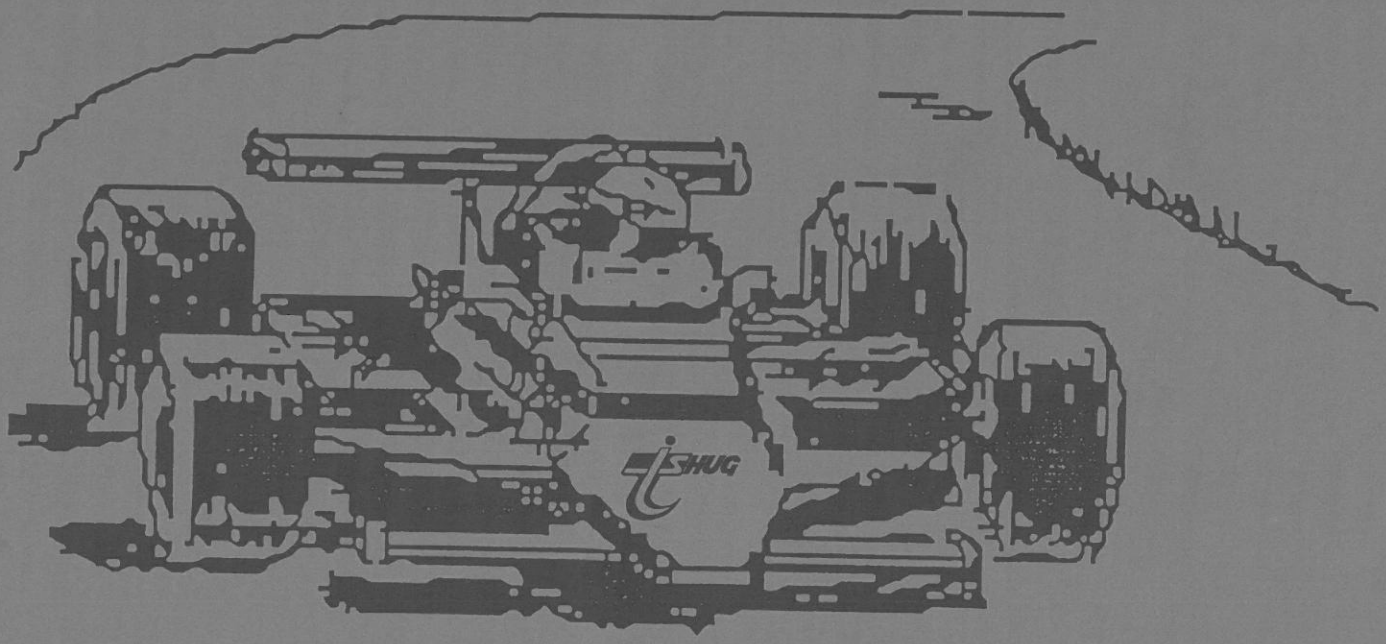
TIsHUG NEWS DIGEST

Focusing on the TI99/4A Home Computer

Volume 10, Number 5 June, 1991

Registered by Australia Post - Publication No. NBH5933

The TI99/4A and TIsHUG



Still way out in Front!

Sydney, New South Wales, Australia $3

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

TIsHUG News Digest

June 1991

All correspondence to:

P.O. Box 214
Redfern, NSW 2016
Australia

The Board

Co-ordinator
Dick Warburton (02) 918 8132
Secretary
Terry Phillips (02) 797 6313
Treasurer
Geoff Trott (042) 29 6629
Directors
Rolf Schreiber (042) 84 2980
Russell Welham (043) 92 4000

Sub-committees

News Digest Editor
Bob Relyea (046) 57 1253
BBS Sysop
Ross Mudie (02) 456 2122
BBS telephone number (02) 456 4606
Merchandising
Percy Harrison (02) 808 3181
Publications Library
Warren Welham (043) 92 4000
Software library
Rolf Schreiber (042) 84 2980
Technical co-ordinator
Lou Amadio (042) 28 4906

Regional Group Contacts

Central Coast
Russell Welham (043) 92 4000
Coffs Harbour
Kevin Cox (066) 53 2649
Glebe
Mike Slattery (02) 692 0559
Illawarra
Lou Amadio (042) 28 4906
Liverpool
Larry Saunders (02) 644 7377
Northern Suburbs
Dennis Norman (02) 452 3920
Sutherland
Peter Young (02) 528 8775

Membership and Subscriptions

Annual Family Dues	$30.00
Associate membership	$10.00
Overseas Airmail Dues	A$60.00
Overseas Surface Mail Dues	A$45.00

TIsHUG Sydney Meeting

The next meeting will start at 2.00 pm on 1st of June at Ryde Infant School, Tucker Street, Ryde. At 12 pm, before the meeting, there will be a beginners' Editor Assembler class for all those interested.

Printed by
The University of Wollongong
Printery

TIsHUG News Digest ISSN 0819-1984

Index

TIsHUG Fairware Author of the Month

The Fairware Author once again this month is Clint Pulley for his c99 compiler and other utilities. c99 is a subset of the C language, which is used extensively in industry. It allows the power of assembler to be used in a high level language. The shop will have version REL4.0 of c99 for sale. All donations collected at the meeting and sent in will be mailed to him.

Editor's Comment

by Bob Relyea

I was encouraged by the numbers that attended the various tutorials at the May meeting. I was able to run two sessions, one before and another after lunch. There was considerable interest in Multiplan and Word-Processing and we were able to get a few things sorted out. I went away with a problem to work on for a member who wanted some suggestions for setting up the financial aspect of his business on Multiplan. One member who attended was able to change the hex code on one of the sectors of the Multiplan disc so that all of my printer parameters could be fit on without me having to type it all in everytime. Everybody got something out of the sessions so I went away feeling 'fulfilled'. Larry Saunders made up some great pictures advertising each tutorial which were posted at each door - thanks Larry. A big thank you for those involved in some way in organising and preparing the sumptuous lunch. I really enjoy those tutorial days! I never get tired of seeing Ross Mudie's train set-up and am amazed at the portability of it. See you at the next meeting. o

Co-ordinator's Report

by Dick Warburton

THE PARABLE OF THE TI

Once upon a time there were some little computers who lived with their parents in America. One of them became an orphan, because his parents did not want him any more. Lots of people tried to look after him, but because they did not get much help,they had to work things out for themselves, and he had a hard time growing up. His computer friends however, had such a good time, because they did not have to do much for themselves, and people did not ever expect them to grow up. Their parent companies had lots of other new computers whom they trained to take the places of the older computers. The little computers were very smart, but the old ones grew lazier and lazier, because their parent companies did lots of things to keep them happy. They just kept on doing the same old things, and because they had slick advertising to support them,they just kept on going as they were. People who looked after them, paid lots of money for the privilege, and bought them all sorts of expensive toys to play with. Some of these toys were very old, and were hard to operate.

People kept on looking after them, and these machines paid for their keep by performing for their masters. Some of them could only really play games and amuse people, but one of them was very good at school, and came to have lots of friends. Unfortunately his parents died too, and he became an orphan. Then almost nobody wanted him, because he was only good at schoolwork. The other one went to work in an office, and he learned lots of things to do. Unfortunately, he was not very versatile, and even though people spent lots of money writing simple instructions for him, he did not always do what he was told, and some people wished that he would grow up. He was also hard to look at, and did not have nice colours. He was very slow, and some people thought that he was stupid. Most of his owners, got fed up with him and locked him away in a cupboard, and bought a shining new model, with lots of new functions and shiny new colour monitors and things. These new machines were called DOS machines. DOS stands for Dammed Orrible System. DOS machines are very good at making their owners feel really helpless and frustrated, and are often used as paperweights or typewriters.

The little orphan had a hard time, and had to work hard to survive. He was able to do lots of things. He was not very big, but he was very intelligent and learned quickly. He was very efficient for his age and size, and because he was only small, he had to learn lots of tricks to get the jobs done properly. People who looked after him, liked him, and helped him to grow up into a very sophisticated computer. He could do almost everything that the other three computers could do, and was was much more obedient. His instructions were much easier to give, and he had users who liked him. They did lots of things to make his job easier. Some of his users gave him a beautiful colour monitor, while others liked to see how fast he could work by providing a ramdisk. He kept on growing stronger and stronger, and better. The other computers almost stopped growing. Because he is made so well he can be fixed up if he gets sick, and hospital bills are small. His friends like to help each other, so they are able to do things to expand him much more cheaply than other computers. This makes them happy in a time when costs are high, and there is not much money. This little orphan expects to live for a long, long time. He has friends all over the world who care for him, and who help to keep him growing. They keep making things for him which give him bigger muscles, and let him run much faster. His owners are very proud of what he can do, when all the costs and comparisons are made.

A simple story, but with much truth. The TI can do so many thing well. With ramdisks and eproms to help it along, it is efficient, user friendly, and will do almost everything I could want of my computer. It is cheap to run, cheap to expand, and has a great bunch of users world wide who help each other.It is a fun machine for real computer users. Through club membership, owners can become proficient in a number of ways. They can get help with programming, using a wide range of software, buying up to date software, and getting help for hardware problems. Costs for club members are kept to a minimum. Members can learn to about computer hardware, add accessories cheaply, expand their machines at minimum cost. All for a miserly $30 per year, let alone receive an excellent magazine and have the use of our own Bulletin Board. If your membership falls due, renew now, so our orphan can continue to grow, and we can continue to provide the range of services to club members.

See you at the next meeting,

Dick Warburton

o

C'est la vie, Ma Cherie Geneve!

by Joe Nualini, CO USA

First, I wish every success to Myarc with their new computer, the TI99/4A follow-on, Geneve. With the home computer market in a shambles it is a bold move on their part. However, I have given some thought about how this new product will impact upon the current TI community. Will it strengthen it or will it weaken it?

Since Texas Instruments exited the home computer market several years ago we have all looked for a replacement or follow-on to the 99/4A. Whenever news of one hit the presses there was an air of excitement and a lot of speculation in all the TI oriented publications. Time after time our hopes were crushed when the news turned out to be nothing more than a "pipe dream". Well now , after all our waiting, it appears that the long sought after follow-on is here! I am afraid, however, that its impact on the community is going to do more harm than good. When we first started looking for a replacement the 99/4A was not much more than a toy. We were fooling around with game cartridges and very basic programs. Data base management was done with Personal Record Keeping. The TE II was the only telecommunications package available and although TI-Writer and Multiplan were available, they were pretty expensive, and therefore not available to everyone. The truly good software was cost prohibitive to many.

Today that is not the case. Exceptional software developers like Paul Charlton, Tom Frerichs, Mike Holmes, Bill Warren, Ralph Fowler, Marty Kroll, Monty Schmidt, Will and Tony McGovern, Barry Traver, Bruce Caron and the like have given us software that would rival that of the finest PC at a price far below their worth. PRBASE has given us the ability to manipulate data as well as any system will. Fast Term and 4A/Talk are exceptional Telecommunications packages. DM-1000 and Marty Kroll's Disk Cataloger allow us to manage our program libraries. Monty's Techie BBS and Ralph Fowler's TIBBS let us keep in touch with each other both locally, within our users groups, and nationally, between groups. TI-Writer, now in the form of FUNLWRITER is as good a word processing package as there is available for any computer.

Multiplan is now within the price range of all. I know because I picked up four when LaBelle's was closing them out for $3.00. This program will do miracles when it come to manipulating figures and is limited only by your imagination. What it lacks in speed it makes up for in flexibility. There are programs too numerous to mention that will do virtually anything you want, and the sad part is that currently the best of these will not work on Geneve and will have to be rewritten, hopefully, before they can be used.

And how about hardware... CorComp and Myarc have given us double density capability. These same

continued on page 14

Secretary's Notebook

by Terry Phillips

I hope all who turned up for the May meeting had a good time and got some information that will be of assistance to them from the wide range of demonstrations and tutorials. Judging from the amount of money collected at the barbeque I would say that attendance was somewhere around 45 which was a little disappointing considering the fine day and the previously mentioned range of activities that were available. Only 2 members turned up for my TI-Base session so, on the surface at least, it appears there is not a great deal of interest being shown in data base usage. Then again, maybe everyone is proficient in data base usage and they do not need any guidance.

At the time of writing, there are 47 members that so far have not renewed. Looking through the list of names I would suggest that the majority of them probably will renew and as all have now been posted a reminder they should start to roll in soon.

This month, the first for some considerable time, we have not received any new membership applications, so obviously there are no new members to welcome. Maybe next month we will have some.

I have just received the March issue of the Hunter Valley Newsdigest and in it President, Peter Smith, in his report advises that his group will no longer bury their heads in the sand and pretend that most of the members do not have other computers, rather, that support as desired, should be given to all members. To do this they intend publishing articles of wide interest or even on specific computer oriented bias. While their basic focus will remain with the TI99/4A, no longer will it be devoted exclusively to that machine.

Now that is an interesting development!

Also from the Hunter Valley March issue comes news of the following new products. If you are interested in anything that is mentioned here, have a word to the Directors.

Aircraft Graphics Disk - more than 30 famous aircraft in TI-Artist format. Available from Randy Packham, RR2 Naticoke, Ontario Canada. $US10 plus postage.

Hard and Floppy Disk Controller - basically an IBM controller modified and interfaced to be compatible with the TI99/4A. Available from Electronic Systems Development Corp, PO Box 23805, Washington DC 20026-3805 USA. $US225 is the suggested price.

Screen Preview - will format your TI99/4A document and display it to the screen in a Greeked format (the text being displayed as dots on the screen). You can justify the centering and margins before your print. Available from Asgard Software at $US12.95.

Rave 99 - have released 2 new expansion boxes both with a 200 watt power supply. One is designed to have the Geneve 9640 installed and the other will accommodate both the 9640 and TI99/4A board. The TI99/4A can be removed from the console and installed in the expansion box. Both the TI99/4A and 9640 can be running at the same time, although you will need 2 monitors and they cannot access the same peripheral simultaneously. A 32 bit bus is also included and is intended for development later. Prices are between $US300 and $US370 plus shipping costs.

That is about it for this month. Hope you can make it to the June meeting. o

continued from page 17

```
CODE   MOVB @SPCHRD,R3 read byte from speech synthesizer
       NOP             delay
       NOP
       NOP
       RT
       MOVB @H50,@SPCHWT instruction to pronounce data
       MOV  R11,R10    save return address
       LI   R5,>8000   talk status bit
SPCHLP LIMI 2          quit pressed?
       LIMI 0
       BL   @READIT    read status
       COC  R5,R3      still talking?
       JEQ  SPCHLP
       B    *R10

CLEN   EQU  $-CODE     length of code

WAIT   LI   R4,4       outermost wait loop
WAITIN CLR  R6         maximum delay
WAITLP LIMI 2          quit pressed?
       LIMI 0
       DEC  R6         inner loop finished?
       JNE  WAITLP
       DEC  R4         outer loop finished?
       JNE  WAITIN
       RT

SCROLL LI   R0,32      screen position of second line
       LI   R1,BUFFER  scroll buffer address
       LI   R2,23*32   23 screen lines to be read
       BLWP @VMBR      read the lines
       CLR R0          screen position of first line
       LI   R2,24*32   write 24 screen lines
       BLWP @VMBW      write lines
       RT

H10    BYTE >10        read byte instruction
H50    BYTE >50        speech ROM data for instruction
HAA    BYTE >AA        ROM identification code
       EVEN

BUFFER BSS  23*32
       TEXT '                '
       TEXT '                '
STACK  BSS  >20        return addresses
LOW    BSS  >20        less than pointers
HIGH   BSS  >20        greater than pointers
ADDR   BSS  >20        speech data addresses
NAMES  EQU  $          words/phrases labal

DEFPRT LI   R0,10*32+2 program name screen position
       LI   R1,DEFNAM  program name
       LI   R2,6       length of name
       BLWP @VMBW      put program name on screen
       RT

DEFNAM TEXT 'SPEECH'   program name

       END  DEFPRT     program name for automatic start
```

o

continued from page 18

Lines 11 and 12 summarise the printer commands.

Line 13 contains the printer commands which determine how the labels will be printed out. The dots on and after this line outline the label print area. Confine, and, if necessary, centre your text within the dotted area.

Additional labels may be added to the print file by copying the 6 lines which outline a label to after the last label.

Line 31 resets the printer and is normally included as the last line to be printed.

When you have finished setting up the label, erase the dotted outline without moving the position of the text.

Notes

It is best to work in non-word-wrap mode (hollow box cursor) to avoid the possibility of accidently reformatting the text.

Save the newly formed Label Print File under a new name to preserve the original file for future use. o

TIsHUG Shop with Percy Harrison

Further to the RAM CARD details outlined in my article last month please note that in addition to requiring a 74LS08 IC for every four Ram chips or part thereof you must have at least three 32K Static Ram Chips (62256) on the board and an 8K Static Ram Chip (6264LP) in socket U11, these are not included with the kit but are available from the club shop.

This month we are also able to give you a couple of small price reductions for both 5.25 DSDD and High Density disks, again due to our better buying of these products (see below for new prices).

PRICE LIST

5.25 in. DSDD Disks (Boxes of ten)	$6.00
5.25 in. HD Disks (Boxes of ten)	$12.00
3.5 in. DSDD Disks (Boxes of ten)	$12.00
3.5 in. SSDD Disk Drives	$20.00
5.25 in. DSDD Half Height Drive (New)	$65.00
5.25 in. DSDD Half Height Drive (Used)	$50.00
12 Volt AC Transformer	$3.50
13 Volt Arlec Transformer	$12.00
8.5, 17 Volt Transformer	$25.00
60 VA Transformer	$20.00
MFC Printed Circuit Board	$30.00
MFC Kit (Disk Controller)	$102.50
32K Kit for MFC	$26.50
PIO/RS232 (single port) Kit for MFC	$42.50
Combined 32K and PIO/RS232 Kit	$60.00
Music Kit with PCB	$65.00
32K Memory PC Board	$7.00
Eprom Ram PC Board	$45.00
Eprom Ramdisk Basic Kit	$35.00
Funnelweb Eprom Set (3 Eproms)	$36.00
TI Artist Eprom Set (2 Eproms)	$24.00
ROS Version 8.14	$12.00

N.B. ROS 8.14 must be purchased with first Eprom Set.

