Or What You Can Do when you have the Personal Record Keeping or Statistics modules inserted into your console, or, if loaded from disk, if your console thinks they are inserted... Last issue we dealt with the really easy bit, the equivalent calls to ACCEPT or DISPLAY data.
This issue we shall dive into the mysteries of CALL L (load), CALL P (partition), Call H (header), and Call G (getput). The best way to introduce these perhaps is with a working program, then in the next issue we can get down to a little more detail! Program first, then discussion. This program will use an already created PRK data file, and in the TI Basic environment will read that data and display it on the screen. It will demonstrate several of the calls in a fairly simple manner, so you can get used to the idea of using PRK data in a Basic program!

If you have a disk system, free up memory by typing:
```
CALL FILES(1)
NEW
```
before you load and run this program!

As this program occupies some VDP memory, you may find that a few PRK data files will be too long to load. Try deleting some records to make the data file fit.

Remarks will be in lower case between the program lines!

BEFORE LOADING THE PROGRAM you must allocate an area of VDP memory for the data, by using the partition command, in command mode, thus:

```
CALL P(10900)
NEW
```

Now load the following program and run it:

```
first,load the prk file from disk or cassette
100 CALL L("CS1",Y)
or call l("dsk1.filename") as you wish now to check to
see if the file is loaded
110 IF Y=0 THEN 500
no there was an error else carry on as follows how many
fields in each record? lets find out
120 CALL H(1,5,0,F)
where f will return number of fields.
130 CALL CLEAR
140 CALL D(1,1,28,"# OF FIELDS:"STR$(F))
then we need to know how many records there are in the
data file- think of a record as a page:
150 CALL H(1,6,0,R)
160 CALL D(2,1,28,"# of RECORDS:"STR$(R))
now let's loop through each record and display it on the
screen:
170 FOR RCD=1 TO R
taking each field in turn
180 FOR FLD=1 TO F
but does the field contain a number or a string? let's
find out:
190 CALL H(1,10,FLD,TYPE)
200 IF TYPE=1 THEN 240
as type is 0 data is a number:
get the data from field FLD of record RCD:
210 CALL G(1,RCD,FLD,Z,RD)
and print the result on screen:
220 CALL D(2-FLD,1,28,RD)
and jump over string section
230 GOTO 260
this is string section:
240 CALL G(1,RCD,FLD,Z,RDS)
250 CALL D(2-FLD,1,28,RDS) thats one field dealt with,
on to the next:
260 NEXT FLD
and now all fields in the record are on screen lets give
the record number and a pause
270 CALL D(23,18,10,"RECORD:"STR$(RCD))
280 CALL D(24,1,28,"PRESS ENTER FOR NEXT")
290 CALL H(24,28,1,RC,RTS)
and on to the next record
300 NEXT RCD
310 PRINT "NO MORE"
320 GOTO 320
500 CALL CLEAR
510 PRINT "LOAD ERROR"
520 GOTO 520
530 END
```

CALL P sets aside an area of memory for the data to be stored in, and must of course be sufficient to hold the data, but for Basic use, we must also have enough room for our Basic program! After using CALL FILES(1), disk users have a maximum of 12768 bytes of VDP RAM free to split between Basic program and data area, while cassette users have 13820 bytes available (see how much memory a disk system eats up!). By using 10900 for our data, we have left even disk users with at least 1868 bytes for our little program!

CALL L just loads the data file into the partitioned area, the return variable following the device/file name is a 0 (zero) if an error occurs, for instance, the files is not on the disk, the data is too large for the partition, or other I/O errors. The device/filename may be a string variable if you wish,
e.g. CALL L(A$,A)

CALL H deals with the "header" information- the database specification you create when using PRK/Stats: what sort of data is held in each field and what name have we given the field?

As used in the above example (there are many other uses!!!):

```
CALL H(1,5,0,F)
CALL H(1,6,0,R)
CALL H(1,10,FLD,TYPE)
```

The first digit (1) is a READ instruction. We would use a 0 to write. The fourth digit is a variable placed in or obtained from the header data. The second digit (5,6, or 10 above) is the header item number, from 1 to 14, where 5 is the number of fields per record, and 6 is the number of records. 10 is the type of field, returning to the fourth item, a numeric variable, a 1 for characters (string), a 2 for Integer, a 3 for decimal, and a 4 for scientific notation.
We shall deal with the other 11 header items, and with writing to the header, in a later article.

CALL G deals with the actual data we have stored, and as used above (again there are other uses...):

```
CALL G(1,RCD,FLD,Z,RD)
CALL G(1,RCD,FLD,Z,RDS)
```

Again the first item (1) is a READ instruction, with 0 used to write. The second and third items identify the record and field number that we are going to read/write, while the final item is the variable that the contents

of that particular record/field will be placed in for us to use. Note that we must use a string variable for characters!
That fourth item (Z) is a numeric variable, which can be used to indicate a blank field. It takes a value 0 if data is found, and 1 if data is missing, that is, not entered!
When using CALL G to WRITE, the fourth parameter as above is not used, you just go on to the value you are writing, for instance:
CALL G(0,RCD,FLD,"STRING INPUT")
That fully covers CALL G, apart from a sample of how to write!

CALL S will Save the data area to cassette/disk, very easily indeed:
CALL S(DEVICE$,VARIABLE)
with variable again indicating success or failure!

That leaves us with a great deal to look at with CALL H, and also we need to look at writing data, and (perhaps!) maybe even creating a data file WITHOUT using the PRK create file routine. Until later...
((Is anyone reading this please!!! Is this the right level/approach for you?))

END OF ARTICLE

JUST A ONE LINER

I have five noses, six mouths and seven ears. What am I? Quite ugly.

Just when you thought you were winning the rat race, along came the faster rats.

MULTIPLE FRACTURES

This file should be read in conjunction with the Disk Fixer Manual by Naverone. It may help to you understand how the TI99/4A Disk Operating System fits a file or program into available sectors and still keeps track of the file as the subject is quite hard to follow in the Disk Fixer manual.

Dealing with MULTIPLE FRACTURES
by Terry Atkinson (TI6450)

In the Feb issue of the TINS newsletter, I introduced, among other things, details on how TI DOS handles fractured files. This month, I wish to further expand on that segment.

