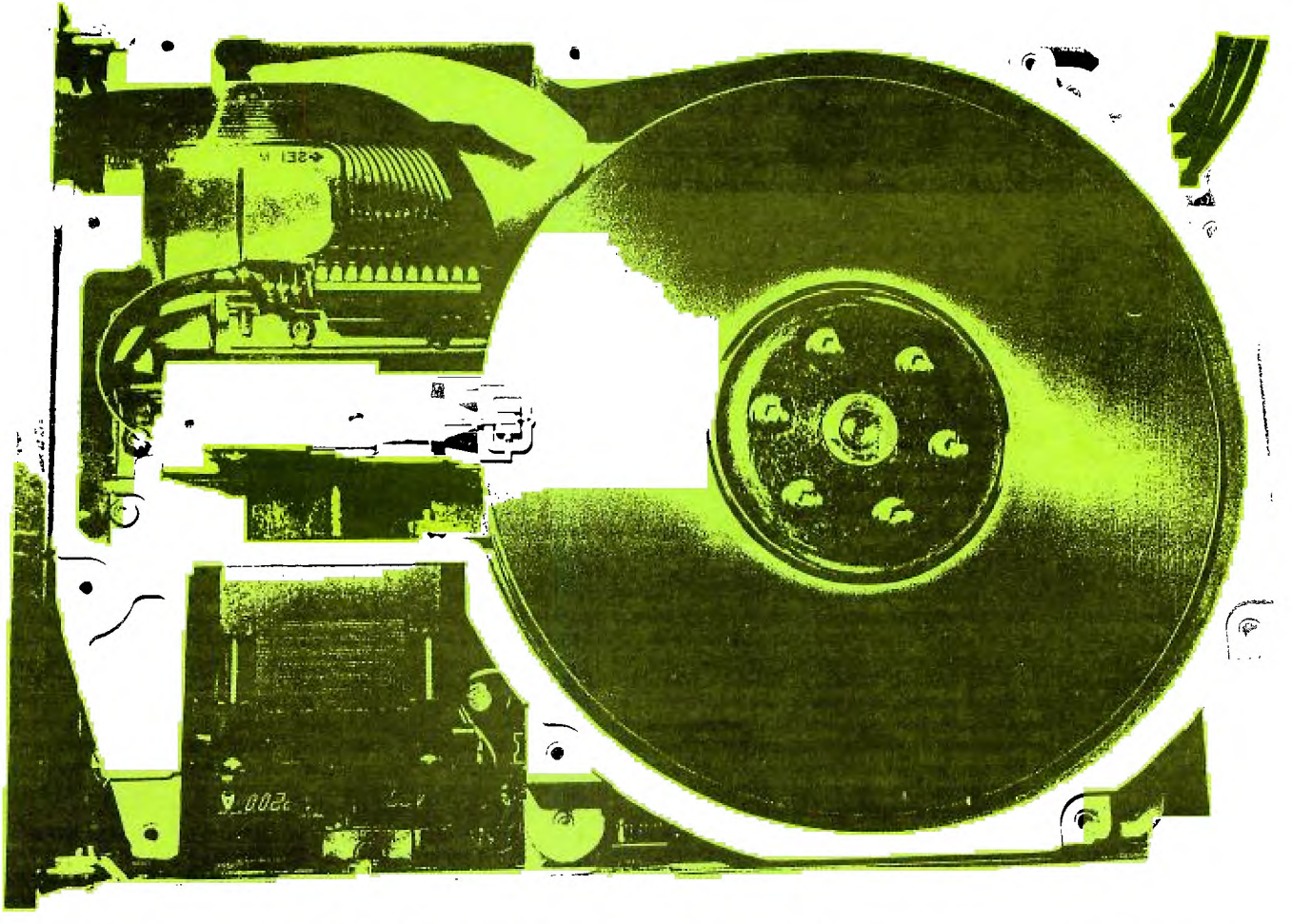
(042 8909

TISHUG NEWS DIGEST

Focusing on the TI99/4A Home Computer

Volume 8, Number 9 September, 1989

Registered by Australia Post - Publication No. NBH5933



Special Issue on Hard Disks

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

TIsHUG (Australia) Ltd.

TIsHUG News Digest

September 1989

All correspondence to:

P.O. Box 214
Redfern, NSW 2016
Australia

The Board

Co-ordinator
Dick Warburton (02) 918 8132
Secretary
Terry Phillips (02) 797 6313
Treasurer
Rolf Schreiber (042) 84 2980
Directors
Robert Peverill (02) 602 4168
Russell Welham (043) 92 4000

Sub-committees

News Digest Editor
Geoff Trott (042) 29 6629
BBS Sysop
Ross Mudie (02) 456 2122
BBS telephone number (02) 319 1009
Merchandising
Steven Carr (02) 608 3564
Publications Library
Warren Welham (043) 92 4000
Software library
Terry Phillips (02) 797 6313
Technical co-ordinator
Lou Amadio (042) 28 4906

Regional Group Contacts

Carlingford
Chris Buttner (02) 871 7753
Central Coast
Russell Welham (043) 92 4000
Coffs Harbour
Kevin Cox (066) 53 2649
Glebe
Mike Slattery (02) 692 0559
Illawarra
Geoff Trott (042) 29 6629
Liverpool
Larry Saunders (02) 644 7377
Northern Suburbs
Dennis Norman (02) 452 3920
Sutherland
Peter Young (02) 528 8775

Membership and Subscriptions

Annual Family Dues	$25.00
Overseas Airmail Dues	AUS$50.00

TIsHUG Sydney Meeting

The next meeting will start at 2 pm on 2nd of September at the Woodstock Community Centre, Church Street, Burwood.

Printed by
The University of Wollongong
Printery

Index

TIsHUG Software Author of the Month

The Software Authors for this month are the Ottawa Users Group for the Program DM1000. All Donations collected at the meeting and sent in will be to them this month

They're off

by Geoff Trott

As you will read elsewhere in this issue, the last meeting was a very busy one in a somewhat restricted area. We did not have time for some of our usual activities, including the collection for the software author of the month. So we will have to do that at the next meeting. I hope you remember that this month it is for DM1000 and the Ottawa Users Group who have provided this excellent program for all those with disk drives. I am sure that everyone will want to contribute something for this program. Even those who cannot attend the meeting can send in a contribution to the Software Author of the month so get to it!

****

I received a letter from John Hagart of Gordonvale, North Queensland in which he asks for help and suggests what I consider is a very good idea. He points to the problem of being so far from someone to ask for help and wonders if TIsHUG has tutorial disks to help users of Editor Assembler, TI-Writer, Multiplan and other software to get started. He was typing in some assembly language program from the TND and did not realise the significance of the asterisks at the start of the lines. He says that he has a tutorial disk for the Extended Display Package which he finds very useful (well done Craig). This sounds to me like an excellent idea for the Special Interest Groups or for another Special Interest Group to attack on its own. It may be only a matter of collecting some suitable material from what is already available and packaging it properly to provide the maximum help. It may also be a marketable idea to other user groups around the world.

continued on page 34

Co-ordinator's Report

by Dick Warburton

What a day at our August Meeting. I estimated that over one hundred people turned up for the bargains. I promised myself that I would not buy too much, but I cannot resist a bargain. I came home loaded. We also signed up three new members on the spot. It is clear that there is a strong interest in hardware, among TI99/4A owners.

Unfortunately the room available was limited, and we were unable to participate in other activities. I think that we should have more days like this one, and also make provision at our monthly meetings for hardware swaps and sales.

Our console repair group is becoming a little more experienced and confident. The new console tester has already been put to good use, and it is clear that some members are becoming much more confident about opening up their consoles and tackling the problems. At the meeting, people are doing a range of things technical. Thanks to Ben Takach and his super desolderer, chips are removed and replaced without drama or disaster. If you have a console which has breathed its last, either bring it over to Cyril's place, or send it, and our budding brain surgeons will attempt to fix it. If you have some old parts, for example keyboards, power supplies, mother boards, etc which you do not need, donate them to the console repair group. There will come a time when we will have to be totally self reliant in these areas, so we need to develop our skills now. If you know people with old unused or broken consoles, tell them about the club and what we are doing. Some might like to donate old gear to the club. Some might like to get their consoles fixed. People who join our club get pretty good value for their subscription, particularly if they get a console fixed cheaply. The willingness to help each other is what makes this club such a great group to belong to.

Our publicity group is flagging. I need more helpers to put some of our good ideas into operation. We need a few hard working people to do a range of things. Perhaps interested members could see me at the next meeting, and get together after the main activities. We need more publicity. If we can attract about ten new members a month, our club will remain strong, and be able to provide help and support for the future. If you know someone with a TI99/4A then tell them about the club. Tell them what we do for each other, and what we can do for them. There are lots of TI99/4As still out there, and we would like to find more of them. The other interest groups are going well. Our TI-Artist group, goes from strength to strength. More and more software is being accessed, and more skill is being developed. The numbers of interested people is growing. The word processing is a small but quite active group which is making considerable progress. I am sure that some of you would like to attend an interest group. If you have other suggestions, please raise them at the next meeting, and we will try to establish a group if the interest is sufficient. Small groups seem to work well. I think we could run all types of groups to meet our members' needs.

I have recently heard that Horizon RAMdisks are out of production because of the present high costs of 32K chips. These chips are selling at US$28 and the cost of completing a RAMdisk is very expensive. In the US, the trend is towards hard drives, which are retailing at approximately US$185. It would seem that the price of hard disks will drop soon, however the cost of the Myarc controller is still quite high. A hard disk would certainly add to the flexibility and power of the TI99/4A. While mentioning hardware, let me offer a vote of thanks to Lou Amadio and Geoff Trott and the boys in Wollongong. What a great series of articles and projects keep appearing in the TND, and they actually get built and demonstrated. Is there no end to what can be done with a TI99/4A?

Do not forget the TI99/4A Faire in Melbourne on 14th October. It sounds really good, and a strong representation from Sydney will help us and them. The Directors are writing to other home computer user groups, to float the idea of a combined home computer faire, in Sydney, in 1990. The trip to Melbourne might help us to prepare for our own exhibition. I have been wondering about the idea of displaying our TI99/4As in Regional Shopping Malls, in areas where TI99/4As were sold. Do we have any volunteers to man a stand in a centre? Let me know what you think of the idea. I think it could generate increased interest in our machine. If you have any ideas for promotion, we will discuss them and try to implement them, if we can get sufficient helpers.

My thanks again to those members who give so much time and effort to make the club what it is. I really appreciate your help. Remember what Dawn Fraser says: "No pain, no gain." If you would like to help the club break some records, but are not sure how you can help, see me and I am sure we can find something rewarding for you to do. See you at the next meeting. O

Fairware Author of the Month

We, the users of the TI99/4A, rely on many people for our enjoyment of our computer, none more so than those who have written software which we use and rely on every time we use our computer. Some of this will be commercial software which we should have paid for and received value for our money in the form of a working program with good documentation, but the majority of software will be Fairware, which may not have cost anything and yet still provides a working program and good documentation. Software authors who produce good useful programs and release them for us all to enjoy under the fairware concept are the ones who are keeping us all going. If you look at the price of commercial programs for other computers which do the jobs that we are able to do with our fairware programs, you will find that $100 will not buy more than 1 program and you may well need $1000 to get a state of the art program. Fairware software costs the price of a disk initially but if we use the program the onus is on us to send a contribution to the authors to repay them for their efforts and encourage them to continue development and perhaps write a new program as well. We can be sure that these authors are not relying on our contributions to live, as they do not ask enough and we do not send enough, if anything at all.