32K Static Ram IC (62256)	$12.00
8K Static Ram IC (6264LP)	$5.00
74LS08 IC (quad Schottky)	$0.50
1K Resistor	$0.05

COMMERCIAL SOFTWARE

Artoons SSSD	$12.00
Character Set & Graphic Design Cataloguer	$6.00
Character Set & Graphic Design I	$12.00
Character Set & Graphic Design II	$10.00
Character Set & Graphic Design III	$14.00
Display Master	$15.00
Genial Traveler (SSSD)	$6.00
Microdex 1 (SSSD)	$16.00
Microdex II (SSSD)	$11.00
Nuts and Bolts #1 (DSSD)	$6.00
Nuts and Bolts #1 (SSSD)	$7.00
Page Pro 99 version 1.6	$28.00
Page Pro Utilities	$17.00
Page Pro Applications #1	$2.00
Page Pro Templates (Vols 1 to 12)	$8.00/vol
Picasso Publisher Version 2.0	$14.00
Picasso Publisher Support Disks	$6.00
Picasso Applications Disk	$2.00
Rockrunner (SSSD)	$15.00
Spell It! (DSDD version)	$24.00
Spell It! (SSSD version)	$27.00
The Missing Link (TML)	$28.00
The Missing Link Companion Disk	$2.00
TI Artist Plus	$25.00
TI Base Vers 3.01 (SSSD)	$25.00
TI Sort SSSD	$15.00

Packaging and postage	Surface	Airmail
1 to 9 Disks	$3.00	3.70
10 to 20 Disks	$4.00	$4.90
TI Artist Plus	$3.00	$3.70
Display Master	$3.00	$3.70
TI Base	$3.00	$3.70
TI Sort	$3.00	$3.70

o

Jenny's Younger Set

Dear Jenny,

I have a simple guessing game for you this month. I hope it will provide some enjoyment over the next month.

VINCENT MAKER

```
100 CALL CLEAR
110 PRINT "HELLO. DO YOU WANT TO PLAY A GAME?":A$
120 IF A$="YES" THEN 140
130 PRINT "WELL I CAN'T HELP YOU THEN, BYE" :: END
140 PRINT "THIS IS A GUESSING GAME OUT OF 10. JUST
    ENTER YOUR GUESS..."
150 RANDOMIZE
160 A=INT(RND)
170 INPUT "ENTER YOUR GUESS.":AB
180 IF AB=A THEN 210
190 IF AB>A THEN 230
200 IF AB<A THEN 250
210 PRINT "CONGRADULATIONS! YOU GUESSED CORRECTLY."
220 END
230 PRINT "TRY GUESSING SMALLER"
240 GOTO 170
250 PRINT "TRY GUESSING LARGER"
260 GOTO 170
```

o

For Sale

30 cm Blaupunkt RGB colour monitor in very good condition. Suitable for 80 column use. Gives excellent resolution and colours with standard TI99/4A. Requires interface (enquire at TIsHUG Shop for details). Also works well with Amiga computers (only requires a cable). Only $200. Phone (042)84 2980.

o

continued from page 16

Head back to Ancient Storage (do not worry about the south end of the tunnel; you will find out about later), and then through to Dead Storage and the basement beyond. Once in the basement again, turn off your flashlight to conserve power; you will be needing light again later, and there are no spare batteries anywhere.

Now go along west, past the Aero basement, to the stairs. Chug upstairs to the Aero Lobby, then south to the start of the Infinite Corridor. Well, look at that: someone left a container here. A check of the label tells you that it is magic wax. Interesting. Bring it with you, you will be needing it soon.

Here is where the fun begins. Up ahead of you is a waxer being run by a maintenance man. He looks like something out of a grade-Z thriller, but make no mistake about it: he can be deadly, given the opportunity. He is also an obstacle, as he will not let you get past him.

What you need to do is wait until he moves off east. Then you move east. Keep this up until you reach the glass fire cabinet. Break the glass with your crowbar, take the fire axe, and go back to where the waxer is plugged into the wall.

CHOP! One good stroke with the axe, and the wire is severed. The waxer grinds to a halt, and the maintenance man is coming after you with murder in his eye. He means it, too. Quick! Open the container and pour the wax on the floor.

Now just stand there, and watch (try not to smirk) as the maintenance man slips and slides around, unable to keep his balance. If you wait long enough, you will see something REALLY creepy happen! After that, you will not have to worry about the maintenance man anymore.

o

Techo Time

with Geoff Trott

Formatting disks

How did you go at recognising valid interlace patterns in the CorComp tables? My answers are: for single density, numbers 1, 2, 4, 5, 7, 9 and 10 with 1, 5 and 9 not very useful; for double density 1, 2, 4, 5, 6, 8 and 10 are reasonable but 5 and 8 are not very useable. I shall return to these later, once I have examined the timing of various operations. Firstly, I must tell you about the systems I used to make the timings.

The first of these has a Myarc hard and floppy disk controller with one hard disk, one 80 track floppy disk and one 40 track floppy disk with 8 millisec head step rate. It has a PEBox with two Horizon RAMdisks, Mechatronics 80 column card and memory expansion on the 16 bit bus inside the console. This is a speedy system which should give the least delays due to instruction execution and should give similar results to a system with a Myarc floppy disk controller. The other system has a Mini-PE system with 32K memory expansion and RAMdisk and two 40 track floppy disks both with 3 millisec head step rate. This is a more standard system and gives access to a CorComp type disk controller. I have not tried a TI disk controller but will make some comments about that at the end. One comment at this time is that the TI controller is only single density and has a fixed interlace which cannot be changed except by not using the format routine in the controller card.

One of the advantages of the Myarc disk controllers and their disk manager programs is that they allow the interlace to be changed over the complete range possible. My methodology was to format a number of disks (single sided only) to different interlaces on the Myarc system and then to run various tests on these using all the available disk manager programs. Then I moved to the CorComp system and, using the same disks, ran the same tests, as far as possible, on that system with all possible disk managers. The use of the same disk managers doing the same tests on the two systems showed how the differences in the systems changed the results. The tests I used were the following: verification or validation; copy files from RAMdisk to floppy; and clone. The interlace patterns I chose to test were: single density, 4 and 5; and double density, 5, 6, 7, 8, 9 and 10. I chose to do more double density ones as they give a better timing resolution with each sector being read in about 1/90 seconds against 1/45 seconds for single density. These interlace patterns were chosen after format tests were run using all the patterns and the Myarc controller. Those timings follow.

Time in seconds for format and verify with Myarc HFDCC

Interlace Number	9 sectors per track	18 sectors per track
1	97	not done
2	103	175
3	97	170
4	50	170
5	58	66
6	66	74
7	74	74
8	81	81
9	51	97
10		97
11		105
12		115
13		122

I did not try the rest of the double density interlace numbers as the pattern seemed clear. It is interesting that an interlace number of 9 for the single density case appeared to default to the fastest interlace number of 4. These times were done with my analog wrist-watch which has introduced its own variations but there are some interesting results here. First note that the format time includes about 17 seconds for the actual format, which means about two revolutions each track for writing the data to the disk and the rest of the time is spent in the verify. There is less time between sectors in the double density case so that a higher interlace is needed for double density before it settles down and it is not until interlace 7 that there is enough time for the double density to change to the next track without an extra revolution. Checking this theoretically, interlace 7 should take 7 * 40 / 5 = 56 seconds to verify, which with 17 seconds for the format, plus some time to move from track 40 to track 0, is just about spot on. Notice also that the times jump by 8 seconds approximately each time, which is the time for one extra revolution per track. The reason that interlace 6 requires the extra revolution in the double density case is probably the fact that there is no gap on changing tracks between consecutive sectors, as it is one of those numbers with a common factor with the number of sectors per track. By using a skew between tracks, this could be avoided making this interlace number more usable. The large times for low interlace numbers are expected, as there is not enough time so that one revolution is required for each sector. This means 9 * 40 / 5 + 17 = 89 seconds for single density and 18 * 40 / 5 + 17 = 161 seconds for double density. There are more wasted revolutions than that, probably from changing tracks or perhaps errors in measurements.

Looking at these results, it seemed not worth considering interlace numbers below 4 in single density and below 5 in double density. I then formatted 8 disks, two at single density (interlaces 4 and 5) and six at double density (5, 6, 7, 8, 9 and 10). Then I ran a validate or similar test on each disk with the following disk managers; Myarc MDM5 Version 1.30, Funnelweb 4.31 Diskreview and Disk Utilities 4.2 on the Myarc controller. The results were as follows.

Validation time with disk managers and HFDCC

Disk	MDM5	FWDR	DSKU	Theoretical
SD04	41	33	41	32
SD05	41	42	44	40
DD05	49	48	153	40
DD06	56	56	57	48
DD07	56	56	57	56
DD08	65	64	65	64
DD09	81	80	81	72
DD10	81	83	83	80

Moving to the CorComp system, it is now not possible to use MDM5 but CorComp supplied their own disk manager (version 2.3) which was used in its place. Using the same disks and the other possible disk managers gave the following results. Disk Manager 1000 version 3.5 was used also for later tests.

Validation time with disk managers and CorComp FDC

Disk	CC	FWDR	DSKU	Theoretical
SD04	37	33	41	32
SD05	41	42	42	40
DD05	48	51	153	40
DD06	56	56	57	48
DD07	65	65	65	56
DD08	73	73	73	64
DD09	81	81	81	72
DD10	81	82	81	80

It is clear that the HFDCC system is faster in operation than the CorComp system and the Funnelweb Diskreview is as fast in general as the supplied disk manager for the particular disk controller. Disk Utilities is rather slow although this only really shows up in double density interlace 5. It may be the time taken to update the screen display that slows it down.

The next test was to copy a number of files to the disk from RAMdisk to test the writing times to see if they were affected by the interlace. The files were all 33 sectors long program files and 10 were copied to the single density disks and 18 to the double density disks. Unfortunately, the CorComp disk manager would not talk to the RAMdisk, so a separate set of results were run copying from disk to disk using the interlace 7 disk to read from. This was only done on the CorComp system and

some runs were done with Diskreview for comparison purposes.

Copy time with disk managers and HFDCC

Disk	MDM5	FWDR	DSKU	DM1000
SD04	54	60	83	58
SD05	55	63	83	63
DD05	68	79	133	190
DD06	75	86	133	93
DD07	75	89	132	95
DD08	81	94	142	103
DD09	95	113	153	117
DD10	95	114	166	117

Copy time with disk managers and CorComp FDC

Disk	CC	FWDR	FWDR	DSKU	DM1000
SD04	113		135	153	130
SD05	113		135	154	131
DD05	203	249	215	258	245
DD06	176	252	223	252	219
DD07			233	265	228
DD08	200		239	277	237
DD09	202		243	278	246
DD10	205		245	287	247

It is clear that the disk manager written for a particular controller card is faster than the other general disk managers but Funnelweb Diskreview is as fast if not faster than any of the general disk managers. Diskreview is the only disk manager able to handle interlace 5 in double density (apart from MDM5) and MDM5 is much faster than the rest as it copies more than one file at a time.

The ability to clone a disk quickly is also important to people like the Software Librarian and the Shop, so this was investigated and gave rather peculiar results. The Clone program (from the program REDISKIT, written by James Schroeder, modified by Mike Dodd) is written for the CorComp disk controller (Mike Dodd released a version of REDISKIT for the TI controller card) and first reads the format of the disk to be cloned, formats the new disk and then copies one disk to the other, one track at a time. It can be very fast. The times given here are for the whole process of formatting and copying.

Clone times for pre-formatted disks

Disk	Clone	interlace gaps for CorComp number
SD04	35	7
SD05	91	2
DD05	errors	3 and 4
DD06	129	13
DD07	73	7, 9 and 16
DD08	160	2 and 3
DD09	105	2, 9 and 11
DD10	53	5

The numbers are very strange, but even stranger is the fact that the extra time is taken on reading the track while the writing is very quick and appears to be independent of interlace. I can only think that the clone program assumes that the interlace will be according to the CorComp patterns and so tries to read the sectors from the track in that order, which is all over the place except for CorComp number 10 which corresponds to interlace 5, one of the fastest interlace patterns for double density. If you return to the CorComp interlace patterns, you will see that most of these have a number of different gaps between adjacent sectors.

Clone times for CorComp single density formatted disks

No.	Verify	Clone	interlace gaps
1	80	82	1
2	41	91	5
3	33	51	3 and 4
4	57	59	7
5	81	67	2
6	32	51	2, 3 and 5
7	33	35	4
8	49	74	5, 6, 7 and 8
9	81	66	2
10	56	59	7

I have listed these for the equivalent CorComp interlace numbers and they do seem to show what is happening. A further test was run on all the single density CorComp interlace numbers by Rolf and I include those times here to emphasize the point that the CorComp interlace is very unpredictable for the clone operation. The verify was done using Diskreview.

From the above times, it can be seen that the interlace numbers for the CorComp systems are not particularly useful. For single density, the most useful number to use is 7 (interlace 4) with 2 (interlace 5) giving a bit more time between sectors for slow programs. For double density, number 10 (interlace 5) gives the fastest useable interlace but if using slower programs it would be better to use number 2 (interlace 6 and 7). Of course you must use the CorComp disk manager to format your disks to even make that choice so I guess you would like to know what your favourite disk manager formats to when you do not have the choice of changing the interlace. As it turns out, all the disk managers tested here do exactly the same on formatting. With the CorComp system, they appear to format to interlace 4 (CorComp number 7) for single density and to interlace 5 (CorComp number 10) for double density. With the Myarc hard and floppy controller card they seem to format to interlace 4 for both single density and double density. This is OK for single density but gives a very slow disk for double density. Perhaps the way of passing the interlace pattern number is not done the same for the two systems. Since the TI controller card does not allow for the interlace to change, there is no standard way of doing it. Looking at the Myarc documentation, there is a 6 bit field in the density byte of the parameters for the interlace number which, if left blank, should cause a default of 4 for single density and 5 for double density. The TI controller card formats to an interlace of 4 always and of course is only single density.

The whole exercise has been interesting and informative. It would appear that there is clear evidence that interlace 4 (CorComp number 7) should always be used on single density disks but it is not quite as certain for the double density case. Interlace 5 (CorComp number 10) is the best in general, but some operations find this a bit too fast. However to select a different interlace requires access to the disk manager which comes with the controller card and even then with the CorComp disk manager there is not much of a choice available. The best disk manager to use is the one which comes with the controller card except that sometimes it may not have all the facilities you require. Of the general purpose disk managers, my vote goes to Diskreview by Tony McGovern. It is a typical program by Tony which aims to do the job as well as it can possibly be done and with the user constantly in mind in terms of ease of use. I am able to use both the 80 column and 40 column versions and suggest that you should think seriously of an 80 column card as your next upgrade, just to enjoy the benefits of Tony's efforts with Diskreview and the rest of Funnelweb on 80 columns.

If you have any questions or comments on what I have presented here, please write me a note. I have probably glossed over some parts a bit quickly as I found it difficult to explain in simple terms what is happening on the disk and relate it to all the different results. Also, there is scope for more investigation on the clone program and formatting on the HFDCC. In fact, I think that the default for the HFDCC does not work as specified in the manual. This has been noted above and I also found when trying to do a backup, that if I allowed the system to format the disks as it was doing the backup (presumably with the default interlace as the question was not asked), the backup was extremely slow, while if I formatted the disks prior to doing the backup, all went much faster. Since I was using 80 track drives at the time the difference was tremendous. o

continued from page 16

```
034  RETURN  Copyright Martin A. Smoley
035  *                          1989
036  *                    TDT6/C
```

o

TI-Bits Number 6

by Jim Swedlow, CA USA

[This article originally appeared in the User Group of Orange County, California ROM]

FORMATTING DISK TEXT FILES

This month we will explore further into using TI Writer and disk files as output. Two simple utility programs accompany this article.

First, a bit about what the Text Formatter does. If you include the command ".FI;AD", the Formatter will right justify your text (so both the right and left columns are straight lines). When you save a file to disk from the Editor, however, you have a "ragged right" (or not right justified). If you want right justification on disk (and to use the other features of the Formatter), all you do is specify a disk file name as the Print Devicename in the Formatter.

There is a small hitch. Each and every line in the disk file will end in a line feed <CHR$(10)>. Then if you print that file without adding ".LF" to the printer name, your text will be double spaced. It will even be stranger if you use underlining and bold face.

The reason is that the Formatter expects to output to a printer. Since line feed and carriage return are about the only two universal printer command codes, the folks who wrote TI Writer had to come up with a way to do bold face and underline using only those two commands.

Here is what they did. Most printers will advance the print one line when they receive a line feed and return the print head to the left column when fed a carriage return.

To underline a word, print the line, execute a carriage return (so that the print head goes back to the beginning of the same line) and print underline characters (FCTN U) under the word to be underlined. Then send line feed and a carriage return and start the next line. Bold face is similar except that TI Writer prints the bold face word four times.

You add ".LF" to the printer name in the Formatter so that TI Writer can control when line feeds are sent. All of this is fine for a printer but not for a disk file.

If you are going to save your formatted text to disk, first do NOT use either bold face or underline. After you have run it through the Formatter, you must load the formatted file into the Editor and then save it back to disk. Why? Well, if a line has 80 characters, the Formatter will add an LF to the end making it 81 characters long. Then when a basic program attempts to read that line, it will lock your system up. By loading and saving through the editor, all lines are trimmed if they are over 80 characters long. Be sure and use Print File to save the file so that the Editor will not add the tabs (see last month's column).