Before I do, however, I wish to bring to your attention, bytes >0012 and >0013 of the FDB. In the article, I described their use for "FIXED" type files. Contrary to some people's belief, they ARE used for variable type files as well, but since they more-or-less duplicate other sector addresses, I left that information out. In fact, for "variable" type files, byte >12 contains the low order bye for the number of sectors used in the file (same as byte >F) and byte >13 contains the high order byte for the number of sectors in the file (same as byte >E). These bytes are not used by program type files.

Having cleared the air, let's get on with the main theme: MULTIPLE FRACTURES. Recall that a file may be fractured up to 76 times (that's all the room there is in the FDB to handle 6 byte fracture pointers). For my example, I will use the following (sure hope Paul doesn't reformat these examples!!). I am looking at FDB >3 and find a program called "TEST". It contains several fractures, the pointers starting at sector >3 address >001C are:

```
ADDR: 001C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B
CONT:   74 50 00 88 30 01 38 31 02 05 40 02 07 50 02 0C
ADDR: 2C 2D 2E 2F 30
CONT: 60 02 14 20 03
```
The remainder of the sector contains all zero's.

Recall that we grouped the bytes in groups of 3, then numbered each nybble from 1 to 6. Then we rearranged the nybble order to read 412563. The 412 nybble became the start sector for the fracture, and the 563 became the number of sectors in the offset. Since we are dealing with multiple fractures, I will mention at this point that the sector offset is cumulative...it always keeps track of the TOTAL number of sectors from the beginning of the file. Also, for the first fracture, you should add "1" to the value, since a file which takes only 1 sector has an offset of "0". I'm sure you can appreciate, in simple mathematics, that 9-1=8, but counting on your fingers from 1 to 9 produces 9

different combinations. "00", therefore, is a combination. So here are the bytes grouped and numbered.

Bytes: 745000 883001 383102 054002 075002 0C6002 142003
#'d : 123456 123456 123456 123456 123456 123456 123456

Now, rearrange in 412-563 groups and separate them. The groups should now look like this:

074 005, 088 013, 138 023, 005 024, 007 025, 00C 026, 014 032

Ok, the hard part is done. We can safely say our program contains 7 fractures. The first fracture starts at sector >074 and continues for >005 sectors. But lets analyze all the fractures. We need to fill space in this newsletter.

First fracture: Starts at sector >074(116) and continues for >005 sectors. The end sector for this fracture is >074+>005=>079...a total of 6 sectors >05+1.

Second fracture: Starts at >088(136) and continues for >13-(>5+1)=>D sectors. The end sector for this fracture is >088+>0D=>095(149).

Third fracture: Starts at >138(312) and continues for >023-(>13+1)=>F sectors. The end sector will be >138+>F=>147(327).

Fourth fracture: Starts at >005 and continues for >024/(>23+1)=0 sectors. The end sector is >005+>0=>005.

Fifth fracture: Starts at >007 and continues for >025-(>24+1)=0 sectors. The end sector for this fracture is >007+>0=>007.

Sixth fracture starts at >00C and continues for >26-(>25-1)=0 sectors. The end sector for this fracture is >00C+>0=>>00C.

Seventh and final fracture is at >014 and continues for >032->26-1=>B sectors. The end sector for this fracture (and the program) is >014+>B=>01F.

The total number of sectors used for this file is the final offset-1 (>32+1=>33 sectors...51 in decimal. If you were to catalog this disk, it would show 52 sectors used for this program. Don't forget the FDB!!.

Ok, so, I think that explains the subject a bit better. Have a look at a few files of your own to see if you can find the start/end sectors and fracture points. With very little practice, you will be able to read the block links without problems. Guaranteed!

END OF ARTICLE

HOW TO ACCESS / FILES

By Jerry Keisler
Retyped by Robyn West from Spirit of 99

I have been asked by several members how to run some of the programs on our DOM. The following is a list of what you may find in your disk directory and how to run it.

If your disk has a LOAD bile, it may run all the files on that disk regardless of type.

PROGRAM FILES (PG)

There are several options for running these files.

EXTENDED BASIC (XB)

Will load and run automatically when you select XB and the disk is in drive #1, if it has a LOAD file. Or can be run by typing:
RUN (enter) or RUN "DSK1.name" (enter)

If a program loads correctly but you get a BAD VALUE error when it runs, you need to load the program into BASIC (no CHARS above 143 are allowed in XB). If the program file is more than 45 sectors and won't load, you have to open up more memory. Do this by typing:

CALL FILES(1) (enter)
NEW (enter)
OLD DSKn.name (enter)
RUN (enter)

BASIC

Programs need to be loaded by typing:

OLD DSKn.name (enter)
RUN (enter)

Most BASIC programs will load and run in XB but not viceversa. If you get a FOR-NEXT error in line xxx and when you edit the line you get a lot of nonsense, the program is written in XB. The same is true if the sectors are greater than 45. More space is needed in the computer. See CALL FILES above. If you still get a memory full and have tried XB then most likely it can be run on tape (OLD CS1) without the expansion box turned on.

EDITOR ASSEMBLER (EA)

If a program file will not load and run in BASIC or XB and gives an I/O ERROR 58, it may be an assembly language program. This can be run using BOOT, the

editor assembler module option #5 with DSKn.name or FUNNELWEB'S RUN option of disk review. These program files are listed in consecutive order such as MASS, MAST, MASU or UTIL1, UTIL2, UTIL3. The files will normally have 33 sectors.

GRAMULATOR

These require a GROM simulator card or box. The files contain 34 sectors and have the same name with the numbers through 66 attached to files 2 through 8. Follow your GROM simulator instructions to load and run.

OTHER PROGRAM FILES

Some specialized program files can only be loaded from special module such as ADVENTURE (54 sectors), PERSONAL RECORD KEEPING, STATISTICS, TUNNELS OF DOOM (52 sectors).

DIS/VAR 80 FILES (D/V 80)

These are usually text or documentation files (DOCS, READ ME, ETC.). They are instructions on how to run programs on the disk. Read or print them using TI Writer, FUNNELWEB BOOT or the (V, T or P) option of several disk managers.

DIS/VAR 163 FILES (D/V 163)

This is an XB file in MERGE format. It can be merged into a program already in computer memory. Type:
MERGE DSKn.name (enter)
You must do this even if no program is in memory. To save a file in MERGE format, type:

SAVE DSKn.name, MERGE - works in XB only.

DIS/FIX 80 FILES (D/F 80)

Use EDITOR ASSEMBLER MODULE, LAR or FUNNELWEB3 to load these. Some files will auto load and/or auto start. Use LOAD and RUN option #3. Type:

DSKn.name (enter)

If the program does not run but ask for a second file name, you must do #1 and #2 or just #2 below.

