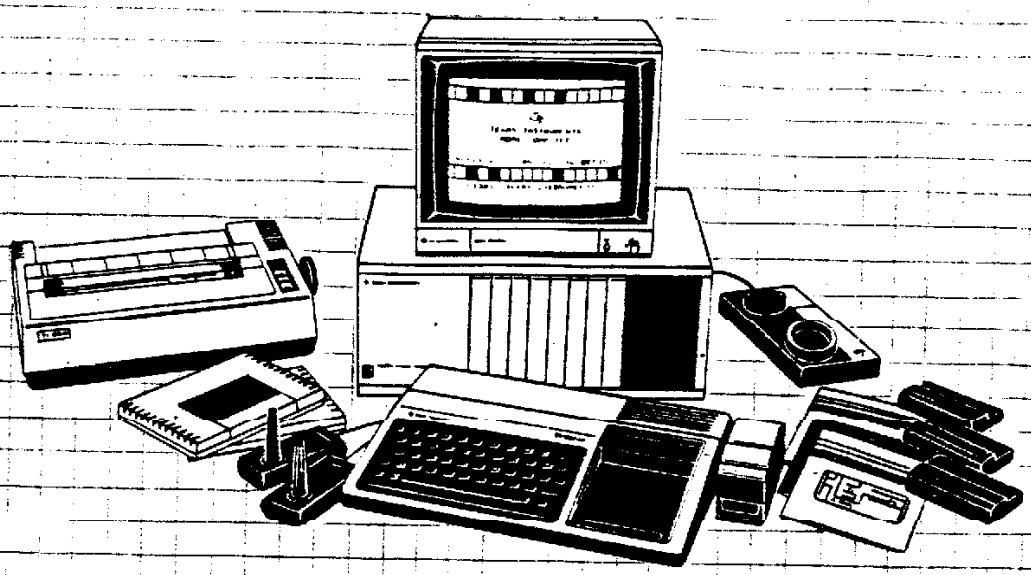
July 85

HUNTER VALLEY 99'ERS NEWS

TI 99/4A

volume.1 No.2



IN THIS ISSUE

MEETING DATES

BEGINNERS BASIC EACH TUESDAY
ASSEMBLER LAST TUE EACH MONTH
MONTHLY MEETINGS 9-7-85
13-8-85
10-9-85
ALL MEETINGS COMMENCE AT 7PM.
FINISH WHEN THE LAST MEN FALL
PHONE COMMITTEE FOR MORE DATA

NEWSLETTER OF HUNTER VALLEY TI99/4A USERS GROUP

TEXAS INSTRUMENTS

****
* SECRETARYS NOTES *
****

Good News! We have been informed that we can stay in the Community Centre until the end of the year, this will give the Committee another six months to find a suitable building to house our group.

I would like to take this opportunity to say well done and thank you all, to the people who contributed to our first edition of HV99ers Mag.

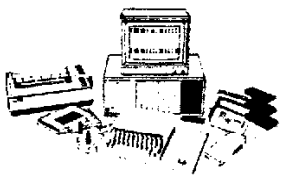
A special thanks goes to Steve Taylor, Joe Wright and Brian Woods for putting the time and effort into producing a High Class Magazine. The feed back and congratulations that are pouring in shows the need for such a Magazine in our area.

The last two Main Meetings were yet again a huge success, if you missed these, you missed Jim Threadgate showing us how easy it is to communicate via Radio, jumping in and out of mail boxes around Aussie Jim tells us he's made contact with other radio buffs around the world with T.I.s and the information he gets will be passed on. well done Jim.

At the second meeting we had Dave Winton with FORTH and a GAME DEMO called ACTUARY that was very interesting, again many thanks David.

Our beginners crash course in BASIC is going now, also the FORTH group is up and running, so if you require any information on these and other titbits give me or your Commitee members a ring, see you all at our next general meeting where once again there will be more interesting software demos.

PETER.C. SECRETARY HV99ers.



YOUR COMMITTEE 1985

A.WRIGHT	PRES.	PH.468120
P.COXON	SECT.	PH.751930
D.RUTHERFORD	TRES.	PH.498184
A.LAWRENCE	LIBR.	PH.486509
S.TAYLOR	EDIT.	PH.487076
A.BYRNE	TECH.	PH.499520
T.MCGOVERN	TECH.	PH.523162
B.MAC.CLURE		PH.437431
B.WOOD		PH.662307
D.WINTON		PH.591882

Editorial

We only just seemed to have put the finishing touches To Volume 1 No.1 and here we are again with issue No.2. Firstly I would like to thank everyone for the kind remarks on our first effort. It is now our aim to improve with each issue, I like to think that what we lack in expertise we more than make up for with enthusiasm,and hopefully work permitting, we should be able to produce an issue of the magazine every six to eight weeks with a "Bumper" issue around the Christmas period.

Since our first edition of HV99'ers NEWS many new members have joined us and we extend to you a very warm welcome to our ever growing group of TI-99/4A users inthe Hunter Valley With winter now well upon us I expect you are all putting your computers to good use to help while away the cold winter nights. In this issue I feel we have struck a balance to suit most members, both those with just the basic system as well as those with fully expanded outfits. In this issue Garry Jones returns with more pertinent infomation which I'm sure we are all finding interesting reading. Garry is well known to HV99'ers for his weekly BASIC classes. Brian Rutherford has contributed a brilliant MINI WORD PROCESSOR program that only requires a basic system to get up and running. Now there's no excuse for not contributing to the magazine !!

Next issue Brian will present the second part of the program a FORMATTER. Those wondering how Brian managed to produce such a powerful utility from such a small program wil be delighted to know that in the future he intends completely dissecting the program and explaining it in minute detail.

Joe Wright our hard working PRESIDENT has contributed the first in a series of articles on SORTING ROUTINES which will be of assistance to all. Tony MC.Govern is back again with two articles, which I'm sure you will find,as with all Tony's contributions extremely readable

Plus all the regulars like Library News By Al Lawrence, a Software review by Darren MacLure, FORTH REPORT with an update on what is happening in the Forth interest Group and much much more. Happy Reading.

ED.

****
* EARLY BASIC SERIES *
****

The following explanations are for words you may find in the USERS REFERENCE GUIDE or THE BEGINNERS BASIC Book and may have trouble comprehending.

PERIPHERALS- Anything which is attached to the Computer. eg.Printer,Speech synthersizer,Expansion Box etc.

ASCII CODE- ASCII means American Standard Code for Information Interchange. The Code is a different numeric value for all Alphas, numbers and symbols.

BINARY- The Computer's Alphabet. Because the Computer is only a large box of switches and all depending on how these switches are set in Memory is how the Data is stored, calculated and processed. Thus as a switch is only an ON or OFF device and the Computer is similar in operation, the Computer's alphabet needs only two characters. These are "1" and "Ø" and the alphabet is known as BINARY.

BYTE- Usually 8 bits or Binary numbers to make up a byte. The Computer is often rated in the amount of storage of the bytes in R.A.M. eg. 16,384 bytes or 16K for the T.I.99/4A.

CURSOR- The flashing square which indicates where the next character will appear on the screen, also indicates the Computer is ready to use or accept data.

DATA- The information handled or produced by the Computer.