Then use the program LF STRIPPER (elsewhere in this issue) to strip the line feeds from the ends of the lines.

QUOTE OF THE MONTH

"Computers are charting a new course in human history from the age of the muscle to the age of the mind."

---Author unknown

CARRIAGE RETURNS

Sometimes when you load a text file into the Text Editor there are no carriage returns at the end of the paragraphs. This can cause some serious problems. With TI Writer, if you Reformat or Replace String, you will find all of your paragraphs merged into one huge one (FUNELWRITER will not do this).

The other program this month, CR ADDER, will add carriage returns at the end all paragraphs and to all blank lines. It also adds a carriage return to the end of lines that start with a period as they are probably Text Formatter commands.

A note about this program. One thing I had to resolve was how to add a carriage return to a line that was already 80 characters long. After a bit of experimenting, I came up with this (assuming that A$ is the line and C$ is CHR$(13), the carriage return):

```
PRINT #2:A$;C$
```

Just as in printing to a printer, the semi-colon will ensure that the on disk file is properly set up.

However, I could have used this code:

```
PRINT #2:A$\C$
```

This works because the disk controller automatically breaks strings that are longer than the specified record length into record length pieces.

WORD OF THE MONTH

BLATHERSKITE: A person given to voluble, blustery, empty talk; a talkative, foolish person.

"Educators accuse politicians of being blatherskites, compromisers and opportunists. In turn, politicians see educators as stuffy, sanctimonious prigs who are out of touch with reality."

A TI-WRITER TIP

If you want more than one word in a row to be bold face you can start the first word with the "at" sign (SHIFT 2) and then connect the following words with the exponentiation or circumflex or required space (SHIFT 6). This is fine except that the Text Formatter will see all of them as one hugh word and you may have some strange spacing if you are right justifying.

Another way is to place the "at" sign before each word you want in bold face. This also works with underlining if you do not want spaces between words underlined.

Enjoy.

```
100 ! CR ADDER
110 ! BY JIM SWEDLOW
120 ! OCTOBER 22, 1986
130 !
140 CALL CLEAR :: PRINT " Carriage Return Adder": :
150 INPUT "Old File: DSK":A$ :: PRINT :: INPUT "New
    File: DSK":B$
160 PRINT :"Working"
170 OPEN #1:"DSK"&A$,INPUT :: OPEN #2:"DSK
    "&B$,OUTPUT :: C$=CHR$(13)
180 IF EOF(1)THEN 250 ELSE LINPUT #1:A$
190 IF A$=" " OR A$="" THEN PRINT #2:C$ :: GOTO 180
200 IF ASC(A$)=46 THEN PRINT #2:A$;C$ :: GOTO 180
210 IF EOF(1)THEN PRINT #2:A$;C$ :: GOTO 250 ELSE
    LINPUT #1:B$
220 IF B$=" " OR B$="" THEN PRINT #2:A$;C$:C$ :: GOTO
    180
230 IF ASC(B$)=46 THEN PRINT #2:A$;C$:B$;C$ :: GOTO 180
240 PRINT #2:A$ :: A$=B$ :: GOTO 210
250 CLOSE #1 :: CLOSE #2 :: PRINT :"Done" :: STOP
```

```
100 ! LF STRIPPER
110 ! BY JIM SWEDLOW
120 ! OCTOBER 22, 1986
130 !
140 CALL CLEAR :: PRINT "Line Feed Strippe r": :
150 INPUT "Old File: DSK":A$ :: PRINT :: INPUT "New
    File: DSK":B$
```

continued on page 22

XB tips Number 7

by Jim Swedlow, CA USA

[This article originally appeared in the User Group of Orange County, California ROM]

PRODUCT REVIEW: ADVANCED DIAGNOSTICS
by Millers Graphics

I have only had this a few days so this is a preliminary reading. This product is so impressive, however, that it is worth your serious consideration.

Advanced Diagnostics does quite a few things:

- Checks your disk drive's motor speed and operation.
- Checks all of your computer's RAM.
- Checks, displays, edits and copies disks, sector by sector.
- Dumps your screen (including a disk sector's contents) to any device -- printer, etc.
- Formats and checks disks.
- Works with both the TI and CorComp disk controllers.

What is more, your can create command files that allow you to program the 45 commands in Advanced Diagnostics in any sequence you may need. For example, one of the seven Command Files that comes with this will format a box of disks.

This is a powerful disk diagnostic and control LANGUAGE. It costs $20 plus shipping and requires Extended Basic, Editor/Assembler or Mini Memory.

On a scale of 1 to 10 this is an easy 9.5. Well worth the cost and a valuable tool for the experienced programmer.

WHEN ALL ELSE FAILS . . .

TRACE and UNTRACE are powerful debugging tools. The other day I was looking something up in the Extended Basic book and discovered that these can be used as statements in a program line. This makes them all the more helpful in catching that bug!

IMAGE and USING

Extended Basic left justifies all printed and displayed strings and numbers. While this is correct for strings, numbers should be lined up by the decimal point. One hard way to fix this is to turn your numbers into strings and add leading spaces. PRINT USING is much easier.

Enter and run this program:

```
100 FOR I=9 TO 10 STEP .1
110 PRINT USING "##  ##.#  #.##":I,I,I :: NEXT I
```

Note how the first column prints the number rounded to the nearest whole number while the other two display decimals. The string of asterisks shows you that the number was too big for the space allotted

Only the number on the screen is rounded -- the number in RAM is NOT rounded.

The statement after USING can be a string, a string variable or a line number that refers to an IMAGE statement. Examples:

```
10 A$="The answer is ###.##":: B$="John" :: C$="Dear"
20 IMAGE ## + ## = ###
30 PRINT USING A$:2.45678
40 DISPLAY AT(1,1):USING 20:14,19,14+19
50 PRINT #1, USING "$###.##":23.1
60 PRINT USING C$&"#####":B$
```

For more details, look in the Extended Basic book. Remember, when all else fails . . . .

You can make your output look more professional with these tools.

PAYMENT

I wrote this program to find out how long it would take to pay off a charge account if I paid the minimum. It has been modified to show a number of ways to use USING.

A note on the program: the answers are approximations. Most lenders these days have complicated formulas to determine interest while this program uses the old simple formula.

REDO

You are editing a line. As you add some code, the end of the line, including the closing quote mark, disappears. You press ENTER and get an error message. All of your work is lost? No. Press FCTN REDO and your line, complete with errors, will reappear. Now you can correct it and finish editing your program.

Happy programming!

```
100 ! PAYMENT
110 ! VERSION Extended Basic.1.1
120 ! 11 MAR 85
130 ! BY JIM SWEDLOW
140 !
150 DISPLAY AT(6,9)ERASE ALL :"LOAN PAYMENT": : :"Solve
    for monthly <P>ayment or <N>umber of months to
    payoff loan or <E>nd? N"
160 DISPLAY AT(13,1):"Loan Name ": :"Loan Amount "
    :"Annual Interest Rate " :: B$= "### Payments of
    #######.## "
170 ACCEPT AT(11,27)SIZE(-1)VALIDATE("PNE")BEEP:A$ :: IF
    A$="E" THEN STOP ELSE ACCEPT AT(13,14)SIZE(-14)BEEP
    :N$ :: DISPLAY AT(21,1): : : : :
180 DISPLAY AT(15,18):A
190 ACCEPT AT(15,19)SIZE(-9)VALIDATE(NUMERIC)BEEP:A ::
    IF A<1 OR A>1000000 THEN 190 ELSE DISPLAY AT(15,14)
    :USING " #######.##":A
200 DISPLAY AT(16,22):R
210 ACCEPT AT(16,23)SIZE(-8)VALIDATE(NUMERIC)BEEP:R ::
    IF R<1 OR R>30 THEN 210 ELSE Z=R/1200 :: DISPLAY AT
    (16,23 ):USING "##.###":R
220 Y=LOG(100*A*Z+1)/LOG(1+Z):: Y=MIN(INT(Y),720):: X=A*
    Z/(1-(1+Z)^(-Y)):: X=INT(X*100+.5)/100
230 DISPLAY AT(21,1):USING 350:X,Y :: L=0 :: IF A$="N"
    THEN 270
240 DISPLAY AT(19,1):"Loan Life in Months";INT(N)
250 ACCEPT AT(19,21)SIZE(-3)VALIDATE(DIGIT)BEEP:N :: IF
    N<1 OR N>Y THEN 250 ELSE DISPLAY AT(19,21):USING "
    ###":N
260 P=A*Z/(1-(1+Z)^(-N)):: P=MAX(INT(P*100+.5)/100,X)::
    F=N :: GOTO 320
270 DISPLAY AT(19,1):"Monthly Payment ";P
280 ACCEPT AT(19,20)SIZE(-7)VALIDATE(NUMERIC)BEEP:P ::
    I F P<X OR P>A*(1+Z)THEN 280 ELSE DISPLAY AT(19,17):
    USING " #######.##":P
290 P=INT(P*100+.5)/100 :: N=LOG(P/(P-A*Z))/LOG(1+Z):: N
    =MIN(N,Y):: F=INT(N)::IF N-F<.01 OR N-F>.99 THEN
    F=INT(N+.1):: GOTO 320
300 L=A*(1+Z)^F-P*((1+Z)^F-1)/Z :: IF L<.01 THEN L=0 ::
    N=F ELSE IF L<.1*P THEN F=F- 1 :: L=L+P ELSE L=L*
    (1+Z)
310 L=INT(L*100+.5)/100 :: IF F<1 THEN F=1 :: P=L :: L=0
320 DISPLAY AT(21,1):USING B$:F,P :: IF L THEN N=F+1 ::
    DISPLAY AT(22,1):USING 340:L,N ELSE DISPLAY AT(22,1)
    : :
330 DISPLAY AT(24,1):USING "Total Interest ########.##"
    :F*P+L-A :: GOTO 170
340 IMAGE Last Payment is ######.## Number of Payments
    ###
350 IMAGE Minimum Payment ######.## Maximum Loan Life
    ###
```

I Like Brain Games

by Jim Peterson, Tigercub Software, USA

I do not much care for those fast-action arcade type games - the dodge-the-parman, climb-the-ladder, shoot-the-alien type of thing. My grey-haired reflexes are too slow, and my 8-year old grandson can play rings around me.

And I HATE those adventure games that do nothing but print out responses that "I do not know how to do that" or "you cannot go that-away". Sounds too much like the SYNTAX ERROR or BAD VALUE messages that I get when I am trying to write a program!

But I do like brain games! - the ones that challenge me to exercise the grey cells under my grey hair, and give me plenty of time to do so. I also enjoy programming that type of game - although they have certainly proven to be the least popular of anything I have ever done.

The world's premier brain game, of course, is chess. I cannot comment much on that, because I do not know the game - other than the wild Japanese version, where every piece that reaches enemy territory can be promoted and every captured piece can be placed back on the board as your own. I wish that someone would program that game!

Anyway, Western-style chess is available as an old Texas Instruments module and as a public domain program translated by Swiridenko from a version written for some other computer. From reviews, I understand that neither offers much of a challenge to an expert, but that either one is a worthy opponent for an average player.

There are also a couple of TI computer games based on chess. The Queen Board Game, public domain by D. Decker, is a real challenge. Hexapawn is an early computer classic from Ahl's days; the computer starts out by knowing nothing but learns from its mistakes and, after a few games, becomes unbeatable!

The blue-collar, redneck equivalent of chess is checkers. Several versions have been written for the TI, all apparently from scratch. Their programmers deserve credit for tackling a complex subject, but any of their games can be easily beaten by a beginner.

The favourite game of most of Africa, and dating back to 2000 B.C. in the Middle East, is Mancala, also known as Awari or Mawari in other African languages. It is commonly played with pebbles placed in holes dug in the dirt, or gouged out of a slab of wood. Several public domain versions exist, but the best game by far is the assembly version called Mancala, copyrighted in 1982 by Aldebaran and finally released recently by Triton.

Othello is an American board game, based on ancient Oriental games, in which the object is to capture territory by placing markers at both ends of a row. Its weakness is that the player who goes first is at a distinct disadvantage. Several public domain versions have been released for the TI, all quite slow. Dean Cleveland's was the first. I like the version by Rick Mirus, which has a black board. Nguyen Long in France wrote the version which is most difficult to beat but it is also very slow, presumably because the computer researches each move one step further.

Go or Gomoku is a simpler game in which the object is to get 5 markers in a row before your opponent blocks you. There are several public domain versions, but the best by far is Links by Curtis Alan Provance, a unique variant with many features not found elsewhere.

A variant of this game, popular as a toy several years ago but a really challenging brain game, involves stacking chips to get four in a row either vertically or diagonally. One of the best versions was written in the Netherlands.

Tic-Tac-Toe is a child's game, too simple to be called a brain game, but there are 3-dimensional versions, by various authors, which are much more challenging. I have been planning, for years, to write a version in which, if the first player gets 3 in a row and the second player can counter with 3 in a row, the game continues.

The 15 Puzzle was originally a pocket game, consisting of fifteen tiles numbered 1 to 15, randomly arranged in a 4x4 grid, movable but locked within the grid by a frame. The challenge was to slide the tiles around until the numbers were in sequence. The promoter sold hundreds of thousands by offering a large reward to anyone who could solve the puzzle - but his version was impossible to solve! Some public domain computer versions are also impossible, because the programmer has assumed that any random arrangement of numbers was possible. The Texas Instruments version, sold in the early days on cassette, correctly started out with a properly sequenced grid in memory and then scrambled it by a series of random moves. My version did the same, and also offered the option of having two players take turns solving the same puzzle.

Many puzzle games are based on determining, by a series of educated guesses, the sequence in which the computer has randomly arranged coloured squares or what have you. These are most frequently called Master Mind, and the most ambitious was written in assembly, occupying 322 disk sectors (!), by J-L. Bazanegue in France.

Peg Jump was an old favourite board game in which holes on a board, in the form of a cross, were filled with pegs. The object was to jump pegs over each other, removing jumped pegs as in checkers, until only one peg remained in the center hole. Texas Instruments sold a good version of this on cassette; Regena wrote another fine version. Many many years ago I owned one of these puzzles which was accompanied by a little booklet showing about 50 "end games." If I could find that booklet again, it would be fun to program these end games into the TI or Regena versions.

Games of the "fox and geese" type require moving, or blocking moves, along certain pathways. The best of these in the TI world, and very difficult to beat, are Giants and Dwarfs by Barry Traver and Quintus by Sam Pincus.

Another type requires placing geometric figures within a specified area. This is the basis for the L-Game, originally published in Ahl's Creative Computing by Bill Gardner. I have never been able to beat it. Of the same type, but much less difficult, is my Mechanical Aptitude Test, based on the "broken block" problems of S.A.T. tests and other IQ tests.

Many brain games are based on a mathematical theorem or a mathematical progression. These are almost impossible to win until you have puzzled out the secret, and too easy to win thereafter. An example is Pick Up Sticks, in which you and The computer take turns picking up 1 to 3 sticks from a pile of random size, with the player who gets the last stick being the loser. In my version, after the user has lost several games, the computer changes the rules to specify that whoever gets the last stick is the winner - but the computer can still win every time! My Can of Worms lets the user make up all the rules, but he still loses - and Nimbo, based on the Fibonacci series of numbers, is almost impossible to win without knowing the secret.

Other mathematical puzzles depend on logical thinking. Barry Traver wrote a series of three, based on the number 31, which appeared in a recent Genial Traveler. I have written several, mostly as "tinygrams" or short programs in my Tips From The Tigercub. Regena recently published in Micropendium her ingenious Magic Boxes which has several skil levels ranging from fairly easy to extremely difficult.

Most card games played against the computer, such as Twenty-One or Blackjack, are based on pure luck

rather than skill or brainwork. There are also various poker games, but I doubt that anyone has yet programmed on the TI - perhaps not on any computer? - the true odds on a poker hand. Arcade Action Software has released a cribbage game which has been reviewed highly, but I have not seen it - nor do I know how to play the game.

Most solitaire card games are based on pure luck. Quality 99's QS-Solitaire is a beautifully programmed solitaire game in assembly, but it is the standard Klondike game in which no real skill is involved. However, Walt Howe's Chainlink Solitaire is my favourite of all the brain games ever programmed for the TI-99/4A. In this version of solitaire, all cards are visible, so an intelligent choice of moves is available - and an option is available to replay the hand by a different method, if the first try ends in failure. The later fairware versions of this program, with assembly links, are very fast. The commercial version, with ribbons of cards streaming between piles, is something to be seen! Regena published in Micropendium a Poker Solitaire game which also lends itself to some intelligent playing.

Word games are still another category which requires some brainwork - although I would consider the mental exercise to be minimal in the popular wordsearch puzzles, the object of which is to find each of a list of words within a grid of letters. Texas Instruments had this on a cassette. My version offers a somewhat more challenging option, to find words of a specified category which are not listed.

Cryptograms are perhaps the most challenging of word games, but as far as I know, no one has programmed a diskful of them for the TI-99/4A. The simplest word game is Scramble, in which the letters of a word have been reassembled into a random arrangement. I wrote one of those, as did everyone else, but I also wrote a more challenging version called Scrambulation, in which each word of a sentence is scrambled and, optionally, the sequence of words is also rearranged. I also wrote Squinch, which Jack Sughrue described as a fiendish game - two words with their letters randomly intermingled into one. And I wrote Bazoo, in which you must find a word by guessing 5 letters at a time, and Changeroo in which you must change one word into another by changing a letter at a time, making a valid new word each time.

However, I believe that the most unique word game ever written for the TI is Karl Romstedt's Superjot, into which he has programmed every 3-letter word in the English language. You and the computer each select a word, and try to guess each other's word - the computer wins more often than not!

Memory games also qualify as brain games, I believe. The most popular is Concentration, originally based on remembering the locations of pairs of cards in a deck scattered face down. Computer versions, such as my Match A Patch, normally use graphics patterns rather than cards.