#1 If there are multi files for the programe, type: DSKn.name (enter) for each file.

#2 The program name could be START, BEGIN, GAME, LOAD, RUN, etc. FUNNELWEB will give you a list of names found in the program.

DIS/FIX 128 FILES (D/F 128)

These are usually ARCHIVED files. You must unarchive these files before you can run them. Use AAC303 or AAC303G.

INT/VAR 254 FILES (I/V 254)

These files usually have more than 45 sectors and are XB requiring memory axpansion. They do not require CALL FILES (1). BASIC cannot be used. The same commands are used such as RUN or OLD DSKn.name. The programes are usually so long that they connot be saved to tape. (SAVE CS1)

DATA FILES

Files such as INT/FIX 108, INT/VAR 128, INT/VAR 64 and some program files are data files that can be used by a program on the disk. They will not RUN and should be left on the disk with the other programs.

QUICK REFERENCE CHART

By Jerry Keisler

TYPE	SIZE	TRY
PG	-----	EXTENDED BASIC, BASIC, EDITOR/ASSEMBLER
PG	33	EDITOR/ASSEMBLER 5, BOOT, FUNNELWEB
PG	34	GROM SIMULATOR
PG	52	TUNNELS OF DOOM MODULE
PG	54	ADVENTURE MODULE
DV80	-----	TI WRITER, FUNNELWEB
DV163	-----	EXTENDED BASIC MERRGE FORMAT
DF80	-----	EDITOR/ASSEMBLER 3
DF128	-----	ARCHIVER
IV245	-----	EXTENDED BASIC
IV254	-----	EXTENDED BASIC
ANY?	-----	DATA

TI-ARTIST FILES

ENDS WITH	TYPE	DESCRIPTION
_C	PG25	PICTURE / COLOUR FILE
_F	DV80	CHARACTURE FONT FILE
_I	DV80	INSTANCE FILE
_M	DV254	MOVIE FILE
_P	PG25	PICTURES / PATTERN FILE
_S	DV80	SLIDES FILE
_V	DF12	VECTORS FILE

TI-BASE FILES

ENDS WITH	TYPE	DESCRIPTION
/C	DV80	COMMAND FILE
/C	IF48	COMMAND FILE
/D	IF--	DATA BASE DATA FILE
/H	DV80	HELP FILE
/P	IF255	PROGRAME FILE
/S	IF255	DATA BASE STRUCTURE FILE

END OF ARTICLE

TI CARTRIDGES OFFICIALLY RELEASED BY TEXAS INSTRUMENTS

PHM3030 A-Maze-ing
PHM3090 Addition
PHM3027 Addition and Subtraction 1
PHM3028 Addition and Subtraction 2
PHM3041 Adventure
PHM3115 Alien Addition
PHM3114 Alligator Mix
PHM3056 Alpiner
PHM3031 Attack, The
PHM3003 Beginning Grammar
PHM3151 Bigfoot
PHM3033 Blackjack and Poker
PHM3032 Blasto
PHM3054 Car Wars
PHM3148 Championship Baseball
PHM3110 Chisholm Trail
PHM3083 Computer Math Games II
PHM3085 Computer Math Games III
PHM3088 Computer Math Games VI
PHM3038 Connect Four
PHM3097 Decimals
PHM3116 Demolition Division
PHM3001 Demonstration
PHM3093 Division
PHM3049 Division 1
PHM3117 Dragon Mix
PHM3002 Early Learning Fun
PHM3144 Early LOGO Learning Fun
PHM3015 Early Reading
PHM3055 Editor/Assembler
PHM3100 Equations
PHM3009 Football
PHM3095 Fractions
PHM3037 Hangman
PHM3006 Home Financial Decisions
PHM3156 Honey Hunt
PHM3007 Household Budget Management
PHM3023 Hunt the Wumpus
PHM3034 Hustle
PHM3155 I'm Hiding
PHM3024 Indoor Soccer
PHM3094 Integers
PHM3099 Laws of Arithmetic
PHM3156 M*A*S*H
PHM3101 Measurement Formulas
PHM3152 Meteor Belt
PHM3119 Meteor Multiplication
PHM3113 Microsoft Multiplan
PHM3025 Mind Challengers
PHM3058 Mini-Memory
PHM3118 Minus Mission
PHM3131 Moonmine
PHM3092 Multiplication
PHM3029 Multiplication 1
PHM3057 Munch Man
PHM3020 Music Maker
PHM3004 Number Magic
PHM3098 Number Readiness
PHM3050 Numeration 1
PHM3051 Numeration 2
PHM3067 Othello
PHM3112 Parsec
PHM3097 Percents
PHM3022 Personal Real Estate
PHM3013 Personal Record Keeping
PHM3044 Personal Report Generator
PHM3010 Physical Fitness
PHM3082 Reading Flight
PHM3043 Reading Fun
PHM3046 Reading On
PHM3048 Reading Rally
PHM3047 Reading Roundup
PHM3059 Scholastic Spelling--Level 3
PHM3060 Scholastic Spelling--Level 4
PHM3061 Scholastic Spelling--Level 5
PHM3062 Scholastic Spelling--Level 6
PHM3012 Securities Analysis
PHM3150 Sewermania
PHM3146 Sneggit

PHM3157 Sound Track Trolley
PHM3149 Space Bandit
PHM3011 Speech Editor
PHM3225 Star Trek
PHM3014 Statistics
PHM3178 Story Machine
PHM3091 Subtraction
PHM3153 Super Fly
PHM3016 Tax/Investment Record Keeping
PHM3017 Terminal Emulator
PHM3035 Terminal Emulator II
PHM3154 Terry Turtle's Adventure
PHM3026 TI Extended BASIC
PHM3053 TI Invaders
PHM3040 TI LOGO
PHM3109 TI LOGO II
PHM3111 TI Writer
PHM3052 Tombstone City: 21st Century
PHM3064 Touch Typing Tutor
PHM3042 Tunnels of Doom
PHM3008 Video Chess
PHM3018 Video Games I
PHM3005 Video Graphs
PHM3021 Weight Control and Nutrition
PHM3169 Word Invasion
PHM
3185 Word Radar
PHM3039 Yahtzee
PHM3036 ZeroZap

Related titles released on a very limited basis by Scott, Foresman on their own are as follows:

30226 Addition and Subtraction 3
30229 Decimals 1
30244 Decimals 2
30220 Fractions 1
30238 Fractions 2
31177 Frog Jump
30223 Multiplication 2
31189 Number Bowling
31180 Picture Parts
31186 Pyramid Puzzler
30117 Reading Adventures
30115 Reading Cheers
30121 Reading Power
30113 Reading Rainbows
30119 Reading Trail
30123 Reading Wonders
31192 Space Jouney
31183 Star Maze

They are included in this list, since the original plan had been for them to be released jointly by Scott, Foresman and Texas Instruments.