HEXADECIMAL- Is a 16 number base system (Ø-9 and A-F) which is a shorthand way of writing binary.
eg. ØØ11Ø=6,1Ø11=B,1111 ØØ11Ø=F6

INTERGER- A whole number. If used as a function this will round off to the lowest whole number.
eg. 1.5=1, 1.99=1, -3.1=-4, -4.9=5

PROGRAM- A set of instructions which tell the Computer how to perform a Task.

STRING- A series of Alphas, numbers and symbols treated as one unit inside quotation marks.

NULL STRING- A string which contains no characters and has a zero length.

COMMAND- An instruction performed immediately with no line number needed.
eg. RUN, NEW, OLD, SAVE, PRINT

STATEMENT- AN instruction that is preceeded by a line number.
eg. GOTO, GOSUB, IF/THEN

FUNCTION- Allows a number of steps to be achieved under a single instruction.
eg. ASC< INT< SGN< SIN< SQR

PROGRAM LINE- A line that contains a statement or statements for the Computer's operation.

HARDWARE- The following harware devices make up a Computer System: MONITOR, DISK DRIVE, PRINTER, MEMORY EXPANSION.

SOFTWARE- Any type of program used by the Computer which includes programmes in R.O.M.

LOOP- Repeats a part of a program a number of times, usually a specified number.

INPUT- Is data that is being accepted from an external device or the act of accepting data from an external device.
eg. KEYBOARD, MODEM, CASSETTE.

OUTPUT- Is data being supplied by the Computer or the action of supplying data from memory onto a screen, printer, cassette, or disk.

BUG- An error or defect in a program which causes a mis-operation or halt to the program.
eg. Debugging is removing such faults.

SUBPROGRAM- A proceedure usually stored in the R.O.M. of the Computer and is accessible via a CALL instruction.
eg. CALL CLEAR,CALL HCHAR,CALL VCHAR

SUB ROUTINE- Is part of a program that has been loaded into the Computer which can be used many times. Is accessible via GOTO, GOSUB, and RETURN.
eg. complex calculations.

CONT

R.A.M.- Random Access Memory. Part of memory where programmes or your instructions are stored. If T.I. Basic is exited or when power is removed the data in this memory will be erased.

R.O.M.- Read Only Memory. Part of the memory where instructions to drive the Computer are stored. Data is permanently stored, even when power is turned off.

VARIABLE- A value that can be changed during the running of a program.
eg. TOTAL=TOTAL+SUBTOTAL

RESERVED WORD- A word which is used as a statement, command or function and can't be used as a variable name.

BY GARRY JONES HV99ers.

****
TEXAS INSTRUMENTS RECORD YEAR 1984
****
Semiconductor and one time TI99/4A manufacturer TEXAS INSTRUMENTS put in a record year in 1984 with sales of $US5.7 billion and a net profit of $US316 million. The company's profit followed a 1983 loss of $US145.4 million, supposedly brought about by problems in the home computer market and the subsequent withdrawal from that market.

ENTOMOLOGY CORNER

from

FUNNELWEB FARM

Now funnelwebs aren't exactly the nicest critters to be found around Funnelweb Farm, but they do have the virtue that if you don't bother them they leave you alone too. Unfortunately the bugs that infest Extended Basic aren't nearly so accomodating in keeping out of the road in the first place, and insist on making their presence felt.

I have looked at a few in previous XB tutorials where they came up naturally in the subject matter, mostly in ACCEPT AT. From now on Entomology Corner will look at XB bugs as they come to light. This time we have two beautiful specimens. The first of these was exposed in the Spring 85 issue of TI*MES from the UK by John Bingham. To see it at work, enter the following little program

```
100 I=1 :: IF I=1 THEN J=1
: GOSUB 200 ELSE K=1 :: J=2
110 PRINT K;J :: STOP
200 RETURN
```

Before you run this, predict what it is going to print out ! Then run it and see what you get. Next reverse the order of the statements between THEN and ELSE and run it again. Now it should work the way you expected, K=0 and
J=1. If your XB gets it right the first time, it's different from the one I have. The presence of the GOSUB just before ELSE seems to have upset XB's mechanism for keeping track of ELSE, and the program has gone ahead and ignored the ELSE and the statement after it and executed the following statement which it should have ignored entirely while proceeding to the next line. Try substituting a dummy SUBprogram CALL for the empty GOSUB. XB then works just as expected. Yet another reason for using SUBprograms instead of GOSUBs.

The XB manual lays down a few prohibitions on what can go into IF doesn't mention this little beauty. It does seem, despite the warnings, that FOR..NEXT loops can follow the final ELSE without problems, but this usage is not to be recommended as it may not hold good for all XB modules.

I must admit that reading this news had me a little worried, as I have written long and thoroughly debugged XB programs with some tricky IF..THEN..ELSE footwork, and had never picked up this problem. How come ? The first saving grace is that the Tutorial advice I gave is for real, and I use very few GOSUBs and very many SUBprograms unless I am absolutely desperate for more bytes. This was frequently the case in the writing of TXB, and the central SUBprogram, one of 12 in the program, itself contains 12 subroutines written in to save bytes. Careful study of the code for TXB showed that none of the GOSUBs was written in a way that would let this evil bug loose. When you look back at something like this, you wonder whether you had scrapped particular

pieces of code that never quite worked properly, for entirely wrong reasons.

The second bug mentioned has not yet been fully explored. It showed up in a CALL LINK from XB to an assembler routine which also passed in a string variable to be examined by the routine. It happened in the COLIST program, the all-singing all-dancing version of the SIMPLIST program which appeared in a XB Tutorial in those earlier days in SND. The symptoms were that the machine crashed utterly when it was printing out a line of its own listing, after it had already printed several hundred lines, many quite similar to the one that caused to the trouble. Not just an error caught and reported by XB, but a full blown paralytic seizure. It turned out after some TRACE work to be in the very line that was being processed for printing, and to be associated with the LINK name being at a particular position in the line of text being passed in with STRREF. The problem seems to be CALL LINK extending its link name search over places it shouldn't orter, but more research is needed. Further reports in Entomology Corner in a future issue.

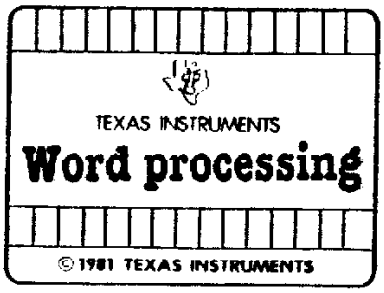
One disappointing discovery came up in the bug hunt. I disassembled the XB machine code utilities loaded by CALL INIT to see if the problem was in STRREF. The good news is that that code looks OK, but the bad news is that STRREF reads strings out of VDP in the same slow way that the console does, 1 byte at a time, resetting the VDP addresses each time. This is the tortuously slow way GPL does it because the console has hardly any CPU RAM, but it scarcely seems necessary in STRREF which can only be used when there is expansion RAM present.

Another bug in CALL LINK has been reported in the US of A in some older XB modules, I suspect prior to the models of V11Ø sold in Australia. This comes when the link name is supplied as a string variable, as in CALL LINK("A$"). If a garbage collection is performed in VDP RAM by the XBasic interpreter in between assigning A$ and using it in CALL LINK, this routine would lose track of where A$. I have not encountered this bug myself, but I will try to stir it up in the XB modules I have. This reported bug brings to mind the strange state of affairs in XB where it is possible to DELETE a file by string variable reference but not to RUN a program file by similar reference, leading to a smal cottage industry of ways around this deficiency.

