NEWS DIGEST

Focusing on the TI99/4A Home Computer

Volume 8, Number 7 July, 1989

Registered by Australia Post - Publication No. NBH5933

ZOUNDS
see TIDBITS EIGHT!!
Voice#

P.O. Box 214, Redfern, New South Wales, Australia, 2016 $2

TIsHUG (Australia) Ltd.

TIsHUG News Digest

July 1989

All correspondence to:

P.O. Box 214
Redfern, NSW 2016
Australia

Membership and Subscriptions

Annual Family Dues	$25.00
Overseas Airmail Dues	AUS$50.00

TIsHUG Sydney Meeting

The next meeting will start at 2 pm on 1st of July at the Woodstock Community Centre, Church Street, Burwood.

Cover by Wade Bowmer

Printed by
The University of Wollongong
Printery

TIsHUG News Digest ISSN 0819-1984

Index

Watch this

space

They're off

by Geoff Trott

We had a very busy meeting last month, at the fix-it booth. It seems like there have been a rash of CPU failures recently. Have there been any power surges in Sydney recently? Or have people been computing when lightning is about? I, and a few grateful owners, are very pleased that someone bought a heap of TMS9900 processors a while back. Good on you Peter from Melbourne! I was hoping to go and listen to a few of the tutorials like the one on c99, but we were so busy that I only had time for a chat with Tony McGovern. I guess that is one of the problems with multi-purpose tutorial days. The workers are stuck in one place for the whole day. I hope that all the other activities were just as busy. Some of our customers seemed to be with us for the whole day too!

continued on page 34

Co-ordinator's Report

by Dick Warburton

Well, the tutorial day has been and gone. What a busy day. After lugging two full systems around in the rain on Saturday, I am beginning to appreciate the virtues of one of Peter Schubert's Mini Systems. It really is a fine system, and we could do many people a real favour by pointing them in that direction. Craig's strenuous efforts were rewarded with a good attendance, and a very busy day. Tutorial days at Woodstock however, are difficult to organize effectively, because of the difficulty of booking sufficient rooms. The setting is excellent in other ways and it is well located for our membership. If our meetings grow much larger, we might have to relocate.

I have been discussing tutorial days with Ross Mudie, and he made the very valid point that tutorials, held even monthly, lose their impact fairly quickly, unless the participants get the opportunity to practice what they learn, and receive some follow up to motivate them to keep going. Ross has wanted to run really effective tutorials for some time, and this led him to offer to conduct the full 2 day programme in assembly in June. The course has been limited to only 15 participants, who must bring their own expanded systems, and will run the whole two days. I am hoping to crash the assembly barrier myself, without too much damage, and emerge intact with my diploma. If I succeed, I will be able to hold a conversation with Shane Ferret. I personally think that Ross is making good sense. Most organizations conduct intensive training in small groups in a setting removed from the normal one. If this tutorial goes as well as it seems it will, I can see a whole range of possibilities for future groups. We might arrange a weekend in Wollongong, learning to do all manner of technical things. We could run classes for newer members in basic areas, and really get people going on their machines. I like the idea, and I will support any reasonable proposals on this line.

I was pleased to meet Tony McGovern who travelled down from Newcastle to join us on the tutorial day. I was impressed with his latest version of Funlwriter, which now runs on an 80 column card, and has many great features.

Congratulations to the Wollongong Push, for their great effort on the hardware side. Consoles were being fixed, power supplies dismembered, and the editor's system was operating. The room seemed full of people all the time. At least two new members joined up after going inside. Despite the rain, it was a great day. There is no doubt that a good number of our membership have a real interest in developing their technical skills. I would like to see TIsHUG members grow sufficiently proficient to be self-reliant, and even contribute to the further technical development of the TI99/4A. We have the talent. We simply need more people to join in and contribute. In time, we might be able to set up a group project successfully.

Interest groups continue to function well. I am finding the console repair group tremendously interesting. I have managed to get another console up and running. Ross tells me I have been very lucky. I get great satisfaction from fixing things. If I can learn, any body can. Come along, and bring a dead console with you. If you have bits and pieces lying about, eg keyboards, power supplies etc. bring them along. But most of all, have a go yourself at fixing them there.

Les seems to be going from strength to strength with TI-Artist, and the group seems to be getting more proficient all the time.

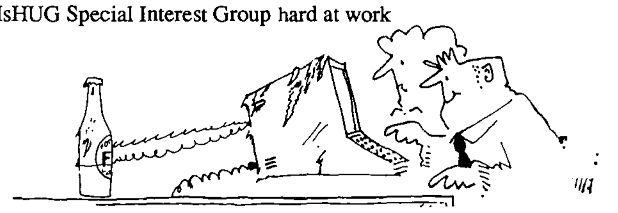
At Percy Harrison's place, word processing continues. Already, the group is sharing ideas and tips and it is developing well.

The Publicity Group, continues to meet in Glebe, and is planning all manner of ways to publicize the club. We have begun to advertise in the Trading post, and already there has been a response. Hopefully,TIsHUG will have a telephone listing in the next phone book. We are considering the possibility of a home computer show in Sydney, involving all the major user groups. If you have ideas about publicity, come along and help us out. We need helpers. By the way, do you know someone with a TI99/4A who is not a club member? Tell them about TIsHUG, and how we can help them. Do you know someone who wants to buy a computer? Tell them about the TI99/4A. We can probably arrange to supply what they want through the club. If you sell any equipment, make sure you let people know about the club. Finally if you know of someone with a TI99/4A outside the club, give me their name and address, so we can arrange for one of our members to contact them, or send them a letter.

I would like to set up a games room at our monthly meeting for the young people who come. I want to encourage more young members to attend and perhaps bring friends with them. I do not know how we can do it at Woodstock, unless they were able to set up outside on the verandah. What do you feel about this? I would like to see technical advice and help available each month. If you have hardware to sell, try it at the meeting through the shop. See Cyril and maybe it will be sold on the spot.

Well I will see you at the next meeting, if I survive Ross's crash course in assembly. Keep those ideas coming in. o

TIsHUG Special Interest Group hard at work



Letter to the Editor

Dear Geoff,

I had nothing to do tonight so I thought that I might write to you as the editor of the TND. I was talking to Joe Wright (Hunter Valley 99ers) the other night and I mentioned to him that we (that is the members of our respective clubs) should get together some time. I seem to remember previous attempts to do just that, without much success. I think, however, that we should not give up. I mentioned to Joe that the proposed 1989 TI-Fair would be a good place. The Fair would probably be held in Melbourne this year and in Sydney next year. Joe said that he was not particularly fond of fairs as they required a lot of organisation and would only appeal to a very limited group (like preaching to the converted). He added that the general public was not really interested in an orphan computer and suggested a "TI99/4A Symposium" with noted speakers such as Dr Tony McGovern, Ross Mudie, Dr Geoff Trott and any others who are willing to make a contribution to such an event. The Symposium, he added, would generate a lot of interest among active and inactive TI99/4A owners, if it was properly advertised. Furthermore it would be a lot easier to organise and would bring together, at least annually, 99ers from other clubs who would otherwise see very little of each other, year in year out.

Naturally this would also be an opportunity to demonstrate the latest in hardware and software.

The real benefit, however, would be the bringing together of the user groups, and in this way we can guarantee a prolonged future for the TI99/4A computer.

Perhaps this is what we need and what TIsHUG should be thinking of organising for 1990?

Sincerely Yours,
Lou Amadio
Illawarra Regional Group o

Secretary's Notebook

by Terry Phillips

The June meeting, despite the continuing Sydney rain, turned out to be well attended with about 70 members in attendance. I gather that most enjoyed the day, based on the comments I heard. Unfortunately I was too busy with the software copying to get around to any of the lectures, but on the couple of occasions I had a break there looked to be very interested groups gathered around all the tables eagerly watching the demonstrations or listening to the lecturers.

It was also a good day in that 4 new members were enrolled.

Let us give a big welcome to them:

Tom Pearson	- Carlton
T Beverley	- Whalan
H Burnicle	- Carlton
William Long	- Terrytown USA

The first mentioned 3 were at the meeting and appeared to be enjoying themselves. William Long was nominated by long time member Hal Payne.

Our membership now stands at 197, including 4 overseas members. As I write this, 46 of the memberships that expired at the end of April have not been renewed. Reminders have gone to all concerned, so I anticipate that some will renew. Looking ahead, it appears we will continue through 1989 with just over the 200 membership mark. Still not too bad I suppose.

From the mailbag comes the following items:

Bob Bunbury has contacted some TI99/4A users and reports that there could be some new memberships coming shortly.

Larry Cross advises that his RS232 card has packed it in and must be removed to reboot the system. Larry asks if it can be repaired. Larry I suggest in the first instance you contact Russell Welham, Central Coast Regional Group. Russell may be able to advise you, and if not able to repair, bring the card down to Sydney for one of our technical experts to have a look at it.

Kerry Harrison writes to say thanks to everyone for all the hard work they put in, particularly the Editor and his gang who produce the TND. Thanks Kerry.

Tony Imbruglia writes to thank everyone for holding the group together and offering his assistance if required. Thanks Tony good to know we can call on you when needed.

Jim Banfield has written saying he would like to see more articles on hardware in the TND and offering contributions. Send them down Jim, I am sure the Editor would appreciate all the material he can get.

Mark Richardson writes to say he is having trouble with a couple of games programs he has typed in, one from the old Bumper Book, Alien Attack which is giving him a FOR-NEXT ERROR and the other from the March TND, Magic Squares, which does not seem to set up properly. Anyone else having trouble or can anyone assist Mark. Mark would also like to see more action games published in the TND and has some modules he would like to trade. If you wish to get in contact with Mark (he is a far distant country member) please write to him care of the Secretary.

Peter Smith writes to pass on his congratulations to the editing committee. The standards have improved immensely with plenty of interesting and varied articles. Thanks Peter, the Editor appreciates this type of feedback.

Paul Robinson (Telephone 043591532) has the following for sale. TI99/4A console, Memory Expansion and RS232, Speech Synthesizer, Joysticks, Joystick Adapter, Cartridge Expander, 20 games cartridges, many tapes and 3 books, together with a brand new NTSC version computer. He is asking $450 for the lot.

Sadly, I am advised that the TISIG of the San Diego Computer Society has disbanded. We have been exchanging with them for some time now.

On the other hand we have now started exchanging newsdigests with the Mid-South 99'ers of Germantown Tennessee. President Gary Cox has written to confirm this arrangement and has enclosed copies of their February, March, April and May 1989 issues. Looks good. Check it out in the Publications Library.

That is it for this month. See you at the meetings or if you cannot make it and have a problem, write. If I cannot answer it, I will pass it on to someone who can.

continued from page 10

```
340 GOTO 110
350 CALL CLEAR
360 PRINT "I DONT THINK YOU OUGHT TO":B$
370 INPUT "DECIDE AGAIN (Y/N)":SD$
380 IF SD$<>"N" THEN 110
390 END
```

Dear Jenny,

Here is a program to test a person's knowledge of Weimar Geramny. I hope it is OK for the magazine.

Vincent Maker

```
100 CALL CLEAR
110 REM MODERN HISTORY
120 REM BY VINCENT MAKER FOR ALL MY FRIENDS AT "ANTIOCH"
130 DISPLAY AT(5,7):"WEIMAR GERMANY."
140 DISPLAY AT(7,7):"RESEARCHED BY V. MAKER"
150 FOR Y=0 TO 300
160 NEXT Y
170 CALL CLEAR
180 PRINT "HERE ARE FIVE QUESTIONS ON  WEIMAR GERMANY,
    ASSESSED IN THE FORM OF MULTIPLE CHOICE."
190 PRINT
200 INPUT "PRESS ENTER ONCE YOU HAVE   READ AND
    UNDERSTOOD THIS.":LKJ$
210 CALL CLEAR
220 DISPLAY AT(3,7):"WHICH OF THE FOLLOWING WAS  THE
    DATE THAT GERMANY BECAME A REPUBLIC?"
230 DISPLAY AT(8,3):"A)11th AUGUST, 1919           B)28th
    JUNE 1918             C)9th NOVEMBER 1918
           D)28th JUNE 1918"
240 PRINT "PRESS YOUR GUESS."
250 CALL KEY(0,J,K)
260 IF K=0 THEN 250
270 IF J=67 THEN RIGHT=1 ELSE WRONG=1
280 IF J=67 THEN PRINT "WELL DONE, CORRECT!!" ELSE PRINT
    "SORRY, INCORRECT."
290 FOR T=0 TO 300
300 NEXT T
310 CALL CLEAR
320 DISPLAY AT(3,7):"WHO LED THE COUP D'ETAT THAT WAS
    DEFEATED IN 1920 BY A GENERAL STRIKE OF WORKERS IN
    BERLIN?"
330 DISPLAY AT(7,7):"A)DR. KAPP
                  B)HITLER
                    C)HINDENBURG
                 D)DAWES"
340 PRINT "PRESS YOUR GUESS."
350 CALL KEY(0,U,H)
360 IF H=0 THEN 350
370 IF U=65 THEN RIGHT=RIGHT+1 ELSE WRONG=WRONG+1
380 IF U=65 THEN PRINT "CONGRATS, RIGHT!" ELSE PRINT
    "SORRY, WRONG."
390 FOR T=0 TO 300
400 NEXT T
410 CALL CLEAR
420 DISPLAY AT(3,7):"WHICH PARTY HELD THE MOST  SEATS
    IN THE REICHSTAG      BETWEEN 1919 and 1930?"
430 DISPLAY AT(7,7):"A)LABOR     B)S.P.D.
    C)U.S.P.D.       D)LIBERAL"
```

continued on page 4

TIsHUG Software Column by Terry Phillips

It was certainly a busy day at the June full day meeting with Les Andrews and my systems going flat out all day to keep up with the demand for disk copies. Thanks for you help Les, and I hope that most of you who came along got what you wanted.

On the subject of disk copies, I have been a little disappointed of late with the response to orders for Jim Peterson's Tigercub software range. A lot of time and effort have been expended in acquiring this software, unarchiving it and putting together a list of what is available together with an order form. To this point orders for only 7 disks have been received. Kind of makes you wonder whether it is worthwhile sourcing out software. Perhaps the membership does not want software if it has to be paid for. Anyway, to refresh your memory, go back to the May TND and have a look at the range of software that I am referring to.

Here is a list of new disks added to the library:

DISK A343 - GAMES DEMONSTRATIONS. Some good games titled NINJA, PANIC and RAMPAGE, together with demonstrations LINES, RECTANGLES, SPEECH and TRIANGLES.

DISK A344 - GAMES, 3 rather good games on this disk. Titles are 3D MAZE, NIGHT FIGHTER and POWERBALL.

DISK A345 - Documentation files for the GAME OF WIT series.

DISK A346 - GAME OF WIT SERIES, titles are GAME OF WIT, NIT WIT, WIT OUT, WITS END and WITTLETAGS. These are good strategy word games.

DISK A347 - DUTCH ASSEMBLY DEMONSTRATIONS, excellent colourful screen demonstrations of Rotating Planets, Kaleidoscopes and Title Screens.

DISK A348 - DUTCH ASSEMBLY UTILITIES. Very good assembly material with documentation file on the disk.

DISK A349 - MUSIC. Contains a selection of popular songs with graphics.

DISK A350 - DUTCH ASSEMBLY UTILITIES. Another disk of good assembly material.

DISK A351 - J C BACH MUSIC DISK. Very well done music, some of the best I have heard. Also contains a lengthy file with information on the life and times of J C Bach.

DISK A352 - CHOPIN MUSIC DISK. Not badly done but not as good as the music on A351.

DISK A353 - RAVEN MYSTERY. This is a type of adventure to guess who committed the crime. Requires TE2.

DISK A354 - PERSONAL RECORDS KEEPER. This looks like quite a useful well programmed utility. I have not yet an opportunity to thoroughly put it through its paces.

DISK A355 - Library copies of TYPEWRITER and BEYOND VIDEO CHESS. Neither of these are for distribution.

DISK A356 - Library copy of SPAD XIII. Again not for distribution.

DISK A357 - MORE DUTCH ASSEMBLY UTILITIES. Yet more to add to your collection.

DISK A358 - PICTURES. Drawn by our own Darren Telford using Picasso but on this disk to load through MAX RLE. Darren is fairly artistic and these pictures reflect that.

Using TI-BASE, I recently completed a data base which contains an alphabetical listing of the entire contents of the software library. There are actually 2 data bases. The first containing the contents of disks 1 to 230 (2118 entries) while the second disks A1 to A358 (875 entries). The prime purpose in doing this was to create an alphabetical cross reference list to enable any program in the library to be readily found. Be warned if you contemplate putting as many entries as this in a TI-BASE data base then be prepared for a long wait while it goes through its sort routine. The one with 2118 entries took some 6 hours to sort and this was using a RAMdisk. With a normal disk drive it would be much longer and thrash hell out of the drive as well. Anyhow, for what it is worth, the 2 data base files, fully sorted are available on disk for anyone who wants them.

If time permits I will copy some of the assembly utilities mentioned above and have them available at the July meeting. If in the meantime you would like something from the library, write or give me a call. o

continued from page 3

```
440 PRINT "PRESS YOUR GUESS."
450 CALL KEY(0,D,F)
460 IF F=0 THEN 450
470 IF D=66 THEN RIGHT=RIGHT+1 ELSE WRONG=WRONG+1
480 IF D=66 THEN PRINT "VERY GOOD, RIGHT!" ELSE PRINT
    "BETTER LUCK NEXT QUESTION, WRONG."
481 FOR T=0 TO 300
482 NEXT T
483 CALL CLEAR
490 DISPLAY AT(3,7):"WHICH INDUSTRIAL AREA OF GERMANY
    DID FRANCE OCCUPY IN 1923 IN ORDER TO FORCE GERMANY
    TO MEET HER REPARATION PAYMENTS?"
500 DISPLAY AT(9,7):"A)RUHR
                      B)BELGIUM
                   C)BERLIN(EAST)
              D)BERLIN(WEST)"
510 PRINT "PRESS YOUR GUESS."
520 CALL KEY(0,J,H)
530 IF H=0 THEN 520
540 IF J=65 THEN RIGHT=RIGHT+1 ELSE WRONG=WRONG+1
550 IF J<>65 THEN PRINT "SORRY, WRONG." ELSE PRINT "WELL
    DONE, RIGHT!!"
560 FOR T=0 TO 300
570 NEXT T
580 CALL CLEAR
590 DISPLAY AT(3,7):"WHO WAS APPOINTED
              CHANCELLOR ON 30th JANUARY, 1933?"
600 DISPLAY AT(7,7):"A)HINDENBURG
                B)LUDENDORFF              C)HITLER
                     D)DR. KAPP"
610 PRINT "PRESS YOUR GUESS."
620 CALL KEY(0,S,D)
630 IF D=0 THEN 620
640 IF S=67 THEN RIGHT=RIGHT+1 ELSE WRONG=WRONG+1
650 IF S<>67 THEN PRINT "SORRY, WRONG." ELSE PRINT "VERY
    GOOD, RIGHT!!"
660 IF RIGHT=0 THEN A$="DON'T TAKE THIS UP FOR A LIVING.
    ZERO OUT OF FIVE."
670 IF RIGHT=1 THEN A$="THIS IS NOT GOOD.  ONE OUT OF
    FIVE."
680 IF RIGHT=2 THEN A$="COULD HAVE BEEN BETTER.  TWO OUT
    OF FIVE."
690 IF RIGHT=3 THEN A$="THAT'S GOOD.  THREE OF FIVE."
700 IF RIGHT=4 THEN A$="THAT'S NEARLY PERFECT.  FOUR OUT
    OF FIVE."
710 IF RIGHT=5 THEN A$="THAT'S AN EXEMPLARY SCORE.
    CONGRATULATIONS, FIVE OUT OF FIVE."
720 FOR T=0 TO 300
730 NEXT T
740 CALL CLEAR
750 DISPLAY AT(7,7):A$
755 PERCENT=100*(RIGHT/5)
760 DISPLAY AT(10,1):"YOU SCORED ";RIGHT;" RIGHT AND
    ";WRONG;" WRONG.  THAT RATES ";PERCENT;" %."
770 PRINT
780 PRINT
790 PRINT "HOPE YOU DO WELL AT MODERN HISTORY! BYE!"
800 .
```
o

Techo Time

with Lou Amadio

The June all day tutorial was a huge success, at least from the hardware point of view. Demand for repairs was so great that we had to double the number of tables available to strip and diagnose faulty consoles.

However, there were also many other interesting talks going on in several rooms, but unfortunately I was too busy to participate in any of them. In any case, Craig Sheehan should be congratulated for organising such an informative day.

The highlight of the meeting was undoubtedly meeting Tony McGovern of Funnelweb fame. (He is a lot taller than his programs indicate!) Tony demonstrated the latest revisions to the Funnelweb package (soon to be released) as well as enhancements to the 80 column versions of TI-Writer. Tony's use of windows makes for an interesting screen display and we eagerly await the release of these updates.

After the meeting, we "dined" at the local cafe then on to a local railway station to see Tony off for Newcastle. It was nice to have had Tony for the day, and we thank him for making the effort to come to our meeting.

Hardware Workshop

Four faulty CPUs, a faulty VDP RAM and an 8-bit latch were attended to, but unfortunately we ran out of time. Some happy customers went home at the end of the day with working consoles. However, it is obvious that another session is required in the near future as Geoff ended up taking six consoles home for further attention.

I wonder how many other user groups offer such a service to their club members?

The outcome of this workshop plus at least two previous sessions at Cyril Bohlsen's place means that a good number of consoles are now back in service.

My thanks to Geoff Trott, Robert Peverill, Peter Schubert and John Paine for their valuable assistance on the day.

Apart from console repairs, there was also interest in joystick interfaces, multi-modules, 32K memory expansions and the two-way I/O interface which is described this month.

Enhanced Keyboard

At the June meeting I had a chance to play with Derek Wilkinson's modified IBM style keyboard. I mentioned previously that Derek had managed to rewire the the new keyboard and reconstruct some of the dual key functions into single keys, for example, the FCTN and CTRL combinations. As we did not have an expansion system, I was only able to test the keyboard in BASIC. All the keys worked well and it was quite a change, for example, to be able to move the cursor around by just pressing the appropriate key. I believe that this would make a terrific hardware project for Techo Time. How about it Derek?

Hardware Hints

Sooner or later, if you are not careful, during dismantling and re-assembling a console, you may end up breaking the fine wire cable interface between the keyboard and the TI99/4A motherboard. The simple solution is to remove the keyboard when you remove the motherboard. It is only another four screws and is much less work than rewiring the interface.

Loading cassette programs saved on another system can sometimes be very frustrating. The steps usually involve making small adjustments to the volume and tone controls and being guided by the error messages on the screen. If all else fails, you can try adjusting the azimuth of the playback head to match the signals recorded on the tape. This is best done with the tape recorder disconnected from the computer cable. During playback, adjust the azimuth screw (the one next to the head with the spring) to get maximum volume from the speaker. You now have a better chance of loading the program. Take note of the adjustment so that you can put it back to where it was. Use a little nail polish to lock the screw when you are finished.

Mail Box

I have received several letters lately, mostly through Terry Phillips. The first was from Eric Whelan of Healsville, Victoria. Eric wants to interface a Microbee 7030H RGB monitor to a TI99/4A console. With the present interest in monitors we are currently looking at what is possible at a reasonable price.

A letter from J Hagart (Gordonvale, Qld) needing help with console repairs. Apparently the TV screen just shows a blue flash when he switches the console on. It sounds as if the brown inductor associated with the VDP is faulty. We can supply a replacement inductor if required.

A query from Frank Hall (Old Bar, NSW) on whether he can utilise a transformer with 45V secondary. However, it seems to be too high for most electronics projects and too low for an autotransformer for the PEB.

A letter from Greg Bull (Deniliquin, NSW) about a problem with his console which has recently had an internal 32K RAM fitted. The memory works well except that he is no longer to able to run his text to speech program with his speech synthesizer from BASIC, although speech still works with Extended BASIC. Can anyone help with this problem?

Finally a letter from Bill Longmuir (Castle Hill, NSW) enquiring as to the availability of 32K kits and Joytalk kits. 32K memory kits are available from the club for $35 while stocks of the 6264 8K RAM chips are available from the shop. Joytalk kits are not yet available, as a printed circuit needs to be designed. The circuit could, however, be built on Veroboard if necessary. Joytalk needs special software (available from Ross Mudie) as well as 32K and Extended BASIC. o

Direct I/O Interface
for PEB cards

by Lou Amadio

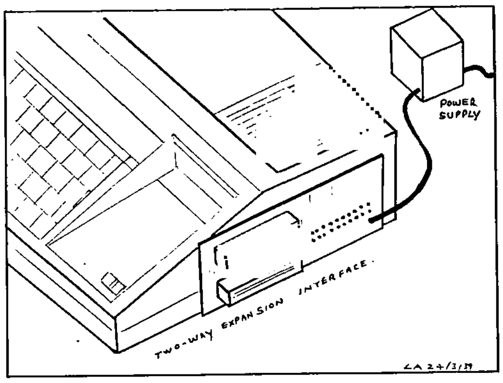
(This project is based on a prototype unit built by Geoff Trott about 3 years ago.)

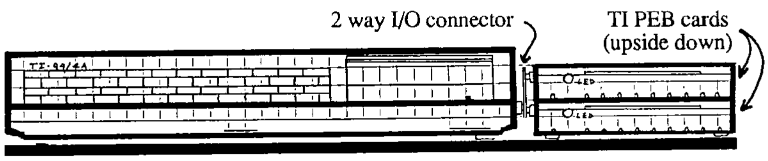
With the limited availability (and high price) of second hand TI99/4A Peripheral Expansion Boxes and the fact that the Peter Schubert Mini Expansion System is no longer readily available, the time is right to outline yet another way to expand your TI99/4A. The device that I am about to describe will allow you to connect up to two TI (or equivalent) PEB cards directly to the console.

A typical configuration would be a 32K RAM and a disk controller, while an all out system could have a Peter Schubert Multi-function Card (32K, RS232, PIO and double density disk controller) with a RAMdisk. Alternatively, you could add a modem or even a PGRAM card! Note that some configurations may require a separate (internal) 32K memory expansion to work properly.

The interface has its own power supply, which is independant of the console power supply.

Because of the proximity of the PEB cards to the console, no buffer chips are required, simplifying the construction considerably.





Front View of Console and I/O interface   L. Amadio, TIsHUG 23/5/89

Parts Required

1 x 12V, CT, 1.2A transformer
1 x 25V, CT, 1.2A transformer
2 x 3A power diodes
4 x 1A power diodes
1 x 7805 +5 volt IC regulator
5 x 2200 uF 25V chassis caps
2 x 1uF 25V tantalum capacitors
3 x 10uF 25V tantalum capacitors
3 x .1uF 25V capacitors
7 x 47 ohm 0.5 watt resistors
3 x 1000 ohm 0.5 watt resistors
1 x 44-way 0.1" edge connector
2 x 60-way 0.1" edge connector
2 x 60 by 17 hole veroboard
8 x rubber feet (7mm thick)
0.5A fuse and holder
Power cable, grommet and clamp
Aluminium box

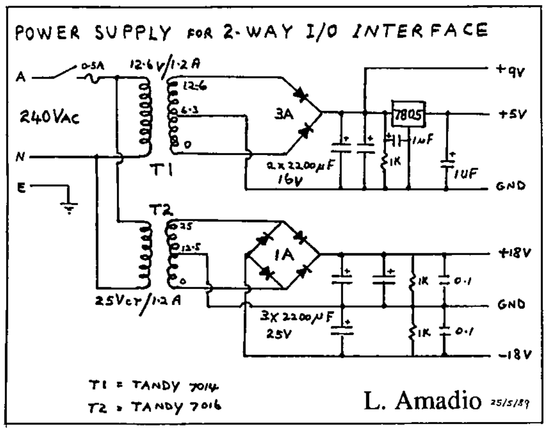
The Power Supply

Power requirements for two PEB cards is approximately:

+18 volts at 0.4 amps
-18 volts at 0.1 amps
+9 volts at 0.9 amps

Non TI cards are likely to require lower currents, attributable to the usage of modern low-power chips. In any case, the unregulated voltages should not exceed the levels indicated above otherwise the card voltage regulators may run too hot. For this reason, metal cased TI PEB cards are preferred for this application as the clamshell provides a heatsink for the voltage regulators. Refer to the notes on how to adapt higher voltages to the task.