This list was compiled by Barry A. Traver, 835 Green Valley Drive, Philadelphia, PA 19128 (telephone:215/483-1379). It does not include third- party cartridges, which may be handled in a separate list.

Comments, additions, or correc- tions are welcome!

END OF ARTICLE

PUZZLE

This months list of words is based around the subject of "WATER"

S R L V E Z B W V Y D B I O K G K O N S
N L I J T I J Q L R X J I F L Y G J O Y
G B G L S H N R W R L I V P P G I Z L Y
B W W D E F N L K Z I C V C T W L P O B
Z O H R I A P W Y H O A S C C O P L N R
F Y Q T B M B H A L E X I B T U H Y H H
U M T X A I K Y O K R E G S S T M N C J
K L O O B W F U R X X T I B Y P P W H B
R E B O F O R U L K R P O H H K S V X A
Y V W J N S I G Q H X V Y W J O X D B Y
O E G V L T R U B V Q D R Y E V L E K F
V L G B F A A I X Y R E N M S R S E T M
S V W D V S K P Y A N N E K K H A Z V A
V F O E A P Q G N I Y L K P O T O R Y I
N W D H I O O T V F O V O R I D I L G N
N L H W P R Z I N N O Y T D L L N Z O V
H F R M C T D Z S M C A P U I F Z M W P
T H S W T S K E V P G M H L P Q X Q G Q
P X E Q O N Y U T E W Y Y H A J F O S K
G G D I N U J D P Y L I N B U Z O M I G

Find these hidden words

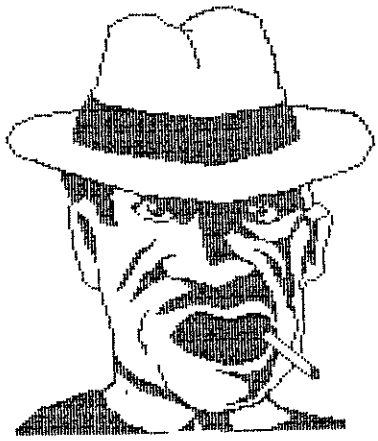
HOW TO PLAY

In this puzzle there are (20) words somewhere, horizontally, vertically, diagonally even backwards.

GOOD LUCK

BABIES	COLOURS	DIVINER
DOWN	FOWL	GRAVE
HOLE	HYDRANT	LEVEL
LILY	MAIN	MELON
NYMPH	ONTAP	PISTOL
POLO	SHORTAGE	SPORTS
SUPPLY	TOWER	

This puzzle was compiled using Ashley Lynn's prograne "Word Puzzle" which can be ordered through TIsHUG.



TREASURER'S REPORT

by Cyril Bohlsen

Income for previous month ..... $ 1427.00
Expenditure for previous month .. $ 1425.61
Profit for previous month ..... $ 1.39
Membership accounted for $ 116.00 of Income.
Shop sales ..... $1311.50 of Income.

The expenditure was made up of the following

Printing & Postage of TND ..... $ 286.61
Administration cost ..... $ 220.00
Shop purchases ..... $ 919.00

END OF ARTICLE

LANGUAGES

CLUB INFORMATIQUE MONTREAL

August 1988.

(INFORMATIQUE means (science) Computer science;(techniques)data processing.)

CIM99 ----------aout 88

Ce texte a ete remis par Bernard Pelletier pour publication dans CIM99.

CIM99

The following is from the May 1986 issue of the Chicago Times, the newsletter of the Chicago TI-99/4A Users Group. If you want to share this please do but please state where it came from.

HOW TO OFFEND EVERYONE.

(Or what language you program in, reveals your personality.)

BASIC:

The universal language of microcomputers. BASIC was probably the first computer language most of us learned. BASIC users are flexible and don't care that it isn't fast. It gets the job done with minimum fuss, hence, BASIC users are get-it-done-types, pragmatic and either too busy or too lazy to learn another language, depending on your point of view. This has also made them a bit defensive about their programming skills, as they may not always be in touch with the more exotic aspects of advanced programming. For example, to many BASIC programers the term "relocatable object code" means the street address of their new home. BASIC programmers most likely work in the Family Room while wearing blue jeans and a User Group T-shirt and while drinking Coke or Pepsi. The motto of the BASIC user is probably: "I can do everything I need to do in BASIC and I'am too busy (lazy) to learn another language."

LOGO:

The language for Kids. LOGO users rarely confess that they program in this language, since I suspect that most of them are really adults. This has made these programmers "Closet regressives". How many adults do you know who will admit that they talk to turtles?. LOGO programmers are also unnecessarily enamored of something called "recursion", which is the ability of a routine to, Call itself, like this. This also demonstrates that they are easy to please. LOGO programmers work at a small desk in the bedroom while wearing those pajamas with feet on the bottom and drinking milk Motto: "My kids love it ".

PASCAL:

For the 99/4A user this requires the p-Code card and writing progames in something known as UCSD Pascal. Which stands for Unclear Code Structered Diligently. Pascal is the programming language most often taught in colleges. Being a more educated group, Pascal programmers tend to be eletist and usually suffer from superiority complexes. They know that not everyone can afford a p-Code card (the name for which, incidentally was inspired by e.e. cummings), making a special group. It is a little known fact that the same people who program in Pascal drive VOLVOS and know how to operate a Cuisinart. That's right, this is the programming language of Yuppies. They program in the den with a view of the lake while wearing designer jeans and Izod shirt and Topsider deck shoes while drinking either white wine chilled to exactly 52 degrees or fresh carrot juice. Motto: "You mean you DON'T program in Pascal?".

