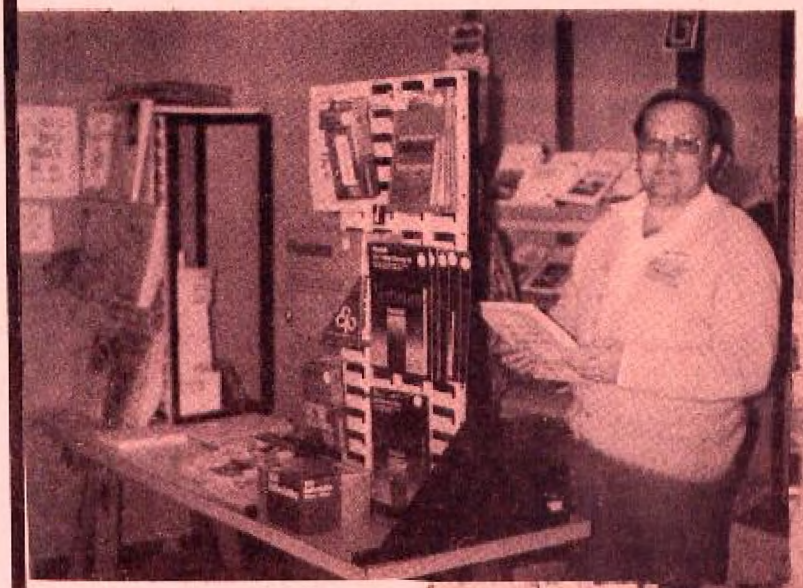
TIsHUG NEWS DIGEST

AUGUST'86

TIsHUG, PO Box 149, PENNANT HILLS, NSW Aust.2120 Registered by Australia Post Publication #NBH5933

Myarc's John Keown at the TI Faire

Software Co-Ordinator, Terry Phillips readies the stand in Melbourne in preparation for the big event.



Publib Co-ordinator, Brian Graham manning the TIsHUG stand in Melbourne

TIsHUG NEWS DIGEST
Vol.5 No.5 June'86

The official Newsletter of the T.I.'s Homecomputer Users' Group (Australia).

Editor and Layout designer:
SHANE K. ANDERSEN

Photographer:
MAURICE STEWARTSON

CONTRIBUTORS:

IN ORDER OF ARTICLES

FRED MORRIS
BRIAN GRAHAM
ROLF SCHREIBER
E.H.REITINGER
CHICK De MARTI
JIM PETERSON
GEOFF TROTT
ROSS MUDIE
SAM MUDIE
PETER MUDIE
TERRY PHILIPS
BEN TAKACH
ANDY COOPER
JENNY
ROBERT PEVERILL
SCOTT DARLING
J.M.VINCENT
RICHARD STANFORD

Published by:
TIsHUG - Australia

Printed by:
APPLE PRINT & COPY CENTRES
160 Castlereagh St,Sydney.2000
(02)264 8111

Club Address:
P.O.Box 149,
Pennant Hills,
N.S.W. 2120

Monthly Meeting Place:
WOODSTOCK COMMUNITY CENTRE
Church Street, BURWOOD.
First Saturday of each month
except January Public Holidays

Staring at 2pm except on FULL DAY TUTORIAL/WORKSHOPS.

Co-Ordinator:
FRED MORRIS . . . (02)871-3873
Secretary:
JOHN ROBINSON . . (02)848-0956

TISHUG NEWS DIGEST

Up Front

IN THIS ISSUE, you'll find a host of different items to add that variety which I hope you will enjoy. There are programs in Extended Basic, Basic, Assembler, Forth plus tips and tricks on both software and hardware programming. In this particular issue, however, there is a strong overseas content, because of the lack of articles from our TIsHUG members. Come on gang, lets hear from you. Even our Younger Set members have slowed down to a complete halt.

Here now is the listing of all those goodies we have for you this month...

Well, have a good month of fun with your 99/4(A) Computer, and we'll see you at the coming meeting (see FACE TO FACE for details of Meeting hall address).

Face to Face

CO-ORDINATOR REPORT
AUGUST 1986

Winter is finally with us! Does this mean that all of our programmers are in doors, keeping warm, and busily writing all of those programs which were planned but never completed? Time will tell!

It's now a month since TI-FAIR was held and the visit of John Keown, Director of Myarc Inc., to our group. Interestingly, more members of TIsHUG responded to the quick 'phone around extending the invitation to meet John and view his company products than was the paid attendance at TI-Fair! If this then is the measure of interest and enthusiasm of TIsHUG members I am certain that we still have far to go. Thank you all for this.

While I am throwing bouquets around (something which I don't normally do) maybe I should also congratulate the members of the Illawarra Regional Group for their consistent and valuable contribution to TIsHUG. For the past few months we have enjoyed some pretty good "stuff" received and published in this magazine. They have set the pace - which Regional Group is game enough to take them on? Let's see. Congratulations Illawarra!

On the subject of Regional Groups, this will be of interest to Country members, it has become clear that a tremendous opportunity for TIsHUG service is needed by our members living outside of the Sydney Metropolitan area. A survey of member addresses indicates an almost even division between Town and Country. As the distances between towns, for country members, is so vast it is not practical for Regional Home Groups to be formed. So, what do our country members do if they need assistance and advice? Well, the mailbag content says it all! The vast majority of incoming mail and telephone calls are received from Country members. (hopefully we do get to answer all queries and calls for help - eventually!)

Your committee debated this issue and and decided, in the interests of improved communication within TIsHUG, to form the TIsHUG "COUNTRY CLUB" Regional Group. All non-metropolitan members of TIsHUG, not belonging to another RHG, are automatically members of the "COUNTRY CLUB" which will be co-ordinated from Sydney. The idea is to provide a reference/communication point for all country members in their dealings with their TI.99/4A matters. Fortunately, Brian Graham has made himself available to act as the Co-Ordinator of this group. As you already know, Brian is also the Publications Librarian plus he has been instrumental is producing the Articles of Association needed for the Incorporation of TIsHUG as a registered company. (Legal Beagle - as he is known!)

To contact Brian, please address all correspondence to COUNTRY CLUB RG, P.O. BOX 149, PENNANT HILLS 2120. NSW. or TEL. (02) 774-3223. In due course Brian will expand further on how he sees this group operating. Brian, thank you for volunteering your services - I am certain that the need for this group does exist and that your assistance here will further enhance the TIsHUG reputation of SERVICE to members. Over to you!

Another little success story comes from both Keith de Haan and Peter Young. Remember they offered to Co-Ordinate the North Shore and Sutherland Regional Groups? Well, they seem to be getting thing organised - I believe we will see more coming in from them. Thanks chaps.

To sum up - while we do admit to falling membership, there is still more than enough on going commitment and development around us which ensures, for those members who choose to stay with the TI.99/4A, continued satisfaction from their hobby. There is still so much ground still to be covered in SOFTWARE DEVELOPMENT - hopefully, we, of TIsHUG, will be making some positive contributions in this area. So, for the TIsHUG PROGRAMMERS, I recommend that they stay indoors this winter - EXPECTING them to emerge from hibernation with some software goodies to keep us going!

Regards to you all,

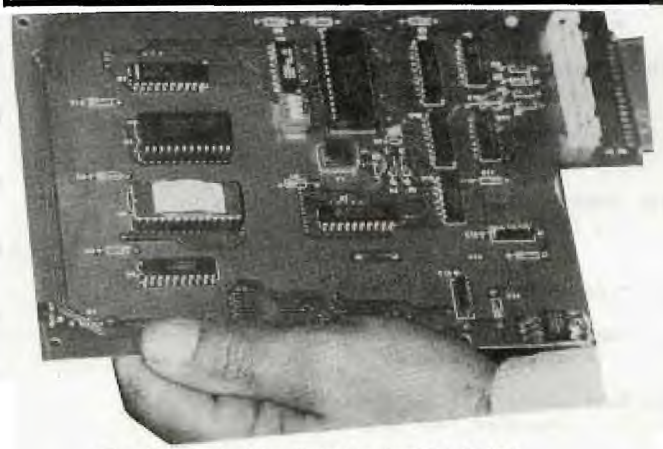
Fred Morris.

PS. I hear that Michael Slattery is busy with an X/B program that generates ASSEMBLY SOURCE code - Keep going Michael!

...ew meeting venue - please make a note of this as the venue can be a little difficult to locate. The address is Shirley House, Ethel Street Burwood, Gregory's reference Map 30 (H8) - one street down from Church St.

August Meeting - it is hoped to have some special guests at this meeting. The big auction day will be carried over to the October meeting. Don't forget that the September meeting will be the big full day technical workshop.

An interested group of members oberserving the Myarc demonstration at the hastily convened special meeting.



The Myarc Double Density Double Sided Disk Controller Card

TI - PUBLIB REPORT.

WELCOME to my first column since taking charge of the Publications Library from our overworked Co-Ordinator and I hope that I can continue to provide that same standard of service to ALL club members. I have had the Publications Library only a few days and would like to gain some feedback from members as to the type of books and magazines YOU would like to see provided by the Publications Library for the TI 99/4A and its family. The first task I see is the provision of an up to date list of Library Publications held by the Club for those like me who did not know what's available . Well, no sooner said than done. However, don't forget that to borrow from the Publications Library you must join the service but this will ONLY cost $ 5.00 per year.

Available to TIsHUG (Australia) Ltd. Publication Library members ONLY.

***** AUSTRALIAN MAGAZINES *****

(1) ADELAIDE T.I. COMPUTER CLUB (A.T.T.I.C)
February 1983 to January 1986

(2) BRISBANE T.I. USERS GROUP (TIBUG)
August 1982 to October 1985

(3) CANBERRA T.I. USERS GROUP (CHUG*A*LUG)
November 1982 to October 1985

(4) HUNTER VALLEY T.I. USERS GROUP (HV 99er's)
Newsletters 1, 2, 3, 4, 5, 6, 7, and 9.

(5) ILLAWARRA T.I. USERS GROUP
February 1985 to February 1986

(6) MELBOURNE T.I. USERS GROUP (TIMES)
February 1982 to October 1985

(7) PERTH T.I. USERS GROUP (TITBITS)
May 1982 to November 1985

(8) TASMANIAN T.I. USERS GROUP
November 1983 to February 1986

(9) SOFTEX MAGAZINE
November 1983 to March 1985

(10) SYDNEY T.I. USERS GROUP (SND & TND)
May 1985 to May 1986 .....##

***** CANADIAN MAGAZINES *****

(1) CHANNEL 99 HAMILTON (APPLICATION)
August 1984 only

(2) NOVA SCOTIA T.I. USERS GROUP (TINS)
November 1984 only

(3) SASKATOON T.I. COMPUTER CLUB (S.T.I.C.C.)
January 1985 to September 1985

***** UNITED KINGDOM MAGAZINES *****

(1) TI HOME (TIdings)
February 1981 to August 1982

(2) TI HOME (Quarterly News)
Autumn 1983 to Summer 1985

***** UNITED STATES OF AMERICA MAGAZINES *****

(1) CENTRAL OHIO USERS GROUP (Spirit of 99)
January 1983 to August 1985 .....##

(2) CIN - DAY USERS GROUP (Cincinnatti - Daytona)
August 1981 to September 1985

(3) HOUSTON USERS GROUP (H.U.G.)
January 1982 to April 1986 .....##

(4) LEHIGH 99 ' ER USERS GROUP (Pennsylvania)
June 1984 to June 1985

(5) MID ATLANTIC GROUP (Manners Newsletter)
June 1983 to November 1985 .....##

(6) MINNESOTA USERS GROUP (MSP 99)
September 1982 to November 1985

(7) NORTHWEST OHIO USERS GROUP
(OH - MI - TI & New Horizons)
August 1983 to February 1986 .....##

(8) ROCKY MOUNTAIN 99'ERS USERS GROUP (TIC TALK)
April 1983 to May 1985 .....##

(9) SACRAMENTO USERS GROUP (Network)
June 1985 to January 1986

(10) SAN GABRIEL VALLEY USERS GROUP (Printout)
March 1983 to September 1984

(11) SMART PROGRAMMER (Miller Graphics)
March 1984 to September 1984

(12) SOUTHERN CALIFORNIA USERS GROUP
(The Computer Voice)

(13) TACOMA EASTSIDE 99ER'S GROUP
November 1984 to January 1986

(14) TIGERCUB SOFTWARE GROUP
(Jim Peterson)
Various Tips, Hints, Etc.

***** OTHER OVERSEAS MAGAZINES *****

(1) 4A TODAY
September 1984 only

(2) INTERNATIONAL 99/4 USERS GROUP
February 1981 to August 1983 .....##

(3) 99/4 USERS OF AMERICA
November 1982 to July 1983 .....##

(4) BAKERSFIELD USERS GROUP (TEX - BUG)
September 1983 to February 1984

(5) CORCOMP CURSOR
November 1984 to August 1985 .....##

(6) DAYTONA BEACH COMPUTER CLUB (Daytona 99'ers)
June 1985 only

(7) JACKSONVILLE USERS GROUP (J.U.G.S.)
September and October 1985 only

(8) KENTUCKIANA 99/4 COMPUTER SOCIETY
November 1982 to May 1983 .....##

(9) ERIE 99'ER USERS GROUP
December 1985 only

(10) LOS ANGELES - LA 99ers USERS GROUP (TopIcs)
January 1984 to April 1984 .....##

(11) LUBBOCK COMPUTER CLUB (Computerbase)
January 1983 to June 1983 .....##

(12) LUV - TRONICS USERS GROUP
September 1983 only

(13) MENOMEE FALLS USERS GROUP
March 1985 only

(14) MASS USERS & COMPUTER HOBBYISTS (M.U.N.C.H.)
March 1984 only

(15) NUTMEG T.I. 99ers
November 1985 only

(16) SAN DIEGO COMPUTER SOCIETY (TI - SIG)
March 1985 only

(17) SOUTHWEST NINETY - NINERS
August 1985 only

(18) TEXAS INSTRUMENTS COMPUTER DIVISION
Official Users Group Newsletter
August 1980 to April 1982 .....##

(19) YOUNG PEOPLES LOGO ASSOCIATION (Turtle News)
June 1982 only

(20) UNOFFICIAL 99/4 [A]
May 1983 only

(21) WESTERN REGION USERS GROUP
January 1984 only

(22) TIjdingen (from Holland in Dutch)
July 1983 only

.....## Denotes Incomplete Series

The best by far current T.I.99/4A applications magazine held in the Publications Library is MICROPENDIUM. Copies are available each month (shipping permitting) for purchase from the Club's shop. The Publications Library is extremely fortunate to have a complete set of this magazine containing a wealth of information. So, if you have relegated your T.I. to the attic or the basement or the closet ..... NOW ..... is the time to get it out and take it for a fast workout in the company of your copies of MICROPENDIUM.

As time is fast running out and space limitations don't allow for further ponderance watch out next month for the next gripping instalment of the Publications Library listing. I must state now that it is impossible for the entire Publications Library to be readily transported to each meeting so if there is a particular publication you wish to peruse or borrow then contact me prior to the meeting by phoning (02) 774-3223 (evenings only) or write to me at the Club's address. I will ensure that you get the material at the earliest possible time.