The circuit diagram for the power supply is shown below. It is fairly conventional and consists of two low cost transformers and a number of rectifiers and filter capacitors (two transformers were used as they were found to be cheaper than a multiple winding unit).



If you use chassis mounted types for the main filter capacitors, you can mount the diodes directly onto the capacitor lugs and use point to point wiring for input and output. This greatly simplifies construction. Use the circuit diagram to ensure correct polarity of the diodes and capacitors. The bleeder resistors across the capacitors ensure that the power supply will discharge on power off even if no cards are connected.

The 5 volt regulator is included to supply a constant voltage for a number of "pull up" resistors.

All the components are mounted in an aluminium box which is suitably earthed. The box also makes an excellent heat sink for the 5 volt regulator. Bolt the regulator directly to the box, close to the filter capacitors. There is no need to insulate the metal tab as it is at earth potential. Solder 1uF tantalum capacitors between the input and earth and the output and earth (negative side of the capacitor goes to earth).

Parallel the primary windings of the power transformers and connect them to the mains via a 3 pin plug, cable and 0.5 A fuse. A power switch is optional as most computer systems are switched off at the wall when not in use. Ensure that the mains cable is safely secured to the box using a grommet and anchor lug and that the earth lead (green) is firmly connected to the metal of the box. A cable with 6 wires (rated at 1 amp each) is required between the power supply and the I/O interface.

The Interface

The console I/O has 44 contacts (2x22) and the PEB cards have 60 contacts (2x30) each. I used wire wrap edge connectors for this project as these have longer pins which are suitable for stand-off soldering. The connectors were cut down from longer units, as these were cheaper than buying the correct size. If you decide to make your own, cut the edge connectors for a neat fit with both the console and TI99/4A expansion cards. Remove unwanted pins by pulling them out from the wire wrap end with a pair of fine nose pliers. Chamfer the top and bottom edges of the 44-way connector to aid insertion into the console I/O port. Thoroughly clean the connector to remove any particles of plastic which may be lodged between the metal contacts.

The connectors were mounted on 2 strips of 150 mm wide (0.1 inch hole spacing) veroboard. Cut two pieces of veroboard 60 holes by 17 holes as indicated. Note the direction of the copper tracks. Place the boards back to back (copper side out) and mark the location of the edge connectors using the diagram for assistance (the boards are used back to back for added strength).

It is important that the edge connectors are spaced sufficiently apart to fit two standard PEB cards. With a hole spacing of 9 this will be a very neat fit. Use a spacing of 10 holes (that is 1 inch) if you want a little more clearance.

The vertical clearance between the 44-way and the lower 60-way connector was designed to allow for 6 mm thick rubber feet beneath the PEB card. The card is mounted upside down for convenient peripheral connection at the rear. Cards not housed in a metal (or plastic) clamshell must be supported by at least two standoffs. In this case, a cover should be made from a piece of aluminium with suitable holes drilled for the activity LEDs.

The points marked with an "x" on the "veroboard" diagram indicate where the tracks must be cut on both the top and bottom boards. Do this carefully with a 3mm bit and a hand operated drill. Run the sharp edge of screwdriver blade across the cut tracks to ensure that there are no thin copper slivers still shorting the tracks together. Place the two strips of veroboard together and fit the 3 edge connectors in place. Note that the console (44-way) edge connector is located on the opposite side to the PEB card connectors. Ensure that the tracks cut previously align with the edge connector pins. Allow approximately 5 mm gap between the back of the edge connector and the veroboard and solder only 2 pins (opposite corners) of each connector to the board on one side only. Carefully check for alignment using a set square and ruler in both the horizontal and vertical planes and realign if necessary. Correct alignment here will ensure trouble free operation later. Check for correct spacing between the 60-way connectors by plugging in two TI PEB cards. If all is well solder the rest of the pins in place on both sides (see footnotes below). Ensure that the veroboards are kept firmly together during this operation. When finished, check with a resistance meter that no two pins are connected together.

Mount the 47 ohm pull-up resistors first to the points indicated on the table below, and then to a common point above the board which can be connected directly to the regulated +5 volt supply.

The next stage is to make the direct links between the 44-way and the bottom 60-way connectors. The 44-way connector has pin 1 on the bottom closest to the front of the console, and the 60-way connector has pin 1 on the top closest to the rear of the console. Follow the instructions in the table below and complete the connections one at the time. As there are a large number of connections to be made, it is best to use fine multistrand coloured wire. Some of the connections will be fairly close so leave a little extra length on the wire to allow for soldering. For each link in turn, measure, cut, strip and pre-tin the ends of the wire and also pre-tin the start and finish points on the board prior to soldering the link. Make the connections to the veroboard tracks adjacent to the pins rather than to the pins themselves. Taken one step at the time you will soon have finished the most difficult part of the project. When completed, double check everything with a resistance meter. It helps to have a friend check your connections while you call the start and finish points.

Connect the rest of the power supply leads to the appropriate pins as per above table. Use a suitable common point for all of the ground connections. Decouple the power connections by soldering 10uF capacitors to earth. Take care with supply polarity.

The Connections

I/O #	Function	PEB #	I/O #	Function	PEB #
1	+5V SP	NC	23	GND	7,20
2	SBE	NC	24	CPUCLK	50
3	RESET	6	25	GND	27,47
4	EXT IN	17	26	WE	54
5	A5	40	27	GND	49,53
6	A10	33	28	MBE	NC
7	A4	39	29	A6	37
8	A11	30	30	A1	44
9	DBIN	52	31	A0	43
10	A3	42	32	MEMEN	56
11	A12	31	33	CRUIN	55
12	READY	4	34	D7	19
13	LOAD	NC	35	D4	24
14	A8	35	36	D6	22
15	A13	32	37	D0	28
16	A14	29	38	D5	21
17	A7	28	39	D2	26
18	A9	36	40	D1	25
19	A15	30	41	IAQ	NC
20	A2	41	42	D3	23
21	GND	3,5	43	-5V SP	NC
22	CRUCLK	51	44	Audio	NC

The following relate to the PEB 60-way connector:

Pins 1 and 2 to +9V unregulated.

Pins 3, 5, 7, 20, 27, 47, 49, 53 connected to ground.

Pins 12, 13, 15, 16, 45, 46 and 48 have 47 ohm (0.5 watt) resistors to +5 volts regulated.

Pins 57 and 58 to -18V unregulated.

Pins 59 and 60 to +18V unregulated.

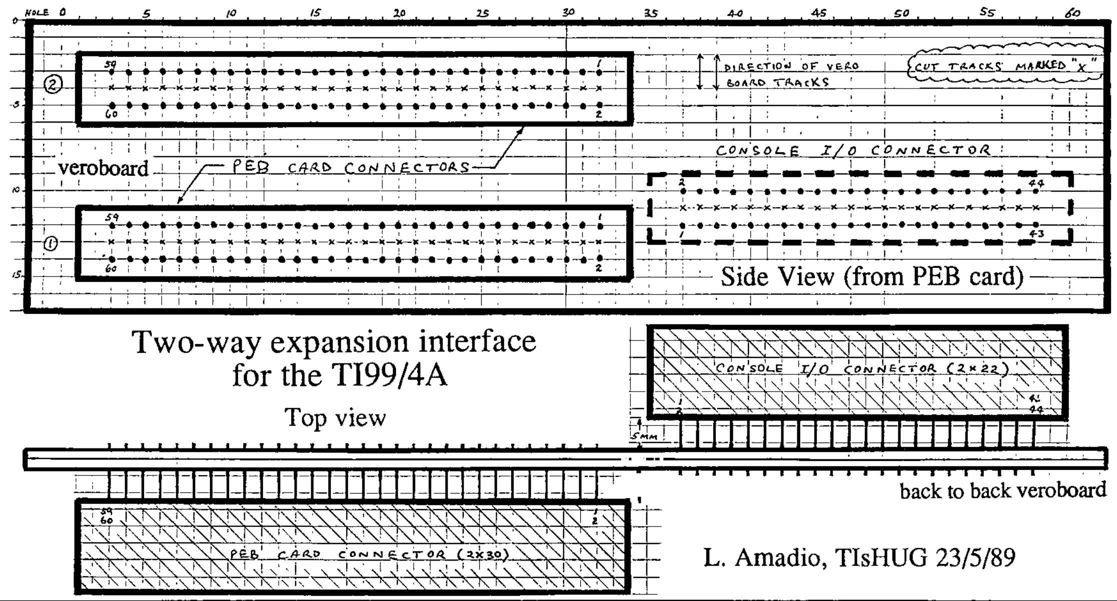
Pins 8, 9, 14, 18, 11, 12, 13, 15, 16, 45, 46 and 48 are not connected.

Power Up

Note: Under no circumstances should you plug or unplug PEB cards with the power on. Allow at least one minute after switching the power off.

With the interface detached from the console, and without plugging in any PEB cards, switch on the power and check that the correct voltages are present where you expect to find them on the edge connector contacts.

60-Way: +9V on pins 1 and 2
60-Way: -18V on pins 57 and 58
60-Way: +18V on pins 59 and 60
60-Way: Ground pins 3, 5, 7, 20, 27, 47, 49 and 53


HOLE 0 5 10 15 20 25 30 35 40 45 50 55 60
DIRECTION OF VERO BOARD TRACKS
CUT TRACKS MARKED "X"
veroboard
PEB CARD CONNECTORS
CONSOLE I/O CONNECTOR
Side View (from PEB card)
CONSOLE I/O CONNECTOR (2x22)
5mm
back to back veroboard
PEB CARD CONNECTOR (2x30)

Two-way expansion interface for the TI99/4A

Top view

L. Amadio, TIsHUG 23/5/89

44-Way: No voltages present on any of the contacts when disconnected from the console

If all is well, switch off the power, connect one PEB card and plug the interface into the console I/O port. Support the card with self adhesive rubber feet. Switch on the power to the console only. The console should operate normally. If not, unplug the interface and check your wiring. (Note: Make sure that the I/O contacts are clean). If the console operates normally, switch on the interface power and check the voltages present on the pins indicated above.

Once the bottom 60-way connector is working satisfactorily, it is time to connect the two 60-way connectors in parallel. Wiring is done with direct links between the pins, ie from pin 1 of the bottom connector to pin 1 of the top connector, and so forth. Recheck as above first without, then with the PEB cards. Measure the unregulated voltages going into the cards. These should be no lower than 16 volts and 8 volts for the supplies in question.

Veroboard

60-way Veroboard may be hard to get, so a limited supply, cut to size, will be available from the shop.

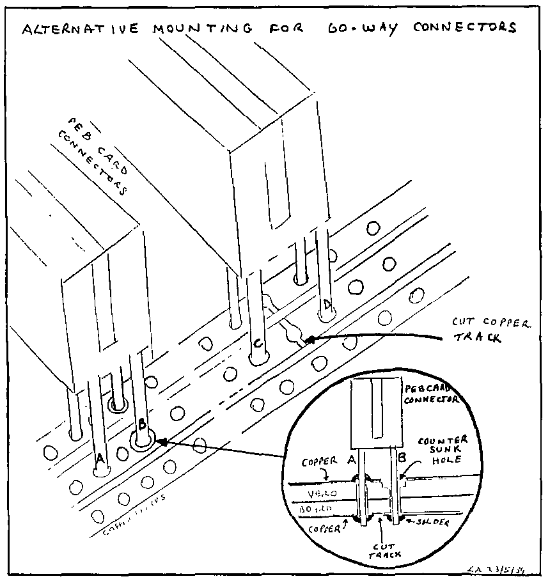
Excessive Voltages

If the unregulated voltages from your power supply exceed those indicated above, they may be reduced by inserting a 5 watt power resistor between the 2200 uF filter capacitors in either the +9 volt and/or the +18 volt supplies. Use the table below as guide:

Measured	Required	Resistor
+12V	+9V	3.3 ohm
+21V	+18V	6.8 ohm

Alternative Construction Method

During the construction stages of this project, it occured to me that the back-to-back veroboards could be used to better advantage by utilising the available tracks to connect the two 60-way connectors in parallel. The idea is illustrated in the diagram below. What we need is a way of connecting point "A" to point "C" while at the same time bypassing pin "B". If we countersink the hole for pin "B" on the top board (without drilling all the way through) we ensure that the pin can pass through the board without making contact with the upper track. This same track is cut between pin "C" and "D"



Similarly the hole for pin "C" is countersunk on the bottom board and the track between pin "A" and pin "B" is cut. If this process is repeated for all 60 tracks, then the two edge connectors will be in parallel simply by soldering all pins except the countersunk ones. If you use this method to connect the 60-way connectors, then you must make a small change to the way that you connect the direct wire links from the 44-way connector. The difference should be obvious if you have followed the above instructions.

Final Note

As a word of encouragement, it took a lot longer to prepare and write this article than to actually build the I/O interface!

(You may also need to remove some jumpers in TI Disk controller cards and TI RS232 cards. These are located close to the edge connector and are wire links which can be cut easily. Their purpose is unclear as they are associated with interrupts which are not used by the TI99/4A, but they cause the console to lock up when used without buffers. GWT) o

MiniMemory
Meandering Memories

by Stephen Shaw, England

Back in Issue 6, now alas out of print, we carried a bit of a glut of articles on Mini Memory usage in TI BASIC, on Graphics, Speech and Sound. In later issues we put these together into a demonstration program, and also covered the use of VDP registers.

After such a long break, with a few changes of owners, and possibly more Mini Memory owners than in 1984, it does not seem a bad idea to repeat a little now, but not all at once. In this issue a brief look at graphics, and if you want more for Mini Memory BASIC usage, perhaps you could drop me a line and vote for one of the other items!!!

Mini Memory adds a few commands to TI BASIC, including the ability to read and write directly to VDP ram using CALL PEEKV and CALL POKEV.

The VDP Ram is used to hold your TI BASIC program, but also contains the screen contents, and colour and character definitions. There is also room for sprites (add that to the list of things to come!!!).

Here is the program, which will run in TI BASIC provided you have the Mini Memory or Editor Assembler module installed. With only slight variation, you could use Extended BASIC, but you will need to use the PEEKV/POKEV machine code utility we have already published, and you will need 32k RAM to put it in. The extra RAM is not required for Mini Memory or Editor Assembler.

```
100 CALL CLEAR
110 PRINT "      ]"
120 PRINT "     ]]"
130 PRINT "    ]]]"
140 REM ] IS FCTN[T]
150 CALL PEEKV(1152,A,B,C,D,E,F,G,H)
160 CALL POKEV(1232,A,B,C,D,E,F,G,H)
170 FOR T=1152 TO 1231
180 CALL PEEKV(T,A,B,C,D,E,F,G,H)
190 CALL POKEV(1512,A,B,C,D,E,F,G,H)
200 NEXT T
210 GOTO 170
220 END
```

Try it!!!

continued on page 35

Jenny's Younger Set

Dear Jenny,

Thank you to you and the group for the certificate. I feel very proud to have received it. I hope that this improved Footy Tab program will help those interested in taking a bet on the football. Hopefully it will work out well for those taking Footy Tab.

Yours faithfully,
Vincent Maker

Dear Jenny,

I have here another improved version of my Footy Tab program. With the football season almost upon us, I think this is good time to send it in. I hope it will come in handy for those taking Footy Tab cards this year on the football.

Vincent Maker

```
100 ON WARNING NEXT
110 REM *******************
120 REM *FOOTY TAB PROGRAM*
130 REM *                 *
140 REM *BY VINCENT MAKER.*
150 REM *                 *
160 REM *     MK.III      *
170 REM *******************
180 CALL CLEAR
190 DISPLAY AT(5,1):"FOOTY TAB PROGRAM."
200 DISPLAY AT(7,1):"MARK III VERSION."
210 DISPLAY AT(9,1):"BY VINCENT MAKER."
220 FOR T=0 TO 500
230 NEXT T
240 CALL CLEAR
250 ONE=0
260 TWO=0
270 INPUT "NAME TEAM ONE:":ONE$
280 PRINT
290 INPUT "NAME TEAM TWO:":TWO$
300 CALL CLEAR
310 REM GROUND CONDITIONS
320 CALL CLEAR
330 PRINT "IS IT A HOME GAME FOR ";ONE$;" (YES/NO):"
340 INPUT GROUND$
350 PRINT "IS IT A HOME GAME FOR TEAM ";TWO$;"
    (YES/NO):"
360 INPUT GROUND2$
370 IF GROUND$="YES" THEN ONE=1
380 IF GROUND2$="YES" THEN TWO=1
390 REM RAIN
400 INPUT "HAS IT BEEN RAINING 24 HOURS PREVIOUS THE
    MATCH(YES/NO)?":WEATHER$
410 IF WEATHER$="NO" THEN 490
420 PRINT
430 PRINT "DOES ";ONE$;" PLAY GOOD WET WEATHER FOOTBALL
    (YES/NO)?"
440 INPUT RAIN$
450 PRINT "DOES ";TWO$;" PLAY GOOD WET WEATHER FOOTBALL
    (YES/NO)?"
460 INPUT RAIN2$
470 IF RAIN$="YES" THEN ONE=ONE+1
480 IF RAIN2$="YES" THEN TWO=TWO+1
490 PRINT
500 REM RECENT FORM
510 PRINT "DID ";ONE$;" HAVE A WIN LAST TIME THEY PLAYED
    (YES/NO)?"
520 INPUT GAME$
530 PRINT "DID ";TWO$;" HAVE A WIN LAST TIME THEY PLAYED
    (YES/NO)?"
540 INPUT GAME2$
550 IF GAME$="YES" THEN ONE=1+ONE
560 IF GAME2$="YES" THEN TWO=TWO+1
570 PRINT
580 REM POINTS ON THE TABLE
590 PRINT "TYPE ";ONE$;" POINTS ON THE TABLE:"
600 INPUT POINTS
610 PRINT "TYPE ";TWO$;" POINTS ON THE TABLE:"
620 INPUT POINTS2
630 IF POINTS>POINTS2 THEN ONE=ONE+1
640 IF POINTS2>POINTS THEN TWO=TWO+1
650 REM INJURIES
660 PRINT
670 PRINT "HOW MANY INJURIES DO ";ONE$;" CARRY AT THE
    MOMENT?"
680 INPUT INJURY
690 PRINT "HOW MANY INJURIES DO ";TWO$;" CARRY AT THE
    MOMENT?";INJURY2
700 INPUT INJURY2
710 IF INJURY>INJURY2 THEN ONE=ONE+1
720 IF INJURY2>INJURY THEN TWO=TWO+1
730 REM PENALTIES
740 PRINT
750 PRINT "HOW MANY PENALTIES DID ";ONE$;" GIVE AWAY IN
    THEIR LAST MATCH?"
760 INPUT PCOUNT
770 PRINT "HOW MANY PENELTIES DID ";TWO$;" GIVE AWAY IN
    THEIR LAST MATCH?"
780 INPUT PCOUNT2
790 IF PCOUNT>PCOUNT2 THEN TWO=TWO+1
800 IF PCOUNT2>PCOUNT THEN ONE=ONE+1
810 REM TRIES SCORED
820 PRINT
830 PRINT "HOW MANY TRIES DID ";ONE$;" SCORE IN THEIR
    LAST MATCH?"
840 INPUT TRY
850 PRINT "HOW MANY TRIES DID ";TWO$;" SCORE IN THEIR
    LAST MATCH?"
860 INPUT TRY2
870 IF TRY>TRY2 THEN ONE=ONE+1
880 IF TRY2>TRY THEN TWO=TWO+1
890 PRINT
900 REM GOALS KICKED
910 PRINT "HOW MANY GOALS DID ";ONE$;" KICK IN THEIR
    LAST MATCH?"
920 INPUT GOAL
930 PRINT "HOW MANY GOALS DID THEY MISS?"
940 INPUT MISS
950 PERCENT=GOAL/GOAL+MISS
960 PRINT "HOW MANY GOALS DID ";TWO$;" KICK IN THEIR
    LAST MATCH?"
970 INPUT GOAL2
980 PRINT "HOW MANY GOALS DID THEY MISS?"
990 INPUT MISS2
1000 PERCENT2=GOAL2/GOAL2+MISS2
1010 IF PERCENT>PERCENT2 THEN ONE=ONE+1
1020 IF PERCENT2>PERCENT THEN TWO=TWO+1
1030 REM MATCHES PLAYED
1040 PRINT
1050 PRINT "HOW MANY MATCHES HAS ";ONE$;" PLAYED IN
     BETWEEN THEIR NEXT SATURDAY/SUNDAY MATCH (EG
     PANASONIC CUP/PLAYOFFS ETC.)?"
1060 INPUT TIRE
1070 PRINT
1080 PRINT "HOW MANY MATCHES HAS ";TWO$;" PLAYED IN
     BETWEEN THEIR NEXT SATURDAY/SUNDAY MATCH (EG
     PANASONIC CUP/PLAYOFFS ETC.)?"
1090 INPUT TIRE2
1100 IF TIRE>TIRE2 THEN TWO=TWO+1
1110 IF TIRE2>TIRE THEN ONE=ONE+1
1120 REM RESULTS
1130 IF ONE>TWO THEN WINNER$=ONE$
1140 IF TWO>ONE THEN WINNER$=TWO$
1150 IF ONE=TWO THEN WINNER$="IT WILL BE VERY CLOSE, IF
     NOT A DRAW.  TAKE THE TEAM WITH THE START."
1160 CALL CLEAR
1170 REM SHOW RESULTS
1180 CALL CLEAR
1190 CALL SCREEN(2)
1200 FOR T=1 TO 8
1210 CALL COLOR(T,2,16)
1220 NEXT T
1230 DISPLAY AT(3,1):"THE TEAM TO TAKE ON THE CARD IS:"
1240 DISPLAY AT(5,1):WINNER$
1250 DISPLAY AT(11,1):"BEST OF LUCK."
1260 PRINT "PICK ANOTHER WINNER (Y/N)?"
1270 CALL KEY(0,H,J)
1280 IF J=0 THEN 1270
1290 IF H=89 THEN 100
1300 END
```

I am sorry, Vincent, that it has taken so long for your contributions to be printed. I hope all your readers will understand, but my friend Geoff had a lot of trouble reading your tapes into the computer. It seemed to be a problem with the fact that your tape was recorded on a stereo player and that makes it very difficult to get the head alignment correct. I do not understand it all but I think that there is an article in this issue by Wade Bowmer which explains it. Eventually Geoff managed to read your tapes and so we now have a bumper issue of my Younger Set for all to enjoy. I think there is still time to use the Footy Tab program. I am not sure which is the latest one. Something for you all to figure out! Thank you Vincent.

Dear Jenny,

I have one program here which is an improved version of one program I sent you in 1987. I hope you like it.

Vincent Maker

```
100 ATTEMPT=0
110 ALPHA=0
120 CALL CLEAR
130 REM IMPROVED GUESS A NUMBER/LETTER BY VINCENT MAKER
140 INPUT "WHICH ONE (NUMBER/LETTER) WOULD YOU LIKE TO
    PLAY?":A$
150 IF A$="NUMBER" THEN 200
160 RANDOMIZE
170 A=INT(90*RND)+65
180 IF A>90 THEN 170
190 GOTO 460
200 CALL CLEAR
210 INPUT "WHAT NUMBER WOULD YOU TO    GUESS OUT OF (EG
    OUT OF 10...)?":GUESS
220 CALL CLEAR
230 RANDOMIZE
240 G=INT(RND*GUESS)
250 INPUT "OK,WHICH NUMBER WOULD YOU   LIKE TO
    GUESS?":ANSWER
260 ATTEMPT=1+ATTEMPT
270 IF ANSWER=GUESS THEN 300
280 IF ANSWER>GUESS THEN 360
290 IF ANSWER<GUESS THEN 410
300 PRINT "CONGRATULATIONS,YOU GOT IT IN ";ATTEMPT;"
    GOES!!"
310 PRINT
320 PRINT
330 INPUT "WOULD YOU LIKE TO GO AGAIN (Y/N)?":D$
340 IF D$="Y" THEN 120
350 GOTO 120
360 PRINT "YOUR GUESS IS TO GREAT, TRY GUESSING
    SMALLER."
370 FOR T=0 TO 500
380 NEXT T
390 CALL CLEAR
400 GOTO 250
410 PRINT "YOUR GUESS IS TO SMALL, TRY GUESSING
    GREATER."
420 FOR T=0 TO 500
430 NEXT T
440 CALL CLEAR
450 GOTO 250
460 PRINT "TO GUESS LETTERS PRESS THE  LETTER YOU WANT
    ON THE       KEYBOARD. THE COMPUTER WILL DO THE
    REST."
470 PRINT "PRESS YOUR GUESS."
480 CALL KEY(0,K,L)
490 IF L=0 THEN 480
500 ALPHA=ALPHA+1
510 IF K=A THEN 550
520 IF K>A THEN 590
530 IF K<A THEN 650
540 IF K>90 THEN 710
550 PRINT "CONGATULATIONS,YOU DID IT   IN ";ALPHA;"
    GOES!!"
560 INPUT "TRY AGAIN (FOR EITHER, Y/N)?":D$
570 IF D$="Y" THEN 100
580 END
590 CALL CLEAR
600 PRINT "TRY GUESSING NEARER THE START OF THE
    ALPHABET."
610 FOR Y=0 TO 500
620 NEXT Y
630 CALL CLEAR
640 GOTO 470
650 CALL CLEAR
660 PRINT "TRY GUESSING CLOSER TO THE  END OF THE
    ALPHABET."
670 FOR U=0 TO 500
680 NEXT U
690 CALL CLEAR
700 GOTO 470
710 PRINT "THAT IS NOT A LETTER.  YOU ARE PENALISED ONE
    GO."
720 FOR T=0 TO 500
730 NEXT T
740 CALL CLEAR
750 GOTO 470
```

Now, we are also fortunate to have some correspondence from our expert on Adventure games.

Dear Jenny,
This is Crocodile Jones. I have received another letter...

Dear Crocodile Jones,
I would like you to give me assistance with Adventure #7, Mystery Fun House. I am being shot when I enter the room with the clay pigeons. How can I stop this?

Charlotte Maer

Dear Charlotte,
Take the sign which says,"OUT OF ORDER" from the fortune machine to the shooting gallery and leave it there. Then go through the grate to the room with clay pigeons.

Crocodile Jones.

Dear Jenny,
I've got another letter here from Christopher. He writes as follows:

Dear Crocodile Jones,
How do you get out of the pit in Mystery Fun House?

