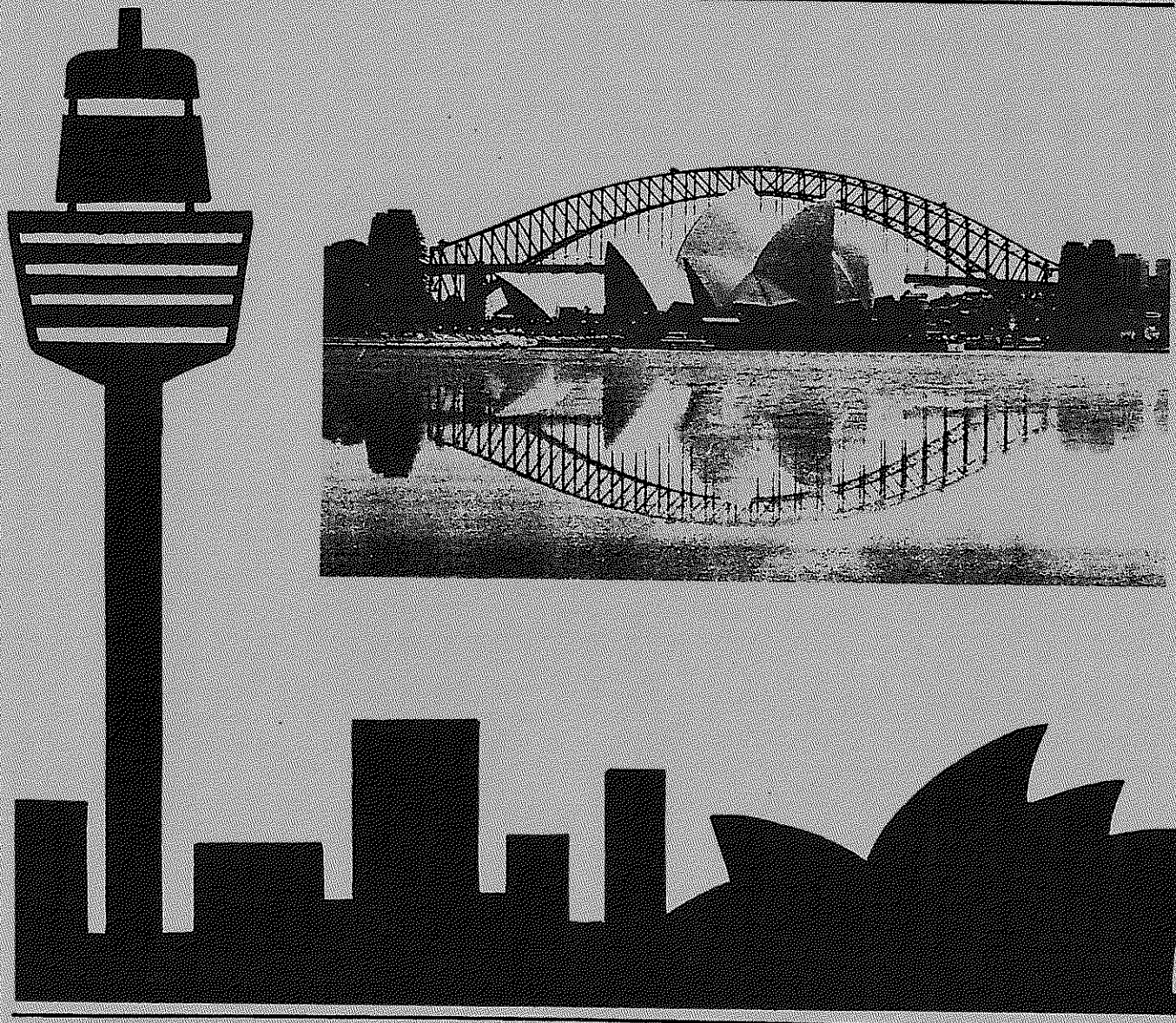
TISHUG NEWS DIGEST

Focusing on the TI-99/4A Home Computer

Volume 7, Number 7 **August 1988**

Registered by Australia Post - Publication No. NBH5933



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

TIsHUG News Digest ISSN 0819-1984

TIsHUG (Australia) Ltd.

TIsHUG News Digest

August 1988

All correspondence to:

P.O. Box 214
Redfern, NSW 2016
Australia

The Board

Co-ordinator
Chris Buttner (02) 871 7753
Secretary
Terry Phillips (02) 797 6313
Treasurer
Percy Harrison (02) 808 3181
Directors
Cyril Bohlsen (02) 639 5847
Russell Welham (043) 92 4000

Sub-committees

News Digest Editor
Geoff Trott (042) 29 6629
BBS Sysop
Ross Mudie (02) 456 2122
Merchandising
Bob Bunbury (02) 601 8521
Publications Library
Warren Welham (043) 92 4000
Software library
Terry Phillips (02) 797 6313
Technical co-ordinator
John Paine (02) 625 6318

Regional Group Contacts

Carlingford
Chris Buttner (02) 871 7753
Central Coast
Russell Welham (043) 92 4000
Coffs Harbour
Kevin Cox (066) 53 2649
Glebe
Mike Slattery (02) 692 0559
Illawarra
Bob Montgomery (042) 28 6463
Liverpool
Larry Saunders (02) 644 7377
Northern Suburbs
Dennis Norman (02) 452 3920
Sutherland
Peter Young (02) 528 8775

Membership and Subscriptions

Joining fee	$5.00
Annual Family Dues	$25.00
Overseas Airmail Dues	AUS$50.00
	or £22.00
	or US$30.00
Publications Library	$5.00
Texpac BBS	$5.00

TIsHUG Sydney Meeting

The next meeting will be at 2 pm on 3rd September at Woodstock Community Centre, Church Street, Burwood.

Printed by
The University of Wollongong
Printery

Index

Title	Description	Author	Page No.
64K on multifunction card	Hardware review	Schubert,Peter	10
ArchiverII V2.4 colour change	Software hints	Schreiber,Rolf	26
Cassette care	General interest	Shaw,Stephen	24
Cheap computer	General interest	Kanitz,Werner	18
Communicators	BBS information	Mudie,Ross	5
Computer war	Software review	Brown,Robert	18
Computer war	Software review	Judd,Stephen	18
Disk assembly file formats	Software hints		6
Disk drive troubles	General interest	Bunbury,Bob	26
Extended BASIC into the console	Hardware	Mudie,Ross	20
FORTRAN	General interest	Harris,D.N	28
Forth column	Forth forum <4>	Smyth,George L	27
From the bulletin board	Mail to all		5
Games information	General interest	Brown,Robert	13
Games information	General interest	Judd,Stephen	13
Gemini printer control codes	Word processing	Buehler,Bob	9
Hard disk controller news	General interest	Takach,Ben	3
Letter to directors	Club affairs	Amadio,Lou	30
Letter to editor	Club views	Mudie,Ross	4
Letter to editor	Club views	Ryan,John	4
Link it #17	Software hints	Mudie,Ross	7
Logic for beginners	General interest	Mudie,Ross	21
Logo: debugging procedures	Software hints	Felzien,Rick	24
Myarc hard/floppy controller	General interest	Christensen,Garry	19
Newsletter snippets	General interest	Meldrum,George	4
PRBASE, a user's point of view	Software review	Amadio,Lou	29
Program to type in	Battle at sea	Balthrop,W.K	15
Program to type in	Program peeker	Mineo,G	17
Program to type in	Typing for accuracy	Regena	14
Regional group reports	General interest		31
SEARCH program	Software hints	Shaw,Stephen	10
Secretary's notebook	Club news	Phillips,Terry	2
TI NET press release	General interest		3
TI99/4A users UK annual meeting	General interest	Shaw,Stephen	2
TIsHUG software column	Club software	Phillips,Terry	3
They're off	General interest	Trott,Geoff	1
Tips from the Tigercub #48	Software hints	Peterson,Jim	25
What is in a file?	Software hints	Shaw,Stephen	6
Younger set	Program	Maker,Vincent	11
Younger set	Program	Szemere,Ian	11

They're off

by Geoff Trott

As I write this we have missed the printer's deadline for production by the Sydney meeting date. Unfortunately both members of the production team were extremely busy for the first two weeks after I came back from Holidays so there just was not the time to do the job. We decided it was better to be a bit late and continue our high standard rather than produce a rush job with errors. I hope that you find this issue worth the wait. I think that it is a good one with a lot of contributions from the local scene which are worth reading. In doing my job I read most contributions about 3 times while they are passing through my TI-Writer, so I do not spend much time reading the actual magazine when it emerges. Fortunately I have friends who are alert enough to spot the mistakes that have slipped through.

continued on page 30

TI99/4A user group (UK)

Annual meeting

from Stephen Shaw, England

The User Group, with members from all corners of the UK, and several overseas as well, has about 230 members. The Chairman of the Group thought that 120 persons attended the annual meeting. As the day was sunny, and the start of an extended week-end (the Monday following being a national holiday) the attendance was satisfactory.

The meeting was partly sponsored by the last commercial supplier in the UK of TI99/4A products, PARCO ELECTRICS, who in addition to the good range of the older modules (and cassettes) also had with him some of the newer modules from Databiotics, and the new Triton Extended BASIC.

There is a considerable shortage of Expansion Boxes in the UK, which is hampering many owners from fully appreciating their machine, so the new modules are much appreciated here.

There was a GENEVE computer in attendance, using a very high resolution Philips monitor, which gave a very sharp rock steady display (with interlace off). We were duly impressed by the 80 column word processor and Multiplan. Unfortunately, the disk containing the demonstration hi-resolution graphics turned out to be blank (accidents do happen) so we did not see the very best that the machine was capable of. A very creditable showing, and a machine Lou Phillips (who has received some quite barbed criticism from some quarters) can be justly proud of.

Our Hardware secretary Mike Goddard was busy with soldering iron in hand, converting members consoles to work with joysticks whether the alpha lock was up or down, by inserting a single diode on the keyboard PCB. As no two keyboards seemed to have been made by the same manufacturer, there was always a degree of experimentation finding the right bit of PCB to use!

Mike also had with him a home made module, with four EPROMS on board, each with a different game, selected by rotary switch.

Peter Brooks, the proprietor of a different user group, the "International Texas Instruments User Group" based in Oxford was present. Peter's group has not produced a newsletter (normally monthly) since February 88, as Peter, who does ALL the work (almost) has been rather busy switching from one job to another, with consequent claims upon time and funds. Peter hopes to be back in operation sometime this year.

We learned that Parco Electrics has a very good stock of COLLINS starter packs. These were produced by a leading UK publisher on behalf of Texas Instruments, who dropped the home computer before publication, but after production. There are four titles, Starter Packs 1 and 2, and Games Writers Pack 1 and 2. They comprise a book and a cassette, are very well written, and form a powerful resource for the TI BASIC programmer. Here in the UK, consoles continue to change hands, and our membership seems to be growing steadily, if slowly. There remains a need for resources like this for the new owner, and indeed for some of the older, unexpanded owners too!

**[[Sample pack is enclosed for your evaluation. If you wish to place a bulk order (say more than 10 copies of one title), there are plenty to go round, please would you contact the supplier concerned to negotiate a price. Postage is likely to exceed unit cost. Parco Electrics, 1 Manor Close, Weston, HONITON, Devon, ENGLAND, EX14 OPE]]

Parco also has a good supply of SHAMUS modules, so again, contact them if you would like to place a bulk order.

We did not have the heavy dealer support of the US Faires that we read of, but we did enjoy meeting together with fellow TI99/4A owners. Our wide geographic spread means that some owners have no other opportunity of meeting a fellow TI99er.

**TEXPAC SYSOP NOTE: The sample pack which was received by TIsHUG was "Games Writer 1" which will be available for inspection at the next TIsHUG meeting. Anyone interested should see Terry Phillips.

Secretary's Notebook

by Terry Phillips

A warm TIsHUG welcome is extended to 2 new members who have recently joined us. They are:

Tony Bell from Bomaderry and
Ken Trotman from Pendle Hill.

Ken was a member a couple of years ago so it is good to welcome him back.

Among correspondence received and actioned at the last Directors meeting was the following:

From Stephen Shaw of the UK Users Group a sample package entitled Games Writer Pack 1. This book and tape was produced by a leading UK book company just prior to TI exiting the home computer market. Parco Electronics, one of the remaining supporters of the TI in the UK has large stocks of the sample provided as well as other titles, Starter Pack 1 and 2 and Games Writer Pack 2. It has been suggested by Stephen that we could probably import these for around $3 a copy. The sample copy received has been given to Warren Welham to go through and do a review.

Geoff Warner of TIUP (Perth) wrote to advise the results of elections held recently in Perth. Geoff has been elected as Secretary, and with a new committee hopes to soon have TI interest in the west up and away again.

Tony McGovern wrote to thank TIsHUG for their donation for Funnelweb and he enclosed a new copy with some minor bugs fixed. This version was distributed at the July meeting.

Some years ago, the group purchased a high speed tape duplicator and overhead projector. As little use is now made of either of these assets, it has been decided to sell them. Advertisements will be placed in the appropriate classifieds of the Sydney press, however if you know of anyone who may be interested in purchasing both or either please contact me for further details of make, model, price etc.

Do not forget that the next meeting will take the form of a buy/swap/sell meet and will kick off at about 2pm. But remember that the early birds will probably get the best bargains, so if you have something you want to sell or buy get there early and perhaps strike a bargain. As previously mentioned all goods will be sold or swapped on an as is basis with the group taking no responsibility for non working items.

I guess all members will be pleased to know that despite early indications, membership has now climbed to 211, and I expect to get more renewals after having mailed a reminder to all non renewing members. Response so far to this mail out has been encouraging.

A big thanks to Shane Ferrett who put TI-BASE through its paces at the July meeting. Good work Shane, you certainly showed us all the versatility of this fine piece of software.

That is all for this month. See you at the next meeting.

o

continued from page 4

being reproduced elsewhere. It always gives me a bit of a lift to see one of my articles in an interstate or overseas magazine.

I have a personal preferance for a blank line between paragraphs when I am reading. To me it creates less clutter on the page and makes it easier to read. I realise that this is contrary to getting as much in to a page as possible but I think that readability is very important. (Does anyone else have a view on this? Should we put more blank lines in between paragraphs, even if it means that you have to turn to another page to read the last few lines of an article? Your comments would be appreciated. Ed)

About the Regional Group reports on the BBS. Each Regional Group now has Sub-Editor status on the BBS and as such can input their own file to the BBS without any need for SYSOP intervention. There is no longer any need to send these items as mail to the SYSOP.

Hope you have not found the owner of that snake skin yet!

Regards...Ross Mudie.

o

TIsHUG Software Column by Terry Phillips

This last month has been a particularly good one insofar as receipt of exchange disks is concerned, and a selection of the better Freeware will be available at the meeting to be held in September.

Here is what will be available:

DISK A224 - TERR-WARE GAMES. If you like playing or watching others play the card machines that seem to be proliferating in hotels and other places, then you will like JOKER POKER which is one of the offerings on this disk. A very good simulation, of this card game, in fact one of the best I have seen. In the same category is the second offering on the disk, BLACKJACK, which I feel is the finest programming I have yet seen on this type of card game. But that is not all that is on the disk. The best of all is the WHEEL OF FORTUNE game, just like the one on TV. Guaranteed hours of family fun with this disk. It requires Extended BASIC and 32K expansion.

DISK A225 - MORE WHEELS OF FORTUNE AND MONTE CARLO. This will let you compare various programmers efforts as this disk contains 2 Wheel of Fortune games and a truly great MONTE CARLO roulette game. Hope you break the bank! It requires Extended BASIC and 32K expansion and comes on a flippy disk

DISK A238 - TASS 2001. With this great utility program you can create slide shows of your favourite TI Artist, Graphx and RLE pictures. Full instructions are on the disk as are some sample pictures. One of the most interesting programs of its kind that I have had the pleasure to play around with for some time. Extended BASIC and 32K expansion required.

DISK A244 - CHARACTER DESIGNER. This one is from the UK and would have to be one of the best around at the present time. Has a host of features not found on others and is sure to impress those programmers who are into character design. It requires Extended BASIC and 32K expansion.

As well as these disks some 30 others were received, many of which have not yet been fully catalogued. Included are a host of adventures, utility and games for all ages. More to be released over the coming months.

And remember if you want a program that is in the library, drop me a line or ask for a complete catalog.

I have just received a copy of TELCO V2.0, which will also be available at the August meeting. This is the archived and compressed version so you will need the Archiver program to decompress and de-archive. o

TI NET press release

A new and innovative way to market nationally and guarantee a fair monetary return on Fairware programs has been announced by Jeff Guide, System Manager of TI NET on Delphi.

TI NET is proud to announce a first in National Telecommunications Networks, Surcharge Software. According to Guide, "Surcharge Software permits authors of fairware programs to market their products at a very low cost and in turn assure a return on their investment. Every time a program is downloaded the author receives a royalty."

Members of TI NET are permitted to upload as many fairware programs as desired. You decide the price and a cheque is sent directly to you within 45 days by the System Manager. A small service charge of 10% is charged for each download by Delphi and mailing costs by the System Manager. A very small price to pay for the opportunity to market your program nationally.

Non-members may sign up online and shortly thereafter take advantage of Surcharge Software. For the introductory price of $US29.95, you receive a lifetime membership to Delphi and TI NET, a 500 page Delphi User Manual and one free hour of non-prime connect time. A small price for national exposure to your Fairware programs. Dial 1-800 365-4636 or 617-576-2981. Once conected press return twice. At password, type TINET and press return.

Where else can you guarantee payment for your programs when they are downloaded? Only on TI NET!

Contact:
Jeff Guide
System Manager, TI NET
P.O. Box 244,
Lorton, Virginia 22079 o

Hard Disk Controller News, Good and Bad

by Ben Takach

Good News no.1

Lou Phillips has confirmed my 18 months old order for 5 hard disk controllers (HFDC). These are ready for shipment. These are not the half baked prototypes, but the real McCoys! The controllers will support up to 3 hard disk drives, 4 floppys, one streamer tape, a real time clock (clocks are popular lately eh!), and if you happen to be the proud owner of a Geneve the controllers will support DMA (who ever she is, it seems immoral to me to go to bed with Geneve and support DMA on the side).

Good News no.2

Arm twisting all around resulted in rock bottom landed costs:

-the Reserve Bank got tired of playing silly games with the Aussie dollar,
-the Customs Department dropped the 2% import duty (bless their greedy little heart),
-Lou Phillips extended the distributors discount,
-I managed to screw the (honi soit qui mal y pense, I was never intimate with Geneve!) freight charges down to US$71.60 for the lot.

So it is a bargain compared to the advertised US price of US$325.

Good News no.3

The asking price to you ex Sydney is not A$399.95, but a mere $398.90.

End of Good news, now the Bad News.

Bad News no.1

Hopefully the Australian side of the good news mentioned under G.N.no.3 will enjoy the sunshine for just 3 more weeks, otherwise my calculations will be short by a dollar or two. This is the lesser of the bad news.

Bad News no.2

I did not want 5 controllers, but only one. Nor did I part with nearly 2000 dollars cheerfully! I had to do it to qualify for the big discount. So I will get the surplus 4 cards off my hands as fast as possible. So speak up now or else you will miss out. This would be really bad news. My intestinal feeling indicates that many of these 5 at once orders are most unlikely (unless the TI fraternity suddenly became the first ever big spenders). There will not be many 99'ers who will wish to part with $398.90 to enable the placement of several repeat orders.

Finally, no Bad News, no Good News, just News no.1 and last.

The cards shall be in landed and cleared by the 24th of July. So phone me on 4894492 or write to me or leave a message for me on the BBS if you wish to get one of these cards. Bye-bye for now. Ben Takach

o

Newsletter Snippets

by George Meldrum

Our Editor normally gives a brief rundown of what you can read in other newsletters which are available in our clubs publications library. As Geoff is on holidays for a couple of weeks the task is over to me, so here goes.

The Ottawa TI99/4A Users Group Newsletter June 1988

DM 1000 warning :

An apology from the Ottawa TI99/4A Users Group which recommends that anyone using DM1000 3.6 or higher go back to 3.5. It is the most stable version at the moment, and later versions may contain potentially serious bugs. Version 4.0 is an escaped beta copy and should not be used under ANY circumstances. They hope to have a new version (thoroughly beta-tested) ready to go in the fall.

A rave review about Legends 1.1. A game suited for Tunnels of Doom fans but which is, quote, "much more challenging, interesting, and sophisticated". Supplied by Asgard Software, P.O. Box 10306, Rockville, MD 20850 for $27.95 (US) plus postage.

Tips and tutorials on TI BASIC and Extended BASIC.

MICROpendium May 1988

Trials of a c99 beginner: a program written in the c language to type in.

An article about printer control characters and fonts.

How to modify a TRS-80 mouse to work with TI-Artist V2.01.

A review of PLUS!, a word processing companion.

Various Geneve tips.

CIM 99 Club Infomatique MONTREAL Juin 1988

La Revue des Evenements
La Carte HDCC se fait attendre
MULTIPLAN: Comment faire un Budget - 2e partie
WRITER+plus: Comment faire le Multi-Colonnes
Graffittis du Babillard
PRBASE 2.0: Utiliser PRBASE avec TI-Writer
From the Library: Fairware Quick Reference list
Le Marche aux Puces
Le Calendrier des Evenements

The Tacoma Informer June/July 1988

The Tacoma 99ers Users Groups software librarian offers for their members selected software on disk. A catalogue of club software is made available at each meeting from which members make their wanted lists. A fee of $2.00 is paid for filling a SSSD disk which the librarian brings to the next meeting.

A review of the Desk Top Publisher cartridge. This module allows you to create a graphic picture and then include the picture in your text. Cost is US$69.95.

Northern NJ 99ers Users Group May 1988

An article on mounting the Extended BASIC cartridge in the console.

Monitors, or why a TV set is not good enough for 80 columns. A technical description complete with diagrams.

Spirit of 99 Central Ohio 99ers User Group June 1988

Part 9 of TI-Writer tutorial by Stan Katzman, covering the dot commands used to set margins, right adjust, indenting the beginning of a paragraph and centering text headings.

An interesting article by Jack Sughrue, Impact-99, "Good old Days". It deals with his personal experiences in the TI99/4A world, in particular how club meetings become too technical for the ordinary home user.

A review of Braille 'N Speak, a new computer for the blind.

A circuit to slow down the TI99/4A, hmmm.

A program for cassette + 32K memory systems to load Extended BASIC programs in half the usual time. If you are still using a cassette system then this program is a must, so they say. Send US$5.00 to Pittsburgh User Group, P.O. Box 8043, Pittsburgh, PA 15216. Mark it Attn: Pug Librarian, for your copy of Clyde Colledge's High-Speed Cassette Loader.

The Pug Peripheral Pittsburgh Users Group May 1988

Talk of what to expect from the new Databiotics GRAND RAM card.

Multiplan article dealing with the EXTERNAL COPY command.

PRBase Bug Report, written by Bill Warren, author of PRBase. A worthy report for those into PRBase.

TI Artist tip: if you have a design you want to work out some fine details on, zoom in to the area and save it to disk. Now bring it back from disk and zoom it again.

The Pug Peripheral Pittsburgh Users Group June 1988

News of a Forth system that loads into the 8K in the Supercart module.

Turbo Pascal is being offered by L.L.Conner Enterprise of 1521 Ferry Street, Lafayette, Indiana 47904 for US$59.95.

Multiplan article featuring windows.

Hunter Valley 99ers Users Group Newsletter June 1988

A parcel from Workshop Rheinland, West Germany, containing a GRAM-LOADER-MODULE and support software on disk such as special versions of TI-Writer, Multiplan, E/A, Logo, etc..

A locally produced QED RAMdisk board for 512K/1 Meg to fit in the PE Box. Anticipated cost for the board will be approx $55 - $60.

A modification by Neil Quigg to the CPU PAD memory. With an extra memory chip you can load different data at >8200 to that at >8300.

A Struggling Forth article by Richard Terry on a Forth screen print utility.

An extensive description of the PLUS! utility package.

Some examples of programming using disk files.

An outline of the latest Funnelweb version 4-10. o

Letters to the Editor

Dear Geoff

I am happy to be back home and am now free to help you in any way possible. I promised you that I would offer a suggestion to foster new members and hold the interests of the dedicated.

The proposal is to produce a bumper all embracing edition of TND in September. This edition will set the standard for future editions. It will be mailed to all past/present Members. In addition to every known Owner of the TI99/4A. A joining fee $10 will be offered as an inducement to join.

To fill this edition with the right material every member will be canvassed to provide ideas, such as a copy of his PET program, I believe we all have our favourites. Mine is a lazy and cold way to check the Lotto results for that elusive win.

Perhaps offered as a gift to all attending group or Sydney meetings. An advertisment in the local paper or radio may be heeded by the needed! Publicity of our group coupled with value to be seen must be brought to the notice of all.

Your efforts are appreciated by my fellow BANANA Coast members and you can rely on their support.

Summing up the proposal -:

(A) TDN to produce a Gift edition.
(B) TDN mailed to past/present users.
(C) All members to submit text.
(D) Distributed free to visitors attending group meetings.
(E) Publicise meetings on the radio and in the press. Community news item (Country). Paid press advertisements to attract TI99/4A users to meetings.
(F) Cost from publicity fund approved by the directors.

Your views on this or any other matters will be shared by all Banana Coast members.

Regards for the present,
John Ryan.

Dear Geoff,

Having just received my TND I feel I must comment very favourably at the material contained in the issue. A dose of the flu or not, it is still a good issue of the magazine. I agree with your comment on material

continued on page 2

The Communicators

Special Interest Group for
Users of the TEXPAC BBS.
by Ross Mudie, 12th July 1988.

