TIHUG

NEWS DIGEST

Focusing on the TI99/4A Home Computer

Volume 9, Number 8 September, 1990

Registered by Australia Post - Publication No. NBH5933

Living with Spiders . . .

. . the Saga Continues

P.O. Box 214, Redfern, New South Wales, Australia, 2016 $3

Health warning: Reading the TND may be addictive

Co-ordinator's Report

by Dick Warburton

Recently I visited a friend's home, and was surprised at the number of computers in the house. All were fairly new, and would have cost a tidy sum. For Example....

*A top of the range Compaq AT with a 60 meg drive plus VGA.
*A brand new expanded Archimedes with hard disk and colour monitor.
*A fairly new Amiga 500 with 2 drives.
*Finally, a slightly older BBC, again expanded.

I estimated the value of the systems to be in excess of 15000 dollars, without software. The software they have bought is also expensive. This set me thinking about the cost of being an up to date computer user, and where it is leading. All the different computers in this household are used for different things. When I arrived, the son was using the Archimedes (with its RISC chip technology) as a word processor. The BBC is used as an educational tool to prepare work for school, while they all use the Amiga for games, and desktop publishing, while the faithful old Compaq spends its time crunching business software. And here we are, using a ten year old computer to do most of the things done by all of them, and at a fraction of the cost. One can expand a TI994A today for a small fraction of the cost of other computers. Software costs are negligible compared to standard program costs. eg. I use Funlweb with all its excellent utilities and features as freeware, but if I bought Word Perfect for the IBM compatible or the Amiga, I would have to outlay between 500 and 600 dollars.

What I am saying is that we have a most versatile machine which can be used in a wide variety of ways. We all enjoy its games capability, its graphics, its ability to be used for almost any task we really want. With Ramdisks, Rambo boards, Hard drives, we can run a variety of programs which certainly surprise other computer users, and would I suspect those who actually designed this machine as well. It is still slow but seems to work fast enough for what I want it to do. Do you really need a 33 meg 386 or 486 to play games or do word processing. Imagine Parsec or Munchman running at a genuine 33 megs. Challenging? Software continues to improve, and more recent releases have proved popular. TI users can have as much fun, and probably do a wider range of things with their machines than other computer users. Furthermore, TI users seem to learn a lot about computers generally, probably because they have to look after themselves and have become more self reliant and knowledgable.

Many people who own other makes of more popular computers, are locked in to what the manufacturer wants them to do. Commodore 64 machines do not use standard disk drives, and naturally are dearer to buy. Earlier Amstrads used a 3" disk drive with extremely dear floppy disks. With many of the modern machines, it is difficult to make changes to their configuration, because the design will not allow it to happen. If you want to expand, you do it the manufacturer's way and get ripped off. The obvious examples are the fitting of hard disks to machines like the Macintosh. With the TI, one can keep expanding the system cheaply, as long as we are content to lope along behind the galloping technology.

People I know with more modern expensive systems, do not seem to spend time programming them. One TI member, whose identity shall remain secret, uses his Macintosh to run his business, but programs on the TI. It seems that many a modern user has only a specialised use for his computer, and the wider range of skills available to TI users is slowly being lost, as more and more users learn less and less about what computers are. This is in keeping with the current Mushroom Theory so beloved by politicians. People are being conditioned to accept technology without learning to control their own destiny. We are slowly becoming a dependent and servile society, because we are accepting what is given out without question. Motor vehicles are becoming more complex, and less repairable, with computerised fuel systems etc. With computers, The more modern user is accepting machines which certainly run faster, but which are software dependent, and the hardware often cannot be repaired. Take disk drives as an example. I bought a new Teac 3.5" drive and burned out the stepper chip because the 5 and 12 volt leads were reversed. I am completely unable to get it fixed, because of the greedy marketing strategies of the Japanese manufacturers who will not make spare parts available. I have to throw it away, or perhaps use it for parts if I am able to buy another one like it before they change the model. In fact the more I think about this issue, the more concerned I am becoming that Australians are being enslaved to overseas technical interests, because we are losing our independence and self reliance.

I suspect that I am becoming aware of these issues because I have grown up on the TI, and I am slowly becoming more self reliant. Owning and using a TI is fun. It can be challenging. There is so much that I want to do and find out about TIs. The way its going, I cannot see myself using another machine seriously before the turn of the century. There are so many projects I wish to try as soon as possible. I want to learn to program effectively, and perhaps master Assembly. I want to complete a multifunction card, and build a mini expansion system using the 3.5" drives in the shop to take to work.

With a program like PC Transfer, I am now able to create or edit text files on the TI and transfer them to the IBM machines at work. I suspect that there are quite a few people like myself in the TI community, who are still challenged by this machine, and who value self reliance and independence. Certainly, the shop is busy, and mail orders are on the increase. Kits are being prepared, eproms are being burnt, and a new run of ramdisk cards is likely. We are buying carefully to get the best available prices for our members, and the range of items available through the shop is increasing steadily. Hopefully, there will be another release of colour monitors, and interface kits. There are 3.5" drives, brand new,for only $20 each. With a double density card these will format to 180K or 720 sectors. With a TI disk controller card, they can still be used but only as single sided single density drives. Two of these drives can make an excellent small TI system.

It seems to me that we can all join the computer rat race if we want to, (at high cost) or we can move along more sedately in these economic times, steadily widening our knowledge of the TI, improving it, and being challenged as well. In short we can have fun together.

See you at the next meeting, Dick Warburton o

Assembly Tutorial

Before the start of each meeting

by Ross Mudie

At the TISHUG meeting on 4/8/90, a number of members requested that I commence a series of BEGINNERS ASSEMBLY LANGUAGE tutorials which are held at the monthly TISHUG meetings at Ryde. The needs of this small group is to start at the beginning with a regular tutorial session. The group want entry to the class limited to those who start in the group so that there is no need to go back for new people joining the group. Obviously, if these people are as serious as they claim to be, they will be successful in learning assembly.

The initial sessions of the assembly class will be held at the next three TISHUG meetings as follows...

September 1st - 12noon to 2pm. (Buy, swap and sell).
October 6th - 10am to 1pm. (Full day tutorial).
November 3rd - 12noon to 2pm.

continued on page 18

Letter to the Editor

Dear Bob,

I have been reading the news digest and trying to think of ways that I could contribute to it. I have been a member of the TIsHUG for about seven years now, and although I would not class myself as an avid computer user, I do use it in spurts!

I first purchased the TI in 1982 primarily as a games machine and to help my two boys (pre_schoolers then) learn the basics of education. The TI is still the best for this in my opinion! I also used it in my employment to work out some electronic problems. Then, it would end up sitting there gathering dust, not being used by anybody.

All this time I maintained my membership in TIsHUG. Why? I enjoyed reading the magazine. Even if I did not use the computer from one month to the next, it was still interesting to read what others were doing. I did not get into programming, and I do not see that I ever will other than in simple basic. I have not attended many meetings, mainly because I live out of Sydney and my week ends are taken up by my own and my childrens sporting time.

So what am I leading to; well I would not like to see the club fold! How can I help? Having read in the Digest over the last year or so that you (the committee) would like to know what it is that we, the members, want, I thought that I would come forward with a suggestion. There is no doubt that there are members out there like myself who would like to put their TI's to use but do not know how to go about getting the best program to work in the most efficient way. I feel a little embarrassed to ask for help because the ones I would ask are into "higher" things to do with the computer. Are there others out there that feel as I do? Please believe me, this is not a complaint or bitch against the club or any member. If I was unhappy with TIsHUG I would not have remained a member for so long.

Now, having said that, I would like to suggest that members be invited to write in with there own problems and that the members of TIsHUG could, via the TND, be able to give their solutions. This would help that person, and at the same time maybe get some renewed interest in the TI and the club. I could like to start the ball rolling.

Here is a use I would like to put my TI to. I am in the local sport fishing club and I would like to use the computer to keep a record of the fish captures for the month, calculate the points value of each capture, sort them into their various catagories and place the highest point scoring fish to the top of the list. Then compare the monthly captures to the club record for that fish and if it beats the current record to update the record list.

The formula for working out the points value of a capture is:

$$\frac{\text{weight} \times 100 \times \text{fighting factor}}{\text{line class}}$$

For example, weight = 2.9kg, f/factor = 1.0, line class = 2kg.

$$\frac{2.9 \times 100 \times 1.0}{2.0} = 145$$

The information is supplied in the following format:

SPECIES	WEIGHT	LINE	DIV(ISION)	*ANGLER	POINTS
Aust salmon	1.9	2	1	J.McEWAN (L)	114
Tiger shark	200.0	15	4	P.SHARLAND	1066

*L:lady, J:junior

Of course the points would not be worked out when the capture form is handed in to me. The above format is the way that the results need to be printed out for the club news letter.

I have tried using Multi Plan for this task, and it did work. However, I could not get it to do all the things I wanted. This is where the more adept members of the club may be able to come to my assistance.

I hope my ideas are of use to the club, and I look forward to reading of any way that I can utilise my venerable old TI99/4A for this task.

Kind regards

Michael Ball
8 Caroola Pde
North Nowra
NSW 2540

o

Editor's Reply

Dear Michael,

Thank you for your refreshing letter and the problem that you have given to challenge us with. I am sure that there are many TI users out there that have similar thoughts and would appreciate a portion of the TND given to answering questions like the one you proposed. If I get enough response to your question and have others write in asking similar things, I will form a Question-and-Answer section in the magazine. When anybody in the past has written in asking for help with a problem (to my knowledge) there have always been members who have responded to the call. This is what it is all about! You have made a good point in that we sometimes get caught up in the "higher" things that we lose sight of the more down-to-earth basic skills with the computer.

I was thinking about passing out a survey asking members which of the articles they read the most in the magazine. If we are putting things in that most people never use then we want to know about it. Similarily, if we are not meeting the needs of any section of our membership then we want to know that as well. Letters like yours Michael are always welcome.

Well members, who is going to take up the challenge and come up with a program to solve Michael's problem? Even if you are not totally successful, let Michael or myself know what progress you have made.

o

Letter to the Editor

The future of TIsHUG

This is an open letter consisting of comments on the Co-ordinator and Secretary's reports regarding the future of TIsHUG and the BBS.

Firstly, an apology....I was not be able to attend the August meeting due to being rostered for day shift at work.

Next, the BBS. I have been "off the air" for the past 3 weeks or so due to being in hospital, but am usually a fairly regular user of the service. I make use of several other bulletin boards as well as TEXPAC and my impression is that all or most are suffering a drop in useage. This factor, as well as high costs and the activities of "ratbags" and electronic vandals, has actually caused some good boards to close. A case in point was the late, lamented PROPHET BBS, a very good multi-line board offering access to the FIDONET international conferences and mailing services. These services do not come free, but I felt that I was getting value for money.

Perhaps there is a possibility of arranging a merger of TEXPAC with another existing BBS. This would mean paying a bit more, but there is the possibility of a wider range of services. What do others think?

continued on page 6

Secretary's Notebook

by Terry Phillips

The weather cleared nicely for the August meeting and a roll-up of around 60 members was the result.

The Special General Meeting commenced the days proceedings and after discussion on the proposed amendments to our Articles of Associate, the voting took place with the following results:

1. Paragraph 26 - carried by a majority that the clubs annual general meeting be held in December, if practicable, in lieu of February, in each and every year.

2. Paragraph 49(a) - carried unanimously that the clubs financial year be from 1 July to 30 June in each and every year in lieu of 1 January to 31 December in each and every year.

3. Paragraph 52(a) - carried unanimously that this paragraph be amended to include reference to the clubs financial year and that reference be shown to paragraph 32 in lieu of paragraph 31.

4. Paragraph 62(h) carried by a majority that this paragraph be amended to have a quorum equal to four-fifths (4/5) in lieu of the current two-thirds (2/3).

With these formalities out of the way the meeting then took on its more familiar role, and I noticed that Geoff was very busy all afternoon with his repair shop and Percy, as usual, was flat out in the shop.

We have two new members to give a big welcome to this month. They are:

Maria Kelly of Greystanes, and, Victor Muller of Thornleigh.

Victor was at the August meeting with his non-working TI. Geoff has taken it away for surgery and it should be back and operating at the September meeting.

Pleasing to see was Ross Mudie's report for the month of July which shows BBS usage of 168 calls against the 96 recorded the previous month.

At the next meeting we have the ever popular format of BUY SWAP SELL. These days are always well attended and most members seem to be able to sell what they bring along or buy what they came to buy. I understand there will be a couple of full systems for sale at the meeting and that they will go fairly cheaply. So if you are looking to expand, make sure you come along to this meeting as it may be a rare opportunity to expand at the right price.

See you at the meeting. o

The Computer is Innocent!

by Ben Takach

Errare humanum est. This Latin saying, an inheritance of Roman days, will prove that human error is not an invention of our electronic age. One may also reach the conclusion that it will be more common as life gets more and more complicated. Publicly admitting a mistake is certainly embarrassing, however it always prompts a generous forgiving.

The above introduction was inspired by a recently received letter, sent by a state government corporation- albeit not the first one of its kind. The corporation is one of the many bodies which prospers by the user pays principle. The user is invariably a passive subject, with absolutely no say or input with respect to the function of the body. The only active part allocated to the user is to pay the ever increasing fee for the service, which used to be free anyway. One can no longer obtain a copy of a birth or death certificate for the payment of the stamp duty, the cost per word is higher than the charges for an overseas telegram. Oops, this was my own human error,a telegram is also a thing of the past, we are forced to send electronic mail instead, - whatever the difference may be, apart from the steeply increased cost. However, this is another story, let me return to the original subject.

I will selectively quote from the letter received- enough to make the matter clear:

"...Recently you would have received a Gold....Licence to replace the.....certificate of Registration held by you. As a qualified individual your Gold Licence should have a Q clearly visible on the front of the licence. Unfortunately, a computer malfunction has resulted in some licences incorrectly having the Q over printed.."...etc.

Well, you and I know better! The mistake had nothing to do with computer malfunction! It would have been more truthful to say: "...Recently one of our ill-trained and disinterested idiots has made a mess of printing the newly issued Gold Licences. His mistake was further aggravated by his superior's omission to check his work, claiming it could have breached some paragraphs of the anti-discrimination act"..

Just in case my crystal ball has failed to project a clear picture of the electronic licence issuing process, hereunder is the second likely version.