Christopher

Dear Christopher,
Take the tramopline from the small room next to shooting gallery back to the pit get on it and jump out of the pit.

Crocodile Jones

Dear Jenny,
Here is another program that I have typed up for the Younger Set. I hope it is OK for it.

Vincent Maker

```
100 REM DECISION MAKER BY VINCENT MAKER
110 CALL CLEAR
120 PRINT "HELLO,THERE!"
130 PRINT
140 PRINT
150 PRINT
160 INPUT "ARE YOU SICK OF MAKING DECISIONS (Y/N)?":A$
170 IF A$="Y" THEN 200
180 PRINT "WELL, I CAN'T HELP YOU THEN.  I'M AFRAID
    WE'LL HAVE TO LEAVE IT HERE.  GOOD-BYE"
190 STOP
200 INPUT "YOU'VE COME TO THE RIGHT PLACE THEN.  ENTER
    YOUR DECISION-":B$
210 RANDOMIZE
220 D=INT(10*RND)
230 IF D>5 THEN 260
240 IF D=5 THEN 310
250 IF D<5 THEN 350
260 CALL CLEAR
270 PRINT "I THINK YOU SHOULD":B$
280 INPUT "ANOTHER DECISION (Y/N)-":SD$
290 IF SD$<>"N" THEN 110
300 STOP
310 PRINT "PLEASE ASK ME TO DECIDE AGAIN"
320 FOR A=0 TO 300
330 NEXT A
```

continued on page 3

The Communicators

Special Interest Group for members of the TEXPAC BBS
by Ross Mudie, SYSOP, May 1989

After the installation of the new modem on the BBS last month it quickly became apparent that many members were having a lot of trouble logging on. At very short notice, just before the last TND was posted out at the end of April, I phoned Geoff Trott and asked if it was possible to include the two modem files in a loose leaf insert in the TND. As you all would have seen last month the files were included. This saved the need for a special mailing, saving the club around $30 and saved me a lot of work. My sincere thanks to TND Editor, Geoff Trott, for his co-operation and assistance.

The major problem that members appear to be having in logging on to the BBS at present is getting used to connecting their modem on line before the BBS modem answers. This ensures that the calling modem tone will be on line before the BBS modem tests for the tone to determine the appropriate data rate. The BBS will accept data rates of 2400/2400, 1200/1200, 1200/75 or 300/300 baud. The format is 8 bits, No parity and one stop bit, (8N1).

Members who have modems capable of VIATEL operation, (1200 baud receive and 75 baud transmit), may not have a suitable terminal program to support the split baud rate. There is currently an implanted version of of Mass Transfer Version 4.1 available in the program download area and there is a documentation file in the News menu. The program is suitable for disk or cassette loading under Extended BASIC. A special EPROM will be required in the user's RS232 to download programs when 1200/75 is in use.

The BBS tests for presence of Data Carrier at the modem and if the carrier is lost, at any time, the BBS will reset to the idle state. The reset is immediate in all areas of operation, except when the menu is being rebuilt after a Sub-Editor file has been uploaded or after a User program/file upload. In all the normal areas of program download and file reading, the user may hang up at any time without ill effect to the BBS operation.

I was really pleased to see Geoff Trott's "terrible predicament", (page 1 May 1989 TND), of too many contributions. Reading some of the overseas and other User group magazines, we have an almost unique situation. Please do not stop producing information for the BBS or the TND since it is far better for the Editor and SYSOP to have too much, rather than not enough material for the magazine and BBS. If you have been doing something special or different with your TI99/4A computer why not tell your fellow members about it? You may stimulate the interest of a fellow member or find that someone else has been doing similar things with their computer and together you may work towards better things. o

From the Bulletin Board

MAIL TO : ALL
MAIL FROM : PAINEY

***** FOR SALE *****

One time special sale. One only IBM Compatible AT style Portable computer. Zero wait state 10 Mhz (running at 13Mhz), 640K memory, 1.2 Mbyte Floppy, choice of 20Mbyte or 40Mbyte hard disk. Liquid crystal display with ability to connect external monitor and keyboard.

40 Mbyte configuration $1800
20 Mbyte configuration $1500
Phone John Paine (02)819 7200 work or home (after 7.00 pm, please) (02)625 6318.

MAIL TO : ALL
MAIL FROM : SCIFI-BBS

Gary Christensen of the Brisbane user Group tells me that NTSC version Geneves can be had very quickly, by those wanting them, as they are "on-shelf" and ready to go. As for PAL-D versions, that is another story. Myarc are not known for their service reputation. Why people may like to opt for NTSC is that a proper monitor can handle the RGB without modification, so I believe. Hope that helps.
Regards, Greg.

MAIL TO : ALL
MAIL FROM : LARRY

I have a new BIZTEL 2400 Modem. Computer Fix sell them for $299 each.

MAIL TO : ALL
MAIL FROM : DOBELL

For those who wish to access Viatel free for a limited time here is how to do it ring 01955 when the opening screen appears enter user number 4444444444 (10 digits). Then, when asked for password type 4444 (4 digits) and you are in. You will need Mass Transfer or the club Viatel software.

MAIL TO : ALL
MAIL FROM : LARRY

Tips for MAZE of GROG

Get needle before leaving frame 1. Use teleports frame 2 and 3. Do not go to frame 4 until you have enough energy to unlock all the doors. You need about 2500 in energy to have a chance at getting PENTOLOPE. Teleport in frame 3 on the top right takes you back to frame 1 and new level. If you do not get needle in frame 1 GROG will jump on you. END OF GAME.
Bye for now, Larry.

MAIL TO : ALL
MAIL FROM : REDDRAGON

********Wanted!!!*********

Does anyone out there have an RGB mutisync-type monitor that they wish to sell. If so leave mail for REDDRAGON.

--- o

Programs from the TI*MES Library

by Stephen Shaw, England

>JOHN SEAGER1: A UK programmer offers: GOLF- an Extended BASIC game and an enhanced Extended BASIC game with machine code links, with source code. An Extended BASIC character designer and a much enhanced version with machine code links with source code. The most useful character designer I have! And a suite of four programs to manipulate your Extended BASIC programs (amends DV163 files), intelligent block move of lines, extractor to save a section of a program, delete to delete a section of a program, and renumber to renumber a portion of a program. Slow but useful.

>JP HODDIE. Games from the Master. Machine code Asteroids and Snake programs plus Extended BASIC: Fish, Frog, Spacewar, for 2 players, Kong (?-misnamed!). Disk filled with a machine code version of Space Station Pheta with Extended BASIC loader and documentation. Can be very fast indeed!

>LINKER by RA GREEN, V2. This program changes DF80 object code to memory image format, with options on locating the image, a compact output, and a clever way to resolve unresolved references. A library file is supplied with common references and the program incorporates just what it needs into the image. Also see the next disk below. This disk also contains an alternative TI Writer modified by

continued on page 18

Mass-Transfer Version 3.5
With MXT file transfers

by Stuart Olson, IL USA

If you are using this program, and have not yet paid for it, please send $10 check or money order to the author for his work.

I am allowing you to try this software before you decide to buy it. As such, I am trusting you to pay for it if you should decide to keep it. If you do not like it, please pass it along to other users, as they may find a need for this program. Good response from you will encourage programmers like myself to provide you with software at a reasonable price and without copy protection. Like they say, you can start paying us now or pay a software dealer 3 times the price for their programs.

Stuart Olson, SYSOP: TI-North (312)587 3490
25322 W. Wayside Place
Lake Villa, ILL. 60046

MASS-TRANSFER is a terminal emulator program written in 9900 assembly language. Through its menu driven screens, it supplies the user with a variety of options with which to enhance the telecommunications of the TI9/4A computer.

SYSTEM REQUIREMENTS

The minimum system necessary for operation is a TI99/4A console, 32K Memory Expansion, RS232 card, disk drive, and a modem. Either the Editor Assembler or Extended BASIC cartridge may be used.

GENERAL INFORMATION

This software employs a unique feature provided by the TI-99/4A computer. The RS232 card contains its own programming inside a special Device Service Routine (DSR) ROM. This ROM normally handles the computers file management. It also allows for the use of Circular Interrupt Buffer (CIB) operation. This routine interrupts the running program each time a character is received and places that character into a special buffer in memory. This program periodically checks the CIB and upon finding new characters, processes them as either control characters or data. The CIB writes over itself if more than 255 characters are received and not processed. This type of programming results in an interrupt driven input for the computer, allowing the user to leave the terminal mode and return to the main menu for short periods of time without any data loss.

MASS-TRANSFER maintains its own user managed telephone directory. The accompanying Extended BASIC program allows the user to load the phone directory with up to 20 of the most frequently called numbers. The directory can be easily updated with the program's editing feature. The desired phone number can be accessed via the auto-dial feature. This auto-dialing can only be accessed if a smart modem with auto-dial capacity is being used. Each of the 20 listings allow the user to send up to 50 characters to the modem. This allows the user to send special modem control codes prior to the actual dialing of the desired phone number.

Note: In order to use the auto-dial phone directory option, the Extended BASIC program that creates the directory should first be used. To do this, load and run the following program OLD DSK1.PHONEMAKE. If this program is being run for the very first time, then select option 1 from the menu, which will create a blank directory on the disk you have in drive #1. This program is also menu driven. Follow the screen prompts to load the directory with your phone numbers.

INTRODUCING "MXT" (Multiple Xmodem Transfer)

The MXT file transfer option is an all new concept for the TI99/4A and its growing Xmodem file transfer capabilities . The MXT supervisor routine allows the user to select any number of files from a disk, and send them without the task of typing in numerous filenames and other associated key combinations as used by other Xmodem transfer programs. If used with a smart modem, the program will even disconnect itself from the phone line after the transfer is complete or should more than 10 retries be encountered on any one record.

If the MXT system is used on both sending and receiving computers, the modified Xmodem transfer is automatic, allowing both users to walk away from the computer, returning later to turn off the equipment.

MXT is capable of sending your selected disk files to other Xmodem terminal emulators also. The receiving system will still have to type in filenames, however the MXT supervisor sends the computer the proper filename for the file being sent. This allows MXT to work with programs such as Fast-Term and 4A/Talk.

MXT uses a modified file header to convey the necessary information for the receiving MXT system, including the proper filename and status as to any more files to be sent after the present one is finished. Although the file header is modified, it still works with the other Xmodem terminal programs. Since the other Xmodem programs do not send the modified file header, MXT can not be used for receiving files from these other programs. Instead, just select the regular Xmodem transfer from the menu.

ERRORS

Mass-Transfer uses on screen I/O error messages if problems are encountered while accessing a peripheral. During the Xmodem or MXT routines, the error codes are simply replaced with a disk error message. For your convenience, the error codes are listed below.

Code	Meaning
00	Bad device name
01	Device is write protected
02	Bad open attribute such as incorrect file type
03	Illegal operation; ie., an operation not supported on the peripheral or a conflict with the open attributes
04	Out of table or buffer space on the device
05	Attempt to read past end of file
06	Device error. Covers all hard device errors
07	File error such as program/data file mismatch

SYSTEM DEFAULTS

Upon entering the program, the following defaults are in affect:

RS232 port #1
300 baud rate
8 bit character length (terminal mode masked to 7)
1 stop bit
No parity check
Auto Hangup Off
Local echo Off
Remote echo Off
Buffer capture On
Log file Off
Linefeed Off

For Hayes and Novation modem owners, the above defaults are compatible with your modems defaults.

CONTROL and FUNCTION KEYS

The following control and function keys are active:

CTRL[G]	Bell	- Produces a beep
CTRL[H]	BS	- Move cursor one space left, deleting the previous character
CTRL[J]	LF	- Scrolls the display screen one line
CTRL[L]	FF	- Clears the screen and homes cursor
CTRL[/]		- Causes the program to generate a break character

CTRL[=]		- Changes screen colour.
FCTN[S]	BS	- Left arrow, provides same function as CTRL[H]
FCTN[X]	LF	- Down arrow, provides same function as CTRL[J]
FCTN[1]		- Changes text colour.
FCTN[4]		- CLEAR allows the user to abort most functions
FCTN[7]		- AID exits the Terminal mode and returns to the main menu screen
FCTN[=]		- Prompts you for exiting the program
ENTER	CR	- Positions cursor at the beginning of the line.

LOADING THE PROGRAM

Editor Assembler

Select the RUN PROGRAM FILE option and enter filename as:
DSK1.MASS

Extended BASIC module

An Extended BASIC program image loader is included. Just type in: RUN"DSK1.LOAD"

MAIN MENU INSTRUCTIONS

<R> Reconfigure I/O port

This option allows the user to select communication parameters other than those listed in the program defaults section. To reconfigure your system for a different setup, select R from the menu. The program will take you to the Configuration Setup screen. You will then be able to enter the desired communication parameters.

Note: Both file transfers use the 8 bit character length with no parity. If the defaults have been changed, ensure the program is reconfigured to these parameters.

<M> Multiple Xmodem Transfer (MXT)

This option allows the user to send and receive an unlimited number of disk files without having to attend to the computer during the transfers.

When you select the send option, the program will catalog the disk and display the directory on the monitor, 10 files at a time. To select a file for transfer, position the cursor next to the desired filename and press the <ENTER> key. You may use the up and down arrow keys to move the cursor. To delete a previously selected file, position the cursor in front of the filename, and press FCTN[1] (DEL). To proceed on to the next screen of directory entries, press FCTN[6] (PROC'D). When finished with the last directory screen, press FCTN[6] one more time and the MXT transfer will begin.

To abort during a transfer, press FCTN[4] (CLEAR). The abort will be cancel the current file being transferred and any other files still remaining in the supervisor list.

When using MXT to send files, the sending computer will pause for 7 seconds between files. This is so that the receiving system (if non-MXT capable) can type in the next filename as it appears on his monitor. If you wish to cancel the 7 second pause, just press the <ENTER> key.

Note: If using MXT in the receive mode, use a clean disk. The MXT supervisor routine will create the filenames on your disk just as they were labeled on the sending disk. If you have files on your disk, they could be erased and written over with the newly received file if the filenames are the same.

<A> Auto Hangup Selection

This option allows the user to select the protocol desired for the auto hangup routine. The user should select the protocol that matches the smart modem presently being used.

MASS-TRANSFER uses the smart modem defaults to execute the auto hangup (ie., modem control character, guard time, etc). It is necessary for your modem to be set at these defaults.

The auto hangup will be executed when the MXT transfer is either successfully completed or aborted. This feature allows the user to leave the computer unattended for a period of time, and return later to see the status of the disk transfer.

<C> Clear download buffer

This option clears the 11.6K buffer. Any data previously received will be lost. After the buffer has cleared, the program returns to the main menu.

<E> Echo remote status = OFF
monitor status = OFF

This option allows the user to select the necessary echo depending upon the requirements. The user has control over both the remote and monitor echo. Varied use of this option will permit the user to interface with almost any other computer, regardless of what the remote computer provides or requires in the area of character echo.

To change either the remote or monitor echo status, press the E key. A four choice menu appears, allowing the user to turn on or off each echo. A beep will be heard, when a valid choice is selected from the menu. When all choices have been made, press ENTER. The program will return to the main menu, with the new echo status displayed.

<1> Remote echo - ON

Echo all received data back to the sending computer.

<2> Remote echo - OFF

Normally used when communicating with BBSs .

<3> Monitor echo - ON

This mode shows what you are typing but does not echo the received characters back to the remote computer.

<4> Monitor echo - OFF

Suppresses the echo of your typed characters to your monitor.

Note: The above combinations of echoes gives the user an excellent choice to use under various situations. However, one should keep in mind the following idea. If both computers are using an echo ON mode, the program usually starts typing a string of feedback characters since both computers are re-sending the same character over and over. If this should occur, it can be stopped by returning to the menu and changing the remote echo to off. Auto-dialing should not be done with the remote echo on. This echoing of modem responses usually causes the modem to misdial or lock up.

<U> Upload DIS/VAR 80 file from disk

This option allows the user to send a display variable 80 file from a disk drive. Upon selection of this option, the screen will prompt you for the filename. If the desired file was called REPORT and located on a disk in drive #2, then you would enter DSK2.REPORT If you wish to add a carriage return to the end of each outgoing line, enter a Y at the prompt. Use of this option depends upon the requirements of the receiving computer and also what editor was used to create the file. The abort message and single line prompt will be displayed. If it is desired to send the file, one line at a time, press the Y key. If single line transfer was selected, you must press the Y key to send each line of data. (Single line transfers expect the data you send to be echoed back, such as if you were accessing a BBS. You should only use single line transfers when the receiving computer is echoing your data.) If you press N, lines of data are sent one after another, with no pause in between. While the single line transfer is in progress, the screen changes to split screen scroll. The top 8 lines will scroll and display the outgoing and incoming data, allowing the rest of the screen to display the prompts. This type of single line transfer allows the user to easily leave messages on a BBS. If single line transfer was not selected, the program does not expect any incoming characters. To abort the transfer, (from either mode), press FCTN[4] (CLEAR). This option returns to the terminal mode.

The program uses the XON/XOFF protocol when uploading a file if "N" was given as the response to the single line prompt.

<A> Auto-Dial from directory

This option enables the auto-dial feature of MASS-TRANSFER. Upon selection of this option, the screen will clear and disk drive #1 will access the phone directory. The directory will appear on the monitor, along with the prompt "Rapid Dial (Y/N)". The rapid dial feature allows a Smart-Cat modem to redial a busy number 1.5 seconds after the modem detects a busy signal. (The Hayes modems require a longer time.) The rapid dial will continue until the user aborts the option or the modem finally rings the desired number. If rapid dialing is not desired, press N. The next prompt will appear, asking you to select the desired phone number. Press the key (A-T) that corresponds to the desired phone number. The screen will clear and display the number dialing message. The smart-modem will switch into the dial mode, and dial the number. If you wish to cancel the number after it is dialed, consult your modems owners manual. If the dialed number is busy for several minutes, the screen will blank. Pressing a key will unblank the screen but will also abort the dialing process. To avoid this, return to the menu, (FCTN[7]) and then press ENTER. The program will alert you when connection has been made, by producing an audible beep, provided your monitor has audio capabilities. Upon completion of this option, the program is in the terminal mode, ready to communicate.

Note: The rapid dial option has been used on some non-Hayes manufactured modems with satisfactory results. Since one can never be sure if the so called Hayes compatible modem is 100% identical in operation, this option may not work on your modem.

<D> Dump download buffer

This option copies the 11.6K buffer to a storage device. This device can be any legal device name that supports a maximum file length of 80 characters, except cassette. Upon selection of this option, enter the appropriate filename, such as PIO, DSK1.SAVEFILE, RS232/2.BA=4800.DA=8, etc. You can also abort the dump by following the screen prompts. In order to allow the same buffer contents to be saved to several devices, the buffer is not cleared after the dump routine.

Note: This buffer is shared with the LOG file. However, even if you have the LOG file on, you may still dump the buffer contents to any device you desire.

Note: If you elect to dump the buffer back to the modem, you will have to return to the main menu and reconfigure the RS232 port attached to the modem after the dump. If this is not done, you may have problems sending or receiving any further data.

<S> Set up Log File status = OFF

This option allows you to assign a peripheral device to accept the contents of the 11.6K buffer when the buffer fills. You will be asked for a filename. Most users would use either the printer or a disk in order to keep a copy of the buffer contents for later review.

Once the log file has been toggled on, you may turn it off by simply pressing the "S" key from the main menu. If there is data in the buffer, it will be sent to the device you had named as the log file before it closes the file. If you do not desire to have the buffer contents sent to the log file, then clear the buffer before closing the log file. Likewise, if you should happen to quit the program while the log file is open, you will be asked if you wish to save the buffer contents.

The LOG file uses an append format which allows the computer to add on to the previous file contents, if any. This means that a file on disk will have new contents added to it each time the buffer fills and dumps to the disk. The LOG file will not write over an existing file on disk.

Note: The program sends a XOFF (CTRL[S]) to the other computer when spooling the buffer contents to the assigned peripheral. After the buffer is emptied, the program sends an XON (CTRL[Q]) to allow the sending computer to continue. This is commonly referred to as XON/XOFF protocol.

<V> View buffer contents

This option allows the user to view the contents of the 11.6K buffer without destroying it. As an added feature, any portion of the buffer can be saved to any peripheral via the screen dump option. This allows the user to read or save a message without having to dump the entire buffer. Scrolling of the displayed text is controlled by the space bar. Press the space bar once to stop the scrolling and again to start it. Pressing FCTN[4] (CLEAR) will abort the viewing option and return to the main menu. To use the screen dump option, allow the screen to scroll to the desired text. Stop the scroll and press the "P" key to start the screen dump. You will be prompted for the output device name. After the screen dump is completed, you will be returned to the viewed screen. You may continue the viewing, printing more screens if necessary, or abort back to the main menu with the FCTN[4] keys.

<B> Buffer capture status = ON

This option allows the user to control the flow of data into the 11.6K buffer. When the status indicates ON, all data, except D/V 80 uploads and Xmodem transfers, are stored in the buffer. By pressing "B", the buffer will toggle between ON and OFF.

<L> Linefeed toggle status = OFF

This option allows the program to provide a linefeed after each received carriage return. Pressing the L key will toggle the option between ON and OFF. These linefeeds are not stored in the buffer, since they are not received data, but rather generated locally by the program. This option is very handy when accessing a BBS that does not supply linefeeds after carriage returns or when receiving an uploaded file from another computer. When toggled ON, it also causes double spacing of the buffer contents when using the view option.

<X> Xmodem File Transfer

This option allows the user to transfer a file to another computer using the Xmodem protocol. As with the other Xmodem file transfer programs available for the TI99/4A computer, this routine provides the user with compatibility with these other programs. It supports both CRC and checksum error detection.

cont p16

Tips from the Tigercub #32

by Jim Peterson, Tigercub Software, USA
Copyright 1986
Tigercub Software
156 Collingwood Ave.
Columbus, OH 43213

Distributed by Tigercub Software to TI99/4A Users Groups for promotional purposes and in exchange for their newsletters. May be reprinted by non-profit users groups, with credit to Tigercub Software.

Over 130 original programs in BASIC and Extended BASIC, available on cassette or disk, only $3 each plus $1.50 per order for postage and packing. Entertainment, education, programmer's utilities.

Descriptive catalog $1, deductable from your first order.

Tips from The Tigercub, a full disk containing the complete contents of this newsletter Nos. 1 through 14, 50 original programs and files, just $15 postpaid.

Tips from the Tigercub volume 2, another disk full, complete contents of Nos. 15 through 24, over 60 files and programs, also just $15 postpaid. Or, both for $27 postpaid.

Nuts & Bolts (No. 1), a full disk of 100 Extended BASIC utility subprograms in merge format, ready to merge into your own programs. Plus the Tigercub Menuloader, a tutorial on using subprograms, and 5 pages of documentation with an example of the use of each subprogram. All for just $19.95 postpaid.

Nuts & Bolts No. 2, another full disk of 108 utility subprograms in merge format, all new and fully compatible with the last, and with 10 pages of documentation and examples. Also $19.95 postpaid, or both Nuts & Bolts disks for $37 postpaid.

Tigercub Full Disk Collections, just $12 postpaid! Each of these contains either 5 or 6 of my regular $3 catalog programs, and the remaining disk space has been filled with some of the best public domain programs of the same category. I am not selling public domain programs. My own programs on these disks are greatly discounted from their usual price, and the public domain is a free bonus!

TIGERCUB'S BEST PROGRAMMING TUTOR
PROGRAMMER'S UTILITIES BRAIN GAMES BRAIN TEASERS
BRAIN BUSTERS! MANEUVERING GAMES ACTION GAMES
REFLEX AND CONCENTRATION TWO-PLAYER GAMES
KID'S GAMES MORE GAMES WORD GAMES ELEMENTARY MATH
MIDDLE/HIGH SCHOOL MATH VOCABULARY AND READING
MUSICAL EDUCATION KALEIDOSCOPES AND DISPLAYS

For descriptions of these send a dollar for my catalog!

I have found a bug in the Tigercub Menuloader V.#5 which will not let you print a disk catalog if the disk contains the maximum 127 files. This should fix it.

```
340 I=I+1 :: IF I>127 THEN K=X :: GOTO 430
520 DISPLAY AT(X+5,12)SIZE(12):" #?" :: ACCEPT
    AT(X+5,15)SIZE(3)VALIDATE(DIGIT):KD :: IF KD<1 OR
    KD>NN THEN 520
```

I think that all program listings should be printed in 28-column format, exactly as they appear on the screen; it makes it so much easier to key them in without errors. I combined parts of two of my programs to make the following. It is written for the Gemini 10X but the lines of printer control codes are annotated to help others make adjustments.

```
100 DIM K$(240):: LN=100 :: DISPLAY AT(3,4)ERASE
    ALL:"TIGERCUB PROGLISTER": :" Will convert a
    program":"listing to 28-column format,"
110 DISPLAY AT(7,1):"exactly as it appears on
    the":"screen, and print it in 4":"columns."
120 DISPLAY AT(11,1):" Program must be RESequenced":"and
    LISTed to disk by":"RES (enter)":"LIST
    DSK1.(filename) (Enter)"
130 DISPLAY AT(18,1):"Filename? DSK" :: ACCEPT AT(18,14)
    BEEP:F$
140 OPEN #1:"DSK"&F$,DISPLAY ,VARIABLE 80,INPUT
150 IF EOF(1)=1 THEN 260 :: LINPUT #1:A$
160 IF LEN(A$)<80 THEN LN=LN+10 :: GOTO 210
170 LINPUT #1:B$ :: IF POS(B$,STR$(LN),1)=1 THEN FLAG=1
    :: LN=LN+10 :: GOTO 210
180 A$=A$&B$ :: IF LEN(A$)<160 THEN LN=LN+10 :: GOTO 210
190 LINPUT #1:B$ :: IF POS(B$,STR$(LN),1)=1 THEN FLAG=1
    :: LN=LN+10 :: GOTO 210
200 A$=A$&B$ :: LN=LN+10
210 S=1
220 L$=SEG$(A$,S,28)
230 IF L$<>"" THEN 240 :: IF FLAG=1 THEN FLAG=0 :: A$=B$
    :: GOTO 160 :: ELSE GOTO 150
240 X=X+1 :: K$(X)=L$ :: S=S+28 :: IF X=240 THEN 250 ::
    GOTO 220
250 X=0 :: CALL PRINTER(K$()):: GOTO 220
260 CLOSE #1 :: FOR J=X+1 TO 240 :: K$(J)="" :: NEXT J
    :: CALL PRINTER(K$()):: PRINT #2:CHR$(12):: END
270 SUB PRINTER(B$()):: IF F=1 THEN 340 :: F=1
280 OPEN #2:"PIO.LF",VARIABLE 132 :: PRINT
    #2:CHR$(15);CHR$(27);"N";CHR$(6);!condensed print
    and perforation skip
290 PRINT #2:CHR$(27);"G";! - double-struck printing,
    optional
300 PRINT #2:CHR$(27);CHR$(42);CHR$(0);!download normal
    characters - required if lines 310-330 are used
310 PRINT #2:CHR$(27);CHR$(42);CHR$(1);CHR$(48);CHR$(0);
    CHR$(64);CHR$(30);CHR$(96);CHR$(17);CHR$(72);
    CHR$(5);CHR$(66);CHR$(61);CHR$(0);!slash the zero -
    optional
320 PRINT #2:CHR$(27);CHR$(42);CHR$(1);CHR$(42);CHR$(0);
    CHR$(8);CHR$(34);CHR$(8);CHR$(0);CHR$(62);
    CHR$(0);CHR$(8);CHR$(34);CHR$(8);!broaden the
    asterisk - optional
330 PRINT #2:CHR$(27);CHR$(36);CHR$(1);!activate
    redefined characters - required if lines 310-320 are
    used
340 FOR C=1 TO 60 :: IF B$(C)="" THEN 360 ::
    PRINT #2:TAB(10);B$(C);TAB(41);B$(C+60);TAB(72);
    B$(C+120);TAB(103);B$(C+180);CHR$(10)
350 NEXT C
360 SUBEND
```

I had trouble in debugging that program because printing the control codes gave me unwanted line feeds, and using semicolons to prevent line feeds will interfere with tabs in the first line of text. An article by Art Byers in the Central Westchester UG newsletter gave me the solution. Suppress all the line feeds by opening the printer with PIO.LF, and put them back in where you need them with CHR$(10)!