Don't forget to join the Publication Library Borrowing Service ..... it ONLY costs $ 5.00 for the period ending in April of each year and by joining the service you are providing a useful tool for it to grow. If you know of any books or magazines that should be held in the Publications Library contact me and I will try to get the required publication for all to use and enjoy.
.....Libris Studius.

Console Writer 2.1

A Review by Rolf Schreiber
Illawarra regional group - TISHUG

Console Writer is, as the name implies, a word processing 'package' requiring only the TI99/4A console and a suitable cassette recorder. The 'package' consists of a plug-in ROM module and an 8 page manual.

Although Console Writer can be used with both disk and cassette based systems, it is aimed mainly at the cassette based user interested in word processing. By no stretch of the imagination could it be considered a viable alternative to TI-Writer, but it should prove vastly superior to XB word processing programs such as 'Tex Scribe'.

Console Writer does have some VERY useful features and, in my opinion, some glaring deficiencies. This article is not meant to be a full blown review, but will only mention the more prominent of the module's good and bad points.

BAD POINTS

1) Does not have a 'wrap-around' text mode. During input, words are truncated at column 80 and the input continues on the next line

2) The editing screen is only the standard 28 chars wide, and 3 'windows' are required to cover the 80 columns

3) A loud 'DONG' accompanies every prompt and many of the inputs. This feature may not annoy everyone, but I don't like it.

4) The buffer is limited to 115 lines of text (irrespective of the number of columns used) before it becomes necessary to SAVE (to either cassette or disk.) For long articles this would mean saving (or loading) several files each time you wanted to EDIT or PRINT that article.

5) The module will NOT work with 1983 V2.2 consoles.

6) Does NOT allow formatting output to a printer. This program uses a what you see is what you get approach, so you must screen format prior to printing.

7) Does NOT have the 'FIND STRING' and 'REPLACE STRING' features of TI Writer.

8) Each time you SAVE to cassette, the whole of the text buffer is SAVEd (all 9200 bytes!), even if you only typed in one word.

9) The only way to exit from the program is to turn off the computer!

GOOD POINTS

1) It is probably the most versatile word processing program for a cassette based user (with or without 32K memory expansion). N.B. this program DOES NOT require (or even make use of) the 32K memory expansion. The maximum possible storage is 9200 characters (115x80 bytes), or approximately 2 pages of text.

2) It allows loading and saving text to cassette as a 'memory image' file, which is similar to the way our computer stores BASIC programs, and is 10-30 times FASTER than using INT/FIX 64 or INT/FIX 192 data files.

3) It is possible to verify the data after it has been SAVEd to cassette in the same way that programe can be verified after being SAVEd.

4) The program is instantly available once the cartridge is plugged in and Console Writer is selected.

5) Becoming familiar with the program only requires about a 10 minute reading of the brief, but sufficient, instruction manual.

6) Saving text to disk is also possible. The file structure is compatable with TI-Writer, so that a Console Writer text file saved to cassette can be later loaded back in and saved to disk. The resultant disk file can then be EDITed and FORMATted with TI Writer. Converting a DIS/VAR 80 disk file to a cassette file is also possible. This facilitates the transfer of text or data between cassette and disk.

In my opinion, for a cassette-based user wanting to do word processing, the advantages far outweigh the disadv[illegible], GO FOR IT!!!

TYPE IT!

Here's some great little routines, easy to type in, that you will be impressed with. All came via Jim Peterson, although other author credit is shown where applicable. Hope you like them. The first one is COLOR EDITOR and shows how to mix colors on the screen. It requires Extended Basic. The program DISK MEMORY will quickly show you how much space remains on a disk. The program SLANT will give you a new perspective on your computer. Probably the most interesting one is SLASHER which when run will turn your screen dark blue with white lettering and put a / through all zeros. This program remains in memory and can only be stopped by typing BYE. It requires 32K expansion. By the way Jim sent dozens of these small programs and if you like this sample arrangements could be made to publish more here for you to type in or they could be released on disk through the club shop.

```
1 !     COLOR EDITOR
   for mixing any desired
   2 colors with joyst or
        keyboard.
2 ! With Greetings

       E.H. REITINGER
       Vienna, Austria
3 ! TI99-Journal-Klub
       A-1150 Wein
    Felberstrabe 24/26
 (published in The Smart Pro
grammer May 1984)
4 ! Use E and X keys to move
 mouse to desired color, the
n any key on left side to ch
ange foreground to that colo
r; press again to change the
5 ! background to the same.
10 CALL SCREEN(16):: CALL CL
EAR
20 M$="55AA55AA55AA55AA" ::
A=122
30 CALL MAGNIFY(2):: CALL CH
AR(64,RPT$("F",16),34,"FF181
8FFFFFF",128,"FFFFFFFFFFFFFF
",73,M$):: CALL COLOR(3,16,2
,4,16,2,6,1,1,5,2,1)
40 CALL VCHAR(1,27,64,192)::
 CALL HCHAR(23,1,64,162):: H
=1
50 G=-2 :: FOR I=3 TO 16 ::
CALL SPRITE(#I,64,I,(G+I)*12
,230):: NEXT I :: CALL SPRIT
E(#2,34,16,5,230)
60 CALL SPRITE(#1,42,2,A,231
)
70 FOR S=4 TO 22 :: CALL HCH
AR(S,3,73,24):: NEXT S
80 CALL JOYST(1,X,Y):: ON (S
GN(Y)+2)GOTO 90,130,110
90 A=A+12 :: IF A>170 THEN A
=2
100 CALL LOCATE(#1,A,231)::
GOTO 130
110 A=A-12 :: IF A<0 THEN A=
170
120 CALL LOCATE(#1,A,231)
130 CALL KEY(1,K,S):: IF S=0
 THEN 80
140 IF K=5 THEN 110 :: IF K+
1=1 THEN 90
150 F=INT(A/12+2):: CALL SOU
ND(200,660,2):: GOTO 180
180 CALL COLOR(6,F,H):: DISP
LAY AT(24,9)SIZE(7):USING "
## ## ":F,H :: H=F :: GOTO 8
0
```

```
    turns screen blue with w
hite characters, and slashes
 zeros, until you return to
the title screen
100 CALL CLEAR :: DISPLAY AT
(11,5):"ONE MOMENT PLEASE..
" :: CALL PEEK(8198,A):: IF
A<>170 THEN CALL INIT
110 FOR A=1 TO 126 :: READ B
 :: C=C+B :: NEXT A :: IF C<
>10146 THEN PRINT "DATA ERRO
R! PLEASE CHECK." :: STOP
120 RESTORE :: CALL PEEK(819
4,A,B):: C=A*256+B :: CALL P
EEK(8196,D,E):: F=D*256+E ::
 IF F-C<126 THEN PRINT "NOT
ENOUGH MEMORY TO LOAD!" :: E
ND
130 FOR D=0 TO 125 :: READ E
 :: IF E>255 THEN GOSUB 330
:: GOTO 150
140 CALL LOAD(D+C,E)
150 NEXT D
160 B=B+106 :: IF B>255 THEN
 B=B-256 :: A=A+1
170 CALL LOAD(8194,A,B)
180 CALL PEEK(8196,A,B):: B=
B-24 :: IF B<0 THEN B=B+256
:: A=A-1
190 CALL LOAD(8196,A,B):: F=
C :: C=A*256+B
200 FOR D=0 TO 19 :: CALL PE
EK(F+106+D,G):: CALL LOAD(C+
D,G):: NEXT D
210 CALL PEEK(-31804,C,D)::
CALL LOAD(F+12,C,D):: CALL L
OAD(-31804,A,B):: END
220 ! DATA FOR MAIN PROGRAM
230 DATA 244,0,0,58,68,76,84
,100,68,184,0,60,0,0,152
240 DATA 32,256,1,131,68,22,
37,2,1,0,135,208,96,256,0
250 DATA 216,1,140,2,6,193,2
16,1,140,2,2,1,0,72,216
260 DATA 1,140,2,6,193,216,1
,140,2,2,0,0,32,216,32
270 DATA 256,0,140,0,6,0,22,
251,2,1,128,68,216,1,140
280 DATA 2,6,193,216,1,140,2
,2,1,256,2,216,49,140,0
290 DATA 2,129,256,10,22,251
,192,32,256,12,19,1,4,80,4,9
1
300 ! DATA FOR INTERRUPT SER
VICE
310 DATA 192,32,32,2,2,128,2
56,106,26,2,4,96,256,14,4
320 DATA 224,131,196,4,91
330 READ G :: E=E-256+A :: G
=G+B :: IF G>255 THEN G=G-25
6 :: E=E+1
340 CALL LOAD(D+C,E,G):: D=D
+1 :: RETURN
```

```
100 REM
110 REM DISK MEMORY AVAILABL
E
120 REM BY CHICK DE MARTI 19
83
130 REM IN LA 99ers Computer
 Group TOPICS Newsletter Dec
. 1983
140 CALL CLEAR
145 PRINT "SHOWS DISK NAME,
SECTORS":"USED AND SECTORS A
VAILABLE.": : : :
150 PRINT "PRESS <S> SCREEN
ONLY": :
160 PRINT "       <P> COPY AL
SO": :
170 PRINT "    <ENTER> TO EX
IT"
180 FOR ROLL=1 TO 6
190 PRINT
200 NEXT ROLL
210 GOSUB 380
220 OPEN #1:"DSK1.",INPUT ,R
ELATIVE,INTERNAL
230 INPUT #1:A$,J,J,K
240 IF AN$="P" THEN 250 ELSE
 270
250 OPEN #2:"PIO",OUTPUT
260 PRINT #2:" - DISKNAME=";
A$;"AVAILABLE=";K;" USED=";J
-K
270 DISPLAY "DISKNAME - ";A$
:"AVAILABLE=";K;" USED=";J-K
280 PRINT
290 PRINT "------ENTER NEXT
DISK------"
300 PRINT
310 IF AN$="P" THEN 320 ELSE
 330
320 CLOSE #2
330 CLOSE #1
340 AN$="NUL"
350 GOTO 210
360 CALL CLEAR
370 END
380 REM CHOICE FROM MENU
390 CALL KEY(3,A,S)
400 IF S=0 THEN 390
410 IF A=13 THEN 360
420 IF A=83 THEN 440
425 IF A=80 THEN 430 ELSE 39
0
430 AN$="P"
440 RETURN
```

OF FLAGS AND STATUS BITS

by Geoff Trott
Illawarra regional group - TIsHUG

Some of the hardest things for assembler language programmers to come to terms with are the status bits or flags. These are bits which are set or cleared as the result of the last operation performed. This means that a decision can then be made on that result using one of the Jump on a condition instruction. Let us examine each of these flags to find out what information each one gives to us about particular instructions.

EQ or Equal flag. This is the easiest one to understand as it is set if the result of an arithmetic (A AB AI ABS DEC DECT NEG S SB INC INCT) or logical (ANDI ORI INV SOC SOCB SZC SZCB XOR) or shift (SLA SRA SRC SRL) or move (LI MOV MOVB LDCR STCR) instruction is zero. If the instruction is a word instruction, then all 16 bits must be 0. If a byte instruction, then just the 8 bits addressed must be 0. In comparison instructions (C CB CI), the flag is set if the two pieces of data being compared are the same. For the Compare Ones Corresponding (COC) and Compare Zeros Corresponding (CZC) instructions, the EQ flag will set if the bits specified by the source data are all 1's (COC) or all 0's (CZC). For the CRU instruction Test Bit (TB), the value of the bit selected is put into the EQ flag. The EQ flag is set if the bit is 1 (not zero), which may seem strange.

C or Carry flag. The C flag indicates that the operation has required one more bit than is available for the instruction (i.e. 9 or 17 bits). The C flag is only used for arithmetic and shifting instructions. In the shifting instructions (SLA SRA SRC SRL), the C flag is used to hold the last bit shifted out of the end of the data. When using unsigned numbers in addition operations it is easy to understand that if the result is greater than 65535 (for words) or 255 (for bytes) then the C flag will set. Subtraction involves the use of negative numbers, as it is performed by negation and addition. This means that we need to look at the representation of negative numbers before the C flag will make sense for subtraction.

Negative numbers are represented in 2's complement code by the processor. The essence of this is that the most significant bit is the sign bit of the number and if it is 0 the number is positive, and if it is 1 the number is negative. To find the negative of a number, complement all bits and add 1 to the result. Complement means to change all 1s to 0s and all 0s to 1s. To find the value of a negative number, negate it as above and then convert it to decimal. For example in bytes we have :

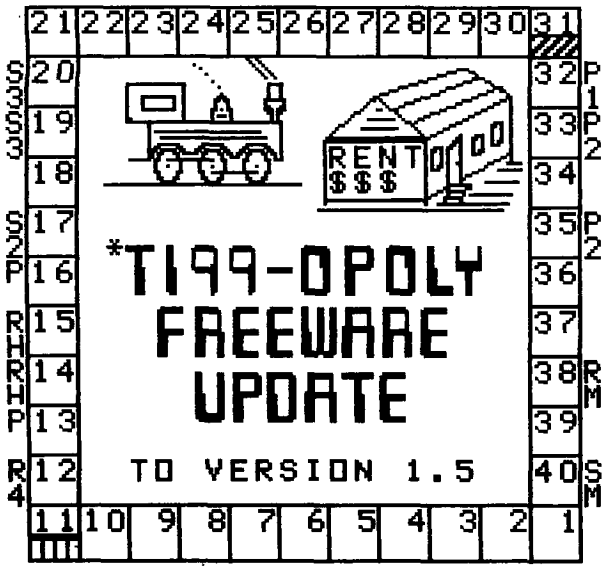
```
  +1 = 00000001 = >01
  -1 = 11111111 = >FF
+127 = 01111111 = >7F
-127 = 10000001 = >81
 +16 = 00010000 = >10
 -16 = 11110000 = >F0
```

There is one odd number in this code, and that is the smallest negative number, -128 (bytes) and -32768 (words). The negative of these gives the same codes back again. If 1 is subtracted from these two numbers, the largest positve numbers are obtained, 127 (bytes) and 32767 (words) and if 1 is added to these largest positive numbers, the smallest negative numbers are obtained again. The relationship between negative numbers and unsigned numbers is that the negative numbers are all unsigned numbers larger than the largest positive number. Let us now return to the carry flag.

For signed numbers and addition (A AB AI), at least one operand must be negative to set the C flag. If both operands are negative the C flag will set, or if the result of adding a positive to a negative number is a positive number, the C flag will set. For subtraction (S SB) similar conditions hold. The C flag will set if we subtract a positive number from a negative number (i.e. add two negative numbers). It will also set if there is a positive result after subtracting numbers which are either both positive or both negative (i.e. adding a positive and a negative number with positive result). For unsigned numbers, the C flag will set if a smaller number is subtracted from a larger number (including zero result). Increment (INC INCT) is the same as adding 1, and the C flag will set when the result changes from a negative number to a positive number. Decrement (DEC DECT) involves adding -1 (or -2) and the C flag will always set except when the value changes from positive to negative. The C flag will set if 0 is negated (NEG). The C flag is always cleared by an absolute value instruction (ABS).