1. REGIONAL GROUPS and SUB-EDITORS.

A number of TIsHUG's Regional groups are placing their meeting notices and reports directly in the NEWS area of the BBS. Users can select the wanted Regional Group file from the menu without having to read all the other groups' information first. The BBS date stamps all these files in the first line of the file so you do not have to waste time reading the file just to find that it has not been updated from last time that you were on the BBS.

Any Regional Groups who are not currently using the BBS Sub-Editor system to place group information on the BBS should contact the BBS SYSOP for more information. The Regional Group page in the TND is compiled by the Editor using much of the information from these files on the BBS.

2. BBS HELPERS.

I continually appreciate the help, assistance and encouragement that I receive from the people who supply information regularly or just occasionally for the BBS. I wish to thank especially Stephen Shaw in England, Robert Brown, Stephen Judd, John Paine, George Meldrum, Terry Phillips, Chris Buttner, John Ryan and Shane Ferret for assistance provided with files, programs and valuable information. I also greatly appreciate the effort involved on the part of the Sub-Editors who have cut my file maintenance work load allowing me more time for other creative activities, (I managed to put three other articles together for the TND this month). If I have missed out thanking anyone who has recently provided material then please accept my apology, and do not let my terrible memory discourage you from sending some more information or a program along.

3. BBS FAILURES, the Murphy's Law Department!

There has been another system failure, this time on the monumentally wet night with very strong winds of 5th July 1988. Again it is suspected that the cause was a glitch on the power mains as the failure was not usage induced and there were severe glitches occuring on the power mains in many areas. The BBS was restored at 7.10am on 6th July 1988.

If the BBS fails, the WATCHDOG TIMER takes the modem off line after 12 minutes. This prevents the modem from answering incoming calls when the BBS is unable to function. Callers prior to the 12 minute timeout will be answered by the modem then get no response whilst after 12 minutes the modem will not answer calls.

When calling the BBS the modem answers after the third ring, if it rings longer than this then you are ringing into a wrong number or the BBS is off line.

4. UN-INTERRUPTABLE POWER SUPPLY FOR THE BBS?

Some months ago the BBS 240V mains was supplied via a small Uninterruptable Power Supply (UPS) which has a 12 volt battery to supply power via an inverter for a few minutes to overcome the effect of short power breaks. Unfortunately the long term reliability of the UPS proved to be lower than the power mains, at least until recently.

I have considered providing alternate simultaneous DC power sourcing, via the normal mains power supply and from batteries, but I question the worth of the needed effort in view of the falling usage of the BBS.

5. BBS USAGE.

1988	February	March	April	May	June
CALLS	329	328	302	250	231
USERS	61	52	59	55	51
Usage Hrs.Mins	119.39	123.40	120.13	103.37	94.24

I am concerned at the way that BBS usage seems to be falling away. I would like to believe that the reduction of usage is due to users' computers being located somewhere cold and the warm lounge room in front of the TV is more comfortable. Is this the problem, or am I wasting my time selecting programs, implanting when necessary for download and writing, reviewing, editing general information files? I am not receiving any feedback from the members about wanting anything different on the BBS. Maybe the novelty of home computing is starting to wear off for a lot of the TI99/4A users. Use it, or it may not be there when you log on after all the usage has dropped to nothing!

From the Bulletin Board

MAIL TO : ALL
MAIL FROM : JAYJAY

I have just joined 2000 & BEYOND BBS and can highly recomend it to all users that enjoy adventure games as it has an online service. It also has several other points of interest, for example, an online magazine, a home shopping service, and of course a user uploaded program area. It has a lot of other sections that can be of interest.

So if you are interested in being a visitor just call this number 522 6514.

Oh by the way, the cost per year is $10.00 (pretty cheap for what it is offering).

BYE Jay Jay Jr

MAIL TO : ALL
MAIL FROM : SECRETARY

An article in the current issue of the Ottawa UG magazine warns against the use of DM1000 V4.0 as unpredictable results may be obtained. In fact the article suggests that no version higher than 3.5 can be used safely without risk. You have been warned ....
Regards Terry

MAIL TO : ALL
MAIL FROM : JAYJAY

I am member to a BBS that has a different download to the TI99/4A's TEII, and I would like to have one that does this so if anyone could tell me about some I would be gratefull. Thanks

JayJay Jr(JJJ)

MAIL TO : ALL
MAIL FROM : PAINEY

For Sale

I have been advised that there are two GramKrackers for sale in Australia. Price about $300 ea.

Contact Painey on this BBS if interested

MAIL TO : ALL
MAIL FROM : LOU

For Sale

One only CorComp system, complete with 2 double sided, double density drives, TI99/4A computer, Extended BASIC cartridge, RS232 built in, PIO ports. Complete $850 or $650 without keyboard or Extended BASIC cartridge. Also has memory expansion built in.

For Sale

Modem card to fit PE box $50.

Phone LOU 639 8888

MAIL TO : ALL
MAIL FROM : PETESAKE

As the club shop now has sufficient stocks of the old horizon type RAMdisk board, and as the Board has decided to ignore my development work on the new AT RAMdisk+clock, which was near completion some time before the boards were ordered, I have no alternative than to shelve this new board design, and so no further work will be done on it unless it can be shown that there is sufficient interest. Economically, it is not feasible unless at least 50 boards are produced.

On another matter, I now have stocks of MiniRAM+ boards for the Mini-PE system, and those who ordered these should soon hear from me. Regards, PETESAKE.

What's in a File?

from Stephen Shaw, England

Disks for the TI99/4A can have a number of different file types, which can serve several different purposes. How can you tell which is which? In some cases, you are reduced to trial and error! But the following notes may help.

*** PROGRAM FORMAT:

Unwisely named, as not all PROGRAM files are programs. This type of file is better described as MEMORY IMAGE. It is just a byte by byte image of a particular area of memory in the computer, which MAY be a program in BASIC or machine code, or some kind of data; graphics, adventure data and so on.

FUNLWEB will identify BASIC and most machine code images for you. Use SD from TI Writer, and once the directory is on screen press the equals (=) key. The right hand column will now be marked BA or EA as appropriate.

Other memory image files can be identified as follows:

File name ends in _P or _C: A picture to load with TI Artist or MAX/RLE.

File is 54 sectors long: MAY be a picture to load with Graphx or MAX/RLE.

File is 25 sectors but does not end in _P or _C: MAY be graphics for CERT99.

File is 8 sectors: May be a graphics screen for Fractal Explorer.

File names are identical except for last letter:

Type 1:	Type 2:
INVADERS	INVADERS
INVADERT	INVADERS1
INVADERU	INVADERS2
Ed/As Option 5	Gram Kracker

Machine Code program Machine Code program.

If you place the files on a blank new disk, and inspect the header (first 3 bytes) on Sector 22, with a sector editor, as supplied with Funlweb, you may also determine:

TUNNELS OF DOOM: 0406 0504 0400

Scott Adams Adventures: 2020 2020 2020

Module RAM required: If third byte is >8000 or >7000

If the first byte is 0000 you MAY be dealing with graphics such as a CHARA1 file or a GRAPHX picture. On standard E/A Option 5 files, where there is more than one file, eg INV1,INV2 and so on, the first byte on the LAST file is >0000 while all preceding files have a first byte of >FFFE (Gram Kracker files are >FF05,>FF06 etc).

*** DISPLAY VARIABLE 80:

Almost all DV80 files can be loaded into TI Writer for inspection. If they will not load they may have been prepared with the European version of TI-Writer. Just use the little utility program given elsewhere in this issue to force them to load!

DV80 files are most commonly:

TEXT- documentation.

SOURCE CODE - which may contain instructions!

GRAPHICS: Files ending _I, _F, _S are for TI ARTIST.

5 sector and 2 sector files MAY be for Picasso.

Files may be RLE graphics.

*** DISPLAY FIXED 80:

Usually used for machine code object files, to load with Extended BASIC or Editor Assembler modules. Can also be loaded into TI Writer for inspection.

36 and 68 sector files MAY be fonts for THE PRINTERS APPRENTICE.

*** DISPLAY FIXED 128:

Used for RLE GRAPHICS.

MAY be ARCHIVED files which need unpacking.

Could be a special format machine code file requiring a special loader.

*** INTERNAL FIXED 128:

Used for JOYPAINT FONTS.

Also used for COMPRESSED ARCHIVED files which need uncompressing.

*** DISPLAY VARIABLE 163:

Used for Extended Basic MERGE files. Type MERGE DSK1.FILENAME.

*** DISPLAY FIXED 254 is used for Draw a Bit graphics.

*** INTERNAL VARIABLE 254 is used for:

LONG Extended BASIC programs (Use OLD DSK1.FILENAME as usual).

Data for Creative Filing System

Data for TRIO SINGS program.

Data for CSGD, watch for file names ending in /CH and /GR etc.

*** DISPLAY FIXED 255:

Used for INFOCOM data files, usually GAME1 and GAME2.

Used for Super Disk Cataloguer data files.

That will give you a start anyway!

Disk Assembler File Formats

Author unknown

DISPLAY FIXED 80 (uncompressed) Tagged Object code may be loaded by option 3 using Editor/Assembler, option 1 using Mini Memory or using CALL LOAD in TI BASIC with either the Editor/Assembler or Minimemory modules. It can be Absolute or Relocatable. The Absolute code must always be loaded at the same place in memory while Relocatable code can be loaded anywhere. If the Tagged Object code has references to other files or subroutines they will be resolved by the loader, except in the case of the Extended BASIC loader. If the source code does not contain an AORG directive then the code will be, by default, relocatable.

DISPLAY FIXED 80 (compressed) Tagged Object code is like uncompressed except that the program data is saved in bytes allowing it to load faster. It contains characters outside the printable ASCII range and cannot be modified by the Editor/Assembler editor. Compressed object files are created by using the C option with the assembler. They are NOT loadable by the Extended BASIC loader, however they are loadable by BASIC under Editor/Assembler and using Editor/Assembler option 3 etc.

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

Example: The file GAME is loaded. The loader then looks for GAMF, GAMG etc., if such files are required due to program size. Memory Image assembly files have a 3 word header followed by the data to be placed in memory as follows:

1) The first word is a 'FLAG'. If it is not 0 (zero) (for example >FFFF), then this file is not the last in a multi-file program. For example, if the flag for GAME is >FFFF then there HAS to be at least a file named GAMF, and so on.
2) This word is the length of the Memory Image file in bytes, including the six byte header. The largest value here is >2000.
3) This word is the CPU RAM address where the file is to be loaded.

Execution always begins at the first byte of the first segment loaded.

Linking Extended BASIC to Assembly

17 LINK IT 17

with Ross Mudie

1. OVERVIEW.

Many of our members are still having difficulty in coming to grips with assembly language. This article is written in simple terms to try to show how the example program operates and it also contains a very functional program which performs a novel task.

2. INTRODUCTION.

The program in this article blinks a segment of the screen, beeps and waits for a single key input with the cursor somewhere else on the screen. I wrote the program for use with an Extended BASIC/assembly program which records entrant times through check points of sporting events such as Endurance Horse Rides and Motor Bike Dirt Rider Club events. When the computer detects possible duplicated data during data entry, it alerts the keyboard operator by blinking the field which could be duplicated in the entrant's data. The program also prompts the operator with a low tone and poses the question "Issue data query Y/N" with a special cursor flashing on the "N". By pressing "N" or <ENTER> a "N" is returned in the return variable else by pressing "Y" without <ENTER> a "Y" is returned which results in a Data Query being printed under control of the main program.

The program is entered by using:

CALL LINK("BLINK",RV$,+ Up to 6 option values)

The options allow specification of Cursor Row, Cursor Column, Blink Row, Blink Column, Size of Blinking area, and number of columns in use on the screen.

The program may be used with a 32 or 40 column screen providing that the screen table starts at VDP RAM 0 and may be changed in a running Extended BASIC program. If the full argument list is not used then arguments are removed from the right hand end of the list. Missing arguments are handled by the defaults in the assembly program or once changed from Extended BASIC they remain changed whilst the program remains in memory or until they are changed again.

There is no checking of the Row and Column values by the program to detect out of screen range values since the application program actually has data input spread over three lines of 40 columns. The column value from the Extended BASIC program can be up to 100, so values greater than 40 simply go into the next or subsequent line.

3. PROGRAM DESCRIPTION.

As usual I have provided a short Extended BASIC program to demonstrate the routine and a typical CALL LINK. The documentation of the program is more narrative this time. Instead of saying what each line of the assembly program does the documentation talks about each module. The modules are separated by blank lines to help you to recognise the functional modules. The first part of the program consists of EQUates, Blocks of memory assigned with BSS and data specified with DATA, BYTE and TEXT.

After the program entry, at the label BLINK, the return address is saved and the processor is told where the Work Space registers are. After specifying the key board scan parameter, ASCII characters 126 and 127 are redefined for the blanking and the special cursor. By using the byte value at address hex 8312, the number of arguments (parameters) in the CALL LINK are determined. If there are more than 7 arguments the program could seriously malfunction, so the quantity is checked and changed to 7 if 7 is exceeded. The arguments are read into the assembly from Extended BASIC by a small loop. In assembly the screen position consists of a single value between 0 and 767 for 32 columns or 0 and 959 for 40 columns. 0 is at the top left corner and the high number is the bottom right corner. To calculate the screen position a value of 1 is subtracted from the row value and then the decremented row is multiplied by the number of columns in use on the screen, 32 or 40. The column value is then decremented and added to the row value. The same process is used to determine both the position for the cursor and the start of the blink line. The character in the cursor position and the text in the area to be blinked is then saved in buffers. The conditions are then set up to start the prompt beep and to put both the cursor and the blanking characters on the screen. A value is placed in register 3, (R3), to control sound.

The start value in R3 is 252 and the value in R3 is decremented by one each time the program loops. When the value in R3 is 250 the sound generator is loaded with the values for the required tone and turned on at the maximum volume. The program continues to put up the cursor, blank the blinking area and scan the keyboard whilst decrementing R3 on each loop. When R3 gets down to 1 the sound is turned off. When R3 gets down to zero it is not decremented any further.

R5 is used to control the flashing of the cursor and the blinking text area. When R5 contains zero both the cursor and the blanking characters are placed on the screen. A value of 700 is then placed in R5. As the program loops the value in R5 is decremented. When the value in R5 gets down to 500 the text is rewritten in the blink area and remains on the screen until R5 gets down to zero. When R5 gets down to 350 the character which was in the cursor position is placed back on the screen until R5 is down to zero. The reason for the different values is to provide different periods for the blinking of the text and the cursor. The period of overwriting of the text is much shorter than the cursor since the over writing character is very distinct and the shorter period enhances readability of the text.

On each loop the program scans the keyboard using the KSCAN utility. If no key is pressed then hex FF, equivalent of -1 with BASIC CALL KEY, is returned from the key scan. When no key is detected the program goes back to the top of the loop. If a key is detected then hex 60 is added to it and it is displayed on the screen in the cursor position. The key press is compared with Y, N and <ENTER> which are the only valid keys. If none of these keys are found the program loops back to the top. If the key was Y or N the value is placed in the CHaracter BUFFer. If it was <ENTER> then the character which was originally taken from the cursor position is has the hex 60 screen offset removed so that it is now the normal ASCII value. If the character on the screen in the cursor position was not Y or N and <ENTER> is pressed then this character will be returned, however invalid key presses will display in the cursor position whilst the key is held pressed but will be otherwise ignored.

The sound turn off byte is sent to the sound chip in case sound was still on. The character is written to the cursor position and the text is rewritten in the blinking area, in case the exit occured whilst the cursor or blanking were on the screen.

The key press is returned to Extended BASIC by the STRASG utility. The program looks again at the keyboard to ensure that the key is not still pressed. If any key is pressed the program waits in a loop for the key to be released. The program then returns to the Extended BASIC program.

4. EXTENDED BASIC PROGRAM.

The following Extended BASIC program provides a brief introduction to interfacing to the BLINK program.

```
100 ! SAVE DSK1.LOAD
110 CALL CLEAR :: CALL SCREEN(6)
120 FOR S=0 TO 12 :: CALL COLOR(S,16,1) :: NEXT S
130 CALL INIT :: CALL LOAD("DSK1.0")
140 CALL HCHAR(1,1,65,600)
150 DISPLAY AT(20,1):"Enter Y/N"
160 CALL LINK("BLINK", RV$, 20, 11, 10, 13, 4, 32)
170 DISPLAY AT(24,1):"Char in RV$="; RV$; " Asc val=";
    STR$(ASC(RV$))
180 DISPLAY AT(22,1)BEEP:"Press any key"
190 CALL KEY(0,K,S):: IF S=0 THEN 190
200 DISPLAY AT(22,1): : : :
210 CALL KEY(0,K,S):: IF S<>0 THEN 210 ELSE 160
```

5. ASSEMBLY SOURCE FILE.

```
* S=BLINK  0=0  29/6/88 Ross Mudie.
*CALL LINK("BLINK",RV$[,Cursor Row][,Cursor Col][,Blink
*Row][,Blink Col][,Size of blink line][,screen columns]

        DEF  BLINK

NUMREF EQU  >200C     Pass numeric from x/b to assembly
STRASG EQU  >2010     Pass a string from assembly to x/b
XMLLNK EQU  >2018              Handy utility routines
CFI    EQU  >12B8 Data for XMLLNK to Convert a Floating
*                  Point number to an Integer (CFI).
KSCAN  EQU  >201C        Utility to scan the keyboard
VSBW   EQU  >2020             Video Single Byte Write
VMBW   EQU  >2024           Video Multiple byte Write
VSBR   EQU  >2028             Video Single Byte Read
VMBR   EQU  >202C           Video Multiple Byte Read

FAC    EQU  >834A   Floating point ACcumulator, used by
*                                  NUMREF and XMLLNK.
STATUS EQU  >837C            GPL STATUS byte location
GPLWS  EQU  >83E0      Address of the GPL Work Space
SOUND  EQU  >8400      Address of the SOUND generator

WS     BSS  32         32 bytes for register work space
SAVRTN BSS  2         2 bytes to save return address in
CHBUFF DATA >0100    CHaracter BUFFer for key press and
* used by STRASG. 1 byte string only hence first byte 1
BLBUFF BSS  40       Blink text buffer, stores the text
*                 from the blinking area of the screen.

*                                Character re-definitions.
BLAPAT DATA >00FF,>FFFF,>FFFF,>FFFF  Blanking character
CURPAT DATA >CCCC,>CCCC,>CCCC,>FCFC      Special cursor

TONE   DATA >AD17 for about 300Hz, use >A005 for 1400Hz
T2ON   BYTE >B0    To turn Tone 2 on at 0dB attenuation
T2OFF  BYTE >BF           To turn tone generator 2 off
Y      TEXT 'Y'     A 'Y' to test against for valid key
N      TEXT 'N'     A 'N' to test against for valid key
THREE  BYTE 3         Byte of 3 used to specify keyscan
ENTER  BYTE 13   Value of the ENTER key to test against
FF     BYTE >FF Value of no key pressed to test against
       EVEN Always a good precaution after TEXT or BYTE

*           Default values if all values not specified
ARGS
CSRROW DATA 24                                 CurSoR ROW
CSRCOL DATA 1                                CurSoR COLumn
TEXTRO DATA 10  TEXT ROw for the start of blinking area
TEXTCO DATA 1    TEXT COlumn for start of blinking area
TEXTSZ DATA 4     TEXT SiZe, size of area to be blinked
SCRCOL DATA 32              Use 40 for 40 column screen
CSRPOS BSS  2 CurSoR POSition address after calculation
TEXTPO BSS  2   TEXT POsition address after calculation

BLINK  MOV  R11,@SAVRTN  Save the return address to x/b
       LWPI WS     Use the assigned Register Work Space

       MOVB @THREE,@>8374  UPPER CASE only from Keyscan

       LI   R0,1776          Location of character 126
       LI   R1,BLAPAT   Location start of redefinitions
       LI   R2,16                How many bytes to write
       BLWP @VMBW    Char 126 (Blanking)  127 (cursor)

       MOVB @>8312,R3      The byte at >8312 contains a
       SRL  R3,8    value for the number of arguments in
       CI   R3,7    the link.  It is shifted into a word
       JLE  QTYOK   sized value, tested for too many and
       LI   R3,7    changed to the maximum if  too many.

QTYOK  LI   R4,ARGS       R4 points to the start of the
       CLR  R0        argument storage area in assembly.
       LI   R1,2      R3 is the loop counter (value from
NEXTAR DEC  R3        >8312). The values are transferred
       JEQ  ENDARG    from x/b to assy by NUMREF and the
       BLWP @NUMREF floating point numbers converted to
       BLWP @XMLLNK  integers by XMLLNK  using the FAC.
       DATA CFI         The resultant integers are then
       MOV  @FAC,*R4+  placed in the appropiate storage
       INC  R1        location  (*R4)  then the next is
       JMP  NEXTAR    done. R1 points to the next arg in
*the CALL LINK whilst the + on *R4+ auto increments R4.

ENDARG MOV  @SCRCOL,R3      The value for the number of
       MOV  @CSRROW,R4   SCReen COLumns is placed in R3
       DEC  R4             the CurSoR ROW in R4 has one
       MPY  R3,R4        subtracted to make it work out.
       MOV  @CSRCOL,R4  After multiplying row number by
       DEC  R4           no of columns the byte address
       A    R4,R5         for the row is found then the
       MOV  R5,@CSRPOS column number (less 1) is added.
*         The CurSoR POSition is stored away in CSRPOS.

       MOV  @TEXTRO,R4  The start address on the screen
       DEC  R4            for the text to be blinked is
       MPY  R3,R4        calculated the same as for the
       MOV  @TEXTCO,R4     cursor, using the appropiate
       DEC  R4         values from the arguments in x/b.
       A    R4,R5        When MPY is used the resultant
       MOV  R5,@TEXTPO  value is right justified in two
*                      registers, in this case R4  R5.


       MOV  @CSRPOS,R0        Get the character off the
       BLWP @VSBR           screen in the position to be
       MOVB R1,@CHBUFF+1  used by the cursor  store it

       MOV  @TEXTPO,R0     Get the text off the area of
       LI   R1,BLBUFF     screen which is to be blinked
       MOV  @TEXTSZ,R2   store it in the BLink BUFFer.
       CI   R2,40      Watch out for blink size too big
       JLE  BLSZOK      which would over run the buffer
       LI   R2,40         load a default maximum value
       MOV  R2,@TEXTSZ                   if necessary.
BLSZOK BLWP @VMBR

       CLR  R5            When R5 is zero the cursor and
*                                 blink blanking occur.

       LI   R3,252      The value of 252 in R3 sets the
DOBEEP MOV  R3,R3     time duration of the beep.  R3 is
       JEQ  NOBEEP      decremented by one on each scan
       DEC  R3         loop until R3=zero when the rest
       CI   R3,250     of the beep routine is bypassed.
       JEQ  MAKEBP      Beep is turned on when R3=250
       CI   R3,1       off when R3=1.  By doing one key
       JEQ  BEEPOF     scan loop before turning sound on
       JMP  NOBEEP    a click is avoided if it is turned
*                                 off again immediately.

BEEPOF MOVB @T2OFF,@SOUND              Turns tone 2 off
       JMP  NOBEEP

MAKEBP MOVB @TONE,@SOUND    Sound bytes have to be sent
       MOVB @TONE+1,@SOUND   to the sound chip one at a
       MOVB @T2ON,@SOUND    time.

NOBEEP MOV  R5,R5       Test if R5 is at zero, if it is
       JEQ  BLANK     then blank text and put cursor on.
       CI   R5,500     Timing for on/off blinking ratio
       JEQ  WRITXT                        of the text.
       CI   R5,350     Timing for on/off blinking ratio
       JEQ  WRITCU                      of the cursor.
       JMP  SCANIT

BLANK  MOV  @CSRPOS,R0     This is the cursor position.
       LI   R1,>DF00      Cursor, asc 127 + >60 for x/b
       BLWP @VSBW               Write cursor on screen.
```

```
        MOV  @TEXTPO,R0  Write the blanking character in
        LI   R1,>DE00          the window being blinked.
        MOV  @TEXTSZ,R4  Hex DE is ASCII 126 with hex 60
WBLANK  BLWP @VSBW             added for x/b screen offset.
        INC  R0           R4 counts the number of blank
        DEC  R4          chars written, R0 is the screen
        JNE  WBLANK      position.
        LI   R5,700       The value in R5 sets the blink
        JMP  SCANIT    cycle rate, larger is slower etc.

WRITCU  MOV  @CSRPOS,R0  Write character back on screen
        MOVB @CHBUFF+1,R1   after cursor. The character
        BLWP @VSBW                was stored in CHBUFF+1.
        JMP  SCANIT

WRITXT  MOV  @TEXTPO,R0  Write text back on screen after
        LI   R1,BLBUFF         after the blanking period.
        MOV  @TEXTSZ,R2    R0=where on screen, R1=where
        BLWP @VMBW      text is stored  R2=no of bytes.

SCANIT  DEC  R5    Controls display of cursor  blanking
        BLWP @KSCAN       Do the actual scan of keyboard
        CB   @>8375,@FF          Test for no key pressed
        JEQ  DOBEEP                 Go to top of routine

        MOV  @CSRPOS,R0     Display a pressed key in the
        MOVB @>8375,R1    cursor position, put the ASCII
        AI   R1,>6000    value in R1, add hex 60 for the
        BLWP @VSBW     x/b screen offset and display it.

        CB   @>8375,@Y           Test for Y from keyboard
        JEQ  GOTY
        CB   @>8375,@N           Test for N from keyboard
        JEQ  GOTN
        CB   @>8375,@ENTER  Test for ENTER from keyboard
        JEQ  GOTENT
        JMP  DOBEEP        None of the above loop to scan

GOTY    MOVB @Y,@CHBUFF+1              Put Y in the CHBUFF
        JMP  FIXSCN

GOTN    MOVB @N,@CHBUFF+1              Put N in the CHBUFF
        JMP  FIXSCN

GOTENT  MOVB @CHBUFF+1,R1     If enter key get character
        AI   R1,->6000       out of CHBUFF and take away
        MOVB R1,@CHBUFF+1          the x/b screen offset.

FIXSCN  MOVB @T20FF,@SOUND    Turn off sound if still on

        MOV  @CSRPOS,R0        Put the result of the key
        MOVB @CHBUFF+1,R1   press on the screen in place
        AI   R1,>6000       of the cursor if it was still
        BLWP @VSBW                              displayed.

        MOV  @TEXTPO,R0    Restore the text in the blink
        LI   R1,BLBUFF          area in case it was still
        MOV  @TEXTSZ,R2                         blanked.
        BLWP @VMBW

        CLR  R0          Element zero in the x/b variable
        LI   R1,1             First argument in LINK list
        LI   R2,CHBUFF Where STRASG will find the string
        BLWP @STRASG                Send the string to x/b

WAIT    BLWP @KSCAN          Wait for any operated key on
        CB   @>8375,@FF      the keyboard to be released.
        JNE  WAIT

END     CLR  R0               A safe way to return to the
        MOVB R0,@STATUS                   calling program.
        LWPI GPLWS
        MOV  @SAVRTN,R11
        RT

        END                                              o
```

Wanted Wanted Wanted

Any old games cartridges, whether working or not are wanted if they have copper on both sides of the PCB (check for contacts on both sides at the edge connector). Examples are Parsec, Alpiner, TI-Invaders etc. Wanted for spare parts only. Prices negotiable. Phone (042)84 2980 7pm to 10pm and ask for Rolf.