We have not had a random music player in a long time. This one is called ECHO but I do not know where it came from.

```
100 RANDOMIZE :: DEF X=INT(7*RND):: FOR B=0 TO 6 ::
    A(B)=VAL(SEG$("247262294330349392440",3*(B+1)-2,3))
    :: NEXT B :: B,C,D=X
110 CALL SOUND(-900,A(B),0,A(C),9,A(D),19):: D=C :: C=B
    :: B=X :: GOTO 110
```

Sound effects - thanks to Greg Healy in the Edmonton User Group newsletter:

```
100 CALL INIT
110 FOR J=2000 TO 2300 STEP 10 :: CALL LOAD(-31568,J)::
    NEXT J
```

To go directly from Extended BASIC to console BASIC - thanks to Greg Healy in the Edmonton User Group newsletter:

CALL INIT :: CALL LOAD(-31962,8787) Enter. Ignore the error message. Type NEW and Enter.
> TI BASIC READY

This routine will read a file of 28-character records and scroll them up the lower half of the screen without disturbing the upper half.

```
100 DISPLAY AT(12,1)ERASE ALL:"FILENAME? DSK" ::
    ACCEPT AT(12,14)BEEP:F$ :: CALL CLEAR
111 OPEN #1:"DSK"&F$,INPUT
112 DIM M$(480)
113 X=X+1 :: LINPUT #1:M$(X)
120 DISPLAY AT(24,1):M$(X)
125 R=24
130 FOR T=X-1 TO 1 STEP -1 :: IF R>13 THEN R=R-1 ::
    DISPLAY AT(R,1):M$(T)
140 NEXT T :: IF EOF(1)<>1 THEN 113 ELSE CLOSE #1
```

```
10 !ONE-LINE MORTGAGE PAYMENT CALCULATOR BY SAM MORABITO
100 CALL CLEAR :: INPUT "ENTER P,R,N WHERE P=AMOUNT,
    R=RATE, N=YEARS":P,R,N ::
    PRINT "$";INT(100*(P*R/1200)/
    (1-1/(1+R/1200)^(12*N))+.5)/100;"PER MONTH"
```

A number always prints out with a blank space before and after it (except that a negative number is preceded by -). This is not always desirable when formatting a screen or printout. The solution is to change the number to a string by using STR$:

```
100 CALL CLEAR
110 PRINT "  MULTIPLICATION TABLES": :
120 FOR J=1 TO 9
130 FOR K=1 TO 9
140 PRINT TAB(3*K-2);STR$(J*K);
150 NEXT K
160 PRINT : :
170 NEXT J
```

Regarding the CHECKER program in Tips #31, I should have mentioned that the two programs to be compared must first be LISTed to one disk by LIST "DSK1.(filename) using a different filename for each.

We are still finding new ways to skin the kitty. In Tips #26 I listed three algorithms to alternate between the two joysticks.
Rick Humburg sent me another which is the simplest and fastest of all:

```
100 Z=2
110 Z=3-Z :: CALL JOYST(Z,X,Y).......and back to 110!
```

Here are some more dark secrets Texas Instruments did not tell us. The User's Reference Guide claims that the computer can produce frequencies up to 44733 Hz, "well above human hearing limits", but then admits "the actual frequency produced may vary from 0 to 10 percent depending on the frequency." According to Jim Hindley, the highest frequency actually produced is 37287 (which is certainly not above the hearing range of some humans, but neither is 44733!), and the maximum error rate far exceeds 10 % because any frequency you call for from 31953 to 43733 ends up as exactly 37287! Not to worry, the frequencies in the normal range of music are accurate enough and your TV speaker probably cannot reproduce frequencies above 20000 anyway.

And did you know that TI really gave us only 15 volumes, not 30? Listen and count them :

```
100 FOR V=0 TO 29 STEP 2
110 CALL SOUND(1000,500,V)
120 CALL SOUND(1000,500,V+11)
130 FOR D=1 TO 500
140 NEXT D
150 NEXT V
```

And the duration values are just as inaccurate. Experimenting with a series of 8 CALL SOUNDs in a loop repeated 100 times, I found that execution time was 40 seconds for any duration between 1 and 49, or a negative duration; 54 seconds for any duration between 50 and 66; 67 seconds between 67 and 83; 80 seconds between 84 and 99; 94 between 100-116; 106 between 117-133....!

I guess I have been neglecting those who do not have the Extended BASIC module, so:

```
100 CALL SCREEN(16)
110 CALL CLEAR
120 PRINT TAB(8);"GREENSLEEVES": : : : : : : : : : : : :
    :"programmed by Jim Peterson"
130 DIM S(15)
140 FOR N=1 TO 12
150 READ S(N)
160 NEXT N
170 M$="421800995ABDC324E7DBA518669918240O425A00DBC35
    A66A5243C7E81994200A57E66BD3CA5423C187E423CBD5A
    810099FFC3"
180 RANDOMIZE
190 FOR R=1 TO 12
200 CALL COLOR(R+1,1,1)
210 CALL CHAR(32+8*R,CH$&CH$)
220 FOR T=R TO 25-R
230 CALL HCHAR(T,R,32+8*R,34-2*R)
240 NEXT T
250 NEXT R
260 CALL SCREEN(2)
270 FOR R=1 TO 12
280 CALL COLOR(R+1,R+2,1)
290 CH$=SEG$(M$,2*INT(47*RND+1)-1,8)
300 CALL CHAR(32+8*R,CH$&CH$)
310 NEXT R
320 DATA 247,277,294,311,330,370,392,440,494,523,554,587
330 DATA 2,5,5,4,7,5,2,8,5,3,9,5,1,10,1,2,9,3,4,8,3,2,6,
    3,3,3,1,1,5,3
340 DATA 2,6,1,4,7,5,3,5,2,1,4,2,2,5,2,4,6,1,2,4,4,4,1,1
350 DATA 2,5,1,4,7,5,2,8,5,3,9,5,1,10,5,2,9,5
360 DATA 4,8,3,2,6,3,3,3,3,1,5,3,2,6,3,3,7,5,1,6,2,2,5,1
370 DATA 3,4,1,1,2,2,2,4,1,4,5,1,2,1,5,6,5,1
380 DATA 2,12,9,2,12,7,2,12,3,3,12,12,1,11,9,2,9,7
390 DATA 4,8,6,2,6,3,3,3,3,1,5,5,2,6,3,4,7,5,2,5,3
400 DATA 3,5,5,1,4,4,2,5,5,4,6,1,2,4,1,6,1,1
410 DATA 6,12,9,3,9,12,1,11,8,2,9,7,4,8,6,2,6,3,3,3,3
420 DATA 1,5,3,2,6,2,3,7,5,1,6,6,2,5,5,3,4,1,1,2,2,2,4,
    4,6,5,1,1,1,5,7,5,1
430 FOR J=1 TO 223 STEP 3
440 READ T,A,B
450 GOSUB 530
460 FOR TT=1 TO T
470 CALL SOUND(-999,S(A),0,S(B),7)
480 NEXT TT
490 NEXT J
491 FOR V=0 TO 20
492 CALL SOUND(-999,S(A),V,S(B),V+7)
493 NEXT V
500 CALL SCREEN(INT(14*RND+2))
510 RESTORE 330
520 GOTO 270
530 CALL COLOR(A+1,INT(14*RND+2),1)
540 CALL COLOR(B+1,INT(14*RND+2),1)
550 RETURN
```

```
1 !from 9 T 9 UG newsletter August 85
100 PRINT """Hello"" said TI "
110 PRINT "Press ""ENTER"" to continue"
```

If you bite the hand that feeds you, you will go hungry tomorrow. Do not be a pirate!

Memory full to busting Jim Peterson

continued from page 14

The filename for the transfer is assigned after entering this routine. Retry, record, and sector (current and total) indicators are updated constantly on the monitor, along with disk access errors. Transfer status is indicated upon completion or aborting of the file transfer. You may use FCTN[4] to abort the transfer.

I would like to thank the many users of TI-North BBS who provided me with many ideas on how to improve my program. I would also like to thank my wife, who I have made a computer widow during the countless hours spent working on this program.

Review of TI Runner Level Editor

by Stephen Shaw, England

Advertised for US$18.95, overseas postage is not indicated.

EB Software, 905 West Middlefield #963, Mountain View, CA, USA, 94043.

TI Runner is a well known popular game on the TI99/4A, with versions available on disk (50 screens), tape (40 screens) and module (25 screens), the disk version from EB Software (or by virtue of paid licence, from this group), the module version by Databiotics (called Star Runner) and the cassette version, for Extended BASIC plus 32K RAM, from licensed user groups (including our own group).

This Editor allows users to create their own screens for use with the Disk version, creating merry mazes with the three types of brick; those you can make holes in, those you cannot, and those you fall through! And of course the ladders, including the ladders which only appear when you have collected all the treasures you need to.

Supplied on disk, and requiring Extended BASIC plus 32K RAM, the program was supplied with 9 pages of A4 documentation, stapled at top left. The disk contains the original 50 levels in one file, and a further 40 advanced levels in another file. These are provided for you to edit if you wish. You can of course create a second games disk by copying your original TI Runner program to a fresh disk, and copying the ADVANCED file to it, renaming it LEVEL28.

LE (TI Runner Level Editor) allows you to create entirely new files of screens, or to copy individual screens into a new file in a new order, as well as editing existing screens.

This product is of interest to any TI Runner player who has made it to screen 50 (anyone?) or who finds it quite impossible and would like to edit a problem screen to make it easier!

When using any product of this nature it really is wise to make up those suggested back up copies of your disks, and edit only the back up copies, never a master copy!

The documentation and program caused me a few sleepless nights, but most of the difficulties have now been solved and can be passed on to you. The game disk received was not marked as a beta test version, but was received as a final commercial offering. EB Software have received an advance copy of these corrections and it is to be hoped that subsequent sales will contain them!

To deal with the three sections of the program in turn.

1. Level Management. This is where you edit a screen, add a screen, copy a screen, or delete a screen, all very straight forward!

However if you choose to add or edit a level, the documentation requires a little elucidation:

The first selection when you choose to add/edit a screen is Change Colour. This selection must be chosen also if you wish to amend (or see) the number of treasures that must be collected to complete a screen. You will need to know how many treasures the screen needs, so use this option!

Changing colours is a little odd. Instead of entering 1 to 16 to input colours, the program requires you to enter CHR$(48) to CHR$(63), that is, 0 to ?, which is indeed displayed on screen, but is not a normal method of thinking of colours. This can be amended to a more normal operation with very little code, and is merely an indication of poor design.

The documentation lists a number of keys to press to select the drawing character; the various types of ladder, treasure and so on, which can then be placed by cursor keys or joystick. The documentation fails to mention that instead of pressing keys 1 to 9 to select a character, you can select space to erase characters! The auto-repeat from left to right or from top to bottom is a nice touch in this section. When placing characters using keyboard or joystick, they move 8 pixels at a time.

The documentation indicates that you may leave the editor by pressing enter. It would have been nice to be told in advance that this does not immediately leave the level but rather then allows you to place first your own man and then the three baddies.

The men can be moved by keyboard or joystick, but if you move them by joystick they move only 1 pixel at a time. The documentation carries the following text:

"You must be very careful when placing the men with the joystick since they may act strangely when the TI Runner game is played.".

That is it, that is your warning, no further explanation! No, I cannot explain this cryptic comment (I am useless at crosswords too) except to suggest that it may be possible to place a man such that he may be one pixel into a brick! It is safe to place the men one pixel above the bricks, provided their heads are not thereby placed one pixel into an overhead brick!

Bug: If you placed the last man using the joystick fire button, you then had distinct problems in carrying on using the program! You could not enter anything to select from subsequent menus unless you wishes to enter the default value, and if you scrubbed that out could not reenter it. .

This caused me some headaches. What was happening was that although the ACCEPT AT routine for the menu was perfectly alright, the console was still mapping the keyboard as CALL KEY(1...), something you cannot emulate in BASIC, but the routine to move the men is in machine code, and that code is not resetting the key scan flag on exit! Fortunately you can reset the keyscan flag in BASIC, so this bug is easily resolved.

Edit the program Editor as follows:

Into the prescan lists at the start of the program add the variables @ and @@, and the CALL KEY. Then add a new line as follows:

```
1545 CALL KEY(3,@,@@)
```

and that bug is quite squashed!!!

When editing or creating a new screen, there are several things you must watch out for. It is up to you to ensure that the screen really is able to be completed .

Remember I advised you to note how many treasures were required to finish a screen? Two quotes from the documentation will explain this:

"You can enter 5 treasures, but have six on the screen and fool the player into going for impossible treasures."

"A level created with more treasures on the screen than are needed to win that level will behave strangely. When playing this type of level, the Runner man will not be able to get up any ladders if more than the set amount of treasures are taken. At this point the level is impossible to win."

I consider this a design fault. The editor should have been able to note the number of treasures required, and flashed a warning if you tried to leave the editor with too many (or just as bad, too few) treasures on screen. It is not fatal, just requires care!

We now come to:

2. File Management. Which contained a very interesting bug indeed, pure Extended BASIC, and the first time I have met him. Sub-option 1 should allow you to "change drive number". This allows you to create a new file on a blank disk. LE uses a data file SCRLST to record the different level files on your disk, faster than using the disk directory! So when you select a new disk, it is necessary for the file SCRLST to be created. My copy of the program steadfastly refused to do this, preferring to terminate the program.

This error was caused as follows:

The program has one ON ERROR routine.

Specific errors are caught by making use of CALL ERR.

Change Drive Number caused an error if the program could not read from a file SCRLST on the new disk.

This error was tested for by a test for the LINE NUMBER the error occurred in.

Know something? When your program has:

```
1000 CALL ERR(A,B,C,D)
1010 IF D=2030 THEN...
```

where D refers to a potentially difficult program line, if you then resequence your program, that line reference test in line 1010 is not amended, and your error check fails to work!

This is what happened, and was spotted, as a REM line very rarely causes errors (ever?). So, again you must edit the disk file Editor:

Change line 2910 to read:

```
2910 IF Z=1710 THEN EMPTY=1 :: etc etc
```

And that is that bug squashed.

Finally, Option 3, Prepare a Screen File for Play. I can understand (almost) why this option was provided, but my best suggestion is that you do not use it. The LE creates a data file with a name chosen by you. In order to use that file, its name must be changed to LEVEL28, the name that the TI Runner program expects! Option 3, calmly and coolly creates a new file LEVEL28 and then copies the data across. We are talking here about a 100 sector data file!

It is very much faster to use your favorite disk manager to change the file name and copy it over to a duplicate program disk, remembering to call the disk name TR.

There is a further error in the documentation supplied at this point. "Be prepared to swap disks if you have only one disk drive". This option will not ask you to, nor indeed let you. Your disk manager will.

Subject to bug squashing it is a simple matter to use this program, and the difficulty is not in using the program but rather in designing new screens! In any case, you have an extra 40 screens to work your way through (wow are they hard!).

In summary, subject to the changes and explanations here provided, this is a product that TI Runner afficionados need. The product can be improved in design and documentation, but the price is not very high.

Report Card:

Heading:	As supplied:	Debugged:
Performance	C	A
Ease of Use	D	B
Documentation	C	A
Value	A	A
Final Grade	C	A-

Following on from the above review, sent to you a few weeks back, I have now received notification from EB Software that the version sent to MICROpendium for review was a beta test copy (there was no indication of such in the program).

EB Software have sent to me a disk claiming itself as Version 1.3 of the Editor. In addition to an ADVANCED set of screens to play with your TI Runner program there is also a further 50 screens. So this disk would add two further sets of screens for you to play, regardless of whether you use the editor.

Despite finding some bugs in the original review copy, I was able to make it work.

I regret to report that while Version 1.3 has every appearance of working correctly, the files that it produces are highly unreliable. Typical faults include incorrect behaviour of hidden ladders, which sometimes work while invisible, and sometimes fail to appear when all the treasures are taken, and the unreliable operation of the bad guys, one or more of whom now run along the screen horizontally, regardless of any walls or lack of walls, or of any obstructions.

Using the Create Screen option makes things even worse, causing severe corruption to the playing screen.

If Version 1.3 is the current commercial version then I regret I must downgrade all gradings to E-.

Having now received two versions of the program, both of which failed to work when received I must recommend caution! I was able to make the first version work, but the fault in the current commercial version seems to be in the machine code section. If you send for the commercial version and it fails to work, at least you have the two sets of screens and if you send me your original purchased disks, I can at least send you a modified and working beta test version of the editor, as first reviewed above! (Return post and packing please!).

I have also tried a Freeware editor. The first disk received was apparently blank, and the second one, while apparently creating new screens, also failed to produce screens that would play correctly.

Stephen Shaw o

continued from page 11

M A Ballman, and a loader for it written by J A Johnson, both of Florida. Again the original Show Directory is heavily modified!

>LINKER LIBRARIAN by Tom Bentley. Perhaps essential for the above! This program allows you to build up your own libraries quite easily, and a c99 library file is included. When you use LINKER to create a memory image of a C99 program, the LINKER will search the library for the c99 library entries the program needs. For instance, PRINTF would be automatically loaded. Neat.

>JEUX 2. A Wycove Forth version of Billard, self contained, you do not need Wycove Forth. Crashes on me fairly frequently but not always, and a good game. Space Station Pheta (Extended BASIC -PD), Largage, Poursuite, Puissance4, Quintus, Randonnee, and Ruins.

>JEUX 3. Achille, Blackfish (Documentation in English), Chenille, Dames, Dominoes, Meltdown (from HCM), Minotaur, Monkey, Perdu (French hangman!), Tircroise.

On the above 3 disks, the games have instructions in French, but its fairly easy to follow. Just use joystick or ESDX! There are some older games I have managed to miss so far, and some really superb games from our Continental brothers! o

Tips from the Tigercub #57

by Jim Peterson, Tigercub Software, USA

156 Collingwood Ave.
Columbus OH 43213

I am still offering over 120 original and unique entertainment, educational and utility programs at just $1 each, or on collection disks at $5 per disk.

The contents of the first 52 issues of this newsletter are available as ready to run programs on 5 Tips Disks at $10 each.

And my three Nuts & Bolts Disk, $15 each, each contain over 100 subprograms for you to merge into your own programs to do all kinds of wonderful things.

My catalog is available for $1, deductable from your first order (specify Tigercub catalog).

****
TI-PD LIBRARY

I have selected public domain programs, by category, to fill over 200 disks, as full as possible if I had enough programs of the category, with all the BASIC only programs converted to Extended BASIC, with an Editor Asembler loader provided for assembly programs if possible, instructions added and any obvious bugs corrected, and with an autoloader by full program name on each disk. These are available as a copying service for just $1.50 postpaid in US and Canada. No fairware will be offered without the author's permission. Send SASE for list or $1, refundable for 9 page catalog listing all titles and authors. Be sure to specify TI-PD catalog.

****

I like little programs that load quickly and do just what I want to do at the moment. And one of the things I wanted to do quickly was to find phone numbers. So, I used Funnelweb to create a little file:

```
SMITH,JOHN (999) 111-2222
BUSH, GEO. (000) 123-1234
GHADDAFI, O. (666)66-6666
```

and all my other frequently called numbers. I SAVEd it as DSK1.PHONELIST and wrote this little routine to use it.

```
100 CALL CLEAR
110 OPEN #1:"DSK1.PHONELIST",INPUT
120 DISPLAY AT(12,1):"LAST NAME?" :: ACCEPT AT(14,1):N$
130 LINPUT #1:M$ :: IF POS(M$,N$,1)<>0 THEN DISPLAY
    AT(16,1):M$ :: RESTORE #1 :: GOTO 120
140 IF EOF(1)<>1 THEN 130
150 DISPLAY AT(16,1):"NAME NOT FOUND" :: RESTORE #1 ::
    GOTO 120
```

Now actually, that was all I needed, (even though it did take several seconds to find a name at the end of the file), and it was easy enough to load the file into Funnelweb when it needed updating. But, programmers are never satisfied, so I decided to write a self-contained program:

```
100 CALL CLEAR
200 DATA "ALDA, ALAN 888-9999"
201 !@P-
300 DATA "BUSH, GEORGE 111-1111"
400 DATA "PRESLEY, ELVIS 000-0000"
499 !@P+
500 DISPLAY AT(12,1):"LAST NAME?" :: ACCEPT AT(14,1):N$
600 READ M$ :: IF POS(M$,N$,1)<>0 THEN DISPLAY AT(16,1):
    M$ :: RESTORE 200 :: GOTO 500
700 ON ERROR 800 :: GOTO 600
800 DISPLAY AT(16,1):"NAME NOT FOUND" :: RESTORE 200 ::
    GOTO 500
```

That funny thing in line 201 turns off the prescan and speeds up initialization. This routine is no faster than the last, but can be updated by editing the program itself. It is limited to about 500 records due to the least-known and greatest weakness of the TI, that string storage is limited to console memory.

But, computer users are paranoid about speed, so I decided to put my data into a pre-loaded array with self incrementing subscript numbers, and find the data by a binary search.

```
100 !QUICKFINDER by Jim Peterson
200 DIM D$(50):: GOTO 300 :: D$(),X :: !@P-
300 X=X+1 :: D$(X)="ALDA, ALAN (999) 666-1234"
400 X=X+1 :: D$(X)="BUSH, GEORGE (111) 111-1111"
500 X=X+1 :: D$(X)="GHADDAFI, OMAR (999) 456-1234567"
600 X=X+1 :: D$(X)="KHOMEINI, AYATOLLAH (666) 666-6666"
700 !@P+
800 INPUT "NAME? ":M$
900 IF M$>D$(X)THEN PRINT "NOT FOUND":"CLOSEST IS":D$(X)
    :: GOTO 800
1000 IF M$<D$(1)THEN PRINT "NOT FOUND":"CLOSEST
    IS":D$(1):: GOTO 800
1100 H=X :: S=INT(X/2)
1200 S$=D$(S):: IF POS(S$,M$,1)=1 THEN 1700
1300 S$=D$(S+1):: IF POS(S$,M$,1)=1 THEN S=S+1 :: GOTO
    1700
1400 IF S$>M$ THEN H=S :: S=INT(H/2):: GOTO 1600
1500 S=S+INT((H-S)/2)
1600 IF S=S2 THEN 1800 ELSE S2=S :: GOTO 1200
1700 PRINT D$(S):: GOTO 800
1800 PRINT "NOT FOUND":"CLOSEST ARE"
1900 IF D$(S2)>M$ THEN PRINT D$(S2-1):D$(S2+1):: GOTO
    800
2000 PRINT D$(S2+1):D$(S2+2) :: GOTO 800
```

Note that in this case the records must be in alphabetical sequence. New records can be inserted in intermediate line numbers, in alphabetic sequence, always preceded by X=X+1 :: D$(X)= . Obsolete records can be deleted, and records can be corrected in place if the correction does not change the alphabetic sequence.

This idea did not work out as well as I hoped. The maximum number of records is less than 300, for the reason mentioned above, and this leaves so little free memory that even a binary search is slow. However, for a smaller file this is perhaps the best method.

For a large file, the best method is certainly a fixed sequential disk file, accessed by a binary search routine. But, that requires other routines to delete, add or change records, and had best be the subject of another Tips.

There is apparently a mistaken belief that sprites cannot be used together with my BXB routine. Not so. You can use all 28 of them! However, you cannot change their colour with CALL COLOR(#,N). The only other limitations of BXB that I can think of, are that a single CALL COLOR cannot be used for multiple character sets and a single CALL CHAR can only reidentify one character. CALL CHARPAT cannot return the hex code of an ASCII above 143 because those ASCII's were not supposed to be available in Extended BASIC.

I have used BXB on hundreds of BASIC only programs and have had only two rare problems. If the program contains multiple line feed colons ::::::, the computer may rearrange them into pairs of double colons :: :: and lock up. Or, if the colons are before the text, as in PRINT :"something" you may get a puzzling error message.

Also on rare occasions you might get an error message indicating the subprogram was called from a line containing a CALL CHAR, if the programmer had inadvertently put more than 16 characters in the hexadecimal code. BASIC just ignores any extra characters, and Extended BASIC uses them to reidentify the following ASCII, but BXB crashes.

From the T*I*M*E*S newsletter from England, here is an extremely useful bit of assembly which should be assembled as ALPHA/O and placed on the disk of every joystick program, or imbedded in it with ALSAVE.

```
        DEF   ALPHA      * save old R12
ALPHA   MOV   R12,@>FFFC        * 9900 CRU base=0
        CLR   R12        * signal alpha lock key line
        SBZ   21         * check alpha lock other side
        TB    7          * jump if state=on
        JNE   STATE
* state=off
        SETO  @>FFFE     * as off skip next line
        JMP   JUMPA
* state=on
STATE   CLR   @>FFFE     * stop sending to alpha key
JUMPA   SBO   21
        MOV   @>FFFC,R12        * restore R12
* standard Extended BASIC return now
        SB    @>837C,@>837C   * clear error for BASIC
        B     @>0070     * return to calling program
        END   ALPHA
```

Now, put this in the first lines of the joystick program:

```
1 ! by M. Gikow, Andover        MA August 1988
2 ! used with ALPHA/O,          will detect whether
          Alpha Lock is up (A=          255) or down
   (A=0)
3 CALL CLEAR :: CALL INIT :: CALL LOAD("DSK1.ALPHA/O")
4 CALL LINK("ALPHA"):: CALL PEEK(-1,A):: IF A=0 THEN
   DISPLAY AT(12,1):"RELEASE ALPHALOCK" :: GOTO 4 ELSE
   CALL CLEAR
```

I published this one in the C.O.N.N.I. newsletter. Barry Traver picked it up and put it in the TI Forum in Computer Shopper, but their typesetter garbled it, so here is how it was supposed to be:

According to the TI-Writer Reference Guide, page 77, when you select the PrintF command, then type C and space once and then the device name, any control characters with ASCII less than 32 are removed before the file is printed.

With Funnelweb, at least, this is not quite true. A carriage return character, ASCII 13, or a line feed character, ASCII 10, at the end of a line is actually not deleted but is changed to the space bar character, ASCII 32. This can be proved by running this little routine:

```
100 OPEN #1:"DSK1.(filename)",INPUT
110 LINPUT #1:M$ :: PRINT M$:LEN(M$):: IF LEN(M$)>0 THEN
   PRINT ASC(SEG$(M$,LEN(M$),1))
120 CALL KEY(0,K,S):: IF S=0 THEN 120 ELSE 110
```

Therefore, when a file is Filled/Adjusted and the line feed characters are stripped with the C option, the lines are one character longer than they appear to be. An apparently blank line also contains ASCII 32.