While we are in the business of making corrections to all and sundry, there is a minor one to last month's issue. Nothing substantive but only to emphasis of a comment made in passing. That was in respect of finding the DIMensions of an array passed as an argument by CALL LINK. It is defined for E/A Basic in the stack entry built by LINK, but you have to extend this to XB by implication along with a lot of other poorly defined items. The discussion does not tell how this stack entry relates to the internal variable table.

For this issue Entomology Corner will also serve as the Extended Tutorial, and the regular subject matter will be resumed next time. The price you pay for getting the Tutorials written is that you have to put up with my whims of the moment. I have been hard at work on running TI-Writer from XB. An early version is available at meetings or from Funnelweb Farm as a public domain program. Further developments, not yet stabilized enough for general release, include a Formatter smart enough to know the file-name last used in the Editor before switching to the Formatter, and a Show Directory that works from within the Editor, just like the real thing except better.

A significant measure of the worth of a User Group is the range and quality of Public Domain software produced by the members. The HV99 group intends to have a high profile in this area, both in programs released and in original items published in the HV99 News. We are aiming to be one of the groups with a solid and useful output of original material, along with thoroughly evaluated material from other like minded groups. I don't see us reprinting much from TISHUG Newsdigest on its recent form. Glossy covers and two page graphics spreads will rank near the bottom of our list of priorities.



Oh, so easy WP

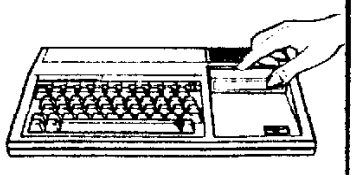
MINI WORD PROCESSOR.

This small word processor is designed for people who only have the console and a cassette recorder. This is the word processor part, the formatter programme will be printed in next months magazine (all going well). The main idea is for you to type in your article for the magazine and save it on tape, you then give a copy to the editor, who has TI WRITER. He then uses the formatter programme to convert your cassette files into files the TI WRITER can read, and print out to.

After you have typed in the programme and run it, the main menu is displayed with all the options plus the words "free 8000". That is the number of characters you can input before the memory is full. That figure is constantly updated as you input text or edit text. Any time you need to know how much more you can type in, you only need to go back to the main menu by typing // at the begining of a new line. You may type in five lines at a time before you need to press ENTER. If you are part way through a word at the finish of the fifth line do not hyphenate the word just carry on typing it when you start the next line. Also if you finish a word or leave a space at the end of the fifth line, leave a blank at the start of the next line, otherwise the formatter will join the two words together. To leave a blank line just press enter to input a null string. It is a good idea to always leave a blank line between paragraphs so the formatter does not get itself mixed up. To change a line with the change line option, the ACCEPT with out AT is used, along the lines of my article in the June magazine. That is you have to move the cursor over the parts of the line you want using the FCTN D key, to enable the computer to read what is on the screen. The other options lead you through as you use them. The options change line, delete line and insert line allow you to opt out of that mode if you realise you have pressed a wrong key. By specifying a text line number that is invalid. i.e. line number zero or a number higher than you are up to. Also change line and delete line show you the line first and ask yes/no, which is another failsafe for you. The change word option allows you an out by typing a null when the prompt OLD WORD is shown. When you save your files the computer tells you how many lines there are to save, I advise you to write that number down on the cassette that you save your file on, as that is the first thing you will be prompted for when you reload the file for any reason. The screen will also show you what line it is saving or reading, so you know where it is up to. Nothing is more irritating with cassette files is not knowing where they are up to. Use tapes with about ten minutes a side if you have long files. Lastly if you find ANY BUGS please let me know, as it is not always easy to think of all the possible situations that can go wrong with a programme of this type.