OV or Overflow flag. The OV flag only gives information about signed numbers. In effect it is a carry for the number excluding the sign bit. If two numbers of like sign are added (A AB AI) to give a result which is the opposite sign, the OV flag will set. Similarly, if one subtracts (S SB) numbers of unlike sign and the result is not the same as the first number the OV flag will set. The OV flag sets if you try to negate (NEG) or take the absolute value (ABS) of the minimum negative number (see above), and when you increment (INC INCT) past the largest positive number or decrement (DEC DECT) past the smallest negative number. It also sets to indicate that a division (DIV) cannot take place because the result would need more than 16 bits. If the sign of the number changes while doing a Shift Left Arithmetic (SLA) the OV flag will set.

A> or Aritmetic greater than flag. This flag indicates that the result of an arithmetic (A AB AI ABS DEC DECT NEG S SB INC INCT) or logical (ANDI ORI INV SOC SOCB SZC SZCB XOR) or move (LDCR LI MOV MOVB STCR) or shift (SLA SRA SRC SRL) operation is positive and not zero. For comparison instructions (C CB CI), it sets if the first operand is greater than the second when both are interpreted as signed numbers.



FREEWARE VERSION of TI99-OPOLY.
from Ross Mudie of TIsHUG

The freeware version of TI99-OPOLY has been raised from version 1.4 to version 1.5 to overcome a bug which charged rent for an already mortgaged station when a player is advanced to the station by a Community Chest card "advance to the nearest railway". This card charges twice the normal rent if owned or permits the purchase of the property if unowned.

To raise a copy of the program disk to V1.5 first make a backup copy of the disk in case of mishap.

Using a Disk Manager, remove write protection from the programs named LOAD and TI99-OPOLY.

Go to extended basic with no disk in drive 1. When * READY * and the cursor prompt is visible, place the TI99-OPOLY disk in drive 1 and type OLD DSK1.LOAD .

Modify the LOAD program, when in memory, in lines 120 and 1090 as follows:

```
120 ! TI99-OPOLY V1.5 LOAD
8th June 1986, Ross Mudie

1090 DISPLAY AT(1,5):"LOADIN
G AND RUNNING": :"TI99-OPOLY
 V1.5 - Ross Mudie"
```

Resave the LOAD program by typing SAVE DSK1.LOAD .

Place the TI99-OPOLY program in memory by typing: OLD DSK1.TI99-OPOLY .

Modify the TI99-OPOLY program in line 100 and add line 2045 as follows:

```
100 OPTION BASE 1 :: ON WARN
ING NEXT :: ON BREAK NEXT !
TI99-OPOLY V1.5  Ross Mudie
8th June 1986

2045 IF D(PSN(P),2)=6 THEN C
ALL D(14,B$(123)):: CALL D(1
5,B$(124)):: GOTO 2170
```

Do NOT resequence the program.
Resave the TI99-OPOLY program to disk by typing: SAVE DSK1.TI99-OPOLY .

Return to the Disk Manager and re-apply write protection to both programs.

UPDATING AN OLD VERSION OF TI99-OPOLY.
Any person who owns an original copy of the pre-Freeware version of TI99-OPOLY may return the original disk and booklet, forward postage paid. The disk and booklet will be updated to the latest freeware version at no further cost and the return postage will be paid by the author.

After enclosing the return disk and booklet in adequate packaging and including YOUR OWN NAME AND ADDRESS, post to: Ross Mudie,
47 Berowra Waters Rd,
Berowra. N.S.W. 2081.
AUSTRALIA.

USER PROBLEMS.
I have had some enquiries of how to exit a game without turning the computer off. ...In the design of the program I did everything possible to prevent accidental loss of a running game. The best method that I can suggest is to save the game under a file name for unwanted games, e.g., DSK1.UNWANTED . A new game may then be started with previously saved data or a totally new game may be commenced or the game may be ended.

L> or Logical greater than flag. This flag indicates that the result of an arithmetic (A AB AI ABS DEC DECT NEG S SB INC INCT) or logical (ANDI ORI INV SOC SOCB SZC SZCB XOR) or move (LDCR LI MOV MOVB STCR) or shift (SLA SRA SRC SRL) operation is not zero. For these instructions it is just the complement of the EQ flag. For comparison instructions (C CB CI) however, it sets if the first operand is greater than the second when both are interpreted as unsigned numbers.

OP or Odd Parity flag. This flag is only used for byte instructions (AB CB MOVB SB SOCB SZCB), and for the CRU move bit instructions (LDCR STCR) if 8 or less bits are moved. The OP flag will be set if the result of the operations contains an odd number of 1's.

X or Extended operation flag. This flag is set when one of the 16 XOP instructions is excuted. The XOP enables an effective BLWP to be performed with an instruction of only one word. Since a subroutine entered with an XOP instruction could have been entered with an BLWP instruction, this flag allows the subroutine to find which one was used. The XOP instructions use fixed addresses to store the context switch information, starting at address >40. These addresses are in the System ROM of the 99/4A and so cannot be set up by us. In fact the later ROMs have the first 3 XOP addresses set up for use as follows. The first one (XOP Gs,0) causes a subroutine in what is obviously a TI debugging device to be called. The second one (XOP Gs,1) causes a subroutine at >FFF8 to be called using a workspace area starting at >FFD8. This is used by DEBUG for breakpoints. The last one (XOP Gs,2), causes a subroutine at >8300 (System RAM) to be called with a workspace area starting at >83A0. The addresses which would be used by the other XOP's (3 to 15) have programme code stored in them which makes them not easy to use for this purpose.

STATIC ZAP !!?

DANGER of STATIC ELECTRIC DISCHARGE to the COMPUTER.
by Ross Mudie of TISHUG.

This article is by no means a complete work on this complex subject. It is intended to provide a practical introduction to the subject of the winter zappies.

TI in their literature on the TI99/4A home computer warn against the destructive effect of static discharge to the computer.

A static charge in the order of tens of thousands of volts can build up on your body without you being aware of the condition. Many people have experienced this when getting out of a car and then touching, or even getting near the door handle.

The static electricity problem is most prevalent in the winter months when the humidity is low, but do not think that it is a winter only problem. If a heater is being used, by raising the temperature the humidity is lowered. Airconditioners actually remove moisture from the air, accentuating the problem.

The Static Charge can build up on a body, human or other, when there is friction, or even movement between two bodies which are separated by good electrical insulation. Low levels of humidity contribute to good electrical insulation and Sydney's westerly winter winds are an absolute menace.

When you walk around a room, especially on a nylon carpet wearing synthetic soled shoes or get up from a synthetic material or plastic covered chair, you are likely to be carrying a highly destructive static charge. If you discharge this static energy into a solid state module or your computer, static sensitive CMOS integrated circuits are likely to be destroyed and you may not even feel it until the repair bill comes.

SOME METHODS of CONTROLLING STATIC BUILDUP.

Antistatic sprays are available for carpets and synthetic cloth furniture. These sprays tend to trap and retain a low level of moisture which allows static charges to leak away harmlessly. These treatments must be repeated periodically to retain effectiveness.

The table top on which the computer is located can be covered by an antistatic mat to assist with the safe dissapation of static charges. The earthed mat should protrude in front of the keyboard to the edge of the table and either side of the console so that the operators hands will regularly contact the mat. If it is not feasible to provide a mat then even a conveniently positioned earth connection will suffice. If an earth connection is provided it should have a resistance of between 470Kohms and 1Megohm to earth. The earth connection may be obtained from the rear case of the expansion unit, the metal case of other earthed appliances or from an earthed water pipe. Anti static floor mats are also available to provide additional protection in areas of very high static risk. Floor mats should be earthed in a similar manner to the table mat.

Immediately before you touch a command module or the computer, momentarily touch the antistatic mat or earth point to drain away any static charge. If your chair has a metal frame then touch the frame whilst getting off the chair to equalise any possible static build up as it occurs. Avoid touching the TV screen, but if you must, then discharge the static charge which results from touching the screen as previously discussed. It is also good policy to turn the console off before inserting or removing a solid state cartridge.

The principle of the mat or resistive earth is to provide a high resistance path to earth giving a safe discharge current which is limited by the resistance of the mat and earth connection.

Never touch any connector area of the computer or solid state cartridges since a static discharge to a static sensitive connection will result in failure.

STATIC PRECAUTIONS if DOING TECHNICAL WORK.

If any technical work is contemplated, then an antistatic work place MUST be established. An antistatic mat must be placed on the work surface with a resistive earth connection provided to the mat. Any soldering iron tip must be connected to earth at the mat and the person carrying out the work must be connected to the mat via a resistive wrist strap. The earth of the computer must be in contact with the mat or an additional earth lead provided for this purpose.

WHERE to OBTAIN an ANTISTATIC MAT.

I have not done a full market survey but there are several suppliers of antistatic products including 3M, Circuit Components A'sia and Royston Electronics. The mat which I use is available from:
ROYSTON ELECTRONICS, 59 MOXON PDE, PUNCHBOWL.
PHONE (02)709 5099.
The part number is 1420ASK, price is $68.88, ex stock Melbourne. This kit contains a black, high carbon content mat, 460 x 610mm with two straps, one for the earth connection, the other for the wrist (when performing technical tasks or operating under extreme static conditions).

*** WARNING ***

Static electricity is one of the few things which works in total cooperation with Murphy's Law.

...or... One flash and the CMOS is ash.

Don't forget about the predominantly Summer static discharge problem, thunderstorms! See my article on power and telephone line filters on page 5 in the April 1986 issue of the SND.

service bench

by Geoff Trott
Illawarra Regional Group - TIsHUG

As time passes, some of our trusty consoles develop problems and need some repairs. I have seen a few of these now, and have been able to fix some of them up, while others are still being investigated. Here is the story of a few of them.

One of the first faults I had a look at concerned a keyboard where over vigorous use by several children at once had caused the alpha lock key to "explode" into its constituent parts. The owner had carefully collected all the pieces he could find, and even shaken some out of the console itself, without taking it apart. I must admit that when I first looked at the pieces I could not imagine how they all went together. There were two pieces of stiff wire, one straight and quite long, and the other bent into a particular shape. This one matched up with some parts of the plastic moulding and was responsible for the mechanical locking. Then there was a spiral spring and a little round plastic piece about the same diameter as the spring. By taking the cap off my own alpha lock key I could see that the spring went up the middle of the square plastic, and the bent wire had one arm which came up the outside of the plastic as expected. The straight wire was the means of locking the bent wire, and so the key itself into its holder and goes across the bent wire and is held by some shoulders in the plastic base. The plastic circle fits on the end of the spring and then is also on top of the bent wire. So I had all the pieces and it was only a matter of fitting the Chinese puzzle together. That was a bit fiddly and required that the long piece of wire was quite straight. One satisfied customer in short order!

The next successful customer had a console which worked fine when running a module plugged in to the cartridge port but would not give the TI BASIC option and if no module was plugged in gave a message to PLEASE INSERT CARTRIDGE instead. This clearly meant that the BASIC GROM was not being recognised, and sure enough when the first of the two BASIC GROMs was replaced all returned to normal. There are 3 GROM chips in the console, all of them in sockets. The lowest number of these (2155) is the system GROM which initialises the system on reset, produces the title screen and finds what is plugged into the system when the first key is pressed. It does this by looking at the data in the first bytes of all the possible GROMs (addresses >2000, >4000, >6000, >8000, >A000, >C000, >E000), and if version 1.1 system also at ROM address >6000, to see if any have data >AA there. If they do that indicates that a programme is present in that GROM or ROM and it then gets the programme name or names and puts them in the menu next to a number. When we press a number the system transfers control to that particular programme. So in this console, the GROM with the >AA byte was not working so the system could not find TI BASIC, but could find anything else which happened to be plugged in.

The third success was a console which came up with a title screen which had some wrong colours in the colour bars and some wrong characters in the text on the screen. Otherwise it seemed O.K. and would enter TI BASIC, still with what only seemed like bad spelling. By typing the alphabet in BASIC and comparing what appeared (ADDED..) to the ASCII codes of the letters typed, it appeared that one bit was always 0. This pointed to a possible failure of a memory chip as the VDP memory is uses eight 16K by 1 bit chips. There was another strange abberation which did not fit that theory, but I decided to investigate this first. Looking at the data signal out of the suspect chip it was always too low and so was the culprit. When replaced the title screen became correct except for the appearance of odd ! characters where there should have been spaces. These were grouped near the text, usually below it. Most peculiar I thought, and tried to see if it was another memory chip or the VDP processor perhaps. The data signals looked all the same now, but I ran a memory check programme which has a clock on the screen and noticed that when the clock had odd numbers the character 2 rows below the odd number changed from a space to a ! and back to a space when the number was even. Then I realised that all the ! were 2 rows below (and above sometimes) the characters with an odd ASCII code. The memory must be storing the same bit in several addresses at once. Replacing that memory chip and all was fine. Unusual to have 2 faults at the same time and both different, but similar.

In case you think that all is success down here there are some problems which are still defying me. One is my own console (I have bought a spare) which works fine for hours of TI-Writer or some other cartridge, but when trying to return to the title screen gets into all sorts of problems. Sytem GROM you say? My thoughts also but changing it did not solve the problem, just changed the time taken to fail. Obviously a heat problem, but spraying with cooler on various chips does not help. Turning off and back on again only helps if enough time elapses to allow it to cool down. I shall find it one day. The other problem at the moment is a console which does a similar thing when in BASIC, but when turned off and on again behaves normally straight away. That fault appears to have now developed into a more major one and is still unsolved.

First up this month news of recent software acquisitions from both local and overseas sources. I will give a brief description of each. All will be released over the coming months at the TIsHUG Shop with notification being in this column.

FUNLWRITER V3.3 - a copy of this was purchased at the TI Faire in Melbourne. Those wanting to update to this version will require 2 disks unless you have double sided drives when only 1 disk is needed. There is a lot of documentation included with this updated version with a copy of the latest version of 'C' thrown in for good measure. Those with Myarc controllor cards will be able to load that version of disk manager to the Funlwriter disk and access it from the Utilities option.

CHARACTER DEFINITION UTILITY - written in Extended Basic by a young TIsHUG member, Craig Sheehan. With his program you may design graphics on an 8x8 grid, place graphics on a 4x4 character pad, manipulate graphics, display hex code of graphics and save and reload pads to cassette. Craig has forwarded a very detailed instruction manual with his program as well as a Quick Reference Guide. There are also some demo pads on the cassette. Well done Craig, keep programs of this standard coming.

A DISK OF HYMNS - all with graphics, written in Extended Basic by Bill Knecht of the Houston Users Group. If you like music then this is for you.