TIsHUG now offers us alternatives to sending the money direct to the author. Of course sending the money direct to the author is the best way to get on an author's mailing list and to ask some pertinant questions about the software or about improvements which might be made in the next release. Another method is to contribute each month by mail or in person to the monthly fairware collection or to send in a contribution to be spread amongst several products and their authors. If you use this last method, be sure to send in a list of software and the amounts for each.

The Fairware software product for this month is DM1000, which comes from a user group, the Ottawa user group in Canada's national capital. Everyone with a disk system must have used this product at some time and I cannot remember my computer system without DM1000. Other similar programs have come along and in some respects replaced DM1000 but if you have ever used it then you should have made or now should make a contribution towards its development and possible evolution. Note that DM1000 is supplied with Funnelweb on the fairware principle. They are separate products and if you use both then you should contribute something for both products. Just get your contribution into TIsHUG as soon as you can and they will ensure that all of it reaches Ottawa. Do not forget the other authors of fairware software that you use and take up TIsHUG's offer of forwarding your contribution on to them. O

Secretary's Notebook

by Terry Phillips

A nice day brought members out in droves to attend the Buy, Swap and Sell day held at the August meeting. Judging by the tables loaded with software and hardware, and the number of smiling sellers and buyers the day was certainly a success. This type of activity appears very popular, so if you would like a repeat performance have a word in Craig Sheehan's ear. He might just be able to organize an event along similar lines in the not too distant future.

Les Andrews, a member of the group's Publicity Committee, placed an advertisement in the Weekly Trading Post, advertising the August meeting. The result; an outstanding success! Three people who read the advertisement turned up at the meeting and joined TIsHUG on the day.

An information sheet on the group's activities has also been produced and copies given to TI Australia who will make them available to people bringing in consoles and hardware for repair. This method of advertising the group will also, hopefully, bring in a few new members. If you have an idea for advertising for prospective members please see any of the Directors at the meetings or contact one of us by mail or telephone.

Speaking of the new members, it is a big welcome to the following:

Steven Newland - Alice Springs
John Korpusinksi - Canley Vale
Gus Martinez - Ermington
Mark Nielsen - Ryde
Phil Thomson - Keiraville

Hope you all enjoy your membership of the group.

Amongst the new members is a name some of you may remember. Mark Neilsen was a member some years back and was instrumental in getting the BBS up and running. Great to see you back with us Mark.

And, continuing on the subject of membership, a new class of membership was approved by the Directors at their meeting on the 4th of August. This class of membership to be known as Associate Member. An Associate Member will be entitled to social privileges of the group, however will not be eligible to receive the monthly News Digest. An annual membership fee of $10 shall apply for Associate Membership. This class of membership was agreed to as there are a number of people, mainly friends of members, who like participating socially at meetings however in most instances they do not own a TI99/4A, so our published information is not of great benefit to them. If you know someone in this category bring them along and sign them up for membership. In order to accommodate this class of membership it was necessary to frame a new By-Law in accordance with Paragraph 9 of our Articles of Association. All members should add the following new By-Law to their copy of our booklet: Memorandum and Articles of Association:

By-Law 19 - The club will admit Associate Members who shall be entitled to social privileges, however this class of member shall not be entitled to receive monthly issues of the TIsHUG News Digest. Fees for this class of member shall be as set from time to time by the Board.

As from our next meeting it is hoped to kick off a games room in the small upstairs room near our main meeting room. This is at the request of some of the younger members who are not so interested in other group activities discussed at meetings. With a bit of luck we will be able to set up a couple of computers in this room and let the younger, and young at heart, members have a bit of fun during the afternoon.

A letter has been forwarded to other major computer user groups on the subject of a proposed home users computer fair either to be held late this year or early during 1990. I have written to Apple, Amiga, Commodore etc., and now wait on a response. If this does get off the ground it sounds like it could be a major activity. Keep watching for further information as it comes to hand.

If any member happens to be on the east coast side of the USA during September, then I am advised that the Mid Atlantic Ninety-Niners will be holding a TI99/4A International Expo on Saturday September 16 1989 at the Howard Johnson Inn, 5821 Richmond Highway, Alexandria, Virginia from 9am to 5pm. It looks like it will be a very big event judging by the pre-show publicity they have mailed.

Here are a few items for sale. Contact the members involved for further details.

Percy Harrison - telephone 8083181 - has a Cicada 312 Modem complete with press button redial hand set . In perfect condition and for sale at $140.

Paul Barton - 4 Northrop Street Raby NSW 2566 - seeks assistance in finding a good home for the following:

Grey Console with built-in 32K $75, PEB $250, DSSD disk drive $75, TI disk controller $75, RS232 card $100, PIO cable $10, RS232 cable $10, Horizon RAMdisk $450, Editor Assembler manual $25, TI-Writer manual $20, CTI CPB80 dot matrix printer $200, RGB interface $10, Modified joysticks $10, Extended BASIC $20, PRK module $10, Meteor Multiplication, Video Games 1, Blasto, Disk Manager 2, TI Invaders, Munchman, Integers, Alligator Mix at $5 each. Also a collection of disks in a lockable disk box, plus cassette programs and a cassette recorder with TIsHUG News Digests going back to 1983.

That is it for this month. See you at the meetings, and please remember to bring along your membership card as these should be shown when making purchases from the shop. This is necessary because of Taxation Office rules. O

---

continued from page 12

```
    count = 1;
    do {printf("%d\n",count);
        count = count + 1;
       }
    while (count<10);
    exit(0);
  }
```

Figure 5b - The do .. while loop.

```
/* A "for" loop */

extern printf();

main()
  { int count;

    for (count=1 ; count<10 ; count++)
      printf("%d\n",count);
    exit(0);
  }
```

Figure 5c - A for loop.

Whilst all of these programs produce the same output, they all show some of subtleties of the 'C' language. A "while" loop only executes the statement following it if the condition being tested, in this case if "count" is less than ten, is true. Once the statement has been executed, the process is repeated until the condition becomes false. "do .. while" loops are slightly different in that the statement is executed first, and then the test is made. This is done until the condition is false. continued on page 11

TIsHUG Software Column

by Terry Phillips

Advice has been received from Dennis Faherty, Inscebot Inc. of a soon to be released program titled TI-Sort. An advance copy, minus any documentation, has been received and it certainly looks a very professional piece of software. Its prime use will be in sorting large data bases created with TI-Base, where its sort routines are much more efficient and faster. As a companion disk to TI-Base it is highly recommended. 10 copies have been ordered and should arrive soon. Likely cost is around the $20 mark. Dennis also advised that Chris Faherty is currently working on the latest release of TI-Artist which has changed sufficiently to be almost a new product. It will contain such things as shrink, expand, rotate and a greatly enhanced print capability. It should be ready sometime later this year.

Some further copies of TI-Base and TI-Artist have been received and placed in the shop for sale. Get in early if you want a copy of either. TI-Base is $25 and TI-Artist is $20.

Jim Peterson has written to advise that he has reduced the costs of his 3 Nuts & Bolts disks to $10 (down from $15) and his 5 Tips from the Tigercub disks to $5 each (down from $10). As your group is selling Jim's range of software to members, these reductions in prices are also applicable here. As I have said before these disks are very highly recommended and I await your orders.

Still not much happening in the way of new software receipts, however I have placed 4 new disks in the library and these commence the new "B" series of disks.

DISK B1 - Pictures from Darren Telford. This disk was previously numbered A358.

DISK B2 - Tigercub Disk - contains Tips #58, a file on setting up printers and a great addictive game called TI-Tris.

DISK B3 - Melbourne TIMES disk - contains files included in their March News Digest .

DISK B4 - Melbourne TIMES disk - contains files included in their April/May News Digest .

Nothing more to advise this month. If you need something from the library give me a call or write to the club's address. O

The Bug Report

by Geoff Trott

I am one of the fortunate few who have been running a hard disk on a TI99/4A (cou r of you all) for some months now. I am using a DSR)M numbered H11 and Myarc Disk Manager 5 version 1.29. I have to say that overall I am very pleased with the operation of both the disk controller and the disk manager program for all the editing, spell checking and formatting that I do each month for the TND and would not like to do without the hard disk now. I must also say that I am using a Mechatronics 80 column card, which may or may not cause some of the problems I am about to describe. I would just like to put on record my experiences with this major piece of equipment for others to ponder and perhaps for those better placed than I to do something about fixing.

First there are the programs that do not work properly with the hard disk. If programs are written to allow the entry of complete names of files and not just a disk number and file name then they should be able to use the hard disk with little trouble. Unfortunately authors try to make it simpler for the user by just allowing a number to select a drive. This is not a bug in the controller card, but is just a nuisance for someone with a hard disk. However sometimes I have found that using floppies on the controller card with some of these programs can cause problems. For example, the original PRBase will not work as the controller card thinks the disk is not formatted or formatted in an impossible way. This is because the controller card reads the information on sector 0 to find out how to set itself up to read the disk. Multiplan will not produce a catalog of the disk with the arrow keys in Transfer Load. Then there is the problem of fractured files.

Using floppies once again, either real ones or emulated ones, if your files start to fracture then the controller seems to lose track of which sectors are used by which file and overwriting of files can occur. This has occurred to me using Multiplan, which I use for generating the index each month as well as keeping track of the magazines which have been reviewed each month. This means that I am constantly adding data to files and creating new files which is guaranteed to produce fractured files. I found that I had a number of files on my floppy which were no longer readable by Multiplan and when I looked at the file information blocks with Disk Utilities I found that the files were using the same sectors, but only where the files were fractured. That is the first part of the file was OK, but the piece stored somewhere else on the disk was also used by some other file. Then the same thing happened while writing and debugging a c99 program. This involves creating a source file with the editor, compiling it which generates an assembler source file, assembling this to produce a D/F 80 object file which is then loaded into memory. If all these files are stored on the same disk and then the source is changed and added to, fractured files are produced at every step. This caused the c99 compiler no end of problems as it wrote over the fractured end of the source file. Not very nice! If you know that there is this problem then you can make sure that you do not produce fractured files by deleting all the intermediate files before editing the source.

The other main problem occurs with DM5 and copying files from floppy to hard disk. Sometimes the Disk Manager will not copy a file from floppy to hard disk. It just hangs up! If you copy the file to RAMdisk and then to hard disk it all works! There does not seem to be any rhyme or reason for this one.

Ben Takach has mentioned the lack of a viable backup mechanism, although a second hard disk may be a good option, but another lack is that of a good exerciser program to locate and flag sectors which are marginal and may give trouble in the future. I have had a hard disk which developed bad sectors and it has proved very difficult to recover the contents of that disk because the only program that will allow files to be copied is DM5 and this requires that the bit map sectors be readable in order that it can calculate the number of sectors used. One of those sectors is bad on this disk and so the program just hangs. Not very satisfactory for me. The strange thing is that all the programs will execute or can be read, but I cannot write anything to the disk as this requires access to the bit map sectors. If bad sectors are such a problem there should be a way of checking out the disk more thoroughly than is currently possible.

Please do not be put off by all these problems. I have not been as I have bought myself a hard disk system which my son is currently trying to fill up with all the games he can lay his hands on and to get running from a set of menus. Perhaps I should get him to write an article about this when he has finished. Also there is a very useful utility which every user of a hard disk should have and use. It allows the hard disk to be safely parked before turning it off, which should prolong the life of the disk. O

Techo Time

with Lou Amadio

The August Swap and Sell meeting turned out to be a fun day. A great deal of hardware and software changed hands on the day, and this is how club members get to try different items. I do not know how many people were there, but the room was packed, and not many items were left unsold. We need to have at least two such meetings per year.