Other memory games are based on remembering a sequence of numbers or colours, etc. - these are the Simon games. The most viciously difficult of these is one that I wrote several years ago called Nervous Breakdown - it challenges you to simultaneously remember the sequence of three flashing colours, the highest and lowest of three numbers, and the highest and lowest of three tones!

Maze games, if played intelligently rather than by guesswork, are also brain games. The best of these are the "hallways" type which graphically depict your progress through the maze in 3-dimensional graphics.

And there are many other types of brain games - the many versions of the Towers of Hanoi; coin switching puzzles and coin weighing puzzles and liquid measuring puzzles; the classic Nim, and the other classic computer puzzles such as Black Box, Explosion, and others. And I hardly know where to classify some that I have written, such as Reverso and Bassackwards - and Preachers, Lawyers and Salesman.

continued on page 14

Putting it All Together #8

by Jim Peterson, Tigercub Software, USA

The hard part of learning to program is not in learning what the various commands do - it is in learning how to put them together to do what you want them to do!

Key in this little program and run it to see what it does, then read the explanation of how it does it.

In the early days, when computers had tiny memories, much emphasis was placed on efficient programming - the pioneer David Ahl called it "elegant" programming. The old 99'er magazine published some one-liners. My Tips From The Tigercub contained some one-line programs, even some no-line programs that could be keyed in and run in the immediate mode. In order to cram over a hundred subprograms onto a disk, I made great use of compact programming techniques on my Nuts Bolts disks. Later, Mike Stanfill originated the name "tinygram" for a program that would fit on one screen, and wrote some great ones. Ed Machonis wrote a disk full of tiny printer utilities.

Richard Mitchell in his Super 99 Monthly once called me the "king of the one liners", but this title rightly belongs to John Martin. The following is an example.

```
1 IF F THEN INPUT #1:A$,A,J,K :: IF J THEN PRINT A$;TAB(
12);J;TAB(18);SEG$(B$,ABS(A*2)+1,2);K;TAB(27);A<0 :: GOT
O 1 ELSE RUN ELSE B$="AVDFDVIFIVPG" :: INPUT "DSK":F ::
OPEN #1:"DSK"&STR$(F)&".",INTERNAL,RELATIVE,INPUT :: GOT
O 1 !BY JOHN M
```

An undefined numeric variable has a value of 0 which is the value of F when the program is first run. IF F THEN is interpreted as "if F is other than 0" so program execution jumps to the first unpaired else. IF J is paired with ELSE RUN so execution jumps to ELSE B$; a string is assigned to B$, the INPUT asks for a disk number, and file #1 is opened, without a filename, as an internal relative file, for input. When it is opened, the first sector of the disk can be read; it contains information regarding the disk and its contents. GOTO 1 goes back to start over. The variable F now as a value other than 0 (from the INPUT disk number) so the values for A$, A, J and K are read from the disk. On the first pass, these are the disk name, a 0, the number of sectors initialised, the number of sectors available, and a 0. IF J THEN is interpreted as "if J is other than 0" and it is because it contains the number of sectors, so the disk name is printed, followed by the number of initialized sectors at tab 12. Since a 0 was read into A, the ABS(A*2)+1 is 0 times 2 plus 1, which is 1, so the segment of "AVDFDVIFIVPG" starting with the first character and consisting of two characters (AV) is printed (meaning "available"), followed by the number of available sectors read into K (preceded by a space because it is numeric). Since a 0 was read into A, the statement A<0 (A is less than 0) is false and has a truth value of 0, so a 0 is printed at tab 27. Execution returns to the beginning, and values are read into the variables again. Now, A$ will be a filename. A will be a number from 1 to 5, indicating the type of file - 1 for display fixed, 2 for display variable, 3 for internal fixed, 4 for internal variable, 5 for a program. If the file is protected, the number will be negative. J will be the number of sectors occupied by the file, and K will be the record length of the file (0 in the case of a program). The filename is printed, and its sector length at tab 12. ABS converts the A from negative to positive, if necessary, and the formula selects the letters DF, DV, IF, IV or PG to print, followed by the record length from K. If the file is protected, A has a negative value and A<0 therefore has a truth value of -1, otherwise a 0, printed at tab 27. Execution goes back to the beginning and this continues until blank records are read. J will then have a value of 0 so execution jumps to ELSE RUN, which re-runs the program, thereby zeroing out the value of F.

```
1 REM COPYRIGHT 1984, EMERAL
D VALLEY PUBLISHING CO.
100 REM ******************
110 REM * BOOLEAN BRAIN *
120 REM ******************
130 REM BY WILLIAM K. BALTHR
OP
140 REM HOME COMPUTER MAGAZI
NE
150 REM VERSION 4.4.1
160 REM TI EXTENDED BASIC
170 REM
180 RANDOMIZE :: CALL CLEAR
:: CALL SCREEN(2):: RESTORE
930 :: FOR Z=0 TO 20 :: READ
 A,A$ :: CALL CHAR(A,A$):: N
EXT Z
190 CALL MAGNIFY(3):: FOR Z=
1 TO 9 :: CALL SPRITE(#Z,128
,7,210,100):: NEXT Z
200 DIM R$(24),L$(19),RM$(10
,2),GT(9,5),G(9)
210 FOR Z=1 TO 24 :: READ R$
(Z):: NEXT Z :: FOR Z=1 TO 1
9 :: READ L$(Z):: NEXT Z
220 DEF RV(C)=VAL(SEG$(RM$(R
,2),C,1))
230 FOR Z=1 TO 10 :: READ RM
$(Z,1),RM$(Z,2):: NEXT Z ::
R=1 :: SC=0 :: DR=0 :: I=0 :
: DIR=1
240 FOR Z=1 TO 9 :: CALL COL
OR(Z,2,8):: READ GT(Z,1),GT(
Z,2),GT(Z,3),GT(Z,4),GT(Z,5)
:: NEXT Z
250 GOSUB 360
260 GOSUB 920 :: IF K=69 THE
N DIR=1 ELSE IF K=87 THEN DI
R=2 ELSE IF K=78 THEN DIR=3
ELSE IF K=83 THEN DIR=4 ELSE
 GOTO 260
270 IF R<>RV(DIR)THEN 320
280 FOR Z=1 TO 40 :: CALL CO
LOR(RND*12+1,RND*12+1,RND*12
+1):: CALL SOUND(-100,-3,RND
*9):: NEXT Z :: FOR Z=1 TO 8
 :: CALL COLOR(Z,16,2)
290 NEXT Z :: CALL SCREEN(2)
300 CALL CLEAR :: DISPLAY AT
(12,1):"YOU HAVE BEEN ZAPPED
 BY A   BAD DISK SECTOR.":"T
HE COMPUTER LOCKS UP, AND  Y
OU ARE LOST FOR EVER."
310 GOTO 900
320 IF RV(4+DIR)=1 THEN R=RV
(DIR):: GOSUB 370 :: GOTO 26
0
330 GOSUB 430 :: RM$(R,2)=SE
G$(RM$(R,2),1,DIR+3)&"1"&SEG
$(RM$(R,2),5+DIR,4-DIR):: R=
RV(DIR)
340 IF DIR=1 OR DIR=3 THEN T
D=DIR+1 ELSE TD=DIR-1
350 RM$(R,2)=SEG$(RM$(R,2),1
,TD+3)&"1"&SEG$(RM$(R,2),5+T
D,4-TD):: IF R=0 THEN 780 EL
SE 250
360 CALL CLEAR :: CALL COLOR
(1,2,12,2,12,6,4,2,12,8,2,8,
9,2,8,10,2,8,11,12,6,12,12,6
):: FOR Z=1 TO 24 :: DISPLAY
 AT(Z,1):R$(Z);:: NEXT Z
370 CALL VCHAR(1,31,106,96):
: DISPLAY AT(24,2)SIZE(26):R
PT$(" ",26):: DISPLAY AT(24,
(28-LEN(RM$(R,1)))/2+1)SIZE(
LEN(RM$(R,1))):RM$(R,1)
380 ON DIR GOSUB 390,400,410
,420 :: RETURN
390 DISPLAY AT(11,18)SIZE(1)
:"E" :: CALL COLOR(12,12,7-R
V(5),11,12,7-RV(7),2,12,7-RV
(8)):: RETURN
400 DISPLAY AT(11,18)SIZE(1)
:"W" :: CALL COLOR(12,12,7-R
V(6),11,12,7-RV(8),2,12,7-RV
(7)):: RETURN
410 DISPLAY AT(11,18)SIZE(1)
:"N" :: CALL COLOR(12,12,7-R
V(7),11,12,7-RV(6),2,12,7-RV
(5)):: RETURN
420 DISPLAY AT(11,18)SIZE(1)
:"S" :: CALL COLOR(12,12,7-R
V(8),11,12,7-RV(5),2,12,7-RV
(6)):: RETURN
430 CALL CLEAR :: CALL COLOR
(1,2,2,2,3,2,8,3,2,9,7,2,10,
7,2,3,16,2,4,16,2):: FOR Z=1
 TO 19 :: DISPLAY AT(Z,1):L$
(Z);:: NEXT Z
440 FOR Z=1 TO 9 :: GT(Z,1)=
INT(RND*2+1):: G(Z),GT(Z,2),
GT(Z,3)=0 :: CALL PATTERN(#Z
,128+(GT(Z,1)-1)*4):: CALL L
OCATE(#Z,GT(Z,4),GT(Z,5))
450 NEXT Z
460 CL=ABS(CL-1):: FOR Z=1 T
O 9 :: IF G(Z)=0 THEN CALL C
OLOR(#Z,6+CL)
470 NEXT Z :: CALL COLOR(9,6
+CL,2,10,6+CL,2):: CALL KEY(
0,K,S):: IF S=0 THEN 460
480 IF K<48 OR K>57 THEN 460
490 SC=SC+1 :: ON K-47 GOTO
590,500,510,520,530,540,550,
560,570,580
500 IF GT(1,2)=1 THEN GOTO 4
60 ELSE GT(1,2)=1 :: CALL HC
HAR(2,4,45,2):: IF GT(1,1)=1
 OR(GT(1,1)=2 AND GT(1,3)=1)
THEN GOTO 600 ELSE 460
510 IF GT(1,3)=1 THEN GOTO 4
60 ELSE GT(1,3)=1 :: CALL HC
HAR(3,4,45,2):: IF GT(1,1)=1
 OR(GT(1,1)=2 AND GT(1,2)=1)
THEN GOTO 600 ELSE 460
520 IF GT(2,2)=1 THEN GOTO 4
60 ELSE GT(2,2)=1 :: CALL HC
HAR(6,4,45,2):: IF GT(2,1)=1
 OR(GT(2,1)=2 AND GT(2,3)=1)
THEN GOTO 620 ELSE 460
530 IF GT(2,3)=1 THEN GOTO 4
60 ELSE GT(2,3)=1 :: CALL HC
HAR(7,4,45,2):: IF GT(2,1)=1
 OR(GT(2,1)=2 AND GT(2,2)=1)
THEN GOTO 620 ELSE 460
540 IF GT(3,2)=1 THEN GOTO 4
60 ELSE GT(3,2)=1 :: CALL HC
HAR(10,4,45,2):: IF GT(3,1)=
1 OR(GT(3,1)=2 AND GT(3,3)=1
)THEN GOTO 640 ELSE 460
550 IF GT(3,3)=1 THEN GOTO 4
60 ELSE GT(3,3)=1 :: CALL HC
HAR(11,4,45,2):: IF GT(3,1)=
1 OR(GT(3,1)=2 AND GT(3,2)=1
)THEN GOTO 640 ELSE 460
560 IF GT(4,2)=1 THEN GOTO 4
60 ELSE GT(4,2)=1 :: CALL HC
HAR(14,4,45,2):: IF GT(4,1)=
1 OR(GT(4,1)=2 AND GT(4,3)=1
)THEN GOTO 660 ELSE 460
570 IF GT(4,3)=1 THEN GOTO 4
60 ELSE GT(4,3)=1 :: CALL HC
HAR(15,4,45,2):: IF GT(4,1)=
1 OR(GT(4,1)=2 AND GT(4,2)=1
)THEN GOTO 660 ELSE 460
580 IF GT(5,2)=1 THEN GOTO 4
60 ELSE GT(5,2)=1 :: CALL HC
HAR(18,4,45,2):: IF GT(5,1)=
1 OR(GT(5,1)=2 AND GT(5,3)=1
)THEN GOTO 680 ELSE 460
590 IF GT(5,3)=1 THEN GOTO 4
60 ELSE GT(5,3)=1 :: CALL HC
HAR(19,4,45,2):: IF GT(5,1)=
1 OR(GT(5,1)=2 AND GT(5,2)=1
)THEN GOTO 680 ELSE 460
600 CALL COLOR(#1,3):: G(1)=
1 :: CALL HCHAR(2,8,95,2)::
CALL HCHAR(4,10,45,3):: IF G
T(6,2)=1 THEN 460
610 GT(6,2)=1 :: IF GT(6,1)=
1 OR(GT(6,1)=2 AND GT(6,3)=1
)THEN 700 ELSE 460
620 CALL COLOR(#2,3):: G(2)=
1 :: CALL HCHAR(6,8,95,2)::
CALL HCHAR(5,10,45,3):: IF G
T(6,3)=1 THEN 460
630 GT(6,3)=1 :: IF GT(6,1)=
1 OR(GT(6,1)=2 AND GT(6,2)=1
)THEN 700 ELSE 460
640 CALL COLOR(#3,3):: G(3)=
1 :: CALL HCHAR(10,8,95,2)::
 CALL HCHAR(12,10,45,3):: IF
 GT(7,2)=1 THEN 460
650 GT(7,2)=1 :: IF GT(7,1)=
1 OR(GT(7,1)=2 AND GT(7,3)=1
)THEN 720 ELSE 460
660 CALL COLOR(#4,3):: G(4)=
1 :: CALL HCHAR(14,8,95,2)::
 CALL HCHAR(13,10,45,3):: IF
 GT(7,3)=1 THEN 460
670 GT(7,3)=1 :: IF GT(7,1)=
1 OR(GT(7,1)=2 AND GT(7,2)=1
)THEN 720 ELSE 460
680 CALL COLOR(#5,3):: G(5)=
1 :: CALL HCHAR(18,8,95,7)::
 CALL HCHAR(17,15,45,5):: IF
 GT(8,3)=1 THEN 460
690 GT(8,3)=1 :: IF GT(8,1)=
1 OR(GT(8,1)=2 AND GT(8,2)=1
)THEN 740 ELSE 460
700 CALL COLOR(#6,3):: G(6)=
1 :: CALL HCHAR(4,15,95,9)::
 CALL HCHAR(10,24,45,2):: IF
 GT(9,2)=1 THEN 460
710 GT(9,2)=1 :: IF GT(9,1)=
1 OR(GT(9,1)=2 AND GT(9,3)=1
)THEN 760 ELSE 460
720 CALL COLOR(#7,3):: G(7)=
1 :: CALL HCHAR(12,15,95,2):
: CALL HCHAR(16,17,45,3):: I
F GT(8,2)=1 THEN 460
730 GT(8,2)=1 :: IF GT(8,1)=
1 OR(GT(8,1)=2 AND GT(8,3)=1
)THEN 740 ELSE 460
740 CALL COLOR(#8,3):: G(8)=
1 :: CALL HCHAR(16,22,95,2):
: CALL HCHAR(11,24,45,2):: I
F GT(9,3)=1 THEN 460
750 GT(9,3)=1 :: IF GT(9,1)=
1 OR(GT(9,1)=2 AND GT(9,2)=1
)THEN 760 ELSE 460
760 CALL COLOR(#9,3):: G(8)=
1 :: CALL HCHAR(15,27,42,2):
: CALL HCHAR(16,27,42,2):: C
ALL HCHAR(16,26,95)
770 FOR TD=1 TO 1000 :: NEXT
 TD :: ND=ND+1 :: CALL DELSP
RITE(ALL):: RETURN
780 CALL CLEAR :: CALL VCHAR
(1,31,106,96):: FOR Z=1 TO 8
 :: CALL COLOR(Z,12,12):: NE
XT Z :: FOR Z=1 TO 8 :: PRIN
T TAB(Z);"#";TAB(29)-Z;"!"
790 NEXT Z :: FOR Z=1 TO 8 :
: PRINT TAB(9);"AHPXAHPXAHPX
" :: NEXT Z :: FOR Z=8 TO 1
STEP -1 :: PRINT TAB(Z);"!";
TAB(29-Z);"#";:: NEXT Z
800 FOR Z=5 TO 8 :: CALL COL
OR(Z,2,12):: NEXT Z
810 CALL COLOR(1,2,12):: FOR
 Z=1 TO 20 :: FOR Z1=6 TO 9
:: C1=Z1 :: C2=Z1+1 :: C3=Z1
+2 :: C4=Z1+3 :: IF C2>9 THE
N C2=C2-4
820 IF C3>9 THEN C3=C3-4
830 IF C4>9 THEN C4=C4-4
840 CALL COLOR(5,C1,C1,6,C2,
C2,7,C3,C3,8,C4,C4)
```

```
850 CALL SOUND(-100,RND*5000
+1000,0):: NEXT Z1 :: NEXT Z
 :: FOR Z=1 TO 8 :: CALL COL
OR(Z,2,12):: NEXT Z
860 CALL CLEAR :: DISPLAY AT
(12,1):"YOU FOUND THE PROCES
SING    CONTROL ROOM." :: S=
INT(ND/SC/ND*10000)
870 DISPLAY AT(16,1):"YOUR S
CORE:";S
880 IF S>599 THEN DISPLAY AT
(18,1):"YOU REPAIR THE CPU A
ND ARE  RETURNED TO THE REAL
 WORLD." :: GOTO 900
890 DISPLAY AT(18,1):"THE CP
U IS BEYOND REPAIR.   YOU AR
E TRAPPED HERE."
900 DISPLAY AT(22,1):"PLAY A
GAIN  Y / N ?Y" :: ACCEPT AT
(22,20)SIZE(-1)VALIDATE("YN"
):S$
910 IF S$="Y" THEN SC,S,ND=0
 :: RESTORE 1280 :: GOTO 230
 ELSE CALL CLEAR :: END
920 CALL KEY(0,K,S):: IF S=0
 THEN 920 ELSE RETURN
930 DATA 33,0102040810204080
,35,8040201008040201,36,8080
8080808080800000000000000FF
0101010101010101FF81FFFF81AB
D5FF
940 DATA 96,FF,97,8080808080
8080800101010101010101000000
00000000FF,100,FF80808080808
080FF01010101010101
950 DATA 102,80808080808080F
F01010101010101FF00,112,00,1
13,0103070F1F3F7FFF,120,00,4
0,00,41,80C0E0F0F8FCFEFF
960 DATA 60,8182848890A0C080
,61,8141211109050301,106,FFF
FFFFFFFFFFFF00000000FF00000
0010101010000000000000000010
10101
970 DATA 110,01010101FF00000
000000000FF010101,92,00
980 DATA 128,FC3F0F07FF01000
000000107FF0F3FFC0000C0F0F8F
CFEFFFFFEFCF8F0C0
990 DATA 132,FFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFF80E0F8FCFEF
EFFFFFFFFFEFEFCF8E080,45,000
00000FF,95,00000000000000FF
1000 DATA "#
     !"
1010 DATA " #
      !"
1020 DATA "  #
       !"
1030 DATA "   #
       !"
1040 DATA "    #
       !"
1050 DATA "     #
      !"
1060 DATA "      #
     !"
1070 DATA "       #
    !"
1080 DATA "        d````````
``e"
1090 DATA " %%%%   ahhhcccch
hhb   %%%% "
1100 DATA "&pppp$  ahhbxxxxa
hhb  &(((($"
1110 DATA "&pppp$  ahhbxxxxa
hhb  &(((($"
1120 DATA "&pppp$' ahhbxxxxa
'hb  &(((($"
1130 DATA "&pppp$  ahhbxxxxa
hhb  &(((($"
1140 DATA "&pppp$  ahhbxxxxa
hhb  &(((($"
1150 DATA "&pppp$  fccgxxxxf
ccg  &(((($"
1160 DATA "&pppp$ !
    # &(((($"
1170 DATA "&pppp$!
     #&((((($"
1180 DATA "&pppp<
     =((((($"
1190 DATA "&pppq
      )(((($"
1200 DATA "&ppq
       )((($"
1210 DATA "&pq
        )($"
1220 DATA "&q
         )$"
1230 DATA "!
          #"
1240 DATA "","1kk  cc","2kk
  b","      1kkk  ccccccccc"
,"      mkkk          b","3k
k  cca            b","4kk
              b"
1250 DATA "
   b","                    b
","5kk  cc              1kk
c","6kk   b              mkk"
1260 DATA "      1kkk  cc
   b","      mkkk   b      b
","7kk  cca     b      b","8
kk          b      b","
       1kkk  cca"
1270 DATA "            mkkkkk
","9kk  ccccccca","0kk"
1280 DATA KEYBOARD\INTERFACE
,54870000,INPUT\PORT,4369000
0,VIDEO\PROCESSING,25740000,
SOUND\CONTROL\ROOM,12350000,
RAM\ROOM,31480000
1290 DATA DISK\CONTROLERS\RO
OM,78920000,DISK\DRIVE,76130
000,ROM\ROOM,69510000,PORT\C
ONTROL,80260000,CENTRAL\PROC
ESSING,90000000
1300 DATA 0,0,0,9,40,0,0,0,4
1,40,0,0,0,73,40,0,0,0,105,4
0,0,0,137,40,0,0,0,25,96,0
,0,89,96
1310 DATA 0,0,0,121,152,0,0,
0,73,200                    o
```