"Due to the unavoidable communication gap between the officers of the Corporation and the Computer Programmers of the Department of Electronic Services, the 'Gold Licence' program is flawed. Consequently the issued licences are incorrect. We have not yet established the identity of the moron who omitted to check these, and ensure that the cards are correct, as such an inquiry may constitute a breach of one or more acts of Parliament.."

I imagine this second version could be true, alas, no self-respecting bureaucrat would admit to any departmental inefficiency. Rather, they would willingly double the staff to check the checkers - after all the user will pay for it.

So, let us try a watered down absolutely neutral third version: " Due to circumstances beyond our control, some of the recently issued Gold Licences were incorrect. We regret the inconvenience.".

Clearly an admission of not being in control is out of the question. It would open the door for an inquiry, which could have some very embarrassing consequences. First, the idea of the Gold Licence was the brain-wave of the Big Boss. Although it was totally unnecessary, costly and time-consuming, who the hell would be foolish enough to argue with the Chief!

Second, since the recipients are still holding the original - admittedly not very gold, but equally effective - Certificates, they can prove their competence if ever it becomes necessary. Thus, the best way out of the predicament is to blame it on the computer.

Well, as I said before, you and I do know better: The computer is innocent! o

Wanted

Wanted: Starting Forth by Leo Brodie is anyone willing to sell? 066 512485. o

TIsHUG Shop with Percy Harrison

We have ordered another twenty coloured monitors and already more than half of these have been spoken for so any member still wanting to acquire one should get their order and money in to me as soon as possible as our source for these superb monitors is quickly drying up. Price is still $122.00 including the PC Interface Board.

I am sorry that we ran out of monitor Interface Kits at our last meeting but this has now been rectified and they will be available at our September meeting. Optional extras for this kit will be Jiffy Boxes and Transformers, see below for prices.

If your computer is playing up do not forget that we now have an exchange system through the shop where for $30.00 you can trade your "sick" model in for a working one with a twelve month guarantee. As I am writing this we are in the middle of the August deluge and I am hoping that our country members living (or should I say swimming) in those flood-bound areas have done the right thing and grabbed their most valuable possession, the TI99/4A, before evacuating their flooded homes.

We now have the upgraded version of TI-Base (Vers 3.0) available and those members who want to upgrade from version 2.03 or later can do so for a fee of $15.00 by returning their two original disks to the shop. TI Artist Plus is on order from the States and hopefully will be available at the September meeting or shortly after. Those members who have already ordered and paid for this program will receive their copy as soon as it arrives.

I would like to personally thank Alf Culloden for his kind assistance in assembling Kits for those members who do not have the expertise or who for other reasons are prepared to pay the club for assembling their kits and also to the Wollongong Technical Group for testing and setting the MFC boards and repairing consoles. Without the help of these people I am afraid that many of our members would be in dire trouble when their computer goes down or they want to expand their system.

Items available from the shop:

New release software disks	$2.00
(See "Software Column" for details)	
Asgard News Vol. 2 No.3	$3.25
5.25 in. DSDD Disks-Box of 10	$6.50
5.25 in. HD Disks-Box of 10	$16.00
3.5 in. DSDD Disks-Box of 10	$15.00
Horizon Ramdisk EProms	$20.00
Multifunction PC Board	$30.00
Multifunction Dsk Controller Kit	$102.50
Two-way Interface PC Board	$23.00
Monitor Interface Kit (no jiffy box)	$21.00
Jiffy Box (for Monitor I/F Kit)	$7.00
Mon. Power lead (needs std 3 pin plug)	$1.00
Music Kit (including PC Board)	$65.00
Lithium Battery 3.6V	$4.00
Mini Speaker - 8 Ohm	$1.50
Transformer - 60VA	$20.00
Transformer - 13V Arlec	$12.00
Monitor Transformer - 12V ac	$3.50
32K Printed Circuit Board only	$7.00
Module PC Boards (Single chip vers.)	$9.00
32K Memory Chips	$17.00
TI Technical Data Manual	$15.00
TI Sort	$15.00
Display Master	$15.00
TI Base upgrade (see above)	$15.00
Adventure Hints Booklet	$2.00
Pro-Mod Booklet	$1.00
TI Writer for Novices Booklet	$1.00
Extended Business Graphs Booklet	$1.00

Packaging and Postage Charges:
Up to 2 Disks----------$1.00
Over 2 up to 5 Disks-----$2.10
Over 5 up to 12 Disks----$3.00
3.5 in. Drives---------$10.00
Other Items at P/P Costs.

Note:August TND prices for 60VA and 12V ac Transformers were incorrect, my apologies for any inconvenience this may have caused.

Finally our best wishes for a speedy recovery goes to our long standing member and good shop supporter Ron Kemp who has undergone massive heart surgery which was followed by a very serious setback. I am happy to report that, after twelve weeks in hospital, he is now well on the way to recovery and we hope to see him back at the club meetings in the near future. All the best to you Ron from all your friends at TIsHUG.

o

TIsHUG Software Column

by Rolf Schreiber

Firstly, I must apologize to those of you who bought the TI Print Shop disk at last month's meeting and were unable to get it to work. Sample data files, of the type GR..TXT and GR..XXX, were missing and without these files, or a conversion program to convert TI Artist Instances into TIPS compatible picture files, one cannot do very much with the program. I am trying to obtain a complete TIPS package from overseas, so please be patient.

Secondly, I would like to bring to your attention a bug in the latest version of Disk Review in Vn 4.30 of FWB. When you try to initialize a disk as DSSD, the program formats the disk as SSDD (if your controller card can handle DD), which can be a problem if you pass on such a disk to someone whose system cannot handle DD. The details of the fix will be published next month.

The following disks will be available from the shop, or by mail order, from this month:

DISK A309 contains a set of four archived music disks by Sam Moore Jr. These disks were originally released by Jim Peterson as Public Domain Disks TC600 to TC603, and need Archiver to unpack them.

DISK A392 is Don Shorock's Educational Disk #2.

DISK A398 is SBUG II Version 2.0, by Edgar Dohmann, reviewed by the author in the TND in February 1990.

DISK A399 is NUCLEAR 99'ER, an interesting simulation game involving running a nuclear power station.

DISK A402 is TI Writer V4.5, by R.A. Green. An earlier version was reviewed in the TND in February 1990.

DISK A403 is TI KEYS by Wes Johnson. This utility was reviewed in the May 1990 issue of the TND.

DISK A404 is called Bright Beginnings, a collection of educational games, suitable for children aged 3 to 8.

continued from page 13

As you can see, variables up to three characters in length ran in about the same time. Once the length was longer, however, each additional character in the variable name increased the run time by about one tenth of a second for 1000 executions or .1 millisecond for one.

I also ran this one:

```
10 C=0 :: FOR I=1 TO 1000 ::
A=0 :: NEXT I
```

The average run time was 7.06 seconds. There is a cost when substituting variables for numbers. TINSTAAFL (there is no such thing as a free lunch)!!

DID YOU KNOW?

You can have more than one item for a VALIDATE. For example,

```
10 ACCEPT AT(5,1)VALIDATE(DIGIT,"Q")
BEEP:F$
```

Will accept the ten digits and the letter Q. It does. Enjoy.

o

Techo Time

by Lou Amadio

At the July meeting we voted on a few changes to the articles of association. The discussion and voting took up an awful long time considering most of the changes were relatively minor. Anyway, all of the proposals were eventually accepted. There was no formal discussion on the future of TIsHUG although several members said that they agreed with the sentiments of the Illawarra Regional Group published in the August issue of the TND.

Zeno Board

I demonstrated a working Zeno "Internal" Board at the July meeting. There was a good deal of interest in the board which basically allows you to add (internally) 32K memory, Extended BASIC, Speech Synthesizer, clock as well as a number of GROM based modules (up to 6 chips). I was able to demonstrate all functions except the clock which has not yet been added. The Zeno board is soldered directly to the back of the GROM (module) port with an additional connection to the mother board via a 16 way ribbon cable and plug. The hardest part of building this project was removing the chips off the the old Extended BASIC and Speech boards. Alternatively, chips may be purchased from TI in the USA. Please contact me if you need the address.

The whole assembly is supported by a strip of foam rubber and fits neatly above the console mother board. As I have said before, why did we not have this 5 years ago....?

Those that are interested in building an internal board should enquire with Percy Harrison at the club shop. Orders will be placed with the suppliers in the USA if there is sufficient interest.

Adding Modules Inside the Console

Although the idea of installing modules inside the console is not new, interest in this hardware modification was renewed at the July meeting, mainly due to the Zeno Board demonstration.

Installing some of your modules inside the console makes sense as it makes them available at the flick of a switch and independent of the GROM socket, which is probably starting to show signs of wear and tear on most consoles.

Ross Mudie described how to install the Extended BASIC module inside the console in the August 1988 issue of the TND. This is probably the most worthwhile module to add internally. Those that want to attempt this modification should contact me regarding some errata on the subject. As the Extended BASIC module contains 8 chips (4 GROMs, 2 ROMs and 2 TTL) as well as a number of other components, it makes sense to simply wire the module (minus case) to the back of the GROM port. As you will see from Ross's article some switching is required so that you are still able to utilize the GROM port for other modules.

Fortunately, most of the other useful modules simply contain GROM(s) and nothing else. This includes Editor Assembler (1), Disk Manager (2) and TI-Writer (1). Modules may contain up to 5 GROMs, for example Multiplan.

It is relatively easy to transfer "GROM only" modules into the console. The GROM chips are simply removed from the module PCB and soldered directly over the system GROMs on the mother board behind the TMS9900 CPU. (The system GROMs are socketed and numbered CD2155, CD2156 and CD2157.) All pins of the added GROM chips are soldered except for pin 14. If more than one GROM is added to the mother board , each module function must be separately switched via pin 14 on the GROMs.

For example, if you want to add Editor Assembler (1 GROM chip) remove the chip from its original PCB using solder wick, bend out pin 14, place the chip over one of the system GROMs (it does not matter which one, but be careful of the orientation) and solder all pins (except 14) of the E/A GROM to the system GROM. Locate the -5 volt supply on the mother board (use a multimeter and look for the brown wires from the power supply) and connect it to pin 14 of the E/A chip via a switch. Mount the switch on a convenient part of your console (I used the rib to the right of the On/Off switch). You will find that if you close the switch contacts and reset the computer, that Editor Assembler will appear on the TI menu screen.

You can add as many GROM based modules as you wish (within reason) using the above procedure. Remember to bend out all of the pin 14's for each module group. If there is more than one GROM in a module, wire all of the pin 14's together and then to the switch which will select -5 volts. If you wish to add two modules, then you will need a three position switch, three modules need a four position switch etc. The extra position is used to switch all modules off so that you are able to use the module port of the console. Also, more than three positions usually call for a rotary rather than a slide switch (single pole, unless you also install Extended BASIC which requires a double pole switch).

I also found it convenient to install a console reset switch. This obviates the need to turn off the power to reset the console (see TND, June 1989).

Next Month

Geoff is currently working on a PIO only version of the Multi Function Card for those who want to use a parallel printer. o

continued from page 3

With regard to the future of TIsHUG, I am very much in favour of widening our membership to users of other types of computers. I use an MS-DOS machine at work, but my 99/4A serves me quite adequately at home. I know that there are quite a few other members in a similar situation. I quite often transfer text files between the TI and the PC using the TEXPAC mailing facility.

On the subject of "obsolete" computers, there is an excellent article in this month's edition of "Electronics Australia" on page 58. The TI itself did not get a mention, but a few others of the same vintage were included. The title of the article is "The Easy way into Computers" and is written by Tom Moffat.

I suspect that there are a lot of users of other brands of "orphan" computers floating around out there who would be potential members of TIsHUG in an expanded role.

I look forward to hearing others thoughts on these matters. With regards...Tony Beuermann...1 August 1990

Since my last message, a couple of days ago, on the subject of the future of TIsHUG, I have read the letter in the August TND from Lou, Rolf, Geoff and George. I found it very interesting and thought-provoking, but cannot agree entirely with their views on expansion of our membership base to include users of other types of computers. If I may just clarify what I had in mind, I do not envisage the situation where our TI users are swamped by a crowd of Johnny-come-lately's equipped with the latest AT PC and suchlike gee-whiz technology.

Rather, I feel we could possibly offer something to users of other true "orphans" such as the VIC, Tandy Co-Co, etc. (Yes, there are still some of those in use). I feel that such people would be more likely to upgrade to TI equipment than to "take us over". Certainly more discussion on these matters is warranted. The other questions raised by Lou and Company, i.e. incorporation, are also worthy of serious consideration, especially as the Wollongong people take a very active part in the running of our group. o

Building an EPROMdisk

by Craig Sheehan

An EPROM Disk is virtually the same as a standard RAMdisk, except that EPROMs replace some of the RAM chips. EPROMs are similar to RAM chips, but are read only. Once data has been placed on an EPROM, using an EPROM programmer, data can only be changed by erasing the entire EPROM, by exposing it to ultra violet light, and starting the process again. For the average user who installs EPROMs on their RAMdisk, this means that they will be unable to change the contents of any EPROMs purchased.

Although writing to EPROMs is not possible for most users, there are two main reasons for using them. Firstly, EPROMs are cheaper on a per kilobyte basis, and most users have programs installed on their RAMdisk which are never altered; for example programs like Funiwriter, after it is configured. TIsHUG now possesses the hardware and software necessary to program EPROMs, with just about anything which can be run from a disk or RAMdisk.

It is envisaged that TIsHUG will sell Commercial and Freeware/Shareware software on EPROMs, for use with this system. The cost of purchasing software in this format will be more expensive than in disk format. EPROMs cost about 8 cents per kilobyte, while a double sided double density disk costs about one third of one cent per kilobyte. However, EPROMs cost much less than Static RAM Chips, and are just as fast, which makes them heaps faster than a floppy drive.

The following articie describes how to modify an HRD RAMdisk, using 32K RAM Chips. Your RAMdisk must have an 8K static RAM chip as U11, as an EPROM ROS cannot be used. The RAMdisk must also contain at least 3 32K Static RAM Chips. For RAMdisk users with an all 8K Chip board, who wish to do this EPROM modification, your best option is to build a new RAMdisk. Note, you can run both 32K and 8K chips in the same board together. Before beginning the modification, it must be noted that neither the author, nor TIsHUG, accept any responsibility for damages arising from making this modification.