Since these characters are blank, they normally do no harm. However, they can create problems when records are read into programs for multiple column printing or concatenation of strings. In these cases, this routine can be used to strip out any ASCII below 33 at the ends of records.

```
100 DATA INPUT,OUTPUT
110 FOR J=1 TO 2 :: READ J$ :: DISPLAY AT(12,1)ERASE
   ALL:J$&" FILENAME?":"DSK" :: ACCEPT AT(13,4):F$(J)::
   OPEN #J:"DSK"&F$(J),UPDATE :: NEXT J
120 LINPUT #1:M$
130 IF ASC(SEG$(M$,LEN(M$),1))<33 THEN
   M$=SEG$(M$,1,LEN(M$)-1):: IF LEN(M$)>0 THEN 130
140 PRINT #2:M$ :: IF EOF(1)<>1 THEN 120 :: CLOSE #1 ::
   CLOSE #2
```

Attention all newsletter editors! If you are going to print my Tips (or anything else that contains program listings!) through the Formatter, PLEASE first replace and transliterate the ampersand, asterisk, period, carat and "@" sign!

Print this one through the Formatter and see why:

```
100 A=A*264 :: @=1
110 PRINT "1 . . . 2 . . . 3 . . . 4 . . . 5 . . . 6 . .
   . 7 . . . 8 . . . 9 . . . 0"
120 M$=M$&A$&B$&C$ :: K=K^3
```

becomes:

```
100 A=A4 :: =1
110 PRINT "1 .   .   . 2 .   .   . 3 .   .   . 4 .   .   . 5
   .   .   . 6 .   .   . 7 .   .   . 8 .   .
120 M$=M$A$B$C$ :: K=K 3
```

Here is how you do it.

Load the above in the Editor, position the cursor at the beginning of the 1st line, hit FCTN[9], type RS and ENTER, then /&/}/ and ENTER. At the prompt, type A. Now get the cursor back to the beginning, repeat the above with /*/|/, and then /./\/ and /^/~/ and /@/{/ and the file should now look like this:

```
100 A=A|264 :: {=1
110 PRINT "1 \ \ \ 2 \ \ \ 3 \ \ \ 4 \ \ \ 5 \ \ \ 6 \ \
   \ 7 \ \ \ 8 \ \ \ 9 \_\ \ 0"
120 M$=M$}A$}B$}C$ :: K=K~3
```

Now use FCTN[8] to open 5 lines at the top and add this transliteration:

```
.TL 92:46
.TL 123:64
.TL 124:42
.TL 125:38
.TL 126:94
```

Save the result, go to the Formatter and print it.

If my multi-column Printall program (Tips from the Tigercub #45) will not run on your Epson-compatible printer, try changing line 250 to:

```
250 ACCEPT AT(12,3)VALIDATE("123")SIZE(1):P :: IF P=2
   THEN PRINT #1:CHR$(27);CHR$(77)ELSE IF P=3 THEN
   PRINT #1:CHR$(15)
```

You might also need to change the 136 in line 280 to 132.

If your printer offers the elite condensed option, you might want to add:

:" (4) ELITE CONDENSED" to line 240, change the VALIDATE string in 250 to "1234", add ELSE IF P=4 THEN PRINT #1:CHR$(27);CHR$(77);CHR$(15) to the revised line 250 and add +160*(P=4) to the first statement in line 280.

Memory almost full, Jim Peterson

Fairware Author
of the Month

Tony McGovern has been selected as the first TIsHUG Fairware author of the month. If you would like repay him for his efforts in producing the Funnelweb system, please bring in your money to the next meeting or send a cheque (no matter how much) to the Secretary before the next meeting (July 1st). TIsHUG will then send Tony a cheque on behalf of all members.

The August Fairware Author of the Month will be the Ottawa Users Group for DM1000.

Tidbits Eight

by Wade Bowmer

GREETINGS!

This is my first article written with Extended BASIC's ! statement. It is certainly different to what I am used to.

REWIND

I have since discovered another snippet of information regarding ACCEPT: the UALPHA option within VALIDATE also permits spaces! Is that not sneaky?

I discovered it quite by accident. I was writing a program that accepted words only, and had used CALL KEY(3,K,S) and VALIDATE(UALPHA), except that spaces seem to keep getting through...

I remember some while ago, about three years in fact, there was almost a whole TND devoted to Assembly. Neat. Ross Mudie wrote an article about how to check if a particular machine language routine was already installed in memory by CALL PEEKing the REF/DEF table.

I found out another hitherto hidden quirk of Extended BASIC while trying to use a similar technique. I had done a CALL PEEK(I,A,B,C,D,E,F) where I was the address of the entry in question, and proceeded to compare it with a string as such:
IF CHR$(A)&CHR$(B)&CHR$(C)&CHR$(D)&CHR$(E)&CHR$(F)= "COLOUR" THEN etc. However, I kept receiving * STRING-NUMBER MISMATCH error messages, until I realized that it needs to be like this:
IF (CHR$(A)&CHR$(B)&CHR$(C)&CHR$(D)&CHR$(E)&CHR$(F))= "COLOUR" THEN etc. Note the ()s? Very Important, because the order of precedence for mixed expressions is Numeric and Arithmetic, then Relational and Logic and finally String Concatenation! And this time it is in the manual. Look on page 41, in the "Relational Expressions" subsection. Incredible, is it not?

TAPE TRAUMAS.

Have you ever had to go through the arduous process of changing your tape drive's head position so that you can load someone else's tape? And then have to reposition it so that you can load your own programs again? Your tape drive is never quite the same again!

But there is a little trick that has a 50/50 chance of actually working. Let me indulge a moment...

All tape drives (for the layman: also called tape recorders or cassette players or ...) have a little tiny screw that can be adjusted to change the angle of the head in relation to the tape running past it. On some machines, the whole contraption needs to be pulled apart so that you can actually play a tape while you adjust the head. On others, you do not have to; just insert a Jeweller's Screwdriver into the right hole at the right angle, and turn. Any small screwdriver will do, it is just that a Jeweller's is recommended.

The ideal position for the head is parallel to the tape, but sadly, it can often not be ascertained exactly where this occurs. Normally it is adjusted for best sound quality (just listen for the treble) and, if on a stereo deck, equal quality on both channels.

And now my little trick makes its appearance. If you push down on the head with just the right amount of force, and keep it there, you can load any program without having to adjust the head! It works best on recordings that are far enough out for the TI99/4A to understand them with the occasional * ERROR IN DATA DETECTED. Of course, it only works if the recording was recorded at an azimuth (yes, that is what it is called folks!) "lower" or further "into" the drive than yours. As I said, a 50/50 chance!

Speaking of tapes, in the April '88 TND someone wrote about a "tip" to wait until OPEN #1:"CS1.",INPUT had scrolled up the screen before actually pressing Play. Why? Because he or she thought that OPEN waited a far longer length of time than the length of the EEEEEEE of the recording's leader!

Fair enough, except that that was not what OPEN 's delay is for. Texas Instruments had designed the cassette system to be used at the start of cassettes. Yes, standard audio cassettes, the ones with leaders!!

Anyway, take a look at Listing 1. I have written it to be semi-independant of this article. Basically, it writes a file of 16 records and an EOF marker to tape, and then reads it back again. All designed to show people just how to File Process with Tape. If you did not type it in earlier, you can do it now. I will still be here when you have finished.

The first comment I need to make is that if you RUN it, follow all instructions exactly, no fancy tricks with the Pause key, etc. Now, a couple of pointers:

1> The program essentially outputs a table of data that could possibly be of some use at a later date.

2> I have deliberately used DISPLAY instead of PRINT so that the distinction between writing to a file and writing to the screen is clearly evident. If you like, you can replace all the DISPLAYs with PRINTs OR PRINT #0:s and the program will function exactly the same. I used LINPUT# instead of INPUT# for a similar reason.

3> You will have noticed the pause of about 12 seconds from when you pressed ENTER to when the screen scrolled up a line. OPEN merely skips the tape's leader.

4> The technique of outputing the record number (yes, they start at zero) is in fact a good way of checking which one you are actually up to when reading them back in. Kind of a fail safe mechanism. The program does not check this later, but you can get the idea.

5> After all the data is output, notice that an "EOF marker" is written. "EOF" stands for End Of File and is required so that the program reading the data knows when to stop. The EOF() function used with disk files will not work with tape, I am sure you can see why. The other way is to count the number of data items beforehand and output a special record at the start alerting the reading program to just how many records follow. You could combine all three methods to have a really foolproof system.

6> After you Rewind your tape to read it back in again, you may notice that towards the end of OPENs pause, you hear EEEEYOO then WEEEEEEE as the first record is read. It is nothing to really worry about, the TI99/4A is quite capable of reading a record from part way through the leader (not the data!). If it really does bother you, then press ENTER first, and then press Play.

7> Notice the file number is different. I think you should all know by now that because it is an entirely new cycle of OPEN-I/O-CLOSE, BASIC does not care about the file number. I just used a different one to show you that it is a different operation; INPUT as opposed to OUTPUT.

Did it get all the data right? I had not mentioned the Remote Plug earlier because those of you without, would probably know how to get by in anycase. (Let me know if you are different.) It is just a matter of pressing Play or Record at the same time as ENTER. Actually, the table of hexadecimal constants I had the program output and input is a prime example of a useful piece of information stored for later use.

Which reminds me. It is possible to save a bit of space when you save programs on tape by waiting eight seconds or so between pressing ENTER and pressing Record. I myself use the Pause key. However, the 12-15 second blank section recorded can be useful if you later decide to add your voice before the program just to say what it is. As usual, it _is_ up to you.

ZOUNDS!

Music... Hmmm, let me see. Yes, there _was_ an article on Extended BASIC Music programming a year or so back.

Which is a bit too long ago. Now, can any of you remember Sound Lists? Those marvellous strings of numbers one puts in VDP RAM so the computer can play a tune, while you are playing a game with it. It has definitely been too long since they were mentioned. Let me rectify that.

October '85 TND carried a fairly detailed article by Fred Hawkins aimed at presenting a readable approach to these sound lists. It is now my turn to do similarly.

A sound list consists primarily of sound chip instructions that the interrupt routine feeds regularly to the sound chip. The general format is <number of instruction bytes> <instruction bytes> <duration in 50ths of a second> For readers with PAL consoles the above rings true, but for NTSC consoles, the duration is measured in 60ths of a second, due to the different frame speed. In case you did not know, the VDP chip provides the interrupt pulse once every Vertical Blanking Interval. (It can also be called Vertical Blacking Interval, for obvious reasons.)

But I digress. Now to descibe the sound chip instructions. The high nybble, which is the most significant four bits, is the actual command. All the values are _greater than >8_, so that the chip ignores values without bit 0 set. Sometimes it does not, though, which is for later. The middle two bits designate the voice, voice 3 is noise. The least significant bit of the nybble determines whether it is Volume or Frequency (Noise Factor 1 for voice 3), set it for volume.

Sensory overload? Okay, let us try a different tack. To set the frequency of the voices, use >8, >A or >C for voices 0, 1 or 2. For volume, use >9, >B or >D, respectively. For the noise channel, use >E to set the noise factor, and >F to set its volume.

That is just the commands. Now for the parameters. For the volume, the low nybble sets the attenuation. >0 is maximum volume, or zero attenuation, and >F is voice disabled. They correspond to CALL SOUND levels 0 and 30, respectively. The Noise factor is set similarly, except it only goes from >0 to >7. They correspond directly to noises -1 to -8 in order.

Another way of looking at that is that bit 1 in that nybble designates periodic or white noise. Set it for white. Bits 2 and 3 set the shift rate. If the shift rate equals Binary 11, then the frequency of voice 2 determines the shift rate. See Tidbits in September '86 TND, but in brief it is

```
CALL SOUND(duration,x,30,x,30,shift-rate,30,-4 or -8,30)
```

the x depicting any old frequency.

For a quick demonstration, try this in Extended BASIC immediate mode. First enter:

```
SOUND=-31744::CALL INIT::CALL LOAD(SOUND,231)
     Then enter:
CALL LOAD(SOUND,240):: I=.7 :: FOR J=1 TO 20 :: I=I*1.25
   :: CALL LOAD(SOUND,192,0,I) ::NEXT J :: CALL
   LOAD(SOUND,255)
```

Try it several times (get it back with REDO). Try varying the stepping factor. Try a different starting value. Try a different length of loop. Be careful, though, not to let I get bigger than 63. It sounds good through a 10W amplifier.

That was the easy part, now it gets tougher. The frequency is sent to the chip in 10 bits, the four least significant with the command, and the six most significant as a second byte. That explains, by the way, why the second value poked in above should not exceed 63. The poor old sound generator can do some mighty strange things if the value is too big.

If you had a certain frequency in mind, and you had it in Hertz, and you wanted to tell it to the sound chip, then here is how you do it. Divide the frequency _into_ 111860.6. Round the result if you want, then take the appropriate division: six most significant bits and four least significant, and bung them in!

In case you were wondering about the number 111860.6, well, it has something to do with the clock frequency the chip gets.

If you really want to know, the clock frequency the sound chip receives is about 447kHz. (If the 111860.6 is accurate, then the clock is really 447,442.4 Hz.) The sound chip divides it by 4, which is consistent with what I have read about it elsewhere, and then does another fancy division to make its music. The strange number, 111860.6, is simply 447,442.4 divided by 4.

The clock frequency itself comes from the VDP chip (of course), and happens to be the same regardless whether the chip is PAL or NTSC. I conducted a simple experiment with a tuning fork to prove this.

Those of you with Technical manuals may know that the chip could get 3.58MHz instead of 447kHz. True. In that case, the sound chip is slightly different, and divides the clock frequency by 32 instead of 4. So that means it would be 3,579,539.2Hz. Incidentally, that also comes from the VDP Chip.

High time to put everything into a table. Do you not agree?

Frequencies :

	Byte One		Byte Two
	Cmd	4LSB	Six MSB
Voice 0:	>8	>0	>00
	128	0	0
Voice 1:	>A	to	to
	160		
Voice 2:	>C	>F	>3F
	192	15	63
Noise :	>E	0to7	Not
(Voice3)	224		Needed

Volumes:

	Cmd	Attn
Voice 0:	>9	>0
	144	0
Voice 1:	>B	to
	176	
Voice 2:	>D	>F
	208	15
Noise :	>F	
(Voice3)	240	

It all comes as plain as day when you see it like that. This next table is for converting music to frequencies. The reason it is presented across the page is because that is the nature of written music.

Note:	A	A#/Bb	B	C	C#/Db	D	D#/Eb	E
Freq:	110	117	123	131	139	147	156	165
Hex:	93F	03C	A38	735	732	A2F	F2C	72A
Dec:	9,63	0,60	10,56	7,53	7,50	10,47	15,44	7,42

Note:	F	F#/Gb	G	G#/Ab	A	A#/Bb	B	C
Freq:	175	185	196	208	220	233	247	262
Hex:	128	D25	B23	B21	C1F	01E	51C	C1A
Dec:	1,40	13,37	11,35	11,33	12,31	0,30	5,28	12,26

Note:	C#/Db	D	D#/Eb	E	F	F#/Gb	G	G#/Ab
Freq:	277	294	311	330	349	370	392	415
Hex:	419	D17	816	315	014	E12	D11	D10
Dec:	4,25	13,23	8,22	3,21	0,20	14,18	13,17	13,16

Note:	A	A#/Bb	B	C	C#/Db	D	D#/Eb	E
Freq:	440	466	494	523	554	587	622	659
Hex:	E0F	00F	20E	60D	A0C	E0B	40B	A0A
Dec:	14,15	0,15	2,14	6,13	10,12	14,11	4,11	10,10

Note:	F	F#/Gb	G	G#/Ab	A	A#/Bb	B	C
Freq:	698	740	784	831	880	932	988	1047
Hex:	00A	709	F08	708	F07	807	107	B06
Dec:	0,10	7,9	15,8	7,8	15,7	8,7	1,7	11,6

Note:	C#/Db	D	D#/Eb	E	F	F#/Gb	G	G#/Ab
Freq:	1109	1175	1245	1319	1397	1480	1568	1661
Hex:	506	F05	A05	505	005	C04	704	304
Dec:	5,6	15,5	10,5	5,5	0,5	12,4	7,4	3,4

Note:	A	A#/Bb	B	C
Freq:	1760	1865	1976	2093
Hex:	004	C03	903	503
Dec:	0,4	12,3	9,3	5,3

See music scales on page 34

By now, it is very likely that you are positively itching to do some direct sound. Sure thing, it is really quite easy. The decimal numbers (from the Dec row) above are the ones you put in. To the first one, add the voice command from the previous table. For example, You want to put that highest F into voice 1. Okay, voice 1 is 160, and the F is 0,5, so you do
CALL LOAD(-31744,160,0,5)

Now, before you think anything, 160+0=160, and the 0 in the CALL LOAD is because the sound chip is only connected to the high byte on the 16 bit bus. Now for some volume. Voice 1 has volume command 176, and do we want full volume? Yes, so
CALL LOAD(-31744,176)

When you have had enough, add 15 to the volume command and CALL LOAD that, i.e.
CALL LOAD(-31744,191)

That particular tone above is quite recognizable as the BEEP tone. Also, you can replace -31744 with a variable, such as SOUND, the one I used earlier. And do not forget the CALL INIT before you start, or your CALL LOADs will generate * SYNTAX ERRORs.

Another example? I think so, a lower note this time; the Eb above middle C. The decimal values are 8,22, and we want it in Voice 1 again, so we
CALL LOAD(-31744,168,0,22)
then
CALL LOAD(-31744,180)

And it is not full volume this time, attenuation of 4. That should make more sense.

It does not? Oh dear. The frequency... Oh, I see. Why did I use two CALL LOADs ? Well, it is a bit difficult to explain, partly because I do not really know why. You see, there is some funny quirk with the sound chip that I do not understand. When you send multiple values in a CALL LOAD, it usually does not work correctly. So it is just a lot easier (and safer) to do it in separate CALL LOADs. If you like, you can separate each number with ,"",-31744, I think it saves a bit of memory. Using sound lists totally avoids this strange problem, fortunately.

A word about those frequencies. You might notice that the numbers in the Hex row seem to be all over the place. I thought so once, but look at the middle digit as the highest, and the first as the one that is normally on the end. Now you will notice that they decrease with increasing frequency. That is correct. Remember the division with 111860.6? The number we actually send to the chip is the division it carries out on its clock frequency. It is really very simple. It is also slightly easier to see with the decimal numbers (sorry!), just mentally reverse them and you will see the order.

By the way, the higher the frequencies, the closer the numbers are together, which is why you should stick to the lower ones, as the higher one goes up the scale, the less accurately the sound chip renders the note.

That is all very well, but we cannot really program with a great heap of CALL LOADs now, can we?

The easiest way to set up a sound list is with CALL CHAR. I hear notes of doubt. The pattern string you use to create characters is really Hexadecimal. Which makes it very easy to put Hexadecimal bytes into VDP RAM.

But before we do anything towards putting sound lists in, we first should disable the sound routine. Remember CALL LOAD(-31806, 16) to disable that accursed QUIT key? Well, the same location is used for a number of similar purposes. CALL LOAD(-31806, 64) stops all sprite motion (but QUIT and sound are still enabled), CALL LOAD(-31806, 128) stops everything, and CALL LOAD(-31806, 32) disables sound. You want QUIT disabled too? And quite right, okay, use CALL LOAD(-31806, 48) instead. And re-enable? Easy: CALL LOAD(-31806, 0) or, with the QUIT key still out, CALL LOAD(-31806, 16)

So turning off the sound routine is done first. Then we tell it where it is and that there is one. That is done with a CALL LOAD to -31796. Working out just where the list is, is very simple if we use CALL CHAR. Just add 96 (or >60) to the character number used and multiply by 8 and that is where it is. Only, to poke it, we need it in two bytes, rather than a word. High byte= INT(address/256), Low byte= address AND 255. The trigger value is 1. So we
CALL LOAD(-31796,Hi, Low ,1)

But that is not all. We then have to do another CALL LOAD to tell it that the list is in VDP instead of GROM. That is just
CALL LOAD(-31747,1)

Just use a 0 if you ever want to play a sound list in GROM (or GRAM).
Enable sound routine now? No! What about the sound list? That goes in next. Listing 3 shows you one way to do it. The other is in Listing 4. Now we can re-enable the routine, as described above, with either a
CALL LOAD(-31806,0)
or
CALL LOAD(-31806,16)

You could totally ignore -31806 and load the sound trigger now if you like. Both ways work just as well.

There are some other important things to talk about before you go racing off. The Count byte indicates how many Bytes follow, not the number of instructions, because, as you know, frequency takes up two bytes. So if you set a frequency and two volumes, then the Count byte is 4, not 3. Duration is the number of interrupt cycles between this instruction sequence and the next. Duration of 1 means the next sequence is attended to on the next interrupt. The interrupt for the routine comes from the VDP chip (where else?) and in the PAL consoles, comes 50 times a second. In the NTSC consoles, it comes 60 times a second. I think I have already explained that.

But there are some special situations that can occur if Count or Duration are zero. Duration first. If it is zero, as mentioned, the sound list stops. However, the address of the next byte is saved at -31796.

It is a little different if Count is zero. First of all, the next two bytes specify the address of a new sound list, in High/Low format (what did you expect?). Then Duration is set to 1, and saved in Trigger (-31794). So, on the next interrupt, the sound routine ploughs through another sound list, oblivious it has just been redirected. However, if the previous list was in VDP RAM, the new address also refers to VDP RAM, and the same for GROM/GRAM.

But it can go over to the other, if Count is instead >FF or 255. This toggles the bit at >83FD (-31747), and proceeds as if Count were zero. Incidentally, one can use this "Jump" instruction to have the sound list loop continuously. Just remember to take 1 off the last duration, so that it does not "miss a beat".

My more astute readers will notice that the list can be in GROM or GRAM (I did actually mention it, though...). Which explains all those Command Modules that play some irritating tune in the background while it beats the living daylights out of you. But honestly now, how else did you think it would be done?

Something else that comes to mind after my experiments a few days ago, is that CALL SOUND waits for any current sound list to finish before playing its own note(s). I expect a negative duration would cut it short. (Try it!) After all, CALL SOUND uses a very simple sound list of its own, anyway.

I must thank Stephen Shaw for his article MINI-MEMORY SOUND from TI*TIMES, which TIsHUG reprinted in August '85 TND: that was the article that initially sated my interest in direct sound... for a while! I also must thank Fred Hawkins (mentioned before) for XPERIMENTER'S XBASIC from LEHIGH 99'er COMPUTER GROUPS magazine, Volume 3 #3, March '85. TIsHUG reprinted it, too, of course, in October '85 TND. Many thanks for your flowchart of the sound interrupt routine! It has been positively invaluable.

I might warn you now, once you start dabbling in sound lists, you will get hooked. It is very easy to get the hang of them, for which reason I have included the essence of the Editor Assembler's CRASH DEMO below. When typing it in, do not actually type any of the spaces in the Hexadecimal string, I just spaced it out so you could see each individual sequence.

```
10 !CRASH DEMO
110 CHIP=-31744::CALL INIT
120 CALL DISABLE
130 CALL LOAD(-31796,7,0,1)
```

This sets the trigger and loads the address of the sound list. (128+96)x8 = 7x256+0 = 1792, remember?

```
140 CALL LOAD(-31747,1)
    The list is in VDP...
150 CALL CHAR(128,"
    03 9FE4F2 05
    02 E4F0 0C
    02 E4F2 0A
    02 E4F4 08
    02 E4F6 06
    02 E4F8 04
    02 E4FA")
160 CALL CHAR(132,"
         02
    01 FF00")
```

That was the list. Remember, do not type the spaces!

```
170 CALL ENABLE
180 PRINT : : : :"CRASH"
190 CALL SOUND(100,440,2)
200 PRINT "BEEP"
210 SUB ENABLE
220 CALL LOAD(-31806,0)::SUBEND
230 SUB DISABLE
240 CALL LOAD(-31806,32)::SUBEND
```

I am sure you know what the remaining lines do. Try halving the durations. Try different volumes. Try 02000700 instead of 0201FF00 in line 160. What happens now? Can you think of other experiments?

Take a look at Listing 2. Yes, I know, it is a bit long, but it is that length for a good reason. Type it in if you like. It sounds good.

Notice the Clicks that come with regularity through the music? That is because CALL SOUND puts the desired note(s) in the voices, and after the specified duration, turns them off again. Which explains why, even though the succeeding CALL SOUNDS play the previous notes for a reverberative effect, there is still a very steady pulse and those occasional Clicks. By the way, they are only obvious when the notes are double the normal length. The music it plays should be on the front cover of this magazine, although it may not be, due to circumstances at this point in time beyond my control. As I should rightly cite my sources, it is the introduction to a song called "Fool's Gold" from Petra's "Back To The Street" album. (Talk to me if you want to hear the whole song properly.)

But, back to the program. If you typed SIZE, and then a simple calculation, you will discover that it is 4288 bytes long. That is just over 4K for just the introduction!

Now have a look at Listing 3. This program plays the same piece of music with sound lists. Sounds better, too. You might care to find out just how much shorter than Listing 2 it is. Beware, however, that there is more overhead in Listing 3.

You may have noticed all the numbers on the front cover (Yes! I am banking heavily that it will be printed!), three digits to a note and another underneath. Well, I prepared a sheet by way of a photocopy that would be copied again to be the front page contribution, and had written on the music all the direct sound frequency data. Deliberately. And the voice numbers. Something like that is positively invaluable for sound list programming, otherwise you are trying to look at three or four things at once. Heavens, it is bad enough referencing frequency tables when programming ordinary music (I mean with CALL SOUND) so to do it with sound lists...! That is just asking for trouble.

Oh dear, now I have to explain myself. I really did use the altered music to program Listing 3. Contrariwise, I used the frequency list in the back of the Extended BASIC manual (Appendix D) along with the original music to do Listing 2. You can probably guess which is saner.

If you are lucky enough for your music to have a regularity to its (low level) structure, then that makes your sound list that much easier to program. The regularity of my music, is the reverberative effect: each note lasts through the next one or two. You very quickly learn the voice and volume names! If your music is rather different, for example, three independent lines, then it can become quite a chore, however. Programming sound lists definitely requires more concentration than using CALL SOUND. I know, I have done it!

Those with a musical bent may, especially upon examining the programs, notice that Listing 2 plays it in E Flat, whereas Listing 3 plays it in E Natural, which is as the music is written. There is a very good reason. (Just give me a moment to think of one...) When I play a Organ, Keyboard or Synthesizer, I do not generally like playing in Four Sharps (= the Key of E Natural). Consequently, I programmed Listing 2 in Three Flats (= E Flat), the key I sight transpose Four Sharps to. I had not realized, though, that the music would really be playing slightly different music to what was written, (partly because I hit upon the idea of having the music on this month's front page after writing Listing 2) so I had a little dilemma. It was easily solved when I noticed that the computer simply does not care what key it is in. Besides, doing it in Four Sharps gave me an advantage in using Frederick Hawkins Frequency table on page 12 of October '85 TND. And it was just as easy to write the frequency data in Four Sharps as Three Flats, so why not?

It was just as well I wrote the voice number under each note, because I got rather tangled up in Line 270 with the notes. However, when you do it, write the voice names instead. I know I should have. It makes it much easier to correct.