THE MS ADVENTURE SERIES - is made up of the MS adventure program and three MS adventure data-bases, The Search for Murgens Keep, The Enchanted Keep and The New King. Written in assembly, the MS series requires Extended Basic, 32K expansion and disk drive. Frustrated adventures will like this one because if you get stuck the solutions are also on the disk.

DISK MANAGER 99 - this is a resident disk manager which stays in memory at all times. Requires either Extended Basic, Mini Mem or Editor Assembler plus 32K expansion and a disk drive. With DM99 loaded you can catalog, rename, initialise, rename, change protection and test a disk all from command mode without destroying the program in memory.

The Forth competition has now closed with only the single entry being received from Terry Johnsen. Terry's entry is a spell checking program which contains a dictionary of about 28,000 words expandable to about 34,000. 2 disk drives are required to operate his program. The program searches the file being checked and catalogs words not in the resident dictionary, outputs them to the printer and suggests 3 alternatives. Terry wishes to market this program through the shop. Copies have been given to committee members to evaluate and a decision is expected soon.

Now for this months software releases.

Disk:

Universal dis-assembler. See the July TND for a description. This was in fact on sale at the July meeting, however I omitted to announce its availability in this column last month.

The MS Adventure Series, also mentioned above. Please try not to cheat and look up the solutions before attempting to solve the puzzles.

Tape (No. 1986/8):

Will contain 8 programs with all bar Blackbox requiring Extended Basic. Here they are -

Blackbox - full instructions are included in the program. The idea is to locate objects by shooting into the blackbox. It is quite an entertaining, yet difficult game.

Blockz - instructions are also included with this one. Try to keep your "man" alive for as many movements as you can.

Crab Attack - no instructions with this one but you move your crab with the arrow keys and try to eat the fish while avoiding the killer crabs.

Go Kart - again no instructions but you control your racing Go Kart with a joystick. Once you complete course No.1 you move to course 2 with a 100% increase in speed. If you complete the second course I guess you go to No. 3 with a 50% increase in speed. I didn't get that far when playing.

Golf - no instructions but easy to play either with a joystick or keyboard. Press the fire button or "Q" key to increase driving power, move joystick left or right to change directional arrow or keys "S" or "D" do the same. Try to avoid the trees and water hazards.

Ninja - simple instructions included in the game. Belt your opponent before he does it to you.

Survival - instructions included. Try to complete the screens without falling into pits.

TI Targets - instructions again included. Shoot the moving targets with your supply of 30 arrows.

These games will also be available on disk if you prefer. On the disk is also a copy of a load program with the TIsHug logo. This program was recently written by Russel Welham and will be included on all future appropriate disk releases.

Recently I commenced corresponding with Jim Peterson of Tigergub Software. We have exchanged copies of each others software catalogs and are now in the process of exchanging disks on an ongoing basis. Jim has a very extensive public domain library and his catalog runs to well over 2000 separate programs some of which sound very interesting. Watch this column for more news.

BUG Department - Xwingpilot released on tape 1986/7 has some bugs throughout. The problems have occured when I resequenced the program after adding some lines at the start and deleting some REM's in the listing. To correct the program make these changes in the line numbers indicated -

Line 390 - change the 32767 to 420
Line 440 - change the 32767 to 470
Line 510 - change the 32767 to 560
Line 650 - change the 32767 to 340
Line 720 - change the 32767 to 840

Well that's it for this month. More software news next time.

If copying time permits there will also be some copies of FUNLWRITER 3.3 available at the shop at the August meeting.

LITTLE-ENDIANS AND BIG-ENDIANS

by Geoff Trott
Illawarra Regional Group - TIsHUG

In the world of computers there are two classes of people; those who are Little-endians and those who are Big-endians. These classifications have nothing to do with anatomy, but rather with the prejudices people have about the order in which the bits in data and addresses are numbered. Just a storm in a tea cup you say? Well I guess that is right, but the different ways used to number bits and bytes do cause confusion, particularly to assembly language programmers, so I will attempt to explain clearly what it is all about.

If you look at the circuit diagrams for the 9900 processor, or any description of the instruction bit codes for example, you will notice that the most significant bit of the 16 data bits or address bits is numbered 0 and the least significant bit is numbered 15. This is the pattern for the Big-endian order of the world. Also the first byte in the word, which is the even address, is also in the most significant position. This means that if words are written down in their numerical order across a page, the order of the bits and bytes all start from the top left and read naturally to the right. The pairs of bytes go together naturally to make words and the character strings are in their correct sequence. This is all so logical that it is hard to imagine why another way would be thought about or even preferred.

Let us have a look at the Little-endian way of numbering bits and bytes. The obvious difference is that the least significant bit is numbered 0 and the most significant bit has the largest number. The first byte would also go into the least significant byte position. This means that if we lay out the data in bytes starting from the top left of the page, then the words are in the wrong order bytewise as it were, or if the words are written down in their correct form, the bytes are not consecutive.

Let us take an example of a character string of 5 characters, and assume the format used by TI of the first byte containing the length of the string. The bytes would be 05 48 65 6C 70 21 in hexadecimal for the string 'Help!'. For the Big-endians this string would pack into consecutive words like this. 0548 656C 7021 For the Little-endians it would pack into words in the following way. 4805 6C65 2170 The same sort of problem arises with the Little-endian way when 16 bit data is broken into 2 bytes, it is in low byte followed by high byte order.

Well why would anyone want to be a Little-endian? For those of us who delve into the hardware, it is clear that the hardware manufacturers are confirmed Little-endians. This is because the number of bits used by a piece of hardware is generally less than that used by a computer. Several pieces of hardware are connected in parallel to do the job. In this case it makes more sense to consistently number the least significant bit as 0, because this will be correct for at least one of the pieces (for a Little-endian). If you consider the address lines of memory chips. These can have anywhere from 8 to 16 address lines, and there may be families of these components which change only by an increase in address lines. The Little-endian approach allows the address lines to be increased without requiring all the other address lines to be renumbered. This is the main advantage of the Little-endian approach, the ability to increase the number of address lines with the smallest impact on the numbering of the existing address lines. There are also advantages when transferring bytes to words and vice versa, as the bytes are in the correct part of the word to carry on normal arithmetic on the byte value. With the Big-endian approach the transfer of a byte into a word leaves the byte at the high end of the word, which then requires a byte swap within the word before normal arithmetic can be used - INC or DEC for example.

Some problems arise when the two concepts are mixed in the same system. This occurs within the design of some systems (DEC computers and the Motorola 68000 are examples), but it can occur in any system if the software is written accordingly. Most of the 99/4 computer was designed by Big-endians, but some of the disk operating system, was written by a Little-endian. It may be that the disk operating system is reasonable close to some other disk operating system from another computer, or just that whoever wrote it was a Little-endian at heart. This is why the file segment entry table appears mixed up, and the bit map is also in an odd order. So if you are delving into the disk operating system, or you find what appears to be an odd ordering of bytes in a word, brush up on your Little-endian thinking!

The Illawarra Regional Group

This group holds regular meetings in Saint Matthews Church Hall, Philip Crescent, Mangerton, on the third Monday of each month (except January) at 7.30 p.m. We also hold occasional hardware and other special interest group meetings at irregular intervals. We are offering memory expansion and other simple hardware expansions upon request, and are working on software for systems without disks but with memory expansion.

The meetings normally start with a tutorial session on Extended BASIC, followed by a talk and demonstration of some other topic of interest. This leads to some refreshments while members meet each other and chat about problems and interests. We maintain various libraries for the use of members.

LINKING EXT'D BASIC-ASSEMBLY

#6 LINK IT. #6

WITH ROSS MUDIE.

This program allows values to be examined in the VDP RAM (PEEKV), poked into the VDP RAM (POKEV) and poked into the VDP registers (POKER). Take care with what you poke where in VDP RAM and registers since the wrong thing in the right place will crash ext'd basic or the VDP, especially poking to the VDP registers.

The program is shortened from that in tutorial disk MUDIE_86/3 by using XMLLNK with DATA >20 to eliminate some maths.

```
*     VDP PEEKV, POKEV & POKER
*  Based on a program by John Brown in Millers Graphics
*  The Smart Programmer, April 1984. Modified & remarks
*  added by Ross Mudie of TISHUG, 26th June 1986.
*     PEEKV & POKEV can peek or poke up to 15 values.

       DEF  PEEKV,POKEV,POKER
NUMREF EQU  >200C
NUMASG EQU  >2008
XMLLNK EQU  >2018
VMBW   EQU  >2024
VMBR   EQU  >202C
VWTR   EQU  >2030
FAC    EQU  >834A
BUFF   BSS  18
STORE  DATA 100
MYWS   BSS  >10

* PEEKV reads VDP RAM Values.
* Extended basic format:  CALL LINK("PEEKV",add,v,v...)
* where v is from 1 to 15 numeric variables.
PEEKV  LWPI MYWS
       LIMI 0
       CLR  R0              Simple variable for NUMREF
       LI   R1,1      Point to first argument in NUMREF
       BLWP @NUMREF       Gets address (first argument)
       BLWP @XMLLNK     To convert floating point value
*                  for address in FAC to integer in FAC
       DATA >12B8       Data value for CFI using XMLLNK
       MOV  @FAC,R0      Integer value of address in R0
       LI   R1,BUFF     Address of Temp store for peeks
       MOVB @>8312,R2    Number of arguments in LINK is
*                             placed in left byte of R2
       SRL  R2,8     Swaps byte in R2, left byte zeroed
       AI   R2,-1    Number of bytes to be read by VMBR
*   is the number of arguments less one for the address
       BLWP @VMBR       Bytes from VDP RAM, put in BUFF

       CLR  R0               Simple variables in NUMASG
LOOP1  MOVB @BUFF-1(R2),R4   Takes the last byte first;
*     subsequently, (as R2 is decremented) working left
*        along BUFF with each LOOP until all bytes are
*                                    transferred to x/b
       SRL  R4,8       Swaps byte R4 & zeroes left byte
       MOV  R4,@FAC       R4 value in first word of FAC
       BLWP @XMLLNK   Convert Integer to Floating point
       DATA >20                Data for CIF in XMLLNK
       MOV  R2,R1          One less than argument number
       INC  R1             Equal to argument number now,
* remember that the peek address was the first argument
*   in LINK which we do not want to try to transfer to.
       BLWP @NUMASG   Transfer to x/b variable from FAC
       DEC  R2           To point to next peeked value,
*             (back one, working backwards thru BUFF).
       JNE  LOOP1          If more to do jump to LOOP1
       B    @RETURN

* POKEV Writes VDP RAM Value(s)
* Extended basic format: CALL LINK("POKEV",Add,nv,nv..)
* where nv is 1 to 15 numeric values or num variables.
POKEV  LWPI MYWS
       LIMI 0
       CLR  R0              Simple variable in NUMREF
       LI   R1,1           First argument in call link
       MOV  R1,@STORE      Place a 1 in STORE as a word
       BLWP @NUMREF      Gets first address of where to
*                      poke into FAC in radix 100 format
       BLWP @XMLLNK      Convert floating point address
*                           in FAC into integer in FAC
       DATA >12B8        DATA for CFI routine in XMLLNK
       MOV  @FAC,@BUFF   Places integer address in BUFF
*            the first poke address is a byte at BUFF+1
       AB   @>8312,@STORE+1      STORE (first word) now
*   contains the number of arguments in the x/b LINK +1
       LI   R3,2           Ready for NUMREF to point to
*                                 argument 2 in x/b LINK
LOOP2  MOV  R3,R1        Put R3 value in R1 to point to
*               argument number in LINK & byte in BUFF
       BLWP @NUMREF      Get value to be poked into VDP
       BLWP @XMLLNK       Convert floating point in FAC
*                                    to integer in FAC
       DATA >12B8        Data for CFI routine in XMLLNK
       MOVB @FAC+1,@BUFF(R3) Moves the byte size values
*             to be poked into BUFF starting at BUFF+2
       INC  R3             To point to next argument in
*                              LINK & next byte in BUFF
       C    R3,@STORE                  All poked yet?
       JNE  LOOP2                    No! get next one
       MOV  @BUFF,R0   Got all values, place poke start
*                                 address in R0 for VMBW
       LI   R1,BUFF+2   R1 contains the address in BUFF
*                              of the first 1st to poke
       MOV  R3,R2       R2 has the # of args in link +1
       AI   R2,-2   Actual # of bytes for VMBW to write
       BLWP @VMBW      Transfers poke values to VDP RAM

RETURN CLR  R0
       MOVB R0,@>837C   To indicate no errors on return
       LWPI >83E0                  Load GPL Workspace
       B    @>0070          To return to extended basic

* POKER Writes to a VDP Register
* Extended basic format:   CALL LINK("POKER",VReg #,nv)
* where nv may be a numeric value or numeric variable.
POKER  LIMI 0
       CLR  R0               Simple variable in NUMREF
       LI   R1,1              First argument in NUMREF
       BLWP @NUMREF            Get VDP register number
       MOV  @FAC,@STORE    Places both exponent & value
*    VDP register number (in radix 100 format) in STORE
       LI   R1,2                Second argument in link
       BLWP @NUMREF      Get value from link & place in
*                                FAC in radix 100 format
       BLWP @XMLLNK        Convert floating point value
*                             in FAC into integer in FAC
       DATA >12B8        Data for CFI routine in XMLLNK
       MOV  @FAC,R0   Integer value in right byte of R0
       MOVB @STORE+1,R0    Move byte for VDP register #
*         from right byte of STORE into left byte of R0
       BLWP @VWTR              Write to VDP reg from R0
       RT                   To return to extended basic

       END
```

BRANCHING ON TWO VARIABLES
SOME LESS THAN OBVIOUS HINTS ON EFFICIENT PROGRAMMING TIPS BASIC X-BASIC

Many problems have more than one correct solution. For instance, how to branch to one of six locations depending upon a particular combination of two variables. For reasons of memory and speed efficiency, we needed the absolute minimum number of variables and lines of code. The two variables involved were X and Y. X could be equal to either 1, 2, or 3, and Y could equal either 2 or 17.

The problem: How to combine X and Y in such a way as to have the total equal 1, 2, 3, 4, 5, or 6.

One solution to this problem is to temporarily change Y to either 0 or 3, then Y can be added to X to achieve the desired output. This can be done with a series of IF-THEN statements of a "dummy" variable for Y. However, the number of lines and extra variables required in this solution proves to be excessive.

Upon re-reading the ON-GOTO information in the User's Reference Guide, one will find when the numeric expression is evaluated, the result is reduced to an in- teger. By re-reading information about the INTeger function, we will discover that the function rounds the fractional values down. This means that a positive fraction which is less than 1 will yield an integer result of 0 and a decimal number of 3, plus a fraction will yield

3. We now have a possible solution.
Dividing Y by 5 will yield 0.4 when Y=2 and 3.4 when Y=17. When those numbers are added to X, the result will be 1.4, 2.4, 3.4, 4.4, 5.4, or 6.4. When the computer reduces the result to an integer, the expression will evaluate to 1, 2, 3, 4, 5, or 6, respectively.