TI-Writer Special Character Mode
Gemini Printer Control Codes

by Bob Bueiher, K-Town,USA

Gemini instructions are fully adequate for printer control in BASIC, and combined with Transliterates are fully adequate for printer control in TI-Writer Format mode. It is sometimes convenient, however, to have comparable printer control in TI-Writer Text Editor mode. This tries to supply that need.

It began months ago at a K-Town 99ers meeting. I eavesdropped on Dave Ramsey telling Mrs. Lovelace that page 98 of TI-Writer provided the key to Text Editor printer control. On request, Dave sent to me his "CHARMODE", applicable to a Panasonic 1091 printer. It includes print control symbols for some 30 printer functions. Glory be!! Many of them worked with my Gemini. Yet full pitch control and underlining did not work. So I wrote to Star Micronics on November 13, 1985 for help. The answers came back on January 13, 1986.

With the Function Code Summary of pages 242 through 245 of the Gemini Manual as a basis, note the following:

```
        0      CTRL[U] SHIFT[2] CTRL[U]
        1      CTRL[U] SHIFT[A] CTRL[U]
        2      CTRL[U] SHIFT[B] CTRL[U]
     etc, (C is for 3, D is for 4)
        26     CTRL[U] SHIFT[Z] CTRL[U]
        ESC=27 CTRL[U] FCTN[R] CTRL[U]
        FS=28  CTRL[U] FCTN[Z] CTRL[U]
        GS=29  CTRL[U] FCTN[T] CTRL[U]
        RS=30  CTRL[U] SHIFT[6] CTRL[U]
        US=31  CTRL[U] FCTN[U] CTRL[U]
```

There now follows function codes to control printing styles, including line spacing. Some printing styles can be controlled in more than one way. Controls can be combined (see below) or set anywhere in your text file.

```
10 cpi          CTRL[U] FCTN[R] CTRL[U] B
                                CTRL[U] SHIFT[A] CTRL[U]
12 cpi          CTRL[U] FCTN[R] CTRL[U] B
                                CTRL[U] SHIFT[B] CTRL[U]
17 cpi          CTRL[U] FCTN[R] CTRL[U] B
                                CTRL[U] SHIFT[C] CTRL[U]
Set Power On    CTRL[U] FCTN[R] CTRL[U] SHIFT[2]
                        (Releases prior settings)
Set 1 Line
      Double    CTRL[U] SHIFT[N] CTRL[U]
                        (SHIFT[N]=ASCII 14,pg 146)
    Release     CTRL[U] SHIFT[T] CTRL[U]
                      (1< line,SHIFT[T]=20)
Condensed       CTRL[U] SHIFT[O] CTRL[U]
                        (SHIFT[O]=15)
    Cancel      CTRL[U] SHIFT[R] CTRL[U]
                        (SHIFT[R]=18)
Double Width    CTRL[U] FCTN[R] CTRL[U] W
                                CTRL[U] SHIFT[A] CTRL[U]
    Cancel      CTRL[U] FCTN[R] CTRL[U] W
                                CTRL[U] SHIFT[2] CTRL[U]
Condensed Dbl   CTRL[U] SHIFT[O] SHIFT[N] CTRL[U]
Underline       CTRL[U] FCTN[R] CTRL[U] -
                                CTRL[U] SHIFT[A] CTRL[U]
    Cancel      CTRL[U] FCTN[R] CTRL[U] -
                                CTRL[U] SHIFT[2] CTRL[U]
Emphasized      CTRL[U] FCTN[R] CTRL[U] E
    Cancel      CTRL[U] FCTN[R] CTRL[U] F
Double Strike   CTRL[U] FCTN[R] CTRL[U] G
    Cancel      CTRL[U] FCTN[R] CTRL[U] H
Italics         CTRL[U] FCTN[R] CTRL[U] 4
    Cancel      CTRL[U] FCTN[R] CTRL[U] 5
Superscript     CTRL[U] FCTN[R] CTRL[U] S
                                CTRL[U] SHIFT[2] CTRL[U]
Subscript       CTRL[U] FCTN[R] CTRL[U] S
                                CTRL[U] SHIFT[A] CTRL[U]
Sub/Super
    Cancel      CTRL[U] FCTN[R] CTRL[U] T
Line Spacing
  1/8           CTRL[U] FCTN[R] CTRL[U] 0
  7/72          CTRL[U] FCTN[R] CTRL[U] 1
  1/6           CTRL[U] FCTN[R] CTRL[U] 2
  n/144         CTRL[U] FCTN[R] CTRL[U] 3 n
```

See TI-Writer manual page 145 for 'n' values of 33 to 126, and page 146 for 'n' values of 0 to 31. o

SEARCH Program

from Stephen Shaw, England

Here is a tiny program which will search through an XB program for a variable name, keyword, or quoted string, or line number, or just about ANY text- for example, FOR I=1 TO or CALL KEY(3 or IF A>B AND..... up to 100 occurences will be listed for you.

This is the source code which is to be assembled into the file FIND/OB:

```
        DEF  FIND
START   EQU  >8330  * START OF LINE NO TABLE
FINISH EQU  >8332  * END OF LINE NO TABLE
XMLLNK EQU  >2018  * FOR EX BAS USE !!!
NUMASG EQU  >2008
FAC     EQU  >834A
GOTO    BYTE 134,0  * Filter GOTO
FIND    BLWP @FIND1
        RT
FIND1   DATA FINDWS,FIND2
FINDWS BSS  32
FIND2   MOV  @START,R1
GETPAT BL   @MOVE
        CI   R6,32000 * PATTERN IN LINE 32000
        JEQ  GOTPAT
        JL   ERROR
        AI   R1,4
        JMP  GETPAT
ERROR   LI   R0,>2100  * DATA ERROR
        BLWP @>2034
GOTPAT INCT R1
        BL   @MOVE
        MOV  R1,R8
        MOV  R6,R9
        DEC  R9
        CLR  R10
        MOVB *R9+,R10  * PATTERN LENGTH TO R10
        SWPB R10
        CB   *R9,@GOTO * GOTO FILTER
        JNE  NO
        INC  R9
        DEC  R10
NO      MOV  @FINISH,R2
        DEC  R2
        CLR  R0
LOOP    C    R2,R8
        JEQ  DONE
        MOV  R2,R1
        BL   @MOVE
        MOV  R6,R3
        DEC  R3
        CLR  R15
        MOVB *R3+,R15
        SWPB R15
LOOP2   DEC  R15
        JEQ  NOMTCH
        MOV  R3,R4
        MOV  R9,R5
        MOV  R10,R12
LOOP3   DEC  R12
        JEQ  MATCH
        CB   *R5+,*R4+
        JEQ  LOOP3
        INC  R3
        JMP  LOOP2
MATCH   MOV  R2,R1
        DECT R1
        BL   @MOVE    * LINE NO TO R6
        MOV  R6,@FAC  * TO F/PT ACCUM
        BLWP @XMLLNK  * TO F/POINT NUMBER
        DATA >20
        INC  R0
        LI   R1,1
        BLWP @NUMASG  * LINE NO TO BASIC ARRAY
NOMTCH AI   R2,-4
        JMP  LOOP
DONE    RTWP
MOVE    INC  R1
        MOVB *R1,R6
        SWPB R6
        DEC  R1
        MOVB *R1,R6
        RT
        END
```

The listing which follows is in Extended BASIC and should be saved in MERGE format. To find a section of an Extended BASIC program, load the program, then MERGE this in to it. Then type:

32000 and the thing you are searching for, eg

32000 "A" (string only)

32000 A (string + var. name)

32000 GOSUB (all GOSUBs)

there are a few things Extended BASIC will not permit you to type in. There is also an exception- line numbers. To find a NUMBER 100 use:

32000 100

To find any reference to LINE NUMBER 100 however (for instance, GOTO,GOSUB, IF, etc) you must use:

32000 GOTO 100

The first entry of GOTO on line 32000 is ignored. To search for all GOTOs, use:

32000 GOTO GOTO

The program:

```
1 CALL FIND :: !@P-
32000 !
32010 !@P+
32020 SUB FIND
32030 ! SEARCH FOR CONTENT OF LINE 32000
32040 DIM A(100)
32050 ON ERROR 32080 :: CALL LINK("FIND",A()) :: ON
    ERRO R STOP
32060 FOR I=1 TO 100 :: IF A (I)=0 THEN STOP
32070 PRINT A(I); :: NEXT I :: STOP
32080 CALL ERR(B,C) :: IF B= 84 THEN PRINT "ERROR:
    INSERT PATTERN IN 32000" :: STOP
32090 IF B=57 THEN RETURN 32060 ! SUBSCRIPT ERROR
32100 IF B=14 OR B=135 THEN CALL INIT :: CALL
    LOAD("DSK1 .FIND/OB") :: RETURN 32050
32110 RETURN
32120 SUBEND
```

It IS possible to transfer the machine code into CALL LOAD format, and add it to the MERGE file, but on a long Extended BASIC program, this adds considerably to both the time taken to MERGE the file in, and then to RUN it.

This code is from STEVEN KARASEK and was first published in:

COMPUTER BRIDGE Vol 5 No 11 November 86

Retyped by sjs. Queries/complaints to S Shaw!

The utility uses the on-board tokenisation facilities on line 32000 and compares the tokenised form to the entire program in memory.

NOTE: If your BASIC program contains a line with !@P+ in it, those symbols will almost certainly have to be removed in order for this utility to run properly!

Repeated use of this program alternating with running of your Extended BASIC program is possible just by removing lines 1 and 32000 to RUN YOUR Extended BASIC program, and by re- inserting lines 1 and 32000 to again use the utility. The CALL LOAD will not be repeated provided your program does not have a CALL INIT in it... this will save on the frustrations of the speed of MERGEing....

64K Memory on Multifunction Card

by Peter Schubert

With a simple upgrade the A.T MFC can have an extra bank of 32K memory added. Switching from one bank of 32K to the other is simply a matter of writing to CRU address >111A. Changing the value of zero here to a one will change banks, and vice-versa to return. It is planned to add a CALL COMMAND to the MFC EPROM so the memory can also be switched from BASIC or XB.

Hopefully some interesting uses will be found for this extra memory, and perhaps some modifications to existing programs can take advantage of the extra space. For example how about an extra 32K of receive buffer when using your favourite Terminal Program at 1200 or 2400 baud? Or perhaps a Hot Key that can be used anytime to activate a CALL CATALOGUE function without disrupting the program you are using? There are many possibilities for our talented programmers.

The cost of the extra 32K fitted to your MFC is $40.

Jenny's Younger Set

Dear Jenny,
Here is a mathematics program designed for primary school students. I hope it is satisfactory for the TND.

Yours faithfully
Vincent Maker

```
100 ON WARNING NEXT
110 REM **********************
120 REM *                    *
130 REM *JUNIOR MATHEMATICS*
140 REM *                    *
150 REM *        BY          *
160 REM *                    *
170 REM *                    *
180 REM *  VINCENT  MAKER    *
190 REM *                    *
200 REM *                    *
210 REM *     DESIGNED       *
220 REM * SPECIFICALLY FOR   *
230 REM *                    *
240 REM *  PRIMARY SCHOOL    *
250 REM *      STUDENTS      *
260 REM *                    *
270 REM *  FOR MR JOHNSON    *
280 REM **********************
290 RANDOMIZE
300 A=0
310 B=0
320 C=0
330 D=0
340 E=0
350 F=0
360 G=0 :: H=0
370 RIGHT=0 :: WRONG=0
380 A=INT(10*RND)
390 B=INT(10*RND)
400 C=INT(20*RND)
410 D=INT(10*RND):: IF C>D THEN 110
420 E=INT(20*RND)
430 F=INT(15*RND)
440 G=INT(18*RND)
450 H=INT(10*RND):: IF H>G THEN 110
460 I=INT(10*RND)
470 CALL CLEAR
480 INPUT "YOUR NAME PLEASE? ";NAME$
490 PRINT
500 CALL CLEAR
510 PRINT "Q1.";A;"+";B;"="
520 INPUT K
530 IF K=A+B THEN RIGHT=1 ELSE WRONG=1
540 IF K=A+B THEN PRINT "VERY GOOD, ";NAME$ ELSE PRINT
    "WORK ON ADDITION, "&NAME $&". YOU GOT THAT ONE
    WRONG"
550 PRINT "Q2.";D;"-";C;"="
560 INPUT L
570 IF L=D-C THEN RIGHT=RIGHT+1 ELSE WRONG=WRONG+1
580 IF L=D-C THEN PRINT "WELL DONE! RIGHT" ELSE PRINT
    "WORK ON THAT SUBTRACTION, ";NAME$&". YOU GOT THAT
    ONE WRONG"
590 PRINT "Q3.";E;"*";F;"="
600 INPUT M
610 IF M=E*F THEN RIGHT=RIGHT+1 ELSE WRONG=WRONG+1
620 PRINT "Q4.";G;"/";H;"="
630 INPUT DF
640 IF DF=G/H THEN RIGHT=RIGHT+1 ELSE WRONG=WRONG+1
650 IF DF=G/H THEN PRINT "WELL DONE, RIGHT" ELSE PRINT
    NAME$&", WORK ON THAT DIVISION."
660 PRINT "THIS IS A HARD ONE:"; I; "+"; H; "-"; D;
    "*"; C; "="
670 INPUT GD
680 IF GD=I+H-D*C THEN RIGHT=RIGHT+1 ELSE WRONG=WRONG+1
690 IF GD=I+H-D*C THEN PRINT "EXTREMELY WELL DONE.
    CONGRATULATIONS, ";NAME$
700 CALL CLEAR
710 DISPLAY AT(3,1):"RESULTS OF YOUR PERFORMANCE"
720 IF RIGHT=5 THEN RESULT$="EXTREMELY WELL DONE!!,
    "&NAME$&". ALL CORRECT (5 RIGHT)."
730 IF RIGHT=4 THEN RESULT$="WELL DONE, YOU MADE ONLY
    ONE ERROR (4 RIGHT)."
740 IF RIGHT=3 THEN RESULT$="FAIR PERFORMANCE. YOU
    MADE TWO ERRORS (3 RIGHT)."
750 IF RIGHT=2 THEN RESULT$="POOR PERFORMANCE. YOU
    MADE THREE ERRORS (2 RIGHT). "
760 IF RIGHT=1 THEN RESULT$="VERY POOR PERFORMANCE.
    YOU MADE FOUR ERRORS (1 RIGHT)."
770 IF RIGHT=0 THEN RESULT$="EXTREMELY POOR
    PERFORMANCE. YOU DID NOT GET A QUESTION CORRECT (0
    RIGHT)."
780 IF RIGHT=0 THEN PERCENTAGE=0
790 IF RIGHT=1 THEN PERCENTAGE=20
800 IF RIGHT=2 THEN PERCENTAGE=40
810 IF RIGHT=3 THEN PERCENTAGE=60
820 IF RIGHT=4 THEN PERCENTAGE=80
830 IF RIGHT=5 THEN PERCENTAGE=100
840 DISPLAY AT(5,1):RESULT$
850 DISPLAY AT(9,1):"YOUR PERCENTAGES ARE..."
860 XP=100-PERCENTAGE
870 DISPLAY AT(10,1):"PERCENTAGE CORRECT"; PERCENTAGE;
    "PERCENTAGE INCORRECT"; XP
880 INPUT "WOULD YOU LIKE TO PLAY AGAIN(Y/N)?": G$
890 IF G$<>"Y" THEN END
900 GOTO 110
```

Dear Jenny,
I have just thought of how I could improve my mathematics program that I sent in last month for a clear screen when the program is run. A line 45 could be inserted thus:

```
45 CALL CLEAR
```

for a clear screen every equation. Lines 330 and 340 could be changed thus:

```
330 IF B$="Y" THEN 45
340 IF B$="YES" THEN 45
```

If you plan using this program on Extended BASIC, line 40 is advisable.

```
40 ON WARNING NEXT
```

Then, of course, lines 330 and 340 would have to go back to 40 rather than 45. This helps in cases where a numeric overflow might occur.

Yours faithfully
Vincent Maker

Well Vincent, thank you for your letters and the program. I had a bit of time so I tried out your program and made a few little changes. Unfortunately I could not see how to put in your improvements and I suspect that we may have different versions of the program. I did find some problems while running the program, particularly in the division question. Once, I was asked for the answer for 7/0, which I thought was a bit tough. Also dividing by 7 is a bit hard if the answer is not an integer. How about everyone having a go at this and suggesting improvements to this program. You never know, I might just find a prize for the best effort!

16 Peveril Street
3150 Glen Waverley
Melbourne, Vic
12/06/88

Dear Jenny,
My name is Ian, I am 11.5 years old. I enjoy your Younger set column very much. I have had my Texas Instruments Home Computer for 4.5 years. I do quite a bit of BASIC programming. One of the programs I have created is SHOOTOUT (the program below).

INSTRUCTIONS:

When you run the program the title screen will appear, press any key to start the game. In the upper left hand corner you will see the current score. You are the pink square and the two blue squares are your enemies. To move you must use the following keys:

(I) to move up
(M) to move down
(J) to move left
(K) to move right

You can shoot with the following keys:

(E) to shoot up
(X) to shoot down
(S) to shoot left
(D) to shoot right

Your enemies cannot shoot in any direction. You can move only one space at a time, but your enemies can move one or two spaces at a time. If you manage to shoot both the enemies you go up a level, which gives you bonus points. Every time you move or shoot you lose one point. Good luck!

EXPLANATION OF THE PROGRAM:

lines 100-130 are remarks
line 140 ON ERROR in case you move off the screen, which means that you lose.
lines 150-170 title screen
lines 180-230 start of main program
lines 240-260 character definition statements
lines 270-1100 main program
lines 1120-1160 you lose and end of game
lines 1170-1370 subroutines

When typing the program you should leave out the ON ERROR 1110 statement on line 140, in case you make some typing errors, and type the statement when you have debugged the program.

Yours sincerely
Ian Szemere

Thank you Ian for your words of encouragement, but especially for your excellent contribution. I hope that there are others out there who also get some pleasure out of this part of the magazine. It only appears because of you and Vincent and the others who send me letters. If there are others out there reading this, do not be shy, we are all friendly here, send us a letter with your programming efforts or your game scores, or even any problems that you have come across. Here is Ian's program.

```
100 REM ************
110 REM * SHOOTOUT *
120 REM ************
130 REM BY IAN SZEMERE
140 ON ERROR 1110
150 CALL CLEAR :: DISPLAY AT(5,11): "SHOOTOUT"
160 DISPLAY AT(23,1):"PRESS ANY KEY TO CONTINUE"
170 CALL KEY(0,K,S):: IF S=0 THEN 170
180 RANDOMIZE
190 LEV=1 :: SCO=0
200 P=0
210 H=0
220 X=0
230 GOSUB 1280
240 CALL CHAR(88,"FFFFFFFFFFFFFFFF")
250 CALL CHAR(96,"FFFFFFFFFFFFFFFF")
260 CALL CHAR(104,"FFFFFFFFFFFFFFFF")
270 A=10
280 B=12
290 C=20
300 D=10
310 E=9
320 F=20
330 CALL COLOR(8,14,1):: CALL COLOR(9,5,1) :: CALL
    COLOR(10,7,1)
340 CALL CLEAR
350 DISPLAY AT(1,1):"SCORE=";SCO
360 IF P=300 AND H=300 THEN GOTO 1210
370 CALL HCHAR(A,B,88)
380 CALL SOUND(25,252,0)
390 IF P=300 THEN GOTO 410
400 CALL HCHAR(C,D,96)
410 IF H=300 THEN GOTO 430
420 CALL HCHAR(E,F,96)
430 IF P=300 AND H=300 THEN GOTO 1210
440 CALL KEY(0,K,S):: IF S=0 THEN 440
450 G$=CHR$(K)
460 CSO=SCO/1
470 IF G$="1" THEN A=A-1
480 IF G$="J" THEN B=B-1
490 IF G$="K" THEN B=B+1
500 IF G$="M" THEN A=A+1
510 IF G$="E" THEN GOTO 700
520 DISPLAY AT(1,1):"SCORE=";SCO
530 IF G$="X" THEN GOTO 790
540 IF G$="S" THEN GOTO 880
550 IF G$="D" THEN GOTO 970
560 IF P=300 THEN GOTO 610
570 IF A<C THEN C=C-1
580 IF A>C THEN C=C+1
590 IF B>D THEN D=D+1
600 IF B<D THEN D=D-1
610 IF H=300 THEN GOTO 660
620 IF A>E THEN E=E+1
630 IF A<E THEN E=E-1
640 IF F>B THEN F=F-1
650 IF F<B THEN F=F+1
660 IF A=C AND B=D THEN GOTO 1110
670 IF A=E AND B=F THEN GOTO 1110
680 GOTO 1060
690 GOTO 330
700 X=B
710 Y=A
720 Y=Y-1
730 IF Y=0 THEN GOTO 330
740 IF Y=C AND X=D THEN GOTO 1170
750 IF Y=E AND X=F THEN GOTO 1190
760 CALL HCHAR(Y,X,104)
770 GOSUB 1360
780 GOTO 720
790 X=B
800 Y=A
810 Y=Y+1
820 IF Y=24 THEN GOTO 330
830 IF Y=C AND X=D THEN GOTO 1170
840 IF Y=E AND X=F THEN GOTO 1190
850 CALL HCHAR(Y,X,104)
860 GOSUB 1360
870 GOTO 810
880 X=B
890 Y=A
900 X=X-1
910 IF X=0 THEN GOTO 330
920 IF Y=C AND X=D THEN GOTO 1170
930 IF Y=E AND X=F THEN GOTO 1190
940 CALL HCHAR(Y,X,104)
950 GOSUB 1360
960 GOTO 900
970 X=B
980 Y=A
990 X=X+1
1000 IF X=32 THEN GOTO 330
1010 IF Y=C AND X=D THEN GOTO 1170
1020 IF Y=E AND X=F THEN GOTO 1190
1030 CALL HCHAR(Y,X,104)
1040 GOSUB 1360
1050 GOTO 990
1060 IF A=C THEN RET=INT(RND*LEV)+1 :: IF RET<>1 THEN
     C=C+1
1070 IF B=D THEN RET=INT(RND*LEV)+1 :: IF RET<>1 THEN
     D=D+1
1080 IF A=E THEN RET=INT(RND*LEV)+1 :: IF RET<>1 THEN
     E=E+1
1090 IF B=F THEN RET=INT(RND*LEV)+1 :: IF RET<>1 THEN
     F=F+1
1100 GOTO 690
1110 CALL CLEAR
1120 DISPLAY AT(1,1):"YOU LOSE"
1130 DISPLAY AT(2,1):"ON LEVEL";LEV-1
1140 DISPLAY AT(3,1):"SCORE=":SCO
1150 CALL KEY(0,K,S):: IF S=0 THEN 1150
1160 CALL CLEAR :: END
1170 P=300 :: C,D=0
1180 GOTO 330
1190 H=300 :: E,F=0
1200 GOTO 330
1210 CALL CLEAR
1220 DISPLAY AT(1,1):"LEVEL";LEV
1230 SCO=SCO+100
1240 SCO=SCO+QWE
1250 FOR TIM=1 TO 500
1260 NEXT TIM
1270 GOTO 200
1280 CALL CLEAR
1290 IF LEV>1 THEN GOTO 1300
1300 QWE=150
1310 LEV=LEV+1
1320 FOR DE=1 TO 50
1330 NEXT DE
1340 CALL CLEAR
1350 RETURN
1360 CALL SOUND(10,500,0)
1370 RETURN
```

For Sale

2 slot mini PE box + TI Disk controller + TI Disk managermodule + 3.5 inch disk drive + 20 3.5 inch disks with games + Axiom parallel printer interface - $240

Extended BASIC module + manual - $40

Contact Pieter Moerkerken at work on (02)663 0139 or send mail to SYSOP.