Lurking Horror part 1

This walk through is by Scorpia,

Baby, it is cold outside! REAL cold. With high winds and heavy snow and drifts up over your head. A good night to be snuggled up in a cozy room with a warm drink. Unfortunately, you are snuggled up in a terminal room with a term paper to write. Only, you will not doing much writing as the evening progresses. Instead, you will be spending your hours with ghosties and ghoulies and things that go bump in the night....some really unpleasant things.

Lurking Horror is a genuine horror story, with real monsters that can cause you some real trouble, unlike the fake ghost in Moonmist. It is a bizarre blend of Lovecraft and Stephen King, with an overlay of high tech. Unfortunately, even when you get to the end, very little is really explained; you can only make surmises as to what has been happening.

Like many Infocoms, Horror is pretty much non-linear, which means most of the puzzles can be solved in almost any order. This walkthru only provides one path to completion; there are others, that you may wish to try for yourself. Remember to save every now and then, in case you make a mistake.

Also, somewhere along the line, you will start to feel tired. When that happens, take a drink of Coke. The caffeine will make you alert again. You do not want to sleep anywhere, because no place is safe. After you have had your sip of Coke, you can drop the bottle.

You will run into an urchin every now and then. Until the proper time comes, just ignore him, but make sure you do not drop anything important while he is around; he will steal it.

So, here you are in the terminal room of GUE Tech, trying to finish your term paper while a blizzard howls outside. The only other person in the room is a hacker (in the original sense of the word), who will prove helpful shortly. Meanwhile, let us get this show on the road.

Turn on the PC and login with the Student number on your ID card. The password is in the back of the docs that come with the game. Now click the menu box, then click your term paper.

Hmmmm....this sure does not look like your paper. In fact, whatever you have here is definitely unpleasant, even though most of it makes little sense (it would not surprise me if someone was working on a translation of the Necronomicon). Keep clicking more until an illustration appears that is so nasty you faint outright (in this day and age, it must be REALLY nasty!).

You find yourself on a lifeless plain, with only one way to go, down. This takes you to a platform, which might be some sort of altar. Right about now, you are probably feeling a bit nervous, and I do not blame you! However, you are here for a purpose, so examine the platform.

Aha! A strange stone. Take it quickly, because something is coming. Something you do not really want to meet. Too bad there is no way you can escape. Oops! Looks like you are caught now....

continued on page 16

Rambles

by Stephen Shaw, England

One item of hardware mods omitted from Mikes excellent compendium (plug) is a Load Interrupt Switch- this appeared on page 46 of issue 12. I now have some information on a program many of you have which can use it! Here once more are the details....

At any time you can by means of a hardware switch tell the computer to stop what it is doing and go to a specific memory location and run the program it finds there. This has been used for example for dumping screen displays to printer, although it is also done (better) using a software interrupt, whereby the computer runs the main program and keeps taking time off to say scan the keyboard for a command key- effectively doing two things at once.

TI have provided us with a program which can use a load interrupt switch, although they did not tell us about it! It is the DEBUG program that comes with Editor Assembler, and this is how you utilise it...

Insert the Editor Assembler module, and have the disk with DEBUG on it in drive one. Choose TI Basic and type in and run this program:

```
100 CALL INIT
109 REM SEE NOTES BELOW RE LINE 110!
110 CALL LOAD(8228,96,0)
120 CALL LOAD("DSK1.DEBUG")
130 CALL LOAD(-4,131,224,112,190)
140 CALL LOAD(8228,160,0)
150 PRINT "PRESS Q THEN ENTER"
160 CALL LINK("DEBUG")
170 END
```

This loads DEBUG. Enter BYE and select an option of Editor Assembler- eg LOAD AND RUN. While the LOAD AND RUN (D/F 80) program you have selected is running- say a game! - press your load interrupt switch and you will be in DEBUG and can have a look around!

For the technically minded, pressing the switch causes the computer to do a BLWP to >FFFC where >FFFC contains the WS pointer and >FFFE contains the Program Counter. Line 130 above puts the addresses into these locations.

Now, if you have DEBUG in memory, then load a program, where is debug going to be? Hmmm? The listing above places DEBUG out of the way into ram area >6000 - which you need to provide! Either by modifying an Editor Assembler module by adding 8k ram or by use of a SUPER SPACE module or similar. Or omit lines 110 and 140, use only relocatable code, and hope there are no clashes... Line 110 resets FFAH (First Free Address in High Memory) to >6000 while line 140 changes it back to >A000.

OK... that switch! You need to connect a simple switch between to tracks on the right hand port - using a speech synthesiser is best as you do not then need to take the console apart. First, just a simple switch, which due to what is called switch bounce, may send multiple signals to the CPU causing a crash (a good switch will work OK around 60% of the time!) then a slightly improved circuit. Looking at the edge of the card, at the upper right of the console, pin 13 (LOAD) is the seventh pin from the left, on the bottom. This is one end of your switch. Inside the speech synthesiser you will find that on the SAME side there are four tracks connected- 11,12,13 and 14 from the left - this is the other end of your switch. For improved operation connect a 0.1mF by-pass capacitor across the switch and a 2.2k resistor in series with it.

```
                         CONSOLE TOP

C    2  4  6  8  10 12 14 16 18 20 22 24 26 28 30 -->  C
OF   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -       O
NR   ===========================================       NB
SO   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -       SA
ON   1  3  5  7  9  11 13 15 17 19 21 23 25 27 29      OC
LT                     |                |              LK
E                      |     \          |              E
                       |______\_________|
                              S

                        CONSOLE BOTTOM
```

```
Better version:            |              |
                           |       S      |
                           |  R__   \     |
                        ___ ~~~ ___ \_____
================                  |    |
RIGHT HAND PORT                   |    |
VIEWED FROM RIGHT                 | || |
================                --||--
                                  ||
                                   C
```

As ever, any hardware modifications are at YOUR risk !!

==

WordPerfect is an expensive word processor for some other computer, which has a Spell Checker. The manufacturer has sent out a newsletter with -on one page- the following interesting words:

IMNAGES SIMILARE DESIGNE

Word processors may have spell checkers but who uses them? Not many folk.
(New Scientist 6 Oct 90).

==

REMEMBER WHEN.... A year ago, back in issue 27 (page 20) I set a couple of puzzles. No response still to TEST 4 but here is a response to TEST 5 which - as you have forgotten it! -is to set up a circle of things and count them off removing every tenth one. If you start at position 1 and the first you remove is number ten, what are the numbers of the last two left?

```
100 ! JOHN SEAGER
110 OPTION BASE 1 :: DIM ARRAY(100)
120 CALL CLEAR :: INPUT "NUMBER OF ITEMS 3>100:":TOTAL
    :: IF (TOTAL>2)+(TOTAL<101)<>-2 THEN 120
130 FOR I=1 TO TOTAL :: ARRAY(I)=I :: NEXT I
140 OLDPOS,NEWPOS,MISS=1
150 ARRAY(NEWPOS)=ARRAY(OLDPOS)
160 MISS=MISS+1 :: NEWPOS=NEWPOS+1 :: OLDPOS=OLDPOS+1
170 IF OLDPOS>TOTAL THEN TOTAL=NEWPOS-1 :: OLDPOS,
    NEWPOS=1
180 IF TOTAL=2 THEN 210
190 IF MISS<10 THEN 150
200 OLDPOS=OLDPOS+1 :: MISS=1 :: GOTO 170
210 PRINT "":"REMAINING:":ARRAY(1),ARRAY(2)
220 END
```

==

REMEMBER 1983? Back when TIHOME was extinguished just BEFORE TI pulled the plug, leaving TI owners in the UK with NOTHING to fall back on? I was interested in reading a review of US computer magazines by Bill Gaskill to see that amidst the likes of COMPUTE!, ENTHUSIAST 99, 99ER MAGAZINE, and THE SMART PROGRAMMER, there was TIDINGS - apart from the rather glossy IUG magazine, the ONLY UG publication to be listed. How nice I thought. Now I have a copy of "FREE SOFTWARE FOR YOUR TI99/4A" published sometime in 1984 by ENRICH DIV/OHAUS in the USA.

This now very out of date book lists the IUG- which produced its last magazine in June 1984- as having 80,000 members (!!!!) -all TI owners - and gives the late Guy Stefan-Romanos San Francisco address as the IUG Library. Of much more interest to us is the entry for the UK, where the last issue of TIHOME was dated May

continued on page 22

TML Graphics programs

part 3, compiled by Stephen Shaw, England

Well yes, this IS a graphics program, but it is NOT a Fractal program, not even a chaotic program! (Shock, horror!).

What this program does is draw a face! The facial features are set by ten parameters.

The program is written in XB for use with The Missing Link but it can easily be modified for use with other graphics programs that allow pixel addressing. The image is small enough to be used with the TI BASIC listing given in an earlier issue.

Notice how much work the computer is doing to draw a smiling or frowning mouth!

```
90 RANDOMIZE
99 Z=150
100 ! PICKOVER B3 P327 FACES
110 RANDOMIZE
120 FOR I=1 TO 10
130 P(I)=5+INT(RND*6)-INT(RND*6):: CALL LINK("PRINT",I*
    10+40,120,STR$(P(I)))
140 NEXT I
150 CALL HEAD(P(1))
160 CALL EYE(P(2),P(7),P(8))
170 CALL PUPIL(P(3),P(7))
180 CALL EYEBROW(P(4))
190 CALL NOSE(P(5))
200 CALL MOUTH(P(9),P(6),P(10))
210 CALL LINK("DUMP") ! to printer
220 FOR I=1 TO 500 :: NEXT I :: CALL LINK("CLEAR"):: RUN
400 SUB HEAD(P1)
401 Z=150
404 DEF CD(X)=COS(X/180*PI)
407 DEF SD(X)=SIN(X/180*PI)
410 EX,EY=0 :: R=30
420 IF P1>5 THEN EX=(P1-5)*2 ELSE IF P1<5 THEN EY=ABS((
    P1-5)*2)
430 FOR T=1 TO 360
440 X=(R+EX)*CD(T)+50
450 Y=(R+EY)*SD(T)+50
460 IF T=1 THEN CALL LINK("PIXEL",Z-Y,X)
470 IF T<>1 THEN CALL LINK("LINE",Z-OY,OX,Z-Y,X)
480 OX=X :: OY=Y
490 NEXT T
500 SUBEND
510 SUB EYE(P2,P7,P8)
511 Z=150
514 DEF CD(X)=COS(X/180*PI)
517 DEF SD(X)=SIN(X/180*PI)
520 EX,EY=0 :: R=5
530 IF P2>5 THEN EX=(P2-5)*2 ELSE IF P2<5 THEN EY=ABS((
    P2-5)*2)
540 P7=P7-5
550 P8=(P8-5)/2
560 FOR T=1 TO 360
570 X=(R+P8+EX)*CD(T)+40-P7
580 Y=(R+P8+EY)*SD(T)+60
590 IF T=1 THEN CALL LINK("PIXEL",Z-Y,X)
600 IF T<>1 THEN CALL LINK("LINE",Z-OY,OX,Z-Y,X)
610 OX=X :: OY=Y
620 NEXT T
630 FOR T=1 TO 360
640 X=(R+P8+EX)*CD(T)+60+P7
650 Y=(R+P8+EY)*SD(T)+60
660 IF T=1 THEN CALL LINK("PIXEL",Z-Y,X)
670 IF T<>1 THEN CALL LINK("LINE",Z-OY,OX,Z-Y,X)
680 OX=X :: OY=Y
685 NEXT T
690 SUBEND
700 SUB PUPIL(P3,P7)
701 Z=150
710 PS=P3/5 :: IF PS=0 THEN PS=.1
720 FOR K=PS TO 0 STEP -.2
730 CALL LINK("CIRCLE",Z-60,40-P7,K):: CALL LINK("CIRCLE
    ",Z-60,60+P7,K):: NEXT K
740 SUBEND
800 SUB EYEBROW(P4)
801 Z=150
810 Y1,Y2=70
820 Y1=(P4-5)+Y1
830 Y2=Y2-(P4-5)
840 CALL LINK("PIXEL",Z-Y1,35)
850 CALL LINK("LINE",Z-Y1,35,Z-Y2,45)
860 CALL LINK("PIXEL",Z-Y2,55)
870 CALL LINK("LINE",Z-Y2,55,Z-Y1,65)
880 SUBEND
900 SUB NOSE(P5)
901 Z=150
910 P5=(P5-5)/2
920 CALL LINK("PIXEL",Z-55,50)
930 CALL LINK("LINE",Z-55,50,Z-45-P5,46)
940 CALL LINK("LINE",Z-45-P5,46,Z-45-P5,54)
950 CALL LINK("LINE",Z-45-P5,54,Z-55,50)
960 SUBEND
1000 SUB MOUTH(P9,P6,P10)
1001 Z=150
1010 P9=P9-5
1020 X1=40-P9 :: Y1=35
1030 X2=60+P9 :: Y2=35
1040 X3=(X2-X1)/2+X1
1050 Y3=(P6-5)+35
1060 REM
1070 FOR K=1 TO 2
1080 IF K=2 THEN Y3=Y3+P10/2
1090 OX=X1 :: OY=Y1
1100 D=X1^2*(X2-X3)+X1*(X3^2-X2^2)+X2^2*X3-X3^2*X2
1110 A=(Y1*(X2-X3)+X1*(Y3-Y2)+Y2*X3-Y3*X2)/D
1120 BB=(X1^2*(Y2-Y3)+Y1*(X3^2-X2^2)+X2^2*Y3-X3^2*Y2)/D
1130 C=(X1^2*(X2*Y3-X3*Y2)+X1*(X3^2*Y2-X2^2*Y3)+Y1*(X2
     2*X3-X3^2*X2))/D
1140 REM
1150 FOR I=X1 TO X2
1160 X=I :: Y=A*I^2+BB*I+C :: CALL LINK("LINE",Z-OY,OX,
     Z-Y,X):: OX=X :: OY=Y :: NEXT I
1170 NEXT K
1180 FOR K=1 TO 400 :: NEXT K
1190 SUBEND
1200 END
```

o

continued from page 2

companies, along with Horizon, and others have given us additional memory. PRBASE, that took 22+ seconds to load from a conventional disk, loads in under two seconds from the Horizon ramdisk. Craig Miller has given us Gram Kracker as well as some exceptional software products. The old system that was limited to 48k can now be greatly expanded. Mine has 320k now and has room for much more. Clock cards, buffers and the like have given this system tremendous capability.