Shown here is the algorithim submitted. Three alternate ones are also shown to illustrate the space and efficiency savings when using the ON-GOTO numeric expression. Submitted algorithim: 100 ON X+(Y/5)GOTO 200,300,
400,500,600,700 200 (Code for X=1 & Y=2)
: 300 (Code for X=2 & Y=2)
: 400 (Code for X=3 & Y=2)
: 500 (Code for X=1 & Y=17)
: 600 (Code for X=2 & Y=17)
: 700 (Code for X=3 & Y=17)

```
****************************** ALTERNATE NO. 1 100 IF Y=17 THEN 120 110 ON X GOTO 200,300,400 120 ON X GOTO 500,600,700
****************************** ALTERNATE NO. 2 100 D=0 110 IF Y=2 THEN 130 120 D=3 130 ON X + D GOTO 200,300,400,
   500,600,700
******************************
ALTERNATE NO. 3 100 IF Y=17 THEN 130 110 IF X=1 THEN 200 120 IF X=2 THEN 300 ELSE 400 130 IF X=1 THEN 500 140 IF X=2 THEN 600 ELSE 700
******************************
```

CAVEAT EMPTOR

by Ben Takach

The program copy session went like house on fire. Not being a game addict, I have paid my $6.- for three disk full of utilities.
One had plenty of time to chat once queued up armed with ready to use initialized disks.
Then it was my turn. One of my choices was some kind of disk fixer utility, which would only copy on virgin uninitialized disk. So there went two of my dollars down the drain. The remaining programs were on my two disks in a few minutes.
Would you believe it, I was chained to my computer for the rest of the weekend playing electronic Sherlock Holmes trying to get them run. To be more precise, trying to get them down-loaded into X.MEM. Alas, I remembered too late: "Caveat Emptor!". One particular batch of 6 programs were especially stubborn to leave the cosy grooves of DSK1.TRANSFER1. These all cataloged as programs, but got zapped with an I/O ERROR 50 as soon as XB called them OLD. I knew that XB is far more informative than a mere error code number, and would have given me an explicit explanation -even if this would at times be absolutely wrong-, so the message had to come from DISK MANAGER. However DISK MANAGER does not use error codes between 45 99. It's BASIC interfering with transactions between XB and TI-DOS! Why does it do so? It is none of its business! Let this be a lesson to you: there is no respect for privacy in the computer business! Anyhow, lets get back to the problem.
I tried TI-WRITER and the various options of ED-ASS. in desperation, only to be greeted with other versions of error messages. By then I was too tired to look up the respective manuals to decode them all. It was time to call on DISK FIXER to give me a hand. It was friendly and accomodating. The programs definitely looked like BASIC or XBASIC, yet none of the tricks I know would shift them.
At 3 AM on Monday morning my patience was definitely at its end. I was tempted to phone the guy who copied the files.
Then as a last ditch effort, a final defiant act, I bashed in his stubborn brain:

```
10 PRINT "GET WELL AND TRULY *@XC&**!!!"
20 RUN "DSK1.COMPRESS-3"
RUN
```

You have guessed it! It sensed instantly that I was not joking. The drive clicked and EXP.MEM swallowed the lot without any fuss or complaint like the sweetest of sweet mothers milk. Then I discovered that COMPRESS-3 is the EDITOR program of the TI PROGRAM AID-3 disk in disguise. I purchased this disk some 18 months ago for $29.95.
This is the essence of the proverb CAVEAT EMPTOR.
I still have some programs, which so far defied every attempt to do anything useful apart from painting the screen dark green and locking up the system tight. Someone with a weird sense of humor called them AIDE and AIDF. Any of you, who has the urge to possess a permanently locked crash proof green screen should phone me. AIDE and AIDF can be yours for just $2.-.
The moral to the story?
If the guy looks much stronger than you are, then it is wiser to think up an apt proverb than phoning him at 3 AM in the morning.
Thus, yours with caveat emptor.

TIPS by J.P

TIPS FROM THE TIGERCUB

#22

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213

This challenge was printed in Tips #21-July TND

```
100!The Unprintable Unkeyabl
e Program!
110!To shuffle the numbers 1
 to 255 into a random sequen
ce without duplication
120!The strings contain the
ASCII characters 1 to 127 an
d 128 to 255
130!Most of the ASCII charac
ters below 32 or above 159 c
annot be input from the keyb
oard
140!So how was this program
programmed?
150 M$="
          !""#$%&'()*+,-./0
123456789:;<=>?@ABCDEFGHIJKL
MNOPQRSTUVWXYZ[\]^_`abcdefgh
ijklmnopqrstuvwxyz{|}~"
160 M2$="

                          "
170 M$=M$&M2$
180 L=LEN(M$):: RANDOMIZE ::
 X=INT(L*RND+1):: N=ASC(SEG$
(M$,X,1)):: M$=SEG$(M$,1,X-1
)&SEG$(M$,X+1,LEN(M$))
190 PRINT N;:: IF LEN(M$)=0
THEN STOP ELSE 180
```

And here is the answer - It was written by a program that writes a program! Key this in and run it to create a MERGE format disk file. Then type NEW, then type MERGE DSK1.LONGSTRING and you will have a RUNable program consisting of lines 150-170 of the puzzle!

```
100 OPEN #1:"DSK1.LONGSTRING
",VARIABLE 163
110 LN=100 :: GOSUB 190 :: A
$=L$&"M$"&CHR$(190)
120 FOR J=1 TO 127 :: C$=C$&
CHR$(J):: NEXT J :: A$=A$&CH
R$(199)&CHR$(127)&C$&CHR$(0)
130 PRINT #1:A$
140 GOSUB 190 :: B$=L$&"M2$"
&CHR$(190)
150 FOR J=128 TO 255 :: D$=D
$&CHR$(J):: NEXT J :: B$=B$&
CHR$(199)&CHR$(128)&D$&CHR$(
0)
160 PRINT #1:B$
170 GOSUB 190 :: F$=L$&"M$"&
CHR$(190)&"M$"&CHR$(184)&"M2
$"&CHR$(0)
180 PRINT #1:F$ :: PRINT #1:
CHR$(255)&CHR$(255):: CLOSE
#1 :: END
190 L$=CHR$(INT(LN/256))&CHR
$(LN-256*INT(LN/256)):: LN=L
N+10 :: RETURN
```

Now type in the remaining lines, and you will have a speeded-up version of the Tigercub Scramble which was published in Tips #10. It is still not as fast as the CALL PEEK versions but is much more useful because you can modify it to scramble a sequence of any length anywhere between 1 and 255. For example, to shuffle the numbers 100 to 150 into a random sequence without duplication, just add a line 175 M$=SEG$(M$,100,50).

The method of writing a "program that writes a program" was fully explained by John Clulow in the 99er magazine Vol. 1 Nos. 3 and 4. It is a little-used but very valuable technique.

For instance, Tips#9 contained the following routine to turn the alphabet upside-down.

```
100 FOR CH=33 TO 127 :: CALL
 CHARPAT(CH,CH$):: FOR J=1 T
O 16 STEP 2 :: X$=SEG$(CH$,J
,2)&X$ :: NEXT J :: CALL CHA
R(CH,X$):: X$="" :: NEXT CH
110 INPUT A$ :: GOTO 110
```

The only trouble with that is that it takes about 50 seconds to run. Try this instead -

```
100 FOR CH=33 TO 127 :: CALL
 CHARPAT(CH,CH$):: FOR J=1 T
O 16 STEP 2 :: X$=SEG$(CH$,J
,2)&X$ :: NEXT J :: CALL WRI
TE(CH,X$):: X$="" :: NEXT CH
1000 SUB WRITE(CH,X$):: IF F
LAG=1 THEN 1010 :: FLAG=1 ::
 OPEN #1:"DSK1.WRITE",OUTPUT
,DISPLAY ,VARIABLE 163 :: LN
=3000 :: GOSUB 3000
1010 X=X+1 :: L$=L$&CHR$(200
)&CHR$(16)&X$ :: IF X<5 AND
CH<127 THEN L$=L$&CHR$(179):
: SUBEXIT
1020 X=0 :: PRINT #1:L$&CHR$
(0):: L$="" :: IF CH=127 THE
N 1030 :: GOSUB 3000 :: SUBE
XIT
1030 PRINT #1:CHR$(255)&CHR$
(255):: CLOSE #1 :: GOTO 301
0
3000 L1=INT(LN/256):: L2=LN-
256*L1 :: L$=CHR$(L1)&CHR$(L
2)&CHR$(147):: LN=LN+10 :: R
ETURN
3010 SUBEND
```

RUN that, type NEW, then MERGE DSK1.WRITE, and you will have a program consisting of DATA statements containing the hex codes for all the upside-down characters. Add a line 100 FOR CH=33 TO 127 :: READ CH$ :: CALL CHAR(CH,CH$):: NEXT CH, and you can turn everything upside-down in only 12 seconds.

Someone sent me a classified ad, clipped from an unknown publication, which read -

TI-WRITER COMPANION. Loaded with ingenious ways to make your TI-Writer more effective. Well written. Send $6.50 to Dr. Bill Browning, 7541 Jersey Avenue North, Brooklyn Park, MN 55428. Money back guarantee.

I sent off my money and have just received 29 pages, 3-hole punched, loaded with useful and ingenious tips and ideas for getting more out of TI-Writer. I recommend it - it's worth twice the money and then some! NOTE! Now $6.50!

The K-Town newsletter recently published a utility routine that is so useful that I want to pass it on to everyone. If a program is not resequenced after it is modified, this will compare it with the original and prepare a MERGE format file of all the changes, for the use of others to update their copy.

```
100 !***********************
110 !*  COMPARE PROGRAM    *
120 !    by Mike Dodd       *
130 !***********************
131 ! In K-Town 99'er V.2 #1
      April 1985
140 !Version 85.0406.1XB
      Requires disk drive.
      Compares two programs,
      gives list of all diff-
      erences.
150 !SAVE old program in
 MERGE format (SAVE DSK1.(ol
 dfilename),MERGE). SAVE up-
 dated program in MERGE for-
 mat(SAVE DSK1.(newfilename)
,MERGE)
160 !RUN this program, answe
r prompts for OLD FILE name,
 NEW FILE name, and a differ
ent OUTPUT FILE name.
170 !When finished, type NEW
, then MERGE DSK1.(outputfil
ename) and ENTER
180 !Can be MERGED into othe
r copies of OLD program to
update them
190 DEF @(@$)=ASC(SEG$(@$,1,
1))*256+ASC(SEG$(@$,2,1))
200 A$=CHR$(255)&CHR$(255)::
 DISPLAY AT(1,1)ERASE ALL:"O
LD    FILE:": :"NEW    FILE:
": :"OUTPUT FILE:"
```

```
210 ACCEPT AT(1,13)BEEP:B$ :
: ACCEPT AT(3,13)BEEP:C$ ::
ACCEPT AT(5,13)BEEP:D$ :: OP
EN #1:B$,INPUT ,VARIABLE 163
220 OPEN #2:C$,INPUT ,VARIAB
LE 163 :: OPEN #3:D$,OUTPUT,
VARIABLE 163
230 LINPUT #1:@$ :: LINPUT #
2:E$ :: F$=SEG$(@$,1,2):: G$
=SEG$(E$,1,2):: A=@(F$):: B=
@(G$)
240 IF F$=A$ AND G$=A$ THEN
CLOSE #1 :: CLOSE #2 :: PRIN
T #3:A$ :: CLOSE #3 :: STOP
250 IF B>A THEN PRINT #3:F$&
CHR$(131)&" **DELETED LINE *
*"&CHR$(0):: LINPUT #1 :: @$
 :: F$=SEG$(@$,1,2):: A=@(F$
):: GOTO 240
260 IF A>B THEN PRINT #3:E$
:: LINPUT #2:E$ :: G$=SEG$(E
$,1,2):: B=@(G$):: GOTO 240
270 IF @$<>E$ THEN PRINT #3:
E$
280 GOTO 230
```

Thanks to some ideas from Joyce Corker, I have made some more improvements to the Tigercub Menuloader, and I have used the above utility routine to list all the changes made since it was published in Tips#15.

```
100 !by A. Kludge/M. Gordon/
T. Boisseau/J. Peterson/etc.
modified in Tips #22
102 OPTION BASE 1 :: DIM PG$
(127),VV(127),VX(127):: GOTO
 110
105 @,A,A$,B,C,D$,FLAG,I,J,K
,KD,KK,N$,NN,P$,PG$(),Q$,S,S
T,T$(),TT,VT,VV(),VX(),W$,X,
X$,K2,S2
106 CALL INIT :: CALL LOAD :
: CALL LINK :: CALL PEEK ::
CALL KEY :: CALL SCREEN :: C
ALL COLOR :: CALL CLEAR :: C
ALL VCHAR :: CALL SOUND :: !
@P-
150 ! **DELETED LINE **
160 T$(1)="d/f" :: T$(2)="d/
v" :: T$(3)="i/f" :: T$(4)="
i/v" :: T$(5)="pro" :: ON WA
RNING NEXT
170 IMAGE ###
180 DISPLAY AT(1,4):"TIGERCU
B MENU LOADER"
210 D$="DSK1." :: OPEN #1:D$
,INPUT ,RELATIVE,INTERNAL ::
 INPUT #1:N$,A,J,K :: DISPLA
Y AT(1,2)SIZE(27):SEG$(D$,1,
4)&" - Diskname= "&N$;
230 FOR X=1 TO 127 :: IF X/2
0<>INT(X/20)THEN 260
240 DISPLAY AT(24,1):"Type c
hoice or 0 for more 0" :: AC
CEPT AT(24,27)VALIDATE(DIGIT
)SIZE(-3):K :: IF K=0 THEN 2
50 :: IF VV(K)<>5 THEN 411 :
: IF K>0 AND K<NN+1 THEN 420
 ELSE 240
290 DISPLAY AT(X+4,2):USING
170:NN :: DISPLAY AT(X+4,6):
P$ :: PG$(NN)=P$ :: DISPLAY
AT(X+4,18):USING 170:J :: DI
SPLAY AT(X+4,22):T$(ABS(A))
291 VV(NN)=ABS(A):: VX(NN)=A
BS(B)
295 X$="  "&STR$(B):: DISPLA
Y AT(X+4,26):SEG$(X$,LEN(X$)
-2,3):: VT=VT+J
350 DISPLAY AT(X+6,1):"    C
hoice?" :: ACCEPT AT(X+6,16)
SIZE(3)VALIDATE(DIGIT):K ::
IF K<>NN AND K<>NN+1 THEN 41
0
410 IF K<1 OR K>127 OR LEN(P
G$(K))=0 THEN 320
411 IF VV(K)=5 OR(VV(K)=4 AN
D VX(K)=254)THEN 420
412 ON ERROR 417 :: CALL CLE
AR :: OPEN #2:D$&PG$(K):: CA
LL SCREEN(16)
413 LINPUT #2:W$ :: PRINT W$
 :: IF EOF(2)THEN 416
414 CALL KEY(0,K,S):: IF S=0
 THEN 413
415 CALL KEY(0,K2,S2):: IF S
2<1 THEN 415 ELSE 413
416 CLOSE #1 :: CLOSE #2 ::
END
417 DISPLAY AT(12,10):"UNLIS
TABLE" :: CALL SOUND(200,110
,0):: RETURN 400
430 ON ERROR 417 :: CALL INI
T :: CALL PEEK(-31952,A,B)::
 CALL PEEK(A*256+B-65534,A,B
):: C=A*256+B-65534 :: A$=D$
&PG$(K):: CALL LOAD(C,LEN(A$
))
```

The Menu Loader will now list up to 127 programs and files, showing the number of sectors in each and the file type, record type and record length of each file. It will stop at the end of each page, and continue on a default value of 0, or will stop for selection when any key is pressed. It gives disk name, number of sectors used and available. It adds up sectors actually used and gives a warning if all sectors are not accounted for. It will load and run any program which can be loaded from Extended Basic, displaying the program being loaded. It will delete any program or file, after first displaying the filename and requesting verification. It will list any listable file to the screen, pausing on any key input, and can be very easily modified to list to a printer. If a file is not listable, it will inform you so, and restart the menu selection. It has the pre-scan option to speed it up.