Correspondence

Two letters were received this month:

The first from Les Moore (Christies Beach, SA) pointing out the errors in the Direct I/O Interface (see errata below). Thanks Peter.

The second from Michael Ball (Nowra, NSW) asking if a console power supply is suitable to power the Direct I/O Interface. In a word, no, but the console power supply may be used to power a stand-alone floppy disk drive. Michael is also having intermittent console lock-up problems. From his letter it sounds as if his modules and/or module port socket in the computer need cleaning. Module PCB contacts are easily cleaned using a cotton bud moistened with alcohol. The module port may be initially cleaned with a piece of cardboard cut to size and very lightly moistened with alcohol. The cardboard is inserted into the edge card socket inside the module port, cut back and re-inserted as necessary to clean the contacts. If this fails to cure the problem, then the computer will have to be dismantled and the port washed out with alcohol.

Hard Disk Display

The internals of a hard disk drive were displayed at the August meeting. The display aroused a lot of interest among members so we will be redisplaying the unit at the September meeting, hopefully powered up and running.

Hardware Hints

In the past, I have advocated the use of power resistors in order to reduce the voltage to a circuit and so improve heat dissipation. If, however, the current load is not likely to be constant, resistors will not provide the correct voltage drop. It is better in these circumstances to use a power zener diode of the appropriate voltage rating. For small voltage drops (for example, 1 to 2 volts), you could use one or more power diodes (3A rating). At a current of 1 amp, power diodes will drop approximately 1 volt each.

Errata: Direct I/O Interface, TND July '89

Unfortunately gremlins have struck again and caused an error in the wiring instructions for the Direct I/O Interface article which appeared in the July '89 issue of the TND. Please make the following corrections to your original article on page 7:

I/O pin 8 (A11) connects to PEB pin 34

I/O pin 17 (A7) connects to PEB pin 38

Thanks to Les Moore (ATTIC) for pointing out the errors, and I apologise for any inconvenience that it may have caused anyone who was building the interface.

Please note the additional errata on this subject published in the August '89 issue of the TND.

The power supply for the I/O interface is easily modified to allow it to power a floppy drive as well as the Interface. The modification will be described next month and includes 2A transformers, in lieu of 1A, and the addition of a 7812 (+12V) regulator to the +18V rail.

Errata: Special Function Key Emulation - August '89, page 28.

The diodes across the relays are 1N4148 high speed devices. A capacitor (0.1 uF) should also be mounted across each diode.

Hardware Projects

I welcome any hardware contributions from club members for this column. If you share your ideas and experiments, we can all benefit.

This month we have contributions from two authors. Derek Wilkinson has written an article on how to convert an IBM style keyboard to interface directly with the console. With a little luck, we should have a photograph included in this issue of the TND. A keyboard matrix table is included to aid in the wiring process. Derek also suggests a test circuit to assist in making the correct connections. If you decide to go ahead with this project, make sure that the keyboard that you intend to modify has switch contact keys and that the switches can be disassembled and converted to dual key action (for FCTN and CTRL sequences).

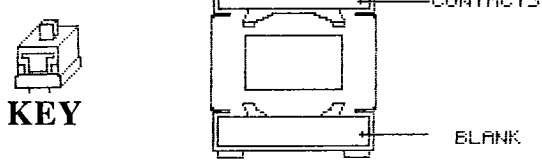
The other contribution comes from Geoff Trott and will no doubt be greeted with a sigh of relief from owners of the Peter Schubert Mini PE System which includes a RAMdisk. Geoff has modified the power supply to prevent corruption of the RAMdisk ROS during power up. O

Interface to an IBM style Keyboard

by Derek Wilkinson

First you need a style of keyboard that uses contacts that provides a short circuit output when they are operated. Then by rearranging the wired matrix of the keyboard you can arrange the resulting matrix to provide the same 15 outputs provided by the TI99/4A keyboard.

The function keys require a little more work, in that each function key requires double key operation. This can be obtained by modification of the function keys to operate twin contacts. The main drawback being that you will require extra keys. Spare keys are available from places like Sheridans in Redfern. The keys are usually designed to allow for twin contacts to be mounted where a blanking piece is inserted in the spare side of the key.



Makeup of Keys

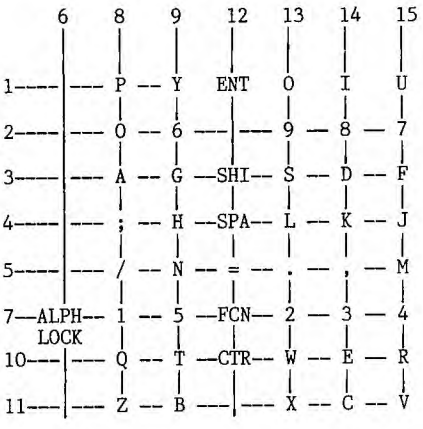
To allow for all the various function keys to operate from one key you will reqire an extra 26 keys. If you do not require all at once they can easily be added later. You can also configure control funcions in the same way. I have the Escape and Tab keys configured.

You will first have to remove the electronic parts from the keyboard. Some of the tracks on the board need to be cut and be restrapped to provide the correct matrix wiring. This job is not hard but can be very time consuming. A small rotary engraving tool was found to be ideal for cutting tracks and drilling holes in the backboard for the extra contacts.

One obstacle I encounted was the replacement of the electronic capitals lock key by a mechanical one. These are hard to come by without pulling one out of a TI99/4A console.

When reconfiguring the keyboard matrix use a different colour wire for each of the 15 leads of the matrix. This will allow ease of tracing when something goes wrong or you want to add extra keys to the keyboard.

KEYBOARD MATRIX

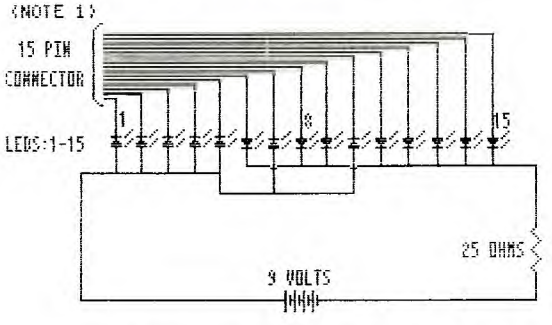


MATRIX COLOUR CODE

1	WHITE
2	RED
3	ORANGE
4	YELLOW
5	GREEN
6	BLUE
7	BLACK
8	GREY
9	RED-GREEN
10	LIGHT BROWN
11	BROWN
12	PURPLE
13	PINK
14	RED-BLACK
15	RED-YELLOW

I have been using the keyboard now for over 12 months. The problems encounted have been a couple of sticking keys and getting lost when I use the normal TI99/4A keyboard. I used a 25 pin D connector mounted in the side of my console, this allowing the use of the original keyboard if required without stripping the console.

I also made up a keycheck indicator to allow ease of checking the matrix and key operation. It is a simple cicuit but saves a great deal of time when you are having problems.



Keyboard Matrix Checker

1. CONNECTOR TO MATCH YOUR KEYBOARD.
CHECKS CORRECT MATRIX OPERATION OF EACH KEY.

see photograph on page 31

Function	FCTN key + key	Function	FCTN key + key	Function	FCTN key + key
Delete Char	1		0	{	F
Insert Char	2	Quit	+	}	G
Erase	3	~	W	\	Z
Clear	4	[	[	`	C
Begin	5	T	]	|	A
Proceed	6	_	U	Arrow up	E
Aid	7	?	I	Arrow down	X
Redo	8	'	O	Arrow left	S
Back	9	"	P	Arrow right	D

Similarly the control functions may be reproduced. O

RAMdisk Power up problems

by Geoff Trott

I was asked by a member to look at his miniPE RAMdisk which consistently lost its ROS when power was turned off. He was also having problems with floppy disks being corrupted by his system, which is probably another story. I had heard of at least one other similar problem and was wondering what the problem was and if there was an easy solution. When it arrived I first checked the memory chips which were OK and then made two modifications to the circuitry.

The first of these was to install a pullup resistor on the WE (Write Enable) line to make sure that it was high when the power was removed (battery backup). Then I removed the components (resistor, diode and capacitor) that were provided for the power up reset and wired the reset to the reset line out of the console, which is a suggested modification for Horizon RAMdisks. These modifications did nothing to help with the problem so I started looking further.

Using a storage oscilloscope I looked at the power from the external power pack which was used for the supply when it was turned on. I noticed that it took about 3 charge cycles (30 milliseconds) before the voltage reached a stable value and sure enough, during this time signals like WE were changing level. This seemed to me to point to a problem so I installed a switch after the 3 terminal regulator to test this out. Even with switch contact bounce there was no loss of data. So I then designed and built a little circuit which waits for a while (30 milliseconds) after the input voltage exceeds 6 volts and then turns on the regulated voltage as quickly as possible. As well as that I included a 5uf tantalum capacitor on the output of the 3 terminal regulator to stop it oscillating. This then enabled the power to be turned off at any time without any loss of data. I wonder if the same problem arises sometimes with the Horizon RAMdisk card in the PEB!

see circuit diagram on page 32 O

Powering Your Hard Disk Drive

by Lou Amadio

What? Not another power supply!

If you are wondering why there have been so many articles on power supplies lately, it is because that is what I have been working on. Consequently, it is much easier to decide what to write about, especially in the absence of any specific requests from club members. Anyway, since this is a special "Hard Disk" issue, it can be considered a timely article.

So, here we go again.....

Last month I described how to build a power supply and enclosure for floppy drives. Although a hard drive will fit in the same physical space as a floppy, its power requirements are quite different. In the past few weeks I have been testing the power requirements of a number of hard drives. The results are presented below:

Make	Model	Cap	+5V	+12V
NEC	D5124	10Mb	0.7A	2.5/1.1A
NEC	D5126	20Mb	0.6A	2.5/1.4A
Shu	DE4766	10Mb	0.7A	2.5/0.5A
MS	#3425	20Mb	0.45A	3.0/1.2A

(MS=Miniscribe, Shu=Shugart. First figure for +12V is the start current).

If you compare these figures with those published last month for floppy drives, you will see that hard disks require approximately 50% more power. More

importantly, however, we need more than twice the current capability for the 12 volt rail since a hard drive has more inertia during start up. Consequently the motor draws much more current during the first few seconds while the disk comes up to speed. For example, a typical hard drive will draw approximately 2.5 amps from the 12 volt rail at switch on, falling to approximately 1.2 amps when the drive reaches full speed. Unless the power supply can deliver the initial high current, the drive will simply not start.

Because of the wide range of current requirements, I decided to construct the power supply using two transformers, one for each rail. This ensures that one supply does not affect the other, especially during the critical start up phase. It is also likely that two smaller transformers will be cheaper than one multiwinding high current transformer, since the smaller transformers are mass produced.

The 12 Volt Rail

Referring to the circuit, the +12 volt rail is powered from a 2 amp 15 volt AC transformer. The transformer is used with a full wave bridge rectifier to produce 21 volts (unloaded) across the filter capacitor. Under maximum load, during start up, the unregulated voltage feeding the 12 volt regulator drops to about 15 volts, rising to about 17 volts when the drive is up to speed. This is a satisfactory voltage margin for the regulator. The current capability, however, needs to be boosted as the voltage regulator on its own can only handle approximately 1 amp maximum before it shuts down. We need to boost the output to at least 3 amps or more. This is the purpose of the pnp power transistor, and this combination is much cheaper than using a high power regulator. The only drawback in using a transistor in this configuration is that we lose the current overload protection inherent in the IC regulator.