Games Information

by Robert Brown and Stephen Judd

Welcome to a bumper issue of GAMES INFO. Last month's article was not compiled due to difficulties that cannot be revealed and discussed in this family column, so for this month we have created a bigger issue for your enjoyment. In this huge issue we look at the entries and give the solution to the TI RUNNER COMPETITION. Also we have our world famous reviews on TI GAMES. This month Sewermania and Star Trek.

Now to begin. Our TI RUNNER COMPETITION created a enormous buzz in the TI99/4A community. We felt sincerely sorry to the other members who did not enter so we decided to just give up and give you the solution and how to obtain unlimited misses.

Quite a lot of the general entries had 5 or 6, but nobody had the complete 10 answers. Sure, lots of people picked the easy ones, but there were other ones that required a trained eye to spot. These are the 10 screens:

SCREEN 9 - A castle
SCREEN 20 - An Atari symbol
SCREEN 27 - A blue person
SCREEN 28 - An IBM symbol
SCREEN 33 - A boat
SCREEN 34 - An Apple symbol
SCREEN 36 - JR - The Music Man
SCREEN 41 - Another castle
SCREEN 45 - A head
SCREEN 49 - The TI symbol

Although these screens are sometimes a bit hard to recognize, it doesnt really matter cause everybody gets the prize, as follows. To do this you must use the converted TI RUNNER (from the BBS).

1. Load and Run TI RUNNER
2. Once at title screen press FCTN QUIT
3. Select Extended BASIC and wait for ready prompt.
4. Type in the following :

```
CALL LOAD(-18146,16,1) <ENTER>
CALL LINK("RUNNER") <ENTER>
```

And presto TI RUNNER will start with unlimited misses. If you do no have this TI RUNNER version, leave a message for games on the BBS, or talk to Robert Brown (see address at end). Games Info would like to thank GEORGE MELDRUM for supplying this useful technique.

Now here is Star Trek...

Star Trek is an arcade style program available on many computers and Sega were smart enought to produce it on the TI. Joystick and keys may be used, but if you choose joystick it must be #2. The speech synthesization (breath) is not bad, but could be better. Graphics are very average and sounds are OK though game excecution is good.

If you choose to use keyboard here are the controls.....

S ---- rotate left
D ---- rotate right
K ---- forward (impulse power)
H ---- forward (warp drive)
J/Y -- phasers
L ---- photon torpedoes
P ---- pause (any key to resume)
8 ---- new game
9 ---- return to title

Now if you chose joystick....

LEFT/RIGHT ----- LEFT/RIGHT
FORWARD ----- impulse power
BACK+FIRE ------ warp drive

As captain of the starship Enterprise you have a range of weapons and power. The following is a manual for the use of choosing weapons or power.

Weapon	Purpose	Keyboard	Joystick	Supply	+ Maximum
Phaser	Destroy	J or Y	Fire	Unlimited	
Photon Torpedo	Kill in explosion area	L	Back	5	Unlimited
Warp Drive	Forward fast	H	Back+Fire	5	10
Energy	Protect you	Auto	Auto	5	Unlimited

Your mission is to destroy Nomad. To get there you must survive many attacks.

Once at Nomad you must get through the mine field and destroy Nomad before it lays the 9th mine. If the 9th mine is laid all the others are detonated. For every 10,000 points you earn an extra energy shield, photon torpedo and warp drive unit.

At the end of each round all gauges are resupplied for the next round. Each round is represented by a sector number up to 6. Round 1 would be sector 1.1 and round 8 would be sector 2.2.

Red and yellow Klingon battle cruisers will attack you. First with fire then will turn white and try to ram you. You will lose 1 shield per hit and 2 when rammed. If you have no shields left, a ram will lose all your photon and warp units. If you have no shield or photons left, a ram will detroy you. At higher levels Klingons turn white quicker. Yellow Klingons attack your green star bases, if they accomplish this they will change colour and attack you. Blue anti-matter saucers attach themselves to your ship and drain warp energy. If, when attached, you have no warp energy left they will disappear.

Nomad can only be destroyed by phaser and only appears in the last attack of each sector. It scatters a protective layer of 8 mines and when the 9th is laid they will all detonate and you will move onto the next sector with no bonus. Scoring is as follows...

Klingons	round number * 25
Anti-matter saucer	5000
Nomad	30000 + round * 25
Used starbases	round * 250
Unused starbases	round * 1000

It is good to keep moving always and try to dock a lot at the early stages of the game. Also fly over remainging starbases before destroying your last Klingon. You will get more bonus points in higher levels by not using starbases. To destroy anti-matters turn your ship toward them and fire. Try not to get rammed on your side. Use impulse power and save warp for emergencies.

If you get very good you will dicover that all sectors are basically the same except for speed and amount of enemies.

And now for Sewermania (Requested by ALF RUGGERI.)

Sewermania is a Milton Bradley program which can use the MBX Speech Recognition system, however the MBX system is not a necessaty.

This is a 2 screened game in a Sewer. To control your man, use the joystick or standard keys. The idea of the game is to find a bomb hidden in the sewer and take it to the surface.

To start the game you will get dropped off the bus and are to run to a manhole before you get run over. To open the manhole, press fire when near it, and it will open automatically.

To do this, you must run around the maze avoiding the crocodiles and the red rats. It is possible to kill the black rats only using a stick which you find around the maze. You also have the power to lock and unlock doors to protect yourself. To find the bomb you must search all over the maze for it. It will show up and flash when you run over it.

When you find the bomb you have 30 seconds to pick it up and take it to the surface to safety, but you have to wait until all the cars have left the surface so you do not get run over and injure the people driving the cars.

continued on page 18

```
100 REM ********************
***
110 REM * TYPING FOR ACCURAC
Y *
120 REM ********************
***
130 REM 99'ER VERSION 9.81.1

140 REM BY REGENA
150 REM
160 REM
170 DIM A$(20)
180 Y=20
190 DATA 15,FAD,A,AS,DAD,AD,
SAD,LAD,FALL,ALFALFA,SASS,LA
SS,DADS,LADS,FALLS,FADS
200 DATA 16,HAD,HAS,GAS,SAG,
HALL,HALLS,LADS,SAGS,HAG,LAG
,LAGS,SLAG,SHALL,SASH,DASH,F
LASH
210 DATA 17,DEAF,DEED,SEED,F
EED,HEED,LIKE,KILL,FILL,FEEL
,FEES,LIED,DIAL,SLIDE,FLIES,
LIFE,SLID,GLIDE
220 DATA 17,TAG,HAT,TALL,THY
,DAY,HAY,JAY,GAY,LAY,TAR,RAT
,STAR,STAFF,FAST,TRY,SAY,YAR
D
230 DATA 20,WISH,EXACT,TEXT,
TWO,WON,SOW,WASH,WORLD,OWE,W
ORD,LOOK,LOSE,SOD,WOW,TOW,TE
XAS,OXEN,MIX,WORSE
240 DATA 16,QUAKE,QUIZ,QUIP,
ZAP,QUIT,PIQUE,PLAQUE,PUZZLE
,PLAZA,SAP,ZIPPER,PRIZE,QUIC
K,SQUEEZE,ZEAL,ZIP
250 DATA 18,CALM,CAN,MEN,NIM
BLE,EXACT,EXAM,MIX,NIX,BUZZ,
ZOOM,NAVY,CAB,BACK,BOMB,ZOMB
IE,CAVE,VACATE,VARMINT
260 GOSUB 970
270 GOSUB 1590
280 GOSUB 1740
290 CALL KEY(0,KEY,STS)
300 IF KEY<49 THEN 290
310 IF KEY>56 THEN 290
320 ON (KEY-48)GOTO 330,350,
370,390,410,430,450,960
330 RESTORE 190
340 GOTO 460
350 RESTORE 200
360 GOTO 460
370 RESTORE 210
380 GOTO 460
390 RESTORE 220
400 GOTO 460
410 RESTORE 230
420 GOTO 460
430 RESTORE 240
440 GOTO 460
450 RESTORE 250
460 CALL CLEAR
470 CALL COLOR(1,2,1)
480 CALL SCREEN(8)
490 GOSUB 1940
500 READ N
510 FOR I=1 TO N
520 READ A$(I)
530 NEXT I
540 SCORE=0
550 RANDOMIZE
560 FOR K=1 TO 10
570 W=INT(N*RND)+1
580 IF A$(W)="0" THEN 570
590 XC=24-K
600 CALL SOUND(1500,-1,2)
610 FOR J=1 TO LEN(A$(W))
620 CALL HCHAR(XC,J+17,ASC(S
EG$(A$(W),J,1)))
630 NEXT J
640 INPUT B$
650 IF B$=A$(W)THEN 680
660 CALL SOUND(1000,-7,1)
670 GOTO 700
680 CALL SOUND(1000,-2,1)
690 SCORE=SCORE+1
700 IF K<>1 THEN 760
710 CALL HCHAR(23,Y-4,98)
720 CALL HCHAR(23,Y+4,98)
730 CALL HCHAR(23,Y-3,99)
740 CALL HCHAR(23,Y+3,100)
750 GOTO 790
760 IF K<>2 THEN 790
770 CALL HCHAR(23,Y-4,99)
780 CALL HCHAR(23,Y+4,100)
790 CALL HCHAR(23,Y-1,105,3)
800 CALL SOUND(1,44000,30)
810 CALL HCHAR(XC-1,Y-2,98,7
)
820 CALL HCHAR(XC-1,Y+5,32,3
)
830 CALL HCHAR(23,1,32,15)
840 A$(W)="0"
850 NEXT K
860 CALL CLEAR
870 FOR I=2 TO 8
880 CALL COLOR(I,2,1)
890 NEXT I
900 SC=10*SCORE
910 PRINT : : : :"YOUR SCORE
 IS":SC,"PERCENT ACCURACY."
920 PRINT : : :"PRESS ENTER
TO CONTINUE."
930 CALL KEY(0,KEY,ST)
940 IF KEY<>13 THEN 930
950 GOTO 280
960 END
970 CALL CLEAR
980 CALL SCREEN(5)
990 T=500
1000 CALL SOUND(T,880,3,698,
8,294,10)
1010 PRINT : : :TAB(11);"T Y
 P I N G"
1020 CALL CHAR(104,"FFFF00FF
FF00FFFF")
1030 CALL SOUND(T,932,3,784,
8,196,11)
1040 PRINT : :TAB(15);"FOR"
1050 CALL COLOR(10,16,6)
1060 CALL COLOR(11,13,1)
1070 CALL SOUND(T,784,3,659,
8,262,10)
1080 PRINT : :TAB(12):"ACCUR
ACY"
1090 CALL CHAR(112,"00E0F8FE
FFFFFFFF")
1100 CALL SOUND(T,880,3,698,
8,175,12)
1110 CALL CHAR(113,"00000000
80E0F8FE")
1120 CALL CHAR(114,"80E0FCFF
FFFCE08")
1130 CALL CHAR(115,"FEF8E08"
)
1140 CALL SOUND(T,698,3,587,
8,233,10)
1150 PRINT : : : : : : :TAB(
15);"BY"
1160 PRINT :(13);"REGENA"
1170 CALL CHAR(116,"FFFFFFFF
FEF8E")
1180 CALL SOUND(T/2,784,3,16
5,10)
1190 CALL CHAR(96,"80C0E0F8F
8E0C08")
1200 CALL SOUND(T/2,698,3,16
5,10)
1210 CALL CHAR(97,"80C0E0F0F
8FCFEFF")
1220 CALL SOUND(T/2,659,3,27
7,10)
1230 CALL CHAR(98,"FFFFFFFFF
FFFFFF")
1240 CALL CHAR(117,"FFFFFFFF
FFFFFFFF")
1250 CALL SOUND(T/2,784,3,27
7,10)
1260 CALL CHAR(99,"FFFEFCF8F
0E0C08")
1270 CALL SOUND(T*2,698,2,58
7,8,147,12)
1280 CALL HCHAR(15,7,117,7)
1290 CALL HCHAR(16,7,117,9)
1300 CALL HCHAR(17,7,117,7)
1310 CALL HCHAR(16,16,114)
1320 CALL HCHAR(15,15,113)
1330 CALL HCHAR(17,15,115)
1340 CALL HCHAR(15,14,112)
1350 CALL HCHAR(17,14,116)
1360 CALL SOUND(T,466,4,165,
10)
1370 FOR XX=15 TO 17
1380 CALL HCHAR(XX,1,104,6)
1390 NEXT XX
1400 CALL SOUND(T,440,3,175,
10)
1410 CALL COLOR(2,16,1)
1420 FOR YY=18 TO 26 STEP 2
1430 CALL HCHAR(16,YY,42)
1440 NEXT YY
1450 CALL SOUND(T,698,3,440,
8,294,10)
1460 CALL HCHAR(16,29,49)
1470 CALL HCHAR(16,29,48,2)
1480 CALL HCHAR(16,31,37)
1490 CALL SOUND(T,784,3,587,
8,233,10)
1500 CALL CHAR(100,"7F3F1F0F
070301")
1510 CALL CHAR(101,"00010307
0F1F3F7F")
1520 CALL SOUND(T/2,698,3,39
2,8,262,10)
1530 CALL CHAR(102,"00001818
3C3C7EFF")
1540 CALL SOUND(T/2,659,2,26
2,10)
1550 CALL CHAR(105,"DBDBDBDB
DBDBDBDB")
1560 CALL SOUND(4*T,698,2,44
0,8,175,10)
1570 CALL SOUND(1,44000,30)
1580 RETURN
1590 CALL CLEAR
1600 CALL SCREEN(7)
1610 CALL COLOR(2,2,1)
1620 PRINT "PICK A TYPING CA
TEGORY."
1630 PRINT : : :"YOU WILL SE
E A WORD"
1640 PRINT "IN THE ROCKET."
1650 PRINT :"TYPE AND ENTER
IT BEFORE THE TONE ENDS."
1660 PRINT :"IF YOU ARE CORR
ECT,"
1670 PRINT "ANOTHER TONE SOU
NDS,"
1680 PRINT :"IF YOU ARE INCO
RRECT."
1690 PRINT "YOU WILL BE BLAS
TED.": : : :
1700 PRINT "PRESS TO CONTINU
E."
1710 CALL KEY(0,KEY,ST)
1720 IF KEY<>13 THEN 1710
1730 RETURN
1740 CALL CLEAR
1750 CALL SCREEN(12)
1760 CALL COLOR(1,2,12)
1770 FOR I=2 TO 8
1780 CALL COLOR(I,1,12)
1790 NEXT I
1800 PRINT : : : :"CHOOSE ON
E": :
1810 PRINT :" 1 HOME KEYS"
1820 PRINT :" 2 HOME ROW"
1830 PRINT :" 3 TOP ROW,MIDD
LE FINGER"
1840 PRINT :" 4 TOP ROW,POIN
TER FINGER"
1850 PRINT :" 5 RING FINGER"
1860 PRINT :" 6 LITTLE FINGE
R"
1870 PRINT :" 7 BOTTOM ROW"
```