So what are we to gain from Geneve? Speed, of course, a bit more memory, more colours, better graphics are a few of the things. For me, I am not sure they are worth the money if I have to give up 4A/Talk, Fast Term, Funlwriter and the like. I know of only one member of our users group, which numbers about 95, who currently plans to purchase Geneve. I would probably buy a PC before I bought Geneve, Although I have absolutely no intention of doing that. I can do everything I want with my handy 99/4A, so why spend the funds on a PC when I will just have to start the software hunt all over at much higher prices, I might add. I would imagine that software unique to Geneve will be expensive also because of the limited market. I see the Geneve'ers as a small splinter group that will break with the current TI99/4A community and go off on their own. I wish them and Myarc success, but I honestly feel that Geneve has come too late for most of us. When we were looking for a replacement we did not recognize just how powerful and versatile the 4A was. Now we do and I think that most of us will hang on to them for a long time. C'est la vie, ma chere Geneve!

o

continued from page 10

Most of these I have mentioned are in the public domain, not even fairware. So, if you are tired of trying to zap the invading aliens, if the text adventure has brought you back to the starting point for the umpteenth time, why not try doing something intelligent for a change?

o

TI-Base Tutorial #10

by Martin Smoley, North Coast 99ers USA

I am reserving the copyright on this material, but I will allow the copying of this material by anyone under the following conditions. (1) It must be copied in its entirety with no changes. (2) If it is retyped, credit must be given to myself and the NorthCoast 99ers, as above. (3) The last major condition is that there may not be any profit directly involved in the copying or transfer of this material. In other words, Clubs can use it in their newsletters and you can give a copy to your friend as long as its free.

This tutorial (I hope) will give you some ideas on how to use the DATE or, D-type fields. Not too long ago I was under the impression that you could not do much with a Date type field. Of course I knew you could sort a date field, and find records through the date field, but I did not think past that point. Then I got into a situation where I needed to calculate the age of a ewe. That is right, I said a ewe, the mother of a lamb. I could not do a thing with D fields using MM, DD or YY, so I called Dennis for some help, and did I ever feel like a dummy. Dennis said you can extract a month by using MONTH, a day by using DAY and a year by using YEAR. "Ask the right question and you will get the right answer (I thought)." Not from Dennis, from TI-Base. Now I will try to explain it to you. NOTE: In the CFs included in this article the lines with numbers are actual CF or program lines and the lines starting with REC or 0000 are the product of the DISPLAY commands that immediately preceded them. If you are entering these CFs, enter only the lines that have line numbers, but do not enter the line numbers. The {nnn means refer to a specific line number.

A date field is used to store a date in the form "MM/DD/YY". "I am sure that this is old news for most of you." This form can be used in a Db field, or in a local variable. An example would be LOCAL BORN D 8 {005}. TIB then creates a local variable space 8 units in length under the name BORN and the designation D Type. D type enables TIB to perform Date operations on whatever data it finds stored in that field or variable. BORN is presently empty. REPLACE BORN WITH "02/12/43" {012} would store the date February 12, 1943 in BORN in it's proper form (MM/DD/YY). From this point there are many things that TIB can do with the data stored in BORN. If you enter LOCAL MO N 3 {007} and then REPLACE MO WITH MONTH(BORN) {015}, TIB would extract 02 from BORN and place a copy of it in MO. The command DAY(BORN) would extract 12 and YEAR(BORN) would extract 43. If you created AGE N 3 and the current date "06/14/89" was in a date field named CURDT, REPLACE AGE WITH YEAR(CURDT) - YEAR(BORN) {020} would place 46 in AGE. Unfortunately it does not work in the other direction. REPLACE YEAR(BORN) WITH "45" does not work (as far as I can tell). If you enter REPLACE BORN WITH "45" {023}, the 45 will be placed in the far left portion of BORN, which is the month area. DISPLAY BORN would then produce (45), without the parenthesis. Concatenation (¦) can be used to get the results you want as far as placing data into a date field {025}. You should notice that the second "/" was eliminated {024} to allow for the right most space in AGE {004} which has a length of 3.

```
001  CLEAR
002  CLOSE ALL
003  CLEAR LOCAL
004  LOCAL AGE N 3
005  LOCAL BORN D 8
006  LOCAL CURDT D 8
007  LOCAL MO N 3
008  LOCAL DY N 3
009  LOCAL YR N 3
010  LOCAL TEST N 6
011  DISPLAY BORN,CURDT,AGE
-------------------------------------
REC   BORN      CURDT     AGE
0000
-------------------------------------
012  REPLACE BORN WITH "02/12/43"
013  REPLACE CURDT WITH "06/18/89"
014  DISPLAY BORN,CURDT,AGE
-------------------------------------
REC   BORN      CURDT     AGE
0000  02/12/43  06/18/89
-------------------------------------
015  REPLACE MO WITH MONTH(BORN)
016  REPLACE DY WITH DAY(BORN)
017  REPLACE YR WITH YEAR(BORN)
018  DISPLAY MO,DY,YR,AGE
-------------------------------------
REC   MO   DY   YR   AGE
0000   2    12   43
-------------------------------------
019  REPLACE AGE WITH YEAR(CURDT);
020                   - YEAR(BORN)
021  DISPLAY BORN,CURDT,AGE,TEST
-------------------------------------
REC   BORN      CURDT     AGE  TEST
0000  02/12/43  06/18/89   46
-------------------------------------
022  REPLACE BORN WITH "45"
023  DISPLAY BORN,CURDT,AGE,TEST
-------------------------------------
REC   BORN      CURDT     AGE  TEST
0000  45        06/18/89   46
-------------------------------------
024  REPLACE CURDT WITH "00/00" ¦AGE
025  DISPLAY BORN,CURDT,AGE,TEST
-------------------------------------
REC   BORN      CURDT     AGE  TEST
0000  45        00/00 46   46
-------------------------------------
026  REPLACE BORN WITH "06/31/44"
027  REPLACE TEST WITH DAY(BORN);
                       * MONTH(BORN)
028  DISPLAY BORN,CURDT,AGE,TEST
-------------------------------------
REC   BORN      CURDT     AGE  TEST
0000  06/31/44  00/00 46   46     186
-------------------------------------
029  RETURN Copyright Martin A. Smoley
030  *                              1989
031  *                    TDT3/C
```

After you become familiar with their basic functions you can do some interesting things with Date type fields, but you must be careful to remember what you have put in a field and what its position is. For example, REPLACE DATE WITH "Ma/rt/in" will place the character string Ma/rt/in into DATE. You could store Ma/rt/in in a DB and retrieve it later. However, TIB will recognize these as Characters and will not allow you to do anything with them. If you could multiply Ma by 2.2, I cannot imagine what the result might be. But on the other hand, TIB seems to be able to recognize numbers which have been entered as characters and then place them into a date type field. I have attempted to show the different aspects of this theory in TDT5 and TDT6. Notice that DATEC is a character variable {006}. I then placed "12345.78" into DATEC as a character string {012}. The reason I did it in this manner was to make sure that TIB considered 12345.78 to be characters at this point, even though it looks like a number. I then transferred DATEC to DATE {013}. DATE is a D or Date type field {005}. I was then able to do any Date type function involving DATE, including multiplication of its parts {019}, or multiplication by a constant, or LITERAL, {020}. Then I went through the same steps using a numeric (N) type field. Notice in line # 24 that TMP {004} is a numeric field and that 77.77 has no quotes around it. I am attempting to guarantee that 77.77 is a number. Then I placed TMP into DATE {026} and performed the Date type functions on 77.77. The reason I added the zeros was to demonstrate what I said previously, that you need to remember the exact position of the data in a field if you expect to use it in this manner. TMP has a width of 8 {004}. This means it would fit right into a Date type field. When I added the "00", I moved the spacing to put "7." in the year portion of DATE. Check the " / / " spacing to see where the slashes are located. This means, when YR is multiplied by 100 {032}, TIB is multiplying 7 by 100.

If you work with this idea, understand it and are very careful, you could use the date type field to extract specific parts of a number. A very simple demonstration of this would be to place a dollar and cents type number into a date field with the cents portion in the year segment of the date field. You could then use REPLACE CENTS WITH YR(DATE) to extract the cents, if that is what you need. Then REPLACE dollars WITH dollars - CENTS would give you the whole dollar amount. I realize that in most cases this idea is to cumbersome to use, but if there is an instance when no other procedure will work, this idea just might do the trick.

SORT by INSCEBOT

I do not know if SORT is the name that will be used, but sort is what it does. I am currently testing this new sort program from those TI-Base guys and it looks great. It will sort TI-Base files, Fix file, Variable files and hopefully soon Basic display files. I have tried it on TIB files and it is real fast and easy to use. should be available soon as a separate disk for under $15.00 (I think). The fact that it works on many different file types will make it a very useful program.

```
001  CLEAR          I have pulled TDT5 and TDT6 together
002  CLOSE ALL      to save space.  I hope this does not
003  CLEAR LOCAL          confuse everyone.
004  LOCAL TMP N 8 2
005  LOCAL DATE D 8
006  LOCAL DATEC C 8
007  LOCAL MO N 3
008  LOCAL DY N 3
009  LOCAL YR N 3
010  LOCAL TEST N 12 2
011  **********************************
012  REPLACE DATEC WITH "12345.78"
013  REPLACE DATE WITH DATEC
014  DISPLAY DATE,TMP,DATEC
-------------------------------------
REC    DATE       TMP        DATEC
0000   12345.78              12345.78
-------------------------------------
015  REPLACE MO WITH MONTH(DATE)
016  REPLACE DY WITH DAY(DATE)
017  REPLACE YR WITH YEAR(DATE)
018  DISPLAY DATE,MO,DY,YR,TEST
-------------------------------------
REC    DATE       MO   DY   YR   TEST
0000   12345.78   12   45   78
-------------------------------------
019  REPLACE TEST WITH DY * YR
020  REPLACE TMP WITH YR * 2.2
021  DISPLAY DATE,TMP,TEST
-------------------------------------
REC    DATE       TMP        TEST
0000   12345.78    171.60        3510.00
-------------------------------------
022  *                              TDT5/C
023  **********************************
024  REPLACE TMP WITH 77.77
025  DISPLAY DATE,TMP,TEST
-------------------------------------
REC    DATE       TMP        TEST
0000   12345.78     77.77        3510.00
-------------------------------------
026  REPLACE DATE WITH "00" ¦ TMP
027  REPLACE MO WITH MONTH(DATE)
028  REPLACE DY WITH DAY(DATE)
029  REPLACE YR WITH YEAR(DATE)
030  DISPLAY DATE,MO,DY,YR,TMP
-------------------------------------
REC    DATE       MO   DY   YR   TMP
0000   00   77.     0    0    7     77.77
>>==>>   /  /    «==« Note Spacing
-------------------------------------
031  *
032  REPLACE TEST WITH YR * 100
033  DISPLAY DATE,YR,TEST
-------------------------------------
REC    DATE       YR   TEST
0000   00   77.     7         700.00
-------------------------------------
```

continued on page 6

continued from page 12

And then you faint and wake up back in the terminal room (does not the very name "terminal room" make you feel uneasy, like an omen of things to come?). The PC screen is a mess of jumbled characters, and the hacker comes over to take a look.

While he is examining the screen, take a good look at him. He certainly seems to be carrying an unusual number of keys. If you ask the hacker about them, he will mention his sideline in locksmithing, and casually show you the master key, which is what you need. However, you will have to bribe him for it first.

Leave the room by going south, then head west to the dreary kitchen. Open the fridge door, revealing a large bottle of classic coke and a carton. Inside the carton is some Chinese food, cold at the moment. Take both the coke and the carton.

Ok, now put the carton in the microwave, set the timer to 5 minutes, the cooking power to high, and turn the microwave on (of course, you remembered to close the door first, right?). The funnybones on the table are not really needed in the game, so you do not have to take them with you (in fact, you can eat them while you wait).

When the five minutes are up, take the carton (which now contains some volcanic Chinese food) from the oven and return to the terminal room. Give the food to the hacker, who will appreciate your gesture by wolfing it down like a starving shark, and give you the master key when you ask for it. All right, now you are ready for the serious stuff.

Leave the room and press the down elevator button. When it arrives, get in and take it to the lobby. While you are on your way down, open the panel in the wall and get the flashlight...some places are dark, and have worser things than grues lurking in the shadows.

Exit the elevator at the lobby level, then take the stairs down to the basement. Sure is a little creepy, wandering around a mostly-deserted building at this time of night, eh? And just think: it will get worse before it gets better! Now, is not that a cheery thought?

Once in the basement, go east to the Temporary Basement to pick up the gloves and crowbar. Wear the gloves now, so you will not forget later. Now, turn around and amble westward until you reach the Aero basement with the forklift. Get into the forklift and start it. Guess what? You go back east again, this time driving the forklift. Keep going until you reach Dead Storage. You will know you are there, because it is dark.

Quickly turn on the flashlight. What a mess! Dead Storage sure was a good name for this place... there is junk everywhere, piled high up, almost to the ceiling. You would never be able to move this stuff by hand. Good thing you have the forklift to help you. So, move the junk with the forklift, until the passage shows.

Now you can turn off the forklift and exit it (the passage is not big enough for it, and you do not need it anymore). This takes you to an even grubbier room, Ancient Storage, where the junk seems to have coalesced into absolute unrecognisability. Luckily, you do not have to mess with the mess.

Turn your attentions to the manhole cover in the floor (wonder what that is doing here?). With the crowbar, you can pry it up. Grunt! The dark opening may not look inviting, but it is necessary to go down there.

You find yourself in a brick tunnel, with two ways to go. I suggest north, to the Renovated Cave, and then down to what looks a lot like an altar, with some suspicious stains on it. There is a plate in the floor, which you do NOT want to move, now or ever. Just leave it alone, and grab the knife that someone carelessly left lying around.

continued on page 4

Speech

by E.P. Rebel, Netherlands

Program Purpose:

SPEECH V1.0 is a program to test the contents of the ROMs in your speech synthesizer. It will speak every word contained in the ROMs.

Using the Program:

Load and Run the program from MiniMemory or Editor Assembler module. The start name of it will be displayed automatically so you have only press <ENTER> twice to start the program. All the words SPEECH V1.0 finds, will be displayed on the screen and spoken by the synthesizer. Press <QUIT> to end the program.

About the Source:

I am sorry but the comments to the source are in Dutch. This program was written for the Dutch Users Group Newsletter. Feel free to change the source and make the program better.

Public Domain, Freeware or What-soever.

This program may be duplicated in any form without notice of the author. You may distribute it via your users group or give it away to your friends. But please pass along the source and this documentation too. If you like the program do not send me $10.00 (although I would not mind) but send me a program of your own that I can distribute via the Dutch users group. Thank you.

The Author's Name and Address:

SPEECH V1.0 and this documentation were written by:

Eric-Paul Rebel
Merelstraat 27
1223 NR HILVERSUM
The Netherlands
Phone: 31-35832929

Do not ring me up when you do not speak Dutch and live in another continent because the time in Hilversum is different and my English, Spanish and Japanese are bad! My apologies for the bad English but I suppose you prefer it over the Dutch version.

```
**************
*   SPEECH   *
**************
* E.P. REBEL *
* 05-03-1987 *
**************
       DEF  SPEECH
       REF  VMBW,VMBR
       REF  SPCHRD,SPCHWT

       DORG >8300
WORKSP BSS  >0020       workspace
READIT BSS  >000C       name for start of program code
SPEAK  EQU  $

       RORG
SPEECH LWPI WORKSP      use fast registers
       BL   @INIT       re-locate code to fast memory
LOOP   CLR  R15         stack pointer
       CLR  R4          speech ROM addres >00000
       BL   @LOAD       load speech ROM address
       BL   @GETBYT     read first byte from speech ROM
       CB   @HAA,R3     is valid ROM present?
       JNE  ERROR       no, stop
       BL   @SUB        read out binary
       BL   @WAIT       wait a while
       JMP  LOOP        keep going

ERROR  BLWP @0          error, reset.

SUB    MOV  R11,@STACK(R15) save return address on stack
       INCT R15         increment stack pointer
       MOV  R15,R2      determine string size
       SLA  R2,4        times 16
       AI   R2,NAMES
       BL   @GETBYT     get byte from speech ROM
       MOVB R3,*R2+     this is the length of word
       JEQ  NAMEOK      length 0 should not occur
       MOV  R3,R4
       SRL  R4,8        length in lsb
NAMGET BL   @GETBYT     get byte from speech ROM
       MOVB R3,*R2+     form word or phrase
       DEC  R4          finished all bytes?
       JNE  NAMGET
NAMEOK BL   @GETBYT     get byte from speech ROM
       MOVB R3,@LOW(R15) msb of less than pointer
       BL   @GETBYT     get byte from speech ROM
       MOVB R3,@LOW+1(R15) lsb of less than pointer
       BL   @GETBYT     get byte from speech ROM
       MOVB R3,@HIGH(R15) msb of greater than pointer
       BL   @GETBYT     get byte from speech ROM
       MOVB R3,@HIGH+1(R15) lsb of greater than pointer
       BL   @GETBYT     get byte from speech ROM
       JNE  ERROR       nibble 0 must be 0 (ROM is 8k)
       BL   @GETBYT     get byte from speech ROM
       MOVB R3,@ADDR(R15) nibble 1, 2 of speech data
       BL   @GETBYT     get byte from speech ROM
       MOVB R3,@ADDR+1(R15) nibble 3, 4 of speech data
       MOV  @LOW(R15),R4 less than current pointer?
       JEQ  NOLOW
       BL   @LOAD       load new speech address
       BL   @SUB        scan left branch of binary tree
NOLOW  BL   @SCROLL     scroll screen
       LI   R0,23*32+2  screen position of lowest line
       MOV  R15,R1      determine string room
       SLA  R1,4        times 16
       AI   R1,NAMES
       MOVB *R1+,R2     get length
       JEQ  PRTOK       should never be 0
       SRL  R2,8        length in lsb
       BLWP @VMBW       put word on screen
PRTOK  MOV  @ADDR(R15),R4 address of speech data
       BL   @LOAD       load address
       BL   @SPEAK      pronounce word/phrase
       MOV  @HIGH(R15),R4 greater than current pointer?
       JEQ  NOHIGH
       BL   @LOAD       load new search address
       BL   @SUB        scan right branch of binary tree
NOHIGH DECT R15         decrease stack pointer
       MOV  @STACK(R15),R11 restore return address
       RT

INIT   LI   R0,READIT   start of replacement code
       LI   R1,CODE     replacement program code
       LI   R2,CLEN     length of code
CODELP MOV  *R1+,*R0+   move code
       DECT R2          finished?
       JNE  CODELP
       RT

GETBYT MOV  R11,R10     store return address
       MOVB @H10,@SPCHWT read instruction byte from ROM
       BL   @DLY12      wait
       BL   @READIT     read byte
       B    *R10

LOAD   LI   R5,5        five nibbles to be loaded
LOADLP MOV  R4,R0       address
       SLA  R0,12       remove excess nibbles
       SRL  R0,4        put nibble in correct place
       ORI  R0,>4000    turn into correct instruction
       MOVB R0,@SPCHWT  load instruction
       SRL  R4,4        next nibble
       DEC  R5          finished all five?
       JNE  LOADLP
       MOV  R11,R10     restore return address
       BL   @DLY42      wait a bit
       B    *R10

DLY42  LI   R6,10       long delay
DELAY  DEC  R6
       JNE  DELAY
       RT

DLY12  NOP              short delay
       NOP
       RT
```

continued on page 3

Labels with TI-Writer

by Lou Amadio

Not long ago I was talking to Rolf about various software programs available for the TI99/4A when he mentioned that he needed a flexible label producing program that could also incorporate graphics. Programs are available that can mix text and graphics on labels, but, generally speaking, the ability to produce out-of-the-ordinary text is limited, if not difficult.