Brian R.

```
100 REM !~~~~~~~~~~~~~~!
110 REM !     MINI     !
120 REM !WORD PROCESSOR!
130 REM !      BY      !
140 REM !   BRIAN R.   !
150 REM !______________!
160 OPTION BASE 0 :: DIM L$(
66)::  DISPLAY ERASE ALL :: F
OR L=0 TO 12 :: CALL COLOR(L
,16,1):: NEXT L :: L=0 :: L$
(0)="0000"
170 CALL SCREEN(5):: CALL ME
N(L$()):: CALL CH(9,K)
180 ON K-48 GOSUB 200,210,22
0,230,240,250,260,270,280
190 GOTO 170
200 CALL IN(L$(),L):: RETURN
210 CALL CL(L$(),L):: RETURN
220 CALL DL(L$(),L):: RETURN
230 CALL INS(L$(),L):: RETUR
N
240 CALL CV(L$(),L):: RETURN
250 CALL SCREEN(14):: CALL S
F(L$(),L):: RETURN
260 CALL SCREEN(14):: CALL R
F(L$(),L):: RETURN
270 CALL PF(L$(),L):: RETURN
280 CALL SCREEN(7):: DISPLAY
 AT(10,8)ERASE ALL BEEP:"1 P
urge file": :"       2 Exit"
 :: CALL CH(2,K):: IF K=49 T
HEN L=0 :: L$(0)="0000" :: R
ETURN
290 DISPLAY AT(10,1)ERASE AL
L:"HAVE YOU SAVED YOUR FILE
Y/N"
300 CALL KEY(3,K,S):: CALL S
OUND(-10,110,5):: IF K=78 TH
EN RETURN ELSE IF K<>89 THEN
 300
310 DISPLAY ERASE ALL :: STO
P

320 SUB MEN(L$()):: DISPLAY
AT(1,12)ERASE ALL:"MENU" ::
DISPLAY AT(2,11):"_____" ::
 DISPLAY AT(4,8):"1 Add text
": :"       2 Change line"
330 DISPLAY AT(8,8):"3 Delet
e line": :"       4 Insert l
ine": :"       5 Change word
": :"       6 Save file": :"
      7 Load file"

340 DISPLAY AT(18,8):"8 Prin
t file": :"       9 Purge fi
le & exit": :"
  free ";L$(0):: SUBEND

350 SUB CH(X,K):: DISPLAY AT
(23,2)BEEP:"Your choice"
360 CALL KEY(3,K,S):: IF S<1
 THEN 360 ELSE IF K>48 AND K
<(X+49)THEN SUBEXIT
370 DISPLAY AT(24,1)BEEP:"A
NUMBER BETWEEN 1 AND";X :: G
OTO 360
380 SUBEND

390 SUB LI(A,B,A$,B,L):: DIS
PLAY AT(A,B):A$
400 ACCEPT AT(A,19)VALIDATE(
DIGIT)BEEP:A$ :: IF A$="" TH
EN 400 ELSE B=VAL(A$)
410 IF B<1 OR B>L THEN DISPL
AY AT(23,1):"<<< No such lin
e >>>" :: CALL KC :: B=0420
SUBEND

430 SUB KC :: DISPLAY AT(24,
1)BEEP:"PRESS ANY KEY TO CON
TINUE"
440 CALL KEY(3,K,S):: IF S<1
 THEN 440
450 SUBEND

460 SUB IN(L$(),L):: CALL KE
Y(5,K,S):: DISPLAY AT(15,1)E
RASE ALL BEEP:"Type // to le
ave input mode" :: PRINT L$(
L)
470 L=L+1 :: LINPUT "":L$(L)
:: IF L$(L)="//" THEN L$(L)=
"" :: L=L-1 :: SUBEXIT
480 IF L=66 OR VAL(L$(0))<0
THEN PRINT "<<< MEMORY OR AR
RAY FULL >>>" :: CALL SOUND(
500,110,10):: SUBEXIT
490 L$(0)=STR$(VAL(L$(0))-LE
N(L$(L))):: GOTO 470
500 SUBEND

510 SUB CL(L$(),L):: DISPLAY
 ERASE ALL :: CALL LI(23,8,"
Which line",B,L):: IF B=0 TH
EN SUBEXIT
520 CALL LC(L$(),D,K):: IF K
=78 THEN SUBEXIT

530 LE=LEN(L$(D))/28 :: IF L
E<>INT(LE)THEN LE=INT(LE)+1
540 CALL KEY(5,I,K):: DISPLA
Y AT(16,1)ERASE ALL BEEP:STR
$(D);"!-":L$(D):: P=1 :: A$=
""
550 FOR I=1 TO LE :: P$=SEG$
(L$(D),P,28):: DISPLAY AT(24
,1):P$ :: ACCEPT P$ :: A$=A$
&P$ :: P=P+28 :: NEXT I
560 IF LEN(A$)>191 THEN DISP
LAY AT(16,1)ERASE ALL BEEP:"
LINE TO LONG TRY AGAIN" :: C
ALL KC :: GOTO 540
570 DISPLAY AT(16,1)ERASE AL
L BEEP:A$ :: DISPLAY AT(24,1
):"Is change OK? Y/N"
580 CALL KEY(3,K,I):: IF K=8
9 THEN L$(0)=STR$(VAL(L$(0))
+LEN(L$(D))):: L$(D)=A$ :: L
$(0)=STR$(VAL(L$(0))-LEN(L$(
D))):: P$,A$="" :: SUBEXIT
590 IF K<>78 THEN 580 ELSE 5
40
600 SUBEND

610 SUB DL(L$(),L):: DISPLAY
 ERASE ALL :: CALL LI(23,8,"
Which line",B,L):: IF B=0 TH
EN SUBEXIT
620 CALL LC(L$(),B,K):: IF K
=78 THEN SUBEXIT
630 L$(0)=STR$(VAL(L$(0))+LE
N(L$(D))):: FOR I=B TO L-1 :
: L$(I)=L$(I+1):: NEXT I ::
L$(L)="" :: L=L-1 :: SUBEND

640 SUB INS(L$(),L):: DISPLA
Y ERASE ALL :: CALL LI(23,1,
"Before which line",B,L):: I
F B=0 THEN SUBEXIT
650 L=L+1 :: FOR I=L TO B+1
STEP -1 :: L$(I)=L$(I-1):: N
EXT I :: CALL KEY(5,I,S):: L
INPUT "":L$(B):: L$(0)=STR$(
VAL(L$(0))-LEN(L$(B))):: SUB
END

660 SUB LC(L$(),B,K):: DISPL
AY AT(1,1)ERASE ALL BEEP:STR
$(B);"!-":L$(B):: :"This line
.....Y/N?"
670 CALL KEY(3,K,S):: IF K<>
78 AND K<>89 THEN 670
680 SUBEND

690 SUB CV(L$(),L):: CALL KE
Y(5,I,J)
700 DISPLAY AT(16,1)ERASE AL
L:"Old word": :"New word" ::
 ACCEPT AT(16,10)BEEP:OW$ ::
 IF OW$="" THEN SUBEXIT ELSE
 ACCEPT AT(18,10)BEEP:NW$
710 CALL LI(20,8,"From line"
,SL,L):: IF SL=0 THEN 710
720 CALL LI(22,8," To  line"
,EL,L):: IF EL=0 THEN 720
730 LE=LEN(OW$):: FOR I=SL T
O EL :: IF LEN(L$(I))<LE THE
N 790 ELSE A=1
740 P=POS(L$(I),OW$,A):: IF
P=0 THEN 790 ELSE IF P=1 THE
N 750 ELSE IF SEG$(L$(I),P-1
,1)<>" " THEN 760
750 CH$=SEG$(L$(I),P+LE,1)::
 IF CH$="" OR CH$=" " OR CH$
="." OR CH$="," OR CH$=";" T
HEN 770
760 A=1+P :: IF A>LEN(L$(I))
THEN 790 ELSE 740
770 CH$=SEG$(L$(I),1,P-1)::
C$=SEG$(L$(I),P+LE,LEN(L$(I)
)):: L$(0)=STR$(VAL(L$(0))+L
EN(L$(I))):: L$(I)=C$&NW$&CH
$
780 L$(0)=STR$(VAL(L$(0))-LE
N(L$(I))):: GOTO 740
790 NEXT I :: CH$,C$,OW$,NW$
="" :: SUBEND

800 SUB SF(L$(),L):: DISPLAY
 AT(9,11)ERASE ALL BEEP:"Cas
sette" :: DISPLAY AT(10,11):
"_______" :: DISPLAY AT(11,
5):"Press:-"
810 DISPLAY AT(13,6):"1 For
CS1" :: DISPLAY AT(15,6):"2
For CS2" :: CALL CH(2,K):: I
F K=49 THEN C$="CS1" ELSE C$
="CS2"
820 DISPLAY AT(15,1)ERASE AL
L:"There are";L;"lines to sa
ve" :: OPEN #1:C$,INTERNAL,O
UTPUT,FIXED 192
830 FOR I=1 TO L :: DISPLAY
AT(12,4)ERASE ALL:"Saving li
ne number";I :: PRINT #1:L$(
I):: NEXT I :: CLOSE #1 :: S
UBEND

840 SUB RF(L$(),L):: DISPLAY
 ERASE ALL :: ON WARNING NEX
T :: INPUT "How many lines t
o read ":L :: GOTO 860
850 GOTO 840

860 OPEN #1:"CS1",INTERNAL,I
NPUT ,FIXED 192 :: FOR I=1 T
O L :: DISPLAY AT(12,4)ERASE
 ALL:"Reading line number";I
 :: INPUT #1:L$(I)
870 L$(0)=STR$(VAL(L$(0))-LE
N(L$(I))):: NEXT I :: CLOSE
#1 :: SUBEND

880 SUB PF(L$(),L):: DISPLAY
 ERASE ALL :: FOR I=1 TO L :
: PRINT STR$(I);"!-";L$(I)::
 IF I/4=INT(I/4)THEN CALL KC
890 NEXT I :: CALL KC :: SUB
END
```



END

****
* SORTING OUT SORTS. *
****

The Aim of this Article is to explore Sorting Routines written in T.I. Console Basic. It may well turn out that several Articles are required to complete the Task! The programmes which will be presented will be able to be RUN on other Basic Dialects with the assistance of only Minor Surgery. The Multi-statement line feature of Extended Basic can be used to compact the programmes for use in Extended Basic Programmes.

IN THE BEGINNING.