It is possible that I have totally confused someone by referring to the sound voices as Voice 0, Voice 1, Voice 2 and Voice 3/Noise. I thought it might be obvious why I did, but looking back, it might not be. When you use numbers to stand for things through BASIC or what is essentially TI99/4A written code, the numbering usually begins at one. This is a big departure from computing tradition, because everyone in computers knows that computers start counting at zero. Sometimes it makes sense, though. (I cannot imagine CSO, can you?) Usually, it does not really matter. Most computers start numbering their screen positions at (0,0) or just plain 0 if it is of the TRaSh-80 genre, although there are some exceptions (GW-BASIC's LOCATE statement). Colours, too, almost always start at zero on other computers. Not the TI99/4A. And I must admit, the fact that TI99/4A colours start at one can be a real pain at times. Oddly enough, the colour list on the Editor Assembler Card starts at zero. But then there is nothing to "go through" when writing colours to VDP RAM or otherwise, and the VDP chip starts its colour counting at zero, for obvious reasons.

But I am straying terribly.

The reason I started sound voice numbering at zero, is partly because nowhere in TI99/4Adom does one need to specify sound voice number. The other part is that the two bits that effectively designate exactly which voice is in question are binary 00, 01, 10 and 11. Pretty darn obvious, is it not??

I made note of Listing 4 earlier. In case you wondered, it is an Extended BASIC game with no help from Assembly save the sound list routine. It is, in fact, a good application of sound lists. If you are curious, the music that plays is not the same as Listings 2 or 3! (Ask me if you would like to know what it really is, though.) Released on an upcoming tape, hopefully this month, are all the Listings from this article. I hear sighs of relief! However, Listing 4 is not listed as "Listing 4", because there is another, much longer, and much much better program that plays exactly the same game anyhow. The difference is that it is up to a more professional standard, and Definitely Freeware! Besides which, it has a special surprise! What is it? You will have to buy the tape to find out, and if you think I am going to tell you how I wrote Listing 4, you are wrong! That remains a secret!

I CONCLUDE.

Now, if you want to comment about my article, there are several ways to go about it.

1> You can write me a letter. Mail it to:

WADE BOWMER
45 YANDERRA AVE.
BANGOR
NSW 2234

2> You can give me a call. Remember it must be after 7pm or on weekends or public holidays. Dial (02)5435180. That is Sydney, Australia for our overseas readers.

3> You can see me at the next meeting (June). I shall be holding a session about Extended BASIC.

4> You can write your own article. If you have a complete system, you can do it with TI-Writer or FunnelWeb Writer or one of a host of others. The Editor(s) will take it by disk or through the BBS. Just mark it "REPLY TO WADE BOWMER". (Joke! No really, only if you want to.)

You can also write one if you have nothing more than a console and a cassette. That is what I did. To give you some idea, This article is about 30K long in cassette programs. Remember to end each Paragraph with a } mark. If you have read the numerous TI-Writer articles, you may understand how to use the various commands. Remember that they must be on their own line.

I think that is all. Anyway, I am running out of tape.

Enjoy!

The above section entitled "ZOUNDS" contains several examples and subsequent encouragements for experimentation. The preceding explanations obviously do not exhaust the subject under consideration. That is the purpose of experimentation. It is not too hard to discover the little things not mentioned. It is very important to remember that the information is quite complete: it is not as though it will not all work or that there is something vital missing from the explanations. Rather, various explanations may seem to be a little brief or not involved enough. They will come to light readily upon experimentation.

Now, should you discover something, quite obviously peripheral, but nevertheless very relevant, do not be afraid of writing it up, and submitting an article, however short (or long), to TIsHUG.

As an added incentive, I am offering a prize of the sum total of programs I have submitted to TIsHUG (2 tapes/disks worth). They will consist of the contents of the September '87 disk "Wade's Programs", with version 4 of Graphic Designer, released Dec '88, and the contents of the disk "Tidbits 8/*Zounds*", to be released this month.

I will judge the winning article, provided you send them on tape or paper to the address above. Yes, I will see that they all get printed. The closing date is July 17. (I will accept entries postmarked that day, however.) One extension is possible if I receive fewer than seven entries. You may submit multiple entries, if you can. The competition will not proceed should I receive fewer than three entries. Those submitted, however, will be published.

I wish you success in your experiments!

Listing 2

```
10 !               LISTING 2    !
20 ! First of Programs to
30 ! Demonstrate Sound List
40 ! Advantages.
50 CALL CLEAR
60 DISPLAY "THIS PROGRAM PLAYS THE      INTRODUCTION
   FROM A PIECE OFMUSIC USING CALL SOUND.":
70 SQ=140 :: A=10 :: B=23 :: C=27 :: D=30 !TIMING
   (SEMI-QUAVER) AND VOLUME
80 CALL SOUND(SQ,262,A):: CALL SOUND(SQ,262,B,311,A)::
   CALL SOUND(SQ,262,C,311,B,349,A)::
   CALL SOUND(SQ,392,A,311,C,349,B)
90 !@P- THIS SAVES TIME
100 CALL SOUND(SQ,392,B,262,A,349,C)::
    CALL SOUND(SQ,392,C,262,B,311,A)::
    CALL SOUND(SQ,349,A,262,C,311,B)::
    CALL SOUND(SQ,349,B,392,A,311,C)
110 CALL SOUND(SQ,349,C,392,B,262,A)::
    CALL SOUND(SQ,311,A,392,C,262,B)::
    CALL SOUND(SQ,311,B,349,A,262,C)::
    CALL SOUND(SQ,311,C,349,B,392,A)
120 CALL SOUND(SQ,262,A,349,C,392,B)::
    CALL SOUND(SQ,262,B,311,A,392,C)::
    CALL SOUND(SQ,262,C,311,B,349,A)::
    CALL SOUND(SQ,233,A,311,C,349,B)
130 CALL SOUND(SQ,233,B,156,A,349,C)::
    CALL SOUND(SQ,233,C,156,A)::
    CALL SOUND(SQ,208,A,156,B)::
    CALL SOUND(SQ,208,B,156,C,233,A)
140 CALL SOUND(SQ,208,C,156,A,233,B)::
    CALL SOUND(SQ,311,A,156,B,233,C)::
    CALL SOUND(SQ,311,B,156,C,208,A)::
    CALL SOUND(SQ,311,C,233,A,208,B)
150 CALL SOUND(SQ,175,A,233,B,208,C)::
    CALL SOUND(SQ,175,A,233,C)::
    CALL SOUND(SQ,175,B,233,A)::
    CALL SOUND(SQ,175,C,233,B,262,A)
160 CALL SOUND(SQ,175,D,233,C,262,A)::
    CALL SOUND(SQ,311,A,233,D,262,B)::
    CALL SOUND(SQ,311,B,233,A,262,C)::
    CALL SOUND(SQ,311,C,233,B,262,A)
```

```
170 CALL SOUND(SQ,196,A,233,C,262,B)::
    CALL SOUND(SQ,196,A,233,D,262,C)::
    CALL SOUND(SQ,196,B,262,A)::
    CALL SOUND(SQ,196,C,262,B,294,A)
180 CALL SOUND(SQ,196,D,262,C,294,A)::
    CALL SOUND(SQ,311,A,262,D,294,B)::
    CALL SOUND(SQ,311,B,262,A,294,C)::
    CALL SOUND(SQ,311,C,262,B,294,A)
190 CALL SOUND(SQ,233,A,262,C,294,B)::
    CALL SOUND(SQ,233,A,262,D,294,C)::
    CALL SOUND(SQ,233,B,311,A)::
    CALL SOUND(SQ,233,C,311,B,349,A)
200 CALL SOUND(SQ,233,D,311,C,349,A)::
    CALL SOUND(SQ,262,A,311,D,349,B)::
    CALL SOUND(SQ,262,B,262,A,349,C)::
    CALL SOUND(SQ,262,C,262,B,233,A)
210 CALL SOUND(SQ,156,A,262,C,233,B)::
    CALL SOUND(SQ,156,A,262,D,233,C)::
    CALL SOUND(SQ,156,B,208,A)::
    CALL SOUND(SQ,156,C,208,B,233,A)
220 CALL SOUND(SQ,156,D,208,C,233,A)::
    CALL SOUND(SQ,311,A,208,D,233,B)::
    CALL SOUND(SQ,311,B,208,A,233,C)::
    CALL SOUND(SQ,311,C,208,B,233,A)
230 CALL SOUND(SQ,175,A,208,C,233,B)::
    CALL SOUND(SQ,175,A,208,D,233,C)::
    CALL SOUND(SQ,175,B,233,A)::
    CALL SOUND(SQ,175,C,233,B,262,A)
240 CALL SOUND(SQ,275,D,233,C,262,A)::
    CALL SOUND(SQ,311,A,233,D,262,B)::
    CALL SOUND(SQ,311,B,233,A,262,C)::
    CALL SOUND(SQ,311,C,233,B,233,A)
250 CALL SOUND(SQ,208,A,233,C,233,B)::
    CALL SOUND(SQ,208,A,233,D,233,C)::
    CALL SOUND(SQ,208,B,262,A)::
    CALL SOUND(SQ,208,C,262,B,233,A)
260 CALL SOUND(SQ,208,D,262,C,233,A)::
    CALL SOUND(SQ,311,A,262,D,233,B)::
    CALL SOUND(SQ,311,B,262,A,233,C)::
    CALL SOUND(SQ,311,C,262,B,233,A)
270 CALL SOUND(SQ,233,A,262,C,233,B)::
    CALL SOUND(SQ,233,A,262,D,233,C)::
    CALL SOUND(SQ,233,B,294,A)::
    CALL SOUND(SQ,233,C,294,B,233,A)
280 CALL SOUND(SQ,233,D,294,C,233,A)::
    CALL SOUND(SQ,311,A,294,D,233,B)::
    CALL SOUND(SQ,311,B,294,A,233,C)::
    CALL SOUND(SQ,311,C,294,B,233,A)
290 CALL SOUND(SQ,262,A,294,C,233,B)::
    CALL SOUND(SQ,262,B,311,A,233,C)::
    CALL SOUND(SQ,262,C,311,B,349,A)::
    CALL SOUND(SQ,392,A,311,C,349,B)
300 CALL SOUND(SQ,392,B,262,A,349,C)::
    CALL SOUND(SQ,392,C,262,B,311,A)::
    CALL SOUND(SQ,349,A,262,C,311,B)::
    CALL SOUND(SQ,349,B,392,A,311,C)
310 CALL SOUND(SQ,349,C,392,B,262,A)::
    CALL SOUND(SQ,311,A,392,C,262,B)::
    CALL SOUND(SQ,311,B,349,A,262,C)::
    CALL SOUND(SQ,311,C,349,B,392,A)
320 CALL SOUND(SQ,262,A,349,C,392,B)::
    CALL SOUND(SQ,262,B,311,A,392,C)::
    CALL SOUND(SQ,262,C,311,B,349,A)::
    CALL SOUND(SQ,392,A,311,C,349,B)
330 CALL SOUND(SQ,392,B,262,A,349,C)::
    CALL SOUND(SQ,392,C,262,B,311,A)::
    CALL SOUND(SQ,349,A,262,C,311,B)::
    CALL SOUND(SQ,349,B,392,A,311,C)
340 CALL SOUND(SQ,349,C,392,B,262,A)::
    CALL SOUND(SQ,311,A,392,C,262,B)::
    CALL SOUND(SQ,311,B,349,A,262,C)::
    CALL SOUND(SQ,311,C,349,B,392,A)
350 !@P+
360 FOR I=A TO D STEP 2.5 ::
    CALL SOUND(SQ,233,I,392,I,311,I):: NEXT I
370 PRINT :"NOW TYPE ""SIZE"" ENTER":"AND THEN ""PRINT
    24486-(2ND NUMBER)"" ENTER IF YOU HAVE 32K,"
380 PRINT " OR ""PRINT 13928-(NUMBER)""  IF YOU DON'T."
390 PRINT "THE RESULT SHOULD BE 4482."
```

<u>Listing 3</u>

```
10 !              LISTING 3   !
20 ! Second Program to show
30 ! the Memory Advantages
40 ! of Sound Lists.
100 DATA 03841995,049BA315B5,059EBBCE12D5,05BEDB8D1095
110 DATA 05DE9BA419B5,059EBBC315D5,05BEDB8E1295,
    05DE9BAD10B5
120 DATA 059EBBC419D5,05BEDB831595,05DE9BAE12B5,
    059EBBCD10D5
130 DATA 05BEDB841995,05DE9BA315B5,059EBBCE12D5,
    05BEDB851C95
140 DATA 05DE9BA72AB5,03DF9EB6,059FBBCC1FD5,05BEDB851C95
150 DATA 05DE9BA72AB5,059EBBC315D5,05BEDB8C1F95,
    05DE9BA51CB5
160 DATA 059EBBCD25D5,039FBED6,05BFDB851C95,05DE9BA419B5
170 DATA 03DF9EB6,059FBBC315D5,05BEDB851C95,05DE9BA419B5
180 DATA 059EBBCB21D5,039FBED6,05BFDB841995,05DE9BA816B5
190 DATA 03DF9EB6,059FBBC315D5,05BEDB841995,05DE9BA816B5
200 DATA 059EBBC51CD5,039FBED6,05BFDB831595,05DE9BAE12B5
210 DATA 03DF9EB6,059FBBC315D5,05BEDB831595,05DE9BA51CB5
220 DATA 059EBBC72AD5,039FBED6,05BEDB8C1F95,05DE9BA51CB5
230 DATA 03DF9EB6,059FBBC315D5,05BEDB8C1F95,05DE9BA51CB5
240 DATA 059EBBCD25D5,039FBED6,05BFDB851C95,05DE9BA419B5
250 DATA 03DF9EB6,059FBBC315D5,05BEDB851C95,05DE9BA51CB5
260 DATA 059EBBCC1FD5,039FBED6,05BFDB841995,05DE9BA51CB5
270 DATA 03DF9EB6,059FBBC315D5,05BEDB841995,05DE9BA51CB5
280 DATA 059EBBC51CD5,039FBED6,05BFDB881695,05DE9BA51CB5
290 DATA 03DF9EB6,059FBBC315D5,05BEDB881695,05DE9BA51CB5
300 DATA 059EBBC419D5,05BEDB831595,05DE9BAE12B5,
    059EBBCD10D5
310 DATA 05BEDB841995,05DE9BA315B5,059EBBCE12D5,
    05BEDB8D1095
320 DATA 05DE9BA419B5,059EBBC315D5,05BEDB8E1295,
    05DE9BAD10B5
330 DATA 059EBBC419D5,05BEDB831595,05DE9BAE12B5,
    059EBBCD10D5
340 DATA 05BEDB841995,05DE9BA315B5,059EBBCE12D5,
    05BEDB8D1095
350 DATA 05DE9BA419B5,059EBBC315D5,05BEDB8E1295,
    05DE9BAD10B5
360 DATA 059EBBC315D5,039FBED6,02BFD8,01DA
370 DATA 04DC8A3897,02DD99,02DE9B,02DF9D
1900 DATA 029FBF00,
2000 DISPLAY ERASE ALL:"THIS PROGRAM PLAYS THE SAME
    PIECE OF MUSIC AS LISTING 2, BUT WITH A SOUNDLIST.":
2005 PRINT :"NOTICE HOW MUCH SMOOTHER IT SOUNDS.": : :
2010 PRINT "  LOADING THE DATA,":"    WON'T BE LONG...":
    :"0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
    `abcdefghijklmnopqrstuvwxyz{|}~";
2020FOR I=127 TO 139 :: PRINT CHR$(I);:: NEXT I :: PRINT
    : : : : : : :: CALL INIT
2030 CALL DISABLE :: CH=48 !START AT "0"
2040 CALL LOAD(-31796,4,128,1)!SET SOUND LIST AND
    TRIGGER
2050 CALL LOAD(-31747,1)!IT'S IN VDP
2060 READ M$ :: C$=C$&M$&"08"
2070IF LEN(C$)>64 THEN CALL CHAR(CH,C$)::
    C$=SEG$(C$,65,64):: CH=CH+4 :: GOTO 2060
2080 IF M$="" THEN CALL CHAR(CH,C$)ELSE 2060
2090 CALL ENABLE
2100 FOR I=0 TO 14 :: CALL COLOR(I,4,13):: NEXT I ::
    CALL MAGNIFY(4):: CALL SPRITE(#1,48,16,40,50,1,4)
2110 FOR A=48 TO 139 :: CALL PATTERN(#1,A):: CALL
    COLOR(#1,13*RND+3,#1,13*RND+3) :: NEXT A
2120 CALL DELSPRITE(ALL):: CALL SOUND(10,440,30):: CALL
    CHAR(139,"")
2130 PRINT :"NOTICE HOW THE PROGRAM CAN BE BUSY DOING
    SOMETHING ELSEWHILE THE MUSIC PLAYS."
2140 PRINT :"EVEN WHILE SPRITES MOVE."
2200 SUB ENABLE :: CALL LOAD(-31806,0):: SUBEND
2210 SUB DISABLE :: CALL LOAD(-31806,32):: SUBEND
```

<u>Listing 4</u>

```
10 !               LISTING 4    !
20 ! Program to demonstrate
30 ! a Real Application for
40 ! Sound Lists.  (Among       ! other things!)
50 ! Simple game called
60 ! BOX HIM!
70 !
80 DISPLAY ERASE ALL :: CALL SCREEN(16):: FOR I=5 TO 8
    :: CALL COLOR(I,13,12):: NEXT I
```

```
90 CALL CHAR(40,"7E81A581BD99817E0",48,
   "7E81A581BD99817E0",56,"AA55AA55AA55AA55")::
   CALL COLOR(1,13,12,2,5,8,3,7,16,4,11,15)::
   CALL TITLE
100 CALL CLEAR :: R1,R2=12 :: C1=5 :: C2=24 :: D1,D2=1
   :: E1,E2=0 :: CALL MUSIC(1)
110 DISPLAY AT(R1,C1):"(";:: DISPLAY AT(R2,C2):"0";::
   CALL JOYST(1,X1,Y1):: CALL JOYST(2,X2,Y2)
120 IF X1=0 XOR Y1=0 THEN E1=SGN(X1):: D1=-SGN(Y1)
130 IF X2=0 XOR Y2=0 THEN E2=SGN(X2):: D2=-SGN(Y2)
140 CALL GCHAR(R1+D1,C1+E1+2,G1):: IF G1=32 THEN
   DISPLAY AT(R1,C1):")";:: R1=R1+D1 :: C1=C1+E1
150 CALL GCHAR(R2+D2,C2+E2+2,G2):: IF G2=32 THEN
   DISPLAY AT(R2,C2):"1";:: R2=R2+D2 :: C2=C2+E2
160 IF G1=32 AND G2=32 THEN 110
170 CALL MUSIC(0):: IF G2=32 THEN R=R1+D1 :: C=C1+E1
   ELSE R=R2+D2 :: C=C2+E2
180 FOR I=0 TO 20 STEP 3 :: CALL SOUND(-100,-6,I)::
   DISPLAY AT(R,C):" ";:: DISPLAY AT(R,C):"8";:: NEXT I
190 CALL DIZZY :: GOTO 100
200 SUB TITLE :: CALL CLEAR
210 DISPLAY AT(8,11):")))))))))";:: DISPLAY AT(9,10):"1
         )";:: DISPLAY AT(10,10):"1BOX HIM!)";
220 DISPLAY AT(11,10):"1        (";::
   DISPLAY AT(12,10):"111111110";
230 CALL MSET :: CALL DIZZY
240 SUBEND
250 SUB DIZZY :: RESTORE 290
260 FOR I=1 TO 18 :: READ F,B :: CALL COLOR(4,F,B)::
   CALL KEY(0,K,S):: IF K>0 THEN SUBEXIT
270 IF I/2=INT(I/2)THEN DISPLAY AT(24,9):"PRESS A
   KEY!";ELSE DISPLAY AT(24,9):"888888888888";
280 NEXT I :: RESTORE 290 :: GOTO 260
290 DATA 3,5,5,2,5,13,7,14,10,11,11,3,13,9,14,3,14,9
300 DATA 13,7,4,10,5,7,6,5,9,7,10,15,13,3,13,14,14,6
310 SUBEND
320 SUB MSET :: CALL CHAR(96,"03893F940A028A380A02851C
   0A028C1F0A028B210A02872A0A028D250A028A38")
330 CALL CHAR(100,"0A028B230A019F0101940902851C0A028C
   1F0A028B230A02872A0A028D250A02")
340 CALL CHAR(104,"893F0A019F01019409019F01019409019F
   01019409028A380A0287320A028A38")
350 CALL CHAR(108,"0A02893F0A019F01019409028A380A019F
   010194090287320A028A2F0A019F01")
360 CALL CHAR(112,"019409019F0101940902872A0A028D250A
   02893F0A028A380A02851C0A028C1F")
370 CALL CHAR(116,"0A028B210A02872A0A028D250A028A380A
   028B230A019F0101940902851C0A02")
380 CALL CHAR(120,"8C1F0A028B230A02872A0A028D250A0289
   3F0A019F01019409019F0101940901")
390 CALL CHAR(124,"9F01019409028A380A0287320A028A380A
   02893F0A019F01019409058C1FAE0F")
400 CALL CHAR(128,"B71404851CA20E14048C1FAE0F0A04851C
   A20E0A04872AA3150A048D25AE1209")
410 CALL CHAR(132,"0006000000000000957E64CA63D63F1A",
   136,"AA55AA55AA55AA55")
420 CALL CHAR(138,"000808080808080000000007E")
430 CALL CHAR(140,"D6D6FEFEFEFEC6C6FFFF3C3F3CFFFF00C6
   C6FEFEFEFED6D6FFFF3CFC3CFFF F"):: SUBEND
440 SUB MUSIC(CMD):: IF CMD=0 THEN CALL LOAD(-31794,0)::
   CALL LOAD(-31744,159,"",-31744,191)::
   CALL CHAR(96,"03893F940A028A38"):: SUBEXIT
450 IF CMD=2 THEN CALL LOAD(-31806,48)::
   CALL CHAR(132,"029FBF"):: CALL LOAD(-31806,16)ELSE
   480
460 CALL PEEK(-31794,A):: IF A THEN 460
470 CALL CHAR(132,"0006",96,"03893F940A028A38")::
   SUBEXIT
480 IF CMD<>1 THEN SUBEXIT
490 CALL LOAD(-31747,1):: CALL LOAD(-31796,6,0,1)::
   CALL LOAD(-31806,16)::
   CALL CHAR(96,"03893FBF0A028A38"):: SUBEND
500 SUB CLEAR :: DISPLAY ERASE ALL AT(24,1):
   RPT$("8",56):: CALL VCHAR(1,30,56,144)::
   CALL VCHAR(1,31,31,96):: SUBEND
```

<u>Listing 5</u>

```
10 ! BOX HIM!
20 !
30 ! A simple painting game
40 ! using Sound Lists and
50 ! Modular Programming
60 ! Techmiques.
80 DISPLAY ERASE ALL :: CALL SCREEN(16):: FOR I=5 TO 8
   :: CALL COLOR(I,13,12):: NEXT I
90 CALL CHAR(40,"7E81A581BD99817E0",48,
   "7E81A581BD99817E0",56,"AA55AA55AA55AA55")::
   CALL COLOR(1,13,12,2,5,8,3,7,16,4,11,15)::
   CALL TITLE
100 CALL CLEAR :: R1,R2=12 :: C1=5 :: C2=24 :: D1,D2=1
   :: E1,E2=0 :: P1$=")" :: P2$="1" :: CALL MUSIC(1)
110 DISPLAY AT(R1,C1):"(";:: DISPLAY AT(R2,C2):"0";::
   CALL JOYST(1,X1,Y1):: CALL JOYST(2,X2,Y2)
120 IF X1=0 XOR Y1=0 THEN E1=SGN(X1):: D1=-SGN(Y1)
130 IF X2=0 XOR Y2=0 THEN E2=SGN(X2):: D2=-SGN(Y2)
140 CALL KEY(1,K1,S1):: IF K1=18 AND P1$<>" " THEN P1$="
   " ELSE P1$=")"
150 DISPLAY AT(R1,C1):P1$;:: R1=R1+D1 :: C1=C1+E1 ::
   CALL GCHAR(R1,C1+2,G1) 160 CALL KEY(2,K2,S2):: IF
   K2=18 AND P2$<>" " THEN P2$=" " ELSE P2$="1"
170 DISPLAY AT(R2,C2):P2$;:: R2=R2+D2 :: C2=C2+E2 ::
   CALL GCHAR(R2,C2+2,G2)
180 IF G1=32 AND G2=32 THEN 110
190 CALL MUSIC(0):: IF G2=32 THEN R=R1 :: C=C1 ELSE R=R2
   :: C=C2
200 FOR I=0 TO 20 STEP 3 :: CALL SOUND(-100,-6,I)::
   DISPLAY AT(R,C):" ";:: DISPLAY AT(R,C):"8";:: NEXT I
210 CALL DIZZY :: GOTO 100
220 SUB TITLE :: CALL CLEAR
230 DISPLAY AT(4,12):"WADE'S";::
   DISPLAY AT(5,11):"PROGRAMS";
240 DISPLAY AT(7,11):")))))))))";:: DISPLAY AT(8,10):"1
         )";:: DISPLAY AT(9,10):"1BOX HIM!)";
250 DISPLAY AT(10,10):"1        (";::
   DISPLAY AT(11,10):"111111110";
260 DISPLAY AT(14,6):"USE THE JOYSTICKS";::
   DISPLAY AT(15,5):"TO DRAW LINES AROUND";::
   DISPLAY AT(16,8):"YOUR OPPONENT";
270 DISPLAY AT(18,5):"PRESS THE TRIGGER TO";::
   DISPLAY AT(19,8):"LEAVE A HOLE!";
280 DISPLAY AT(21,3):"DONATIONS TO WADE BOWMER";::
   DISPLAY AT(22,5):"C\0 TIsHUG [AUS] LTD";
290 CALL MSET :: CALL DIZZY
300 SUBEND
310 SUB DIZZY :: RESTORE 350
320 FOR I=1 TO 18 :: READ F,B :: CALL COLOR(4,F,B)::
   CALL KEY(0,K,S):: IF K>0 THEN SUBEXIT
330 IF I/2=INT(I/2)THEN DISPLAY AT(24,9):"PRESS A
   KEY!";ELSE DISPLAY AT(24,9):"888888888888";
340 NEXT I :: RESTORE 350 :: GOTO 320
350 DATA 3,5,5,2,5,13,7,14,10,11,11,3,13,9,14,3,14,9
360 DATA 13,7,4,10,5,7,6,5,9,7,10,15,13,3,13,14,14,6
370 SUBEND
380 SUB MSET :: CALL CHAR(96,"03893F940A028A380A02851
   C0A028C1F0A028B210A02872A0A028D250A028A38")
390 CALL CHAR(100,"0A028B230A019F0101940902851C0A028C
   1F0A028B230A02872A0A028D250A02")
400 CALL CHAR(104,"893F0A019F01019409019F01019409019F
   01019409028A380A0287320A028A38")
410 CALL CHAR(108,"0A02893F0A019F01019409028A380A019F
   010194090287320A028A2F0A019F01")
420 CALL CHAR(112,"019409019F0101940902872A0A028D250A
   02893F0A028A380A02851C0A028C1F")
430 CALL CHAR(116,"0A028B210A02872A0A028D250A028A380A
   028B230A019F0101940902851C0A02")
440 CALL CHAR(120,"8C1F0A028B230A02872A0A028D250A0289
   3F0A019F01019409019F0101940901")
450 CALL CHAR(124,"9F01019409028A380A0287320A028A380A
   02893F0A019F01019409058C1FAE0F")
460 CALL CHAR(128,"B71404851CA20E14048C1FAE0F0A04851C
   A20E0A04872AA3150A048D25AE1209")
470 CALL CHAR(132,"0006000000000000957E64CA63D63F1A",
   136,"AA55AA55AA55AA55")
480 CALL CHAR(138,"000808080808080000000007E")
490 CALL CHAR(140,"D6D6FEFEFEFEC6C6FFFF3C3F3CFFFF00C6
   C6FEFEFEFED6D6FFFF3CFC3CFFFF"):: SUBEND
500 SUB MUSIC(CMD):: IF CMD=0 THEN CALL LOAD(-31794,0)::
   CALL LOAD(-31744,159,"",-31744,191)::
   CALL CHAR(96,"03893F940A028A38"):: SUBEXIT
510 IF CMD=2 THEN CALL LOAD(-31806,48)::
   CALL CHAR(132,"029FBF"):: CALL LOAD(-31806,16)ELSE
   540
520 CALL PEEK(-31794,A):: IF A THEN 520
530 CALL CHAR(132,"0006",96,"03893F940A028A38")::
   SUBEXIT
540 IF CMD<>1 THEN SUBEXIT
```

```
550 CALL LOAD(-31747,1):: CALL LOAD(-31796,6,0,1)::
    CALL LOAD(-31806,16)::
    CALL CHAR(96,"03893FBF0A028A38"):: SUBEND
560 SUB CLEAR :: DISPLAY ERASE ALL AT(24,1):
    RPT$("8",56):: CALL VCHAR(1,30,56,144)::
    CALL VCHAR(1,31,31,96):: SUBEND
```