It occured to me that perhaps TI-Writer could be used to produce labels, by utilizing the multifont capabilities found on most modern dot matrix printers. Of course this does not solve the problem of including graphics, but this may be solved at a later date. After all, TI-Writer can produce graphics now through special DV80 files.

Before embarking on how to produce the actual labels, it would probably be worthwhile to reflect back on how to control the various print functions of your printer. Although quite a lot of features are now available via the front control panel of modern printers, for the purpose of this exercise, we will use the built in commands and invoke them via control codes.

Printer control codes may seem a little intimidating at first, but you will find them fairly easy to understand once you have used them once or twice.

Most control codes are initiated by sending the ESCAPE <ESC> character (ASCII 27) followed by one or more conventional characters as appropriate. It might be worthwhile if you spend about half an hour with your printer manual and read the section on Printer Control Commands. In any case, if you follow the instructions given in this article, or get a copy of the appropriate DV80 files from the club library, you will be making your own labels in no time.

Special Character Mode

Special Character Mode is used to generate the characters needed to control the printer options.

In TI-Writer, special characters are introduced into the text by first pressing CTRL[U] (ie press and hold the CTRL key while you press the "U" key). You will notice that the cursor changes to a thin underline shape. If you were now to press any of the other keys on the keyboard, you will notice that what you press is not what you get on the screen! The reason for this is that by first pressing CTRL[U], what you do, in fact, is shift the characters generated by the keyboard down the ASCII table to incorporate the normally inaccessible codes below 31. So, to operate the keyboard effectively in the CTRL[U] mode, we need a table to translate our key presses so that we can make sense of them. You will find this table on page 146 of the TI-Writer manual. The table shows the first 31 ASCII codes, their abbreviation, their function, which key to press and what appears on the screen. The most important character in this table is 27, the <ESC> character. This is generated by pressing FCTN[R] (ie press and hold the FCTN key while you press "R") when in Special Character Mode.

Epson compatible printers use the <ESC> character, in combination with other characters, to "switch on" the various features of your printer. On the screen, <ESC> appears as a small super script 1 followed by a small subscript b. In fact 1b is 27 in hexadecimal, but we need not worry too much about this here.

Once the Special Character is generated, you return to normal edit mode by pressing CTRL[U] again. Immediately following the <ESC> character we now generate a "normal" character from the keyboard corresponding to the required printer function that we want to invoke. For example, if we wanted to print double height characters in our label, the characters immediately following the <ESC> would be "w1". Conversely, <ESC> followed by "w0", would return to normal printed characters.

TI-Writer Label Print Files

I have created twelve DV80 files which will allow you to print out to two different fan-feed label sizes: 25 mm and 35 mm. Read the short documentation file supplied on disk for information on the number of characters and lines which will fit on each label. The files are designed to utilize the fonts built into the Star NX1000 printer, but could easily be modified to use the special features of your printer. Some files (eg compressed and expanded print) will run on any Epson compatible printer without modification.

Because of the different sizes associated with non-standard fonts, I had to fiddle with the line spacing in some of the files to ensure that the printout aligns accurately on successive labels. The font size also affects the number of characters that will fit on each horizontal line. Consequently, compressed print labels will appear to occupy a much larger area on the screen when compared with expanded print labels.

The best way to find out the limits of each font and label combination is to experiment. First of all try your output on ordinary paper before committing to labels. This will allow you to check for horizontal and vertical alignment as well as text to label centering.

A typical Label Print File is shown below. As control characters cannot be printed out, I have surrounded the control codes with angle brackets. For example, <ESC> is used for "ESCAPE", <18> for the character whose ASCII code is 18 decimal and <1> for the character whose ASCII code is 1.

```
01 <ESC>@FILE: 2HILABELS
02 This file will print out 35 x 100 mm
03 labels allowing up to 4 lines of
04 double height text with up to 36
05 characters per line.
06
07 Print File "13 LL PIO"
08
09 where LL is the Last Line to print.
10
11 Double Height/Width, 4 Lines/Label.
12 Reset/4lpi/2HI/Sans/NLQ/Dstrike
13
<ESC>@<ESC>A<18><ESC>w<1><ESC>k<1><ESC>x<1><ESC>G.......
14 .                                    .
15 .                                    .
16 .           THIS IS LABEL 1          .
17 .                                    .
18 ....................................
19 ....................................
20 .                                    .
21 .                                    .
22 .           THIS IS LABEL 2          .
23 .                                    .
24 ....................................
25 ....................................
26 .                                    .
27 .                                    .
28 .           THIS IS LABEL 3          .
29 .                                    .
30 ....................................
31 <ESC>@
```

Line 1 resets the printer to the switch-on defaults (<ESC>@).

Lines 2 to 5 give a short description of the print and label size supported.

Lines 7 to 9 remind you to start the printout from line 13 (the first line of the first label) and to include the last line of the last label (LL) in the Print File command.

continued on page 3

Multiplan Exercises #4

by Herbert Schlesinger, USA

COPYING FORMULAS:

The copy option from the main menu allows us to copy labels, numbers and formulas. Mostly we use it for formulas since the others do not often repeat themselves. Here is how it is done:

a. Place the cell pointer in R3C4 (the cell with the formula we want to copy).
b. Select COPY from the main menu, and you get:

COPY: Right Down From

c. Select Down. Then you have these options:

COPY DOWN number of cells: starting at: R3C4

d. Type 4 (to copy downward four rows).
e. Press <ENTER>.

Now the commission for each row will be in the proper cell; but if all you got was the same figure down the entire column, you had previously asked the OPTIONS option not to recalculate. In that case, press the RECALC key (FCTN 8) and the proper amounts are entered in the cells. You can also see that the formulas are in the proper cells by printing out the sheet using the Formulas option in the Options part of the Print option of the main menu. (See display of formulas, page 14).

Formulas using the RC[-n] notation are called relative cell reference. This could also be used as R[-n]C. What we are doing is basing the formula on a cell and referring back to that cell to make it work.

When we use the standard formula RxCx*RxCx we are using the absolute cell reference. There is no way these formulas can be used for other rows or columns.

Now let us use some of this knowledge; bring up the file "TENYEAR". Remember how?

From the main menu-(T)ransfer ; (L)oad ; type "TENYEAR"; <ENTER> and there it is. Remember, if you do not know the file name, press an arrow key after the "L", the screen reminds you that you can do this, and move the highlight to the desired filename and press <ENTER>.

First, notice that the top line is continuous across the screen. This is done by placing the cell pointer in the home (R1C1) position; select FORMAT; then CELLS: type :R1C8 as the cells to format; press CTRL A twice to get to the format code section; select Cont; and press <ENTER>. Now select Alpha; type in the label and press return. The other labels are typed in at their proper places: The sheet looks like this:

```
#1    1      2      3      4      5      6

 1 Ten Year Projection for Commercial Real Estate
 2
 3 Tenant       Rate
```

Place the pointer in R3C3; press the = key and enter the formula;

1984+COLUMN()-3

Press <ENTER> and R3C3 should show 1984. Select Copy from the main menu; select Right; type in 10 for the number of cells; Press <ENTER>. The years 1984 to 1994 will appear in the columns three thru thirteen. You can check this by scrolling over to the hidden columns. Here is what happened: in R3CR we placed the formula 1984+COLUMN()-3. COLUMN() returns the column number the cell is in so it returned three. So then we had 1984+3-3. As we copied to the right we had 1984+4-3 or 1985, then 1984+5-3 or 1986 etc.

Data to enter:

```
#1    1      2        3        4 etc

 1 Ten Year etc
 2
 3 Tenant    Rate     1984     1985     etc.
 4
 5 ABC Co.   0.15     12000
 6 Zepco     0.17     10000
 7 B-Tree    0.16      9500
 8 Byte Co.  0.17     10000
```

The data in column 1 are labels; column 2 is the annual rate of increase and the third column is the starting yearly rental for each company.

Using the copy command we can enter the rents for each succeeding year by using this formula:

this year = last year + (rate * last year)

To enter the formula properly, do this:

1. Position the cell pointer in R5C4
2. Press the = key.
3. Press the left arrow key so the menu displays RC[-1].
4. Type + so that the formula reads RC[-1]+ and the cell pointer jumps back to R5C4.
5. Type R5C2* (this is the increase rate. We must do this by absolute reference not by moving the cell pointer which would be by relative reference.) The formula now should read RC[-1]+R5C2*.
6. Press the left arrow key, now we have RC[-1]+R5C2*RC[-1].
7. Press <ENTER>.

The final formula, RC[-1]+R5C2*RC[-1] means this cell contains the column to the left of this one (C[-1]) PLUS R5C2 (the increase rate) times the column to the left of this one. OR Last year plus the increase times last year.

This will show on your screen in R5C4. now we must copy this for the ensuing 9 years: Place the pointer on cell R5C4; select copy from the main menu; select Right from the Copy menu; type in 9 as the number for cells for the next prompt; and press <ENTER>. You will see the calculation for all of the years enter automatically. Scrolling to the right will display this to you.

We must do the same thing for each of the other companies, but we can not merely copy this using the Copy option because we used an ABSOLUTE reference in our formula. We must change the formula for the other tenants by using R6C2, R7C2 AND R8C2 in place of the R5C2 we used for the first tenant. You can see that if we did not, the rate of increase would be the same (would always refer to the R5C2 cell) instead of the correct amount as indicated in the proper cells for each tenant.

In a case like this it is unnecessary to have the dollar signs and the two decimal places found in the money mode. Format the cells in the range R5C5:R8C13 into Integer Format, using the Format and Cells commands. This whole area is really dollar amounts, but that would clutter up our chart, and are readily understood without the extra symbols. We can embed commas into the amounts which helps make them clear. Do this by selecting the Format Options command (select Yes for the commas option).

Now lets pretty up the chart:

1. Place the cell pointer at R9C3
2. Select FORMAT; then CELLS; type in the colon (:) and press Fctn 1 (the END Key). The cell pointer will move to the 1994 column and the range will show R5C3:R8C13.
3. Press CTRL A twice to get to the Format Code menu; type C for "continuous" ; press <ENTER>.

4. With the pointer still in R9C3; press = to enter a formula; type in the formula REPT("-",110) ; and press <ENTER>.

Here is what we did: We made Row 9 continuous; then we told Multiplan to repeat the hyphen (-) 110 times across the sheet.

Next type "Total" into R10C1. Calculate the totals thusly:

1. Place the cell pointer in R10C3, in the 1984 column.
2. Press = to enter a formula.
3. Type in the formula SUM(
4. Press the Up-arrow key 5 times so the formula is expanded to SUM(R[-5]C.
5. Press " : " so that the colon is added to the formula and the pointer is back at R10R3.
6. Press the Up-arrow key twice so that the formula now reads SUM(R[-5]C:R[-2]C. Add a closing parenthesis and press <ENTER>. Formula now looks like: SUM(R[-5]C:R[-2]C).

The sum of the column appears in R10C3. Place the cell pointer in that cell; select the Copy option from the main menu; select Right from the copy submenu; enter 10 for the number of "cells to copy to" and press <ENTER>. If you originally elected to HAVE the automatic recalc the proper sums appear across the sheet. If you DID option (NO) for automatic recalc the sums shown across the sheet are the same as the 1984 column. Press FCTN 8 and these figures will change to the proper amounts. WHY? Because we used Relative References in our formula. At this point the sums may be made into integer figures as above, or even this row only into dollar figures.

If you now want to change one or more of the figures you can "what if" all over the place and the figures will change to reflect your entries, either automatically or by the use of FCTN 8 (recalc).

To save this sheet in its present form use Transfer and then Save. If you want to save it under the same name you will notice the program enters that for you. But without a change in the name it asks:" Overwrite existing file? ". If this is what you wish press Y (Yes) and this file will replace the previous one with that name. I suggest that a different name be used so that if necessary you could pass this disk on to some one else and they would be able to build the file as you have done. Here we saved this finished sheet as "TENYEAR1" to be used later.

Recap: We have covered a lot of material in this section of the program. A) We "drew" ranges by moving the cell pointer in developing formulas. B) We copied formulas using the Copy command. C) We used the COLUMN() and the REPT commands and also used the Cont subcommand to use a row continuously instead of labouriously cell by cell. D) We learned a little more about Relative and Absolute references. o

Treasurer's Report

by Geoff Trott

My report is short and sweet this month. Renewals are coming in at this time of the year and so we are "girding our loins" ready for the next year. There is a useful amount of money for our running expenses for the coming year. Thank you to all who have given us a vote of confidence by renewing their membership. We are trying to please you all but do not be too harsh on our short comings. We only have a limited amount of time to devote to the running of the club. I will not be able to attend the June meeting as I have a grand final in our tennis competition on that day and my team requires my presence.

Income for April	$2614.32
Payments in April	$1023.19
Excess of income over expenses for April	$1591.13 o

A Look at Assembler

Entry and Exit: by Art Green, Ottawa Users Group

This month we look at program entrance and exit code. The standard Editor Assembler Option 5 program (called memory image) is loaded then invoked (called) via a BL instruction with the GPL workspace. The standard procedure is to save the return address, the workspace pointer and the GROM address then on program exit to restore these and also set the COND bit of the status byte to zero. The following code will accomplish this.

```
*
* Entrance Code
*
* Called by BL linkage.
* Saves: Return address in R11,
*        Workspace Pointer,
*        Interrupt Mask,
*        GROM Address.
*
       BLWP  $+4             Save WSP and STATUS
       DATA  ENTWSP-18,$+2   Note short workspace
       MOV   @22(R13),R14    Save Caller's R11
* Note: Our workspace now looks as if we were
*       called via BLWP.
       MOVB  *R9,R12         Read GROM address
       SWPB  R12
       MOVB  *R9,R12
       SWPB  R12
       DEC   R12
       B     @BEGIN          Go to start of pgm
*
* Exit Code
*
* Called by B from anywhere in the program
* Restores all values and zeros STATUS
*
EXIT   LWPI  ENTWSP-18       Our short workspace
       MOVB  R12,*R10        Restore GROM Address
       SWPB  R12
       MOVB  R12,*R10
       MOVB  R11,@>837C      Zero STATUS
       RTWP                  Return to OS
ENTWSP EVEN                  Short Workspace
       DATA  >9800,>9C02     R9, R10 (GROM stuff)
       DATA  0               R11 (a constant)
       BSS   2               R12, Saved GROM address
       BSS   2               R13, Old WSP
       BSS   2               R14, Old R11
       BSS   2               R15, Old status
*
* Real start of main program
*
BEGIN  LWPI  ....
       ....
       B     @EXIT           Exit main program
```

This Entrance/Exit code will only be executed once per program so that it should be optimised for space rather than time. No cost analysis of the code has been done, but I claim it to be the minimum required. Can you prove me wrong?

This code is useful in almost every program you are likely to write, but it hardly seems worthy of making it into a macro since it will be used only once per program. It is, however, a good candidate for one of the other Assembler Programmer tools, the COPY statement. Your main program then might look as shown below.

```
       DEF   MYPGM           Define pgm name
MYPGM  EVEN                  Program entry point
       COPY  'DSK*.BEGINEXIT' Standard code
BEGIN  LWPI  MYWSP           Real start of pgm
       ...
       B     @EXIT           Terminate pgm
```

The TI-Writer Option 3 loader uses the same conventions as Editor Assembler Option 5 except that TI-Writer disables the screen before invoking the program. Next time we will look at string searching. o

Beginning Forth - part 6

by Earl Raguse, UGOC, CA USA

MUSIC

Having finished with loops in Beginning Forth #5, I will now take up the subject of Forth Music. This is not per the original schedule, but there is no reason to delay, we have covered most of the basic principles.