To carry out any of the following modifications, a set of wire cutters, needle nosed pliers, a fine tipped soldering iron, and a fine flux core solder, will be needed. You will also require hook up wire at most stages. I personally prefer to use solid core wire, usually sold as telephone type cable, at most electronic stockists. This type of wire can be bent permanently into shape, using the needle nosed pliers, and the ends do not fray while soldering.

Modifying your HRD RAMdisk.

The only major modification involves placing a socket in the vacant U10 space. The following description assumes that you have a functional HRD RAMdisk. If you are yet to construct your RAMdisk, do so by following the instruction guide written by Bud Mills, but further changes involving jumper leads, should follow as detailed from this point on. In either case, additional parts required, include one 14 pin IC socket, insulation tape and hook up wire. For those constructing a new RAMdisk, step 1 can be omitted if no jumpers involving U10 have been installed.

Step by Step.

1. Remove all the jumpers leading to or from the vacant U10 socket. Disconnect both ends of the jumpers concerned.

2. On the underside of the RAMdisk board, connect the following jumpers: U1 pin 2 to U10 pin 1, U1 pin 7 to U10 pin 5, and U10 pin 9 to U10 pin 13. Take care to get the pin connections right while the board is upside down. These jumpers are the same as before, except they are placed on the underside, rather than on the component side.

3. On the component side, run a jumper from U2 pin 19 to U18 pin 4. This is most simply achieved by soldering the jumper onto the pins themselves.

4. Using the wire cutters, cut off all the pins except pins 7 and 14 from the IC socket. Ensure that the pins are cut off flush with the body of the IC socket. Cover the cut off pins with a strip of insulation tape.

5. Insert the socket into the U10, and solder the two pins into place.

6. Check everything carefully. Ideally you should ask someone else to do this for you. Ensure that the jumpers are wired between the correct pins, and that there are no solder bridges. Check that the orientation of the IC socket is correct.

7. Test your RAMdisk just as you would if it were newly constructed.

This is all you need to do. If you built your own RAMdisk, the above modification should be very easy. If you are not confident, ask someone else to help you, at a monthly or project meeting.

Installing the EPROMs - Hardware.

For this stage, you will need to first purchase EPROMs which contain the programs that you want. There is no point in purchasing blank EPROMs, as these will have no useful information. For each EPROM, you will need a 1K resistor and some hookup wire. In addition an IC, a 74LS08 (quad input AND gate: low power Schottky) will be required for every four EPROMs, or part thereof.

Before doing any soldering, you will need to determine the next free chip select pin. The following table shows the order of the chip select pins, and their location on the printed circuit board.

Select number	Chip location	Chip select pin location
1	U17	U17 pin 20 or U2A pin 1
2	U8	U8 pin 20 or U2A pin 2
3	U16	U16 pin 20 or U2A pin 3
4	U7	U7 pin 20 or U2A pin 4
5	U15	U15 pin 20 or U2A pin 5
6	U6	U6 pin 20 or U2A pin 6
7	U14	U14 pin 20 or U2A pin 7
8	U5	U5 pin 20 or U2A pin 8
9	U13	U13 pin 20 or U2A pin 9
10	U4	U4 pin 20 or U2A pin 10
11	U12	U12 pin 20 or U2A pin 11
12	U3	U3 pin 20 or U2A pin 13
13	Top U13	Jack pin 13 or U2A pin 14
14	Top U14	Jack pin 14 or U2A pin 15
15	Top U15	Jack pin 15 or U2A pin 16
16	Top U16	Jack pin 16 or U2A pin 17
17		U2B pin 1
18		U2B pin 2
19		U2B pin 3
20		U2B pin 4
21		U2B pin 5
22		U2b pin 6
23		U2b pin 7
24		U2B pin 8
25		U2B pin 9
26		U2B pin 10
27		U2B pin 11
28		U2B pin 13
29		U2B pin 14
30		U2B pin 15
31		U2B pin 16
32		U2B pin 17

Each RAM chip uses one chip select pin, while each EPROM requires two chip select pins. These chip select pins must be filled, starting from number 1, with no gaps. The first unused chip select pin is the next free chip select pin.

A couple of examples should clarify the above. If the RAMdisk has six RAM chips, then the next free chip select pin is number 7, which corresponds to U2A pin 7. The first six chip select pins are connected to the RAM chips.

Consider a system with 3 RAM chips and 4 EPROMs. In this case, three chip select pins are used for the RAM chips, and eight for the EPROMs; a total of eleven. The next free chip select pin is number 12, which corresponds to U2A pin 13 on the printed circuit board. It should be noted that chip select pins are used in the order, RAM chips first, and then the EPROM chips in order. Chip select pins 1 to 3 are connected to the RAM chips, and chip select pins 4 and 5 are connected to the first EPROM, and so on through to chip select pins 10 and 11, which are connected to the fourth EPROM.

One last example: A RAMdisk with 12 RAM chips and 8 EPROMs, has chip select 29 (U2B pin 14) on the next free chip select pin.

If you are still having difficulty understanding which is the next free chip select pin, it would be advisable to obtain some help while installing your EPROMs.

You will also need to know the the input and output pins for each of the gates on the 74LS08.

Gate number	Input pins	Output pin
1.	1 and 2	3
2.	4 and 5	6
3.	9 and 10	8
4.	12 and 13	11

The last point that needs to be noted is that the EPROMs will be sold in a set containing between one and five EPROMs, depending on the size of the programs. It is important that the following set of instructions is carried out for the first EPROM in the first set, and so on until the last EPROM in the set is installed.

Installation Instructions.

1. Bend out pins 20, 27 and 28 of the EPROM. Either place the EPROM in an empty RAM socket, or piggyback it onto a RAM chip, and solder it into place. If you wish, you could piggyback 28 pin sockets onto the RAM chips instead of the actual EPROM.

2. Solder the 1K resistor between pins 20 and 28 of the EPROM.

3. Connect a piece of hook up wire from pin 28 of the EPROM, to the 5 volt pin of any IC, which is not a RAM chip. Suitable pins include U2A or U2B pins 24, U9A and U9b pins 16, U21 pin 20, and pin 28 of any correctly installed EPROM.

4a If there is no 74LS08 in the socket at U10, bend out all the pins except pins 7 and 14. Insert the IC into the socket, and proceed to step 5.

4b If all gates on the 74LS08 have been used (that is, there are no spare inputs or outputs), piggy back another one on to the top of the last one, and solder pins 7 and 14 of both chips together.

5. Choose any free gate on the 74LS08. Connect a piece of hook up wire from the output pin of that gate to pin 20 of the EPROM (see table).

6. From the next free chip select pin (see table), connect two pieces of hook up wire; one to pin 27 of the EPROM, and the other to one of the input pins of the gate you are using.

7. From the chip select pin that follows the one you just soldered two pieces of hook up wire to, connect a piece of hook up wire to the remaining gate of the 74LS08 which you are using.

8. Repeat the process for all the remaining EPROMs.

If the above steps are not clear, read the following example:

You have a RAMdisk with six RAM chips and no EPROMs. The U10 socket is empty, and you have a set of two EPROMs you wish to install.

First bend out pins 20, 27, and 28 of the first EPROM and piggyback it on top of one of the RAM chips. Solder a 1K resistor between pins 20 and 28. Using a piece of hook up wire, connect pin 28 of the EPROM to U2A pin 24 and pin 3 of the 74LS08 to pin 20 on the EPROM. Connect U2A pin 14 (the next free chip select pin) to 74LS08 pin 1 and pin 27 on the EPROM. Connect U2A pin 15 (the following chip select pin) to 74LS08 pin 2.

Now install the second EPROM in the sequence; Piggyback the EPROM as before and, and install the resistor between pins 20 and 28. Connect EPROM pin 28 to U2A pin 24, and 74LS08 pin 6 to EPROM pin 20. (We are now using the second gate.) Connect U2A pin 16 (the next free chip select pin) to 74LS08 pin 4 and EPROM pin 27. Connect U2A pin 17 (the following chip select pin) to 74LS08 pin 5. The job is now completed.

If you are having difficulty understanding the above instructions, ask for some help while you install your first EPROM. Once you have installed a couple, it will become clear.

Installing the EPROMs - Software

Having physically installed the EPROMs on the printed circuit board, all that remains is to install it, by telling the RAMdisk about it. This is achieved through a program which is almost as simple as loading the ROS into the normal RAMdisk.

First you put the RAMdisk into the Peripheral Expansion Box. Load the CFG program from ROS 7.3 (it may work on the later versions of the ROS, but it has not been tested), and initialize the RAMdisk as normal. At this stage, one of the EPROMs will be recognised and listed. Make sure that all of the RAM is configured into either the primary partition, and the rest into one or more secondary partitions. It is important to leave at least as many secondary partitions as there are sets of EPROMs. Since there are at most 9 secondary partitions, after all the RAM is allocated to the primary partition, nine is the absolute maximum number of sets of EPROMs which can be installed on any one RAMdisk.

After you have saved the ROS to the RAMdisk, load the ESI program, which will be available through the club shop. It is loaded through Funlwriter. The first step is to go to the loaders and select number 5, Scriptloader. Change the disk number to 1, and load the file "LOADINSTAL". When it comes up, all that needs to be done is to enter the appropriate switch number of the RAM disk CRU address (that is, the RAMdisk with the EPROMs). Provided everything is well, a catalogue of all the RAMdisk partitions and their sizes will be shown, together with the EPROM partitions.

In Conclusion.

Provided all the above has been carried out, accessing data on EPROMs is a simple matter. Because EPROMs are in the Secondary partitions, only one set of EPROMs can be accessed at any one time on any one RAMdisk. To access an EPROM partition, use the volume name format, that is, DSK.volume-name.filename. For example, the EPROM has a volume name or disk name "GAMES", and you wish to run a program called "RACING". Simply type in "DSK.GAMES.RACING". Once an EPROM has been accessed in this way, the disk number assigned to the secondary partition, can be used.

There is virtually no limit on the number of EPROMs that can be installed on any one system. Although each individual RAMdisk can have nine sets of EPROMs (of between one to five EPROMs), you can have more than one

continued on page 23

Loading Programs and Files under Extended BASIC

by Ross Mudie

1. INTRODUCTION

This article discusses various methods of loading programs and files under Extended basic. It is intended for people who want to write their own programs. Most users are familiar with loading programs from tape and disk but not all understand the principles of using menu programs in extended BASIC and chaining programs together.

In Extended Basic if a disk system is connected to the computer and when extended basic is first selected, after the initial master screen, extended basic automatically checks disk drive 1 for a program named LOAD. If a LOAD program is found it is loaded into memory and run. The LOAD program can be written to perform what ever the writer wants; this article discusses a few of the common uses of LOAD programs.

This article does not explore the system used in the Horizon Ramdisk which can avoid the TI colour bars master screen. Extended Basic also contains a sub-program named CALL LOAD which can be used to load assembly files from disk or "poke" information into CPU RAM locations.

a) The LOAD program can be specially set up already containing the names of available programs on the disk which can be displayed in a menu list. The required program can then be selected from the menu with automatic load and running.

b) The LOAD program can search the disk drives in the computer for other Basic and Extended Basic programs and build up a menu of programs. The program can "filter out" the files etc and can then allow selection of the required program with automatic load and running.

c) The LOAD program can just be a 2 liner which loads very quickly and displays on the screen that the required program is being loaded. This may load the single dedicated program.

d) A program can load data from a file. A data file can be created from an extended basic program or an EDITOR program. Records from the data file can be loaded into variables or arrays for use in the program. When a program becomes tco big to fit in memory along with the data which is generated or used by the program, it is often practical to break the program up into parts. The separate programs which can be chained together can use data from a common file.

e) Extended Basic programs can load assembly language routines which can be linked into from the extended basic program.

f) When many of the TI and third party games were "cracked" from cartridge to disk format they were in Editor/Assembler option 5 format. These could be loaded by the Editor/Assembler using option 5 or loaded under Extended Basic using "an Option 5" loader.

g) Extended Basic programs do not necessarily have to be just Extended Basic. Programs may be assembly language inside an Extended Basic program with special inbuilt assembly language loaders which may change the loading format. The "Implanted Assembly" Extended Basic programs can be loaded from cassette tape or disk drive and chained together.

h) The program named LOAD does not necessarily have to be a loader, any program can be named LOAD and it will load and run after extended basic is selected.

2. PROGRAM EXAMPLES

These program examples are the minimum no frills versions to show the principles.

a) LOAD WITH PROGRAM NAMES IN THE LOAD PROGRAM.

```
100 ! SAVE DSK1.LOAD
110 DISPLAY AT(6,7)ERASE ALL BEEP:"LOAD EXAMPLES
    1.":"PRESS 1. FIRST PROGRAM": : " 2.
    PROGRAM 2" : :" 3. THIRD PGM"
120 CALL KEY(5,K,S):: IF S=0 THEN 120
130 IF K<49 OR K>51 THEN 110
140 ON K-48 GOTO 150,170,190
150 DISPLAY AT(20,1):"Running DSK1.FIRST_PGM"
160 RUN "DSK1.FIRST_PGM"
170 DISPLAY AT(20,1):"Running DSK1.PROGRAM_2"
180 RUN "DSK1.PROGRAM_2"
190 DISPLAY AT(20,1):"Running DSK1.THIRD_PGM"
200 RUN "DSK1.THIRD_PGM"
```

Line 100 is used to name the program and it can be used to save the program in extended basic by pressing 100 <FCTN> X <ENTER>, <FCTN> 8 (Redo), then edit off the 100 ! and press <ENTER>. This leaves the SAVE DSK1.LOAD which saves the program.

Line 110 Clears the screen, beeps and displays the menu list.

Line 120 Waits for a key to be pressed.

Line 130 Tests the key pressed and rejects any invalid key.

Line 140 Adjusts the ASCII key value in variable K to a value into the numeric range of 1 to 3 and branches to the required line number.

Lines 150 to 200 are in pairs for the required program. The first line of the pairs shows on the screen what is about to happen. The second line of the pairs loads and runs the required program. These lines with RUN contain QUOTED STRINGS which were included in TI's planned facilities for Extended Basic. (This function does not allow RUN to be used with the contents of a string variable, only a direct quoted string). If the program cannot be found then an error will be given.

b) LOAD WHICH SEARCHES THE DISK DIRECTORY.