Listing 6

```
10 ! EXPLOSION !
20 RANDOMIZE
30 DIM RB(7,7),SB(7,7),ST(7,7),SX(6),SY(6)
40 GOTO 1630
50 FOR X=1 TO SI :: FOR Y=1 TO SI :: SB(X,Y)=RB(X,Y)::
    NEXT Y :: NEXT X :: RETURN
110 FOR X=1 TO SI :: FOR Y=1 TO SI :: RB(X,Y)=SB(X,Y)::
    NEXT Y :: NEXT X :: RETURN
120 CALL CLS :: CALL SCREEN(15):: FOR I=1 TO 8 ::
    CALL COLOR(I,7,16):: NEXT I ::
    CALL VCHAR(1,31,31,96)
130 CALL COLOR(9,8,16,13,3,16,14,9,16):: FOR I=0 TO 5 ::
    READ A$ :: CALL CHAR(136-I,A$,136+I,A$):: NEXT I
140 DATA ,0000001818,0000006666,0018180000666 6,
    00666600006666,00666618186666
145 CALL CHAR(96,"181818FFFF18181818181818181800
    0000FFFF")
150 CALL CHAR(120,"0102040810204080804020100804020180
    4020100804020101020408102040 8")
155 CALL CHAR(124,"00003F20202020202020202020203F000000
    00FC04040404040404040404FC" )
160 FOR I=1 TO 6 :: READ SX(I),SY(I):: NEXT I ::
    CALL SPRITE(#1,124,5,193,128):: CALL MAGNIFY(3)::
    MX,MY=1
170 DATA 93,37,109,53,125,69,141,85,157,101,173,117
180 CALL TITLE :: DISPLAY AT(12,2):"HOW BIG THE BOARD- 3
    - 6 ?"
190 CALL KEY(3,K,S):: IF K<51 OR K>54 THEN 190 ELSE
    SI=K-48
210 FOR X=1 TO SI :: FOR Y=1 TO SI ::
    ST(X,Y)=4+(X=1)+(X=SI)+(Y=1)+(Y=SI):: NEXT Y :: NEXT
    X
260 DISPLAY AT(12,2):" WHO TO START- YOU OR ME?"
270 CALL KEY(3,K,S):: IF K<>89 AND K<>77 THEN 270 ELSE
    CM=K=89
280 CALL CLS
330 FOR Y=0 TO SI ::
    DISPLAY AT(2*Y+5,10):RPT$("`b",SI)&"`" ::
    DISPLAY AT(2*Y+6,10):RPT$("a ",SI)&"a" :: NEXT Y ::
    DISPLAY AT(2*Y+4,10)
440 RETURN
450 ! INPUT
460 DISPLAY AT(17,5):"IT'S YOUR TURN." :: IF NT=1 THEN
    CALL SAY("YOUR+TURN")ELSE CALL SAY(".;")
470 CALL LOCATE(#1,SY(MY),SX(MX))
480 CALL KEY(1,K,S):: IF K=18 THEN 610
490 CALL JOYST(1,JX,JY):: X=MX+JX/4 :: Y=MY-JY/4 :: IF
    X<1 OR X>SI OR Y<1 OR Y>SI THEN 480 ELSE MX=X ::
    MY=Y :: DISPLAY AT(17,1):: GOTO 470
610 IF RB(MX,MY)<0 THEN DISPLAY AT(17,5):"THAT ISN'T
    YOURS!" :: GOTO 480 ELSE
    DISPLAY AT(17,5):"THANKS..." :: GOSUB 50
630 X=MX :: Y=MY :: DI=-1 :: GOSUB 700
670 IF ET THEN 1570 ELSE 110
700 ! EXPLOSION
710 SB(X,Y)=SB(X,Y)+PL :: NE=0 :: IF DI THEN
    CALL PIECE(X,Y,SB(,))
760 XP=0 :: FOR X=1 TO SI :: FOR Y=1 TO SI :: IF
    ABS(SB(X,Y))<ST(X,Y)THEN 980
800 XP=-1 :: NE=NE+1 :: SB(X,Y)=SB(X,Y)-ST(X,Y)*PL
820 IF DI THEN CALL SOUND(-100,-5,18-NE/2)::
    CALL PATTERN(#1,120):: CALL LOCATE(#1,SY(Y),SX(X))::
    CALL PIECE(X,Y,SB(,))
860 CALL COLOR(#1,2,#1,5):: EX=X :: EY=Y+1 :: GOSUB 1040
910 CALL COLOR(#1,2,#1,5):: EX=X+1 :: EY=Y :: GOSUB 1040
920 CALL COLOR(#1,2,#1,5):: EX=X :: EY=Y-1 :: GOSUB 1040
930 CALL COLOR(#1,2,#1,5):: EX=X-1 :: EY=Y :: GOSUB 1040
980 NEXT Y :: NEXT X :: CALL PATTERN(#1,124)
1000 ET=(NE>SI*SI):: IF ET THEN RETURN ELSE IF XP THEN
    760 ELSE RETURN
1030 RETURN
1040 ! ADD SURROUNDS
1050 SB(EX,EY)=PL*(ABS(SB(EX,EY))+1)
1090 IF DI AND ST(EX,EY)>0 THEN CALL PIECE(EX,EY,SB(,))
1100 RETURN
1110 ! COMPUTER
1120 BE=1000 :: DISPLAY AT(17,5):"THINKING..." :: FOR
    TX=1 TO SI :: FOR TY=1 TO SI ::
    CALL LOCATE(#1,SY(TY),SX(TX)):: IF RB(TX,TY)>0 THEN
    1320
1160 GOSUB 50 :: IF RB(TX,TY)=0 THEN 1250
1170 X=TX :: Y=TY :: DI=0 :: GOSUB 700 :: IF ET THEN
    MX=TX :: MY=TY :: DISPLAY AT(24,1):: GOTO 1340
1250 GOSUB 1430 :: IF EN>BE THEN 1320
1280 IF EN=BE AND RND>.5 THEN 1320
1290 BE=EN :: MX=TX :: MY=TY
1320 NEXT TY :: NEXT TX :: DISPLAY AT(17,1)
1340 ! ACTUAL MOVE
1350 GOSUB 50 :: X=MX :: Y=MY ::
    CALL LOCATE(#1,SY(Y),SX(X)):: DI=-1 :: GOSUB 700 ::
    IF ET THEN 1570 ELSE 110
1430 ! EVALUATE
1440 EN=0 :: FOR X=1 TO SI :: FOR Y=1 TO SI ::
    EN=EN+SB(X,Y):: IF -SB(X,Y)<ST(X,Y)-1 THEN 1540
1490 EN=EN-2
1500 IF SB(X+1,Y)=ST(X+1,Y)-1 THEN EN=EN+10
1510 IF SB(X,Y+1)=ST(X,Y+1)-1 THEN EN=EN+10
1520 IF SB(X-1,Y)=ST(X-1,Y)-1 THEN EN=EN+10
1530 IF SB(X,Y-1)=ST(X,Y-1)-1 THEN EN=EN+10
1540 NEXT Y :: NEXT X :: RETURN
1570 ! END...
1580 CALL DELSPRITE(#1):: PRINT "AN ETERNAL EXPOLSION
        RESULTED, SO ";
1590 IF PL=1 THEN PRINT "YOU";ELSE PRINT "I";
1600 PRINT " HAVE WON IN";NT;"TURNS." :: IF PL=1 THEN
    CALL SAY("#YOU WIN")ELSE CALL SAY("#I WIN")
1610 PRINT "[Y] TO PLAY AGAIN..."
1615 CALL KEY(3,K,S):: IF K=78 THEN END ELSE IF K<>89
    THEN 1615
1620 RUN 1630 ! MAINLINE
1630 GOSUB 120 :: IF CM THEN 1670
1650 PL=1 :: GOSUB 450
1670 PL=-1 :: GOSUB 1110 :: NT=NT+1 :: GOTO 1650
1710 SUB PIECE(A,B,SB(,))
1720 DISPLAY AT(2*B+4,2*A+9):CHR$(136+SB(A,B));:: SUBEND
2000 SUB TITLE :: DISPLAY AT(2,5):"`b`b`b`b`b`b`b`b`b`"
    :: DISPLAY AT(3,5):"a a a a a a a a a a" ::
    DISPLAY AT(4,5):"`b`b`b`b`b`b`b`b`b`"
2010 DISPLAY AT(6,3):"YOU PLAY THE COMPUTER
    ON":TAB(5);"A VARYING SIZE BOARD":TAB(7);"FROM 3X3
    TO 6X6."
2020 DISPLAY AT(10,2):"EACH SQUARE HAS A
    CAPACITY":TAB(3);"EQUAL TO THE NUMBER OF":TAB(6);
    "SQUARES ADJOINING."
2030 DISPLAY AT(14,4):"WHEN YOU FILL A
    SQUARE":TAB(7);"IT EXPLODES!"
2040 DISPLAY AT(17,3):"TO WIN, GET THE BOARD TO":"
    EXPLODE ALL IN YOUR COLOUR"
2050 DISPLAY AT(21,3):"DONATIONS TO WADE BOWMER":" C/-
    TIsHUG (AUS.) LTD."
2060 DISPLAY AT(3,6):"EaXaPaLaOaSaIaOaN";::
    CALL KEY(0,K,S):: IF S THEN CALL CLS :: SUBEXIT
2070 DISPLAY AT(3,6):" a a a a a a a a ";::
    CALL KEY(0,K,S):: IF S=0 THEN 2060 2080 CALL CLS ::
    SUBEND
32000 SUB CLS :: CALL VCHAR(1,3,32,672):: SUBEND
```

Listing 7

```
10 !*** |\/| INI              !*** |\/| ANCALA
20 ! GRAPHICALLY IMPROVED          BY WADE BOWMER
          FROM THE BOOK
30 !"TERRIFIC GAMES FOR THE        TI-99/4A" BY
40 ! H.RENKO AND S.EDWARDS
50 !
60 !SU/19/03/89
70 DIM M(1,1),L(3,1),R(1,1),C(1,1)
80 GOTO 360
90 FOR I=0 TO 1 :: M(I,0),M(I,1)=2 :: READ
    R(I,0),R(I,1),C(I,0),C(I,1)
100 FOR J=0 TO 3 :: READ L(J,I):: NEXT J :: NEXT I
110 DATA 9,9,12,17,0,1,1,0,14,14,12,17,0,0,1,1
120 CALL CLEAR :: CALL SCREEN(15):: FOR I=1 TO 4 ::
    CALL COLOR(I,2,16,I+4,7,16,I+8,5,16):: NEXT I ::
    CALL COLOR(0,15,15,4,16,16)
```

```
130 FOR I=0 TO 7 :: READ A$,B$ ::
    CALL CHAR(I+48,B$,I+96,A$,I+104,A$,I+112,A$,I+120,A$)::
    NEXT I
140 CALL CHAR(56,"DBDB00C3C300DBDB"):: CALL MAGNIFY(2)::
    CALL SPRITE(#1,77,9,8,97):: CALL VCHAR(1,31,31,96)
150 DISPLAY AT(2,13):"INI":TAB(13);"ANCALA"
156 CALL SOUND(2000,131,6,131.8,6):: A$="9<48=7>52?6;:"
    :: A$=A$&A$&A$&A$&"7" :: FOR I=1 TO LEN(A$)::
    CALL COLOR(#1,ASC(SEG$(A$,I,1))AND 15):: NEXT I
160 FOR I=8 TO 16 :: READ A$ :: DISPLAY AT(I,11):A$ ::
    NEXT I
170 DISPLAY AT(18,1):: DISPLAY AT(20,1):: RETURN
180 DISPLAY AT(18,3):" MOVE STONES FROM?" ::
    CALL COLOR(4,13,16):: CALL SOUND(70,390,6)::
    CALL SOUND(80,260,6)
190 CALL KEY(0,K,S):: IF S=0 THEN 190 ELSE
    LB=INT(POS(" ,<.>",CHR$(K),1)/2):: IF LB=0 THEN 190
200 CALL COLOR(10,5,16,11,5,16):: IF M(1,LB-1)=0 THEN
    DISPLAY AT(20,6):"THIS CUP IS EMPTY" :: GOTO 180
210 CALL COLOR(10+LB,13,16,4,16,16):: DISPLAY AT(18,1)::
    DISPLAY AT(20,1):: RETURN
220 CALL SOUND(100,220,6):: DISPLAY AT(18,6):"NOW IT IS
    MY TURN"
230 LB=3 :: Q=1000*M(0,0)+100*M(0,1)+10*M(1,0)+M(1,1)
240 IF M(0,0)=0 THEN 270
250 IF M(0,1)=0 THEN LB=0 :: GOTO 270
260 IF (M(0,0)>0 AND LEV<3 AND RND*LEV<.4)OR M(0,1)=6 OR
    Q=1430 OR Q=1340 OR Q=6110 OR Q=1160 THEN LB=0
270 DISPLAY AT(20,5):"I MOVE FROM THIS CUP" ::
    CALL SOUND(500,440,30):: CALL COLOR(10+(LB=0),13,16)
280 RETURN
290 G=L(LB,0):: H=L(LB,1)
300 IF M(G,H)=0 THEN
    CALL COLOR(9,5,16,10,5,16,11,5,16,12,5,16):: GOTO
    170
310 LB=LB+1 :: IF LB>3 THEN LB=LB-4
320 I=L(LB,0):: J=L(LB,1)
330 M(G,H)=M(G,H)-1 :: M(I,J)=M(I,J)+1
340 CALL SOUND(10,-5,6)::
    DISPLAY AT(R(G,H),C(G,H)):STR$(M(G,H));::
    DISPLAY AT(R(I,J),C(I,J)):STR$(M(I,J));
350 GOTO 300
360 GOSUB 90
370 DISPLAY AT(18,3):"ENTER LEVEL (123) 2" :: ACCEPT
    AT(18,21)VALIDATE("123")SIZE(-1)BEEP:X$ :: IF X$=""
    THEN 370 ELSE LEV=VAL(X$)
380 DISPLAY AT(18,3):"WOULD YOU LIKE TO START?" ::
    CALL SOUND(70,390,6):: CALL SOUND(80,260,6)
390 CALL KEY(3,K,S):: IF K=78 THEN 410 ELSE IF K<>89
    THEN 390
400 GOSUB 180 :: GOSUB 290
410 IF M(1,1)=8 THEN DISPLAY AT(18,9):"YOU WIN!" ::
    CALL SAY("#YOU WIN"):: GOTO 440
420 GOSUB 220 :: GOSUB 290
430 IF M(0,0)=8 THEN DISPLAY AT(18,10):"I WIN!" ::
    CALL SAY("#I WIN")ELSE 400
440 DISPLAY AT(16,1):: FOR I=1 TO 3 ::
    CALL COLOR(4,16,16,4,7,16):: NEXT I ::
    DISPLAY AT(20,9):"PLAY AGAIN?"
450 CALL SOUND(500,440,30)
460 CALL KEY(3,K,S):: IF K=89 THEN RESTORE :: GOTO 360
470 IF K<>78 THEN 460 ELSE CALL CLEAR
480 DATA 00003F3F30373434,,3434343434343434,0000001818,
    3434343730 3F3F,0000006666,000000FF00FFFF,
    C0C0001818000303
490 DATA 2C2C2CEC0CFCFC,C3C300000000C3C3,
    2C2C2C2C2C2C2C2C,C3C300181800C3C3,0000FCFC0CEC2C2C,
    C3C300C3C300C3C3,0000FFFF00FF,C3C300DBDB00C3C3
500 DATA "`gf  hon","a2e  i2m","bcd  jkl",,,
~","q2u  y2}","rst  z{|","    >"
```

<u>Listing 8</u>

```
10 CALL CLEAR :: CALL SCREEN(16):: FOR I=0 TO 8 ::
   CALL COLOR(I,13,16):: NEXT I
20 RANDOMIZE :: CALL PEEK(-31880,B):: LIMIT=10
30 DISPLAY AT(08,3):"I HAVE A NUMBER BETWEEN":TAB(10);"0
   AND 100" :: CALL SAY(
   "I+HAVE+A+NUMBER+BETWEEN+ZERO+AND+A1+HUNDRED")
40 DISPLAY AT(11,6):"TRY TO GUESS IT." ::
   CALL SAY("TRY+TO+GUESS+IT"):: DISPLAY ERASE ALL
50 IF TRIES=LIMIT THEN 390 ELSE DISPLAY AT(10,1):"YOUR
   GUESS->";N :: TRIES=TRIES+1
60 IF TRIES=1 THEN CALL SAY("YOUR+GUESS")
70 ACCEPT AT(10,14)VALIDATE(DIGIT)SIZE(-2):N$
80 RANDOMIZE :: CALL PEEK(-31808,L):: IF N$="" THEN ON
   L*.0234+1 GOTO 130,140,150,160,170,180,160
90 N=VAL(N$):: D=ABS(N-B):: IF D>30 THEN ON L*.01953+1
   GOTO 190,200,210,220,150,350
100 IF N>B THEN ON D*.267+1
    GOTO 280,270,260,250,240,230,350,360,150
110 IF N<B THEN ON D*.267+1
    GOTO 340,330,320,310,300,290,350,360,150
120 IF N=B THEN 550
130 DISPLAY AT(12,1):"GO ON, HAVE A GO!" ::
    CALL SAY("GO+ON-HAVE+A1+GO"):: GOTO 50
140 DISPLAY AT(12,1):"WHAT IS YOUR NUMBER?" ::
    CALL SAY("WHAT+IS+YOUR+NUMBER"):: GOTO 50
150 DISPLAY AT(12,1):"WAY OFF!" :: CALL SAY("WAY+OFF")::
    GOTO 50
160 DISPLAY AT(12,1):"WHAT WAS THAT?" :: CALL SAY("#WHAT
    WAS THAT"):: GOTO 50
170 IF TRIES<3 THEN DISPLAY AT(12,1):"TRY ""50""" ::
    CALL SAY("TRY+FIFTY"):: GOT 0 50
180 DISPLAY AT(12,1):"THAT IS NOT A NUMBER" ::
    CALL SAY("THAT+IS+NOT+A1+NUMBER") :: GOTO 50
190 DISPLAY AT(12,1):"YOU'RE A LONG WAY OUT" ::
    CALL SAY("YOUR+A1+LONG+WAY+OUT") :: GOTO 50
200 DISPLAY AT(12,1):"OFF COURSE" ::
    CALL SAY("OFF+COURSE"):: GOTO 50
210 DISPLAY AT(12,1):"NOWHERE NEAR IT" ::
    CALL SAY("NO+WHERE+NEAR+IT"):: GOTO 50
220 DISPLAY AT(12,1):"TRY A VERY DIFFERENT NUMBER" ::
    CALL SAY("TRY+A1+VERY+DIFFERENT+NUMBER"):: GOTO 50
230 DISPLAY AT(12,1):"COME LOWER." :: CALL SAY("COME
    LOWER"):: GOTO 50
240 DISPLAY AT(12,1):"YOU'RE TOO LARGE!" ::
    CALL SAY("YOUR+TOO+LARGE"):: GOTO 50
250 DISPLAY AT(12,1):"GO DOWN SOME." ::
    CALL SAY("GO+DOWN+SOME"):: GOTO 50
260 DISPLAY AT(12,1):"YOU'RE GETTING THERE!" ::
    CALL SAY("YOUR+GETTING+THERE"):: GOTO 50
270 DISPLAY AT(12,1):"SMALLER..." ::
    CALL SAY("SMALLER"):: GOTO 50
280 IF L>128 THEN DISPLAY AT(12,1):"NO HELP!" ::
    CALL SAY("NO+HELP"):: GOTO 50 ELSE 340
290 DISPLAY AT(12,1):"WAY TOO SMALL!" ::
    CALL SAY("WAY+TOO+SMALL"):: GOTO 50
300 DISPLAY AT(12,1):"MOVE HIGHER." ::
    CALL SAY("MOVE+HIGHER"):: GOTO 50
310 DISPLAY AT(12,1):"COME UP MORE." ::
    CALL SAY("COME+UP+MORE"):: GOTO 50
320 DISPLAY AT(12,1):"LARGER" :: CALL SAY("LARGER")::
    GOTO 50
330 DISPLAY AT(12,1):"VERY NEAR" ::
    CALL SAY("VERY+NEAR"):: GOTO 50
340 IF L<129 THEN DISPLAY AT(12,1):"UHOH!" ::
    CALL SAY("UHOH"):: GOTO 50 ELSE 28 0
350 DISPLAY AT(12,1):"THAT IS INCORRECT" ::
    CALL SAY("#THAT IS INCORRECT"):: GOTO 50
360 DISPLAY AT(12,1):"TRY AGAIN..." :: CALL SAY("#TRY
    AGAIN"):: GOTO 50
390 DISPLAY AT(8,1):"OUT OF TIME" ::
    CALL SAY("OUT+OF+TIME")
400 DISPLAY AT(10,1):"THE NUMBER IS";B ::
    TENS=INT(B/10):: UNITS=B-10*TENS
410 CALL SAY("THE1+NUMBER+IS")
420 IF TENS=0 THEN TEN$="" :: GOTO 450
430 IF TENS=1 THEN TEN$="" :: RESTORE 520 :: GOTO 450
440 FOR I=1 TO TENS :: READ TEN$ :: NEXT I :: RESTORE
    510
450 FOR I=1 TO UNITS :: READ UNIT$ :: NEXT I ::
    CALL SAY(TEN$,,UNIT$)
460 DISPLAY AT(12,1):"WILL YOU GO AGAIN?" ::
    CALL SAY("WILL+YOU+GO+AGAIN")
470 FOR K=0 TO 1 :: CALL KEY(3,K,S):: NEXT K :: IF K=90
    THEN RUN ELSE DISPLAY ERASE ALL:"GOODBYE" ::
    CALL SAY("GOODBYE"):: END
500 DATA ,TWENTY,THIRTY,FORTY,FIFTY,SIXTY,SEVENTY,
    EIGHTY,NINETY
510 DATA ONE,TWO,THREE,FOUR,FIVE,SIX,SEVEN,EIGHT,NINE
520 DATA ELEVEN,TWELVE,THIRTEEN,FOURTEEN,FIFTEEN,
    SIXTEEN,SEVENTEEN,EIGHTEEN,NINETEEN
550 DISPLAY AT(12,1):"EXACTLY RIGHT!" ::
    CALL SAY("EXACTLY+RIGHT"):: DISPLAY AT(14,1):"VERY
    GOOD" :: CALL SAY("VERY+GOOD")
560 GOTO 460
```

continued on page 30

Anatomy of a Pirate

sent by John Paine

His eyes are bloodshot; he does not sleep. His wife and children used to know him; they no longer do. At one time, he was a fairly nice, easy going guy. He liked to tinker, so he bought a computer. His life will never be the same.

At night, he lurks in the shadows, seeking bad sectors, tearing them apart bit by bit, knowing that, soon, he will have broken the code and will have the world's first illegal copy of that diskette. He will keep his old car three more years, will not get his plumbing fixed, and will only survive on coffee and TV dinners, so that he may afford a third or fourth disk drive or the memory expansion he needs.

Decryption and un-protection are his only goals. He does not care what the disk contains or how useful the program may be; breaking the code is far more challenging to him that completing Zork III.

He broke the Zork series, but never played them. His purpose in life has become all encompassing. He will get sick from lack of rest; he will have marathon sessions trying to undo the last protection check in the program, and, when he finally has reached his goal, he will experience post-partum depression.

He is not after money, he is not after fame. He just wants to prove to himself that he is more intelligent that the one who devised the protection scheme in the first place. He will relate his exploits to a very close circle of friends at the club, and, because they listened, he will give them copies.

His energy and imagination, if harnessed, could be used to create another Telco or DM1000. His mind, unfortunately, is single tracked and lacks the visionary and creative qualities required. He is not unlike a counterfeiter; an electronic safe-cracker who has amassed a wealth of technical knowledge and has invested thousands in tools, only to satisfy that one consuming obsession.

He knows he will never get caught. He knows that, in reality, the ever increasing complaints of software manufacturers, and programmers whose wealth and luxury are threatened by his actions, are but a reflection on their inability to effectively protect their treasures. He knows that if one man can do it, another man can undo it. He knows that computers have rules that must be obeyed, and that all bootable disks must start the same way. That is enough of a crack for him to get through.

He hates unprotected disks; they offer no challenge. He will save enough to buy a new piece of software whose code has not been cracked, and sell it to the highest bidder at the first club meeting which follows his success.

In his public life, he is likely to be non-descript; an underdog who does not shine much at anything he does or says. He probably does not dress well, his physical appearance is of no importance to him. He does not have the charisma and moral fibre of a Long John Silver. His opinions are not sought, his advice is not followed.

He is not respected much, except by the freeloaders who depend on him. After all, he is giving something for nothing.

His darkest secret, however, is that he lives in constant fear that, some day, he will fail. He will not crack the code. He will realize that other club members were fair weather friends and that he lost, in a single stroke of fate, the attention he was so eagerly seeking.

Like the rest of us, he will grow old, his priorities will change, his eagerness will die down. As he looks around him, he may realize that the best times of life have passed him by, and that there is no making up for the lost time. He will be bitter, having left an insignificant mark on the world, having wasted his time in pointless pursuits. No one will miss him.