Most people who write programmes for their own use or for the pleasure of making the Computer work for them tend to be mostly concerned with Logical and Orderly programming, a third factor sometimes considered is available Memory. An important feature of Programming which usually isn't considered untill the programme is "UP" and "RUNNING" is execution time. A familiar comment is "the programme runs fine but it is SLOW". At the time of Programme Design, Execution Time should be considered. This is particularly so in relation to Sorting Routines, for the uninitiated the differences between Sort Routines are quite startling. Perhaps at the end of these Articles we will all have gained some more knowledge about Sort Routines.

DEFINING THE SORT.

As with all good Programming Techniques a start is made by defining what a Sort Routine is required to accomplish. Some examples are,

(1) Re-arrange a Data List into some pre-defined order.
(2) Re-arrange parts of a Data List into a Pre-defined order.
(3) Retain the original sequence of a Data List but still produce a List in a Pre-defined order.
(4) Use only one piece of Data from a Data Set to Re-arrange a Data List but still retain the relationship in the Data Sets.

When writing Programmes a clear understanding of what a Programme is required to do must be obtained. A Professional Software writter simply would not attempt to write a programme unless the Use and Purpose of the Programme had been clearly Defined.

If this proceedure is followed then a reasonable assumption is that the programme will need little modification to make it fit the User's requirements. (Apart from poor programming). Similarly with Sorts, if you know what the Sort is required to do, then selecting the most suitable for the job is more likely to occur.

CHANGING PLACES!

Most Sorting Routines need to be able to interchange two variables without loosing any values. A TEMPorary storage location is needed to do this:

EXAMPLE:

Programme	Operation	Variable A	Variable B	Status TEMP
1 A=5	SET A=5	5	0	0
2 B=10	SET B=10	5	10	0
3 TEMP=A	SET TEMP=5	5	10	5
4 A=B	SET A=10	10	10	5
5 B=TEMP	SET B=5	10	5	5

The Variable TEMP is used to 'HOLD' the value of 'A'. This then allows 'A' to be given the value of 'B'. 'B' is then free to accept the original value of 'A' from 'TEMP'. Thus the variable values are "swopped". The same routine will "swop" string variable values by adding '$' to the variable names.

A LATE PREAMBLE.

The first Sort Routine to be discussed will be a BUBBLE SORT. Progression from there will lead to increasing complex and quicker Sort Routines. Each programme presented will contain two variables, "SWOP" and "COMPS". They will be used to count the number of SWOPS and COMPARISONS made in each Sort Routine. So that realistic comparisons can be made between each Sort the same Data List should be used on the Sorts. The first programme will include a short routine and Data Lines to load 100 numbers into an array for Sorting. This Routine should be saved separately so that it can be reloaded and included in each Sort routine in the following Articles.

When programmes are presented in following articles this load Routine will be assumed to have be already loadedinto the Computer.

It is about time we got down to tin tacks and looked at that BUBBLE SORT or the Editor will not be impressed.

CONT

BUBBLE SORT.

The Bubble Sort is probably the easiest understood of the Sort Routines that will be discussed. the name of the Routine is derived from the fact that it "BUBBLES" the Data Items being sorted into their correct place.

The Bubble Sort tests adjacent pairs of Data and makes a swop if needed, it then tests the next Data Item against the adjacent Item of the pair previously tested and again swops if needed. In the example below the test,
IF DATA(A)<=DATA(A+1) THEN .....
is applied to the Data pairs.

```
DATA    DATA    DATA    DATA    DATA
   A=1     A=2     A=3     A=4
  SWOP?   SWOP?   SWOP?   SWOP?
32\    /32      32      32      32
   NO
98/    \98\    /16      16      16
            YES
16      16/    \98\    /24      24
                    YES
24      24      24/    \98\    /60
                            YES
60      60      60      60/    \98
```

This Demonstrates the BUBBLING of the number 98 (the largest) to one end of the Data List on the first pass of a Bubble Sort Routine. The next pass would BUBBLE the number 60 into it's position in the Data List. The third pass BUBBLES the number 32 into it's correct position in the Data List. This process continues for (N-1) passes (N=Data List Length). In the example above 4 passes. The number of COMPARISONS each PASS is (N-1). In the example above 4 COMPARISONS each pass.

Therefore the number of COMPARISONS to Sort a Data List 5 Items long is 4*4=16. For a Data List N Items long the number of COMPARISONS=(N-1)*(N-1). We will reduce that figure later in the notes.

If the Test used was changed to IF DATA(A)>=DATA(A+1) THEN .....
ie. reverse the '<' sign, the lowest number would have been BUBBLED through the List. The next pass would BURBLE the second lowest into place, continuing untill the Data List was Sorted.

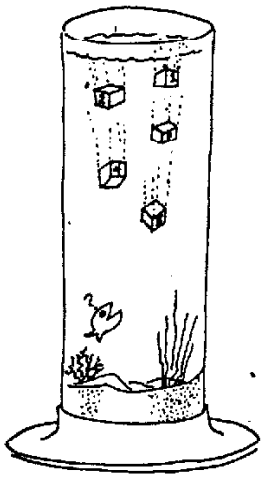
```
100 REM ********************
120 REM *   BUBBLE SORT    *
140 REM *  DEMONSTRATION   *
160 REM *  M.V.99'ers.     *
180 REM *     M.S.W.       *
200 REM ********************
210 CALL CLEAR
220 DIM S(100)
240 FOR A=1 TO 100
250 READ S(A)
260 PRINT S(A);
270 NEXT A
280 REM N=100
290 PRINT "RANDOM NUMBERS NO
N IN ARRAY": :"COMMENCING SO
RT"
300 FOR FIRST=1 TO 99
310 FOR INNER=1 TO 99
320 REM FOR INNER=1 TO N-FIR
ST
330 COMPS=COMPS+1
340 IF S(INNER))=S(INNER+1)T
HEN 390
350 SWOP=SWOP+1
360 TEMP=S(INNER)
370 S(INNER)=S(INNER+1)
380 S(INNER+1)=TEMP
390 NEXT INNER
400 NEXT FIRST
410 PRINT "SORT NOW COMPLETE
D"
420 FOR A=1 TO 100
430 PRINT S(A);
440 NEXT A
450 PRINT "C/SON=";COMPS;"SW
OPS=";SWOP
460 END
470 DATA 210,37,750,220,624,
38,230,51,371,810,1,33,625,2
22,340,626,313,52,372,820
480 DATA 2,44,240,333,320,39
,314,53,373,830,3,55,746,444
,330,780,8,54,374,840
490 DATA 4,66,627,555,340,47
9,7,55,375,850,40,77,735,66,
350,629,6,56,376,860
500 DATA 623,88,11,777,360,4
76,5,57,377,870,5,42,22,888,
361,477,4,58,378,880
510 DATA 700,99,628,999,362,
478,3,59,379,800,66,24,36,75
,63,54,40,89,82,99
```

THIS LOOP CONTROLS NUMBER OF PASSES

THIS LOOP STEPS THROUGH COMPARISONS

TESTS DATA

SWOP ROUTINE



THE PROGRAMME.

When entering the above Bubble Sort DO NOT omit the REM's on Lines 280 and 320. Perhaps at this point inexperienced Programmers might like to consider what those lines will do when included in the Programme? Also what will need to be removed to allow the Programme to RUN error free? CONT

PROGRAMME DESCRIPTION.