ASSEMBLY:

Power to the nerds. To program in Assembly requires both a high intelligence and a low sociability quotient. I believe it states this somewhere in the manual. These people are rugged individualists in the worst sense of both halves of that term. They are highly competive. No assembler programmer ever admired the source code of another assembler programmer. They tend to be Obsessive-Compulsives, and don't just have a tolerance for the minute details of things, they love those details! An assembler programmer will work for 17 weeks on a routine they could have done in 5 minutes in BASIC or 30 seconds with a calculator. They are much less concerned with how a program looks than they are with what it does and they appear to follow this same creed in their choice of clothing. While they are occasionally embarrassed by their lack of social skills, on the other hand Assembly programmers do not really care what you think about them, they already KNOW they can program you under the table. They can often be seen before or after user group meetings demonstrating their latest bit-juggling routines. Those sitting at the front of the Demo oohing and aahing are the BASIC programmers, those sitting in the back whispering that they wrote an even better routine last year are other Assembler programmers. Assembler laguage programmers have a great deal of difficulty with the English language, (don't we all), they program in the garage wearing whatever happens to be on the floor next to the bed when they got up and drinking something they found in the refrigerator which might or might not be orange juice. Motto: "MOV FR1,FR2"

FORTH:

This is the language that TI released into public domain just after they called it quits. FORTH programmers want the power of Assembly but don't want to spend the time pondering the existential significance of memory to memory architecture of the successful disassembly of DSR. As a result of this FORTH programmers feel a bit smug about having all the power of the TI available to them without all the work. Some of them are reformed Assembly language programmers and many of them can speak syntactically correct English much of the time. They occasionally exhibit inferiority complexes due to trying to keep up with Assembly programmers and when their language is put down they can be defensive and have been known to pop their stacks. Considering the syntax of forth invented by Charles Moore apparently while in a temporary psychotic state, many of these programmers develop severe thinking disorders from attempting to actually follow all the DUPs ROTs, and SWAPs they engage in. The language has also attracted a following among Sociopathic con men. Since its code is actually a digital version of the old "shell game". It is also a little known fact that FORTH programmers don't actually write FORTH programs they "develop tools" for solving problems. In fact this is all they do making them the Black and Deckers of programming. FORTH programmers usually work in the utility room or even better in the tool shed, while wearing blue workshirts with their names stitched above the pocket and while driking Miller's beer. Motto: "It's just as good as Assembley, it really is, really."

:C:

There are actually several different types of people involved with this latest addition to the collection of TI programming languages. While it seems that quite a few people in the group have Clint Pulley's adaptation of C for the TI, no one actually does anything with it. Thus, C is for programming procrastinators who have all of the languages mentioned above and intend to learn them all, "whenever I get the time". Unfortunately for them, they aren't fooling anyone. This process also involves "getting a few good C programming books first", meaning a shelf of about 15 volumes. One of them is inevitably titled "From BASIC to C", in which is the intellectual equivalent of going from Philadelphia to Pluto. Those few who actually have done something in C are proganning gadflies who want the latest language available, no matter how unreadable its source code. This is so that they can come to Group Meetings and tell anyone who will listen that they have actually written something in C. As soon as the next language comes along they will go with it. Some C programmers are depressives who have been known to become suicidal searching for a missing bracket in their source code. The language got its start as a term paper of an Assembler programmer. The name came from the Grade it was given for readability and the crayon didn't help any either, C programmers work wherever and whenever it is convienent, wearing the same cloths you saw them in and drinking the latest no sugar, no caffeine, no taste cola. Motto: "And the next language i'll be learning is uh...FORTH, er, Assembley, no, C. Yeah, C. That's it. That's the ticket!"

Retyped for TEXPAC BBS by John Ryan of TIsHUG.

END OF ARTICLE

IBM

Computer Virus Myths
(8th Edition, March 1992)

by Rob Rosenberger
with Ross M. Greenberg

A number of myths have surfaced about the threat of computer "viruses". There are myths about how widespread they are, how dangerous they are, and even myths about what a computer virus really is. We'd like the facts to be known.

The first thing to learn is that a virus is a malicious programming tech- nique in the realm of "Trojan horses." All viruses are Trojan horses, but few Trojan horses can be called a virus.

That having been said, it's time to go over the terminology we use when we lecture:

BBS Bulletin Board System. If you have a modem, you can call a BBS and leave messages, transfer computer files back & forth, and learn a lot about computers. (What you're reading right now, for example, most likely came to you from a BBS.)

Bug: An accidental flaw in the logic of a program which makes it do things it shouldn't really be doing. Programmers
don't mean to put bugs in their program, but they always creep in. Programmers tend to spend more time debugging their programs than they do writing them in the first place. Inadvertent bugs have caused more data loss than all the viruses combined.

Hacker: Someone who really loves computers and who wants to push them to the limit. Hackers have a healthy sense of curiosity: they try doorknobs just to see if they're locked, and they tinker with a piece of equipment until it's "just right." The computer revolution itself is a result of hackers.

Shareware: A distribution method for quality software available on a "try before you buy" basis. You pay for the program only if you find it useful. Shareware programs can be downloaded from BBSs and you are encouraged to give evaluation copies to your friends. Many shareware applications rival the power of off-the-shelf counterparts, at just a fraction of the price. (You must pay for the shareware you continue to use -- otherwise you're stealing software.)

Trojan horse: A generic term describing a set of computer instructions purposely hidden inside a program. Trojan horses tell a program to do things you don't expect it to do. The term comes from a legendary battle in which the ancient city of Troy received the gift of a large wooden horse. The "gift" secretly held soldiers in its belly, and when the Trojans rolled it into their fortified city....

Virus: A term for a very specialized Trojan horse which spreads to other computers by secretly "infecting" programs with a copy of itself. A virus is the only type of Trojan horse which is contagious, like the common cold. If it doesn't meet this definition, then it isn't a virus.

Worm: A term similar to a Trojan horse, but there is no "gift" involved. If the Trojans had left that wooden horse outside the city, they wouldn't have been attacked. Worms, on the other hand, can bypass your defenses without having to deceive you into dropping your guard. An example is a program designed to spread itself by exploiting bugs in a network software package. Worms are usually released by someone who has normal access to a computer or network.

Wormers: The name given to the people who unleash destructive Trojan horses. Let's face it, these people aren't angels. What they do hurts us. They deserve our disrespect.

Viruses, like all Trojan horses, purposely make a program do things you don't expect it to do. Some viruses are just an annoyance, perhaps only displaying a "Peace on earth" greeting. The viruses we're worried about are designed to destroy your data (the most valuable asset of your computer!) and waste your valuable time in recovering from an attack.

Now you know the difference between a virus and a Trojan horse and a bug. Let's get into some of the myths:

"All purposely destructive code comes as a virus."

Wrong. Remember, "Trojan horse" is the general term for purposely destructive code. Very few Trojan horses actually qualify as viruses. Few newspaper or magazine reporters have a real understand of computer crimes, so they tend to call almost anything a virus.