Power dissipation in the transistor is 1.25*(17-12) = 6.25 watts. The transistor (and the 12 volt regulator) should therefore be mounted on a suitable heat sink. This will ensure that the thermal characteristics of the regulator will still provide some protection should an overload occur.

The 5 Volt Supply

The 5 volt supply is a little simpler in that the regulator can handle the full current requirement on its own. The transformer is used in the centre tap mode (0-6.3-12.6) with 2 diodes so it only needs a moderate heat sink and can simply be bolted to a convenient point on the metal chassis. If your hard drive behaves erratically, it may not be getting enough voltage on the 5 volt rail. In this case simply use the 0-7.5-15 volt tappings on the transformer.

Parts List

1 x Arlec 15V/2A multitap transformer (PT6978)
1 x Arlec 15V/1A multitap transformer (PT2155)
1 x mains fuse holder, 0.5A fuse
1 x 240V DPDT power switch
1 x finned Al heat sink
2 x 1A power diodes
4 x 3A power diodes
1 x 2200 uF, 16V electrolytic capacitor
1 x 4700 uF, 25V electrolytic capacitor
4 x 4.7 uF TAG tantalum capacitors
1 x 4.7 ohm, 1 watt resistor
1 x MJE2955 pnp power transistor
1 x 7805 5V positive regulator
1 x 7812 12V positive regulator
1 x disk drive 4 pin power plug (female)
1 x 240V power lead, grommet and cable clamp
4 x adhesive rubber feet
1 x Aluminum box to suit
Miscellaneous wire, bolts etc

Construction

see circuit diagram on page 31

The easiest way to build the power supply is to use point to point wiring. Secure the power cable to the box with a suitable grommet and cable clamp. Make sure that the green/yellow wire in the mains cable is securely earthed to the box. Mount the power fuse (and switch if required) at a convenient location on the metal box taking into account the position of the transformers. If you choose not to use a power switch, the brown wire (mains active) should go to the fuse first and then to each transformer primary in turn. The blue wire (mains neutral) can be connected directly to the other end of each transformer primary. Mount the transformers and solder the power diodes directly to the transformer lugs and then to the appropriate filter capacitor. (The diodes are capable of supporting the filter capacitors).

The 5 volt regulator can be mounted at any convenient location together with the TAG tantalum capacitors (4.7uF) on its input and output legs. Be careful with the polarity of the electrolytic capacitors. Terminate the +5 volt and ground leads to the disk drive power plug as indicated on the circuit diagram.

The 12 volt regulator, TAG capacitors, 4.7 ohm resistor and power transistor are all located on a heat sink prior to mounting to the metal box. Use point to point wiring, making the connections as short as possible. As this supply must be capable of delivering up to 3 amps, use suitable wire between the filter capacitor, power transistor and the disk drive power plug. The collector (metal tab) of the transistor is at 12 volts, so it must be insulated from the heat sink with a mica washer. You must also insulate the bolt, which secures the transistor to the heat sink, with a plastic eyelet. Connect the +12 volt output and ground leads to the disk drive power plug as indicated on the circuit diagram.

Power Up

Double check all of your wiring and connections, especially the polarised devices (diodes, capacitors and transistor). When you are satisfied that all is well, power up and check all the voltages shown on the circuit diagram, especially the position of the +5V and +12V at the output plug. Finally plug in your hard disk, switch on, and, as a final ckeck, measure the +5V and +12V voltages again on the drive printed circuit board after the drive comes up to speed.

Footnote

I was talking to Garry Christensen (TIBUG) recently and he mentioned that he had installed his hard disk into his PEB. He had modified the power supply in the PEB by installing an additional 12 volt regulator, specifically to power the hard disk. He also mentioned that he did not have any problems with his HFDC card which are normally attributed to excessive voltages in the PEB. (Refer Ben Takach, "PE Box Unregulated Voltage Levels", TND March 1989). My theory is that the extra current drawn by the hard disk is reducing the unregulated voltages in the PEB to a satisfactory level.

Modifying The PEB for Hard Disks

An article on modifying the PEB power supply with an additional 12 volt regulator was first published in the March 1988 issue of MICROpendium. The author (Eric Bray) provided step by step instructions on how to add the additional circuitry. In the article, Eric specified a 1 amp regulator (7812) for the hard drive. However, my experience with a small number of hard disks suggests that this device may not be up to the task, and you should substitute a high current regulator or adapt the circuit presented above.

The other area of concern would be the ability of the PEB transformer to supply sufficient current for PEB cards as well as a hard and floppy disk. Naturally this would also be influenced by the number of slots in use in the PEB. In my opinion, the safest way is to build the power supply described above. You should also read Ben Takach's article (see above) before you use your Myarc HFDC card.

Beginners' Corner

by Chris Bobbitt, from Asgard News

This month we are going to discuss data base concepts.

One of the most popular uses for a computer (only after word processing) is "DATA BASE MANAGEMENT". I put this in quotes for very good reason; this term is often misused, rarely defined, and has been so maligned for so long it is practically meaningless. However, there is a definition for it. Data base management is the practice of managing (running, or otherwise controlling) a collection of related data. Data itself can be short lengths of text, number, prices, or really any little bit of knowledge that you would like to store on a computer. Data can be anything from the stuff you write on a PD99 form to the birthdays of your friends and relatives to their names.

A data base, as I said, is a collection of related data. What do I mean by "RELATED?" Well, for instance, a single data base might contain the names and addresses of your friends. It will most likely not contain the pedigrees of of your dogs as well. In a data base, as in life, things tend to be organized together. You could put the pedigrees of your dogs with the names and addresses of your friends (perhaps), but it hardly makes sense. More likely, you might want to have a list of your friends names and their phone numbers as well. That would be related data. We can make an assumption that the only purpose of putting data in a data base in the first place is so that you can get it out later in some coherent fashion. Mixing different data together is not going to promote that. So, a data base is simply a collection of data that seems like it "goes together".

A data base program (often called, improperly, simply a "data-base"), is a program designed to "MANAGE" your data base. What do I mean? Well, to be more specific, it will typically allow you to type data into a data base, find items of interest in the data base, perhaps sort the data in some order, and eventually, print out part or all of the data. There are many types of data bases "Flat files", "relational" (and if you are an old timer), "network" and "hierarchical". But, I am getting ahead of myself now.

As you might have guessed, a data base program is usually actually many little programs, which together are often called a "DATA-BASE MANAGEMENT SYSTEM" (or, in Government-speak, a DBMS). DBMS can be as simple or as complicated as the programmer wants them to be. Some are so complicated that you need a very sophisticated "language" to talk to them in, you need to type in commands in an order that the systems understands in order to accomplish even the most basic tasks (like type data in). The advantage to something that difficult? Well, usually there is a trade off between power and ease of use. Something that is really easy to use usually cannot do anything really sophisticated, and something really hard to use really is not worth using for anything simple, but is the only way to do some complicated stuff.

The two most common types of data base management systems in the TI99/4A and the Geneve world are "FLAT FILE" and "RELATIONAL". Actually, to be more precise, the most common type of TI99/4A data base is the "flat file" method. There are only two relational data base management systems for the TI99/4A (albeit, two of the more popular such programs). The other two types have actually never been developed for the TI99/4A. There is little loss, though, a relational data base is usually as capable as those two ways of managing a data base.

What is the difference between the two? Well, hundreds of articles have been written on the subject, but it basically boils down to the way data is stored and retrieved. In fact, storage and retrieval are the entire reason for a DBMS. Everything in a DBMS is somehow related to put data in the computer (or the data base), and getting it out again. So, when we say that is the "only" difference, we paint it with a very broad brush.

Before you can understand the difference between these two systems, it helps to understand some more basic terminology. A data base system can be viewed on three levels; the physical, the user's view, and the conceptual. The physical level is the way the computer views the data base; it is how the computer actually stores the data in the data base. The user's view is simply how the data is significant to you when you type it in, as perhaps a listing of all your computer equipment. The conceptual view is best understood as the halfway point between the two views. It is the view that both the computer and the user can understand. A typical user usually does not understand (and probably does not care) how the DBMS stores the data. The computer, of course, can hardly understand human thought processes or precepts. The conceptual level is the bridge between the two. It is typically the level in which the user interacts with the DBMS and the data base.

The conceptual level is just that, an easy to understand concept of the data base. A data base is typically conceptualized as a "table". A data base containing your friends names and phone numbers might look like this in a table

Name	Phone Number
Dick	918 8132
Percy	808 3181
Peter	528 8775
Steven	608 3564

Fortunately, a computer does not have too hard of a time understanding what a table looks like either. Hence, both you and the DBMS understand the data.

Now let us go into what a data base itself actually is. If you examine the table a little further, you will notice that it is in two columns; one labeled "NAME" and one labeled "PHONE NUMBER". In DBMS lingo, those columns are known as "FIELDS". A field is exactly like a column in a table. The rows of the fields are the data in the data base (each individual row is known as a /TUPLE", pronounced "TOO-PULL"). Tuples are also called "RECORDS" in some data base programs. When you create a data base, you have to tell it what "FIELDS" you want, what type of fields they are (you will notice that the "NAME" field is all text, while the "PHONE NUMBER" field is all numeric), and how much space you should give each. Once you have told the data base this information, it creates on your data base disk something called a "SCHEMA" (pronounced "SKEEMUH"). The schema is the "DEFINITION" of the data base.

Once you have created the data base definition (or schema), consisting of all your columns (fields) and their lengths and types, you begin the process of actually creating the data base. This is the laborious part of the procedure where you actually type in all the data into the data base. The part of the program where you type in the data in your data base is typically called a "DATA ENTRY SCREEN". This is a bit literal; in the data base world it is called a "FORM". Some sophisticated data base programs will allow you to design the data entry screen, or form, even down to placing the fields exactly where you want them on the screen, and even the screen colours.

In some data base programs, the "FORM" is made to be part of the "SCHEMA". As you type the data into the data base through the form it is stored by the DBMS in another format that is more readily managed by the program. Again, that format is the physical view of the data base, or the way the DBMS sees your data in your data base.

continued on page 18

Extended BASIC Tutorial

by Tony McGovern, Funnelweb Farm

I. Introduction

These Tutorials were originally written several years ago to try to improve the abysmal standard of programming apparent in magazine and User Group Newsletters, and even commercial game programs at the time. They appeared first in the Sydney TIsHUG NewsDigest and later ones in the Hunter Valley 99 Newsletter after the formation of that group as a separate entity. The standard of published programs written in Extended BASIC has on the whole improved since then, but not as much as it should have, and as advanced users move on, new beginners take their places on the basic console with Extended BASIC computers and need to learn how to use them. Extended BASIC is no longer the prime language on the expanded TI99/4A, but it is still a very expressive and powerful language, with features that have only been caught up with in recent times on newer and more pretentious computers. There does seem to be some continuing demand for tutorial material on Extended BASIC, so I hope this series will still be of value to new and old programmers.

The series is intended neither as an elementary course for absolutely raw beginners nor as a reference treatise. It is meant for the interested user who is willing to put some effort into understanding how Extended BASIC works in order to make best use of it, and wishes to develop a feel for how the machine actually goes about its business. Plus assorted ravings and ramblings on. This issue of the Tutorials represents a mild going over of the original files, mainly to remove some of the outdated topical material.