There is nothing special in this program, and it is based lcosely on the program in the TI Disk Memory book page 41. It reads the catalog on the disk, saving only the program names in an array and printing these on the screen. The program only loads the first 21 program names. This limit was set to keep the program simple, (just one screen). When a program is chosen its name is combined with the characters "DSK1." and the computer shows what it is about to do. TI Extended Basic is unable to RUN the contents of a variable, it will only run a quoted string as seen in the previous example, so another technique is required. The last line in a program is easy to find, so the last line is set up with a dummy program name in a format which reserves the maximum possible required space. The program then finds the location in memory of the last program line and byte by byte, pokes first the length of the required program name, then the device and program name, then a zero to finish off the program line in the normal way that extended basic is stored. Once this is complete, the Extended Basic program has actually modified itself and the modified line is then run.

```
100 ! SAVE DSK1.LOAD
110 DIM PROGRAMS$(20)
120 CALL INIT
130 DISPLAY AT(3,1)ERASE ALL:"SEARCHING DISK
    DIRECTORY"
140 OPEN #1:"DSK1.",INPUT,RELATIVE,INTERNAL
150 INPUT #1:A$,D,S,K
160 PRINT "Disk name ";A$:"Size";S;"Free";K;"Used";S-K
170 C=65
180 IF C=86 THEN 210 ELSE INPUT #1:A$,A,J,K
```

```
190 IF ABS(A)=5 THEN PRINT CHR$(C);" ";A$ ::
    PROGRAMS$(C-65)=A$ :: C=C+1 :: GOTO 180
200 IF A$="" THEN 210 ELSE 180
210 DISPLAY AT(24,1)BEEP:"Press letter of choice"
220 CALL KEY(3,K,S):: IF S=0 THEN 220
230 IF K<65 OR K>=C THEN 210
240 P$="DSK1."&PROGRAMS$(K-65)
250 DISPLAY AT(24,1):"Load & Run ";P$
260 CALL PEEK(-31952,A,B)
270 CALL PEEK(A*256+B-65534,A,B)
280 POKE_ADDR=A*256+B-65534
290 CALL LOAD(POKE_ADDR,LEN(P$))
300 FOR I=1 TO LEN(P$)
310 CALL LOAD(POKE_ADDR+I,ASC(SEG$(P$,I,1)))
320 NEXT I
330 CALL LOAD(POKE_ADDR+I,0)
340 RUN "DSK.DISKETNAME.PROGRMNAME"
```

Line 110 specifies a single dimension array of 21 elements, (the elements start counting from zero).

Line 120 loads the assembly language support routines from the extended basic module into the low memory. This will be required for later use by CALL LOAD.

Line 130 writes something on the screen to show what is happening.

Line 140 opens file number 1 to the disk directory area.

Line 150 reads the disk name into A$, a dummy variable D because there is nothing in this variable, the disk size in sectors into S and the amount of free space on the disk into K.

Line 160 prints the details of the disk name header and disk parameters.

Line 170 specifies a starting value for C which controls the letters printed in the menu selection. By using the letters of the alphabet from A to U a single key press could be easily utilised.

Line 180 Tests for a full screen and if full goes to the choice line. If the screen is not full it gets the next program name record. The program or file name is in A$ and the type of program or file is in variable A. If the value in A is 5 then it is a memory image program; if it is -5 the program is "write protected" on the disk. Unfortunately there are 2 types of "memory image" programs, they are BASIC/Extended Basic or Assembly Option 5 which can be loaded by the Editor/Assembler cartridge. (If you attempt to load an option 5 assembly program without a special loader from Extended Basic the program will not load or run and an error message will be issued).

Line 190 uses the ABSolute function to wipe off a minus from the value in A which would occur if the program was write protected on the disk. If a 5 was found then the letter for the counter (C) value is printed on the screen followed by the program name on the screen. The program name is stored in the array PROGRAMS$, the counter is incremented and the program goes to line 180 to check for the next program or file name.

Line 200 is reached if a file name is encountered or there are no more names in the disk directory. It goes back to 180 if there are still more names to be searched.

Line 210 to 230 prompt for a response, accept a single key press and check that any input is valid in range. The use of key unit 3 in the CALL KEY will return upper case only, regardless of the position of the alpha lock key.

Line 240 concatonates (joins together) the characters "DSK1." and the name of the chosen program in the variable P$, since only the program names and not the device name were stored in the PROGRAMS$ array.

Line 250 Shows what is about to occur.

Line 260 gets the address of the end of the line number table.

Line 270 looks at the last entry in the line number table and gets the address of the last line in the program in variables A and B.

Line 280 calculates the address of the string in the last line and places the value to represent the address in the variable POKE_ADDR. The line in its "crunched" form consists of an overall length byte 29, the token for RUN which is the value of 169, followed by 199 to indicated that a quoted string follows. Then follows the length byte of 25 for the string and the actual string followed by the terminator value of zero.

Line 290 places the length the new string in the string length byte of the quoted string in the last line of the program.

Line 300 sets up a FOR-NEXT loop to write the new number of bytes required in the string.

Line 310 places the characters from the name in P$ in the appropriate memory locations. As the value of I is incremented it moves along the string and the memory locations one at a time.

Line 320 controls the loop.

Line 330 places the terminating 0 after the modified contents of line 340. The value of I is one greater than the TO value in the FOR-NEXT loop and this added to the POKE_ADDR points to the appropriate location for the terminator.

Line 340 is now executed with its modified contents.

c) THE 2 LINER LOAD

Some large Extended Basic programs may take a long period to load and then there is a delay for the prescan. To show that something is happening rather than a computer lock-up this very short load program clears the screen and puts up a brief message. The techniques utilised in this program have been previously discussed.

```
100 ! SAVE DSK1.LOAD
110 DISPLAY AT(10,1)ERASE ALL:"Loading.." :: RUN
"DSK1.LOAD-B"
```

d) LOADING DATA FILES

By saving data in a disk file the program size can be limited. The same program can use multiple sets of data from disk or a number of programs can use a common data file.

```
130 DIM D$(20)
140 OPEN #1:"DSK1.DATAFILE",INPUT,DISPLAY,VARIABLE 80
150 FOR I=1 TO 20
160 LINPUT #1:D$(I)
170 NEXT I
180 CLOSE #1
```

Line 130 reserves string space for 21 elements in the single dimension array D$

Line 140 opens file number 1 in input display mode with variable length records of 80 characters per record maximum to be read from disk drive 1, file named DATAFILE.

Line 150 sets up a control FOR-NEXT loop to read 20 items.

Line 160 sequentially reads the records from DATAFILE into the elements of the array D$. The control variable I specifies which element of D$ each successive record from the DATAFILE is placed into.

Line 170 completes the For-Next loop.

Line 180 closes the disk file.

By using RUN "DSK1.PROGNAME" one program can chain to another and then by loading the data from the disk file the same data pool can be used by both programs. When RUN is used all variables are reset to zero or null so everything has to be re-loaded. Before running the next program if any modified data is saved then the modified data is available to the next chained program. Here is an example of saving.

```
240 OPEN #1:"DSK1.DATAFILE",OUTPUT,DISPLAY,VARIABLE 80
250 FOR I=1 TO 20
260 PRINT #1:D$(I)
270 NEXT I
280 CLOSE #1
```

This time the file is opened in OUTPUT mode and the contents of D$ are PRINTed out to the file on disk drive 1 named DATAFILE.

e) LOADING LINKED ASSEMBLY LANGUAGE

All of the programs involving assembly language require that a 32K memory expansion is connected to the computer. The standard format of providing Linked Assembly routines is to write a Source file with the Editor/ Assembler EDIT program. When a source file is saved it is in DISPLAY VARIABLE 80 format. To make the source file usable it must be assembled into machine executable instructions. The process of assembly is achieved with the Editor/Assembler package ASSEMBLER program. The assembler creates an OBJECT file which is in DISPLAY FIXED 80 FORMAT. Extended Basic can only load the D-F 80 files providing that they are in non-compressed format.

```
110 CALL INIT
120 CALL LOAD("DSK1.OBJECT")
    .
    .
200 CALL LINK("ENTRY",A,B$)
```

Line 110 must be provided to place the assembly support routines in the LOW RAM from the Extended Basic cartridge.

Line 120 demonstrates how to load an assembly language object file into memory. The Extended Basic "normal" method loads the assembly language into the LOW RAM. If the code is relocatable, (in other words no specification was made of absolute location of the code in memory), then the loader just loads it in the first free space above the area used by the INIT support routines. The loader also sets up a DEF table at the top end of the LOW RAM. The DEF table is a directory of the specified link names. Each DEF table entry contains a 6 character name plus a 2 byte address which specifies where that assembly routine starts in memory.

Line 200 is typical of a LINK to an assembly routine. The CALL LINK subprogram in this case would search the DEF table for the name ENTRY and if it was found would run the assembly routine. The variables A and B$ are typical of the method of passing parameters between the Extended Basic and the assembly program. Up to 16 variables, values or strings can be used in the arguments contained in the link list. Using CALL LINK is very similar to passing parameters to and from SUB PROGRAMS except an assembly CALL LINK can be much smarter than an Extended Basic sub program.

Using linked assembly routines, functions which are not available in Extended Basic can be incorporated in an Extended Basic program. Linked assembly is also a good way to learn assembly programming techniques. The utility programs provided to transfer values and strings between Extended Basic and Assembly and back are well documented in the Editor/Assembler manual.

Assembly routines can also be loaded using successive CALL LOAD statements in the normal program lines of an Extended Basic program. The method is quite slow and cumbersome but for short assembly routines it is often the easiest method. Use of this method saves having a separate assembly object file which has to be loaded from disk.

```
110 CALL INIT
120 CALL LOAD(8194,36,252,63,248,"",16376,67,79,77,80,
    65,67,36,244) ! Pointers & DEF table area.
130 CALL LOAD(9460,4,32,32,24,0,0,4,91) ! Program
```

This routine which was fully described in the July 1986 TND. It needs a 32K memory expansion to be used and its function is to force a garbage collection when the programmer wants it, not when the computer has run out of string space and has to do it. (This is why the computer occasionally pauses in the running of an Extended Basic or Basic program). To use this little routine in a program just use CALL LINK("COMPAC") at a convenient time rather than when the computer has to do it.

f) LOADING CRACKED GAMES IN "OPTION 5" FORMAT. When games modules became available on disk, the disk contained an extended basic LOAD program, an option 5 loader and the game programs in E/A option 5 format. The function of the option 5 loader was to move the character tables, colour table etc around to where they are needed for the new program which is about to be loaded and to load the required program files into memory then commence execution of the program. These same programs could of course be loaded using the option 5 loader in Editor/Assembler and specifying the first file name of the required program.

g) LOADER TECHNIQUES IN IMPLANTED ASSEMBLY PROGRAMS.

You will not find anything about the techniques of implanting assembly programs inside an Extended Basic program in the TI books. Several of these techniques were developed by members of the various user groups around the world, the method described here was developed by George Meldrum of TISHUG's Illawarra Regional Group.

Single part programs are easy to load because they load into memory in one operation. The implanted program is actually a few Extended Basic lines, the implanted assembly language application program and an assembly language routine to "change the environment" in the computer. The changing of environment is necessary because the character definitions, colour table and pattern table are in different memory locations for Extended Basic and Assembly. The assembly language routine must also move the implanted program to the required memory area and then commence execution of the program.

An assembly program implanted in an Extended Basic can be loaded with OLD, saved with SAVE on cassette tape, disk or RS232. Extended Basic RUN is also used, here is the Extended Basic part of an implanted program.

```
100 ! SAVE DSK1.GAME
110 CALL INIT
120 CALL LOAD(8196,63,248,"",16376,71,65,77,69,32,32,
    255,156)
130 CALL LINK("GAME")
```

Line 120 places the address of the start of the DEF table in the 2 bytes commencing at address 8196. The DEF table entry is placed in the 8 bytes starting at 16376.

Line 130 LINKS to the assembly routine which changes the environment using the DEF table name and address loaded by line 120.

The loading technique used for multi-part implanted programs is absolutely ingenious! Once the first part of the program is loaded and runs it tests to see if the

continued on page 14

Using Teac 3.5" Micro-floppies

by Ben Takach

The 3.5 inch diskette format presents many advantages compared to the 5 and one quarter inch drives.
These are:

- smaller physical size
- higher storage capacity
- lower power requirements
- files on the disk are better protected against loss due to fingerprints etc. on the exposed magnetic media.
- further miniaturisation is likely due to the popularity of battery operated laptops.

Recently, I have purchased a 3.5" micro-floppy, a TEAC type FD-235HF 217U. Using the Myarc disk control card with the 80 track per side eprom fitted, disks will format to 2880 sectors. This is 720Kb of storage capacity per diskette. Admittedly the cost of the 3.5" disks are higher, one can not purchase these under $14 per 10 disks. However the higher cost is well compensated by the convenience of the large capacity. One can work with a single disk without the need for disk swapping during a session.

The drive is extremely compact. Its mass is a mere 370g. The physical size is 102x150x25mm. The electronics are accommodated on two densely populated surface mounted P.C. boards. Some of the tiny capacitors and resistors can hardly be seen without a magnifying glass. It is absolutely noiseless during read-write or formatting operation. Three of these drives will fit comfortably into the P.E. box.

However there is a catch. Obviously the unit was designed to cater for the IBM and its clones, thus the drive select-link options provide for the selection of drive 1 or drive 2. I could not tell how to configure it to work drive 3. The signal connects through a 34 pin IDC male Header connector. The mounting hardware also includes an IDC socket/34 strip card edge adaptor. This is fitted with a jumper to enable or disable line No.34, the ready line. This jumper is normally set to enable line 34. After using the drive in a makeshift fashion for a few days, I decided to put it in a box with its own power supply. There is no reference on the units' name plate as to its power consumption, thus it had to be measured. I used two ampere meters to measure the current of the 5V and the 12V supply. Much to my surprise the meter monitoring the 12V supply indicated zero milliamps. I suspected it had a fault in its current range. A substitute meter also failed to indicate any current reading. A close examination of the P.C. board with a magnifying glass indeed confirmed that the drive did not require 12V supply at all, there is no track connected to pin No.1 of the power connector. The drive works off a single 5V supply. Its stand-by current is 80mA at power up which drops to almost zero when the drive is not in use. During read/write or formatting it consumes 380mA. In other words it can be powered from 5V/500mA regulated plug pack as a stand alone drive.