continued on page 17

```
100 REM *****************
110 REM * BATTLE AT SEA *
120 REM *****************
130 REM 99'ER VERSION 7.81.1

140 REM BT W.K. BALTHROP
150 REM
160 REM
170 REM
180 RANDOMIZE
190 CALL SCREEN(12)
200 CALL CLEAR
210 PRINT TAB(6);"BATTLE AT
SEA"
220 PRINT TAB(12);"BY"
230 PRINT TAB(7);"W.K. BALTH
ROP"
240 PRINT : : : : : : : : :
: :
250 OPTION BASE 1
260 DIM P(10,10),O(10,10),SH
(5,5,2)
270 CALL COLOR(14,7,1)
280 CALL COLOR(14,11,1)
290 CALL CHAR(96,"000000FF7F
3F1F")
300 CALL CHAR(97,"000000FFFF
FFFF")
310 CALL CHAR(98,"3C7EFFFFFF
FFFF")
320 CALL CHAR(99,"000000FFFE
FCF8")
330 CALL CHAR(100,"103070707
0707070")
340 CALL CHAR(101,"707070707
0707070")
350 CALL CHAR(102,"787C7E7E7
F7F7C78")
360 CALL CHAR(103,"707070707
0701010")
370 CALL CHAR(104,"00080403F
F7F3F")
380 CALL CHAR(105,"8C4C3CFEF
FFFFF")
390 CALL CHAR(106,"01023C3FF
FFFFF")
400 CALL CHAR(107,"000204F8F
FFEFE")
410 CALL CHAR(108,"103072747
8787878")
420 CALL CHAR(109,"7C7C70717
A7C7C7C")
430 CALL CHAR(110,"7F7F787C7
C7C7A79")
440 CALL CHAR(111,"7078787C7
C727110")
450 CALL CHAR(112,"00108867F
F7F3F")
460 CALL CHAR(113,"09C5C3F3F
FFFFF")
470 CALL CHAR(114,"000204F8F
FFEFE")
480 CALL CHAR(115,"103072747
8797A7C")
490 CALL CHAR(116,"797A7C7C7
F7F787C")
500 CALL CHAR(117,"7C7C7A797
87C7C1A")
510 CALL CHAR(118,"0000003FF
F7F3F")
520 CALL CHAR(119,"067EFEFFF
FFFFF")
530 CALL CHAR(120,"000000E0F
CFFFE")
540 CALL CHAR(121,"103078787
8787878")
550 CALL CHAR(122,"787C7C7E7
E7E7F7F")
560 CALL CHAR(123,"7C7C78787
8703020")
570 CALL CHAR(124,"03030F1FF
F7F3F")
580 CALL CHAR(125,"006060F0F
FFEFE")
590 CALL CHAR(126,"103070707
8787E7F")
600 CALL CHAR(127,"7C7878707
0707010")
610 CALL CHAR(128,"FF8181818
18181FF")
620 CALL CHAR(136,"815E2C366
A3C2442")
640 CALL SOUND(-3000,220,30,
554,20,1047,20,-8,30)
650 PRINT "        BATTLE AT
 SEA"
660 PRINT :"YOU MUST DESTROY
 THE ENEMY"
670 PRINT "SHIPS BEFORE THE
COMPUTER"
680 PRINT "DESTROYS YOUR SHI
PS."
690 PRINT :"TO SET UP YOUR S
HIPS YOU"
700 PRINT "MUST ENTER COORDI
NATES ON"
710 PRINT "THE 10 X 10 GRID
ON THE     RIGHT."
720 PRINT "ENTER THE ROW, TH
EN THE     COLUMN."
730 PRINT "EXAMPLE: A5"
740 PRINT "AFTER YOUR SHIPS
ARA SET UP"
750 PRINT "YOU WILL TAKE A S
HOT AT THE"
760 PRINT "ENEMY SHIPS BY EN
TERING ONE"
770 PRINT "PAIR OF COORDINAT
ES ON THE"
780 PRINT "ENEMY FIELD."
790 PRINT "THE COMPUTER WILL
 THEN"
800 PRINT "TAKE A SHOT AT YO
UR SHIPS.": :
810 PRINT "THE COMPUTER CANN
OT SEE"
820 PRINT "YOUR SHIPS.YOU CA
NNOT SEE"
830 PRINT "THE COMPUTER'S SH
IPS."
840 PRINT "ENTER ANY KEY TO
BEGIN."
850 CALL SOUND(1,-2,30)
860 CALL KEY(0,K,S)
870 IF S=0 THEN 860
880 CALL SCREEN(6)
890 CALL CLEAR
900 PRINT "   COMPUTER
 YOU"
910 PRINT : : : : : : : : :
: : : : : : : : : :
920 FOR X=5 TO 14
930 CALL VCHAR(X,5,X+60)
940 CALL HCHAR(X,6,128,10)
950 CALL HCHAR(X,18,128,10)
960 CALL VCHAR(X,17,X+60)
970 NEXT X
980 FOR X=6 TO 15
990 CALL VCHAR(15,X+12,X+42)
1000 CALL VCHAR(15,X,X+42)
1010 NEXT X
1020 S1$="CARRIER"
1030 S2$="BATTLESHIP"
1040 S3$="CRUISER"
1050 S4$="SUBMARINE"
1060 S5$="DESTROYER"
1070 FOR S=1 TO 5
1080 ON S GOSUB 1110,1160,12
10,1260,1310
1090 GOSUB 1390
1100 GOTO 1360
1110 PR$=S1$
1120 LE=5
1130 S=1
1140 OS=0
1150 RETURN
1160 PR$=S2$
1170 LE=4
1180 S=2
1190 OS=8
1200 RETURN
1210 PR$=S3$
1220 LE=3
1230 S=3
1240 OS=16
1250 RETURN
1260 PR$=S4$
1270 LE=3
1280 S=4
1290 OS=22
1300 RETURN
1310 PR$=S5$
1320 LE=2
1330 S=5
1340 OS=28
1350 RETURN
1360 NEXT S
1370 CALL HCHAR(22,1,32,64)
1380 GOTO 2230
1390 L=LEN(PR$)
1400 SUS$="ENTER ROW,COL. FO
R "&STR$(LE)&" SPACES"
1410 FOR X=1 TO LEN(SUS$)
1420 SU1$=SEG$(SUS$,X,1)
1430 CALL VCHAR(22,X+2,ASC(S
U1$))
1440 NEXT X
1450 PR$=PR$&" SPACE"
1460 CALL HCHAR(23,2,32,30)
1470 FOR X=1 TO LEN(PR$)
1480 SU1$=SEG$(PR$,X,1)
1490 CALL VCHAR(23,X+2,ASC(S
U1$))
1500 NEXT X
1510 FOR X=1 TO LE
1520 CALL HCHAR(23,20,35)
1530 CALL VCHAR(23,21,LE-X+4
9)
1540 CALL KEY(0,K1,ST)
1550 IF ST=0 THEN 1540
1560 IF K1<65 THEN 1590
1570 IF K1>74 THEN 1590
1580 GOTO 1610
1590 CALL SOUND(100,-2,2)
1600 GOTO 1540
1610 CALL VCHAR(23,23,K1)
1620 CALL KEY(0,KE,ST)
1630 IF ST=-1 THEN 1620
1640 CALL KEY(0,K2,ST)
1650 IF ST=0 THEN 1640
1660 IF K2<48 THEN 1690
1670 IF K2>57 THEN 1690
1680 GOTO 1710
1690 CALL SOUND(100,-2,2)
1700 GOTO 1640
1710 CALL VCHAR(23,24,K2)
1720 SH(S,X,1)=K1-64
1730 SH(S,X,2)=K2-47
1735 IF P(K1-64,K2-47)>0 THE
N 1590
1740 P(K1-64,K2-47)=S
1750 NEXT X
1760 GOSUB 5350
1770 IF SH(S,1,1)=SH(S,2,1)T
HEN 1800
1780 X2=1
1790 GOTO 1810
1800 X2=2
1810 FOR X3=1 TO LE
1820 F=0
1830 FOR X1=1 TO LE-X3
1840 IF SH(S,X1,X2)=0 THEN 1
900
1850 IF SH(S,X1,X2)<SH(S,X1+
1,X2)THEN 1900
1860 SW=SH(S,X1,X2)
1870 SH(S,X1,X2)=SH(S,X1+1,X
2)
1880 SH(S,X1+1,X2)=SW
1890 F=1
1900 NEXT X1
1910 IF F=0 THEN 1930
1920 NEXT X3
1930 FOR X=1 TO LE-1
```

```
1940 IF SH(S,X,1)<>SH(S,X+1,
1)-1 THEN 1970
1950 NEXT X
1960 GOTO 2060
1970 FOR X=1 TO LE-1
1980 IF SH(S,X,2)<>SH(S,X+1,
2)-1 THEN 2010
1990 NEXT X
2000 GOTO 2060
2010 CALL SOUND(100,-2,2)
2020 FOR X=1 TO LE
2030 P(SH(S,X,1),SH(S,X,2))=
0
2040 NEXT X
2050 GOTO 1460
2060 X=S
2070 FOR X1=1 TO 5
2080 IF SH(X,X1,1)=0 THEN 21
80
2090 IF SH(X,1,1)=SH(X,2,1)T
HEN 2120
2100 OSA=1
2110 GOTO 2130
2120 OSA=0
2130 P(SH(X,X1,1),SH(X,X1,2)
)=X
2140 IF X>1 THEN 2170
2150 CALL VCHAR(SH(X,X1,1)+4
,SH(X,X1,2)+17,95+X1+OS+((LE
-1)*OSA))
2160 GOTO 2180
2170 CALL HCHAR(SH(X,X1,1)+4
,SH(X,X1,2)+17,95+X1+OS+(LE*
OSA))
2180 NEXT X1
2190 IF X>1 THEN 2220
2200 CALL HCHAR(SH(1,4,1)+4,
SH(1,4,2)+17,97+((LE-1)*OSA)
)
2210 CALL HCHAR(SH(1,5,1)+4,
SH(1,5,2)+17,99+((LE-1)*OSA)
)
2220 RETURN
2230 LE=4
2240 S=1
2250 GOSUB 2390
2260 LE=3
2270 S=2
2280 GOSUB 2390
2290 LE=2
2300 S=3
2310 GOSUB 2390
2320 LE=2
2330 S=4
2340 GOSUB 2390
2350 LE=1
2360 S=5
2370 GOSUB 2390
2380 GOTO 2610
2390 X2=INT(RND*2)+1
2400 IF X2=2 THEN 2440
2410 X=INT(RND*(10-LE))+1
2420 X1=INT(RND*10)+1
2430 GOTO 2460
2440 X=INT(RND*10)+1
2450 X1=INT(RND*(10-LE))+1
2460 ON X2 GOTO 2470,2540
2470 FOR Y=X TO X+LE
2480 IF O(Y,X1)>0 THEN 2390
2490 NEXT Y
2500 FOR Y=X TO X+LE
2510 O(Y,X1)=S
2520 NEXT Y
2530 RETURN
2540 FOR Y=X1 TO X1+LE
2550 IF O(X,Y)>0 THEN 2390
2560 NEXT Y
2570 FOR Y=X1 TO X1+LE
2580 O(X,Y)=S
2590 NEXT Y
2600 RETURN
2610 M1$="MY SHOT"
2620 M2$="YOUR SHOT"
2630 M3$="SCORE"
2640 M4$="COMPUTER"
2650 M5$="USER"
2660 M6$="YOU MISSED"
2670 M7$="I MISSED"
2680 M8$="**HIT**"
2690 GOTO 2780
2700 FOR V=1 TO 7
2710 CALL HCHAR(18,V+4,ASC(S
EG$(M1$,V,1)))
2720 NEXT V
2730 RETURN
2740 FOR V=1 TO 9
2750 CALL HCHAR(21,V+4,ASC(S
EG$(M2$,V,1)))
2760 NEXT V
2770 RETURN
2780 FOR X=1 TO 5
2790 CALL HCHAR(18,X+22,ASC(
SEG$(M3$,X,1)))
2800 NEXT X
2810 FOR X=1 TO 8
2820 CALL HCHAR(19,X+15,ASC(
SEG$(M4$,X,1)))
2830 NEXT X
2840 FOR X=1 TO 4
2850 CALL HCHAR(19,X+26,ASC(
SEG$(M5$,X,1)))
2860 NEXT X
2870 T=1
2880 IF T=0 THEN 2910
2890 T=0
2900 GOTO 3180
2910 T=1
2920 CALL HCHAR(21,3,32,12)
2930 CALL HCHAR(22,3,32,7)
2940 GOSUB 2700
2950 IF W>0 THEN 3630
2960 RANDOMIZE
2970 X=INT(10*RND)+1
2980 X1=INT(10*RND)+1
2990 H=X
3000 H1=X1
3010 IF P(X,X1)=7 THEN 2960
3020 IF P(X,X1)=6 THEN 2960
3030 CALL HCHAR(19,6,H+64)
3040 CALL HCHAR(19,7,H1+47)
3050 IF P(X,X1)>0 THEN 4460
3060 GOSUB 3100
3070 GOTO 2880
3080 P(X+10,X1)=7
3090 CALL HCHAR(23,1,32,32)
3100 P(X,X1)=6
3110 CALL SOUND(200,-6,2)
3120 CALL HCHAR(23,1,32,32)
3130 CALL VCHAR(X+4,X1+17,14
4)
3140 FOR Y=1 TO 8
3150 CALL VCHAR(23,12+Y,ASC(
SEG$(M7$,Y,1)))
3160 NEXT Y
3170 RETURN
3180 CALL HCHAR(18,3,32,12)
3190 CALL HCHAR(19,3,32,7)
3200 GOSUB 2740
3210 CALL KEY(0,K1,ST)
3220 IF ST=0 THEN 3210
3230 IF K1<65 THEN 3210
3240 IF K1>74 THEN 3210
3250 CALL VCHAR(22,6,K1)
3260 CALL KEY(0,KE,ST)
3270 IF ST=-1 THEN 3260
3280 CALL KEY(0,K2,ST)
3290 IF ST=0 THEN 3280
3300 IF K2<48 THEN 3280
3310 IF K2>57 THEN 3280
3320 CALL VCHAR(22,7,K2)
3330 K3=K1-64
3340 K4=K2-47
3350 IF O(K3,K4)<6 THEN 3390
3360 CALL SOUND(50,110,2)
3370 CALL HCHAR(22,6,32,7)
3380 GOTO 3180
3390 IF O(K3,K4)=0 THEN 3500
3400 CALL SOUND(200,220,2,33
0,2,440,2,-6,2)
3410 CALL SOUND(400,110,2,22
0,2,330,2,-8,2)
3420 CALL VCHAR(K3+4,K4+5,13
6)
3430 SF=O(K3,K4)
3440 O(K3,K4)=7
3450 CALL HCHAR(23,1,32,32)
3460 FOR X2=1 TO 7
3470 CALL HCHAR(23,13+X2,ASC
(SEG$(M8$,X2,1)))
3480 NEXT X2
3490 GOTO 4600
3500 CALL SOUND(200,-6,2)
3510 CALL HCHAR(23,1,32,32)
3520 O(K3,K4)=6
3530 FOR X2=1 TO 10
3540 CALL VCHAR(23,13+X2,ASC
(SEG$(M6$,X2,1)))
3550 NEXT X2
3560 CALL VCHAR(K3+4,K4+5,14
4)
3570 GOTO 2880
3580 CH=1
3590 GOTO 4650
3600 CH=0
3610 ON SF GOSUB 1110,1160,1
210,1260,1310
3620 IF DS(SF)=LE-1 THEN 378
0
3630 IF H=10 THEN 3670
3640 IF P(H+1,H1)<>7 THEN 36
60
3650 IF W>1 THEN 4260 ELSE 4
060
3660 IF H=1 THEN 3720
3670 IF P(H-1,H1)<>7 THEN 37
20
3680 IF W>1 THEN 4260 ELSE 4
060
3690 W2=W
3700 W=W1
3710 GOTO 3560
3720 IF H1=10 THEN 3760
3730 IF P(H,H1+1)<>7 THEN 37
50
3740 IF W>1 THEN 4060 ELSE 4
260
3750 IF H1=1 THEN 3780
3760 IF P(H,H1-1)<>7 THEN 37
80
3770 IF W>1 THEN 4060 ELSE 4
260
3780 L1=INT(RND*2)+1
3790 ON L1 GOTO 3800,3880
3800 X2=INT(RND*2)+1
3810 ON X2 GOTO 3820,3850
3820 X2=1
3830 X3=0
3840 GOTO 3950
3850 X2=-1
3860 X3=0
3870 GOTO 3950
3880 X3=INT(RND*2)+1
3890 ON X3 GOTO 3900,3930
3900 X3=1
3910 X2=0
3920 GOTO 3950
3930 X3=-1
3940 X2=0
3950 IF H+X2>10 THEN 3780
3960 IF H+X2<1 THEN 3780
3970 IF H1+X3>10 THEN 3780
3980 IF H1+X3<1 THEN 3780
3990 IF P(H+X2,H1+X3)=6 THEN
 3780
4000 IF P(H+X2,H1+X3)=7 THEN
 3780
4010 X=H+X2
4020 X1=H1+X3
4030 IF P(X,X1)>0 THEN 4460
4040 GOSUB 3100
4050 GOTO 2880
4060 IF H=10 THEN 4160
4070 H=H+1
4080 IF P(H,H1)=7 THEN 4060
```

```
4090 IF P(H,H1)=6 THEN 4160
4100 X=H
4110 X1=H1
4120 IF P(X,X1)>0 THEN 4460
4130 GOSUB 3100
4140 H=H-1
4150 GOTO 2880
4160 IF H=1 THEN 4070
4170 H=H-1
4180 IF P(H,H1)=7 THEN 4160
4190 IF P(H,H1)=6 THEN 4060
4200 X=H
4210 X1=H1
4220 IF P(X,X1)>0 THEN 4460
4230 GOSUB 3100
4240 H=H+1
4250 GOTO 2880
4260 IF H1=10 THEN 4360
4270 H1=H1+1
4280 IF P(H,H1)=7 THEN 4260
4290 IF P(H,H1)=6 THEN 4360
4300 X=H
4310 X1=H1
4320 IF P(X,X1)>0 THEN 4460
4330 GOSUB 3100
4340 H1=H1-1
4350 GOTO 2880
4360 IF H1=1 THEN 4260
4370 H1=H1-1
4380 IF P(H,H1)=7 THEN 4360
4390 IF P(H,H1)=6 THEN 4260
4400 X=H
4410 X1=H1
4420 IF P(X,X1)>0 THEN 4460
4430 GOSUB 3100
4440 H1=H1+1
4450 GOTO 2880
4460 CALL VCHAR(4+X,17+X1,13
6)
4470 CALL HCHAR(23,1,32,32)
4480 GOSUB 2700
4490 FOR Z=1 TO LEN(M8$)
4500 CALL HCHAR(23,14+Z,ASC(
SEG$(M8$,Z,1)))
4510 NEXT Z
4520 CALL SOUND(200,220,2,33
0,2,440,2,-8,2)
4530 CALL SOUND(300,110,0,22
0,0,330,0,-8,0)
4540 SF=P(X,X1)
4550 CALL VCHAR(19,6,X+64)
4560 CALL VCHAR(19,7,X1+47)
4570 P(X,X1)=7
4580 H=X
4590 H1=X1
4600 FOR X2=1 TO 5
4610 DS(X2)=0
4620 NEXT X2
4630 FOR X2=1 TO 10
4640 FOR X3=1 TO 10
4650 IF CH=1 THEN 4670
4660 IF T=0 THEN 4720
4670 IF P(X2,X3)=0 THEN 4760
4680 IF P(X2,X3)=6 THEN 4760
4690 IF P(X2,X3)=7 THEN 4760
4700 DS(P(X2,X3))=DS(P(X2,X3
))+1
4710 GOTO 4760
4720 IF O(X2,X3)=0 THEN 4760
4730 IF O(X2,X3)=6 THEN 4760
4740 IF O(X2,X3)=7 THEN 4760
4750 DS(O(X2,X3))=DS(O(X2,X3
))+1
4760 NEXT X3
4770 NEXT X2
4780 IF CH=1 THEN 3600
4790 W=0
4800 SCORE=0
4810 FOR Z4=1 TO 5
4820 ON Z4 GOSUB 1110,1160,1
210,1260,1310
4830 IF DS(Z4)=LE THEN 4920
4840 IF DS(Z4)=0 THEN 4870
4850 W=W+1
4860 GOTO 4920
4870 SCORE=SCORE+1
4880 IF T=0 THEN 4910
4890 GOSUB 4990
4900 GOTO 4920
4910 GOSUB 4980
4920 NEXT Z4
4930 IF T=0 THEN 4960
4940 W1=W
4950 GOTO 2880
4960 W=W1
4970 GOTO 2880
4980 SCP=SCORE
4990 CALL HCHAR(23,1,32,32)
5000 FOR X3=1 TO LEN(PR$)+10
5010 CALL HCHAR(23,X3+6,ASC(
SEG$(PR$&" DESTROYED",X3,1))
)
5020 NEXT X3
5030 IF T=0 THEN 5070
5040 CALL VCHAR(20,20,SCORE+
48)
5050 IF SCORE=5 THEN 5120
5060 RETURN
5070 CALL HCHAR(20,27,SCORE+
48)
5080 IF SCORE=5 THEN 5120
5090 RETURN
5100 PRINT "THE COMPUTER WIN
S AGAIN"
5110 GOTO 5130
5120 PRINT "YOU JUST GOT LUC
KY THIS TIME"
5130 PRINT "IF YOU WISH TO P
LAY AGAIN"
5140 PRINT "ENTER ""Y"",IF N
OT ENTER ""N"""
5150 INPUT NG$
5160 IF NG$="N" THEN 5330
5170 IF NG$="Y" THEN 5200
5180 CALL SOUND(200,110,0)
5190 GOTO 5130
5200 FOR L=1 TO 10
5210 FOR L1=1 TO 10
5220 P(L,L1)=0
5230 O(L,L1)=0
5240 NEXT L1
5250 NEXT L
5260 FOR L=1 TO 5
5270 FOR L1=L TO 5
5280 SH(L,L1,1)=0
5290 SH(L,L1,2)=0
5300 NEXT L1
5310 NEXT L
5320 GOTO 880
5330 CALL CLEAR
5340 STOP
5350 NNN=0
5360 AAA=0
5370 FOR X=1 TO LE-1
5380 IF NNN=1 THEN 5410
5390 IF SH(S,X,1)=SH(S,X+1,1
)THEN 5440
5400 IF AAA=1 THEN 2010
5410 IF SH(S,X,2)<>SH(S,X+1,
2)THEN 2010
5420 NNN=1
5430 GOTO 5450
5440 AAA=1
5450 NEXT X
5460 RETURN
5470 END
```

```
10000 !  PROGRAM PEEKER
10010 !  BY G. MINEO
10020 !*********************
10030 !THIS ROUTINE IS SAVED
       WITH THE MERGE OPTION
       IT SHOULD BE MERGED
       WITH PROGRAM TO BE
       ANALYZED.
10040 !ONCE THE PROGRAMS ARE
       MERGED,LIST THE LINES
       YOU WANT TO CHECK.
       TYPE RUN 10000 AND
       ANSWER QUESTIONS.
10050 !THE BOTTOM LINE OF
       THE DISPLAY WILL IN-
       DICATE THE BYTE,INST.
       AND ASCII VALUE OF
       THE INST.
10060 !AN '*' IN FRONT OF AN
       INST MEANS CONDENSED
       FORMAT INSTRUCTION.
10070 !TOUCH ANY KEY EXCEPT
       S TO GO FORWARD.TOUCH
       S TO GO BACKWARDS.
10080 !THIS PROGRAM IS HELP-
       FUL WHEN USING THE
       LOAD STATEMENT.
10090 CALL CHAR(100,"FF")::
CALL HCHAR(23,1,100,32)
10100 DISPLAY AT(24,1):"STAR
T BYTE>-1   (FIRST=-1)"
10110 ACCEPT AT(24,12)BEEP S
IZE(-4):SBYTE
10120 DISPLAY AT(24,1):"HIGH
 BYTE> -20000"
10130 ACCEPT AT(24,12)BEEP S
IZE(-6):HBYTE
10140 IMAGE BYTE###### INST>
#### ASCII #
10150 FOR COUNT=SBYTE TO HBY
TE STEP -1
10160 CALL PEEK(COUNT,INST)
10170 DISPLAY AT(24,1):USING
 10140:COUNT,INST,CHR$(INST)
10180 IF INST>128 THEN CALL
HCHAR(24,19,42)
10190 CALL KEY(0,K,S):: IF S
=0 THEN 10190
10200 IF K=83 THEN COUNT=COU
NT+1 :: GOTO 10160
10210 NEXT COUNT
```

continued from page 14

```
1880 PRINT :" 8 END PROGRAM"
: : :
1890 CALL SCREEN(5)
1900 FOR I=2 TO 8
1910 CALL COLOR(I,2,12)
1920 NEXT I
1930 RETURN
1940 CALL COLOR(9,1,1)
1950 CALL COLOR(10,1,1)
1960 CALL VCHAR(12,Y,98,13)
1970 CALL VCHAR(13,Y-1,98,12
)
1980 CALL VCHAR(13,Y+1,98,12
)
1990 CALL VCHAR(14,Y-2,98,11
)
2000 CALL VCHAR(14,Y+2,98,11
)
2010 CALL VCHAR(13,Y-2,101)
2020 CALL VCHAR(13,Y+2,97)
2030 CALL VCHAR(12,Y-1,101)
2040 CALL VCHAR(12,Y+1,97)
2050 CALL VCHAR(11,Y,102)
2060 CALL VCHAR(22,Y-3,98,3)
2070 CALL VCHAR(22,Y+3,98,3)
2080 CALL VCHAR(23,Y-4,98,2)
2090 CALL VCHAR(23,Y+4,98,2)
2100 CALL VCHAR(22,Y-4,101)
2110 CALL VCHAR(22,Y+4,97)
2120 CALL VCHAR(21,Y-3,101)
2130 CALL VCHAR(21,Y+3,97)
2140 CALL COLOR(9,7,1)
2150 CALL COLOR(10,16,1)
2160 RETURN
2170 END
2180 2180
```

continued from page 13

If you do not have speech, the cars make a noise, but if you have speech you will hear phrases like "FIND THE BOMB", "O.K. BOSS" and other well pronounced words.

Overall this is an excellent game utilizing sound, speech and graphics to their full potential.

Well that bumper issue has tired us out, I dont know if we will gain strength for next month. Remember if you have any difficulties with the above file or anything related to games please refer correspondance to GAMES on the BBS or to the following address :

C/- GAMES INFORMATION
141 BEECROFT ROAD,
BEECROFT N.S.W. 2119

°

Computer War

by Robert Brown and Stephen Judd

The Display

1. The score of the current game is displayed in the top left of the right window. The high score of the current session is displayed in the top right of the screen.
2. The map of the United States shows the location of all the cities targeted by enemy missiles and the NORAD headquarters. Your task is to defend these cities.
3. The left of the screen shows a matrix and below this is the current DEFCON (defence condition). 5 is the start of the game. 1 is game over with all cities and NORAD destroyed.

How to play

1. Press the fire button on the joystick to start the game. Do not touch the fire button again until it is absolutely necessary.
2. From the edges of the screen, enemy missiles will appear, advancing towards selected targets. Move the cross on the screen (with the joystick), until it is immediately over the advancing missile. At this point press the fire button.
3. The display will change to one of two. If the missile was in the centre of the cross, then the display will become the view that a fighter pilot would see through his cockpit window. Beneath the screen are three boxes, depicting current DEFCON status (on the left), the time until impact of enemy missile (on right) and in the centre is a box depicting where the enemy missile is in relationship to the view from the cockpit. To chase the missile, move the joystick towards the missile's location. To move left or right, push the joystick in the appropriate direction. To move up, pull the joystick back, and to dive, push the joystick forward. When the missile is in view, press the fire button. Ammunition is unlimited, so keep firing until the missile is destroyed. When time until impact indicator reaches 10, an alarm will sound.
4. If you do not destroy the enemy missile in the allotted time, you will lose a city.
5. If the cross was not positioned correctly over a missile the message NO ENEMY MISSILES IN INTERCEPT RANGE will appear. The game will return to the map screen, but you will not be given a second chance to knock out that particular missile.
6. After all missiles in a wave have either been destroyed or have hit their targets, the bonus screen is presented. At the middle left of the screen in a 3x3 matrix, is the code you have to match. This is achieved by moving the white 3x3 square around the screen above, until a similar mix is achieved. If an identical mix does not exist, then patterns can be built by spinning the squares by pressing the fire button. When an identical match is found, press the fire button again and the computer will move onto the next square to be matched. If all matches are made, then a bonus of 1000 points is added to your score.
7. The game then starts again, with further attacks of missiles, in increasing number and speed.

The game is over when all cities and NORAD are destroyed. Even if all cities are intact, should NORAD be destroyed, then the game is over. As each city is destroyed, the DEFCON factor is reduced. The game is over if DEFCON reaches 1.

Scoring

150 points per enemy missile shot down.
1000 points for a full code screen cracked.
1000 additional points each time the code screen is cracked.

°

CHOCKY AND TICTACMAN

I am Weak, Got no Control

by Werner Kanitz

"Interested in getting a computer?" says my mate.

No I have an ESR calculator and a snoot full of computers at work.

You ought to get a TI99/4A. I payed $500+ and they are now going for less than $200.

No, but it does sound like a good deal. Why are they so cheap?

They are going to seriously challenge the other mob, then they will go broke in this particular line.

It does sound too good to refuse though, well maybe just for a trial.

Nice Cheap computer

Of course you will want Extended BASIC to make full use of all the graphic capabilities of the machine not to mention speech. No one can play games properly without joy sticks.

I do not even like computer games.

You also need the speech synthesiser for the games to talk to you.

Can you do serious things with this computer.

Get off it! It is a toy. However, if you are prepared to spend:

if you get the cheap home made memory expansion board you can have access to better games.

Take a long time to load do they not?

What you really need are disk drives. There happens to be a PE box on the market with memory expansion card and printer card and heaps of software. The trouble is you have to take the computer as well. Never mind, everyone should have two anyway

Now can we get serious

Are you prepared for the additional cost?

You cannot be serious unless you have TI-Writer and the Editor Assembler. Now that we are serious, we are distressed at having to change cartriges frequently. This can be overcome by putting a number of applications in the one cartrige. This is probably the cheapest exercise of the lot. It was free, almost.

Is it not time we got a printer?

You beaut, we recently acquired a Citizen 120D for $400 (new). It is a real ripper, good value.

unt now ve can brindt peaudifull shtuff like dis here.

Of course you are not really serious unless you have a RAMdisk. OK, OK as soon as I buy some chips I will glue them on, but I definitely do not want a modem yet. I think I am pretty sure about this so please do not tempt me.

I wonder just how much this cheap little computer has set me back.

°

Modules For Sale

Extended BASIC for $35
Terminal Emulator II for $25
Video Chess for $10
Multiplan (complete) for $40
TI-Writer (complete) for $25
Personal Record Keeping for $10
Editor Assembler manual only, in folder for $15

Phone (042)28 4906 evenings, or contact Lou Amadio at the next Sydney meeting.

Myarc Hard/Floppy Disk Controller Card

by Garry J. Christensen, TIBUG

Everyone knows what a floppy disk is and how it can quickly load and save data, not to mention the amount of storage space available. The hard disk takes this a step further.

Myarc have just released a controller card that will give you access to all the advantages of a hard disk. The new controller card is completely compatible with the TI99/4A and includes the floppy disk controller on the one card. You will be supplied with the controller card, the cables to connect to the hard drive and a 68 page manual in a ring binder.

Also with the card is the Myarc Disk Manager 5 that is written particularly for the new card.

What do you need to upgrade to a hard drive? You need a TI99/4A or Geneve computer, and Expansion Box, memory expansion, one or more floppy drives, and of course a hard disk drive.

Features

1. The Hard Floppy Disk Controller Card (HFDCC) will maintain 4 double sided, double density floppy drives. It also will access the 80 track drives that are available today. Set aside for future expansion is a setting for 80 track, double sided, quad density drives. These allow the storage of 1.44Mb on one floppy. Lets hope that this update is developed. The head step rate can be set at 16 or 8 msec for 40 track drives and 2 msec for 80 track drives. The card will control 3, 140Mb hard drives or any compatible size up to that value.
2. The CRU address of the card can be selected to suit your system. You may replace your current disk controller and put the new one at >1100. This will still allow your RAMdisk at >1000 to have precedence if you wish. I do not know if that is necessary as it is stated that the hard disk is 2 to 3 times faster than this. If you have a CorComp controller or one of the new AT controllers from Sydney, you may not wish to lose some of the advantages of your present system. The HFDCC can be placed at any unused address above >1100. Now the HFDCC will function as a Hard Drive Controller only. Any accesses to the floppy drives will go through your first controller.
3. Data transfer rates of 5Mbits/sec for the hard drive and 500Kbits/sec for floppy drives. The data transfer rate of the hard disk means that any programme for the TI99/4A can be loaded in a fraction of a second. The amount of free RAM in the TI99/4A computer (32K) can be filled in 10 msec. Add a short time for finding the data on the disk and for making this data available in the memory and the time is still well under one second. Some of the literature released by Myarc indicates that the hard drive could support full screen animation at a rate of 10 frames per second. The cartoons that you see on television use 8 frames per second.
4. Streaming tape interface allows you to back-up the contents of the hard disk onto a QIC-02 compatible tape drive.
5. Real time clock is used to date your files and is also accessable by external software.
6. Data bus isolation allows data to be loaded from disk into 32K of static RAM on the card while other peripheral cards are in use. When coupled with the DMA controller, data can be read from or written to memory while the computer is performing other functions.
7. Compatibility - The card uses the same format as the standard IBM drives. That is, ST506 compatible. Be careful when choosing a drive. There are two formats that the drives can be. One is MFM and the other RLL. This controller card supports MFM only. In my enquiries I found that the only common size that used RLL was the 30M drives. Most other sizes are OK but check first.
8. The card will load data directly into CPU RAM instead of using VDP RAM. The PAB can also be established in CPU RAM to avoid all use of the VDP. This feature is available to assembly language programmers only.
9. Sub-directories can also be created on floppy drives. This will help if you are using the larger drives.

The Manual

The manual that is supplied with the card covers all that you will need to know about the operation of the card.

It details the system requirements, and the set-up procedure. It covers setting the DIP switches, connecting the floppy and hard drives, how to power the system up and down and the setting of the time and date.

The next section provides directions for the operation of the MDM5.

The manual also discusses the structuring of directories and sub-directories on the hard disk, floppy drive emulation and accessing the hard drive through TI Basic.

Assembly language and technical support is given with a description of the pin-outs to the hard and floppy drives, CRU map of the card, DSR memory map, low and high level software interface specifications and details on the structuring of the data on the sectors on the hard drive.

TI BASIC

There are very few differences between using a floppy drive and using a hard drive. The most obvious is the structuring of the files in the hard drive. You would be aware that the floppy drive establishes a directory of the files that are contained on the disk. Can you imagine the size of a directory when a 40Mb disk is full?

To help make data management easier, hard drives use sub-directories. Consider that the disk can be partitioned into parts, each of these acting like a separate disk. Each partition can also be divided into separate parts, which can be sub-divided again, and so on.

The main directory is called the root directory. It can contain 127 files and 114 sub-directories. Each sub-directory can hold 127 files and 114 sub-sub-directories. This can continue until you run out of disk space.

Some examples may help. To load a file that is stored under the root directory, simple use the same syntax as a floppy drive except DSK becomes WDS.

OLD WDS1.LOAD

Lets consider that you have stored the funnelweb disk under the sub-directory FWB. To load it you would use this command.

OLD WDS1.FWB.LOAD

If FWB also had a sub-directory called LETTERS and your letter was filed under the name FRED it would be accessed by WDS1.FWB.LETTERS.FRED

This system can be continued for any length although many programmes that load files have a maximum length set. In these instances, it is best to keep the names of directories as short as possible.

The path taken through the sub-directories is called the pathname. In general format, any file can be accessed by: 'Devicename.Pathname.Filename'.

The other major changes are that the HFDCC does not require any memory for working space and that there are some small changes in the CALL FILES command.