The aim of this series on TI Extended BASIC was always to concentrate on those features which had not received due attention in User-group newsletters or commercial magazines. In fact most of the programs published in these sources up to the original time of writing had made little use of that most powerful feature of Extended BASIC, the user defined sub-program, or of some other features of Extended BASIC. Part of the reason for this was the practice of some authors, a practice still sometimes in evidence in magazines, of rehashing the same material for several different machines and publications, forcing the use of the lowest common denominator of BASICs. Worse still were the many programs which were object lessons in how to write tangled and obscure code. A much neglected source of help is TI's Extended BASIC Tutorial tape or disk. The programs in this collection are unprotected and so open for inspection and it is worth looking at their listings to see an example of how sub-programs can give an easily understood overall structure to a program.

Well, what are we going to talk about then? Subjects covered are:

(1) User-defined sub-programs
(2) Prescan switch commands
(3) Bugs in Extended BASIC
(4) Crunching program length

Initially the discussion will be restricted to things which can be done with the console and Extended BASIC only. Original intentions were to cover LINKing of assembly routines but the series petered out before that. Ross Mudie has written extensively on this topic in a series of TND articles in recent years so the gap has been filled. Actually, for most game programming in pure Extended BASIC with no assembler help, the presence of the memory expansion does not speed up Extended BASIC all that much, as speed still seems to be limited by the built-in sub-programs (CALL COINC, etc.) which are executed from GROM through the GPL interpreter. The real virtue of the expansion system for game programming, apart from allowing longer programs, is that GPL can be shoved aside for machine code routines in the speed critical parts of the game, which are usually only a very small part of the code for a game. Even so, careful attention to Extended BASIC programming can often provide the necessary speed. As an example, the speed of the puck in TXB, our first major exercise in Extended BASIC coding, is a factor of 10 faster in the finally released version than it was in the first pass at coding the game. Curiously it has turned out that the coding of this game is so finely fitted to the way TI Extended BASIC handles things that the puck speed has to be reduced for Myarc Extended BASIC to handle it without losing track, even though it is supposed to be much faster.

II. Sub-programs in Overview

Every dialect of BASIC, TI Extended BASIC being no exception, allows the use of subroutines. Each of these is a section of code with the end marked by a RETURN statement, which is entered by a GOSUB statement elsewhere in the program. When RETURN is reached control passes back to the statement following the GOSUB. Look at the code segments

```
290 ....
300 GOSUB 2000
310 ....
2000 CALL KEY(Q,X,Y):: IF Y=1 THEN RETURN ELSE 2000
```

This simple example waits for and returns the ASCII code for a fresh keystroke, and might be called from a number of places in the program. Very useful, but there are problems. If the line number of the subroutine is changed, other than by RESequencing of the whole program (and many dialects of BASIC for microcomputers were not even that helpful at the time) then the GOSUBs will go astray. Another trouble, which you usually find when you resume work on a program after a lapse of time, is that the statement GOSUB 2000 does not carry the slightest clue as to what is at 2000 unless you go and look there or use REM statements or tail remarks. Even more confusingly the 2000 will usually change on RESequencing, hiding even that aid to memory. There is an even more subtle problem; you do not really care what the variable "Y" in the subroutine was called as it was only a passing detail in the subroutine. However, if "Y" is used as a variable anywhere else in the program its value will be affected. The internal workings of the subroutine are not separated from the rest of the program, but Extended BASIC does provide four ways of isolating parts of a program.

(1) Built-in sub-programs
(2) DEF of functions
(3) CALL LINK to machine code routines
(4) User defined BASIC sub-programs

The first of these, built-in sub-programs, are already well known from console BASIC. The important thing is that they have recognizable names in CALL statements, and that information passes to and from the sub-programs through a well defined list of parameters and return variables. No obscure Peeks and Pokes are needed. The price paid for the power and expressiveness of TI BASIC and Extended BASIC is the slowness of the GROM/GPL implementation in which the powerful TMS9900 CPU is hobbled by being forced to interpret another language (GPL) for an imaginary 8-bit processor.

DEF function is a primitive form of user defined sub-program found in almost all BASICs. Often its use is restricted to a special set of variable names, FNA, FNB,.., but TI BASIC allows complete freedom in naming DEFed functions (as long as they do not clash with variable names). The "dummy" variable "X" is used as in a mathematical function, not as an array index

```
100 DEF CUBE(X)=X*X*X
```

does not clash with or affect a variable of the same name "X" elsewhere in the program. "CUBE" cannot then be a variable whose value is assigned any other

way, but "X" may be. Though DEF does help program clarity it executes very slowly in TI BASIC, and more slowly than user defined sub-program CALLs in Extended BASIC.

CALL LINK to machine code routines goes under various names in other dialects of BASIC if it is provided (for example USR( ) in some). It is only available in Extended BASIC when the memory expansion is attached, as the TI99/4A console has only 256 bytes of CPU RAM for the TMS9900 lurking in there. All we note now is that the TI99/4A does it in a very civilized fashion, LINKing by name to relocatable assembly routines. Ask your PC or Apple or C64 owning friends if their BASIC supports that. The Funnelweb system supports all necessary assembly development functions with the Extended BASIC module in an expanded system. TI Extended BASIC contains a standard set of interface routines that support the details of linking. Unfortunately these have a great weakness in that transfers of strings between Extended BASIC's VDP storage and the memory expansion is done byte by byte with a complete reset of the VDP read and write addresses for each byte instead of transferring a block at a time. The result is that assembly string handling does not always speed things up all that much and this has been a reason for lack of success of some Extended BASIC support packages.

You should have your TI Extended BASIC Manual handy and look through the section on SUB-programs. The discussion given is essentially correct but far too brief, and leaves too many things unsaid. From experiment and experience I have found that things work just the way one would reasonably expect them to do (this is not always so in other parts of Extended BASIC). The main thing is to get into the right frame of mind for your expectations. This process is helped by figuring out, in general terms at least, just how the computer does what it does. Unfortunately most TI99/4A manuals avoid explanations in depth presumably in the spirit of "Home Computing". TI's approach can fall short of the mark, so we are now going to try to do what TI chickened out of.

The user defined sub-program feature of Extended BASIC allows you to write your own sub-programs in BASIC which may be CALLed up from the main program by name in the same way that the built-in ones are. Unlike the routines accessed by GOSUBs, the internal workings of a sub-program do not affect the main program except as allowed by the parameter list attached to the sub-program CALL. Unlike the built-in sub-programs which pass information in only one direction, either in or out for each parameter in the list, a user sub-program may use any one variable in the list to pass information in either direction. These sub-programs provide the programming concept known as "procedures" in other computer languages, for instance Pascal, Logo, FORTRAN. The lack of proper "procedures" has always been the major limitation of many BASIC dialects as a computer language. TI Extended BASIC is one of the BASICs that does provide this facility. Not all BASICs, such as the GW-BASIC supplied with IBM style ATs that we bought recently for the laboratory, are so civilized. Perhaps the suppliers of these machines do not really want or expect anyone to program their machines seriously in BASIC. You will find that with true sub-programs available, that you cannot even conceive any more of how one could bear writing substantial programs without them (even within the 14 Kbyte limit of the unexpanded TI99/4A let alone on a machine with more memory).

The details of how procedures or sub-programs work vary from one language to another. The common feature is that the variables within a procedure are localized within that procedure. How they communicate with the rest of the program, and what happens to them when the sub-program has run its course varies from language to language. Extended BASIC goes its own well defined way, but is not at all flexible in how it does it.

Now let us look at how Extended BASIC handles sub-programs. The RUNning of any Extended BASIC program goes in two steps. The first is the prescan, that interval of time after you type RUN and press ENTER, and before anything happens. During this time the Extended BASIC interpreter scans through the program, checking a few things for correctness that it could not possibly check as the lines were entered one by one, such as there being a NEXT for each FOR. The TI BASICs do only the most rudimentary syntax checking as each line is entered, and leave detailed checking until each line is executed. This is not the best way to do things but we are stuck with it and it does have one use to be mentioned later. At the same time Extended BASIC extracts the names of all variables, sets aside space for them, and sets up the procedure by which it associates variable names with storage locations during the running of a program. Just how Extended BASIC does this is not immediately clear, but it must involve a search through the variable names every time one is encountered, and appears to trade off speed for economy of storage.

Extended BASIC also recognizes which built-in sub-programs are actually CALLed. How can it tell the difference between a sub-program name and a variable name? That is easy since built-in sub-program names are always preceded by CALL. This is why sub-program names are not reserved words and can also be used as variable names. This process means that the slow search through the GROM library tables is only done at pre-scan, and BASIC then has its own list for each program of where to go in GROM for the GPL routine without having to conduct the GROM search every time it encounters a sub-program name while executing a program. In Command Mode the computer has no way available to find user defined sub-program names in an Extended BASIC program in memory even in BREAK status. Extended BASIC also establishes the process for looking up the DATA and IMAGE statements in the program.

Well then, what does Extended BASIC do with user sub-programs? First of all Extended BASIC locates the sub-program names that are not built into the language. It can do this by finding each name after a CALL or SUB statement, and then looking it up in the GROM library index of built-in sub-program names. You can run a quick check on this process by entering the one line program

```
100 CALL NOTHING
```

TI BASIC will go out of its tiny 26K brain and halt execution with a BAD NAME IN 100 error message, while Extended BASIC, being somewhat smarter, will try to execute line 100, but halts with a SUBPROGRAM NOT FOUND IN 100 message.

The Extended BASIC manual insists that all sub-program code comes at the end of the program, with nothing but sub-programs after the first SUB statement (apart from REMarks which are ignored anyway). Now this is a good way to do things as the main program is right up front there for you to inspect. One of the great annoyances about Pascal is that despite its pretensions to structured style it forces you to hide away the main program right at the end after all the procedures. Extended BASIC then scans and establishes new variable storage areas, starting with the variable names in the SUB xxx(parameter list), for each sub-program from SUB to SUBEND, as if it were a separate program. It seems that Extended BASIC keeps only a single master list for sub-program names no matter where found, and consulted whenever the interpreter encounters a CALL during program execution. Any DATA statements are also thrown into the common data pool. Try the following little program to convince yourself.

```
100 DATA 1
110 READ X :: PRINT X :: READ X :: PRINT X
120 SUB NOTHING
130 DATA 2
140 SUBEND
```

When you RUN this program it makes no difference that the second data item is apparently located in a sub-program. IMAGEs behave likewise. On the other hand DEFed functions, if you care to use them, are strictly confined to the particular part of the program in which they are defined, be it main or sub-program. During the pre-scan DEFed names are kept within the allocation process separately for each sub-program or the main program. Once again try a little programming experiment to illustrate the point.

```
100 DEF X=1 :: PRINT X;Y :: CALL SP(Y) :: PRINT X;Y
110 SUB SP(Z) :: DEF X=2 :: Z=X :: DEF Y=3
120 SUBEND
```

This point is not explicitly made in the Extended BASIC manual and has been the subject of misleading or incorrect comment in magazines and newsletters. A little reflection on how Extended BASIC handles the details will usually clear up difficulties.