This will be kept to the bare essentials.
LINE 210 SETS DIM ARRAY S(100). LINE 220-280 LOAD/PRINT UNSORT LIST. LINE 290 FOR/NEXT LOOP N-1 PASSES
LINE 310 FOR/NEXT LOOP N-1 COMPS
LINE 340 DATA A and A+1 tested for swop if no swop steps to LINE 390.
LINE 350-380 DATA swop Routine.
LINE 390 END COMP LOOP.
LINE 400 END PASS LOOP.
LINE 410-440 PRINT SORTED DATA LIST.
LINE 450 PRINT "COMPS" and "SWOPS".
LINE 470- DATA LIST.

Run the Programme and note the Comparisons, Swops and, using a stop watch the RUN Time.
The results I obtained are below,
COMPARISONS = 9801 (99)
SWOPS = 2516
TIME = 6 min 14 secs.
Run time was taken from when the ENTER KEY was pressed after typing run untill DONE appeared on the screen. Loading the Data into the Array and printing it twice to the screen takes approx. 16 secs. Since this will be constant for all the Routines discussed it has not been subtracted from the resultant Time.
The result is self explanatory. The Bubble Sort is quite slow as you no doubt found when you gave the Programme a RUN. I don't think it would be impossible to beat that time using a card system by hand!.

A SPEED UP!

Referring back to the REM's in LINES 280 and 320. Remove these two REM's and add a REM to Line 310. Run the Programme again and record the result. figures I obtained are,
COMPARISONS = 4950 SWOPS = 2516
TIME = 4 min.
This is a much better result than for the original Bubble Sort. The modified version took 2 min 14 secs off the RUN time. Thats not bad for a beginning. To improve further a new Sort Routine will be needed.
What did the changes do?.
LINE 280 Put the Data List lenght 100 into the variable N.
LINE 320 is a new FOR/NEST LOOP for comparisons using N-1 as the LIMIT.

HOW IT WORKED.

As shown earlier in the notes the first pass of a Bubble Sort BUBBLES either the highest or lowest Data Item to one end of the Data List. Since the end Data Item is now in place there is no need for that Item to be tested again. The LIMIT N-1 steps the Comparison Loop Limit back one Item each Pass. The first pass tests to Item 99, the second pass to Item 98 etc. Thus saving a considerable number of wasted Comparisons from being done. In our case the RUN time was decreased by 35%.

REVERSING THE LIST.

The Sort Routine re-arranged the Data List into descending order. To have the Data List Sorted into ascending order change the '>' sign in LINE 340 to a '<'. That is from Greater than to Less than.

A SORTED LIST.

The Bubble Sort as shown above would take the same number of Comparisons to RUN through a previously Sorted List as it would the Same List unsorted. Add the following Lines to the programme. 305 F=0
385 F=1
395 IF F=0 THEN FIRST = 99
The variable F is a FLAG which is Set to ZERO at the start of each new Pass. If a SWOP occurs during that PASS then F is Set to 1 on Line 385 and the Sort contiues. However if no SWOPS occur during the Pass then F is still at ZERO and Line 395 will end the Sort Routine by setting FIRST to 99. So if a Data List which is already Sorted is encountered then the Bubble Sort will make only enough Comparisons for one Pass and then stop Sorting! Remember this feature!

CLOSE.

In summary the Bubble Sort would not be satisfactory on large Data Lists because of it's lack of speed. However for smaller Data Lists it is more than suitable. The time difference between it and Routines yet to be dicussed on small Data List will be quite small, after all 35% of not much really isn't much! The Sort is relatively simple and effective. A simple effective Programme beats hands down a complex Programme which does the same job only maginally quicker. It also occupies less Memory than some of the more "up market" Sorts yet to be discussed. Out of space for this month. Next month a look at Sorting String variables, Indexing and an interesting counting Sort. By A.Wright.

CONT

MACHINE CODE MASTERY

FUNNELWEB FARM

The ultimate way to get at real potential of the TI-99/4a is to write or run machine code programs. The next best thing is TI-Forth, but that's grist for another mill. Originally TI did not intend that users would ever write their own machine language programs and provided no books at all in console Basic to link to machine language routines, or to allow direct access to machine functions. The same sort of corporate marketing contempt for the customer led them to put calculator keys on the original TI-99/4. They weren't and aren't alone in that attitude of contempt for the user .... look at the IBM PCjr years later, or the Apple Macintosh.

And when they did bring out the expansion system, it still did not realize the potential of the TMS-9900 processor because of the fractured memory map and conversion of the 16 bit data path to successive 8 bit slices for all but a small part of CPU memory space, adding that insult to use of external memory with wait states. However when TI finally made machine code available to users they did it in style, adapting their mini-computer software for the purpose. Some of the programs supplied still contain traces of their origin, such as memory mapper instructions relevant only to the larger 990 minis.

First let's look at how machine code programs are recorded as disk or cassette files, and then survey the modules which allow these files to be loaded and executed. These files come in two forms. The most direct form is as memory image files, in which the actual contents of a block or blocks of memory are stored, with control information appended. These are known as PROGRAM files in TI-99/4a language (and correspond to .COM or .CMD files in other systems). TI Basic and XB programs are also stored in this format, which can be saved to and loaded from cassette as well as disk. The other kind, usable only with a disk system, is the assembly tagged OBJECT file. In normal usage of the TI-99/4a, object files are created to be relocatable in memory by the loader, and the programmer does not have to know explicitly where the loader has placed them, and calls their entry points up by name. None of the primitive USR or suchlike business. There are 4 (maybe 5) modules available which can load and run machine language programs. They all have different capabilities and limitations.

Editor/Assembler

The primary one is EDITOR/ASSEMBLER which is necessary for creation of relocatable object files (... well you could write one with a word processor but that would be masochism of a high order, exceeding even direct POKEing of machine code bytes). E/A will load any of the machine code file types mentioned above, from its menu screen. The LOAD AND RUN option handles both uncompresed and compressed tagged object files, and will resolve REFerences by name from one object file to another, or to standard system names. Uncompressed object files represent bytes in Hex notation, and take about twice as much disk space as compressed object files. Invocation of LOAD AND RUN re-initializes the memory pointers completely while loading the system utility routines such as VMBW from GROM storage, so if a sequence of file loads is interrupted by an error, it must be started all over again.

E/A adds CALL LOAD and CALL LINK to console Basic to allow these same object files to be loaded and accessed from Basic. The standard utilities such as NUMREF for communicating with the Basic program must be loaded as a separate file BSCSUP.

E/A will also load and run from the RUN PROGRAM FILE option, program files of machine code, prepared according to a specific recipe as SAVE files. The details of these will be a subject for HV99 News articles in the future. It will load

them from cassette as well but I can't see anyone doing that in preference to using disks, unless perhaps they have installed the TIUP internal memory expansion mod in the spare console that gets taken away on holidays. No provision exists in Basic to load SAVEd program files from Basic as they could overwrite part of the Basic program in the VDP on the way in.

Extended Basic

The next module which can load assembly code is our old friend EXTENDED BASIC. This is much more limited than E/A in what it will handle. Firstly it recognises only the 8K low memory area from >2000 to >4000 for loading relocatable object code. Absolute code can be loaded, or RAM buffer areas used in the lower part of high memory once it is known how far down this is filled by the XBasic program. The loader does not handle external REFerences, and the utilities loaded by CALL INIT in XB are missing the most useful one - DSRLNK, and GPLLNK as well. The Basic support utilities are loaded by CALL INIT from GROM. The assembly source code has to locate them with EQUates. A minor difference from E/A is that CALL LINK always hands over control in the GPL workspace. Programs written to LINK to E/A Basic will almost always need at least minor modifications to LINK to XB successfully.