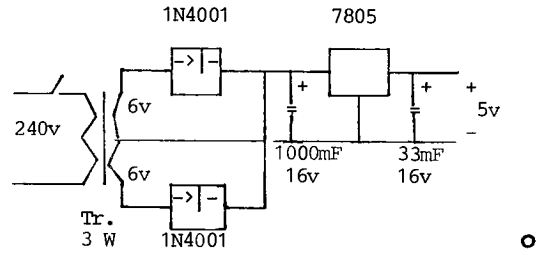
An optional mounting kit is available, to make a retro-fit into existing equipment more easily. It consists of a zinc plated steel cradle with a moulded bezel to suit a 5 1/4" half height drive space, a 4 pin standard power-connector adaptor (the drive is fitted with a miniature 4 pin connector), a packet of 3mm mounting screws and a pair of moulded slide rails to fit a standard P.C. enclosure.

The drive including the mounting kit was obtained from AVO electronics located behind the Gore Hill branch of Jaycar for $150.

I am delighted with this miniature drive, its high capacity compliments my two 5 1/4" units adding comfort, speed and versatility and using only the bare minimum of bench space. I have also tested it with a home-made power supply using a small 6V+6V power transformer. The circuit is simple and straight forward, it is depicted below.

Any complaints? Well yes, there is one. The drive is so light that one has to use both hands to insert or eject a diskette, otherwise it is pushed off the table.

Power supply for the TEAC 3.5" micro-floppy.



TI Hacker's Lament

by Ben Takach

Once upon a time, - Well starting a story with those magical words opens the flood gates and conjures memories. Alas
one cannot turn the wheel of time back; we cannot bring back the past. We are destined to thread the never-ending path of progress, even if one can only afford a 99/4A in the way of tomorrows' progress.

And this is exactly what my opening phrase is all about. As I was saying, once upon a time I had a TI plus a tape deck and a B/W TV set. Many things I hoped to do, but my every wish was countered with the well-worn phrase:"You have to expand your system". Well through the years, expand my system I did, until it is so expanded that even its' maker would not recognise it. Now I am supposed to be able to do absolutely everything.

Sadly, I have to confess it is not the case. At about the rate of my expansion more and more things became impossible.

My Eprom burner would not tolerate the Myarc expansion cum Ram disk card, and I almost had to resort to the use of a stock whip to force a not exactly perfect co-existence between the said Myarc card and the Gramcard. Yet both of these are meant to make ones life more pleasant!

Then came the conflict of disk formats. I was content with 16 sectors on my DSDD disks until it became clear that this format is the ultimate way to protect a disk. Absolutely no-one outside the Myarc community could read one single line of it. Then as progress led me to proceed further up (or down?) the path of progressive expansion, I noted that disks purchased from the club would not run on my system. I humbly beg for forgiveness, but at one stage my suspicious mind suspected a corrupt club practice of selling blank disks with fancy labels at inflated prices. Thanks to the admirable patience of Terry, he kept exchanging the disks without a single word of complaint or rude remark.

Now that my system is fully expanded, almost nothing will work on it. The P.E. box has one empty slot, where the Pascal card used to spend its time idling away uselessly, because of the same incompatibility problem, until it was turfed out to reduce the unnecessary heat generation - and one day when its place is filled, then the system expansion will be complete. From that day on absolutely nothing will work in my system.

continued on page 22

Extended BASIC Tips #15

by Jim Swedlow, CA USA

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

RE-MAPPING THE KEYBOARD

You normally see CALL KEY(0,K,S). There are five other values for the first variable, the key unit. They remap your keyboard:

0 Keeps the keyboard in the same mode as the last time a CALL KEY was executed. If this is the key unit on the first CALL KEY in a program, you stay in 4A mode.

1 Splits the keyboard into two 2 smaller boards. Good for games.

3 Remaps the keyboard as a 99/4 (no A). CONTROL and lower case are inactive.

4 Pascal mode.

5 Remaps to the 99/4A mode with lower case and CONTROL active.

When you use 3, regardless of the position of SHIFT and ALPHA LOCK, all alphas return as upper case. The problem is that this condition continues until you do a CALL KEY with a different key unit. Try this:

```
10 CALL KEY(3,A,B) ::
PRINT : :"Key Unit is 3"
20 INPUT A$ :: PRINT A$
30 CALL KEY(5,A,B) ::
PRINT : :"Key Unit is 5"
40 INPUT A$ :: PRINT A$ ::
GOTO 10
```

With ALPHA LOCK up, try inputting lower case letters and see what happens.

A key unit of 3 is very useful to make sure that only upper case alphas are caught by the CALL KEY. If you want lower case later in an INPUT or ACCEPT, however, you must remap the keyboard with another CALL KEY.

USER FRIENDLY/USER PROOF

When programming, you want your program to help the user. You also want to keep the user from crashing your program. Remember, the user will do most any fool thing. One area of vulnerability is inputting information thru INPUT and ACCEPT statements.

Lets say you want an integer between 1 and 9. Here are a number of ways you can input that number.

INPUT F can cause two problems. First, any number will be accepted. Second, if your user inputs anything but a number, you get: "WARNING: INPUT ERROR IN nnn TRY AGAIN".

This destroys your screen, scares your user and looks bad too.

ON WARNING can help. If you add ON WARNING NEXT the warning message will be suppressed but any number will be accepted. This coding is probably the best that can be done with INPUT F:

```
10 ON WARNING NEXT
20 INPUT F :: IF F<1 OR F>9 OR
F<>INT(F) THEN 20
```

Bad values of F (0, 3.1, etc) will still cause the input prompt to be repeated and mess up your screen but you will get a good value in the end.

ACCEPT AT has a number of features that will help. With SIZE and VALIDATE you can avoid some problems:

```
10 ACCEPT AT(5,10)SIZE(1)
VALIDATE(DIGIT)BEEP:F
```

Your user can still goof you up two ways. Zero is acceptable input and inputting a null will do strange things. SIZE(1) means no bigger than one character but it can be smaller!

ACCEPT F$ will help solve some more problems. Strings will be accepted. This coding is about as user proof as you can get:

```
10 DISPLAY AT(5,1):"1" 20 ACCEPT AT(5,1)SIZE(-1)
VALIDATE(DIGIT)BEEP:F$ ::
IF F$="" THEN 20 ELSE
F=MAX(VAL(F$),1)
```

By making the SIZE value negative, whatever is on the screen at (5.1) will be the default value if ENTER is pressed.

About the only way I have found to mess this up is to have a non-numeric sitting at (5,1). VALIDATE works ONLY on which key is pressed, it assumes that you know what is on the screen!

MEMORY SAVERS

A variable name takes only its length in memory. <A> takes one byte while <MASTER_DEVICE> takes 13 bytes and <A$> takes two bytes.

A number used in a program line, however, takes the number of characters plus two bytes. For example, the number 2.13 would take six bytes of memory and the number 1 would take three bytes.

Strings also take the number of characters in the string plus two. "He won" takes eight bytes.

Some suggestions:

--Use the shortest possible variable and sub-program names.

--If you use a given number often, assign it to a variable and use that instead.

Figure this way: A=2.13 takes eight bytes (one for A, one for = and six for 2.13). Each time you use A instead of 2.13 you save five bytes. Therefore, after two substitutions you are conserving memory.

--Look for places where you can replace numerics with variables. This line:

```
10 C=0 :: INPUT A(0) :: PRINT A(0)
```

Would take four less bytes if done this way:

```
10 C=0 :: INPUT A(C) :: PRINT A(C)
```

Be sure, however, that you only do this when the variable must be the number you intend it to be!!!!!

SPEED

Longer variable names slow program execution. I ran this program with progressively longer variable names substituted for <C>:

```
10 C=0 :: FOR I=1 TO 1000 :: A=C :: NEXT I
```

I ran each three times and averaged the results. Here is what I found:

LENGTH AVERAGE OF SUB- RUN TIME STITUTE (seconds) FOR <C>

1 7.74 2 7.72 4 7.90 6 8.12 9 8.46 10 8.56 15 9.02

continued on page 5

TI Bits * Number 16

by Jim Swedlow, CA USA

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

ON DISKS AND DRIVES

A while back the Disk Doctor attended one of our meetings. He had a number of interesting things to say.

* Do not clean your drives until you need to. Your system will tell you when it is time - you will have trouble reading disks.

* When you do clean your drive, use any brand name commercial disk drive cleaner and follow instructions.

* If this fails, you need to have your drive cleaned professionally. If you want to try yourself and you have a double sided drive, be careful with the second read/write head. It is very, very easy to bend the bracket to the point that the head must be realligned.

* He has tested the amount of residue left on heads with brand name disks ($1.00 + each) and the cheepies ($0.25 or so). He found no difference. This does not mean that they are of equal quality, only that the cheepies are not dirtier.

* He opposes flippies for single side users. His point is that when you flip the disk and it runs backwards in its cover, dirt is loosened and spun into your drive.

* His overall advise is the first rule of engineering: If it is not broke, do not fix it.

SOME MORE THOUGHTS ON BACKING UP DISKS

Over the years I have mentioned the importance of backing up your disks. Simply put, disk drives eat disks. On the first weekend of October, I was working on some letters. This was the weekend where the temperature was well over 100 degrees. I blew both my word processing disk and my data disk.

I had a backup of the word processor, but it was not configured. That night, after it cooled down a bit, it took me about half an hour to recreate a working disk. The data files were simply lost.

The moral? Keep two back ups of your program disks. One of the disk as you received it (the master) and one of your configured working disk (back up working disk). Do not forget to back up your data disks every now and then. This will save you time and aggravation next time your drive gets hungry.

TI WRITER's INCLUDE FILE

One of TI Writers nicer features is Include File (.IF). It has a few limitations, but it extends TI Writers capabilities.

TI Writer cannot work on large files. No books in one file here. As you reach the size limit, the time it takes to load and save files increases markedly. Include File to the rescue.

Suppose your have written two chapters of your next book. Your named your files CHAPTER1 and CHAPTER2 (very original). At the end of Chapter 1 (the very last line), add this:

.IF DSK1.CHAPTER2

Name CHAPTER1 for the Formatter and it will print both chapters. All the formatting commands you set for Chapter 1 will be used when Chapter 2 is printed, so you do not have to restate the margins and such.

Ah, you finish Chapter 3. No problem. At the end of Chapter 1, add another line:

.IF DSK1.CHAPTER3

You cannot do this at the end of Chapter 2, as you cannot chain these commands. Also note that you must specify the drive number (DSK1 in this case).

I prefer to make a master file (called CHAPTER0) will all of the .IF commands:

```
.IF DSK1.CHAPTER1
.IF DSK1.CHAPTER2
.IF DSK1.CHAPTER3
```

Before (not after) your .IF lines, put in your format, header and footer instructions. Now you have all of your format commands in one place that is easy to find and edit. o

TI Bits * Number 17

by Jim Swedlow, CA USA

OUCH!

Every once in a while I goof and someone catches me. This time it was in TI BITS Number 16 in which I said that you cannot chain Include File (.IF) commands in TI Writer.

That is what the manual says and I believed the written word. Not so, says UGOC President Bob Harper. He is correct. You can end each file with a .IF command for the next file. I still prefer, however, a master file that has all of the .IF commands. (Does this apply to all versions of TI Writer I wonder? Advise Stephen please!)

SOLUTION TO PUZZLE RE POWER OF 7

```
100 P$="" :: N$="1" :: P=0
110 CARRY=0 :: FOR F=LEN(N$) TO 1 STEP -1
120 V=VAL(SEG$(N$,F,1))*7+CARRY
130 CARRY=INT(V/10) :: V=V-CARRY*10
140 P$=STR$(V)&P$ :: NEXT F
150 IF CARRY<>0 THEN P$=STR$(CARRY)&P$
160 P=P+1 :: N$=P$ :: P$="" :: PRINT P;"
    ";N$
170 Z=POS(N$,"777777",1)
180 IF Z<1 THEN 110
```

o

continued from page 11

disk system is attached and if not it it then goes to the cassette system to load the next part. The first part was however loaded into the 32K memory as an Extended Basic program, steps must now be taken not to overwrite it as the second part loads. To prevent overwriting, the program tells the computer that the 32K memory is unavailable. This forces the second part of the program to remain in the Video Display Processor (VDP) RAM. When the program has finished loading the second part, it uses a CALL LINK to execute part of the assembly loader which moves the two parts of the program to the appropriate CPU memory areas. If it is only a 2 part program it also reshuffles the tables for the required environment, whilst if another part is required, execution is returned to Extended Basic and RUN "CS1" or RUN "DSK1.NEXTPART" is used. The last part of a three or four part program is just like a single part program.

This article does not fully cover the very broad subject of "loading" into the TI99/4A. Time and inspiration permitting I hope to write another article on the subject. The inspiration to write this article came out of a series of questions from one of TISHUG's members; the best way to prompt this sort of article is to ask. o

Living with Spiders part 2

by Tony McGovern
Courtesy Hunter Valley 99ers

In this episode we will look at some more aspects of writing programs to co-exist with the Funnelweb system. This time it will be on programs which make extensive enough use of memory that any other code is obliterated. A typical example is a disk manager program, where every spare byte is needed for buffers, let alone the program code. The idea of a dual-mode program is less compelling here, but experience with fully adapted programs shows that it is worthwhile. The code excerpts used to illustrate will be drawn from the modifications made to the Ottawa UG's DM-1000 fairware program to make it Funnelweb-aware.

So the necessary steps are

(1) Extract load-time details

(2) Avoid treading on Atrax R.