TI BASICs assign nominal values to all variables mentioned in the program as part of the prescan, zero for numeric and null for strings, unlike some languages (some BASICs even) which will issue an error message if an unassigned variable is presumed upon. This means that Extended BASIC cannot work like TI Logo which has a rule that if it finds an undefined variable within a procedure it checks the chain of CALLing procedures until it finds a value under that name. However, unlike Pascal which erases all the information left within a procedure when it is finished with it, Extended BASIC retains from CALL to CALL the values of variables entirely contained in the sub-program. Some recent BASICs on other machines now allow subprogram variables to be specified either as static (can be very handy) or transient (which saves storage space in between CALLs), but Extended BASIC is at the fully static end of the spectrum. The values of variables transferred into the sub-program through the SUB parameter list will of course take on their newly passed values each time the sub-program is CALLed. A little program will show the difference.

```
100 FOR I=1 TO 9 :: CALL SBPR(0):: NEXT I
110 SUB SBPR(A):: A=A+1 :: B=B+1 :: PRINT A;B
120 SUBEND
```

The first variable printed is reset to 0 each time SBPR is called, while the second, B, is incremented from its previous value each time. Array variables are stored as a whole in one place in a program, within the main program or sub-program in which the DIMension statement for the array occurs. Extended BASIC does not tolerate attempts to re-dimension arrays, so information on arrays can only be passed down the chain of sub-programs in one direction. Any attempt by a Extended BASIC sub-program to CALL itself, either directly or indirectly from any sub-program CALLed from the first, no matter how many times removed, will result in an error. Recursive procedures, an essential part of TI Logo, are not possible with Extended BASIC sub-programs, since CALLing a sub-program does not set up a new private library of values.

All of this discussion of the behaviour of TI Extended BASIC comes from programming experience with Version 110 of Extended BASIC on a TI99/4A with 1981 title screen. Earlier Versions and consoles are not common in Australia, but TI generally seems to have taken a lot of trouble to keep new versions of programs compatible with the old. On the other hand TI has also been very reticent about the details of how Extended BASIC works. The Editor Assembler manual has very little to say about it either, less by far even than it tells about console BASIC.

Another simple programming experiment will demonstrate what we mean by saying that Extended BASIC sets up a separate BASIC program for each sub-program. RUN the following

```
100 X=1 :: CALL SBPR :: BREAK
110 SUB SBPR :: X=2 :: BREAK :: SUBEND
```

When the program BREAKs examine the value of variable X by entering the command PRINT X, and then CONtinue to the next program BREAK, which this time will be in the main program, where you can again examine variable values.

We will now summarize the properties of Extended BASIC sub-programs as procedures in complete Extended BASIC programs, leaving the details of joining up the various procedures to the next section.

(a) Extended BASIC treats each sub-program as a separate program, building a distinct table of named variables and DEFed functions for each.

(b) All DATA statements are treated as being in a common pool equally accessible from all sub-programs or the main program as are also IMAGE statements, CHARacters, SPRITEs, COLORs, and File specifications.

(c) All other information is passed from the CALLing main or sub-program by the parameter lists in CALL and SUB statements. Extended BASIC does not provide for declaration of common variables available on a global basis to all sub-programs as can be done in some languages.

(d) Variable values confined within a sub-program are static, and preserved for the next time the sub-program is CALLed. Some languages such as Pascal delete all traces of a procedure after it has been used.

(e) Extended BASIC sub-programs may not CALL themselves directly or indirectly in a closed chain. Subject to this restriction a sub-program may be CALLed from any other sub-program.

(f) The MERGE command available in Extended BASIC with a disk system (32K memory expansion optional) allows a library of Extended BASIC sub-programs to be stored on disk and incorporated as needed in other programs.

o

Photograph of 2-way interface connectors

continued from page 3

A for loop consists of three expressions, each separated by semicolons (";"). The first expression is executed before anything else is done. Then the second expression is tested and the following statement executed in the same way as a "while" loop. Finally, after the statement has been executed, and before the test is repeated, the third expression is executed. The expression "count++" means exactly the same as "count = count + 1". It should be noted that each of these loops only executes the single statement that follows it. For this reason, curly brackets are used in Figures 5a and 5b in order to make two statements syntactically equivalent to a single statement.

As with last month, do not forget to experiment with the programs and concepts described in this months article. The description of loops above was rather quick and abstract. Write a few programs that use loops, such as a multiplication table or a table of squares, for example. The more you experiment, the sooner you will master 'c99'.

Next Month

An introduction to sub-programs. These form a basis for 'c99' programming, and simplify the task of creating new programs.

---

o

Programming c99

part 2

by Craig Sheehan

In last months article on programming 'C', we had an introductory look at PRINTF, learnt how to use the compiler, as well as leaving you with a problem to solve. Before reading this article, it would be advisable to review the above material. This month we will look further at PRINTF and have a short look at simple loops.

The problem left at the end of last months article was to figure out what that sequence of characters "\n" did in PRINTF, since they were not printed on the screen. For those of you who did attempt to solve this (shame!), Figure 2 gives a program that will allow you observe the effect of "\n".

```
/* What does "/n" do? */

extern printf();

main()
  { printf("Some text\nMore text\n");
    exit(0);
  }
```

Figure 2 - The comment says it all.

Type in, compile and run the program in the same manner as that described in last months article. When the program is run, you will notice that "Some text" and "More text" are displayed on two separate lines. The "\n" is used to symbolize new line. Such a code is required since two different PRINTFs will display their strings directly following each other, not on separate lines. In order to display the next item on the next line, the "\n" is required. The program in Figure 3a, and its output in Figure 3b, should illustrate this more clearly.

```
/* Our third PRINTF demonstration */

extern printf();

main()
  {
    /* The next two lines... */
    printf("ABCD\n");
    printf("EFGH\n\n");

    /* ...have the same effect as: */
    printf("AB");
    printf("CD\nEF");
    printf("GH\n\n");

    exit(0);
  }
```

Figure 3a - Another PRINTF program.

```
ABCD
EFGH

ABCD
EFGH
```

Figure 3b - The output from the above program.

So far we have only printed messages on the screen. To do useful tasks, variables and loops are required. In 'c99', there are two basic variable types: "int", which can hold a single integer (whole) number between -65,536 and 65,535, and "char", which holds a single character. Extensions to these basic types that are available in 'c99' are arrays, pointers, and arrays of pointers (more on these in a later article).

Unlike BASIC, variables must be declared before they are used. By declare, it is meant that the computer is made aware of the variables existence before the program starts. Figure 4 shows a program that uses numeric variables to hold a time.

```
/* Short routine to print a time that is set from */
/*  within the program.                           */

extern printf();

main()
  {  int hour, minute, second;

     /* "Set the time */
     hour   = 9;
     minute = 45;
     second = 32;

     /* Write time to screen */
     printf("%d:%d:%d\n", hour, minute, second);

     exit(0);
  }
```

Figure 4 - Declaring some variables.

The variables are defined by placing the word "int" followed by the names of the variables that are required. Each variable name is separated by commas, and a semicolon follows that last variable name. Character variables are defined in a similar way, except that "char" is used instead of "int". The three lines that follow the declaration "set" the time and are printed with the PRINTF function. Another special feature of PRINTF is also shown in this program. Just as "\n" is a symbol for new line, "%d" prints the next variable in the argument list which must be an integer. In this example, the value of "hour" is printed, followed by a colon, then the value of "minute", another colon, the value of "second" and finally a new line.

Of course, straight line programming can achieve little more than the example in the first part. However, loops permit much more complicated operations. There are three simple types of loops: while, do .. while and for. Figures 5a, 5b and 5c below show each of these loops. The output for all three example are the numbers 1 to 9, each printed on a new line.

```
/* The "while" loop */

extern printf();

main()
  { int count;

    count = 1;
    while (count<10)
      { printf("%d\n",count);
        count = count + 1;
      }
    exit(0);
  }
```

Figure 5a - A simple while loop.

```
/* The "do .. while" loop */

extern printf();

main()
  { int count;
```

continued on page 3

TI-Base Tutorial #2

by Martin Smoley, NorthCoast 99'ers

First some of Marty's shorthand from last month. Marty's theory will signify the beginning of some text which should not be taken as fact, but as my interpretation of an item. <E> means press <ENTER>. Last for now is ">", the greater than sign. I will use ">" when program segments are displayed at the left of every line. The position immediately to the right of the ">" will be column one. Take the example >12345. You should think of the number 1 as column one. The > does not exist. It is for reference only, the same as when you type in an Extended BASIC program, at the head of each line you see > but it is not part of the program. Also, in Tutorial 2, I have listed some Command Files with line numbers instead of ">" in the left most column. This is to allow for explanation of specific lines only. Line numbers are not used in Command Files, but from now on you will have to use FunnelWeb or the Editor Assembler Editor to create the Command Files, and this will be easier on me. Since we are on the subject I might as well fill you in. The editor which comes with TI-Base is not bad. By editor I am referring to the part of TI-Base you use to write and save Command Files. However, in TI-Base version 1.02, when you enter about 33 lines you run out of memory space. If you want or need to use the TI-Base editor you could produce a bunch of Command Files that run each other and get the job done quite well. I prefer to have the luxury of writing larger files if needed. I also prefer the use of embedded control codes as printer commands, which at this point are not available in the TI-Base Editor. There are two more reasons to contemplate an outside editor. The first is that the Command Processor that runs your Command Files truncates or chops off all lines at 40 characters. This means you can set tabs at 40 columns and after typing commands on the right half of the page you can tab over past 40 and type in comments. TI-Base will never see the comment so they will not interfere with the program logic or slow the speed down. Last, I print out lots of hard copies to check my work. It is hard to print files created by the TI-Base Editor in Int/Fix 40 Format.

TNAMES

Now I would like you to make a correction in the database we created for Tutorial 1. The problem is in the "XP" field of the database, "TNAMES". As it was displayed in SCREEN FOUR the XP dates were "Month-Year", (02-88), etc. This configuration does not sort to a desirable conclusion in a character field (further explanation later). In order to get what we want out of a "SORT ON XP" command we need the year first and the month second, that is "Year-Month", or (88-02). Since we only have five names in TNAMES you can edit the file and change them. I have placed a printout of TNAMES below for your convenience. You are, of course, going to have to learn something along the way. Let us say that you are really trying to learn TI-Base and you were working frantically on something when this newsletter arrived. Reading to this point you want to start editing immediately. In order to get going you must CLOSE your present file, point TI-Base at disk 3 (which is where you have the database named TNAMES), un-SORT the file, and change the colours if you do not like the present screen colours. If you had the little program that is listed below, you could type DO EDTN and TI-Base would do the rest. So let us make one. Fire up FunnelWeb, choose the Formatter menu and press 1 for EDITOR. Press CTRL[0] to change from word wrap mode. You should see a hollow cursor. At that point you can type in the Command File, EDTN. When this is done save it to disk under the name EDTN/C, and print out a hard copy which you can compare against the listing below. Remember, you do not type in the line numbers, and any line with an asterisk in the first column is a comment line.

```
0001  * Command File to EDIT TNAMES
0002  *      PROGRAM NAME = EDTN
0003  *      SAVED AS  EDTN/C
0004  *
0005  CLOSE ALL
0006  SET DATDISK=DSK3.
0007  USE TNAMES
0008  SORT OFF
0009  TOP
0010  COLOR WHITE DARK-BLUE
0011  EDIT
0012  CLOSE ALL
0013 RETURN
```

Let us attack this little command file. Lines 1 through 4 can be anything you need to refresh your memory about this program. Line 5 is a good idea for every command file you own. This line has saved me many times. If there are not any databases open, then 5 will do nothing. Line 6 is not really needed and you can leave it out. I do change drives on occasion with this statement, but you should remember to change it back at the end of the command file with a similar line. The reason it is here is to demonstrate that the CLOSE ALL should come at the very beginning of the command file before you do something like line 6 and confuse the system. Line 7 will open TNAMES on drive 3 as per line 6 or where ever DATDISK is located if line 6 is omitted. You can also use "USE DSKx.TNAMES" where x is any drive number, including a HORIZON RAMdisk, which I use. SORT OFF will un-SORT the file and TOP will point TI-Base at the first record in the database, as BOTTOM would point TI-Base at the last record. Line 10 is all you need to change the foreground and background colours. See page 4-2 of the manual for colours available.