In addition to the normal BASIC commands, some extras have been added. These are listed below with the E/A equivalent.

CALL ILR - CALL INIT
CALL LR - CALL LOAD
CALL LLR - CALL LINK
CALL DIR - disk directory
CALL DM _ load MDM5

One final point to discuss is that of floppy drive emulation. There are many programmes that expect certain files to be stored in a device called DSK1 or DSK.Diskname.Filename. To cover these programmes the HFDCC will emulate both these conditions.

The first method is through 'DSK1 Directory Emulation'. If a sub-directory called DSK1 is created in the root directory, this area will be checked for the file before looking at the real DSK1. All the

Extended BASIC permanently in the Console

by Ross Mudie

If you are thinking of putting Extended BASIC, or any other command module permanently in your console, it is quite feasible, but it is a complex job and as such, ONLY recommended for those who have an understanding of electronics, good eyesight and a fine tipped soldering iron. The electrical connections are based on the Naverone cartridge expander. After modification, the lock up problems attributed to poor electrical contact in the GROM port should be a thing of the past.

I initially fell for the trap of trying to use the article in the December 1987 MICROpendium but, I found myself confused, especially because the author of the article does not count the module connections the same way that Texas Instruments did!

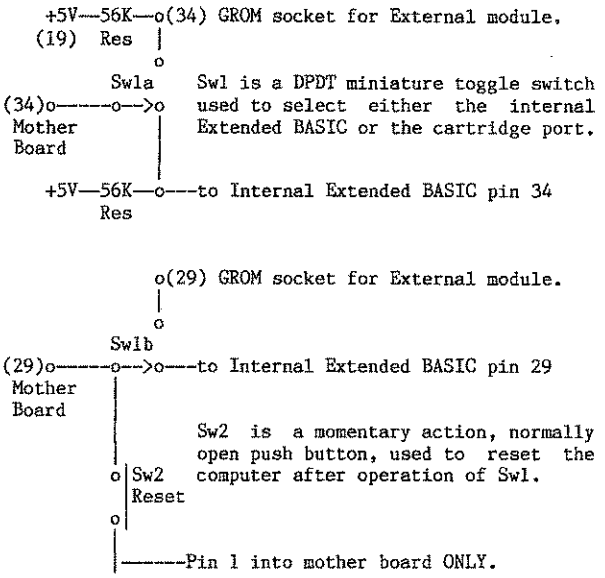
I decided to place the Printed Circuit Board (PCB), removed from the Extended BASIC module, on top of the top shield of the TI99/4A mother board, attached using two layers of double sided foam mounting tape. Two 18 way ribbon cables were used, one for each side of the Extended BASIC edge connector. The ribbon cables are terminated on the back of the GROM port connector, since this allows the Extended BASIC PCB to be simply isolated from the mother board in the event of a fault. Figure 1 shows the GROM port connector from the rear of the computer. The function of each wire is shown in addition to the numbering of each pin.

Eth	ROM-G	* WE	A4	A5	A6	A3	A7	A8	A9	A 10	A 11	A 12	A 13	A 15	CRU In	CRU Ck	Eth
36	34	32	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2
Eth	Vss	GR	-5V	GRC	DBIN	A 14	* GS	+5V	DO	D1	D2	D3	D4	D5	D6	D7	Reset
35	33	31	29	27	25	23	21	19	17	15	13	11	9	7	5	3	1

Figure 1 - Rear View of GROM Port Connector.

A DPDT switch was provided on the side of the console to select either the internal Extended BASIC or the module in the GROM port. A momentary action reset switch was also provided to reset the computer after changing over the module select switch.

To allow the switching it was necessary to cut three tracks on the GROM connector leading to pins 1, 29 and 34. Pin 1 is the reset pin; there must be NO connection between pin 1 of the internal Extended BASIC and the GROM socket. If the resets are connected then the Extended BASIC will malfunction with a command module plugged into the GROM port. One side of the reset switch is connected to pin 1 going into the mother board of the console, only. Pin 29 is the -5V power rail; the DPDT switch must connect -5V to the selected command module or internal PCB. The unswitched -5V also connects to the momentary action, single pole, normally open reset switch. Pin 34, *ROMG, (which is the active low bank select line); the DPDT switch must connect pin 29 from the mother board to the required module. Two additional 56K ohm resistors must be provided from pin 29 of Extended BASIC to +5V and from pin 29 of the GROM port to +5V. The resistor on the Extended BASIC PCB can be conveniently mounted on the PCB, whilst the resistor for the GROM port can be mounted close to the pins on the solder side of the GROM connector, between pins 29 and 19. Figure 2 shows the connections.



Pin 1 to module (GROM) port MUST be open circuit.
Pin 1 into Extended BASIC PCB MUST be open circuit.

Figure 2 - Wiring of Module select and reset switches.

The author or TIsHUG will take no responsibility for any damage that may occur to any TI99/4A as a result of anything contained in this article. This modification will take several hours to perform and should not be attempted unless the person performing it is competent to do so. °

continued from page 19

programmes and files that need to be in the first disk drive should be placed in this field. Care must be taken that there are not two files of the same name (LOAD for example).

The other method is called 'DSK1 File Emulation'. MDM5 will archive a floppy disk into a special file. You may have as many of these files in the hard disk as you wish and they need not be in the root directory. Only one file can be active at any time. Each time a file is required, it is retrieved from the emulate file.

There is a restriction on these methods. For both, the HFDCC must be the first card accessed with a device name of DSK1. That is, if a RAMdisk or other disk controller card has a lower CRU address, the other card will be accessed instead of the hard disk.

The File Emulate method requires that the HFDCC be le. 'DSK
Emulation' allows you to use programmes like Multiplan. In this case, you can create a directory in the root directory called 'DSK'. Then create a sub-directory called the name of the disk and store all the files on the disk in there.

In all these cases, if the file is not found, the controller will then look at the real device.

The HFDCC has been designed to be completely compatible with all the existing software that is available for the TI99/4A computer. It is a peripheral that will vastly enhance the power of your computer.

For further enquiries, contact either Ben Takach of TIsHUG Phone (02)489-4492; Postal Address P.O. Box 114, WAHROONGA NSW 2076 or;

Garry Christensen on Phone (07)284-1841; Postal Address 36 Henzell St, Kippa-Ring, QLD 4020.

Garry will be placing an order with Myarc in August. The following Myarc product prices are quoted as a guide:

Geneve 9640	$750
Hard/Floppy Controller	$390
Floppy controller	$240
RS232/PIO	$135
Myarc Mouse and My-Art	$165
512K RAMdisk	$375
512K Memory Expansion	$370

°

Beginner's Logic, at first

by Ross Mudie

1. INTRODUCTION.

Whilst presenting my tutorial lecture on the Wire Accessory Controller for TI99/4A at the recent TIsHUG Tutorial Day, I realised that many of our members do not have an appreciation of logic, especially as applied to integrated circuits or to many of the conventions used to represent a circuit in a diagram.

In this article I will attempt to introduce the reader to logic elements and some of the conventions used in drawing the circuits. I will use the circuit diagram of the Wire Accessory Interface which was published in the TIsHUG News Digest of July 1988 for specific references and expand on its operation.

Logical functions are not limited to Integrated Circuit (IC) gates, the same techniques can be applied using software in programs and are easily understood in Extended BASIC, e.g.,

IF A=1 AND B=1 THEN C=1 ELSE C=0

This is represented graphically in the truth table of a 2 input AND gate as follows:

INPUTS		OUTPUT
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Table 1 - AND Function.

In the program case I have used two values, 0 and 1, which can be considered as logic 0 and logic 1. When this is applied to semiconductor logic gates the logic 0 and 1 can be defined as voltage levels as you will see later in this article.

2. LOGIC SYSTEMS.

In the TI99/4A positive logic is used, in which logic 1 is a positive voltage with respect to logic 0 which is nearest to system earth. Many of the ICs in the TI99/4A are TTL or TTL compatable. TTL refers to the internal circuitry of the integrated circuits, standing for Transistor Transistor Logic.

Since the development of the original TTL there have been enhancements of Low Power TTL (slower), Shottky TTL (faster) and LS Low Power Shottky (lower power but faster). Then CMOS (Complementary Metal Over Silicon) which was slower but much lower power, followed by High Power CMOS (HC) which consumes less power, especially in the inputs, but is capable of good output drive and speed.

The speed of a logic device is determined by its internal design structure. From the time that a valid input is applied until the output state changes, is known as the propagation delay and as such is the limiting factor in the operating speed of the device.

The power supply requirements of the TTL family of integrated circuits is 4.75 to 5.25 volts DC, however the HC series can operate over the range of 2 to 6 volts. The TI99/4A console operates its TTL from a nominal 5VDC supply.

In TTL logic 0 is a voltage less than 0.8 volts and logic 1 is a voltage greater than 2.0 volts. If an input or output remains within the range 0.8V to 2.0V then unpredictable operation may occur.

3. Logical AND Function.

Consider the AND function as applied to an IC gate package. The AND gate is drawn as shown in Figure 1.

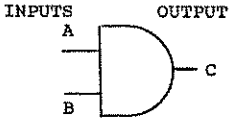


Figure 1 - AND gate.

If input A is logic 1 AND input B is logic 1 THEN output C is logic 1. You have already seen the truth table for the AND gate in Table 1.

4. Logical OR function.

The symbol for the OR gate function is shown in Figure 2 and the truth table is shown in table 2.

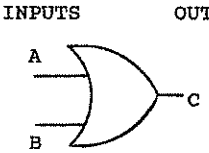


Figure 2 - OR gate.

INPUTS		OUTPUT
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

Table 2 - OR Gate

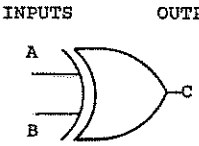
This means that IF input A is logic 1 OR input B is logic 1 THEN output C is logic 1. In an Extended BASIC program the OR function would be:

IF A=1 OR B=1 THEN C=1 ELSE C=0

It should noted that if both input A is logic 1 AND input B is logic 1 THEN output C will still be logic 1 since it is not the EXCLUSIVE OR function.

5. EXCLUSIVE OR (XOR) function.

The symbol for the EXCLUSIVE OR gate function is shown in Figure 3 and the truth table is in table 3.



INPUTS		OUTPUT
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

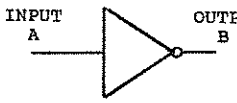
Figure 3 - Exclusive OR gate. Table 3 - EXCLUSIVE OR.

This means that IF input A is logic 1 OR input B is logic 1 THEN output C is logic 1, BUT if A is logic 1 AND B is logic 1 THEN C is logic 0, hence the name "EXCLUSIVE OR". In an Extended BASIC program the XOR function would be:

IF A=1 XOR B=1 THEN C=1 ELSE C=0

6. Logical INVERSION, the NOT function.

An INVERTER simply reverses the logical condition. Logic 0 becomes logic 1 and logic 1 becomes logic 0. Figure 4 shows the symbol for an inverter and table 4 shows the truth table.



INPUT	OUTPUT
1	0
0	1

Figure 4 - Inverter. Table 4 - INVERTER Function.

Inversion when combined into the output of an AND gate makes it into a NAND gate and when applied to an OR gate makes it into a NOR gate. In the symbol the circle on the output indicates that a logic 0 is the significant or active logic state, rather than logic 1.

7. NAND Function.

The NAND function is equivalent to an AND gate followed by an inverter, see Figure 5a. The symbol for a 2 input NAND gate is shown in Figure 5b.

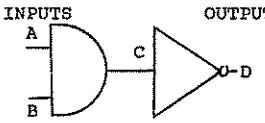


Figure 5a - AND + INVERTER

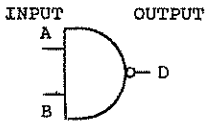


Figure 5b - NAND Gate.

The truth table for the NAND function is shown in table 5. If you compare tables 1 and 5 you will observe that the only difference is that the output of the NAND function is inverted compared to the AND function.

INPUTS		OUTPUT
A	B	D
0	0	1
0	1	1
1	0	1
1	1	0

Table 5 - NAND Function.

The Extended BASIC program equivalent of the NAND gate would be:

IF A=1 AND B=1 THEN D=0 ELSE D=1.

8. NOR Function.

The NOR function is equivalent to an OR gate followed by an inverter, see Figure 6a. The symbol for a 2 input NOR gate is shown in Figure 6b.

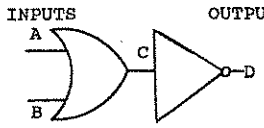


Figure 6a - OR + INVERTER

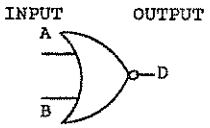


Figure 6b - NOR Gate.

The truth table for the NOR function is shown in table 6. If you compare tables 2 and 6 you will observe that the only difference is that the output of the NOR function is inverted compared to the OR function.

INPUTS		OUTPUT
A	B	D
0	0	1
0	1	0
1	0	0
1	1	0

Table 6 - NOR Function.

The Extended BASIC program equivalent of the NOR gate would be:

IF A=1 OR B=1 THEN D=0 ELSE D=1 .

9. GATES WITH GREATER THAN 2 INPUTS.

Gate packages in AND, OR, NAND and NOR exist with more than 2 inputs to simplify hardware circuit design. Not all logical device types are available in the full range of inputs, however devices are available with 2, 3, 4, 5 or 8 inputs. Gate IC packages are mostly 14 pin ICs and in most cases have as many gates as will fit in the number of package pins. Refer to "Logic IC" data books for greater detail.

10. USING LOGIC '0' SIGNIFICANT.

The discussion so far has been centered on using logic 1 as the "significant" input state, for example,

IF input A=1 AND input B=1 then output D=0.

The logic circuit designer can however use logic 0 for the "significant" input state, especially where economy of IC package count can be achieved.

INPUTS		OUTPUT
A	B	D
0	0	1
0	1	1
1	0	1
1	1	0

The truth table at left is for a NAND gate, but consider the function if logic 0 is the significant state.

IF A=0 OR B=0 THEN D=1 ELSE D=0
IF A=1 AND B=1 THEN D=0 ELSE D=1

Both these statements are true, the first is taking logic 0 as the significant input state, whilst the second takes logic 1 as the significant input state. The NAND gate CAN perform an inverting OR function with logic 0 as the significant input state.

Consider also the NOR gate, which can perform an inverting AND function if the circuit operation gives logic 0 as the significant input state.

11. TRI-STATE OUTPUTS.

When a group of IC devices have their outputs connected to a common "bus" line, if more than one of the devices was electrically connected at a time it would interfere with the signals from the required device. A way around this problem would be to have a switch to isolate all the devices not required and only to allow the required device outputs to be connected onto the bus.

The method used to provide this switching is called the "TRI-STATE OUTPUT" and this allows an output to be logic 0 or logic 1 or "high impedance" which is functionally not connected. As you will see shortly, the input ICs of the Wire Accessory Interface use tri-state outputs so that only one input IC accesses the computer's data bus at a time and then only when called for.

12. PRACTICAL APPLICATIONS.

Examples of these logic principles are contained in the circuit of the "Wire Accessory Interface" on page 6 of the July 1988 TND. Gates are used to decode the computer's address lines to allow selection of the appropiate input or output interface IC. This would be a good time to get out the circuit.

Integrated Circuit (IC) (1A) is a 4 input NAND gate which is used to decode the address lines A0, A5, A6 and A8. When these 4 address lines, which connect to the 4 inputs of IC (1A), are all at logic 1, then the output of (1A) will be logic 0. Putting it in Extended BASIC statement form:

IF A0=1 AND A5=1 AND A6=1 AND A8=1 THEN (OUTPUT IC 1A) =0 ELSE (OUTPUT IC 1A)=1.

The Wire Accessory Interface uses addresses -31104 to -31089 or in hexadecimal >8680 to >868F. Consider the first 3 digits of the address, >868x, as applied to the address lines. The last 4 address lines, A12 to A15) are handled by another part of the hardware design which will be covered later. For those who do not understand the conversion from hexadecimal to negative decimal numbers refer to appendix 1.

The binary address lines are grouped in 4 groups of 4 when referring to an address in hexadecimal notation. Within the group of 4 lines the least significant has a value of 1 and the most significant has a value of 8. The "weighting" of the address lines is shown in Figure 6.

Binary line value	8	4	2	1	8	4	2	1	8	4	2	1	8 ..
Address Line	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12..
>868x	1	0	0	0	0	1	1	0	1	0	0	0	
Value of 4 lines	8				6				8				x

Table 6 - Address Lines A0 to A11.

Each address line can only be logic 0 or logic 1, but when the address as shown in table 6 exists, the condition must be detected to provide an enable signal for the card. As you saw a moment ago, the inputs of IC (1A) are connected to A0, A5, A6 and A8 which happen to be logic 1 significant. Has it occured to you that address lines A1, A2, A3, A4, A7, A9, A10 and A11 are logic 0 significant? Look now at the circuit diagram again and you will find that all these address lines are connected to the inputs of IC (2), which is an 8 input NOR gate. Since logic 0 (by circuit design) is significant on the inputs of the NOR gate, it performs an inverting AND function and the significant output will be a logic 1. IC (4A) is also a NOR gate performing an AND function with logic 0 significant on its inputs. Only when all the inputs of IC (2) are at logic 0 will the output of IC (2) be logic 1. The significant output of IC (2) is logic 1 thus an inverter, part of IC (26), corrects the logic state into IC (4A). In summary of this part, when any address between >8680 and >868F is present on the address bus, the logic 1 on each input of IC(1A) and the logic 0 on each input of IC (2) will result in a logic 0 on the output, (pin 3), of IC (4A). With any other address on the address bus outside the range of >8680 to >868F there will not be ALL logic 1s on the inputs of IC (1A) and/or there will not be ALL logic 0s on the inputs of IC (2) and then the output of IC (4A) will be logic 1.

If the NOR gate had not been used to perform the "AND" function with logic 0 significant, then an additional 8 inverters would be required to provide a logic 1 to each input of an 8 input NAND gate when the appropiate address was present on the address bus. Since inverters come 6 to a package and there are 3 spares in the PCB then an additional IC would have been required.

13. UNDERSTANDING SOME OF THE DRAWING CONVENTIONS.

There are a number of different drawing symbol standards. In this article I have used the same standards as the designers of the TI99/4A to help you to understand the circuit diagram of the computer.

When looking at the circuit diagram of the Wire Accessory Interface, you will have noticed some little pulse symbols adjacent to some of the IC inputs and outputs. Take note of the "pulse" just right of pin 3 of IC (4A), it is representative of the fact that this point is normally high (logic 1) and goes low for the period that the appropiate address is present on the address bus of the computer. These symbols will help you to understand the operation of the circuit by showing the conditions that matter, ie, significant.

The Write Enable line is shown on the diagram with a bar over the top of it, where in text, and sometimes on the drawings its is shown as *WE or WE*. This usually means that it is active, or significant, when it is logic 0. (Note the little pulse symbol near the WE* line just above IC (4B) pin 5, normally high, active low). The *WE line may also be referred to as NOT Write Enable since it is NOT logic 1 when it is active. (No thats not Double-Dutch, its logical!)

A drawing convention used when there are a number of wires all going in the same direction is to combine a group of wires into one "wire" with a symbol at the point of entry into the common "wire" and a matching symbol at the point of exit from the common "wire".

When wires join there is a connection dot at the point of intersection of the wires where unjoined wires just cross without an obvious dot at the point of intersection.

The group of NOR gates used as final address decoding for the outputs also uses an abbreviated drawing technique. The diagram shows 22A, 22B --- 24D whilst the parts list in the article shows that there is a quantity of 3 ICs type 74LS02, being IC numbers 22-24, ie, 22, 23 and 24. The connections for the gates not drawn are similar to the gates which are drawn, except the wires from the outputs of IC (3), in which case the actual connection is spelt out by an arrow on the end of the wire and the destination. Refer to pin 3 of IC (3), it goes to 22/8, ie, IC (22), pin 8.

The outputs of ICs 22-24 each go separately to pin 11 of the respective output buffer IC, one gate output to one Latch Enable (LE) input of the appropriate Octal Latch IC, type 74LS373.

Where ICs are drawn one behind the other it shows that there is some commonality in their connections. Take ICs 6-17, pin 3 of each of these ICs is connected together from pin 18 of IC (25) and also to pin 2 of ICs 18-21. The outputs of IC (6) are however totally separate from the outputs of IC (7) and so on since this is how the 96 outputs are achieved, (12 ICs each with 8 separate output wires). Similarly the "D" inputs of ICs 18 to 21 are each separate wires where the "Q" outputs on the "bus" side are connected from IC to IC.

In some of the more complex ICs you will notice that there are designations within the block of the IC drawing and numbers outside. The numbers on the outside refer to the device package pin numbers, whilst the information on the inside gives an indication of the device function. Look at the ICs numbered 6-17, they take their inputs from the buffered data bus into the Data (D) inputs of memory elements and the outputs are provided via the Q outputs. If logic 1 is present on D1 when the Latch Enable (LE) is active (a pulse of logic 1 according the the symbol on the circuit) then the output Q1 will become logic 1 also. When the LE input goes back to logic 0, the condition present on the D input is stored on the Q output and must remain that way until the LE again becomes active. If D1 was logic 0 when LE was active then Q1 will become logic 0 and this state will be stored until LE is again active. You will notice on ICs 6-17 that pin 1, Control (CTL) is connected to 0V which is logic 0, however on the input ICs, 18-21, the CTL is controlled from the final address decoding gates (IC 5A-5D). The CTL input controls the tri-state output feature of the 74LS373. Tri-state is never required on the output ICs, IC 6-17, thus CTL (pin 1) is permanently active at 0 volts. On the input ICs (IC 18-21) the latch enable is permanently active at logic 1 (+5V) and the devices are tri-state switched on to the buffered data bus only when called for by the address on the address bus and the Data Bus IN (DBIN) signal from the processor.

The bi-directional buffer (IC25) is used to boost the signal on the data bus. If a full complement of ICs were installed on the Wire Accessory Interface there may be excessive loading on the data bus. (Especially if the design is extended beyond 128 Inputs/Outputs). The buffer provides the necessary drive without excessive loading of the data bus when in the A to B direction and provides a tri-state high impedance output when it is not enabled.

IC (3) is a 4 to 16 line decoder. It accepts the binary input on inputs A, B, C and D and gives an active logic 0 output on only one output at a time. This is how the value on A12, A13, A14 and A15, which make up the last hexadecimal digit of the address, provide the final choice of the required input or output IC (with a little help from the gates ICs 22-24 or 5). Inside the outline of IC (3) the designation 0, 1 thru to F indicate the output which will be low (logic 0) when the appropiate input condition is present. The numbers -31104 to -31089 in brackets show the decimal address appropiate for use in CALL LOAD or CALL PEEK from Extended Basic, E/A basic or Mini-Mem basic.

14. EXPANDING BEYOND 128 I/O.

The design could be simply increased from 128 Inputs and Outputs in total to 128 Inputs and 128 Outputs, ie a total of 256 I/O. This upgrade was suggested by the TND editor, Geoff Trott, as each address can support 8 inputs and 8 outputs. At the same address there would be an input 74LS373 and an output 74LS373, however only the correct one would be chosen for input or output functions because of the separate Write Enable (*WE) or Data Bus IN (DBIN) signals from the processor.

The gating inputs of IC (3), pins 18 and 19, have been left permanently active to allow use of the outputs of IC (3) for an extra 256 I/O by decoding address lines A0 to A11 in another 16 byte block. (The mind boggles).

15. SUMMARY.

If anyone has made it to this point without cheating and skipping the bits in the middle, then you are to be congratulated, but spare a thought for me, I had to write it, then read it to ensure that it was as bug free as possible!

I hope that the discussion presented in this article has demystified to some degree, both the function of the simple logic gates in addition to showing how these gates can decode an address.

If you want to go further I recommend that you start looking at the data books for the various integrated circuits used in the design of the Wire Accessory Interface as a study of their functions is probably the next step.

Any TI99/4A User Group who wishes to use this article (or in fact any other TI99/4A associated article which I have written) in their group publication is very welcome to do so, but please acknowledge the original author and source.

16. APPENDIX 1.

I do not claim to be an expert on the subject of number conversion but here goes with my understanding of the subject.

To convert a hexadecimal number to decimal, each digit is given a weighting, just like the binary to hexadecimal example in table 6, except this time we are dealing with values in base 16 instead of base 2.

The weighting is from most significant to least significant 4096, 256, 16 and 1. The hexadecimal value of >8684 is converted as follows:

(4096*8)+(256*6)+(16*8)+(1*4)

32768 + 1536 + 128 + 4 = 34436-65536=-31100

When an address is greater than 32767, the decimal value is found by subtracting 65536.

The Logo Logician

by Rick Felzien, USA

During discussions of Logo with numerous members, they have mentioned having trouble with debugging their program. Hopefully this article will be of help to these individuals.

One of the apparently monstrous tasks when programming in Logo is debugging. Due to the intricacies of the language, along with a lack of thorough understanding of the language and how it works, the task of finding bugs seems to be very hard. As I stated in an earlier article, a Logo program is made up of a number of modules or procedures. These all interact to perform the tasks to gain the desired result from the program.

In many languages, when a bug is encountered it usually dumps an error code number on the screen, possibly along with a cryptic error message. Logo has a few standard messages stating the type of problem and its location. As an example, a message "CAN'T XXXX AT LEVEL (NUMBER 1) LINE (NUMBER 2)", means that the problem arose after (NUMBER 1) procedures were executed adding one to the count for each iteration or recursion. The message also tells you that the problem came up in line (NUMBER 2) of the procedure. Usually you can go right to that line and make the necessary change or changes.

When you encounter "TELL ME HOW TO.....", this means that you have typed a procedure name without writing the procedure, or that you have misspelled a name.