(3) Arrange FWB re-entry

The reasons for (1) are much the same as they were in Part I of this saga. The only thing really left under (2) is to avoid trashing the Mailbox unnecessarily. Item #3 is almost more psychological than real. FWB is so easy to reload that going through the title screen is hardly any more work for the user than setting up a direct return. There are benefits in direct reload though such as colour continuity and retaining of the character set.

```
*  FUNNELWEB System block equates

FWENTR EQU  >E006
SVGPRT EQU  >FF14
RDISK  EQU  >FF18
BTFLAG EQU  >FF1A
INCOL  EQU  >FF26
MODFLG EQU  >FF5A
CMSRET EQU  >FF5C
GRMAD  EQU  >FF5E
NAMBUF EQU  >FF62
LDR11  EQU  >FF9C

*  DM-1000 equates

PAB    EQU  >0FE0
VBUF   EQU  >1000

FWFLAG DATA >0
*
INDSK  DATA >0        Keep
FCOLRS DATA >0        in
SVGPTN DATA >0        just
SVGRAD DATA >0        this
SVMODF DATA >0        very
RFDISK DATA >3131     order.

NULL   BYTE >00
FIVE   BYTE >05
HFF    BYTE >FF
HEXAA  BYTE >AA
COLBUF BYTE >F4

       EVEN

*  Initial entry point

FWSTAT LIMI 0
       C    R11,@LDR11
       LWPI MYREG1
       JNE  NOTFWL

*  FWB load path

       SETO @FWFLAG

*  Retrieve colour info

       LI   R0,>380
       BLWP @VSBR
       MOVB R1,@COLBUF
       LI   R1,FCOLRS
       MOV  @INCOL,*R1+

*  Save system details

       MOV  @SVGPRT,*R1+
       MOV  @GRMAD,*R1+
       MOV  @MODFL,*R1+

*  Save current load paths

       MOV  @RDISK,*R1
       MOVB @NAMBUF+4,@DEFDRV
       JMP  MGRST

*  On to DM-1000 regular entry

NOTFWL EQU  $
```

This code illustrates a few more features than would be necessary in a minimal reloader. All that is strictly necessary is to reload UTIL1 or FW as a program file and branch to it normally, but here we are also going to cater for XB/SSSD users who might wish to have only LOAD on their working disk, and not UTIL1 as well. The FWB code is buried in LOAD so that when XB loads LOAD the assembly code is in its correct position in hi-mem. So the strategy adopted with LOAD is to load it into VDP as if it were any old program file, and then to search for the start of the assembly code, before VMBR'ing this to where it belongs. This means that the normal entry code in LOAD which finds return, GROM, and XML addresses is bypassed so these are saved from the previous time FWB was in control. If you were to do this for UTIL1/FW as well, you would also have to clear the QDFLAG at >FF52 because one of the functions of the lead-in code of FW/UTIL1 is to put the imbedded code for QD in its correct place.

As in Part I the first task is to see if it was loaded from FWB. If not just go to the normal entry code. If so we set the flag and read the information from FWB. The minimum necessary is FWB's internal colour pointer at INCOL (>FF26) so that when FWB is reloaded this can be reset to what it was when FWB was previously exited. If it is not reset FWB will revert to its configured first value which may not be the one you were last using. The next 3 items are only absolutely needed for re-entry by way of LOAD, and contain the GPL return, GROM address to be reset, and the XML address corresponding to that GPL XML instruction in GROM (see FWDOC/REPT) which is also used as an implicit flag for the module in use.

Location RDISK (>FF18) in FWB contains the FWB primary and secondary disk drive numbers in ASCII form, and are saved so that preloaded re-entry prompts may be made. The next item fetches the drive number from the FWB loader name buffer at NAMBUF (>FF62). This is used by DM-1000 for saving its configuration information back to disk, and since it is available in definite form without needing a search we might as well use it.

```
*  FUNNELWEB VN 4.1 Reloader
*  As used in DM-1000 for FWB

ILOAD  LWPI MYREG1
       BLWP @CLRSCN

*  Screen messages

FWBLOD BLWP @DSPTXT
       DATA 6
       DATA 47,TXTCA6,26
       DATA 206,TXTI1,20
       DATA 486,TXTI2,15
       DATA 566,TXTI3,15
       DATA 502,TXTDRV,9
       DATA 582,TXTDRV,9
```

```
* Get drive numbers

IDR     LI   R0,512
        MOVB @RFDISK,R1
        BLWP @GTSKEY
        MOVB R1,@INDSK
        MOVB R1,@FLDISK

        LI   R0,592
        MOVB @RFDISK+1,R1
        BLWP @GTSKEY
        MOVB R1,@INDSK+1
        MOVB R1,@FWDISK
        MOVB R1,@UTDISK

* Load FWB from nominated disk
* Try FW first

LDFWR   EQU  $
        LI   R1,FWPDAT
        BL   @FINDFW
        JMP  FWRITR

* UTIL1 next

        LI   R1,UTPDAT
        BL   @FINDFW
        JMP  FWRITR

* Try XB LOAD then

        LI   R1,FWLDAT
        BL   @FINDFW
        JMP  LODCHK

* Error return

LDFAIN  MOV  @FWFLAG,R0
        JNE  LDFAIN

        BLWP *R0
LDFAIN  B    @ILOAD
```

This next block of code handles the reloading of UTIL1/FW or LOAD. First there are housekeeping details, and then the screen is written up. GTSKEY is a standard DM-1000 routine which accepts a single key with default shown on screen. The nominated drives are stored in INDSK and written into the PAB data for reloading. The E/A side has to be written into PAB data for both FW and UTIL1. It was thought simpler just to repeat the whole PAB. A common load routine FINDFW, code given further on, is used for all versions and success goes to the immediately following JMP, with failure dropping through. In turn it looks for FW on the E/A side drive, and UTIL1 there also. This specific order of search allows the filename UTIL1 to be used for other purposes as it is a name in much demand, and it is the policy of FWB to be as as much an invisible hand in the background as it can.

If both of these fail then it tries for LOAD on the TI-Wr side drive. If this also fails the error return is also taken. If FWFLAG is set it is all tried again, and if not it returns to the title screen. Writing it as shown saves a word of code as R0 of necessity contains >0000.

```
* Load FW/UTIL1 into place

FWRITR LI   R0,VBUF+6
       LI   R1,FWENTR
       LI   R2,>FFD8-FWENTR
       BLWP @VMBR
       MOV  @INDSK,@RDISK
       MOV  @FCOLRS,@INCOL
       CLR  @BTFLAG
       B    @FWENTR
```

When UTIL1 or FW is loaded a simple VMBR of the code into place is followed by rewriting the drive numbers and colour pointer, turning off the boot tracking flag at BTFLAG (>FF1A), and a direct branch to FWENTR. This is a extra safety precaution in case you have a copy of the FWB system with boot tracking enabled, but temporarily residing on a deviant device such as a Myarc RAMdisk, not supported by the boot tracking code. We have seen a flawed attempt to do this in which the interrupt hook was loaded and pointing into the FW entry code area with interrupts still on, even though the programmer had been careful not to have the code with the VMBR destroyed by the incoming block from VDP. There is no way the program can survive this even if its first instruction disables interrupts.

```
* Search for start of FWB in LOAD

LODCHK EQU  $
       LI   R7,>1A00
LDCK20 MOV  R7,R0
       INC  R7
       CI   R7,>2400
       JHE  LDFAIM

       BLWP @VSBR
       CB   R1,@HEXAA
       JNE  LDCK20

* Set up transfer addresses

       STWP R1
       A1   R1,8
       LI   R2,4
       BLWP @VMBR
       CI   R4,>AAAA
       JNE  LDCK20

       CI   R5,>000A
       JNE  LDCK20

* Fetch relevant part of LOAD
* Start CPU pointer to R3

       AI   R7,-3
       MOV  R7,R0
       STWP R1
       AI   R1,6
       LI   R2,2
       BLWP @VMBR

* Calculate length to transfer

       LI   R2,>FFD8
       S    R3,R2
       INCT R7
       MOV  R7,R0
       MOV  R3,R1
       BLWP @VMBR
```

The code after LODCHK looks for where the FWB code is located in LOAD. When this code was first written the LOAD program still contained substantial and variable amounts of XB code, so a wide search range has been left. FWB 4.1 hardly needs this range since the form of the program is now largely fixed, but it is a good idea to leave it flexible. The search uses the knowledge that the start of the main block of FW code has form IDPTR,>AAAA,>000A,.... an ident word >AAAA at IDPTR (=$+2) immediately followed by permanent program data. These are not necessarily on word boundaries in the LOAD file, as XB is byte oriented.

The program searches VDP until it finds byte >AA and then reads that and the next 3 bytes into R4,R5 and checks them there. If the search fails the error exit is taken. If successful the VDP pointer in R7 is backed off to IDPTR and these bytes read into R3 as temporary. The length is then figured from the known endpoint and the transfer to CPU made.

continued on page 20

Of Transformers, Power Supplies and Things

by Tony McGovern

The discussion to follow on external power supplies is not strictly relevant to the usual self-contained TI-99 system, but might well be if you decide to hang external devices off your system, more floppies or perhaps hard disk drives which are starting to appear on some TI systems around Australia using the Myarc HFDC. It came about because Will's Amiga power supply was showing distress. The Amiga A500 power supply is a true successor to the old C64 supply in that it is on the edge of collapsing even with just the basic computer attached. Add another megabyte or two and external disk drives and it gets very dodgy indeed. So a new power supply was his birthday present, and his sister tied it up with a green ribbon for the occasion.

The logical choice by watts and amps for the buck has to be in power supply units for IBM PC clones. I did a little checking around and Richard Terry mentioned the name of a friend of his who was in the PC clone business. So I followed up that lead and ended up buying a 150 Watt XT power supply there. I can recommend this outfit, Penta Cad down in Cardiff, as good people to do business with. The price was competitive, they knew what they were about, and were very helpful with information. No doubt it helped that I knew just the questions I needed to ask and what to do with the answers, but a complete contrast to the norm in computer salesmen. Just try Computer Spot in Charlestown for that total contrast, and you will know what ex used car and hi-fi salesmen are flogging this year. Talking of such computer sales outfits, have you noticed that all of them in Newcastle seem to agree that 5 inch floppies are going to cost you $10 a box this Xmas.

The typical XT supply already comes in a completely enclosed metal box with its own power switch and needs an IEC type mains power lead. A switched power outlet is usually used for mains power to the monitor in PC clones. Dick Smith carries both the lead and the special plugs for the extension power. What I did was remove the standard plug from a multi-way extension and replace it with this new plug. Now the monitor, modem, printer, etc, etc, are all plugged in to this board, and the husky power switch on the XT supply is used as master control. That about completes the mains power side of things. These boxes are just installed as is in the main unit of PCs. One thing to watch for that a reputable supplier will have checked out is that the mains switch is 240 volt rated.

On the low voltage side the output comes on a number of flying leads with connectors attached. In normal PC installations the various low voltage leads are plugged onto the main circuit board or else into disk drives within the main enclosure. In our supply there were 4 disk drive power connectors and 2 with snap on connectors for the PC motherboard. It is easy to identify the ground return (usually black), 5V (usually red), and +12 volt (usually yellow) lines as these come out on the disk drive connectors. Other lines are not so easy to pick by inspection unless you have the detailed information. In the Amiga application only the -12V line is of interest also. You cannot however just turn it on and use your trusty voltmeter to pick which is which, because unlike linear supplies these do not even work without a load. Further if you run them without a load you may do subtle damage internally, because of excessive voltage peaks on the primary side components.

This problem has to be taken care of anyway, because if you are going to use this box as an external supply it will no doubt become detached from its load at some stage. The best solution is to install a power resistor load permanently connected inside the power supply box. The minimum load quoted for this supply was 6 watts, and it did not matter which output was loaded. The one to use is the 5V output as this has the most amps to spare. The supply in question will give 15A at 5V and the Amiga supply was rated at only 4A. A 3.9 or 4 ohm power resistor will do nicely on the +5V line. Its nominal power rating should be much higher than 6 or even 10 W so it does not run hot enough to do any local damage. The fan and air circulation will easily accommodate the extra 6W in a lightly loaded box. I mounted one on a bracket to a free stud inside the box. A better solution would be to use a chassis mounting power resistor and drill a couple of mounting holes in the box in a position which avoids fouling anything inside and is close to the main airstream. If you do this make sure no metal slivers fall on to the circuit board or are left floating around in the box.

Once the power resistor is mounted in place connect it to the 5V supply. The easiest way to do this is to take one of the disk drive connectors - usually you will not need all 4 of them outside the box - clip it off and solder the 5V and the ground return wires to the power resistor, and just tape up the other leads out of the way safely. How short you cut off all these wires depends on whether you will ever want to reconnect the lead outside the box. If a long length is left it clutters up the box inside. This avoids further disassembly of the power supply or soldering to the circuit board.

Now it is safe to turn on and you can double check the + and -12 on the mother board lines. There will be several wires in parallel for each of the 5v and ground leads. With everything turned off and unplugged again, clip off the mother board connectors, with the same compromise as before on length. Take a 6 hole length of plastic mains wiring barrier strip (terminal block), the kind where each wire can be brought into its own terminating hole, with a screw to hold it and make firm electrical connection to the wire coming in on the other side. Install this inside the box - it can float free and otherwise unsupported. Terminate all of the +5V wires into 2 of these, a couple more for all the earth returns, and one each for +12 and -12V lines. Tape up any unused lines safely out of the way (there may also be a -5V supply and a 5V supply ready line). The other side of this block is where you connect your external cable. Depending on the length of this cable you will probably want to use at least two 5V and two earth return lines up to where the power is needed to minimize voltage drops.

In the Amiga A500 application the plug at the computer end is hard to get hold of as Commodore refuse to sell it as a separate spare part, so I just cut the old one off fairly short. Before you make this drastic step check and carefully record what voltages appear on what pins. Repeat the exercise using barrier strip at this end, with doubled wiring in between on the heavy current lines. Only a short length of the Amiga cable should be left on the plug as it is very light gauge and the original A500 supply was run well above 5v to allow for voltage drops. The drive power lines will be used for external 5 inch drives which the original supply could never support. Do a final check that the right voltages are on the right pins before plugging into the computer.

In TI-99 applications where could this be useful? Mostly it would be for powering external disk drives, floppy or hard, or other experimental work, and it would be the disk drive connector lines that would be used for external drives. The main +5V, and +12 and -12V supplies are not of much relevance to the TI-99 PE-Box or its plug-in cards, as these use a higher voltage on the back-plane with separate local regulators on every card. Typical TI hardware intensive approach, but the TI-99 PE-Box is an industrial strength system that makes any IBM/clone type enclosure look like a delicate consumer item.