And Now The Editor!

Line 11, EDIT, will put you in edit mode using whatever database file is open. In this case TNAMES. While you are in EDIT you can use arrow keys or enter to move around. You can then type over any item you want to change. At this time it is the XP field. This is important! You can also use FCTN[6] to page up, or enter, or FCTN[5] to page down to the next record. This could cause a problem as the changes you have made will not always be saved. If you make any changes you should always use FCTN[8] to register, or save, your changes and move to the next record. If you are on the last record in the file, you should still press FCTN[8]. This will not end the editing session and you will remain in the last record. You can then press FCTN[6] to page up, or FCTN[9] to leave the editor. In this case you would be returned to line 12 of our command file and TNAMES would be closed. RETURN will end the program and take you back to the dot prompt. Marty's theory: If I am editing a file without a program, I close the file with CLOSE or CLOSE ALL as soon as I have finished. This allows TI-Base to update all of the records. One last idea on this command file. I either SET TALK ON in the first line of this command file or more often it is already on when I run EDTN. This will allow me to read lines 1 through 7 on the screen while the database is being un-SORTed. I can then see if this is actually the program I wanted, with the right database, and that I have changed the location of the DATDISK. The beauty of little command file programs like this is that you can build on them and add things you realize you want as you go along. The command file will not forget any of the details from one day to the next like I do. Also, once you have the first one done you can copy it to a new name, (COPY DSK2.EDTN/C DSK2.NEWED/C). You can then use MODIFY COMMAND NEWED to

edit this new command file to handle another database, or do whatever you wish. It is much easier than typing a completely new command file from scratch.

Important Tip!

I have discovered that a command file created with FunnelWeb in DV/80 format can be copied or edited by TI-Base and the DV/80 format will not be changed. Therefore, if you create the command file below with FunnelWeb and save it to the name BLNK/C on your DATDISK, you can then copy it to a new name, re-edit it, and save it with MODIFY COMMAND, and it will remain a DV/80 file.

```
>SET TALK ON
>*
>*     Command File BLNK
>*
>*      Save as BLNK/C
>*
>*  Use as a seed file for DV/80
>*
>*  Copy to a name of your choice
>*   and type over this stuff.
>*
>RETURN
```

This means you will then be able to print the command file with FunnelWeb for a hard copy. I try to make sure that all my hard copies have the program name and pertinent comments at the top. Then if I am writing a new command file I can look over these hard copies and then merge chunks of previously written material with FunnelWeb LF Merge capabilities.

Let us get started on this month's project. We need another database to try some new routines. Create TNTST2 using the instructions below. Some of this is a repeat so skip over the parts you know and get right to the data entry. If this does not look slightly familiar you should refer back to Tutorial number one for more help.

```
>CLOSE ALL <E>
>CLEAR <E>
>CREATE TNTST2 <E>
```

When the CREATE screen comes up enter the following fields, and when you enter the 0 (zero) in the last column of field 4 press FCTN[8] and wait for TI-Base to create the file for you.

```
   arrows to move, enter to advance
 FIELD  DESCRIPTOR  TYPE  WIDTH  DEC

    1   TDATE        D      8
    2   NUM1         N      7    2
    3   NUM2         N      7    2
    4   ID           N      7    0
```

After pressing FCTN[8] TI-Base will ask if you want to enter data now. Answer yes and enter the data supplied below. Take your time, there are a lot of numbers here and you may get confused.

```
REC  TDATE     NUM1    NUM2     ID
0000 03/16/88  100.11   100.22 0712881

0001 02/29/88  200.11   200.22 0713831

0002 08/27/88  300.11   300.22 0717851

0003 03/03/88  400.11   400.22 0820871

0004 12/30/87  500.11   500.22 0921861

0005 06/06/88  600.11   600.22 0717851

0006 04/22/88  700.11   700.22 0921861

0007 01/21/88  800.11   800.22 0713831

0008 05/12/88  900.11   900.22 0820871

0009 06/17/88 1000.11  1000.22 0713831

0010 03/01/88 1100.11  1100.22 0921861

0011 08/03/88 1200.11  1200.22 0713831
```

I double spaced the data above to make it as clear as possible. If you make any mistakes, this is a good time to convert EDTN. Type COPY DSK2.EDTN/C DSK2.EDTST2/C <E>. After copying it use Modify Command to change lines 1 through 4, and change line 7 to USE TNTST2. Press FCTN[8] to save and you are done. (Please note that numbers in angle brackets <27> indicate one character whose ASCII code is in the angle brackets. GT)

```
01  * Command File TNTST2
02  *
03    SET TALK OFF
04    SET RECNUM OFF
05    SET HEADING OFF
06    SET LINE=80
07  CLOSE ALL
08    SET DATDISK=DSK2.
09  CLEAR
10  COLOR WHITE DARK-RED
11  WRITE 2,8," TI-Base Demonstration to"
12  WRITE 4,8,"open two databases at one"
13  WRITE 6,6,"time and find data in File #2"
14  WRITE 8,6,"which is related to an ID No."
15  WRITE 10,7,"in File #1. With some very"
16  WRITE 12,7,"simple math implementation."
17  WRITE 14,9,"************************"
18  WRITE 16,9,"   Running:  TNTST2     "
19  WRITE 18,9,"************************"
20  USE TNAMES
21    LOCAL CDATE D 8
22    WRITE 20,2," Enter the Date  MM/DD/YY"
23    WRITE 21,2," Within Quotes"
24    READ 21,18,CDATE
25    WRITE 22,5,"Current Date: ",CDATE
26   SORT ON ID
27  CLOSE
28  USE TNTST2
29   SORT ON TDATE
30  CLOSE
31  SELECT 2
32  USE TNTST2
33  TOP
34  SELECT 1
35  USE TNAMES
36  TOP
37       LOCAL BLNK C 4
38       REPLACE BLNK WITH "<27>E<27>G"
39       PRINT BLNK
40  LOCAL TESTID N 7 0
41  LOCAL TEMP C 60
42    WHILE .NOT. (EOF)
43     REPLACE TEMP WITH "<27>E<27>G" | TRIM(LN) ;
44             | "  " | TRIM(FN) | " " | MI ;
45             | "  " | ID
46          PRINT TEMP
47           REPLACE TESTID WITH 1.ID
48         DO DSK2.NUMTST2
49        COLOR WHITE DARK-BLUE
50     WRITE 17,9," Running: > TNTST2 <     "
51        SELECT 1
52       MOVE
53    ENDWHILE
54  CLOSE ALL
55    SET RECNUM ON
56    SET HEADING ON
57    SET TALK ON
58  RETURN
```

The command file on this page may look complicated, but it is not. We will go through it together, and I will try to explain the important parts. I hope you have read Tutorial 1 so I can skim over the routine parts and concentrate on the rest. Remember do not enter the line numbers. Lines 1 through 9 are strictly house keeping except for CLOSE ALL, and from now on I will consider it house keeping. CLOSE ALL should be part of every MAIN command file. By MAIN I am referring to a command file that may run other command files, but

is not itself run by a previous command file. This command file runs NUMTST2 as you can see in line 48. You would not want to close all the files in NUMTST2 it would bomb the program. I intend to have my data disk in drive 2, line 9 clears the screen and changing the screen colours has no real value. The WRITE statements from 11 through 19 are to demonstrate user prompts. The lines I have included are not important, but it will give you some idea of ROW-COLUMN display. Line 20 is the beginning of the real stuff. USE TNAMES opens that database which it expects xo find on drive 2. After line 8, TI-Base will expect to find all command files and databases on drive 2, and I will not waste space bringing it up again. There is a three line cluster which is important. The lines are 20, 26, and 27. Their purpose is to open, sort and close a file. This is identical to lines 28, 29, and 30. However, lines 21 through 25 are of interest. Line 21 initializes the LOCAL variable named CDATE, which is a D (date type) entry with a length of 8 characters. A variable is a place to store some type of information. In this case it will be the Current DATE (CDATE) which you will type in when asked. Lines 22 and 23 will ask you to enter the date and 24 will place the cursor on the screen one space after "Within Quotes", and wait for your input. Note: with Version 1.02 all Characters, or Dates, which are characters, must be input enclosed in quotation marks, "09/01/88". Line 25 will write the message "Current Date:" and display whatever you type in for CDATE. For your information: I have initialized CDATE close to its use for your benefit. I will continue this procedure in this program, but from then on variables should be initialized at the beginning of a command file. This little chunk (lines 21-25) was stuck in here because TI-Base likes to have a database open before you READ to a variable. And line 25 WRITEs to screen line 22. You will notice that screen line 22 does not scroll like the rest. You can put a message there and it will stay put until a CLEAR or another WRITE 22,x removes it. Some of these things will be apparent when you run this program, or DO TNTST2. Line 31 leads us into a very complicated and confusing area. I will try to cover it as thoroughly as possible. I will re-analyze it many times in the future. It is that important.

Think of a Lazy Susan, or a rotatable table. This table has 5 areas on it with low partitions between each area. You can take from one to five file folders which are filled with sheets of paper and place one in each of the five areas. You must stand in one spot, but you can SELECT one of the five areas to be positioned directly in front of you. The area SELECTed, (1 to 5), is the one where you can do the most work, but you can see over the partitions to do limited things with the information in the files which are not directly in front of you. If you can grasp this concept and visualize the 5 different slots, or areas, you are going to catch on fast. Remembering, or keeping track of what can be done in non-SELECTed slots is a challenge. Now, line 31 and beyond. Line 31 SELECTs slot #2, 32 opens the database named TNTST2 in slot #2 and 33 points TI-Base at the first record in the file. Line 34 SELECTs slot #1, 35 opens the database named TNAMES in slot #1 and 36 points TI-Base at the first record in that file. Remember that both of these databases were previously SORTed to our specifications. Well, we have done it. At this point we have opened two databases at the same time. TNTST2 is open in slot #2 and TNAMES is open in slot #1, and if we do not count all the junk I put in to add flash to the program, we did it with about 12 lines of code. I told you that we would get through this somehow. If you examine and keep track of this stuff one piece at a time, you will get the hang of it sooner than you think.

```
01  * Command file NUMTST2
02  *
03  CLEAR
04    WRITE 15,9,"************************"
05    WRITE 17,9,"    Running:  NUMTST2   "
06    WRITE 19,9,"************************"
07  WAIT 2
08  COLOR WHITE DARK-GREEN
09    WRITE 17,9,"Looking For ID No.",TESTID
10  WAIT 2
11    LOCAL TNUM1 N 10 2
12    LOCAL TNUM2 N 10 2
13    LOCAL STNUM1 N 10 2
14    LOCAL STNUM2 N 10 2
15    LOCAL T C 8
16  REPLACE T WITH "   TOTAL"
17  SELECT 2
18  TOP
19   REPLACE BLNK WITH "<27>4 "
20   PRINT BLNK
21  WHILE .NOT. (EOF)
22    IF TESTID = ID
23      PRINT TDATE,BLNK,NUM1,BLNK,NUM2 ;
24            BLNK,ID
25      REPLACE STNUM1 WITH TNUM1 + NUM1
26      REPLACE TNUM1 WITH STNUM1
27      REPLACE STNUM2 WITH TNUM2 + NUM2
28      REPLACE TNUM2 WITH STNUM2
29    ENDIF
30   MOVE
31  ENDWHILE
32   REPLACE TEMP WITH "--------------------";
33   | "------------------------"
34   PRINT TEMP
35  PRINT  CDATE,TNUM1,TNUM2,T
36   PRINT BLNK
37    REPLACE BLNK WITH " "
38   PRINT BLNK
39  RETURN
```

Note: although there is a command file just above, we will be discussing the previous command file until we reach line 48. Do not get confused. We just left line 36.