The message "XXX DOESN'T LIKE (XXXX) AS INPUT" means that you chose an illegal name for a procedure. Logo will not allow you to use primitives, which are the same as reserved words in BASIC. You cannot use numbers as procedure names in Logo.

The message "OUT OF INK" means that the TURTLE has used up all the tiles allotted (0-31 and 96-255). "OUT OF SPACE AT (LEVEL X) (LINE X) OF XXX" means the program has used up the computer's free memory. This can be avoided by writing efficient procedures and not having blank lines. It also pays to remove all unnecessary procedures from memory that are not needed for the program. "(XXX) HAS NO VALUE" means you asked for a value when the name had not been assigned a value in that program. (e.g. TO POLY :S :A) has values for S and A in the program but not external to it. "(XXX) DIDN'T OUTPUT" tells you that either a primitive or a procedure was used as an input to a second primitive or procedure, but did not produce a value or text as required.

"TELL ME MORE" means that a necessary input was left off a command or operation. Some oddball bugs produce the message "(XXX) WAS GIVEN INSTEAD OF 'TRUE' OR 'FALSE'" or "TELL ME WHAT TO DO WITH (XXX) AT (LEVEL) (LINE) OF (XXX)". The first message comes from supplying improper inputs to test, such as: test 4 + 5 which yields "* WAS GIVEN INSTEAD OF 'TRUE' OR 'FALSE'", where test 4 + 5 = 10 would yield 'TRUE'. The second message is issued when the computer has a number or text with no instructions as to what to do with it. The first example would produce such a message and should be written as in the second.

```
TO FACTORIAL :N                TO FACTORIAL :N
IF :N = 0 OUTPUT 1             IF :N = 0 OUTPUT 1
:N * FACTORIAL :N - 1          OUTPUT :N * FACTORIAL :N - 1
END                            END
```

If a program uses subprocedures, you can see if they have actually been run by using TRACEBACK. TRACEBACK prints the name of the procedures and subprocedures as they are being run. In this way you can see if all are being run and in the right order and whether they are returning control to the calling program or procedure.

There are two very nice procedures for stepping through a program to check each step as it is executed. The other is called CHECKOFF which is called from within STEP.

The STEP procedure.

```
TO STEP :A
PRINT [PRESS ENTER TO RUN THE NEXT LINE.  ]
PRINT SE [NOW STEPPING THE PROGRAM ] :A
CHECKOFF BUTFIRST TEXT :A
END

TO CHECKOFF :STUFF
CALL RC "B
IF FIRST :STUFF = [] STOP
IF FIRST :STUFF = :A [] STEP A
PRINT SE [NOW RUNNING ] FIRST :STUFF
RUN FIRST :STUFF
END
```

If you load the program and then these two procedures, they will be part of your program even while saving and recalling. To use the STEP program, lets say your program is called DRAW. Type in the command STEP "DRAW. The STEP program can be modified to allow you to stop programs inside of stepped programs, or to allow for programs with variable input. o

Cassette Care

from Stephen Shaw, England

The Manchester Central Library have published an excellent leaflet on care of cassettes and cassette players, which is highly relevent to Cassette Users. I have extracted the juicy bits for you.

Musts:

1. CLEAN your recorder regularly. If you use a "wet cassette" you must still clean separately the capstan and pinchwheel, as these special cleaning cassettes only clean the heads properly. See notes later!
2. Keep cassettes in cases, away from heat and magnetic fields and damp.
3. Before putting a cassette in the machine, rotate a spool with your finger to make the tape fully taut.
4. Never use C120s and for best results stick to C60s or shorter. C120s are very likely to stretch and snap, and even C90s may come apart quite quickly. The thinner tapes are more likely to snarl up your machines.
5. Cassettes are not hammers. Observe care.

Detail:

A cassette tape is a VERY thin thing, coated with an even thinner oxide coating, and even under the best conditions, the best tape will shed its coating little by little, onto the surfaces of your recorder, where they stick and begin to scratch your tapes and even more coating comes off.

The PINCHWHEEL is the larger rubber-like wheel which rotates pulling the tape onto the take up spool, while the CAPSTAN is the thin metal wheel which presses the tape against the pinchwheel. If you do not clean these properly, tape is liable to snarl up in your player. The only way to clean them is with COTTON BUDS, dipped in special head cleaning fluid or methylated spirits. Some alcohols may be suitable but NOT isopropyl alcohol, widely used for cleaning disk drives!, as isopropyl alcohol can cause uneven swelling in the pinchwheels.

Clean the heads to avoid undue wear, and losses of audio quality (dullness) which may stop the player being computer compatible.

CLEAN every 10 hours of playing time.

If you use a head cleaning tape, discard it after ten uses, after that it just spreads dirt around.

Tape heads become magnetised over a period of time, as the tape passes over them, the heads gradually build up a magnetic charge from the tape. This could produce increased background hiss, and might eventually result in tapes becoming partly (and irrevocably!) erased while being played. Demagnetise your heads AT LEAST once a year. Special cassette shaped devices can be bought and are safest to use.

With thanks to Manchester City Council Cultural Services.

o

Tips from the Tigercub #48

by Jim Peterson

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 120 original programs in BASIC and Extended BASIC, available on cassette or disk, now reduced to just $1 each!, plus $1.50 per order for cassette or disk, postage and packing paid. Minimum order of $10. Cassette programs will not be available after my present stock of blanks is exhausted. The Handy Dandy series, and Colour Programming Tutor, are no longer available on cassette. Descriptive catalogs, while they last $1, which is deductable from your first order.

Tigercub Full Disk Collections, reduced to $5 postpaid. Each of these contains either 5 or 6 of my regular 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, they are 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 GAMES, MORE GAMES, WORD GAMES, ELEMENTARY MATH, MIDDLE/HIGH SCHOOL MATH, VOCABULARY AND READING, MUSICAL EDUCATION, KALEIDOSCOPES AND DISPLAYS

NUTS & BOLTS disks

These are full disks of 100 or more utility subprograms in MERGE format, which you can merge into your own programs and use, almost like having another hundred CALLs available in Extended BASIC. Each is accompanied by printed documentation giving an example of the use of each. NUTS & BOLTS #1 has 100 subprograms, a tutorial on using them, and 5 pages documentation. NUTS & BOLTS #2 has 108 subprograms, 10 pages of documentation. NUTS & BOLTS #3 has 140 subprograms and 11 pages of documentation. Now just $15 each, postpaid.

Tips from the Tigercub

These are full disks which contain the programs and routines from the Tips from the Tigercub newsletters, in ready to run program format, plus text files of tips and instructions.

TIPS (Vol. 1) contains 50 original programs and files from Tips newsletters No. 1 through No. 14. TIPS Vol. 2 contains over 60 programs and files from Nos. 15 through 24. TIPS Vol. 3 has another 62 from Nos. 25 through 32. TIPS Vol. 4 has 48 more from issues No. 33 to 41. Now just $10 each, postpaid.

```
**************************************************
*       Now ready, TIPS from TIGERCUB VOL. 5     *
* Another 49 programs and files from issues No. 42 *
* through 50.  Also $10 post and packing paid.     *
**************************************************
```

TIGERCUB CARE disks #1,#2,#3 and #4.

Full disks of text files (printer required). No. 1 contains the Tips news letters #42 to #45, etc. No. 2 and 3 have articles mostly on Extended BASIC programming. No. 4 contains Tips newsletters Nos. 46 to 52. These were prepared for user group newsletter editors but are available to anyone else for $5 each postpaid.

If you have ever used TRACE to debug a program, you know that it will not dump to a printer, and that it messes up the screen format. The new Super Extended BASIC, or the Gram Kracker, will dump to the printer, but you still will not know what is going on line by line or within multiple statement lines. Now, Supertrace will break the program into single statement lines and TRACE each statement in the corner of the screen, or dump it to the printer, or both; and you can also pause at any time, or step through it line by line.

```
100 GOTO 140
110 SET,C$,END$,Z$,E$,K$,S$,K,S,IF$,OF$,Q$,FL,TL,M$,
    LN,LN2,P,T,LN$,A$,R,P$,QQ,PD$,KC,KC$
120 CALL CHAR :: CALL CLEAR :: CALL COLOR :: CALL
    SCREEN :: CALL KEY :: CALL SOUND
130 !@P-
140 CALL CHAR(94,"3C4299A1A199423C") :: CALL CLEAR ::
    FOR SET=1 TO 14 :: CALL COLOR(SET,13,15) :: NEXT
    SET :: CALL SCREEN(13)
150 C$=CHR$(157)&CHR$(200)&CHR$(1)&"A"&CHR$(183)&
    CHR$(200) :: END$=CHR$(255)&CHR$(255) ::
    Z$=CHR$(131)&CHR$(147)&CHR$(154)&CHR$(163)
160 E$=CHR$(0) :: K$=CHR$(182) :: S$=CHR$(130)
170 DISPLAY AT(2,5) ERASE ALL: "TIGERCUB SUPERTRACE": :
    "  Tigercub Software for free": "distribution but
    no price orcopying fee may be charged." !programmed
    by Jim Peterson  1/88
180 DISPLAY AT(8,1): " However, if anyone should  feel
    moved to send me a few bucks for the use of this
    program , I would not be": "offended! "
190 DISPLAY AT(15,1): "Jim Peterson": "156 Collingwood
    Ave.": "Columbus, OH 43213"
200 DISPLAY AT(23,8): "PRESS ANY KEY" ::
    DISPLAY AT(23,8): "press any key" :: CALL
    KEY(0,K,S) :: IF S=0 THEN 200
210 DISPLAY AT(2,1) ERASE ALL: " Will break each
    program": "line into single statement": "lines,
    unless they contain"
220 DISPLAY AT(5,1): "an IF, and add a CALL to a":
    "subprogram which will": "display each line number
    in": "the corner of the screen as"
230 DISPLAY AT(9,1): "it is being executed, or": "will
    output it to a printer."
240 DISPLAY AT(13,1): " Program must first be -":
    :"RESequenced to greater in-": "crements than the
    number"
250 DISPLAY AT(17,1): "of statements in any one":
    "line.  (recommend RES 100,20)": : "and SAVEd by":"
    SAVE DSK(filename),MERGE"
270 DISPLAY AT(23,8): "PRESS ANY KEY" ::
    DISPLAY AT(23,8): "press any key" :: CALL
    KEY(0,K,S) :: IF S=0 THEN 270
310 DISPLAY AT(23,8): "PRESS ANY KEY" ::
    DISPLAY AT(23,8): "press any key" :: CALL
    KEY(0,K,S) :: IF S=0 THEN 310 ELSE CALL CLEAR
320 DISPLAY AT(3,1): "INPUT FILENAME?": "DSK" ::
    ACCEPT AT(4,4): IF$ :: ON ERROR 330 :: OPEN #1:
    "DSK"&IF$, INPUT :: GOTO 340
330 CALL SOUND(300,110,0,-4,0) :: DISPLAY AT(6,1):
    "CANNOT OPEN FILE!" :: RETURN 320
340 DISPLAY AT(6,1): "OUTPUT FILENAME?": "DSK" ::
    ACCEPT AT(7,4): OF$ :: ON ERROR 350 :: OPEN #2:
    "DSK"&OF$, VARIABLE 163, OUTPUT :: ON ERROR STOP ::
    GOTO 355
350 CALL SOUND(300,110,0,-4,0) :: DISPLAY AT(9,1):
    "CANNOT OPEN FILE!" :: RETURN 340
355 DISPLAY AT(9,1): " Programs of more than 50":
    "sectors in length may become": "too long to run if
    you break": "and trace all lines."
360 DISPLAY AT(15,1): "Break all lines? (Y/N)" ::
    ACCEPT AT(15,24) SIZE(1) VALIDATE("YN"): Q$ :: IF
    Q$="Y" THEN 390
370 DISPLAY AT(17,1): "From line?" :: ACCEPT AT(17,12)
    VALIDATE(DIGIT): FL
380 DISPLAY AT(17,18): "To?" :: ACCEPT AT(17,22):TL
390 DISPLAY AT(15,1): "TRACE to 1": "": " (1) Screen":
    " (2) Printer": " (3) Both" :: ACCEPT AT(15,10)
    SIZE(-1) VALIDATE("123"): QQ :: IF QQ=1 THEN 405
400 DISPLAY AT(21,1): "Printer? PIO" ::
    ACCEPT AT(21,10) SIZE(-18): PD$
405 DISPLAY AT(3,1) ERASE ALL: " Key code 1 allows the
    pro-": "gram to run until you hold": "down any key.
    It will be"
406 DISPLAY AT(6,1): "difficult to execute CALL": "KEYs
    in the program.": "": " Key code 2 requires a key":
    "to be pressed to execute"
407 DISPLAY AT(11,1): "each program line.  You can":
    "step through the program": "line by line, but this
    may": "be very slow if all lines"
408 DISPLAY AT(15,1): "are being traced.": "": " Key
    code 3does not allow": "stopping the program."
409 DISPLAY AT(20,1): "Key code? 1" :: ACCEPT AT(20,11)
    SIZE(-1) VALIDATE("123"): KC
410 IF KC=1 THEN KC$=CHR$(191)&CHR$(192)&CHR$(200)&
    CHR$(1)&"0" ELSE
    KC$=CHR$(191)&CHR$(200)&CHR$(1)&"1"
411 DISPLAY AT(12,7) ERASE ALL: "Working line"
420 LINPUT #1:M$ :: IF M$=END$ THEN 570
```

```
430 LN=256*ASC(SEG$(M$,1,1))+ASC(SEG$(M$,2,1)) :: IF
    Q$="Y" THEN 440 :: IF LN<FL OR LN>TL THEN PRINT
    #2:M$ :: GOTO 420
440 IF LN>LN2 THEN 460
450 DISPLAY AT(12,1) ERASE ALL BEEP: "ERROR! RESEQUENCE
    PROGRAM TO": "GREATER INCREMENTS AND TRY": "AGAIN."
    :: CLOSE #1 :: CLOSE #2 :: STOP
460 LN2=LN :: IF POS(Z$,SEG$(M$,3,1),1)<>0 THEN
    PRINT #2:M$ :: DISPLAY AT(12,19): LN :: GOTO 420
470 P=POS(M$,S$,3) :: T=POS(M$,CHR$(161),3) :: IF T=0
    THEN 500
480 IF P=0 THEN PRINT #2:SEG$(M$,1,LEN(M$)-1)&S$&C$&
    CHR$(LEN(STR$(LN)))&STR$(LN)&K$&E$ ::
    DISPLAY AT(12,19): LN :: GOTO 420
490 PRINT #2: SEG$(M$,1,P)&C$&
    CHR$(LEN(STR$(LN)))&STR$(LN)&K$&E$ ::
    DISPLAY AT(12,19): LN :: LN=LN+1 :: GOSUB 690 ::
    M$=LN$&SEG$(M$,P+1,255) :: GOTO 430
500 IF P=0 THEN PRINT #2: SEG$(M$,1,2)&C$&
    CHR$(LEN(STR$(LN)))&STR$(LN)&K$&S$&SEG$(M$,3,255)
    :: DISPLAY AT(12,19): LN :: GOTO 420
510 A$=SEG$(M$,1,P-1) :: R=POS(A$,CHR$(132),3) ::
    S=POS(A$,CHR$(201),3)
520 IF R=0 THEN GOSUB 750 :: GOTO 560
530 IF S=0 AND R<>0 THEN GOSUB 700 :: GOTO 420
540 IF S<>0 THEN IF S-R<3 THEN GOSUB 750 :: GOTO 560
550 GOSUB 700 :: GOTO 420
560 LN=LN+1 :: LN2=LN :: GOSUB 690 ::
    M$=LN$&SEG$(M$,P+1,255) :: P=POS(M$,S$,3):: GOTO
    500
570 LN=29999 :: GOSUB 690 :: PRINT #2:
    LN$&CHR$(131)&CHR$(64)&CHR$(80)&CHR$(43)&CHR$(0)
580 LN=30000 :: GOSUB 690 :: PRINT #2:
    LN$&CHR$(161)&CHR$(200)&CHR$(1)&"A"&CHR$(183)&"X"&
    K$&E$ :: IF QQ=1 THEN 630
590 LN=30001 :: GOSUB690 :: P$=LN$&CHR$(132)&"F"&
    CHR$(190)&CHR$(200)&CHR$(1)&"0"&CHR$(176)&
    CHR$(159)&CHR$(253)&CHR$(200)&CHR$(3)&"250"
600 P$=P$&CHR$(181)&CHR$(199)&CHR$(LEN(PD$))&PD$&
    CHR$(130)&"F"&CHR$(190)&CHR$(200)&CHR$(1)&"1"&S$&
    CHR$(156)&CHR$(253)&CHR$(200)&CHR$(3)&"250"&
    CHR$(181)&CHR$(214)
610 P$=P$&CHR$(183)&CHR$(200)&CHR$(2)&"27"&K$&
    CHR$(184)&CHR$(199)&CHR$(1)&"N"&CHR$(184)&
    CHR$(214)&CHR$(183)&CHR$(200)&CHR$(1)&"6"&K$&E$ ::
    PRINT #2: P$
620 LN=30002 :: GOSUB 690 :: PRINT #2:
    LN$&CHR$(156)&CHR$(253)&CHR$(200)&CHR$(3)&"250"&
    CHR$(181)&"X"&CHR$(180)&E$
630 IF QQ=2 THEN 650
640LN=30003 :: GOSUB 690 :: PRINT #2:
    LN$&CHR$(162)&CHR$(240)&CHR$(183)&CHR$(200)&
    CHR$(2)&"24"&CHR$(179)&CHR$(200)&CHR$(1)&"1"&K$&
    CHR$(181)&"X"&CHR$(180)&E$
645 IF KC=3 THEN 670
650 LN=30004 :: GOSUB 690 :: P$=LN$&CHR$(157)&
    CHR$(200)&CHR$(3)&"KEY"&CHR$(183)&CHR$(200)&
    CHR$(1)&"0"&CHR$(179)&"X"&CHR$(179)&"S"&K$
660 P$=P$&CHR$(130)&CHR$(132)&"S"&KC$&CHR$(176)&
    CHR$(201)&CHR$(INT(LN/256))&
    CHR$(LN-256*INT(LN/256))&E$ :: PRINT #2:P$
670 LN=30005 :: GOSUB 690 :: PRINT #2:
    LN$&CHR$(168)&CHR$(0) :: PRINT #2:
    CHR$(255)&CHR$(255)
680 CLOSE #1 :: CLOSE #2 :: DISPLAY AT(12,1) ERASE ALL:
    "Enter NEW": : "Then Enter": "  MERGE DSK"&OF$ ::
    END
690 LN$=CHR$(INT(LN/256))&CHR$(LN-256*INT(LN/256)) ::
    RETURN
700 IF LEN(M$)>150 THEN 720 :: PRINT #2: SEG$(M$,1,2)&
    C$&CHR$(LEN(STR$(LN)))&STR$(LN)&K$&S$&
    SEG$(M$,3,255)
710 DISPLAY AT(12,19):LN :: RETURN
720 PRINT #2: SEG$(M$,1,2)&C$&CHR$(LEN(STR$(LN+1)))&
    STR$(LN+1)&K$&E$
730 DISPLAY AT(12,19):LN
740 LN=LN+1 :: PRINT #2: CHR$(INT(LN/256))&
    CHR$(LN-256*INT(LN/256))&SEG$(M$,3,255) ::
    DISPLAY AT(12,19): LN :: LN2=LN :: RETURN
750 PRINT #2: SEG$(A$,1,2)&C$&CHR$(LEN(STR$(LN)))&
    STR$(LN)&K$&S$&SEG$(A$,3,255)&E$ ::
    DISPLAY AT(12,19): LN :: RETURN
```

This "tinygram" might give you a surprise. SAVE it before you run it.

```
100 CALL CLEAR :: CALL KEY(3,K,S):: ON BREAK NEXT ! by
    Jim Peterson
110 DIM CH$(26):: FOR J=1 TO 26 :: CALL
    CHARPAT(J+64,CH$(J)) :: NEXT J :: FOR J=1 TO 26 ::
    CALL CHAR(J+64,CH$(27-J)) :: NEXT J
120 DISPLAY AT(3,8): "MZNV ZMZOBAVI": "": "GSRH KILTIZN
    DROO ZMZOBAV   BLFI MZNV."
130 INPUT "BLFI MZNV? ": M$ :: CALL
    SOUND(200,110,0,-4,0) :: X=X+1 :: IF X<2 THEN 130
140 DISPLAY AT(12,1): "ZMZOBHRH - ": "": "VRGSVI BLF
    XZM'G HKVOO BLFI LDM MZNV LI MLYLWB XZM
    KILMLFMXVRG."
150 GOTO 150
```

Here's another tinygram that might help you editors who reformat my Tips to wider column widths.

```
100 DISPLAY AT(3,6) ERASE ALL: "TIGERCUB UNFILLER": "":
    " To remove extra spaces from": "a TI-Writer text
    which has": "been Filled and Adjusted by"
110 DISPLAY AT(8,1): "the Formatter, prior to":
    "reformatting.": " It will, however, also": "remove
    paragraph indenta-": "tions and other intended":
    "spacings."
120 DISPLAY AT(15,1): "Input file? DSK" ::
    ACCEPT AT(15,16): IF$ :: OPEN #1:"DSK"&IF$, INPUT
130 DISPLAY AT(17,1): "Output file? DSK" ::
    ACCEPT AT(17,17): OF$ :: OPEN #2:"DSK"&OF$
140 LINPUT #1:M$
150 X=POS(M$," ",1):: IF X=0 THEN PRINT #2: M$ :: GOTO
    170
160 M$=SEG$(M$,1,X)&SEG$(M$,X+2,255) :: GOTO 150
170 IF EOF(1)<>1 THEN 140 :: CLOSE #1 :: CLOSE #2
        Memory almost full...
                Jim Peterson
```

Troubles with Disk Drives

by Bob Bunbury

I bought a couple of 52SA drives from MDS early last year. I never made the time to complete the power supply, or the housing for them, so they sat idle for quite some time. Eventually, something had to happen. It did! The SHOP manager position became vacant, and I filled it. This caused minor domestic disharmony, and created an urgent need for a working set of drives (and a MODEM). A Club Power Supply (and a Club MODEM) should have solved the problem. It did not!

In fact, after several trips to see Bill at MDS, and several more trips to see Peter Schubert (I use a Mini-PE box), I now have a disk system which works most times. It still has no housing, still sits open on my bench, and is still powered by the Club Power supply. So, what were the problems? What were the cures?

In no particular order of merit, or cure, or anything, I list the following points to check:-

a. Disk speed is 300 +/- 2 rpm
b. Clean contacts of ALL connections
c. Check power supply 12v and 5v outputs under load
d. Mini-PE box to console connection is level
e. All cables are connected and no short/open circuits
f. Do not use abrasive cleaners in or near drives
g. Stepping speed of controller and disk drive are matched.

If you find difficulty doing any of the above, contact a service centre before you do any (more) damage to your system.

Colour Fix for Archiver V2.4

by Rolf Schreiber

I did not like the screen and character colours of Archiver V2.4, so with the help of a disk disassembler and sector editor I was able to convert them to white text on a dark blue screen. The colours may be changed by editing sector 22, byte 14 of the disk file. This should be changed from 07CF (green on white) to 07F4 (white on dark blue). It is easier to find sector 22 if the Archiver file is copied onto a freshly formatted disk. Other colour combinations are possible - refer to page 330 of the Editor Assembler manual.

The Forth Column

Forth Forum <4>

by George L Smyth

This month I will present the first of two parts pertaining to the Forth editor. To this point we have been entering programs in the immediate mode. This mode is fine as far as entering words to see if they work as we intended is concerned, but the information cannot be saved in a fashion which will be of real use to us. Because the words tested are in memory, they can be BSAVEd, but not only is that not useful as far as editing is concerned, but does not make sense since there is a much better way to do things. This is by entering the program on a screen or series of screens. To do this, the EDITOR must be loaded into memory. Although I do not think that it is such a good idea, you might also consider using the 64SUPPORT editor, which will give 64 columns across the screen. Since only one editor may reside in memory at one time, load the original system disk and enter -64SUPPORT. Then type '2 EDIT'. Before your very eyes will be your TI screen with 64 columns. I think that you will agree with me that legibility suffers in this mode. You can experiment with this screen, as it is not used by the system. Make sure that the write protect notch is covered, as you may accidentally end up writing to the parts of your disk that the system does use. I speak from experience in this respect. If you goof up information on screen #3, your disk may no longer boot. Anyway, if the write protect notch is on the disk, you can play around without worrying.

Also this month I will show you how to make an automatic loader for your programs. Unfortunately, there is nothing that can be put on a disk like the myriad of Extended BASIC loaders available that will read a disk and present the programs on your monitor. You will have to customize each disk individually, but at least you will see how mine works and be able to use it as a skeleton.

Using the editor

If we were required to write programs the way I have had you enter the examples from previous articles, Forth would have died a quick death long ago. We have been working in the immediate mode so far, so now it is time to enter the programming mode. If you have not configured your disk as per my first article, enter 'EDITOR', and after the load is complete, place a new disk into your drive. I will assume the minimum single drive, single side, single density system for these explanations. After understanding the basics, you can extrapolate to accommodate your expanded system.

A single density disk used in TI's operating environment holds 90K bytes (90X1024 bytes) of data. The Forth disk operating system (different from the disk operating system TI uses) allows the programmer to access the disk in 1K byte increments by presenting 16 lines by 64 columns (16x64=1024) of characters called a screen. This presents the greatest weak point of TI's system, the lack of an 80 column screen. If -64SUPPORT is loaded into the system, the bit map mode is invoked and an entire screen can be viewed on the display. Unfortunately, the characters are so small and ill defined, that readability suffers considerably. Then again, only so much can be expected from characters only three pixels in width. I have the TI monitor and have to put up with the additional problem of having the first and last columns cut off. If you are using a television set, these problems could be greater. The 40 column editor, accessed by entering '-EDITOR' from your original operating system disk, displays a bit over half of the screen, with the ability to toggle screen sides. This is a hassle, but the better of two shortfalls.