The problem with the PE-Box is that it comes with these intermediate supply rails running at very much higher voltages than the circuit diagrams indicate. I suspect this happened because it was designed about the

time brown-outs were common in the USA and the TI designers decided to build in as much tolerance to mains fluctuations as the cards would stand in extra dissipation in their own regulators. This has caused a problem in later years in that some cards now available, particularly from Myarc, have much poorer thermal design than the originals. Switching supplies, at least if they are running well below full output, can be much more tolerant of mains variations and are more efficient, but are more complex and harder to fix. Commodore combined the worst of both worlds in the A500 supply.

Over and above this there is the problem in Newcastle and Australia in general, that while the PE-Box was designed for 110 or 220V AC mains supplies, the actual mains supply is at least a nominal 240V and in Kotara is over 250V much of the time. The transformer in the PE-Box seems to be adequate to handle both the higher voltage and 50 Hz rather than 60 Hz excitation but has no winding taps to allow compensation. This may be responsible for the excess leakage fields that have caused problems for AVPC and Geneve owners in 50 Hz countries who tried to sit their monitors on top of the PE-Box. This is something that had not been noticed very much if at all in USA on 60 Hz and the reason may very well be the rapid increase in leakage fields as transformer cores start to go into saturation when driven too hard. Our immediate concern is the high voltage on the PE-Box supply power rails and high dissipation in all the on-card regulators, which is not a good idea for long term reliability, let alone overheating problems in summer. The solution to this problem has been given in the Sydney TND, but we will talk about it in detail here anyway.

The fix is to use a transformer to reduce the AC supply voltage to the PE-Box. I have heard of people in the USA using a light dimmer but I would not recommend that way, as I suspect it would reduce the fan speed more than the internal heat generation and maybe even make things worse. The transformer does not need to be of full power rating if connected as an autotransformer. Only the secondary winding has to be rated at the PE-Box current level, and a 1 Amp rating should be adequate for a TI system. I used a spare 240V to 30V transformer. My main system is 110V only and already uses a 240/120V transformer in a box (it came with the system when I originally bought Newtech's demo system after Orphan day) so I mounted this transformer in the same box. If starting with a 240V system you would have to provide your own electrically safe box. Ben Takach reported in the Sydney TND that the PE-Box would operate fine down to 185V. There is a trade-off here familiar to linear supply designers. The lower you make the supply voltage the less excess dissipation you have to put up with, but you have less headroom for mains voltage fluctuations before it all collapses. As it stands the TI PE-Box has an enormous margin, but I would not want to push the supply down too far at the risk of losing the memory or corrupting disks when caught by a mains voltage drop.

How to make the connections ? If you want to connect your console up to the reduced voltage also, use a small multiple extension strip with its plug removed for the output lead. Do not create a temptation to plug all sorts of things in, and install a fuse in the incoming active line if you do things this way, in order to protect the transformer. Firstly the green earth lead must be carried through the mounting box by a mechanically and electrically firm connection. Connect the transformer 240V primary across the brown (active) and blue (neutral) wires. Transformers to Australian standards will have a well insulated and separated primary winding. If your house wiring installation is correct there should be almost no voltage between the (blue) neutral and the (green) earth. After this is verified, connect the outgoing (blue) neutral wire to the incoming (blue) neutral. Connect one end of the transformer secondary (30V or thereabouts) winding to the end of the primary where the incoming (brown) active wire is connected. Do not connect the outgoing active lead yet. Now with the respect and care due to 240V AC, plug in the input lead to the mains, turn on the input voltage, and measure the voltage between the other end of the secondary and the common neutral. You have a 50-50 chance of getting it right first time with the secondary voltage subtracting from the primary voltage. If it is higher then turn off, unplug, and start again with the other end of the secondary. When you get it right, again turn it off, unplug, and connect up the outgoing (brown) active lead to the free end of the secondary winding where you were measuring the voltage. Now make sure everything is mechanically and electrically sound, and stand back and admire your handiwork.

This seems to have worked very well for the PE-Box, but has brought a subsidiary problem. Those of you who had the original 110V TI external disk drive units will know that they always ran very hot. I suspect the original power supply in these was marginal even in the USA at 60 Hz. Ours have been gathering dust ever since we put the Chinons in the PE-Box, but recently I hauled one out to install a pair of second hand Panasonic 720K/80T 5 inch drives. On the reduced supply they run just fine though the transformer core still gets pretty hot. This is only true while the mains supply is at its normally very high level, but if it drops to even a more normal level the drives quit working on the reduced supply. If the transformer core stays only warm, there is not enough voltage to run regulators and drives. In other words given the real ratings of the transformer at 50 Hz, these PHP1850C units should never have been released in Australia without a revised power supply. So we are in a bind, and the solution may very well be an external drive supply. And I have had practice in doing that now! We have one of those Burroughs bank boxes but it has clapped out with a mysterious fault I have not been able to diagnose (mysterious to me anyway and I met my first low voltage switching power supply 20 years ago in the Electric Turkey). That Burroughs box was an utter waste of money and only ever had about one day of use. Seems to be the story of home computers - you end up with this trail of obsolete or junk equipment bought at vast expense not long before.

So there it is - power supply problems solved. Remember that any disasters are your own, and if you do not understand what it is about get someone who does to do the actual work. This particularly applies to the 240V side of things. o

continued from page 2

On the September and November meeting days, the E/A class will preceed the normal meeting whilst this class will only be conducted on the morning of the October full day tutorial to allow other activities to be attended.

A TI99/4A computer with 32K and disk drive will be available to demonstrate techniques to the group. All attending are required to bring their Editor/Assembler manual, note pad and writing materials. Work will be set to be completed between meetings of the group. This means that all participants will need regular access to a TI99/4A with a minimum of 32K memory expansion, disk drive and Editor/Assembler module or FunnelWeb and X/B. In addition sufficient time will need to be allocated by each participant to practice that which has been learnt.

The beginners' Editor/Assembler class will be open to all TISHUG members who want to get started with assembly language, just turn up to the first class at 12 noon on 4th September at Ryde Infants School.

A preliminary requirement for all who wish to start in the class is to thoroughly read sections 1 and 2 of the Editor/Assembler manual. The E/A EDITOR should be loaded and used to type in at least a few lines. The work should then be saved to disk. (If you don't know what to type, just copy a few lines of this article with the same line lengths). The first class will be include the concept of assembler SOURCE and OBJECT files, the E/A EDITOR and memory usage of the computer.

Remember 12noon 4/9/90 at Ryde Infants School, please be on time. o

Correction: The date of the Assembler Class is September 1st.

Newsletter Update

by Bob Relyea

It seems like a long time since we last included the newsletter updates in the club magazine but they do not always arrive on a regular basis plus other things getting in the way make for very irregular updates. Anyway, we now have quite a pile to be deciphered, so here goes...

NATIONAL NEWSLETTERS

HUNTER VALLEY 99ERS, April, 1990; besides having a very interesting cover of Sulfur crested cockatoos there was some local news. There was an article on comparing languages (Random Bytes) by Bob Carmany and also an article on Forth. For those thinking about abandoning their TI, there was an appropriate article by Jack Sughrue. Also an article on Modems, Data Transfer between TI Forth screens and TI Extended Basic files, comments on programs such as TRACE and TI SORT (Far Out) by Dick Schaydel, a review on the DEMO disk of the MISSING LINK, a report on the software library.

TI UP TIT BITS (PERTH, W.A.), February/March, 1990; an article (advertisement?) entitled 'Know Your TI 99/4A, a game to type in (A travellers guide to the wandering stars), more on Calendar programs, formatter commands, procedures to set printers by Fred Moore (reprint), a look at other newsletters, conversion of Forth Screens to DV 80 (and back to Screens), an appropriate article on 'Why I continue to use my TI 99/4A', TI UP LOGO V.4.0.

AITCC, Adelaide, June, 1990; local news, Colin Cartwright's Kaleidawriter project (in Coordinators Report), various bits and pieces with TI Writer & DM 1000, Far Out by Dick Schaydel, The Kiddie Corner, The Undocumented Features of 'G', a program to type in (Display Races, a program in 'G'), a summary of TI Writer printing commands, tips on Basic and Extended Basic, a review on Smartkey by Bob Carmany, a listing of available software.

BUG BYTES, Brisbane, May, 1990; local disk library, Tips from the Tigercub #18, What's New, TI Base quick reference, TI-Artist Plus, Multi-colour mode, 31's Part II, Cassette error codes, Modifying PEB Power Supply, calendar programs, Viatel, Basic Training, Newsletter circuit, TI's Unreleased Legends.

BUG BYTES, Brisbane, June, 1990; Missing Link Programme, Modifying Funnelweb Default Colours, Getting into XBasic from ROS or Boot Menu Screen, Corrupted ROS in Ramdisk, a peek at the new "Enhanced Extended Basic" from Millar Graphics - looks good! Prescan, a programme designed to speed up your XBasic programmes, Tips from the Tigercub #19, What's New, a Basic programme to type in entitled 'Muncher', notes on TI Keys by Wes Johnson.

OVERSEAS NEWSLETTERS

MICROpendium, May 1990; Regena on Basic- Plane Geometry, Raves new PEB, BBS Listings (U.S. ones), Translating from other XBasics to TI XBasic, EEproms and the TI, Tank Commander, Checksums and ASCII values, Forth- more on high resolution graphics, questions and answers about Ramdisks, Business graphs and chart programs compared, reviews on P-Gram Card, Mechatronics 80-Column Card, Air Taxi, Rock Runner, TI Base V3.0, another way to beef up the PEB power supply.

LA 99ers, May, 1990; A new Directory programme by Earl Raguse, Geoff's article on 'Interfacing to RS232 PIO Port' with the letter written to the club,Forth Dimension - ANS Forth, Increase the Memory of the TI with TI Forth, Did You Know? with a mention of the MIDI Interface from Asgard due soon, Beginning Forth 20 by Earl Raguse, a list of the software library.

LA 99ers, June, 1990; Technospeak by Earl Raguse which attempts to explain computer language, the announcement of a new TI software supplier called KB Company, TI-Base Command File Merge Utility, Did You Know? (Chick de Marti's farewell issue!) including quite a few hints from Stephen Shaw's 'Rambles' and a few computer 'quickies', Error Protocals and Modems, How to remove the cover from the P-Box, 'Style a Line', a short programme that can be typed in it's entirety without any program lines scrolling off the screen, a list of available software.

Spirit of 99, Ohio, May, 1990; Lima Fair Info, 'What's HOTT - the Rave P-Box & a mention of the Midi conference on the MIDI Interface for the TI or Geneve, Unlimited Basic by Jim Petersen, Star Print-Head Repair, Designing Characters made easy, a page of handy hints for the TI99/4A from past magazines, Root Finder (a maths programme), a random word generator programme to type in, Card Catalog programme to sort out files, TI-Base tutorials 14.1.1,2,3 by Martin Smoley.

Spirit of 99, Ohio, June, 1990; What's Hott - a review of various bits of software, What happened to the Fun of It by Jim Petersen, Air Taxi review, Using a Modem-Part II, Tips from the Tigercub #60, a review of tape libraries, and TI-Base Tutorials 14.2.1 & 14.2.2 by Martin Smoley.

TI Focus, Ontario, May, 1990; News and Views by Tom Arnold - a look at software past and present including the new Advanced Basic for the Geneve, Identifile by Tom Arnold, Root Finder (as in the Spirit of 99, May Issue), Club Page and Club News, Club Library Update, Telecommunications by Tor Hansen, Putting It All Together #7 by Jim Petersen, New Age/99 by Jack Sughure, Ramdisk Folly or How to Build a Solid State Floppy, report of speech after the Melbourne Faire.

TI Focus, Ontario, Summer, 1990; News and Views by Tom Arnold with a mention to such things as the new Rave PEB Box, the Jiffy Card, Spell It, TI-Base V3.0, various gear for the Geneve and GIF Picture Loader, New Age/99 with a general overlook at a lot of disk software, a nice full-page demonstration of Page-Pro 99 as well as a good write-up of this software package. A review of PLUS! V. 2.0 was also given. There was also Club Page by Tor Hansen and Program That Write Programs, Part 6 by Jim Petersen. On the back page was a XBasic Quick Reference Sheet.

THE FRONT RANGER, Colorado Springs, May, 1990; Trouble Shooting With Your TI, Rambo comes to the TI World which is press release by Bud Mills, An interesting article entitled - "Computers, New Today, Vintage Tomorrow", TI Tool Box #8 - Taming the Formatter, an article on resurrecting an XBasic cartridge, TI-Base tutorial by Jack Shughrue, Myarc Geneve 9640 and Randisks, Part IV by J Willforth.

THE FRONT RANGER, Colorado Springs, June, 1990; Mind Game, a challenge of the Intellect, Page Pro Conference - find out what the experts think, Men and Reviews Down Under by Tony McGovern, Tips for your TI from Bits, Bytes and Pixels, This Month In TI History - Catching up with Bill Gaskill, a big advertisement for Spell IT.

UGOC ROM, Orange Beach, Ca. March, 1990; local news, Forth article by Earl Raguse, an article praising the TI, Two Columns with TI Writer - How to do it by Jack Raguse, Dips and Chips by Siles Bazerman.

UGOC ROM, Apr/May, 1990; local news, a review of Certificate 99 by Bill Nelson, And So Forth #47 by Earl Raguse, Tips V. 1.3 - a review, Letterform by Ollie Herbert.

UGOC ROM, June, 1990; local news, Modem Use Tutorial - transferring files between friends, And So Forth #48 by Earl Raguse.

UGOC ROM, July, 1990; local news, And So Forth #49 by Earl Raguse, TI Bits * 31 by Jim Swedlow including an announcement of the release of Funnelweb V.4.3 and multiple printers and computers. Also, the Joys of Belonging by Stan Corbin, and Dips and Chips by Siles Bazerman with a mention of Boot and Root.

continued on page 20

More on 4-D Graphics

by Jim Peterson, Tigercub Software, USA

It seems that my previous article on creating 4-dimensional graphics has created some controversy. Programmers have reported no success in creating these graphics, and some have even questioned my logic.

In fact, these programmers may have succeeded without knowing it. I regret to report that a program to create 4-D graphics will result in nothing more than a blank screen because I have now learned that 4-D objects are invisible!