Fairly often, the disk directory will lose track of one or a few sectors during the process of loading records, even though the Disk Manager showed all 358 were initialized. That's why I put the checking routine in the Menu Loader. The figure shown as "used" is actually 358 minus the number of sectors still available, and is checked against the total sectors of all files.

The loss of a few sectors is no serious matter, but once in a great while you may notice that the "available" and "used" sector quantities have obviously been reversed. I have found that this is a signal that the disk is about to go haywire and you had best back it up immediately!

Programs and files are loaded in the first available sector, and continued in the next available sector. If a number of small files are deleted from a disk, and a long file is then loaded, it may thus be fractured into many parts. If you have a work disk on which you continually add and delete files of various lengths, it will become badly fractured. This can cause disk errors, and it also badly overworks your drive. It is a good idea to recopy your work disk occasionally - file by file, not sector by sector with a quick copier.

TIPS FROM THE TIGERCUB #23

Several different routines have been published which will extract and save a specified series of lines out of a program, but this one by George Steffen of the L.A. 99ers is certainly the best.

```
1 !SUBROUTINE EXTRACTOR by G
eorge F. Steffen. SAVE in ME
RGE format. MERGE into any p
rogram (with line # starting
 above 8). RUN to extract
2 !selected lines. Deletes i
tself. Then BE SURE to SAVE
the selected lines in MERGE
format because the remaining
 lines are still in memory!
3 CALL CLEAR :: CALL INIT ::
 INPUT "Line numbers of rout
ine to be saved: First,Last?
       ":L,M :: G=256 :: CAL
L PEEK(-31952,H,I,J,K)
4 C=INT(M/G):: D=M-C*G :: F=
(J-G)*G+K :: FOR E=(H-G)*G+I
 TO F STEP 4 :: CALL PEEK(E,
A,B):: IF A=C AND B=D THEN 6
5 NEXT E :: PRINT :"LINE";M;
"NOT FOUND!" :: STOP !@P-
6 H=INT(E/G):: I=E-(G*H):: H
=H+G :: C=INT(L/G):: D=L-C*G
 :: FOR E=E+4 TO F STEP 4 ::
 CALL PEEK(E,A,B):: IF A=C A
ND B=D THEN 8 !@P-
7 NEXT E :: PRINT :"LINE";L;
"not found!" :: STOP !@P-
8 E=E+3 :: J=INT(E/G):: K=E-
(G*J):: J=J+G :: CALL LOAD(-
31952,H,I,J,K):: STOP !@P-
```

Some folks were interested in the idea of a program that writes a program, so let's write a program that will write a program to list the token codes that you need to use to write a program that will write a program -

```
100 OPEN #1:"DSK1.TOKENLIST"
,OUTPUT,DISPLAY ,VARIABLE 16
3 :: FOR N=129 TO 254 :: L1=
INT(N/256):: L2=N-256*L1
110 PRINT #1:CHR$(L1)&CHR$(L
2)&CHR$(131)&CHR$(N)&CHR$(0)
:: NEXT N
120 PRINT #1:CHR$(255)&CHR$(
255):: CLOSE #1 :: END
```

Key that in and SAVE it just in case, then RUN it. When READY, type NEW, then MERGE DSK1.TOKENLIST. Now LIST it and you will see a list of ASCII codes 129 through 254 and their token meanings. Delete lines 171 through 175, 185, 198, 226 through 231, and 242. Change the definition of 199 to QUOTED STRING, of 200 to UNQUOTED STRING, and 201 to LINE NUMBER, and add line 255 END OF FILE.

You don't need all those exclamation points, so change the program to a DIS/VAR 80 file by LIST "DSK1.TOKENLIST". Then key in this little routine.

```
100 OPEN #1:"DSK1.TOKENLIST"
 :: OPEN #2:"PIO"
110 LINPUT #1:A$ :: PRINT #2
:SEG$(A$,1,4)&SEG$(A$,6,LEN(
A$)):: IF EOF(1)<>1 THEN 110
120 CLOSE #1 :: CLOSE #2 ::
END
```

RUN it, and print out a list of all the token codes. More on this next month - if someone buys a few programs so that I can afford another month.

Now that we've done about all that we can with the Menu Loader, here is another version to use on your finalized library disks of programs. It lacks the features that you will no longer need, but will list your programs by their full names, up to 24 characters long.

```
100 !NAMELOADER by A. Kludge
/M. Gordon/T. Boisseau/J. Pe
terson/etc.
110 CALL CLEAR :: CALL SCREE
N(5):: FOR S=1 TO 14 :: CALL
 COLOR(S,7,16):: NEXT S :: C
ALL VCHAR(1,31,1,96):: CALL
COLOR(0,2,16)
120 OPTION BASE 1 :: DIM PG$
(99),M$(99)
130 ! List the full names of
 the programs on the disk in
 the DATA statements, in the
 sequence in which they are
 listed by an ordinary disk
 cataloger program
140 !Then SAVE this program
under the filename LOAD
150 DATA
160 DATA
170 DATA
180 DATA
190 DATA END
200 FOR J=1 TO 99 :: READ M$
(J):: M$(J)=SEG$(M$(J),1,24)
210 IF M$(J)="END" THEN M$(J
)=" " :: GOTO 230
220 NEXT J
230 IMAGE ##
240 DISPLAY AT(1,4):"TIGERCU
B NAMELOADER"
250 D$="DSK1." :: OPEN #1:D$
,INPUT ,RELATIVE,INTERNAL ::
 INPUT #1:P$
260 FOR X=1 TO 99 :: IF X/20
<>INT(X/20)THEN 290
270 DISPLAY AT(24,1):"Type #
 of choice or Enter 0" :: AC
CEPT AT(24,27)VALIDATE(DIGIT
)SIZE(-3):K :: IF K=0 THEN 2
80 :: IF K>0 AND K<NN+1 THEN
 390 ELSE 270
280 X=1
290 I=I+1 :: IF I>127 THEN K
=X :: GOTO 370
300 INPUT #1:P$ :: NN=NN+1
310 IF LEN(P$)=0 THEN 350
320 DISPLAY AT(X+3,2):USING
230:NN :: DISPLAY AT(X+3,5):
M$(NN):: PG$(NN)=P$
330 CALL KEY(0,KK,ST):: IF S
T=0 THEN 340 :: FLAG=1 :: GO
TO 350
340 NEXT X
350 DISPLAY AT(X+4,1):" " ::
 DISPLAY AT(X+5,2):USING 230
:NN+1 :: DISPLAY AT(X+5,6):"
Terminate"
360 DISPLAY AT(X+6,1):"    C
hoice?" :: ACCEPT AT(X+6,16)
SIZE(2)VALIDATE(DIGIT):K ::
IF K<>NN AND K<>NN+1 THEN 38
0
370 IF K=NN+1 THEN CALL CLEA
R :: CLOSE #1 :: END
380 !IF K<1 OR K>99 OR LEN(P
G$(K))=0 THEN 350
390 CLOSE #1
400 CALL INIT :: CALL PEEK(-
31952,A,B):: CALL PEEK(A*256
+B-65534,A,B):: C=A*256+B-65
534 :: A$=D$&PG$(K):: CALL L
OAD(C,LEN(A$))
410 FOR I=1 TO LEN(A$):: CAL
L LOAD(C+I,ASC(SEG$(A$,I,1))
):: NEXT I :: CALL LOAD(C+I,
0)
420 CALL VCHAR(1,3,32,672)::
 CALL SCREEN(8):: FOR S=0 TO
 14 :: CALL COLOR(S,2,1):: N
EXT S :: DISPLAY AT(12,2):"L
OADING ";M$(K)
430 RUN "DSK1.1234567890"
```

Last month I forgot to have anything for the kids, or anything in Basic, so -

```
100 CALL CLEAR
110 REM by Jim Peterson of
Tigercub Software
120 PRINT TAB(1);"****AUTOMA
TIC MOUSE MAZE****": : : :"
  Choose your mouse and":"wa
tch it try to find its way"
130 PRINT "through the maze.
": :"   When one of the mice
 has":"taken 50 extra steps,
 the":"cat gets it!"
140 PRINT : :"Touch any key"
150 CALL KEY(0,K,ST)
160 IF ST<1 THEN 150
170 CALL CLEAR
180 CALL CHAR(120,"0078FEFFF
E78")
190 CALL CHAR(121,"1038387C7
C7C7C38")
200 CALL CHAR(122,"387C7C7C7
C383810")
210 CALL CHAR(123,"001E7FFF7
F1E")
220 CALL CHAR(128,"001E61816
11E")
230 CALL CHAR(129,"384444444
4242410")
240 CALL CHAR(130,"102828444
4444438")
250 CALL CHAR(131,"007886818
678")
260 CALL SCREEN(5)
270 T1=610
280 T2=610
290 CALL CHAR(136,"FFFFFFFFF
FFFFFFF")
300 CALL COLOR(14,16,16)
310 CALL COLOR(13,2,16)
320 CALL COLOR(12,2,16)
330 R=10
340 GOSUB 1460
350 R1=10
360 C=2
370 C1=2
380 CALL HCHAR(R,C,136,2)
390 C=C+1
400 M=120
410 M2=128
420 RANDOMIZE
430 A=(INT(2*RND)+1)*2
440 B=INT(10*RND)+1
450 ON B GOSUB 470,470,470,4
70,510,510,550,550,590,590
460 GOTO 420
470 IF C+A>30 THEN 630
475 if (c>20)*(x<10)then 500
480 CALL HCHAR(R,C,136,A)
490 C=C+A
500 RETURN
510 IF R+A>20 THEN 540
515 x=x+1
520 CALL VCHAR(R,C,136,A)
530 R=R+A
540 RETURN
550 IF R-A<2 THEN 580
555 x=x+1
560 CALL VCHAR(R-A+1,C,136,A
)
570 R=R-A
580 RETURN
590 IF C-A<3 THEN 620
595 x=x+1
600 CALL HCHAR(R,C-A+1,136,A
)
610 C=C-A
620 RETURN
630 CALL HCHAR(R,C,136)
640 C=C+1
650 IF C<31 THEN 630
660 R2=R
670 C2=C
680 CALL HCHAR(R1,C1,M)
690 CALL HCHAR(R2,C2,M2)
700 Y=Y+1+(Y=2)*2
710 IF Y=2 THEN 1020
720 CALL HCHAR(R1,C1,136)
730 ON M-119 GOTO 800,900,74
0,850
740 IF C1=31 THEN 950
750 CALL GCHAR(R1,C1+1,G)
760 IF G=32 THEN 850
770 C1=C1+1
780 M=120
790 GOTO 950
800 CALL GCHAR(R1-1,C1,G)
```

(CONTINUED PAGE 19)

POOR MANS DOUBLE DENSITY DISK CONTROLLER

or how to get 360K bytes per drive using the TI Disk Controller Card and 96 tpi double side drives. (eg TEAC 55f, TANDON TM100-4

This code, when substituted for the existing DSR code, converts disk #4 and disk #5 into 40/80 track interlace mode and disk #9 to 40/80 track non-interlace (single side). The modification has been completely compatable with all software tested including P-System, Disk Manager, Editor Assembler and Basic. Note however that files can only be exchanged with "normal" format disks using the disk copy routines or dsk3 (disk #9).

This code is placed into the public domain by the author for non commercial use.

Any questions may be directed to the author at

Andy Cooper
121 Clearview Drive
Downingtown Pa 19335

or Compuserve 71016,1743
Delphi andy4820

```
        .absolute
        .proc newdsk
        .org 4116h
        clr     r7                      ; existing code
        ci      r1,     1440            ; 1440 is max sector #
        jhe     $+152                   ; jump to error if greater
        ci      r1,     1               ; check for sector = 0
        jh      $5                      ; jump if not
        bl      @4524h                  ; if sector = 0 then restore
$5      clr     r0
        cb      @004Ch(r9),@4BA6h       ; compare drive # to 3
        jl      $10                     ; jump if 1 or 2
        div     @441Eh, r0              ; If drive = 3 use 9 sectors per track
        jmp     $20
$10     div     @547Ch, r0              ; if drive < 3 use 18 sectors per track
$20     swpb    r0
        inv     r0                      ; R0 msb contains inverted track #
        bl      @4614h                  ; set up vdp write
        movb    r0,     @0FFFEh(r15)    ; Store new (calc) track #
        movb    r0,     @5FFEh          ; write trk # to 1771 disk controller
        ci      r1,     9               ; check for sector > 8 (side two)
        jl      $30
        ai      r1,     -9              ; If >9 subtract nine and....
        sbo     7                       ; select side two (head 2)
        li      r7,     0100h
$30     swpb    r1
        inv     r1                      ; R1 msb contains inverted sector #
        movb    r1,     @5FFCh          ; write to 1771 sector register
        jmp     $40                     ; waste a word !!!!!!
$40     cb      r0,     @5FF2h          ; Back to original code.

        .end
```

To install a 2732 or 2732A prom on the TI Disk controller card with the above changes perform the following steps.

Remove the ROM at location U26.

On the back (non component) of the board cut the wide (+5v) etch between U26 pins 21 and 24.

On the front of the board cut the wide etch going to U26 pin 21.

On the front (component side) of the board cut the etch going to U26 pin 18.

NOTE: to ensure that the etch is cut I recommend that two cuts 1/16 inch apart be made and the copper trace be removed between the two cuts.

Install a 24 pin socket at U26

On the back (non component) side of the board add the following wires:

U26 pin 18 to U26 pin 20.

U27 pin 18 to U26 pin 21.

Via hole in center of chip area between U26 pin 21,22 to junction of C26 and U36 pin 24 (+5v).

Install prom in socket and.....

thats all!