One day, after being totally frustrated by Forth having no sound generating words, I found a set of screens defining the word SOUND, published in the August 1984 Smart Programmer and written by Rex Nielson. (Forth does have BEEP and HONK, but only if graphics are loaded). The way the word SOUND works, I judge, is that Rex is using the "direct access" alternative described in the E/A manual. Rex's words may be used just like any other Forth words and they have a direct correlation with the BASIC CALL SOUND command. I have included herein, Rex's two screens (#34 and #35) which include instructions for use.

I have also included Screen #36 with some words I created to make it easier to play musical notes. Screen #36 was written because I was lazy. I do not like to re-enter information which has not changed. Let us see if I can explain what I did. The syntax for the word SOUND is:

0 Vol Noise Vol Freq Dur SOUND

You may include up to 3 Vol Freq sequences, and noise is optional. Each element is a number which must be on the stack at SOUND execution time. You may execute SOUND from a Forth colon definition word or in the immediate mode as you wish. You may, as in BASIC, have only a single note or the maximum allowable in each SOUND execution, but each time it is executed, the required values must be on the stack.

I decided that there was no sense in having to enter the note length every time, because in some songs it is the same all the way through, so I defined the word TEM to hold the tempo, from which I could compute a whole note length (which I store in the variable WNL) and hence any shorter length note, which I store in the variable LENG.

I thought only one VOLume was needed and if I did not change it, why should I enter it for each note, so I defined variable VOL. Then, since I planned to define only one Octave of musical notes, (see Screen #37), I defined the variable OCT to hold the Octave number of interest, because this does not ordinarily change much in most musical pieces. I also defined variables NV to hold Noise Volume, N2 and N3 to hold notes 2 and 3.

Then I defined the words V and L to store values in VOL and LENG. TEMPO stores the input value in TEM and computes the value of a whole note length and stores it in WNL. The word RN (Read Note) expects a frequency on the stack, it fetches the VOLume, SWAPs it with frequency, which it in turn multiplies by the contents of OCTave to get the desired Frequency for SOUND. Thus it leaves VOLume and Frequency on the stack for use by SOUND.

The words RN2 and RN3, fetch the values stored in N2 and N3, then execute RN, as above.

Now we come to PN1 (Play 1 Note), this requires one note frequency on the stack. SOUND requires also that a zero be on the stack, (see Screen #35 lines 11 and 12), PN1 supplies the zero, which it SWAPs with Freq, then executes RN and fetches LENG, before executing SOUND. Really quite straight forward.

Now that you follow PN1, PN4 is the same, except that it expects 3 frequencies on the stack, two frequencies to be stored in N3 and N2, VN must also hold a noise volume, although the default value of zero works, but may be too loud, PN2 and PN3 work in a similar fashion. You could be argue that I have made unwarranted use of variables N2 and N3, and you may be right, but it sure simplifies the word PN4.

Next we have Screen #37, which is written to define frequencies in terms of the musical notes, so we do not have to do a frequency translation each time we want to play a standard musical note instead of just a sound. It also defines note lengths in terms of the above computed whole note length. Here also are the Octave control words 01 through 07 which store multiplier values in OCTave. I hate to admit this, but when I first did this, I erroneously put the octave numbers in OCT instead of the powers of 2 as it now is. It took me quite a while to figure out what was wrong with my music.

A natural note is designated by @, sharp by # and flat by $. A rest is denoted by @R. The reason for using the @ symbol in front of all the natural notes is to avoid possible conflict with single letters A-G and R which might be used elsewhere as variables. A note is defined as a frequency in the first Octave below middle C. The fourth octave word 04, for example, just multiplies the frequency by 16, the 4th power of 2.

The note length words are SI thru WH. and are simple abbreviations of the words they stand for, SIxteenth, EIghth, QUarter, HalF, and WHole, a dot increases the note length by 50%.

The Octave number (Ox) must be stated before playing a note, see music example on Screen #42, or the default will be 0, an error. Maybe I should have made this a 1, but did not. You may change that if you wish.

I have included an example using PN1, called CHARGE on Screen #46. You may find CHARGE useful as an attention getter in some of your other Forth programs. It also illustrates how simple minded this can be in Forth.

Because PNn did not turn out to be all I had hoped for when it came to coding music with chords, more than one VOLume may be required, for example; I wrote Screen #47. The only significantly new words here are AV (All Volumes) and AN (All Notes), they read values from the stack and store them in the appropriate variables for later use. TN1-TN3 and GN1-GN3 perform similarly to the RN's from before. The words .NT and .V were just devices for finding out what was stored in the variables when I was confused during the design of CHD which replaces PN1 through PN4. Notice how IT is re-defined at the end of Screen #47, so that FORGETing IT does not dump the SOUND capabilities. (See the discussion of IT in Beginning Forth #5)

Screens #42 and #43 are an example which plays Just A Song At Twilight. The first thing that happens here is to load the SOUND and PN capability, if its not already loaded, with 34 CLOAD CHARGE. If you do not understand this look up CLOAD in the TI Forth Manual. They explain it as well as I could.

Next we FORGET IT to clear any previous application which may be loaded in memory. You must have previously defined : IT ; (see Screen #47) for this to work (See Beginning Forth #5). Notice next how OCTive and TEMPO are entered as the first step. Then the volume level of N1 is set at 0, all the rest are set at 30, with the word AV, since all we are dealing with here is the melody, one note at a time.

If this article was not already too long, I would include an example using CHD, it makes much better music, but is better left until next time.

CU next time, May the FORTH be with you.

SCR #34

```
 0 (Sound Routine - 3 Notes, 1 Noise, Vol 1-16 by Rex
   Nielson)
```

```
 1  FORGET IT  BASE->R HEX
 2  80 VARIABLE OPER   0 VARIABLE TIME
 3  3001 VARIABLE SDTAB  0 VARIABLE NOISY 2 ALLOT
 4  049F VARIABLE SDOFF  4 ALLOT BFDF SDOFF 2+ ! FF00
    SDOFF 2+ 2+ !
 5  : OTEST BEGIN 83CE @ 0 = UNTIL ;
 6  : +SDTAB 1 SDTAB +! ;  : +OPER 10 OPER +! ;
 7  : RSET  80 OPER ! 3001 SDTAB ! 0 NOISY ! ;
 8  : DUR    ABS 4 SRL DUP 1 < IF DROP 1 ENDIF TIME ! ;
 9  : TONE1  0 3000 VSBW 1B4F5. ROT U/ DUP F AND OPER
                                                 @ +
10          SDTAB @ VSBW +OPER +SDTAB 4 SRA 3F AND
                                      SDTAB @ VSBW
11          +SDTAB DROP OPER @ + SDTAB @ VSBW +SDTAB
                                                +OPER ;
12  : TONE2  TIME @ SDTAB @ VSBW SDTAB @ 3001 - 3000
                                              VSBW +SDTAB
13          SDOFF SDTAB @ 6 VMBW 3000 83CC ! 83FD DUP
                                                 @ 01 OR
14          SWAP ! 0100 83CE ! ;
15    R->BASE  -->
```

SCR #35

```
 0  (Sound Routines cont Rex Nielson)  BASE->R HEX
 1  : NOISE ABS 1 - 7 AND E0 + NOISY ! F0 + NOISY
                                          2 + ! ;
 2  : TONE3 NOISY @ 0 > IF 3 0 DO NOISY I + @ SDTAB
                                             @ VSBW
 3          +SDTAB  2 + LOOP ENDIF ;
 4  : -TEST DUP FFF7 U< IF TONE1 ELSE NOISE ENDIF ;
 5  : SOUND DUP 0 > IF OTEST ENDIF DUR BEGIN -TEST DUP
                                          0 = UNTIL
 6      TONE3 DROP TONE2 RSET ; R->BASE
 7  (    Freq's F1 F2 F3  are like BASIC                )
 8  (    Vol's  VN V1 V2 V3 are from 1 through 16       )
 9  (    Dur  is also like BASIC - Positive and Negative)
10  (    You can use from 1 to 3 notes and 1 noise.     )
11  ( note: First item on the stack must be a 0, zero,
                                             or the )
12  (    empty stack message will come up.              )
13  ( Syntax    0 VN  N  V3  F3  V2  F2  V1  F1  Dur
                                             SOUND    )
14  ( Example   0  4 -3   2 1500  2 660   3 880  900
                                             SOUND    )
15   -->
```

SCR #36

```
 0  ( PN PlayNote E G Raguse 4 4 87)
 1  120 VARIABLE TEM  0 VARIABLE OCT    0 VARIABLE WNL
 2    0 VARIABLE VOL  0 VARIABLE LENG   0 VARIABLE NV
 3    0 VARIABLE N2   0 VARIABLE N3
 4  : V VOL ! ;  : L LENG ! ;
 5  : TEMPO TEM ! 120 1000 TEM @ */ WNL ! ;
 6  : RN VOL @ SWAP OCT @ * ;
 7  : RN2 N3 @ RN ;
 8  : RN3 N2 @ RN ;
 9  : PN1 ( 1note ) 0 SWAP RN LENG @ SOUND ;
10  : PN2 ( 2notes) N2 ! 0 SWAP RN RN2 LENG @ SOUND ;
11  : PN3 ( 3notes) N3 ! N2 ! 0 SWAP RN RN2 RN3 LENG @
                                             SOUND ;
12  : PN4 ( 3notes & noise) N3 ! N2 ! 0 SWAP
13        RN RN2 RN3  NV @ -4  LENG @ SOUND ;
14  -->
15
```

SCR #37

```
 0  ( MUSICAL NOTES FIRST OCTAVE BELOW MIDDLE C 262 EGR
                                              4 87)
 1  : @C 131 ;  : SI  WNL @ 16 / L ;          : 01  1
                                              OCT ! ;
 2  : #C 139 ;  : SI. WNL @ 16 / DUP 2 / + L ; : 02  2
                                              OCT ! ;
 3  : @D 147 ;  : EI  WNL @  8 / L ;          : 03  4
                                              OCT ! ;
 4  : #D 156 ;  : EI. WNL @  8 / DUP 2 / + L ; : 04  8
                                              OCT ! ;
 5  : @E 165 ;  : QU  WNL @  4 / L ;          : 05 16
                                              OCT ! ;
 6  : @F 175 ;  : QU. WNL @  4 / DUP 2 / + L ; : 06 32
                                              OCT ! ;
 7  : #F 185 ;  : HF  WNL @  2 / L ;          : 07 64
                                              OCT ! ;
 8  : @G 196 ;  : HF. WNL @ 2 / DUP 2 / + L ;
 9  : #G 298 ;  : WH  WNL @      L ;
10  : @A 220 ;  : WH. WNL @      DUP 2 / + L ;
11  : #A 233 ;  : @R 30000 ;
12  : @B 247 ;  : $D #C ;
13  : $A #G  ;  : $E #D ;
14  : $B #A  ;  : $G #F ;
15    46 LOAD
```

continued from page 13

1983! There is an interview with Paul Dicks, indicating a membership of 2000. The article makes remarkable reading for those of us in from the start and is quite interesting history for you newer owners!

"Although there are now twenty to thirty thousand TI owners in the British Isles, Paul's group is virtually the only resource that TI owners have in England.

"TI users face several obstacles, Paul explains. Although the price of the TI home computer is reasonable, the government encourages the BBC Acorn and will not give schools subsidies to purchase TI 99/4 computers. Also, the price of peripherals is almost prohibitive-we are really working on trying to improve this situation. At this time, there are probably only 12 disk drive owners in the entire British Isles! [me-ss here- I got my disk drive in Autumn 1981]. Third party software is just beginning to appear [in Nov 82 I was selling software by Not Polyoptics, PS Software, Norton Software, Pewterware, and others! By May 83 to this was added Roach Software, Oak Tree Software, FFF Software, Kuhl Software, Maple Leaf Microware- all advertised in TIDINGS- 5 page ad in May 83 issue!!! and there were also ads by UK houses Lantern and Apex]]]- but it is still difficult to even get a copy of 99er magazine from any source other than our service [never mentioned this service in TIDINGS-first I have heard of it!!!ss].

"To help British TI users, Paul founded a by-mail support group that offers services and helps TI owners contact others in their area to form local users groups. "TIHOME offers members a resource for public domain software ,some computer supplies, publications, local contacts, and technical information.

"TIHOME publishes a 100-page magazine six times yearly, which members receive as part of their annual dues. The magazine offers editorials, comments, and advertisements by members; detailed software and book reviews; and in depth tutorials and program listings. A recent issue features LINE DRAWING ROUTINE as part of a discussion on TI hi-res graphics, reviews of TI Writer and game software, Beginners Basic, tutorials on LOGO and Assembly Language, comments on computer shows and other activities, an in depth article on concepting and programming adventure games, and reports from local users groups all over the British Isles.

"The TI99/4 is still the best made computer in its price range, even in England, Paul comments. Despite the obstacles, true British pluck has paid off. TIHOMES membership went up from 180 to 2000 in a year, and its still growing!

[By the time the book was published TIHOME had been dead for at least 7 months! The comment on membership would place the interview at around April 1983- about the time of the very last magazine. Only the last two magazines ran to 102 pages. The LINE DRAWING ROUTINE mentioned as being in a "recent issue" actually appeared in the last one, May 83.]. A curious interview especially in regard to its unfortunate timing!

==========

In April 1982 there were 180 members in TIHOME. In August 1990, there were 158 members in TIUGUK.

==========

continued from page 7

```
160 PRINT :"Working"
170 OPEN #1:"DSK"&A$,INPUT :: OPEN #2:"DSK"&B$,OUTPUT
180 IF EOF(1)THEN 220 ELSE LINPUT #1:A$
190 IF A$=CHR$(10)THEN PRINT #1:" " :: GOTO 180
200 I=LEN(A$):: IF I=1 OR SEG$(A$,I,1)<>CHR$(10)THEN
    I=I+1
210 PRINT#2:SEG$(A$,1,I-1):: GOTO 180
220 CLOSE #1 :: CLOSE #2 :: STOP
```

Regional Group Reports

Meeting Summary For June

Banana Coast	09/06/91	Sawtell
Carlingford	19/06/91	Carlingford
Central Coast	08/06/91	Saratoga
Glebe	06/06/91	Glebe
Illawarra	10/06/91	Keiraville
Liverpool	07/06/91	
Northern Suburbs	27/06/91	
Sutherland	21/06/91	Jannali

BANANA COAST Regional Group
(Coffs Harbour area)

Regular meetings are held in the Sawtell Tennis Club on the second Sunday of the month at 2 pm sharp. For information on meetings of the Banana Coast group, contact Kevin Cox at 7 Dewing Close, Bayldon, telephone (066)53 2649, or John Ryan of Mullaway via the BBS, user name SARA, or telephone (066)54 1451.

CENTRAL COAST Regional Group

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

GLEBE Regional Group

Regular meetings are normally on the Thursday evening following the first Saturday of the month, at 8pm at 43 Boyce St, Glebe. Contact Mike Slattery, (02)692 0559.

ILLAWARRA Regional Group

Regular meetings are normally on the second Monday of each month, except January, at 7.30pm, Keiraville Public School, Gipps Rd, Keiraville, opposite the Keiraville shopping centre. A variety of activities accompany our meetings. At the last meeting we looked at various functions of Multiplan. Contact Lou Amadio on (042)28 4906 for more information.

LIVERPOOL Regional Group

Regular meeting date is the Friday following the TIshug Sydney meeting at 7.30 pm. Contact Larry Saunders (02) 6447377 (home) or (02) 7598441 (work) for more information.

NORTHERN SUBURBS Regional Group

Regular meetings are held on the fourth Thursday of the month. If you want any information please ring Dennis Norman on (02)452 3920, or Dick Warburton on (02)918 8132.

Come and join in our fun. Dick Warburton.

SUTHERLAND REGIONAL GROUP

The April meeting provided the group with an opportunity to review the Spellit software. Derek Wilkinson has spent some time in creating a modified version which allows him to set up his own dictionary files from scratch.

Herbert Schade was also given some driving instructions for his recently purchased copy of TI Artist Plus. Herbert is one of the newer members who has now acquired a disk expansion system.

Derek also gave his new 360k half height drive a run. It turned out to be a real bargain at $40, brand new, from a local IBM dealer.

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

Peter Young
Regional Co-ordinator

TIsHUG in Sydney

Monthly meetings start promptly at 2pm (except for full day tutorials) on the first Saturday of the month that is not part of a long weekend. They are held at the RYDE INFANTS SCHOOL, Tucker Street (Post Office end), Ryde. Regular items include news from the directors, the publications library, the shop, and demonstrations of monthly software.

JUNE MEETING - 1ST JUNE

The June meeting will involve demonstrating the latest software and will begin at the usual time of 2:00pm.

****

The cut-off dates for submitting articles to the Editor for the TND are:

July	9 June
August	7 July
September	11 August

These dates are all Sundays and there is no guarantee that they will make the magazine unless they are uploaded by 6:00pm, at the latest.

****

TIsHUG Meetings for Sydney, 1991

June
Demonstrations of the latest software or hardware.

July
The second Buy, Swap and Sell day. Bring things for sale and also your money for the terrific bargains. You have to be early to beat Rolf to a bargain!

August
Demonstrations of the latest software or hardware. Watch this space for details.

September
Demonstrations of the latest software or hardware. Watch this space for details.

October
The third Buy, Swap and Sell day. Your last chance this year to get some money for Christmas presents or to get an early present for yourself.

November
The second all day tutorial session.

December
The Annual General Meeting followed by some festive eats and drinks. Make sure you attend and give your support to all the workers in the club.

For Sale

TI-994A with

* 32k memory expansion,
* P.E. box,
* D.S. disk drive with a TI controller card,
* Axiom printer interface,
* Brother M1109 printer,
* Piles of books,software,cartridges, etc.

Price....only 600 dollars
(See Dick Warburton for details)

Note also that there are two 184 K ramdisks available at only $150 each.

Printed by University of Wollongong PRINTERY SERVICES