"Viruses and Trojan horses are a recent phenomenon."

Trojan horses have been around since the first days of the computer; hackers toyed with viruses in the early 1960s as a form of amusement. Many different Trojan horse techniques emerged over the years to embezzle money, destroy data, etc. The general public didn't know of this problem until the IBM PC revolution brought it into the spotlight. Banks still hush up computerized embezzlements (as they did during the 1980s) because they believe customers will lose faith in their computer systems if the word gets out.

"Viruses are written by hackers."

Yes, hackers have purposely unleashed viruses, but so has a computer magazine publisher. And according to one trusted military publication, the U.S. Defense Department develops them as weapons. Middle-aged men wearing business suits created Trojan horses for decades before the advent of computer viruses. We call people "wormers" when they abuse their knowledge of computers. You shouldn't fear hackers just because they know how to write viruses. This is an ethics issue, not a technology issue. Hackers know a lot about computers; wormers abuse their knowledge. Hackers (as a whole) got a bum rap when the mass media corrupted the term.

"Viruses infect 25% of all IBM PCs every month."

If 25% suffer an infection every month, then 100% would have a virus every four months assuming the user took no preventive measures -- in other words, every IBM PC would suffer an infection three times per year. This astronomical estimate surfaced after virus expert (and antivirus vendor) Dr. Peter Tippett published "The Kinetics of Computer Virus Replication," á complex thesis on how viruses might spread in the future. Computer viruses exist all over the planet, yes -- but they won't take over the world. Only about 400

different viruses exist at this time and some of them have been completely eliminated "from the wild." (Of course, virus experts retain copies even of "extinct" viruses in their archives.) You can easily reduce your exposure to viruses with a few simple precautions. Yes, it's still safe to turn on your computer!

"Only 400 different viruses? But most experts talk about them in the thousands."

The virus experts who "originate" these numbers tend tto work for antivirus firms. They count even the most insignificant variations of viruses as part of the grand total for advertising purposes. When the Marijuana virus first appeared, for example, it displayed the word "legalise," but a miscreant later modified it to read "legalize." Any program capable of detecting the original virus will detect the version with one letter changed -- but antivirus companies count them as "two" viruses. Such obscure differentiations quickly add up.

"Viruses could destroy all the files on my disks."

Yes, and a spilled cup of coffee will do the same thing. If you have adequate backup copies of your data, you can recover from any virus or coffee problem. Backups mean the difference between a nuisance and a disaster. It is safe to presume there has been more accidental loss of data than loss by viruses and Trojan horses.

"Viruses have been documented on over 300,000 computers (1988)."

"Viruses have been documented on over 400,000 computers (1989)."

"Viruses have been estimated on over 5,000,000 computers (1992)."

These numbers come from John McAfee, a self-styled virus fighter who craves attention and media recognition. If we assume it took him a mere five minutes to adequately document each viral infection, it would have taken four man-years of effort to document a problem only two years old by 1989. We further assume McAfee's statements include every floppy disk ever infected up to that time by a virus, as well as all of the computers participating in the Christmas and InterNet worm attacks. (Worms cannot be included in virus infection statistics.) McAfee prefers to "estimate" his totals these days. Let's assume we have about 100 million computers of all types & models in use around the world. McAfee's estimate means 1 out of every 20 computers on the planet supposedly has a virus. It sounds like a pretty astronomical number to most other virus experts.

"Viruses can hide inside a data file."

Data files can't wreak havoc on your computer -- only an executable program file can do that (including the one that runs when you first turn on your computer). If a virus infected a data file, it would be a wasted effort. But let's be realistic: what you think is 'data' may actually be an executable program file. For example, a "batch file" qualifies as text on an IBM PC, yet the MS-DOS operating system treats it just like a program.

"BBSs and shareware programs spread viruses."

Here's another scary myth drummed up in the big virus panic, this one spouted as gospel by many "experts" who claim to know how viruses spread. "The truth," says PC Magazine publisher Bill Machrone, "is that all major viruses to date were transmitted by [retail] packages and private mail systems, often in universities." (PC Magazine, October 11, 1988.) Machrone said this back in 1988 and it still applies to this day. Almost 50 retail companies so far have admitted spreading infected master disks to tens of thousands of customers since 1988 -- compared to only five shareware authors who have spread viruses on master disks to less than 100 customers. Machrone goes on to say "bulletin boards and shareware authors work extraordinarily hard at policing themselves to keep viruses out." Reputable sysops check every file for Trojan horses; nationwide sysop networks help spread the word about dangerous files. Yes, you should beware of the software you get from BBSs and shareware authors, but you should also beware of the retail software you find on store shelves. (By the way, many stores now have software return policies. Do you know for sure you were the only one who used those master disks?)

"My computer could be infected if I call an infected BBS."

BBSs can't write information on your disks -- the communications software you use performs this task. You can only transfer a dangerous file to your computer if you let your software do it. And there is no "300bps subcarrier" that lets a virus slip through a high speed modem. A joker named Mike RoChenle (IBM's "micro channel" PS/2 architecture, get it?) started the 300bps myth when he left a techy-joke message on a public BBS. Unfortunately, a few highly respected journalists were taken in by the joke.

"So-called 'boot sector' viruses travel primarily in software downloaded from BBSs."

This common myth -- touted as gospel even by Australia's Computer Virus Information Group -- expounds on the mythical role computer bulletin boards play in spreading viruses. Boot sector viruses can only spread by direct contact and "booting" the computer from an infected disk. BBSs deal exclusively in program files and have no need to pass along copies of disk boot sectors. Bulletin board users therefore have a natural immunity to bootsector viruses when they download software. We should make a special note about "dropper" programs developed by virus researchers as an easy way to transfer boot sector viruses among themselves. Since they don't replicate, "dropper" programs don't qualify as a virus in and of themselves. Such programs have never been discovered on any BBS to date and have no real use other than to transfer infected boot sectors.

"My files are damaged, so it must have been a virus attack."

It also could have happened because of a power flux, or static electricity, or a fingerprint on a floppy disk, or a bug in your software, or perhaps a simple error on your part.Power failures and spilled cups of coffee have destroyed more data than all viruses combined.

"Donald Burleson was convicted of releasing a virus."