The operational hangup with XB is that the loader is written in GPL and is painfully slow. A long assembly routine, such as Text to Speech, may take several minutes to load (shades of the Commodore 64's disk system). The usual way round this is to load an assembly language loader which in turn does a faster load of the longer program. The great virtue of the XB module that sets it apart from the others is that it supports auto-RUNning from disk, as soon as the module is selected, of an XB program DSK1.LOAD which can then load further programs. The other reason for preferring XB is that it is a vastly more powerful language than the mildly enhanced console Basic offered by E/A. Unlike E/A it can never load machine code programs without Basic as an intermediary.

Mini-Memory

This module has its own particular charm as the only one which allows access to machine code without the 32K memory expansion and using cassette storage. In this mode a LINE by LINE (or immediate input) Assembler allows standard TMS-9900 mnemonic assembly code to be entered in a restricted format. This is a descendant of TI's board level 990 evaluation systems. Only 700 odd bytes are available in the module's 4K of CMOS RAM after loading the assembler but I can't imagine anyone wanting to do programs longer than that with L by L. Still it's enough to do a pretty fair Game of Life program. MM also contains a full set of system utilities and Basic support routines in ROM and EASYBUG in GROM, a monitor program that is useful but much less powerful than the E/A DEBUG.

MM is even more useful in a fully expanded system. It does not provide the Editor and Assembler features of E/A, but offers more scope for loading and running programs. Firstly there is 5-6K more RAM available, 4K of CMOS RAM in the cartridge and the saving of space in RAM because of the utilities in cartridge ROM. Its principal deficiency as compared to E/A is the lack of a PROGRAM file loader, but this can be easily remedied by writing your own to reside in MM cartridge RAM. Even the L by L Assembler, as well as EASYBUG, remains useful for occasional little purposes anywhere in RAM, and I have prepared a disk based version for convenience.

E/A object code, even compressed, is loaded successfully as long as REFs are used for system utilities and Basic support routines. EQUates as used for XB code will only get it right for one module. The loader has one more space, cartridge RAM, to place relocateable object code as a last resort. I have not yet experimented to see whether the loader will link object files with external references as the E/A loader does. The MM manual, never a fount of information, is silent on this point.

MM does not erase its DEF table

unless it is explicitly done by one of several means. The table survives a return to the title screen, and even switch-off if the internal battery is still alive. This is different from E/A's workings, and must be taken into account for better or for worse. Code in memory expansion does not survive switch-off even if its name lives on.

TI-WRITER

Now we leave the modules which can load files under any name for one which loads program files with particular names. TI-WRITER tries to load an E/A type SAVE file from DSK1 under the name EDITA1 (and successor filenames) when the EDITOR option is selected from the menu screen. If you have followed the E/A manual's advice on using the SAVE utility with TOMBSTONE CITY as the victim, take the file so generated and place it on a fresh disk under the name EDITA1 and place in DSK1. Then fire up TI-WRITER, choose the Editor option, and see what happens. Extension to Formatter and Utility options are obvious. It may provide light relief to heavy TI-WRITER sessions. More seriously, short of writing a PROGRAM file loader to be loaded by XB using DSK1.LOAD, it is the nearest that the TI-99/4a comes to an auto-loader for machine code program files.

****
* FUNNELWEB FARM FUNNELWEB FARM *
****
Articles appearing under the psuedonym of FUNNELWEB FARM are produced by the inimitable TONY MC.GOVERN of "EXTENDED TUTORIAL" fame. Tony is a committee member of HV99'ers and prolific contributer to the magazine. At present he is putting the finishing touches to a TI-WRITER load programme which he hopes to present in the magazine in the near future. He also intends continuing on with his series of EXTENDED TUTORIALS.
As space permits we intend publishing the earlier articles in the EXTENDED TUTORIAL series which originally appeared in TISHUG magazine.
ED.

****
* FLOPPY DISC DRIVE MAINTENANCE *
****

The Numbers of Disc Drives now in use within the Group is steadily increasing. The following has been kindly forwarded to the Group by the Computer Systems Maintenance Supervisor at the Newcastle Plant of B.H.P.

Floppy Disc Drives are low maintenance devices, however the following guide should help reduce the failure of Disc Drives.

1) The Drive door should be kept closed when the drive is not in use to keep out dust. Some Drives require a disc to be inserted before the door may be closed. Use disc blanks for this purpose.
2) The worm drive for the heads should be checked periodically (3-6) Months, clean off any dirt build up on the thread. Turn the thread by hand to check for any sticking. Lubricate with a light smear of graphite grease.
NOTE! Some drives use nylon thread and don't require lubriction.
DO NOT LUBRICATE THIS TYPE OF DRIVE.
3) The heads should be inspected for wear and contamination after every 8 hours of usage. Examine the heads through the door using a torch and a Dentist mirror. Any smears or dirt of the head surface should be removed using cottons buds dipped in absolute alcohol. Dust in the drive should be blown out.
4) Check cooling fans for free rotation, REPLACE faulty fans. Clean the fan as required. Note drives will overheat very quickly if the fan is not running.
5) Floppy disc drives are quite suseptable to noise on the power system. To improve the noise immunity, connect the drive to the 250 volt mains via a line filter.
6) Do not store floppy discs in close proximity to stray magnetic fields ie. speakers, or power cords.
Always return a floppy disc to it's dust cover when not in use.
7) Do not write on to disc labels with anything other than a soft felt tipped pen. Preferably write onto labels before applying them to the disc.

JUNIOR SOFTWARE REVIEW

by DARREN MacCLURE.

This Month Darren has reviewed a programme written by Tony McGovern and which is available through the Club LIbrary.

T.I. Motion.

The game can be difficult and could cause loss of sleep to fanatical fans. Otherwise it is quite safe, with no lasers, missiles to avoid or nuclear warheads to drop.

It is a simple game where the player has to guide his trail around randomly placed obstacles without runing over his previously marked trail.

The programmer has removed all the bugs and made the programme into a well written one which can be relied upon.

The game can require a great deal of skill and can be quite enthralling, entertaining the whole family for hours on end.

The programme is available on cassette at the H.V. 99'ers CLUB MEETINGS. T.I. motion requires only T.I. BASIC to run and is a worth while addition to your software collection.

CLUB LOGO COMPETITION

This issue we are starting a new competition in which you have the opportunity to design a LOGO for the club. The winning entry will be judged at the September general meeting and the winner will receive a valuable educational software prize. The competition is open to all HV99er members.

We are looking for something unique to our club and which is representative of TI99/4A users in the HUNTER VALLEY. We aren't looking for the neatest or prettiest entry so dont worry if yours is a little rough around the edges. It is more the the image created ; the dressing up and fine detailing can be done later.

OK! lets get to work and start those entries rolling in. You may submit as many entries as you wish. Please address entries to.