A screen can be displayed by entering the screen number and the word 'EDIT'. Screens are numbered from 0 to 89, so you must be within this area. Because your quick load Forth system disk takes up the first 31K (screens 0-30), lets look at the first available screen after this. Enter '31 EDIT'. The screen number, and row and column numbering will be written to your display and the contents of that 1K section of the disk will appear. The contents displayed should be blank characters unless a new disk is being used, in which case the monitor will probably show blocks of non-printable characters. The cursor will be flashing in the upper left hand corner, line 0 column 1. Using the arrow keys, the cursor may be moved in any of the four directions. The 40-column editor, does not have auto-repeat, so to move it 5 spaces you must press the arrow key 5 times. This is a real pain after a while but do not despair. Next month I will present an update to the 40-column editor which will include auto-repeat.

As important as knowing how to enter a screen, is knowing how to leave a screen. To exit the editor press 'FCTN[9]' (BACK). The cursor will be placed below the screen when invoked. At this time you should pull out one of those extra strips of plastic TI gave you with the computer, a blank one similar to the one you got with TEII, Multiplan, TI Writer, etc. (you did save them, did you not?). You can use this strip to write the editor's functions instead of trying to remember all of them. Write "EXIT" in the "FCTN" (grey dot) row where "BACK" would normally be.

When configuring your system disk I had you enter a routine that changed the undefined characters (>E5) left from initialization to blank spaces (>20). This is required because the blank space is interpreted by your Forth system as a delimiter to separate words. The character >E5 would be read by your system as a word in lieu of a space and would consequently produce an error. The word that fills a screen with blank spaces is CLEAR (n ----). CLEAR expects a number on the stack and reads that number as the screen number you wish to be filled with blank spaces. '31 CLEAR' will clear screen #31 of all data by writing over it with blank spaces. If that screen is in the buffer (which we will discuss next), you will have to type 'FLUSH' to read it to disk.

Before we continue, I had better explain the buffer system Forth offers. Ignorance of this system probably led to the demise of several backup system disks when I first began playing with this language. When a screen is loaded into memory, as when you typed '31 EDIT', the information resides in an area called a disk buffer. Various system configurations allow differing number of buffers. TI Forth allows 5K for its disk buffering system, which means that 5 screens worth of data can be loaded into memory and can be made available randomly without having to reaccess the disk. After loading five screens, if you want to load another screen not currently in memory, the last most recently written to screen is dropped and the new screen information is put in its place. If no changes were made on that particular screen, then we are okay. But what if you recently edited the information on that screen?

If there is any difference between the information about to be written over and the original information that was on that screen, the screen has been flagged by the word "UPDATE". UPDATE marks each screen as it is left if the information has been altered, so that when that particular screen is the last most recently written to screen, the information will be written back to the disk.

I have a habit of accidentally changing information on a screen and not remembering how to get the screen back to its original form. The "corrected" form is in memory and my fear is that when that screen becomes the last most recently written to screen, that erroneous information will be written back to the disk. Luckily, we can use EMPTY-BUFFERS. This acts like a giant "OOPS" key in that it effectively erases all information in the disk buffer area. Of course, any information you wanted to keep that is in this area is lost also, so this is a mixed blessing. Then again, we all save our work on a regular basis anyway, do we not?

Next month I will finish talking about the editor and present an extension to our present editor which allows auto-repeat. This function is available in the 64 column editor, but not the 40 column editor. I still have not figured out why, but I will let someone

else try to figure that one out.

By the way, the definition of the word 'OVER' which I said last month could be made from words we already know is:

```
: OVER ( n1 n2 -- n1 n2 n1 ) SWAP DUP ROT SWAP ;
```

The Forum

Thought For The Month:

"Inside every large program is a small program struggling to get out."

I said a couple of months ago that I would present a loader but found the FILTRAN program and figured that it was of greater importance. It also gave me time to reconsider what I was trying to do as far as writing a loader was concerned. I was involved in a classic case of making something difficult out of something easy. I went through all my programs and spent a considerable time determining what extensions were required for each program. I then was tagging each program to load the required extensions. What a pain, and for someone not familiar with Forth, impossible. Then hit upon the blatantly obvious, use the system disk we are already using, the one presented in my first article. Most of the extensions are already loaded, and the only problems are those programs which are too large to allow multiple extensions. They will have to be treated differently, but the vast majority of the programs are not that large.

I placed this program on screens 6 and 7 because they are not used. If you feel you need another screen, screen 2 can be erased and used for your program. Since this disk is dedicated to programs, I have changed screen #3 so that it does not have to bother with a number of the other things I like to have available while programming.

To make matters simple, I used the CASE..OF..ENDOF..ENDCASE construct. To use this loader program, merely change the program name on screen 6 and the corresponding initial screen of the program on screen 7. You may notice that each of my programs begins on a screen value that is multiple of three. Not only can I use 'TRIAD' to print the screens conveniently, but I can also write a word which will INDEX every three screens to see what is on the disk.

A hint you might want to follow is to add the definition ': IT ;' to the beginning of every program. I do this so that after finishing an application I can enter 'FORGET IT' and have the program 'erased' from memory. This erases words used in the previous application from the dictionary and allows you more memory to load new programs. Most of the time this will not be a problem, but if you want to run several applications in one session it is a convenience factor.

```
   SCR #3
 0 ( Load screen )
 1 HEX 10 SYSTEM
 2 9 A GOTOXY  ." Smyth's Forth loading!"
 3 14 BLOAD DROP  10 83C2 C!
 4
 5
 6
 7 DECIMAL
 8   0 DISK_LO !     1 VDPMDE !
 9  90 DISK_HI !     71 7 VWTR
10
11
12
13  6 CLOAD INDX
14
15

SCR #6
 0    ( Loader          GLS  21apr85    )
 1    : CHOICE PAGE ." Enter choice:" CR
 2
 3   ."    A.) MOIRE      F.) THEORY  " CR CR
 4   ."    B.) FILTRN     G.) DSR/PEEK" CR CR
 5   ."    C.) RECDEC     H.) SUICIDE " CR CR
 6   ."    D.) ROADER     I.) DIAMOND " CR CR
 7   ."    E.) TREE                   " CR CR
 8   ." Any other key to enter Forth" CR CR
 9
10   BEEP  KEY 64 -  PAGE  5 9 GOTOXY
11   ." To access this listing again" CR CR
12   ."         enter 'INDX' at any time" CR
13   CR CR ;
14
15 -->

SCR #7
 0 : LDR CASE 1 OF 87 LOAD ENDOF
 1            2 OF 78 LOAD ENDOF
 2            3 OF 69 LOAD ENDOF
 3            4 OF 60 LOAD ENDOF
 4            5 OF 54 LOAD ENDOF
 5            6 OF 51 LOAD ENDOF
 6            7 OF 45 LOAD ENDOF
 7            8 OF 36 LOAD ENDOF
 8            9 OF 33 LOAD ENDOF
 9           10 OF 30 LOAD ENDOF
10
11
12       ENDCASE ;
13 : INDX CHOICE LDR ;
14 INDX
15
```

FORTRAN by D.N. Harris, TIsHUG

With respect of FORTRAN once again, I have had a look at the way it handles strings, and some form of string input is permissible. A special type of variable has to be defined, which I will now look up;

```
1000  FORMAT(A80)
      READ(5,1000) ANAME
```

will enable the user to get a line of 80 characters into the program by inputting from a terminal, interactively. As with everything else in FORTRAN, the thing to be input must be defined, and in this case if you want an 80 character line, you have to instruct an 80 character line, with the FORMAT that stipulates it. All potential inputs such as yes or no could be a FORMAT(A3) since yes has 3 characters, or for Y or N options, one could stipulate the FORMAT(A1). There are error trapping procedures, but assuming the user is a polite and computer literate person, this method may suffice.

Now for why you should want a FORTRAN; it is still a widely used Scientific computer language, and a more sophisticated user group ought to have a go at it.

There is a picky sense of fun in getting FORTRAN to run, compared to the sloppy abandonment of programming in BASIC, and the advantages of FORTRAN lie in the speed of execution and efficient and economical use of the CPU, although I have seen a micro with a 4MHZ processor whip through a bit of work maybe a bit faster than a mini running FORTRAN with slightly older equipment. Much depends on the CPU, but clearly there are some advantages of setting the lengths of all the variables, rather than, as in the case of BASIC, having pre-set default lengths for the interpreter to assume and search through. FORTRAN puts a bit of the load on the programmer. That makes it a bit easier on the machine, but not as easy as giving the user the job of machine language programming, or better still, forcing the user to work it out with an Abacus and a Slide Rule, which duo need no electrical power, just a bit of wood polish or similar now and then.

Another reason to look at FORTRAN is that it is the first Computer Language ever developed, giving reasonably English-like and Algebra-like commands that any fairly bright Fourth Year High School Student could master. It is not really a beginner's language, and again, as a mature user group we ought to look at things beyond that which beginners might attempt, to grow from the foundation skills in programming that we have all to some extent already mastered.

Next article I will take on other features of FORTRAN programming that will show more sophisticated ways of handling variables in programming. I do not have access to manuals on the graphics that FORTRAN can generate, so some of our membership may like to have a little scout around, and also try to get a FORTRAN cartridge that can be as simply plugged into the machine as can Extended BASIC.

PRBASE - a User's point of view

by Lou Amadio

I started to use PRBASE initially to satisfy my curiosity about its potential. I decided, however, that to really evaluate this program, I needed an actual application to try it on. As it turned out, our manual telephone directory (the type in which you zip a small lever up or down to find a name) was on its last legs, having barely survived 9 years and 2 small children. This, I thought, would be a simple, if not useful, example of a data base usage (Werner, please note!). Provision was also made for an address to be included as part of the record for future expansion.

PRBASE is a Fairware utility for which the author is asking for a donation of $US10. In my opinion it is worth a lot more. The documentation (supplied on disk) should be printed out and read at least 2 times before attempting to create a database. I found that the documentation was fairly easy to understand apart from the section on indexing.

Initially I attempted to use the program with a RAMdisk, without much success. I eventually discovered that the only way I could run the program (without removing the RAMdisk) was to turn the computer on while holding down the SHIFT key (that is, disable the MENU program, Ed). The Editor/Assembler module (Option 5 Load and Run) was used to run the Create Program "CRT:1". I believe that an Extended BASIC auto run loader is also available.

The database create program allows you to initialize a data disk, create a data entry screen, design a report and set up printer control codes. The data entry screen is very easy to set up, however, be warned that any fields inserted after the database has been set up and used will muck up all subsequent fields, so a little extra time planning all possible data field requirements will be well spent. (The software author indicates that any additional fields should only be added AFTER the last field once the database has been used to store data.)

Having set up the data field names and sizes, the program will then allow you to dump this design to a printer. The printout also includes a numeric reference grid as well as a companion table which specifies the screen position and size of each field. This information is then used to design up to 5 reports, and is very flexible. A report may include any field from the database in any order to suit the particular output required. (Mailing list labels are also supported, although I have yet to try this option.) The report format procedure is perhaps one of the better features of this database as it allows one to precisely format a report in terms of column position and field length.

The create program also allows up to 5 printer control codes to be stored for quick call up - I saved RESET, COMPRESSED ON/OFF and DOUBLE STRIKE. Additional printer control codes may be stored and invoked automatically when a particular report is run, however I preferred to set the printer codes (within PRBASE) independently of the reports.

Having created the database skeleton on a floppy disk, one can then run the Data Entry and Reporting file (PRB:1) again using Editor/Assembler Option 5. Do not forget to hold down the shift key whilst turning the computer on if you have a RAMdisk. Once the database drive is specified, the program will display the previously created data entry/enquiry screen (using the colours specified at the database creation stage). Data entry is simple and is aided by a very useful help screen (available simply by pressing the "H" key).

Once the data has been entered, I found Sorting to be particularly fast (180 records with a maximum of 180 characters per record, in about 5 seconds). Field and Global Searches were also done in an acceptable time. However using the Index to find a string proved to be lightning fast. On the whole I found this program a delight to use. The only possible drawback is having to reload the Create Program if any changes are required to the reports, but with only 32Kbytes of memory in the TI99/4A there does not appear to be any other alternative. By the same token, once the database is properly set up, the Create Program will hardly be used.

And what of the Telephone Directory? Well it turned out better than I expected, allowing names to be added and subtracted at will and an easy to use report format to be readily printed with just a few key strokes. The next stage now is to produce a Mailing List and Mailing Labels format from the same basic database.

In summary then, PRBASE is very easy to use and it provides a great deal of flexibility in report formatting. You will find, however, that careful planning in the initial stages will save you time in the long run as any changes to the screen or report format requires the create program to be re-run.

Notes:

After initializing and setting up my first database disk I found a curious bug on my data disk in that I could only add 32 records (out of a specified 350 for a SSSD floppy) before the data base was full. Rather than start again, I enlisted the help of my friend Rolf who was able to correct the problem in about 3 minutes. Using a disk sector editor (such as Disk Utilities), Rolf changed bytes 226 and 227 on sector 3 of the data disk to read 015E (this being HEX for 350).

I recommend that you make a backup copy of your data disk every time that you update it - Nibbler sector copier is ideal for this. Due to the non-standard format of the data disk, you cannot use DM1000.

Shown below is a sample data base.

```
  0 01234567890123456789012345678901234567890
 40 ****************************************
 80 *                                      *
120 *          TELEPHONE DIRECTORY         *
160 *                                      *
200 * SURNAME: [              ] Ph: [      ] *
240 *                                      *
280 *   NAME1: {              }STD: [ ]     *
320 *                                      *
360 *   NAME2: {              }             *
400 *                                      *
440 *   NAME3: {              }             *
480 *                                      *
520 * ADDRESS: {                          } *
560 *                                      *
600 *    CITY: {                          } *
640 *                                      *
680 *   STATE: [ ]          POSTCODE: {   } *
720 *                                      *
760 * COMMENT: {                          } *
800 *                                      *
840 ****************************************
880                                     840
```

Figure 1. The data entry screen

DESIGN TABULAR REPORT
TELEPHONE DIRECTORY

Item No	Screen Location	No of Chars	Report Line	Column Position
1	291	12	1	1
2	211	15	1	15
3	231	7	1	33
4	311	3	1	42
5	771	20	1	47
6	0	0	1	1
7	0	0	1	1
8	0	0	1	1

Figure 2. The screen field position and field length

continued on page 30

Open Letter to the Directors

Dear Sirs,

I was very disappointed to hear that Peter Shubert has decided not to produce the RAMdisk printed circuit board which utilises the 32K static RAM chips. This new PCB was designed to fit into the PEB and allow up to 1Mb of RAMdisk per card. I understand that the board was at an advanced stage when Peter decided not to go ahead with it. He cited the purchase by TIsHUG of a number of RAMdisk PCBs designed for the older 8K chips as a major reason for his decision. I can well understand Peter's reluctance to commit a large sum of money to manufacture a number of these boards without some sort of guarantee of sales.

I first heard about the new RAMdisk about six months ago, but Peter was busy designing and building the Mini PE System (which I believe has been and continues to be a very worthwhile contribution to our computer). All this prior work was done, I believe, without any financial help from TISHUG.

This is one project, however, that TIsHUG could and should finance for a number of reasons:

1) Peter has proven that he can design and build quality hardware for the TI99/4A.
2) 32K chips are now about 60% the cost of 8K chips based on a $/Kbyte basis.
3) More memory per layer can be fitted to the new RAMdisk board, consequently this will free up valuable slots in PEBs which are using multiple RAMdisks.
4) The new PCB was to incorporate a number of additional features, such as on-board clock, optional ROS in EPROM and CPU memory.

Although the 32K chips can be fitted to the older board, extensive modifications are required, particularly if a large RAMdisk is contemplated.

In light of the above, I urge you, the Directors, to look at financing this, and other similar ventures, to ensure that TISHUG members have access to the latest hardware within a reasonable period of time.

Sincerely yours, Lou Amadio

PS: We in the Illawarra Group would purchase at least 3 of the new 32K RAMdisk PCBs. I wonder how many other TIsHUG members feel the same way?

Wanted Wanted Wanted

Any old games cartridges, whether working or not are wanted if they have copper on both sides of the PCB (check for contacts on both sides at the edge connector). Examples are Parsec, Alpiner, TI-Invaders etc. Wanted for spare parts only. Prices negotiable. Phone (042)84 2980 7pm to 10pm and ask for Rolf.

continued from page 29

Report Format Design

#	Location	Size	#	Location	Size
1	211	15	17		
2	231	7	18		
3	291	15	19		
4	311	3	20		
5	371	15	21		
6	451	15	22		
7	531	27	23		
8	611	27	24		
9	691	3	25		
10	714	4	26		
11	771	27	27		
12			28		
13			29		
14			30		
15			31		
16			32		

Figure 3. A report specification using the data from Figure 2

continued from page 1

I have been using a copy of TI-Writer editor that has been generating problems for me. When saving the file, it has been putting sectors out of order so that sections of text appear in the wrong place. It is bad enough finding spelling and typing errors without having random errors put there by the software. I am very tempted to change to the Funnelweb system but I need to find how to put it on my RAMdisk with its fast load. I wonder if I will have time this month.

****

I did not have time to contact anyone in Adelaide during my 'holiday', as we were too busy visiting relatives, and people also seemed to be away, either at the snow, or at Expo I suspect. However, on the way back we called into Bathhurst, not to drive around the race track but to drop a child off at Mitchell College, and met Ross Mudie doing the same thing. We had a chat about mutual problems and he told me of the 200 sectors of information waiting for me on the BBS. Would you believe it but when I tried to get my mail from the BBS something played up and I lost information while the data was being written to disk every 40 sectors or so. I tried both Mass Transfer and Telco with similar results, and finally just had to do everything twice. I reported my problems to Ross later and he found nothing wrong, as indeed did I the next time I tried it. Oh, well!

****

I hope you like the cover, as it is our intention to have the same cover for each issue from now on, except if a special theme comes along. Any suggestions for improvements would be welcome, but we are aiming for a cover which allows identification with Sydney, Australia as our place of origin. A standard cover will make our life easier, as in the past we would get 31 pages of the magazine together late on Sunday evening and then scratch our heads for something for the cover!

****

I would like to thank George for doing the article on the newsletters received this month. It saved me a few hours and perhaps I can convince him to do it for me every month.

****

I was having a chat to Lou the other night about his letter to the Directors and those of John Ryan and the Banana Coast Regional Group report and it seemed to me that I had no idea what the board (which runs TIsHUG) has in mind for the future of TIsHUG, either in the short term or the long term. I have not been to the Sydney meeting for a few months, so I do not know what has been said there, but that puts me in the same boat as the majority of the members of TIsHUG who also do not go to the Sydney meeting. Each month I receive articles from Terry about Software and correspondence received, but there is nothing about what the Directors are actually doing or planning to do to keep TIsHUG afloat and the membership happy. I am priviledged to be able to write my thoughts down and allow every one to read them, but the only way you can inform the Directors that you are unhappy with TIsHUG if you are unable to go to a Sydney meeting is to write them a letter like John Ryan and the Banana Coast Group has done, or give up and not rejoin at the end of your current subscription. I would hope that you will not give up and so I suggest that if you share the concerns of Lou Amadio, John Ryan and the Banana Coast Regional Group, that you contact the Directors and share your concerns with them. If you can offer suggestions to the Directors I am sure they will be very pleased. As a start, I would like to see more communication between the Directors and the members to tell us what is being discussed and planned and to keep us up to date on the financial state of TIsHUG. One or two pages devoted to Board activities would not detract from the quality of the TND. Another idea I had is for the directors to go around to each Regional Group meeting once a year say, to allow the members to interact more closely with those running TIsHUG and for the Directors to learn of their problems and views. Does anyone else out there have similar ideas? Write them down and send them in.

Regional Group reports

Meeting summary.

Banana Coast	14/8/88 Sawtell
Carlingford	17/8/88 Carlingford
Central Coast	13/8/88 Toukley
Glebe	11/8/88 Glebe
Illawarra	15/8/88 Keiraville
Liverpool	12/8/88 ???
Northern Suburbs	25/8/88 Davidson
Sutherland	19/8/88 Jannali

BANANA COAST Regional Group
(Coffs Harbour area)

Regular meetings are held in the Sawtell Tennis Club on the second Sunday of the month. For information on meetings of the Banana Coast group, contact Kevin Cox at 7 Dewing Close, Bayldon, telephone (066) 53 2649.

Eight members attended the last meeting with a couple of members on holidays and so not able to attend. The afternoon was spent in two groups, one group copying disks recently acquired at the TI-Faire in Brisbane, at which 3 members attended. The other group was teaching a new owner of a TI99/4A the fundamentals of the use of his new acquisition. We hope that a software exchange can begin with one of our member's brother, who lives in another country area. We are also requesting the board to send us a disk each month to demonstrate at our meeting as we are not able to come to the Sydney meetings on a regular basis. We feel that we are disadvantaged by the distance from Sydney and that the local groups have ready access to a lot of programs and hardware that we do not get.

CARLINGFORD Regional Group.

Regular meetings are usually 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.

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.

Last meeting (attendance 12): A good time was had by all. TI-base database, TI-ARTIST v2.01, and other programs were shown. The M.G tute went well and lasted almost 1 hour. August meeting: 11/8/88 Tute topic tba. Hardware demo: (Haste providing) We may have the new Myarc hard disk controller available. Well at present that is the agenda. Weasel.

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 Bob Montgomery on (042) 28 6463 for more information.

LIVERPOOL Regional Group

Regular meeting date is the Friday following the TIsHUG Sydney meeting at 7.30pm. Contact Larry Saunders (02) 644 7377 (home) or (02) 759 8441 (work) for more information.

Meetings coming up.

August 12th 1988 at Steven and Jenny Carr's house. 146 South Liverpool Rd, Busby Near Banks Rd. (02)608 3564
September 9th 1988 ???

Last meeting was at Hans Zeckevic's house. It was a medium turn out. Ran SPLAT III, CPUTEST and some programs from TIsHUG competition entries. Also showed some more ways of using EZ-Keys program. For example, showed the edit mode that is almost as good as Super Extended BASIC Cartridge.

NORTHERN SUBURBS Regional Group.

Hello again, this is Dick Warburton cordially inviting you to attend the next meeting of the Northern Beaches' Group. We have gradually improved our meetings,and have a small but regular group of 5 or 6 attending.

Our meetings for the next two months are as follows:

Thursday August 25th at Dennis' home at 24 Woolrych Crescent, Davidson. A technical night: Clock cards; consoles; drives etc.

Naturally we try to fit in as much copying as possible. We also provide something to eat and drink. We welcome visitors. If you want any information please ring Dennis Norman on (02)452 3920, or Dick Warburton on (02)918 8132. See you soon. Last meeting we covered the effective use of printers with TI Writer. We have organized our meetings so that we cover some aspect of Funnelwriter, look at problems in Extended BASIC programming, and cover a main topic as well.

SUTHERLAND Regional Group.

Any persons interested in joining the Sutherland Group are more than welcome. The format of the meetings are very informal as more often than not the conversation digresses onto matters purely social rather than related to computerisation. The supper is not bad either. Meetings are held on the third Friday of each month. Group co-ordinator is Peter Young, (02) 528 8775. BBS Contact is Gary Wilson, user name VK2YGW on this BBS.

Future meetings will revert to the home of Peter Young at Jannali at 7.30pm.

The latest project of the Sutherland Regional Group was to investigate the suitability of various Data Base software held in the club library. The June meeting spent considerable time delving into the mysteries of PRbase. The results were none too successful. Better results were achieved with Creative Filing System which appeared to be far more user friendly. At least we were able to create files, sort, search etc. Derek Wilkinson brought along his RGB Monitor, which was connected for the evening. The result was a very clear resolution, which points to the next logical step; an 80 column card. Thanks go to all those persons who contributed to the running of the All-day tutorial in June. Special thanks to our tireless software librarian, Terry, who had no shortage of customers for the entire day.

TIsHUG in Sydney

Regular meetings are normally at 2pm on the first Saturday the month, except January and possibly other months with public holidays on that weekend, at the Woodstock Community Centre, Church Street, Burwood.

Meetings planned this year. August 6 - Swap meeting and market day. Do you have some old modules, software or hardware you do not use anymore? Chances are that your unwanted gear is wanted by someone else and this is your chance to sell it, trade it or give it away. This meeting will start at 2pm so bring along your unwanted items and perhaps go home with a few extra $$$ in your pocket. The group can accept no responsibility for the condition of goods on sale or to be swapped. All are sold/exchanged on an "as is" basis.

The Geneve demonstration may also finally become a reality at this meeting

September 3 - Software demonstrations and purchases. Again this will be a normal 2pm start. At this meeting some of the latest software as advertised in MICROpendium will be demonstrated and hopefully we will have imported copies for sale direct to members at reasonable cost.

October 1 or 8 - Joint meeting with Hunter Valley 99'ers. Sorry for confusion on the date but it is yet to be confirmed whether the meeting will be on the long week-end or the weekend following that one. Anyway, contact has been made with our friends in Newcastle and we are awaiting their response as to whether a joint meeting is possible. Arrangements envisaged are a computer meeting in the morning with a social afternoon. There will be more details on this as we get closer to the actual meeting date. If in the event, this proposed joint meeting does not go ahead then alternate arrangments for a topic will be made.