The following is an Editor Assembler program that may be used to convert normal disks to interlace format prior to changing the Disk DSR. If you wish drive 3 can be used to copy single side disks to drives 1 and 2 using the P-System filer or Disk Manager. (adapted from sector RW tutorial writen by Tod Kaplan).

```
MYREG   BSS 32
        DEF START
        REF DSRLNK,VSBW,VMBW
        REF VMBR,KSCAN
START
        LWPI MYREG
LOOP    CLR @>8374
        BLWP @KSCAN
        CB   @KKEY,@SPACE
        JNE  LOOP
        CLR  R4
        CLR  R5
        CLR  R6
LOOP1   CI   R4,360     ;720 FOR DOUBLE SIDE
        JL   LOOP2
        LIMI 0
        LIMI 2          ;WAIT FOR QUIT
        JMP  LOOP1
LOOP2   MOV  R4,@SECTOR ;SECTOR TO PROCESS
        BL   @RSECT     ;READ SECTOR
        MOV  R5,@SECTOR ;SECTOR TO WRITE
        BL   @WSECT     ;WRITE SECTOR
        INC  R4         ;INCREMENT READ SECTOR (LINEAR)
        MOV  R4,R1
        CLR  R0
```

```
810 IF G=32 THEN 740
820 R1=R1-1
830 M=121
840 GOTO 950
850 CALL GCHAR(R1+1,C1,G)
860 IF G=32 THEN 900
870 R1=R1+1
880 M=122
890 GOTO 950
900 CALL GCHAR(R1,C1-1,G)
910 IF G=32 THEN 800
920 C1=C1-1
930 M=123
940 GOTO 950
950 CALL HCHAR(R1,C1,M)
960 IF (C1=31)*(C2=2)THEN 13
20
970 IF C1<31 THEN 700
980 T2=T2-10
990 CALL SOUND(50,T2,5)
1000 IF T2=110 THEN 1340
1010 GOTO 700
1020 CALL HCHAR(R2,C2,136)
1030 ON M2-127 GOTO 1040,120
0,1090,1150
1040 CALL GCHAR(R2+1,C2,G)
1050 IF G=32 THEN 1090
1060 R2=R2+1
1070 M2=129
1080 GOTO 1250
1090 IF C2=2 THEN 1250
1100 CALL GCHAR(R2,C2-1,G)
1110 IF G=32 THEN 1150
1120 C2=C2-1
1130 M2=128
1140 GOTO 1250
1150 CALL GCHAR(R2-1,C2,G)
1160 IF G=32 THEN 1200
1170 R2=R2-1
1180 M2=130
1190 GOTO 1250
1200 CALL GCHAR(R2,C2+1,G)
1210 IF G=32 THEN 1040
1220 C2=C2+1
1230 M2=131
1240 GOTO 1250
1250 CALL HCHAR(R2,C2,M2)
1260 IF (C2=2)*(C1=31)THEN 1
320
1270 IF C2>2 THEN 700
1280 T1=T1-10
1290 CALL SOUND(50,T1,5)
1300 IF T1=110 THEN 1370
1310 GOTO 700
1320 CALL HCHAR(1,1,32,768)
1325 X=0
1330 GOTO 330
1340 GOSUB 1460
1350 PRINT "THE CAT GOT THE
WHITE MOUSE": :
1360 GOTO 1390
1370 GOSUB 1460
1380 PRINT "THE CAT GOT THE
BLACK MOUSE": :
1390 PRINT "TO PLAY AGAIN, T
OUCH ANY KEY"
1400 CALL KEY(0,K,ST)
1410 IF ST<1 THEN 1400
1420 T1=610
1430 T2=610
1440 CALL HCHAR(1,1,32,768)
1450 GOTO 330
1460 CALL HCHAR(23,1,32,32)
1470 PRINT CHR$(120);(610-T1
)/10;TAB(20);CHR$(128);(610-
T2)/10
1480 RETURN
```

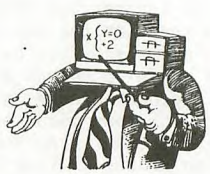
Did you know that ACCEPT AT(1,0) will accept a full line of 28 characters? Did you know that ACCEPT AT (R,0)SIZE(-28) and Enter will accept everything on row R? And did you know that ACCEPT N$ will accept a string of 255 characters?

Need a filler, so -

```
100 !MUSICAL BARGRAPH by Jim
 Peterson
110 CALL CLEAR :: CALL SCREE
N(5):: FOR J=2 TO 14 :: X=J-
(J>4):: CALL COLOR(J,X,X)::
NEXT J
120 DIM N$(13),N(13):: M$="(
08@HPX`hpx"&CHR$(128)&CHR$(1
36):: FOR J=1 TO 13 :: N$(J)
=SEG$(M$,J,1):: DISPLAY AT(J
+6,1)SIZE(1):N$(J):: NEXT J
130 X=110 :: FOR J=1 TO 13 :
: N(J)=X*1.059463094^(J-1)::
 NEXT J
140 A=INT(13*RND+1):: B=INT(
25*RND+1):: DISPLAY AT(A+6,2
)SIZE(28):RPT$(N$(A),B):: CA
LL SOUND(B*40,N(A),0,N(A)*2+
4,0,N(A)*4+6,0)
150 DISPLAY AT(A+6,2):"" ::
GOTO 140
```

MEMORY FULL

Jim Peterson



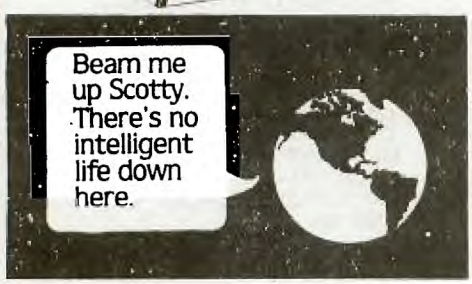

```
        DIV  @NINE,R0    ;CALC TRACK
        MOV  R0,R2       ;TRK TO R2
        ANDI R2,1
        JEQ  EVEN        ;JMP IF EVEN TRACK
ODD
        CLR  R2
        MOV  R0,R3       ;TRK TO R3
        DIV  @TWO,R2     ;DIVIDE IT BY 2
        CLR  R3
        MPY  @NINE,R2
        LI   R2,711
        S    R3,R2
        A    R1,R2
        MOV  R2,R5       ;INTERLACE (SIDE 2) TRK/SECTOR
        JMP  LOOP1
EVEN
        CLR  R2
        MOV  R0,R3       ;TRACK TO R3
        DIV  @TWO,R2     ;DIVIDE TRACK BY 2
        CLR  R3
        MPY  @NINE,R2
        A    R1,R3
        MOV  R3,R5       ;SIDE 1 TRACK
        JMP  LOOP1

*************************************
* READ SECTOR ROUTINE
*
RSECT
        MOV  @DUMPAB,R0
        LI   R1,DUMDAT
        LI   R2,2
        BLWP @VMBW
        MOV  @KD1RD,@DRVFLG ;READ FROM DRIVE 1
        MOV  @DUMBUF,@BUFADD
        MOV  @DUMPAB,@PABADD
        BLWP @DSRLNK
        DATA 10
        RT

*************************************
* WRITE SECTOR ROUTINE
*
WSECT
        MOV  @DUMPAB,R0
        LI   R1,DUMDAT
        LI   R2,2
        BLWP @VMBW
        MOV  @KD2WR,@DRVFLG  ;WRITE TO DRIVE 2
        MOV  @DUMBUF,@BUFADD
        MOV  @DUMPAB,@PABADD
        BLWP @DSRLNK
        DATA 10
        RT
*************************************
* DEFINES AND DATA
*
NINE    DATA 9
TWO     DATA 2
KKEY    EQU >8375
PABADD  EQU >8356
BUFADD  EQU >834E
SECTOR  EQU >8350
DRVFLG  EQU >834C
SPACE   DATA >2020
DUMPAB  DATA >1500
KD1RD   DATA >0101
KD1WR   DATA >0100
KD2WR   DATA >0200
KD3WR   DATA >0300
DUMDAT  DATA >0110
DUMBUF  DATA >1000

        END
```

Younger Set



Hello to all my Younger Set friends. I dont't have a real lot for you this month as once again not many of you have taken the effort to write to me and let me know, plus all other readers, just what you are doing with your computer. Remember I like to hear from you and hear about your high scores on your favourite games. Anyway I did hear from both Sam Mudie (age 13) and his younger brother Peter (age 9). Both sent me in

3 entries each to the Mouse competition which was run a couple of months back. So sorry but their entries were received too late to be included in that competition but as the programs are all short ones I have included them in this column for all other Younger Set members to type in and enjoy. Here are the programs. I hope to have a bit more news for you next month.

```
100 T=96 :: CALL CLEAR :: CA
LL CHAR(T,"000C1E3F5E924038"
)
110 CALL SPRITE(#1,T,16,T,T)
:: CALL MAGNIFY(2)
120 CALL KEY(1,K,S):: CALL J
OYST(1,X,Y)
130 IF K=5 THEN Y=4 ELSE IF
K=0 THEN Y=-4 ELSE IF K=2 TH
EN X=-4 ELSE IF K=3 THEN X=4
140 CALL MOTION(#1,-Y*8,X*8)
:: GOTO 120
```

3 from Peter
then 3 from Sam

```
100 T=96 :: CALL CLEAR :: CA
LL SCREEN(2):: CALL CHAR(T,"
00E0B0D73F1832787F3E1F2D54A3
010000070DEBFC184C1EFE7CF8B4
2AC580")
110 CALL SPRITE(#1,T,9,T,T):
: CALL MAGNIFY(4)
120 CALL KEY(1,K,S):: CALL J
OYST(1,X,Y)
130 IF K=5 THEN Y=4 ELSE IF
K=0 THEN Y=-4 ELSE IF K=2 TH
EN X=-4 ELSE IF K=3 THEN X=4
140 CALL MOTION(#1,-Y*8,X*8)
:: GOTO 120
```

```
100 N=96 :: CALL CLEAR :: CA
LL SCREEN(2):: CALL CHAR(N,"
00E0B0D73F1832787F3E1F2D54A3
010000070DEBFC184C1EFE7CF8B4
2AC580")
110 CALL SPRITE(#1,N,9,N,N):
: CALL MAGNIFY(4)
120 CALL KEY(1,K,S):: CALL J
OYST(1,X,Y)
130 IF K=5 THEN Y=4 ELSE IF
K=0 THEN Y=-4 ELSE IF K=2 TH
EN X=-4 ELSE IF K=3 THEN X=4
 ELSE IF K=6 THEN X,Y=4 ELSE
 IF K=4 THEN X=-4 :: Y=4 ELS
E IF K=15 THEN X,Y=-4 ELSE I
F K=14 THEN X=4 :: Y=-4
140 CALL MOTION(#1,-Y*10,X*1
0):: GOTO 120
```

TECHO TIME

This is just to remind you all that the September meeting is going to be our 32K memory upgrade day.

Those wishing to take advantage of this project should make sure that you have all the equipment as listed in last months magazine, as well as sufficient funds to cover the cost of the kit.

For next month I will provide a circuit diagram for a colour difference to composite video signal circuit (similar to the old TI modulator) to use with the video signal buffer in the March '86 magazine.

Technical manuals (which include circuits of the computer and expansion system) are available for $15 at club meetings or from the club Librarian (please add $3.50 postage & handling).

If anyone has a technical problem or wishes to contribute technical articles then contact me (ROBERT) on -
602-4168 between 5.00-8.30 PM
(PLEASE STICK TO THESE TIMES)
or send a letter to

TECHO TIME
P.O. BOX 149
PENNANT HILLS 2120

ANYONE WISHING TO OBTAIN INFORMATION, PLEASE SEND A SELF-ADDRESSED ENVELOPE (WITH SUFFICIENT POSTAGE STAMPS) TO THE ABOVE ADDRESS.

All members who can offer assistance at the September Techo day should see me at the beginning of the meeting please.

TALKING TO YOUR PRINTER

By Ed Machonis
(QB 99'er Newsletter)

```
100 OPEN #1:"PIO"
110 INPUT A$
120 PRINT #1:A$
130 GOTO 110
```

Ed goes on to elaborate with this 10 line program:

```
1 PRINT ::::"TO INDENT TEXT
OR TO USE A COMMA, BEGIN &
END THAT LINE WITH QUOTE MARKS"::
2 INPUT "PRESS ENTER TO SKIP A LINE.

        HOW WIDE?(80 CHARACTERS MAX):
WIDTH
3 MARGIN=INT((80-WIDTH)/2)
4 OPEN #1:"PIO"
5 INPUT "INPUT A LINE OF TEXT:":TEXT$
6 IF LEN(TEXT$)>WIDTH THEN 7 ELSE 9
7 PRINT :"LINE TOO LONG! SHORTEN TO"::
WIDTH;"CHARACTERS MAX.":SEG$(TEXT$,1,
WIDTH)::
8 GOTO 5
9 PRINT #1:TAB(MARGIN);TEXT$
10 GOTO 5
```

```
100 T=4 :: X=96 :: CALL CLEA
R :: CALL CHAR(X,"2C5EFF3F4F
1D827C"):: CALL SPRITE(#T,X,
2,X,X)
110 CALL KEY(1,K,X):: CALL J
OYST(1,X,Y):: IF K=5 THEN Y=
T ELSE IF K=0 THEN Y=-T ELSE
 IF K=2 THEN X=-T ELSE IF K=
3 THEN X=T
120 CALL MOTION(#T,-Y*T,X*T)
:: GOTO 110
```

```
100 T=4 :: X=96 :: CALL CLEA
R :: CALL SCREEN(T*T):: CALL
 CHAR(X,"000003173F6FFF7F360
C030000040300000C0E0F0F8F8F
C72E1C10618608"):: CALL SPRI
TE(#T,X,15,X,X):: CALL MAGNI
FY(T)
110 CALL KEY(1,K,X):: CALL J
OYST(1,X,Y):: IF K=5 THEN Y=
T ELSE IF K=0 THEN Y=-T ELSE
 IF K=2 THEN X=-T ELSE IF K=
3 THEN X=T
120 CALL MOTION(#T,-Y*T,X*T)
:: GOTO 110
```

```
100 T=4 :: X=96 :: CALL CLEA
R :: CALL SCREEN(15):: CALL
CHAR(X,"000003173F6FFF7F360C
030000040300000C0E0F0F8F8FC
72E1C10618608"):: CALL SPRIT
E(#T,X,T*T,X,X):: CALL MAGNI
FY(T)
110 CALL KEY(1,K,X):: CALL J
OYST(1,X,Y):: IF K=5 THEN Y=
T ELSE IF K=0 THEN Y=-T ELSE
 IF K=2 THEN X=-T ELSE IF K=
3 THEN X=T
115 IF K=6 THEN X,Y=T ELSE I
F K=T THEN X=-T :: Y=T ELSE
IF K=15 THEN X,Y=-T ELSE IF
K=14 THEN X=T :: Y=-T
120 CALL MOTION(#T,-Y*T,X*T)
:: GOTO 110
```

Come on kids, lets hear from you! This is your own section. If you don't write, then we have no Younger Set. Might soon.