Newspapers all over the country hailed a Texas computer crime trial as a "virus" trial. The defendent, Donald Burleson, was in a position to release a destructive Trojan horse on his employer's mainframe computer. This particular software

couldn't spread to other computers, so it couldn't possibly have qualified as a virus. Davis McCown, the prosecuting attorney, claims he "never brought up the word virus" during the trial. So why did the media call it one?

1. David Kinney, an expert witness testifying for the defense, claimed Burleson had unleashed a virus. The prosecuting attorney didn't argue the point and we don't blame him -- Kinney's bizarre claim probably helped sway the jury to convict Burleson, and it was the defense's fault for letting him testify.

2. McCown gave reporters the facts behind the case and let them come up with their own definitions. The Associated Press and USA Today, among others, used such vague definitions that any program would have qualified as a virus. If we applied their definitions to the medical world, we could safely label penicillin as a biological virus (which is, of course, absurd).

3. McCown claims many quotes attributed to him were "misleading or fabricated" and identified one in particular which "is total fiction." Reporters sometimes print a quote out of context, and McCown apparently fell victim to it. (It's possible a few bizarre quotes from David Kinney or John McAfee were accidentally attributed to McCown.)

"Robert Morris Jr. released a benign virus on a defense network."

It may have been benign but it wasn't a virus. Morris, the son of a chief computer scientist at the U.S. National Security Agency, decided one day to take advantage of a bug in the Defense Department's networking software. This tiny bug let him send a worm through the network. Among other things, Morris's "InterNet" worm sent copies of itself to other computers in the network. Unfortunately, the network clogged up in a matter of hours due to some bugs in the worm module itself. The press originally called it a "virus," like it called the Christmas worm a virus, because it spread to other computers. Yet Morris's programs didn't infect any computers. A few notes:

1. Reporters finally started calling it a worm a year after the fact, but only because lawyers in the case constantly referred to it as a worm.

2. The worm operated only on Sun-3 & Vax computers which employ a UNIX operating system and were specifically linked into the InterNet network at the time.

3. The 6,200 affected computers cannot be counted in virus infection statistics (since they weren't infected).

4. It cost way less than $98 million to clean up the attack. An official Cornell University report claims John McAfee, the man behind this wild estimate, "was probably serving [him]self" in an effort to drum up business. People familiar with the case estimated the final figure at under $1 million.

5. Yes, Morris could easily have added some infection code to make it a worm/virus if he'd had the urge.

6. The network bug exploited in the attack has since been fixed.

7. Morris went to trial for launching the InterNet worm and received a federal conviction. The Supreme Court refused to hear the case, so his conviction stands.

"The U.S. government planted a virus in Iraq military computers during the Gulf War."

U.S. News & World Report published a story in early 1992 accusing the National Security Agency of replacing a computer chip in a printer bound for Iraq just before the Gulf War with a secret computer chip containing a virus. The magazine cited "two unidentified senior U.S. officials" as their source, saying "once the virus was in the [Iraqi computer] system, ...each time an Iraqi technician opened a 'window' on his computer screen to access information, the contents of the screen simply vanished." However, the USN&WR story shows amazing similarities to a 1991 April Fool's story published by InfoWorld magazine. Most computer experts dismiss the USN&WR story as a hoax -- an "urban legend" innocently created by the InfoWorld joke. Some notes:

1. USN&WR has refused to retract the story, but it did issue a "clarification" stating "it could not be confirmed that the [virus] was ultimately successful." The editors broke with tradition and refused to publish any of the numerous letters readers submitted about the virus story.

2. Ted Koppel, a well-known American news anchor, opened one of his "Nightline" broadcasts with a report on the alleged virus. Koppel's staff politely refers people to talk with USN&WR about the story's validity.

3. InfoWorld didn't label their story as fiction, but the last paragraph identified it as an April Fool's joke.

"Viruses can spread to all sorts of computers."

All Trojan horses are limited to a family of computers, and this is especially true for viruses. A virus designed to spread on IBM PCs cannot infect an IBM 4300 series mainframe, nor can it infect a Commodore C64, nor can it infect an Apple Macintosh.

"My backups will be worthless if I back up a virus."

No, they won't. Let's suppose a virus does get backed up with your files. You can restore important documents and databases -- your valuable data -- without restoring an infected program. You just reinstall programs from master disks. It's tedious work, but not as hard as some people claim.

"Antivirus software will protect me from viruses."

There is no such thing as a foolproof antivirus program. Trojan horses and viruses can be (and have been) designed to bypass them. Antivirus products themselves can be tricky to use at times, and they occasionally have bugs. Always use a good set of backups as your first line of defense; rely on antivirus software as a second line of defense.

"Read-only files are safe from virus infections."

This common myth among IBM PC users has been printed even in some computer magazines. Supposedly, you can protect yourself by using the DOS ATTRIB command to set the read-only attribute on program files. However, ATTRIB is software -- and what it can do, a virus can undo. The ATTRIB command seldom halts the spread of viruses.

"Viruses can infect files on write-protected disks."

Here's another common IBM PC myth. If viruses can modify read-only files, people assume they can modify write-protected floppies. However, the disk drive itself knows when a floppy is protected and refuses to write to it. You can physically disable an IBM PC drive's write-protect sensor, but you can't override it with a software command.

We hope this dispels the many computer virus myths. Viruses DO exist, they ARE out there, they WANT to spread to other computers, and they CAN cause you problems. But you can defend yourself with a cool head and a good set of backups.

The following guidelines can shield you from Trojan horses and viruses. They will lower your chances of being infected and raise your chances of recovering from an attack.

1. Implement a procedure to regularly back up your files and follow it religiously. Consider purchasing a user-friendly program to take the drudgery out of this task. (There are plenty to choose from.)

2. Rotate between at least two sets of backups for better security (use set #1, then set #2, then set #1...). The more sets you use, the better protected you are. Many people take a "master" backup of their entire hard disk, then take "incremental" backups of those files which changed since the last time they backed up. Incremental backups might only require five minutes of your time each day.

3. Download files only from reputable BBSs where the sysop checks every program for Trojan horses. If you're still afraid, consider getting programs from a BBS or "disk vendor" company which gets them direct from the authors.

4. Let newly uploaded files "mature" on a BBS for one or two weeks before you download it (others will put it through its paces).

5. Consider using a program that searches, or "scans," disks for known viruses. Almost all infections to date involved viruses known to antivirus companies. A recent copy of any "scanning" program will in all probability identify a virus before it gets the chance to infect your computer -- and as they say, "an ounce of prevention is worth a pound of cure." A "scanning" program can dramatically lower your chances of getting infected by a computer virus in the first place. (But remember: there is no perfect antivirus defense.)