To him, I dedicate this epitaph:
Here Lies a Pirate
Who Never Sailed.

continued from page 29

Listing 9, Table Tennis Scorer
by Wade Bowmer

```
10 DATA ZERO,ONE,TWO,THREE,FOUR,FIVE,SIX,SEVEN,EIGHT,
   NINE,TEN,ELEVEN
20 DATA TWELVE,THIRTEEN,FOUR+TEEN,FIFTEEN,SIX+TEEN,
   SEVEN+TEEN,EIGHT+TEEN,NINE+TEEN,TWENTY,TWENTY+ONE
30 DATA TWENTY+TWO,TWENTY+THREE,TWENTY+FOUR,
   TWENTY+FIVE,TWENTY+SIX,TWENTY+SEVEN,TWENTY+EIGHT
40 DIM N$(28):: FOR I=0 TO 28 :: READ N$(I):: NEXT I
50 DISPLAY ERASE ALL AT(8,2):"TABLE-TENNIS SCORE
   KEEPER.": : :"""B""" SERVES FIRST..."
60 DISPLAY AT(18,1):"KEY ""A"" TO INCREMENT
          ""A"""S SCORE,          KEY ""B"" TO
   INCREMENT          ""B"""S SCORE."
70 CALL SAY("#READY TO START# REMEMBER,B-GOES+FIRST")
80 CALL KEY(0,K,S):: IF S=0 THEN 80 ELSE CALL SAY("O+K")
90 A,B,T=0
100 DISPLAY AT(15,3):USING "A: ##        B: ##":A,B ::
    DISPLAY AT(16,1):TAB(16-13*T);" SERVER"
110 IF A=B THEN CALL SAY(N$(A),,"ALL")ELSE IF T THEN
    CALL SAY(N$(A),,N$(B))ELSE CALL SAY(N$(B),,N$(A))
120 CALL KEY(0,K,S):: IF S=0 THEN 120
130 IF K=13 THEN 100
140 IF K=65 THEN A=A+1
150 IF K=66 THEN B=B+1
160 IF (A=21 AND B<20)OR(B=21 AND A<20)OR(ABS(A-B)>1 AND
    A>21 OR B>21)THEN 190
170 IF INT((A+B)/5)=(A+B)/5
    THEN CALL SAY("NEXT "):: T=1-T
180 GOTO 100
190 DISPLAY AT(15,3):USING "A: ##        B: ##":A,B ::
    DISPLAY AT(16,1):TAB(16-13*T);" SERVER"
200 IF A>B THEN DISPLAY AT(16,2):"A HAS WON!" ::
    CALL SAY("A,HAS+WON")ELSE DISPLAY AT(16,2):"B HAS
    WON!" :: CALL SAY("B,HAS+WON")
210 IF T THEN CALL SAY(N$(A),,N$(B))ELSE
    CALL SAY(N$(B),,N$(A))
```

Graphics Interchange Format

A standard defining a mechanism for the storage and transmission of raster-based graphics information.
© 1987 CompuServe Incorporated, 5000 Arlington Center Boulevard, Columbus, Ohio USA 43220

INTRODUCTION

'GIF' is CompuServe's standard for defining generalized colour raster images. This 'Graphics Interchange Format' allows high quality, high resolution graphics to be displayed on a variety of graphics hardware and is intended as an exchange and display mechanism for graphics images. The image format described in this document is designed to support current and future image technology and will in addition serve as a basis for future CompuServe graphics products.

The main focus of this document is to provide the technical information necessary for a programmer to implement GIF encoders and decoders. As such, some assumptions are made as to terminology relevant to graphics and programming in general.

The first section of this document describes the GIF data format and its components and applies to all GIF decoders, either as standalone programs or as part of a communications package. Appendix B is a section relevant to decoders that are part of a communications software package and describes the protocol requirements for entering and exiting GIF mode, and responding to host interrogations. A glossary in Appendix A defines some of the terminology used in this document. Appendix C gives a detailed explanation of how the graphics image itself is packaged as a series of data bytes.

Graphics Interchange Format Data Definition

GENERAL FILE FORMAT

```
+-----------------------+
| +-------------------+ |
| |   GIF Signature   | |
| +-------------------+ |
| +-------------------+ |
| | Screen Descriptor | |
| +-------------------+ |
| +-------------------+ |
| | Global Colour Map | |
| +-------------------+ |
| +-------------------+ |  --+
| | Image Descriptor  | |    |
| +-------------------+ |    |
| +-------------------+ |    |
| |  Local Colour Map | |    |- Repeated 1 to n times
| +-------------------+ |    |
| +-------------------+ |    |
| |    Raster Data    | |    |
| +-------------------+ |  --+
|-    GIF Terminator   -|
+-----------------------+
```

GIF SIGNATURE

The following GIF Signature identifies the data following as a valid GIF image stream. It consists of the following six characters:
G I F 8 7 a

The last three characters '87a' may be viewed as a version number for this particular GIF definition and will be used in general as a reference in documents regarding GIF that address any version dependencies.

SCREEN DESCRIPTOR

The Screen Descriptor describes the overall parameters for all GIF images following. It defines the overall dimensions of the image space or logical screen required, the existence of colour mapping information, background screen colour, and colour depth information. This information is stored in a series of 8 bit bytes as described below.

```
      bits
 7 6 5 4 3 2 1 0  Byte #
+---------------+
|               | 1
+-Screen Width -+   Raster width in pixels (LSB first)
|               | 2
+---------------+
|               | 3
+-Screen Height-+   Raster height in pixels (LSB first)
|               | 4
+-+-----+-+-----+   M=1, Global colour map follows
|M| cr  |0|pixel| 5              Descriptor.
+-+-----+-+-----+   cr+1= # bits of colour resolution.
|  background   | 6 pixel+1= # bits/pixel in image
+---------------+   background= Colour index of screen
|reserved (zero)| 7   background.  (Colour is defined
+---------------+     from the Global colour map or
                      default map if none specified.)
```

GLOBAL COLOUR MAP

The Global Colour Map is optional but recommended for images where accurate colour rendition is desired. The existence of this colour map is indicated in the 'M' field of byte 5 of the Screen Descriptor. A colour map can also be associated with each image in a GIF file but will normally use this global map because of hardware restrictions in equipment available today. In the individual Image Descriptors the 'M' flag will normally be zero. If the Global Colour Map is present, its definition immediately follows the Screen Descriptor. The number of colour map entries following a Screen Descriptor is equal to 2**(# bits per pixel), where each entry consists of three byte values representing the relative intensities of red, green and blue respectively. Each pixel value received is displayed according to its closest match with an available colour on the display based on this colour map. The colour components represent a fractional intensity value from none (0) to full (255). White would be represented as (255, 255, 255), black as (0, 0, 0) and medium yellow as (180, 180, 0). For display, if the device supports fewer than 8 bits per colour component, the higher order bits of each component are used. In the creation of a GIF colour map entry with hardware supporting fewer than 8 bits per component, the component values for the hardware should be converted to the 8 bit format with the following calculation:

```
<map_value> = 255*<component_value>/(2**<nbits> -1)
```

This assures accurate translation of colours for all displays. If no Global Colour Map is indicated, a default colour map is generated internally which maps each incoming colour index to a hardware colour index.

IMAGE DESCRIPTOR

The Image Descriptor defines the actual placement and extent of the following image within the space defined in the Screen Descriptor. Also defined are flags to indicate the presence of a local colour lookup map, and to define the pixel display sequence. Each Image Descriptor is introduced by an image separator character. The role of the Image Separator is simply to provide a synchronization character to introduce an Image Descriptor. This is desirable if a GIF file happens to contain more than one image. This character is defined as 0x2C hex or ',' (comma). When this character is encountered between images, the Image Descriptor will follow immediately.

```
      bits
 7 6 5 4 3 2 1 0  Byte #
+---------------+
|0 0 1 0 1 1 0 0| 1 ',' - Image separator character
+---------------+
|               | 2 Start of image in pixels from the
+-  Image Left -+  left side of the screen (LSB first)
|               | 3
+---------------+
|               | 4
+-  Image Top  -+    Start of image in pixels from the
|               | 5 top of the screen (LSB first)
+---------------+
|               | 6
+- Image Width -+   Width of the image in pixels
|               | 7              (LSB first)
+---------------+
|               | 8
+- Image Height-+   Height of the image in pixels
|               | 9              (LSB first)
+-+-+-+-+-+-----+   M=0 - Use global colour map,
|M|I|0|0|0|pixel|10          ignore 'pixel'
+-+-+-+-+-+-----+   M=1 - Local colour map follows,
                             use 'pixel'
             I=0 - Image formatted in Sequential order
             I=1 - Image formatted in Interlaced order
             pixel+1 - # bits per pixel for this image
```

LOCAL COLOR MAP

A Local Colour Map is optional and defined here for future use. If the 'M' bit of byte 10 of the Image Descriptor is set, then a colour map follows the Image Descriptor that applies only to the following image. At the end of the image, the colour map will revert to that defined after the Screen Descriptor. Note that the 'pixel' field of byte 10 of the Image Descriptor is used only if a Local Colour Map is indicated. This defines the parameters not only for the image pixel size, but determines the number of colour map entries that follow. The bits per pixel value will also revert to the value specified in the Screen Descriptor when processing of the following image is complete.

RASTER DATA

The format of the actual image is defined as the series of pixel colour values that make up the image. The pixels are stored left to right sequentially for an image row. Normally each image row is written sequentially, top to bottom. In the case that the Interlace or 'I' bit is set in byte 10 of the Image Descriptor then the row order of the image display follows a four pass process in which the image is filled in by widely spaced rows. The first pass writes every 8th row, starting with row 0. The second pass writes every 8th row starting at row 4. The third pass writes every 4th row starting at row 2. The fourth pass completes the image, writing every other row, starting at row 1. A graphic description of this process follows:

Row	Pass 1	Pass 2	Pass 3	Pass 4	Result
0	**				**1a**
1				**4a**	**4a**
2			**3a**		**3a**
3				**4b**	**4b**
4		**2a**			**2a**
5				**4c**	**4c**
6			**3b**		**3b**
7				**4d**	**4d**
8	**1b**				**1b**
9				**4e**	**4e**
10			**3c**		**3c**
11				**4f**	**4f**
12		**2b**			**2b**
. . .					

The pixel values are stored as a series of colour indices which map into the existing colour map. The resulting colour value is what is actually displayed. This series of pixel indices, the number of which is equal to image-width*image-height pixels, are passed in the GIF image data stream one byte per pixel, compressed according to a version of the LZW compression algorithm as defined in Appendix C.

GIF TERMINATION

In order to provide a synchronization for the termination of a GIF image file, a GIF decoder will exit GIF mode when the character 0x3B hex or ';' is found after an image has been processed.

Appendix A - GLOSSARY

Pixel - The smallest picture element of a graphics image. This usually corresponds to a single dot on a graphics screen. Image resolution is typically given in units of pixels. For example a fairly standard graphics screen format is one 320 pixels across and 200 pixels high. Each pixel can appear as one of several colours depending on the capabilities of the graphics hardware.

Raster - A horizontal row of pixels representing one line of an image. A typical method of working with images since most hardware is oriented to work most efficiently in this manner.

LSB - Least Significant Byte. Refers to a convention for two byte numeric values in which the less significant byte of the value preceeds the more significant byte. This convention is typical on many microcomputers.

Colour Map - The list of definitions of each colour used in a GIF image. These desired colours are converted to available colours through a table which is derived by assigning an incoming colour index (from the image) to an output colour index (of the hardware). While the colour map definitions are specified in a GIF image, the output pixel colours will vary based on the hardware used and its ability to match the defined colour.

Interlace - The method of displaying a GIF image in which multiple passes are made, outputting raster lines spaced apart to provide a way of visualizing the general content of an entire image before all of the data has been processed.

B Protocol - A CompuServe developed error correcting file transfer protocol available in the public domain and implemented in CompuServe VIDTEX products. This error checking mechanism will be used in transfers of GIF images for interactive applications.

LZW - A sophisticated data compression algorithm based on work done by Lempel-Ziv and Welch which has the feature of very efficient one-pass encoding and decoding. This allows the image to be decompressed and displayed at the same time. The original article from which this technique was adapted is:
Terry A. Welch, "A Technique for High Performance Data Compression", IEEE Computer, volume 17 no 6 (June 1984)
This basic algorithm is also used in the public domain ARC file compression utilities. The CompuServe adaptation of LZW for GIF is described in Appendix C.

Appendix B - INTERACTIVE SEQUENCES

GIF Sequence Exchanges for an Interactive Environment

The following sequences are defined for use in mediating control between a GIF sender and GIF receiver over an interactive communications line. These sequences do not apply to applications that involve downloading of static GIF files and are not considered part of a GIF file.

GIF CAPABILITIES ENQUIRY

The GCE sequence is issued from a host and requests an interactive GIF decoder to return a response message that defines the graphics parameters for the decoder.

This involves returning information about the screen size, number of bits/colour supported and the amount of colour detail supported. The escape sequence for the GCE is defined as:

ESC [> 0 g (g is lower case, spaces inserted for clarity)

GIF CAPABILITIES RESPONSE

The GIF Capabilities Response message is returned by an interactive GIF decoder and defines the decoder's display capabilities for all graphics modes that are supported by the software. Note that this can also include graphics printers as well as a monitor screen.

The general format of this message is:

```
#   protocol   { ; dev, width, height, colour-bits,
               colour-res }...  <cr>
```

'#'	- GCR identifier character (Number Sign)
protocol='0'	- No end-to-end protocol supported by decoder Transfer as direct 8-bit data stream.
protocol='1'	- Can use B-protocol to transfer GIF data interactively
dev = '0'	- Screen parameter set follows
dev = '1'	- Printer parameter set follows
width	- Maximum supported display width in pixels
height	- Maximum supported display height in pixels
colour-bits	- Number of bits per pixel supported. The number of supported colours is 2**colour-bits.
colour-res	- Number of bits per colour component supported in colour palette. If colour-res is '0' then no colour mapping is possible.

Note that all values in the GCR are returned as ASCII decimal numbers and the message is terminated by a Carriage Return character.

The following GCR message describes three standard EGA configurations with no printer; the GIF data stream can be sent using B-protocol:

#1 ;0,320,200,4,2 ;0,640,200,2,2 ;0,640,350,4,2<cr>

ENTER GIF GRAPHICS MODE

Two sequences are currently defined to invoke a GIF decoder into action. The only difference between them is that different output media are selected. These sequences are:

ESC [> 1 g   Display GIF image on screen
ESC [> 2 g   Display image directly to an attached graphics printer. The image may optionally be displayed on the screen as well.

Note that the 'g' character terminating each sequence is in lower case.

INTERACTIVE ENVIRONMENT

The assumed environment for transmission of GIF image data interactively is a full 8-bit data stream from host to microcomputer. All 256 character codes must be transferable. The establishing of an 8 bit data path for communications will normally be taken care of by the host application programs. It is however up to the receiving communications programs supporting GIF to be able to receive and pass on all 256 8 bit codes to the GIF decoder software.

Appendix C - IMAGE PACKAGING AND COMPRESSION

The Raster Data stream that represents the actual output image can be represented as:

```
 7 6 5 4 3 2 1 0
+---------------+
|   code size   |
+---------------+  --+
|  byte count   |    |
+---------------+    |
:               :    +-- Repeated as many times as
|  data bytes   |    |        necessary.
:               :    |
+---------------+  --+
+---------------+
|0 0 0 0 0 0 0 0|  zero byte count
+---------------+
```

The conversion of the image from a series of pixel values to a transmitted or stored character stream involves several steps. In brief these steps are:

1. Establish the Code Size - Define the number of bits needed to represent the actual data.
2. Compress the Data - Compress the series of image pixels to a series of compression codes.
3. Build a Series of Bytes - Take the set of compression codes and convert to a string of 8-bit bytes.
4. Package the Bytes - Package sets of bytes into blocks preceeded by character counts.

ESTABLISH CODE SIZE

The first byte of GIF Raster Data stream is a value indicating the number of bits required to represent the set of actual pixel values. Normally this will be the same as the number of colour bits. Because of some algorithmic constraints however, black and white images which have one colour bit must be indicated as having a code size of 2. This code size value also implies that the compression codes must start out one bit longer.

COMPRESSION

The LZW algorithm converts a series of value codes into a series of codes which may be raw values or a code designating a series of values. Using text characters as an analogy, the output code consists of a character or a code representing a string of characters. The first available string code is a value greater than the largest actual value to occur in the input.

BUILD 8-BIT BYTES

Because the LZW compression used for GIF creates a series of variable length codes, of between 3 and 12 bits each, these codes must be reformed into a series of 8-bit bytes that will be the characters actually stored or transmitted. This provides additional compression of the image. The codes are formed into a stream of bits as if they were packed right to left and then picked off 8 bits at a time to be output. Assuming a character array of 8 bits per character and using 5 bit codes to be packed, an example layout would be similar to:

```
 byte n    byte 5   byte 4   byte 3   byte 2   byte 1
+-..------+--------+--------+--------+--------+--------+
and so on|hhhhhggg|ggfffffe|eeeedddd|dccccbb|bbbaaaaa|
+-..------+--------+--------+--------+--------+--------+
```

Note that the physical packing arrangement will change as the number of bits per compression code change but the concept remains the same.

PACKAGE THE BYTES

Once the bytes have been created, they are grouped into blocks for output by preceeding each block of 0 to 255 bytes with a character count byte. A block with a zero byte count terminates the Raster Data stream for a given image. These blocks are what are actually output for the GIF image. This block format has the side effect of allowing a decoding program the ability to read past the actual image data if necessary by reading block count and then skipping over the data.

continued from page 1

Unfortunately we were a bit late with the magazine as the printing press broke down at the wrong time (when is the right time I wonder?) and I did not pick up the magazine until 2 pm on the Thursday before the meeting. A quick stuffing job and off to the post office by 3 pm and then it is up to Australia Post. Some people get the magazine the day after posting but others take longer. I am writing this on the Sunday of the long weekend after the meeting as we try this month to get the magazine out one week early. We are helped if all our regular correspondents meet the Wednesday after the meeting deadline (thanks Dick) but this week has not been a good one for me. When I got the Editor's system home from the meeting I found that some sectors on the hard disk were corrupted and eventually I was not able to read the root directory which contains all the subdirectories. This means that Myarc's DM5 will no longer work with that drive. It just hangs up. This means that it is not possible to copy files from the disk although particular files can be used. Very strange! I could copy all the Display Variable 80 files by reading them into the editor and writing them out. So I connected up two hard disks and did just that, from one to the other (yes, it can be done!). Luckily I had designed my power supply to handle 2 hard disks. I wrote a BASIC program to read the directories of all the sub-directories I could remember the names of. I am planning to amend a disassembled copy of Disk Fixer to have it look at sector information on the hard disk to see if I can bypass or fix sufficient of the corrupt sectors to enable the Myarc DM5 to work again. This would let me unload the disk and return it to John Vandermey along with his controller card. TIsHUG has bought another controller card and a 20 Mbyte hard disk, which are both working well. The copying process through the editor was effective but not very fast and then on Wednesday my good wife, on her taxi service for the children, managed to come into contact with another moving vehicle and cause major damage to our much loved (and only) 27 year old car. This required more time for insurance and some running repairs, so I am hoping that we can clear up the magazine this weekend so I can do some of the more urgent things which need to be done.

****

We have not received any newsletters to review this month. This may well be because Ross has been flat out with his tutorial sessions last weekend and this weekend. I am not sure how Ross finds time to do all that he does and I hope everyone appreciates his efforts as much as we do. He saves me considerable time by providing a disk of articles which I would otherwise have to load from the BBS. I must apologize to Ross and to you all for overlooking the last Communicator's article. One of the problems with a hard disk is keeping track of all the files, sorting out which have been processed and which are still to do. It appears this month instead.

****

I noticed Mark Richardson wrote to Terry about problems with the Magic Squares program we published some months ago. I hope he read the letter from Jim Peterson in which Jim explains that there is a problem with that puzzle if you do not initialise it correctly. Basically the puzzle will probably be not solvable for a number of random layouts. What the program fails to do is to start with the desired result and then to perform some random moves to mix it up. Then the puzzle is able to be solved.

****

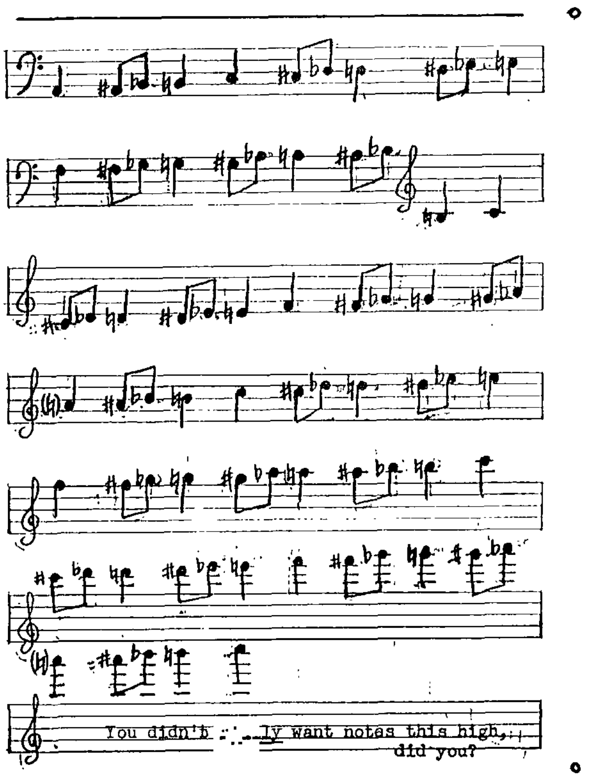
Thanks to all those who wrote in appreciation of our efforts with the TND. Both Rolf and I appreciate some feedback that you do find our efforts are worthwhile. We can only apply our own standards to what we do and hope that means that the result is easier to read, has fewer errors and has a pleasing layout. I have heard some murmurings that I "over edit", whatever that means, but I do not believe, for example, that abbreviations make words easier to read or that showing programs in their 28 column format makes them easy to read. In the program case, it may make it easier to check that you have typed it in correctly and so we use that format for the section on programs to type in, but for general reading I find that format to be very annoying and so I try to break the program lines at natural places, and to highlight the start of each statement. This means that you still should be able to type it in without errors but it may be harder to check. Still debugging, or finding out why a program does not work is a good way to learn to write programs, so perhaps it is better that way. If you feel strongly one way or the other, write and tell me.

****

I though we might get some correspondence over modems going with Daniel Harris' article and Ross' letter on modems. There was quite a bit of talk at the May meeting but no more letters. I thought that both Daniel and Ross had points to make and that the subject had not been fully explored. Has anyone else had experience at sending over the telephone at various baud rates? I am afraid that I do not quite agree with Ross that the baud rate is not present in the signal sent. Using modulation of other frequencies gives a complex signal whose frequency spectrum must contain the data frequencies you are transmitting. It would appear to make more sense that one should get less errors at a lower baud rate but perhaps Daniel has found some quirk where, in fact, there are fewer errors when 1200 baud is used. This can only be tested by experiment as, like Ross, I do not see any reasonable explanation for such a condition. It is not a matter just for experts but for anyone willing to try some experiments. How about having a go, or if you already have some evidence either way let us know.

****

One of the things I have been working on is a set of dictionary files for the spelling checker program which do not contain abbreviations or words of doubtful use or North American spellings. This is a long job and I have only completed about a quarter of the job. However I have written two programs to help, one in Extended BASIC to save the words in the existing dictionaries into files and the other in c99 to put the words in those files back into the dictionaries. These are working quite well, the first very slowly and the second quite quickly (reads in 710 sectors and writes out 345 sectors in 15 minutes) and I was hoping to put an article in this TND about them. The lengthy job is to go through the word files adding all the endings to verbs and nouns and generally checking for odd words not used frequently and other words left out. Then I will only have to keep a check as I use the dictionary for words which should be in there but have been missed for some reason. Anyone interested in the result should write to me.



Regional Group Reports

Meeting summary.

Banana Coast	9/07/89	Sawtell
Carlingford	19/07/89	Carlingford
Central Coast	8/07/89	Toukley
Glebe	8/07/89	Glebe
Illawarra	17/07/89	Keiraville
Liverpool	7/07/89	Northmead
Northern Suburbs	27/07/89	Davidson
Sutherland	21/07/89	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.

CARLINGFORD Regional Group.

Regular meetings are normally on the third Wednesday of each month at 7.30pm. Contact Chris Buttner, 79 Jenkins Rd, Carlingford, (02)871 7753, for more information.

CENTRAL COAST Regional Group.

Regular meetings are normally held on the second Saturday of each month, 6.30pm at the Toukley Tennis Club hall, Header St, Toukley. 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 third Monday of each month, except January, at 7.30pm, Keiraville Public School, Gipps Rd, Keiraville, opposite the Keiraville shopping centre. 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)644 7377 (home) or (02)759 8441 (work) for more information.

All demonstration programs are subject to Air Mail from USA. Some are still on the way

*** ALL WELCOME ***

July Meeting, 7th July, Larry Saunder's home, 34 ColechinStreet, Yagoona West, (02)644 7377. Demonstrations of Font Writer II, Old Dark Caves, Oliver's Twist, Grand RAM card and Press in 80 columns.

August Meeting, 11th August. Demonstration of Press.

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.

Regular meetings are held on the third Friday of each month at the home of Peter Young at Jannali at 7.30pm. Group co-ordinator is Peter Young, (02) 528 8775. BBS Contact is Gary Wilson, user name VK2YGW on this BBS.

The Sutherland Regional Group has made arrangements to purchase a console tester from Geoff Trott which will greatly enhance our capacity to repair TI99/4A consoles. Coupled with the recent tutorials on console repairs, this is a very positive step towards the continuing maintenance and usage of our beloved machine.

It has become very apparent in recent months, that Regional Group members, generally, have become more mobile with visits to other group meetings. This is a good trend which leads to the further interchange of ideas between members.

Peter Young

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 Woodstock Community Centre, Church street, Burwood. Regular items include news from the directors, the publications library, the shop, and demonstrations of monthly software. Meetings for the next couple of months are:

July 1 - Rolf Schreiber will demonstrate his Geneve computer. Well worth seeing if you intend to expand in this direction. In order to capitalise on the information learnt at the full day workshop, the first of a series of mini-lectures will be presented.

Future meetings are on August 5 and September 2.

Craig Sheehan (Meeting co-ordinator).

continued from page 8

Now the explanation. The character definitions are stored in VDP RAM sequentially, each taking up 8 bytes. Character 48 is stored at addresses 1152 1153 1154 1155 1156 1157 1158 and 1159. Then from 1160 comes the definition for character 49, and after that 50 and so on. (Note to machine code programmers; all numbers used here are decimal!).

If you wish you can replace lines 150 and 160 with these. The effect is exactly the same:

```
150 CALL CHARPAT(48,A$)
160 CALL CHAR(58,A$)
```

as the definition of character 58 begins at memory address 1232.

Lines 100 to 130 place some character 93's on screen, and in the T loop we shall continually redefine this character, whose definition is held starting from VDP address 1512.

In order to achieve a smooth scrolling effect, it has been necessary to have a character defined as a number zero at both ends of the numbers, before 1 and after 9! This is what lines 150 and 160 do.

Within the T loop we are reading the definitions of the numbers, but not as normal, with CALL CHARPAT which looks at 8 byte blocks, but with direct addressing, which enables us to increment by one memory address at a time, for an interesting effect. Changing from number 1 to number 2 takes a 8 cycles through the loop and as each cycle reads 8 bytes at a time, that means we are actually reading 64 bytes in order to go from number 1 on screen to number 2 on screen. And of course we are also performing a similar number of writes, making a total of 128 memory operations. All things considered the speed is not too bad!

(There is enough information here for you to work out how to amend the cursor definition using Mini Memory. Drop me a line if you really are stuck and I will put it in next time!!!).

That is all for this issue. Let me know what you would like next! Stephen Shaw. 1989.