HV99ERS LOGO COMPETITION.
C/O 25 RESERVE RD.
WANGI

FORTH REPORT

Our very own "FORTH INTEREST GROUP" is now up and running with the first meeting being held at Richard Terry's house on Thursday the 20 th. June. The group has set itself a series of assignments to complete before their next meeting. One of the prime concerns of the group is to try and standardise routine names amongst themselves and hopefully not duplicate any previously defined routines. To this end all available Forth publications are consulted prior to defining a new term so as to confirm that the hard work hasn't already been done for them.

This months project is to define three routines.

1. INTEGER ONLY ACCEPT WORD, that will accept up to a 5 digit number at any point on screen, validate that it is a digit, if not reaccept.
2. FLOATING POINT TO DO THE SAME.
3. Write a small program which will accept the above digits and reprint them at another spot on the screen.

A number of people who replied to our questionnaire said that they have the FORTH language; well here is your opportunity to join a new group who are all starting off at square one.

Richard Terry has suggested that he may be able to start writing a regular column devoted to FORTH. Thanks a lot Richard, we are all looking forward to reading your column in the next issue. ED.

NOTICE TO COMMITTEE MEMBERS

Committee members are reminded that at least one full system is required at each general meeting. Please check with Joe Wright as to when your turn is due. Anyone else who would like to take their system along to give a demonstration is also welcome.

UH-OH UH-OH UH-OH UH-OH UH-OH

HV. 99er, Bugs.

In the article on ACCEPT the information on FCTN ERASE was wrong. FCTN ERASE works exactly the same as FCTN INS and FCTN DELETE. The cursor has to be passed over the text to read what is on the screen, that is to be erased.

****
* 1985 A NEW ERA FOR CorComp *
****

Reading through the latest issue of CorComp Cursor, the Newsletter of CorComp Inc. The companies financial situation now appears to have been resolved and 1985 should see the introduction of new products to the TI99/4A market place.

According to the newsletter the company is welcoming 1985 with determination and enthusiasm, they have set their goals for the new year with a positive attitude toward financial recovery with a tight rein on all expenditures, production schedules and research and development.

1985 should see CorComp introduce new products for the 99/4A, having learned a lesson common to the computer industry, they will no longer announce any product before having a working prototype ready for Beta testing.

The CorComp 9900 EXPANSION SYSTEM is now available in Australia with power supplies made to Australian specifications. With the additional memory and the new CorComp double sided, double density Disk Controller capability the user can control up to 4 Disk Drives.

The Cor Comp 2.3 Disk Manager is supplied on 5 1/4" floppy disk. This program is considerably faster than TI's. One of the many enhancements of this program is that it allows the user to CONFIGURE the manager to his system. You can select your own text and screen colours, set up each of the different drive types attached to the controller; for number of sides, density and number of tracks. The printer type can be configured for catalog and disk test print outs. the configuration is saved on the disk and used as the defaults when the disk manager is loaded.

The CC-Disk Controller also adds new commands and programming statements to Basic and Extended Basic such as CALL POKE CALL PEEK for rapid reading and writing to CPU Memory etc .

Several of our members already own the 9900 Micro Expansion and may have already put some of the additional features to use. We would like to hear from these members, Perhaps a hardware review and demonstration may be forthcoming.

Perhaps one of the owners of these systems may be prepared to submit a review for publication in our next issue of HV99'ER NEWS.

Feedback

Firstly, let me take the opportunity to thank all those people who took the time to complete and return our recent questionnaire. Hopefully the information obtained will assist the committee to organise and run the club to be as beneficial to as many members as possible.

We are all too aware of the fact that it is impossible to please all of the people all of the time, so if you feel that something just isn't right or you would like to assist in some way please don't hesitate to get in contact with a committee member.

Without doubt the two most important items in your opinion are the news magazine and tutorials. The committee has already acted in this regard with regular classes in all subjects now running. Several members have expressed interest in teaching new classes and as these are organised you will be informed.

As you can see the magazine is alive and well; and something quite unusual in the fact that at present it is made up of original HV99er content. To maintain this high percentage of original content we wil require your assistance in the form of contributions; after all, its not fair to expect the same people to keep filling the magazine issue after issue. So if you feel that you could contribte an article no matter how small or on what topic we would be more than grateful to hear from you.

Some of the younger group members have asked us to establish a SOFTWARE HALL OF FAME. Would anyone wishing to participate please send in their high scores (Authenticated by Parent) and we will publish them each issue. Perhaps we may be able to organise a challenge game at the monthly meetings with a suitable prize for the winner , please let us know your thoughts on this.

In the next issue we hope to start running various new columns like HINTS TIPS, LETTERS TO THE EDITOR, FOR SALE etc. etc. If you would like anything published in these columns please write to.

EDITOR HV99'ERS
15 GAYTON CLOSE.
WARNERS BAY.2282

HUNTER VALLEY 99'ERS LIBRARY NEWS

HI 99'ers,
Once again I put digits (2) to my trusty '4A's a Jolly Good Computer' keyboard. How I wish my fingers and brain were as swift as the tongue and eyes are. Again DONT FORGET the old saying CLEAN HEADS ARE A SOUND IDEA it seems a lot of people dont realise how much more IMPORTANT this is when it comes to cassette recorders for computer use,especially on the -99/4A When you think of the varied number that we can use instead of dedicated computer use only,like some computers around. What this means is CLEAN the HEADS once WEEKLY with heavy use.
Also a very good investment is a GOOD DEMAGNETISER. USED reguarly you will be surprised at the difference to your SANITY and your family wellbeing as that dreaded message-ERROR NO DATA FOUND- or its nasty amigo - ERROR IN DATA DETECTED - dissapears from the screen and the clear crisp sounds fills the air once again as your favorite program LOADS and RUNS sucessfully.
-TI-99/4A-STEPHEN SHAW-TI-99/4A-
GETTING STARTED with the TEXAS TI-99/4A. This is an excellant and worthwile addition to your library bookshelves. It will not gather dust, rather I guarantee it will become a well refered to book. It is not one of your usual CONVERTED for TI-99/4A users with lots of cartoons & simple messages of 2 & 3 syll'ables like a lot of publications from usa. This is by well known English author(S.SHAW)& programmer who writes regularly for a number of computer magazines in the UK He really understands the complex architecture of the 99/4A.He does not say you will be a programmer in a few hours as some do,but that with time & practice you will learn how to better use the power of the computer. Easily readable style with, the feeling the author is talking and guiding you all the way. This book is for everyone no matter what part of the learn'n curve you are on. Our own intrepid XB TUTOR scribe remarked"its the only book on market worth reading" I will have my copy at"meet"so you can see it. The CO-OP BOOKSHOP have a limited number of copies available.@members discount just mention the HV 99'ers or tellem Al sent ya.
Well having returned from searching the 4 corners of the world for more good programes for the library,I had better tell you that I made contact with TI*MES magazine. An excellently produced and full of quality article magazine,which you can borrow copies off, from the library thanks to the generosity of Editors Clive (expat.) and Audrey Scally.It was great to be meeting such enthustastic dedicated TI'ers and exchange ideas,views and see how the TI*MES is produced.Thank you both.

---ATTENTION !! ATTENTION---

Young members who would like to take part and write a GAMES REVIEW for the HV99'ers mag please give me name and phone No.at next meeting / class you see me at.Or phone me 436509 for details.

---ATTENTION !! ATTENTION---

Thats it for now,

Happy Programing.

Al Lawrence.