Regards

Jenny

```
*****************************************
*                                       *
*            CALL SPRITE                *
*     PROGRAM PLACES A HELICOPTER       *
*    SPRITE IN MOTION BY ENABLING       *
*       INTERRUPTS. PRESS ANY           *
*    KEY TO ALTER MAGNIFICATION         *
*                                       *
*****************************************
       DEF  START
       REF  VMBW,VWTR,KSCAN
*
HELI   DATA >007F,>0000,>0107,>0E0E   HELICOPTER PATTERN DESCRIPTION
       DATA >1EBE,>FFBF,>0F07,>020F   BLOCK 2
       DATA >00FF,>8080,>C0F8,>04C2          3
       DATA >DACA,>FEFC,>F8E0,>40F8          4
SDATA  DATA >7080,>8008  INITIAL SPRITE DATA
       DATA >D000        >D0 PREVENTS GHOST SPRITES
*
SPEED  DATA >0A0F,>0000  SPRITE SPEED FOR AUTO MOTION
STATUS EQU  >837C        GPL STATUS BYTE
VDP    DATA >01E0        INITIAL VALUE OF VDP REGISTER 1
MYREG  EQU  >8300        MYREG IN 16 BIT HIGH SPEED AREA OF MEMORY
*
START  LWPI MYREG
       CLR  @>8375       KEYBOARD DEVICE = 0. SCAN ALL.
       MOV  @VDP,R6
       LI   R0,>0400     LOAD  (BASE ADDRESS OF SPRITE DESCRIPTOR TABLE)
       LI   R1,HELI           SPRITE
       LI   R2,32                  DESCRIPTOR
       BLWP @VMBW                        TABLE
*
       LI   R0,>0300     LOAD  (BASE ADDRESS OF SPRITE ATTRIBUTE TABLE)
       LI   R1,SDATA          SPRITE
       LI   R2,6                   ATTRIBUTE
       BLWP @VMBW                        TABLE
*
       LI   R0,>0780     LOAD  (BASE ADDRESS OF SPRITE MOTION TABLE)
       LI   R1,SPEED          SPRITE
       LI   R2,4                   MOTION
       BLWP @VMBW                        TABLE
*
       LI   R1,>0100
       MOVB R1,@>837A    ONE SPRITE IN MOTION
*
LOOP   CLR  @STATUS
       BLWP @KSCAN
       MOVB @STATUS,@STATUS HAS KEY BEEN PRESSED?
       LIMI 2            ENABLE INTERRUPTS FOR AUTO MOTION
       LIMI 0            DISABLE INTERRUPTS SO VDP IS NOT AFFECTED ON READ/WR
       JEQ  LOOP
*
CHECK  INC  R6           R6 IS USED AS A COUNTER TO KEEP
       CI   R6,>01E4     TRACK OF WHICH MAGNIFICATION
       JLT  GO           LEVEL (1 TO 4) WE ARE ON.
       MOV  @VDP,R6
*
GO     MOV  R6,R0        LOAD R0 WITH DATA TO LOAD INTO VDP R1
       BLWP @VWTR        CHANGE THE VDP REGISTER
       B    @LOOP
       END
```

* FOR EXTRA PRACTICE ADD A ROUTINE THAT SHOWS THE X AND Y POSITION OF THE SPRITE
* ON THE SCREEN AS IT MOVES. HINT: Y LOCATION IS 1ST BYTE IN SPRITE ATTRIBUTE
* LIST. X SECOND BYTE. READ THEM, CONVERT TO ASCII DECIMAL AND REDISPLAY WITH
* APPROPRIATE TEXT. WHO'LL BE FIRST?

* NEXT MEETING:

Saturday 2nd August
(2pm-4pm) at ...
•SHIRLEY HOUSE•
Ethel Street,
Burwood...
behind Woodstock.

See you there !!!

SAVE TUTORIAL

By Scott Darling

After reading some messages concerning using the Save utility in the Editor Assembler package I awaited one individuals tutorial on the subject. But none appeared. So I decided to sit down and write one for it. Now some of you may know how to do this, but evidently not everyone does; so here goes.

Obviously to use the Save utility, you need the E/A package. But a couple of programs are helpfull. A sector editor such as DISK-AID, DISKO, and the like. The Mini-Memory module is a nice shortcut for finding addresses.

USING THE SECTOR EDITOR

First off any D/F 80 file will work, compresed or uncompressed. You should copy the program to a clean disk to make it easier to find using the sector editor. Load the editor and search for the end of the program. Disk-Aid will tell you where the last sector of a program lies. Toggle to Ascii and look at the sector info. There should be some names and references to VSBW, VMBW, START and so forth. Compressed code is easier to read by the way. Look carefully to see if the words SFIRST, SLAST, or SLOAD are there. IF they are skip the rest of this and go directly to "USING THE SAVE PROGRAM"!!

If those words are not there. Look to see if there are any lines of code that look like the actual program. The below will serve as an example. It is non-relocatable code. The "9" stands for that. An "A" stands for relocatable. (much easier to work with by the way) (this code was loaded into E/A editor, it will look different in sector editor)

```
0023 9E1D8B03807FD91F
0024 1E0467FEB9F
0025 6E046START 7FD06F
0026 :       99/4 AS
```

You will notice that at the line 24 there is a "1" to designate an auto start program. Change the "1" to a "F". This will defeat auto-start. There may be a "2" there, it also means auto-start.

Directly after that 1 or 2 is an address. This is the "START"ing address of the program. SFIRST and SLOAD will equal this address. The starting address also appears in front of the word "START". As is the case with most "DEFS" in this location.

To determine the SLAST is a little trickier. Just before the 1 or 2 auto-start was an address >E1D8. Now count over 1 byte for each 2 letter number combos. Excluding the B and 7. In other words: 9E1D8_0380_FD91_.

E1D8 9 A B C = E1DC This will make SLAST = E1DC Another example is for RELOCATABLE code:

```
0023 A01D8B03807FD91F
0024 100007FEB9F
0025 60000START 7FD06F
0026 :       99/4 AS
```

In the above example I have changed a few numbers to show what relocatable code might look like. You will notice the "1" for auto-start is still in place, but there now are "0000" instead of and address. But 0000 is an address. It means that no matter what the lead address is, that the start of the program is "x000" So SFIRST equals "A000" and SLOAD is the same. Now slast will work the same as above except this time it equals A0DC.

Confused?? I hope not. If so reread this before continuing!!

ADDING AN ASSEMBLY DEF PROGRAM

I hope that you realize that there was a reason for all of the above jumble! Be- cause now you are going to use those address to build a short ASSEMBLY program!

Type in the following:

```
       DEF SFIRST,SLOAD,SLAST
SFIRST EQU >A000
SLOAD  EQU >A000
SLAST  EQU >A0DC
       END
```

Save this program using whatever name strikes your fancy. Load the assembler and assemble using another name. Bypass list, and bypass options. It will take a whole 2-3 seconds to assemble and you are ready for the next step. This source code can be used indefinitely. Just plug in the appropiatte addresses.

USING THE SAVE PROGRAM

1) Bring up the "LOAD & RUN" option of the E/A module.

2) Load your non auto-start program.

3) Load the "DEF" program that you wrote. (unless the program already has the DEFs in the program)

4) Load the E/A "SAVE" utility

5) Hit enter to "PROGRAM NAME" and hit enter

6) The next prompt will ask for a name. Remember that it will increment the name the next ASCII character if the resulting is going to be larger than 8K (33 sectors), plan accordingly.

7) Now hit enter, and try running your NEW program in "RUN PROGRAM FILE" option.

If it runs you are home free. If not you may want to go back and check to see if used the correct addresses. Most of what I wrote above was trial and error on my part, so experiment and play around with the programs.

The above was a rather short description. There is a shortcuts to find addresses that uses the Mini-Memory.

Initiate the M-M, then load the program. If it starts running you know that you have to defeat the auto-start feature. Escape the M-M and bring up Easybug. Do "M7020" and look at page 74 of the M-M manual. 7020 is default entry address. Or SFIRST and SLOAD. Step down to 7024 and this is the last free address in high memory or the end of the program. IF 7024 is "0000" then you know that the program is probably absolute code.

My only dissapointment is determining how to save low and high memory code. If a program loads at >2000 to >4000 then continues at >A000. The above steps are useless, because it assumes a continuous count between two addresses. I have not figured out or taken the time to attempt this format.

Forth.....

The following is verbatim from CorComp Cursor, a newsletter to TI-99/4 user groups. When I called to see if they minded if I put this up here I was told the intent of the newsletter was that it be given the widest dissemination possible. So here is what they say about TI-Forth and DSDD (and other) set-ups. Hope you find it useful.

Rich Stanford

TI FORTH

This article is intended for all TI FORTH users who have (or plan on having) double density and/or double sided disk capabilities. While the techniques described should work with any disk controller capable of double density, the author's CorComp 9900 Disk Controller card is the only one that has been tested. The purpose of this article is to illustrate both how to access the additional screen capacity and how to modify the FORTH words and disc to be compatable with the new format and Disk Manager.

Throughout this article lowercase letters used in a FORTH definition will indicate a variable value to be entered. The following terms will be used to refer to the various formats a FORTH disc may have.

90 SCRN or SSSD	- the original 90 screen single sided single density format.
180 SCRN	- either a SSDD or DSSD disk when comment applies to both.
360 SCRN or DSDD	- a double sided double density disk.
SSDD	- a single sided double density disk.
DSSD	- a double sided single density disk.

The first step is to use Disk Manager to format (initialize) a 180 or 360 SCRN disk. Next, you must copy FORTH from the 90 SCRN disk to the new 180 or 360 SCRN disk. The disk copy feature of CorComp's Disk Manager will d this properly for you. If you have two drives, the FORTH-COPY word in the -COPY screens will also do it properly (do 0 DISK_LO ! first). However, if you are using TI's Disk Manager II, after copying the three files you must use FORTH to copy screens 1 to 9 because Disk Manager II puts them in the wrong place! To do this, enter the following for each of the nine screens.

```
n BLOCK UPDATE (where n is the screen number to be read from old
                disk)
FLUSH          (after inserting the new disk - note: up to five
                screens may be entered at a time)
```

Now edit screen 3 of your new disk and add the following commands:

```
x DISK_SIZE !  (where x=180 or 360 as appropriate)
y DISK_HI   !  (where y=x times 1, 2 ,3 or 4 depending on the
                number of drives you have)
```

Unfortunately, TI FORTH does not provide a method for configuring each drive individually. Therefore, the user must be cognizant of which screens are available on each drive when there are differences between them.

At this point, FORTH can be booted and it will recognize the full capacity of your 180 or 360 SCRN disk. You can create, edit, list, and load from screens greater than 89. However, neither Disk Manager nor FORTH-COPY will recognize this disk as having more than 90 screens. To fix this problem you must modify the -COPY screens (39 and 40), the disk header (sector 0) and, the SYS-SCRNS file header (sector 4).

First edit screen 39. Change the value 90, which appears once in DTEST and twice in FORTH-COPY to 180 or 360 as appropriate. Next, edit screen 40 as follows:

Line 3 - change 168 to 2D0 for 180 SCRN or 5A0 for 360 SCRN.

Line 4 - change 944 to 1244 for SSDD or DSDD (no change for DSSD).

Line 5 - replace entire line with:

DUP 10 + 2028 SWAP ! DUP 12 + a SWAP ! DUP 14 + 24 0 FILL

where a = 0201 for DSSD, 0102 for SSDD, or 0202 for DSDD.

Line 10 - change 165 to 2CD for 180 SCRN, or 59D for 360 SCRN.

Line 13 - change 4016 to C02C for 180 SCRN, or C059 for 360 SCRN.

Next edit screen 33 to modify the FORMAT-DISK word to:

: FORMAT-DISK 1 + a 33616 ! 18 SYSTEM ;

where a = 258 for DSSD, 513 for SSDD, 514 for DSDD.

Finally, you need to create a word that will modify the header sectors on your new disk. This word only needs to be executed once since copies of this disk, once it's modified, will not require modification. Here is the way to do it:

```
HEX 0 DISK_LO !              ( removes disk fence)
:  DD-FORTH 0 BLOCK UPDATE   ( read screen 0 and mark as updated)
      DUP A + a SWAP !       ( a = 2D0 for 180 SCRN, 5A0 for 360 SCRN)
      DUP C + b SWAP !       ( b = 944 for DSSD, 1244 for SSDD or DSDD)
      DUP 10 + c SWAP !      ( c = 2028 for all versions)
      DUP 12 + d SWAP !      ( d = 201 on DSSD, 102 on SSDD, 202 on DSDD)
          38 + C8 FF FILL    ( flag all sectors as in use)

          1 BLOCK UPDATE     ( read screen 1 and mark as updated)
      DUP E + f SWAP !       ( f = 2A0 for 180 SCRN, 570 for 360 SCRN)
      DUP 1C + g SWAP !      ( g = 4D20 for 180 or 360 SCRN versions)
      DUP 1E + h SWAP !      ( h = 2805 for 180 SCRN, 5205 for 360 SCRN)
          20 + i SWAP !      ( i = F029 for 180 SCRN, F059 for 360 SCRN)
   FLUSH ;                   ( write modified screens to disk)
DECIMAL DD-FORTH
```

Now your new high capacity copy of FORTH is fully compatable with Disk Manager, the FORTH format, copy, test, and header words and your double density and/or double sided drives and controller. Enjoy!

****

Special thanks to Jim Vincent who is the FORTH columist for the 99'ers Users Group Association.

```
90 REM SLANTED UPPER CASE LE
TTERS, NUMBERS, PUNCTUATION
100 DATA 33,000404080808001,
35,0009127F24FE489,36,001F24
283E0A127C,40,00010204080808
04,41,002010101020408,43,000
004083E081
110 DATA 48,001F11222222447C
,49,0001010202020404,50,001F
01023E20407C,51,001F01023E02
047C,52,001111223E020404,53,
001F10203E02047C
120 DATA 53,001F10203E22447C
,54,001F10203E22447C,55,001F
010202020404,56,001F11223E22
447C,57,003F21427E02047C,58,
00000C0C001818
130 DATA 59,00000C0C0018081,
63,000F11020E08001,65,001F11
223E224444,66,001E11213E2242
7C,67,001F10202020407C,68,00
1E112121222427C
140 DATA 69,001F10203E20407C
,70,001F10203E204040,71,001F
10202E22447C,72,001111223E22
4444,73,001F04080808107C,74,
000101020222447C
150 DATA 75,001111223C284444
,76,001020202040407C,77,0037
294A52428484,78,0011112A2A2A
4444,79,001F11222222447C,80,
001F11223E204040
160 DATA 81,001F1122222A447A
,82,001F11223E284444,83,001F
20203E02027C,84,001F04080808
1010,85,001111222222447C,86,
0011111222242438
170 DATA 87,002121424A5294EC
,88,0011110A0C142222,89,0011
11223E081010,90,001F02041820
407C
180 FOR CH=1 TO 45
190 READ CHN,CH$
200 CALL CHAR(CHN,CH$)
210 NEXT CH
220 INPUT M$
230 GOTO 220
```