6. Consider using a program that creates a unique "signature" of all the programs on your computer. Run this program once in awhile to see if any of your software applications have been modified -- either by a virus or by a fingerprint on a floppy disk or perhaps even by a stray gamma ray.

7. DON'T PANIC if your computer starts acting weird. It may be a virus, but then again maybe not. Immediately turn off all power to your computer and disconnect it from any local area networks. Reboot from a write-protected copy of your master DOS disk. Do NOT run any programs on a "regular" disk (you might activate a Trojan horse). If you don't have adequate backups, try to bring them up to date. Yes, you might back up a virus as well, but it can't hurt you if you don't use your normal programs. Set your backups off to the side. Only then can you safely hunt for problems.

8. If you can't figure out what's wrong and you aren't sure what to do next, turn off your computer and call for help. Consider calling a local computer group before you call for an expert. If you need a professional, consider a regular computer consultant first. Some "virus removal experts" charge prices far beyond their actual value.

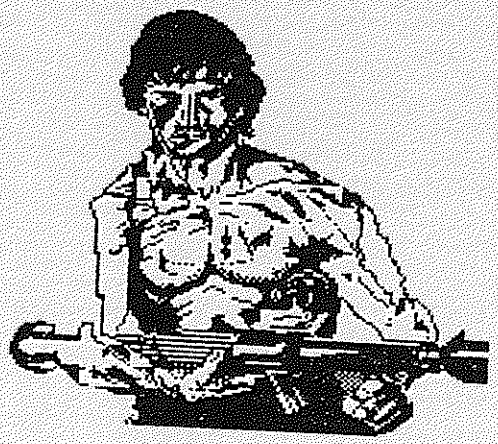
9. [Consider this ONLY as a last resort.] If you can't figure out what's wrong and you are sure of yourself, execute both a low-level and a high-level format on all your regular disks. Next, carefully reinstall all software from the master disks (not from the backups). Make sure the master disks have write-protect tabs! Then, carefully restore only the data files (not the program files) from your backup disks.

We'd appreciate it if you would mail us a copy of any Trojan horse or virus you discover. (Be careful you don't damage the data on your disks while trying to do this!) Include as much information as you can and put a label on the disk saying it contains a malicious program. Send it to Ross M.Greenberg, P.O. Box 908, Margaretville, NY 12254. Thank you.

Ross M. Greenberg is the author of both shareware and retail virus detection programs. Rob Rosenberger is the author of various phone productivity applications. (Products are not mentioned by name because this isn't the place for advertisements.) They each write for national computer magazines. These men communicated entirely by modem while writing this treatise.

 Rosenberger can be reached electronically on CompuServe as [74017,1344], on GEnie as R.ROSENBERGE, on InterNet as '74017.1344@compuserve.com', and on various national BBS linkups. Greenberg can be reached on MCI and BIX as 'greenber', on UseNet as 'c-rossgr@microsoft.com', and on CompuServe as [72461,3212].

END OF ARTICLE



REGIONAL GROUP REPORTS

Meeting Summary For APRIL

Central Coast	08/04/95 Saratoga
Glebe	06/04/95 Glebe
Hunter Valley	09/04 16/04/95
Illawarra	04/04/95 Keiraville
Liverpool	07/04/95 Yagoona West
Sutherland	21/04/95 Jannali

==========

CENTRAL COAST Regional Group

Regular meetings are normally held on the second Saturday of each month, 6.30pm at the home of John Coulton, 34 Mimosa Ave., Saratoga, (043) 69 3990. Contact Russell Welham (043)92 4000.

==========

GLEBE Regional Group

Regular meetings are normally on the Thursday evening following the first Saturday of the month, at 8pm at 43 Boyce Street, Glebe. Contact Mike Slattery, (02) 692 8162.

==========

HUNTER VALLEY Regional Group

The Meetings are usually held on the second or third Sunday of each month at members homes starting at 3pm. Check the location with Geoff Phillips by leaving a message on (049) 428 617. Please note that the previous phone number (049) 428 176 is now used exclusively by the ZZAP BBS which also has TI support. Geoff.

==========

ILLAWARRA Regional Group

Regular meetings are normally held on the first Tuesday of each month after the TIsHUG Sydney meeting at 7.30pm, at the home of Geoff Trott, 20 Robsons Road, Keiraville. A variety of investigations take place at our meetings, including Word Processing, Spreadsheets and hardware repairs. Contact Geoff Trott on (042) 29 6629 for more information.

==========

* LIVERPOOL Regional Group *

Regular meeting date is the Friday folling the TIshug Sydney meeting at 7.30 pm. Contact Larry Saunders (02) 644-7377 (home). After 9.30 PM or at work (02)602 3312 Liquorland Liverpool West for more information.

*** ALL WELCOME ***

7th April 1995

My Place : 34 Colechin st. Yagoona West

Bye for now Larry.

Liverpool Regional Co-Ordinator

==========

SUTHERLAND Regional Group

Regular meetings are held on the third Friday of each month at the home of Peter Young, 51 Jannali Avenue, Jannali at 7.30pm. Peter Young.

==========

TIsHUG in Sydney

Monthly meetings start promptly at 2pm on the first Saturday of the month. They are held at the MEADOWBANK PRIMARY SCHOOL, on the corner of Thistle Street and Belmore Street, Meadowbank. Cars can enter from Gale Street and park in the school grounds. Regular items include news from the directors, the publications library, the shop, and demonstrations of monthly software.

APRIL MEETING - 1st APRIL

MAY MEETING - 6th MAY

****

The cut-off dates for submitting articles to the Editor for the TND via the BBS or otherwise are:

MAY - 15th APRIL

These dates are all Saturdays and there is no guarantee that they will make the magazine unless they are uploaded by 6:00 pm, at the latest. Longer articles should be to hand well before the above dates to ensure there is time to edit them.

****

****

EDITORS COMMENTS

APRIL MEETING

The March meeting went with out a hitch, there was quite a bit of intrest in the products and programs been demenstrated, so much that it was decided to run the same again this month, for hose who couldnt make it and of course for those who want to have an extra fiddle.

One gentleman (that I know off) went away happy with his 80 column card, up and running, knowing that he has sorted out a few bugs.

For the IBM users there will be the CD ROM demo.

And the TI'ers there will be the gathering of the 80 column operators and if you have any operating problems with your 80 col. card come along, you might leave there all fixed up.

See you at the meeting.