The proof of this is found, not in the laws of geometry, but in the study of natural science. I refer to that rare and elusive reptile known to the Indians as Wig-Lum-No-See-Um, which translates as "The Snake That Has Never Been Seen". As all Indians know, when this snake is in danger of being observed, it grabs its tail in its mouth and swallows itself, thereby becoming invisible. When the danger has passed, it spits itself out and again becomes visible but unseen. For this reason, it has never been described by biologists.

Now, my hypothesis was that a 4-D object could be created by the rotation of a 3-D object. A snake is obviously a three-dimensional object, and it must obviously rotate in order to swallow itself. Since this causes it to become invisible, we may safely assume that all four-dimensional objects become invisible while being created. o

continued from page 16

```
*  Restore FW to match previous

FWEXIT EQU  $
       LI   R1,INDSK
       MOV  *R1+,@RDISK
       MOV  *R1+,@INCOL
       MOV  *R1+,@SVGPRT
       MOV  *R1+,@GRMAD
       MOV  *R1+,@MODFL

*  Hand over to FWEB

       LWPI FWREGS
       CLR  R4
       SETO R13
       MOV  @CMSRET,R11
       RT
```

The values that would be established by the normal FWB entry code are now refreshed and an exit made to the TI-Wr side of the central menu screen, as discussed in Part I of this minor opus. The loader BL subroutine and various PAB data follow.

```
*  Setup DSR and do it

FINDFW EQU  $
       LI   R0,PAB
       LI   R2,>20
       BLWP @VMBW
       LI   R0,PAB+9
       MOV  R0,@>8356
       BLWP @DSRLNK
       DATA 8
       JNE  FWSUCC

       INCT R11
FWSUCC RT

*  PAB data for reloading FWB

FWPDAT DATA >0500,VBUF,>0,>2200
       DATA >07
       TEXT 'DSK'
FWDISK TEXT '1.FW'
       EVEN

UTPDAT DATA >0500,VBUF,>0,>2200
       DATA >0A
       TEXT 'DSK'
UTDISK TEXT '1.UTIL1'
       EVEN

FWLDAT DATA >0500,VBUF,>0,>27D0
       DATA >09
       TEXT 'DSK'
FLDISK TEXT '1.LOAD'
       EVEN
```

A possibility implicitly not considered in the previous discussion is that the program being loaded might destroy part or all of the FWB system block at the top of hi-mem. In that case the simplest way out is just to return to the title screen, as there is then no immediate way to check if the program was loaded from FWB. Otherwise a specific option to return to FWB may be included which reloads UTIL1 or FW and branches to its normal entry at FWENTR (>E006). o

continued from page 19

The PUG PERIPHERAL, Pittsburgh, April, 1990; Presidents Page, New Age/99 #2 mentioned elsewhere, Transferring Scott Adams Adventures from Cassette to Disk, Forth Tidbit #10, Risk vs. Anxiety, Speed Reader - a review.

The PUG PERIPHERAL, Pittsburgh, June, 1990; TI-Base Helps, Printers #7 by John Willforth, Removing Cover from PEB Box, Tips from the Tigercub #52, New Age #5 by J. Sughrue (software review), Tetris - a review, TI Writer Tip (Handling Text Files ...)

TIDBITS, Memphis, May, 1990; mostly local news and the announcement of a faire, a review of SPELL IT and that is about it!

TIDBITS, June, 1990; local news, a look at Texaments products such as the Missing Link, A major Review of The Missing Link by Bill Gaskill, some notes on the Berlin TIers, TI-Base V. 3.0 by Bill Gaskill (6 pages worth).

LEHIGH 99ers, February, 1990; new graphics mode for the TI involving a new chip, Fortran from Scratch by Jerry Boyer (part 2).

LEHIGH 99ers, March, 1990; a review of Funnelweb V. 4.21! and the third part of the series of Fortran from Scratch.

LEHIGH 99ers, May, 1990; an article of the (T)exas (I)nstruments (C)omputer (O)wners (F)un (F)aire held in March, Geneology Computing, XBasic Quick Reference Sheet, a look at graphics generated by the Missing Link.

TIC TOC, Rocky Mountain 99ers, April, 1990; TI-Base Tutorials 9.2.1&2 by Martin Smoley, Getting the most from your Cassette system (Part I), Funnelweb Tips, CorComp information, Word Finder by John Willforth.

TIC TOC, May, 1990; Configuring Funnelweb by David Fink, Sub Routines by Bruce Nattrass, The VCR Connection, TI-Base Tutorials 10.1.1&2 by Martin Smoley.

The BOSTON COMPUTER CLUB SOCIETY, May 1990; a report of the 5th annual Fayuh held in Waltham, an introduction to the UCSD P-System (using units and configuring your system) by Ron Williams, ForTI Card on a 9640? from Delphi's Message Base, My-Art File Formats by J. Peter Hoddie.

TACOMA 99ers, June, 1990; an issue devoted primarily to listing and briefly explaining the various modifications available for the TI, Stationery Letterheads by Jim Luque, Notepad Creation Part II with TI-Artist and Page Pro.

CIM 99, Ottawa, May and July issues; I could not understand much of it as it is in French, but if you read French have a look at them. o

Real Programming part 1

Author unknown, USA

```
*********************************************
*     REAL PROGRAMMERS DO NOT WRITE PASCAL  *
*********************************************
```

Back in the old days - the "golden era" of computers, it was easy to separate the real men from the boys (sometimes called "real men" and "quiche eaters" in the literature). During this period, the real men were the ones that understood computer programming, and the quiche eaters were the ones who did not. A real programmer said things like:

DO 10 I=1,10 and: ABEND

They talked in capital letters, you understand. The rest of the world said things like "computers are too complicated for me" and "I cannot relate to computers - they are so impersonal". A previous work (1) points out that real men do not "relate to" anything, and are not afraid of being impersonal. But, as usual, times change. We are faced today with a world in which little old ladies can get computers in their microwave ovens, 12-year old kids can blow real men out of the water playing asteroids and pac-man, and anyone can buy and understand their own personal computer. The real programmer is in danger of becoming extinct, of being replaced by high-school students with TRS-80's.

There is a clear need to point out the differences between the typical high-school junior pac-man player and a real programmer. If the difference is made clear, it will give these kids something to aspire to - a role model, a father figure. It will also help to explain to the employers of real programmers why it would be a mistake to replace the real programmer on their staff with 12-year old pac-man players (at a very considerable salary saving).

LANGUAGES

The easiest way to tell a real programmer from the crowd is by the programming language he or she uses. Real programmers use FORTRAN. Quiche eaters use Pascal. Nicklaus Wirth, the designer of Pascal, gave a talk once at which he was asked "How do you pronounce your name?". He replied "You can call me by name, pronouncing it 'Veert', or you can call me by value, 'Worth'." One can tell immediately from this comment that Nicklaus Wirth is a quiche eater. The only parameter passing mechanism that real programmers endorse is "call by value - return", as implemented in the IBM/370 FORTRAN G and II compilers. Real programmers do not need all those abstract concepts to get their jobs done - they are perfectly happy with a keypunch, a FORTRAN IV compiler and a beer.

* Real programmers do list processing in FORTRAN.
* Real programmers do string manipulation in FORTRAN.
* Real programmers do accounting (if they do it at all) in FORTRAN.
* Real programmers do artificial intelligence in FORTRAN.

If you cannot do it in FORTRAN, do it in assembly language. If you cannot do it in assembly language, it is not worth doing.

STRUCTURED PROGRAMMING

The academics in computer programming have gotten into the "structured programming" rut over the past several years. They claim that programs are more easily understood if the programmer uses some special language constructs and techniques. They do not all agree on exactly which constructs, of course, and the examples they use to show their particular point of view invariably fit on a single page of some obscure journal or another - clearly not enough of an example to convince anyone. When I got out of school, I thought I was the best programmer in the world. I could write an unbeatable tic-tac-toe program, use five different computer languages, and create 1000-line programs that worked (really)!!! when I got out in the real world. My first task in the real world was to read and understand a 200000 line FORTRAN program, then speed it up by a factor of two. Any real programmer will tell you that all the structured coding in the world will not help you to solve a problem like that - it takes actual talent. Some quick observations on real programmers and structured programming:

* Real programmers are not afraid to use GOTO's.
* Real programmers can write five-page long DO loops without getting confused.
* Real programmers like arithmetic IF statements - they make the code more interesting.
* Real programmers write self-modifying code, especially if they can save 20 nanoseconds in the middle of a tight loop.
* Real programmers do not need comments - their code is obvious.
* Since FORTRAN does not have a structured IF, REPEAT ... UNTIL, or CASE statement, real programmers do not have to worry about not using them. Besides, all those structures can be simulated, when necessary, by using assigned GOTO's.

Data structures have also gotten in a lot of press lately. Abstract data types, structures, pointers, lists and strings have become popular in certain circles.

Nicklaus Wirth (the aforementioned quiche eater) actually managed to write an entire book (2) contending that you could write a program based on data structures, instead of the other way around. As all real programmers know, the only useful data structure is the array. Strings, lists, structures, sets - they are all just special cases of arrays and can be treated that way just as easily without messing up your programming language with all the sort of complications. The worst thing about fancy data types is that you have to declare them, and real programming languages, as we all know, have implicit typing based on the first letter of the (six character) variable name.

OPERATING SYSTEMS

What kind of operating system does the real programmer use? CP/M? God forbid - CP/M, after all, is basically a toy operating system. Even little old ladies and grade school students can use and understand CP/M. Unix is a lot more complicated of course - the typical Unix hacker never can remember what the print command is called this week. But when it gets right down to it, Unix is a glorified video game. People do not do serious work on Unix systems - they send jokes around the world on UUCP-net, and write adventure games and research papers. No, your real programmer uses OS/370. A good programmer can find and understand the description of the IJK3051 error he just got in the JCL manual. A great programmer can write JCL without referring to the JCL manual at all. A truly outstanding programmer can find bugs buried in a six-megabyte core dump without using a hex calculator (I have actually seen this done). OS/370 is a truly remarkable operating system.

It is possible to destroy days of work with a single misplaced space, so alertness in the programming staff is encouraged. The best way to approach the system is through a keypunch. Some people claim that there is a time sharing system that runs on OS/370, but after careful study I have come to the conclusion that they were mistaken. o

For Sale

TI 99/4A, EXPANSION BOX, 3 DISK DRIVES, STAR PRINTER, INTERNAL MODEM, VOICE SYNTHESIZER, COMMERCIAL MODULES, DISKS AND MANUALS, NEW DISKS AND BOX, CALL FOR DETAILS. $900 WITH COLOUR TV, OR $700 WITHOUT TV.

WANTED: A CARTRIDGE EXPANDER (WIDGET) PRICE NEGOTIABLE . PHONE NO. 871-1514

o

Regional Group Reports

Meeting Summary For SEPTEMBER

Banana Coast	09/09/90	Sawtell
Carlingford	19/09/90	Carlingford
Central Coast	08/09/90	Saratoga
Glebe	06/09/90	Glebe
Illawarra	10/09/90	Keiraville
Liverpool	07/09/90	
Northern Suburbs	28/09/90	
Sutherland	21/09/90	Jannali

BANANA COAST Regional Group
(Coffs Harbour area)

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

CARLINGFORD Regional Group

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

CENTRAL COAST Regional Group

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

GLEBE Regional Group

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

ILLAWARRA Regional Group

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

LIVERPOOL Regional Group

Regular meeting date is the Friday following the TIsHUG Sydney meeting at 7.30 pm. Contact Larry Saunders (02)644 7377 (home) or (02)642 7418 (work) for more information. The proposed activities for the coming months are as follows:

7th September 1990 - Getting the best out of word processors and their utilities.

12th October 1990 - New programs from Asgard Software

All Welcome. Larry Saunders

NORTHERN SUBURBS Regional Group

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

Come and join in our fun. Dick Warburton.

SUTHERLAND Regional Group

The July Regional meeting was largely involved with the review of library software, which was made available earlier in the year.

Joe D'Ambra was on hand to demonstrate his expertise with the Disk Utilities software, in repairing the Sectors 0 and 1 on a damaged disk. Regular meetings are held on the third Friday of each month at the home of Peter Young, 51 Jannali Avenue, Jannali at 7.30pm.

The format of each meeting is quite informal, with topics ranging from software reviews to hardware modifications with a fair sprinkling of purely social chatter in between.

BBS Contact is Gary Wilson, user name VK2YGW on this BBS.

Peter Young Regional Co-ordinator

TIsHUG in Sydney

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

SEPTEMBER MEETING - 1st September

This month will be a BUY, SWAP SELL DAY so be on hand to find that item you want at the right price. It is also a good time to unload any gear you are not likely to be using. We will also try to give a demonstration of Version 3 of TI-Base, FWB 4.3 in 80 column mode, and the new, faster Multi-plan update. A games room with Shane will be revived for the younger set. Percy will be available with the shop gear, Alf with the library and Geoff should be on hand for assistance with repairs.

Other meeting dates for 1990 will be as follows.

OCTOBER 6
NOVEMBER 3
DECEMBER 1

SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

October Meeting Notes.

Saturday 6th October 1990 meeting will start at 10-00AM.

FULL DAY LECTURE TYPE WORKSHOP.

Alf Ruggeri Larry Saunders - Page Pro
Bob Relyea - Word Processing

Other items to be added as they come available with volunteers to run them.

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

October	9 September
November	7 October
December	11 November

Russell Welham (Meeting coordinator).

SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS o

continued from page 8

RAMdisk with EPROMs. However the other limit on the number of EPROMs on each RAMdisk, is the number of available chip select pins. Since at least three chip select pins are tied up with RAM chips, you must use at least three RAM chips. This imposes a limit of fourteen individual EPROMs on any one RAMdisk. (A mere 896K of EPROMs on your one RAMdisk) If your modifications do not meet with success, have someone else check your work. If this does not prove successful, ask another member who is familiar with the workings of the EPROM modification to test it for you. Any queries can be directed to Dick Warburton ((02)918 8132). If necessary, he can forward the query to me to answer. My home number at present in 1990 is (03)543 2657. You may telephone me first, but Dick can answer most questions, and you will not need to make an expensive STD call.

I wish you all the best in completing your own EPROM RAMdisk modification.

NOTE: If for any reason the data on the RAM chips becomes corrupted, The installation procedure must be repeated from scratch. o