Lines 37, 38 and 39 make up a small group. Their purpose is to initialize the LOCAL BLNK for 4 Characters. Fill it with the control codes that set the printer to Emphasized and Double strike, and send the codes to the printer. Entering the control codes can only be done with TI-Writer or FunnelWeb at this time. There will be some useless repetition concerning control codes. I will explain later.

Line 40 and 41 initialize two more variables to be used in the WHILE loop. TESTID is to hold a Number with the length of 7 and 0 decimal places, and TEMP has been discussed previously. Let us get into the meaty part. The way I have set this loop up it will continue to do everything from line 42 through line 53 until it reaches the End Of File marker, (EOF), for the database TNAMES which we have located in slot #1. Note: TNAMES is in slot #1, and slot #1 is our currently SELECTed slot because the last slot we worked with was slot #1, in line #34. I will keep at this concept as we go along.

Line 42 has a simple but important job. It immediately checks to see if we have hit the EOF in whatever file is open, in the slot we are facing. In this case it is slot 1 and the file is TNAMES. WHILE it does does encounter the (EOF), it goes directly to the line after the ENDWHILE, which in this case is line 54.

Lines 43, 44 and 45 are seen by TI-Base as one continuous line because of the semicolon (;) at the end of lines 42 and 43. So in this line TI-Base is going to take the 2 control codes directly after WITH and hold them. It will then TRIM the trailing blanks from LN and attach it behind the control codes, and then it will stick 2 spaces behind that. It will TRIM the trailing blank spaces from FN, attach it to our growing string, and then throw in another blank space. It will then tack MI on that followed by 2 more spaces, and last but not least ID. We did not TRIM MI because MI does not have any extra blank spaces. We did not TRIM ID because it is of (N)umeric type (a number) and TRIMming is only used on (C)haracter strings. Then TI-Base takes this whole mess we have put together and sticks it into the variable we call TEMP.

Note: If you look back at line 41 you will see we made TEMP with 60 spaces. When you fill up a variable with all kinds of junk, like we just did, you must make sure the variable is big enough to hold it all. In line 46 we PRINT all the junk we just put into TEMP.

I would like to also mention that the junk we filled TEMP with was related to TNAMES, (LN, FN, MI, ID). Using this type of data gathering it is up to us to be sure TI-Base is pointed at the right slot and that that slot contains the database which holds the information we want. In line 47 we put the same ID number from above into the holding area we named TESTID. The phrase 1.ID is another way to tell TI-Base that we want the ID number from slot #1. At this time the 1 is for your information only, and does not have any real effect on the program except to assure me that I am getting ID from slot #1. TI-Base's Author uses this form of data gathering in the command file named PROCESS, manual page 5-5.

Well here comes line 48. When TI-Base hits line 48 it leaves the command file named TNTST2 with everything exactly as it is and executes the command file named NUMTST2 on Disk 2. So now we start looking at the lines in NUMTST2. Line 3 CLEARs the stuff left on the screen by that other command file. Lines 4, 5 and 6 put up a new message. TI-Base WAITs 2 seconds, then it changes the colours to WHITE on DARK-GREEN. In the same instant it reWRITEs a new message to line 17, followed by the TESTID. This is the Identification Number we brought with us from that other command file. We will use it to find related data in the database you typed all those numbers into a short time ago.

TI-Base WAITs a couple more seconds just for kicks and we are on our way. Lines 11 through 15 initialize all the variables we will need in this program. We can also use variables from that other command file, but we cannot send these variables (lines 11-15), back there. If we needed to send something back over there, we could put it in one of the variables from that command file (like TEMP) just before we RETURN and then we could use that information when we RETURNed to that other command file. I did not use the names of the 2 command files in that explanation because it was even more confusing that way.

Line 16 places some blank spaces and the word TOTAL into T for later use. In line 17 we SELECT slot #2, which is where TI-Base holding the database TNTST2. TOP in line 18 is only to make me feel secure. We should already be at the TOP of the file. Lines 19 and 20 stick the control code for Italics into BLNK and PRINT it.

Note two things: one, we needed a blank line printed anyway, which this gives us, and two, that BLNK came over from that other command file. Now, in line 21 we have another WHILE loop. The WHILE loop that runs from line 21 through line 31 has the same definition I gave earlier, but we will do different things while we are inside this loop, and it will be looking for the (EOF) for TNTST2 in slot #2.

I cannot believe I have written so much. Well, since I am using up so much Newsletter space, here is a promotion for my sponsor.

Join The NorthCoast 99'ers UG

NorthCoast has 3500 plus programs in its library and produces a great little Newsletter. You can take full advantage of the club's services by mail, and you will be certain of receiving my wonderful tutorials in the future. The membership cost for someone living in the continental United States is only $15.00. You can send your membership fee to me, Martin A. Smoley, 6149 Bryson Drive, Mentor, Ohio, 44060. Make all checks payable to NorthCoast 99'ers User Group, do not send cash, and I will expedite your membership personally.

OK, NUMTST2, line 22. When we get to this point TI-Base is looking at the first record in NUMTST2, which we have SELECTed in slot #2. Therefore, in line 22, IF the value in TESTID matches or is equal to the value in ID, then TI-Base will execute all the lines between the IF (line 22) and the ENDIF (line 29). Remember, TESTID holds the ID number which matches the LN, FN and MI we just printed in from the database TNAMES. ID holds the ID number from the current record of the database TNTST2. I will not follow the program accurately because TI-Base will not find a match to make the IF true until the sixth record of this database. So let us say it finds a match which makes line 22 true.

Line 23 prints the information held in TDATE, NUM1, NUM2, and ID under the person's name from TNAMES. Lines 25 and 26 make up an accumulator that keeps a running total of the NUM1 part of any matching records. Similarly lines 27 and 28 keep a running total of the numbers in NUM2 if the ID match is true. Coming from line 28 to line 29, TI-Base ignores 29 and goes directly to line 30. This line tells TI-Base to MOVE its pointer to the next record in the file. So we are now looking at the next record in the database TNTST2.

The ENDWHILE in line 31 is not ignored by TI-Base, and TI-Base is sent back to line 21 to test the new ID we now have against TESTID which remains the same. This loop goes around and around. Each time it does, it moves to the next record and then checks for (EOF). If it is not the End Of File, and it has data to work on, it immediately tests to see IF the ID numbers match, etc.

When it runs out of data or hits the (EOF), line 21 sends TI-Base directly to line 32, the first statement after the ENDWHILE. TI-Base then puts the dashed line into TEMP and prints it. TI-Base then prints the current date (CDATE), which you entered at the beginning of that other command file, the totals in TNUM1 and TNUM2, and the word TOTAL. In lines 37 and 38 TI-Base turns Italics off, at the printer. We then RETURN to that other command file named TNTST2. In doing so we throw away all the LOCALs we initialized in this command file.

When we land back in the command file named TNTST2 we land on line 49, which changes the screen colours. Line 50 WRITEs this command files name to the screen over screen line 17, which was left there by that other command file. Line 51 SELECTs slot #1, so we are once again working with TNAMES. Line 51 MOVEs TI-Base's pointer to the next record, for a new name, and line 53 sends us back to line 42 to start the whole process over again.

These two loops will ratchet through the names in TNAMES one at a time, and for each name in TNAMES, will completely search TNTST2 for any information that is related to that name by comparing ID numbers in TNAMES to ID numbers in TNTST2. It will continue to search until it runs out of names, or records, in TNAMES. At that time 42 will send TI-Base to line 54. ALL databases will be CLOSEd, things that were turned off will be turned back on and the whole thing is finished. In line 58 you are RETURNed to the Dot Prompt.

That just about wraps this tutorial up except for a few things I said I would get back to. I threw around a lot of control codes in this set of command files. If you are using FunnelWeb to produce your command files, you can carry these ideas back to the LABEL program we did last month. Fire up FunnelWeb and retype the command file called LBLS1/C, but this time name it LBLS2/C. There are only about 32 lines and most of them are very short. Leave out the present line that reads LOCAL BLNK C 1. Next, add lines 37, 38 and 39 from TNTST2. Insert them between the line that says TOP and WHILE .NOT.(EOF). This will cause your printer to print in Emphasized and Double strike Mode. If you do not like that, try what I did in line 43. You can concatenate (|) control codes on the front and rear of a character string. There are lots of ways to do it. Before my mind goes completely I am giving up. I copied the printout from this months stuff below. I would also like to add that this set of command files make a nice club demonstration.

```
Vivannovitch  Elexxie I.  0712881
03/16/88      100.11      100.22     0712881
-----------------------------------------------
09/11/88      100.11      100.22     TOTAL
```

continued on page 26

Games Information

by Robert Brown

Welcome to a new look Games Info. Stephen has now lost interest in the TI99/4A and has left it up to myself to continue these excellent articles. I am heavily involved in school and I have not had time to write more excellent, fab articles, so since this is in the holidays, I can now right many articles in store for future issues of the TND.

In this new look issue, we continue our series on Zork, this month Zork III...

Well, you have come a long way since you first stood by the mailbox outside the house in the forest. You have defeated the thief, outwitted the Wizard of Frobozz, and now, you stand at the foot of the endless stairs, ready to embark on the final part of your journey. So, pick up the lamp, turn it on, and head along due South until you come to the shore of the lake.

Drop the lamp (say goodbye to it; you will not be seeing it again), and jump into the lake. Brrrr!!! Pretty cold! So, do not stay in there long; swim west and then go South into the Scenic Vista. Kind of a strange place, with changing numbers on the wall and a bare table.., not quite all that scenic, eh? Well, get the torch, and wait for the number to change to "II." Then, touch the table.

My oh my! You are in a room from Zork II, Room 8, as a matter of fact. However, you do not have much time to sight see, so get the can of Grue repellant, then try moving East, and you will find yourself back in Scenic Vista again. Now wait for the number to change to "III," then touch the table again. This time, you are in a Damp Passage. Drop the torch, and just wait there until you are pulled back to Scenic Vista.

Okay, you are finished here, so move along North to the shore, and again jump in the lake. Splash! It has not gotten any warmer; in fact, you just dropped the can of repellant. So, go Down, and you will be on the lake bottom. Ah, there it is! But, could there be something else there, too? "Get all", and you will have not only the repellant but also an amulet. This is one of those "wonderful" variable things; it may take more than one try on your part to get both items. In the meantime, you cannot stay in the icy waters too long, and sooner or later a hungry fish will come looking for you.

Therefore, it is best to save the game before you jump in from the Western Shore. So if you die in the water, or get eaten by the fish, or picked up by the Roc (while you are swimming on the surface), you do not have to start all the way back at the beginning. By the way, this is the only one of the Zorks where you do not lose points if you die. But, all the items you have collected so far get scattered all around, and it is time consuming to go look for them.

Okay, now you have the can and the amulet, so head Up to the surface, then South to the Southern shore. You can see a cave to the South, and it looks kind of dark. In fact, it *is* dark in there, which is why you have the repellant. So, spray the smelly stuff on yourself, and go South, and you will find yourself in a Dark Place. Go South again, then East, and you will be in the key room. Whew! At least there is some light in here! And by the light you can see a strange key. Get the key, then move the manhole cover and go down.

And here you are on an aqueduct. Since you cannot go back (the Grue repellant would not have lasted that long), you might as well go forward. So, just head along North and you will come to the Water Slide. Go North down the slide, and guess where you are? In the Damp Passage! And there is the torch, so pick it up, because you are certainly going to need a light source, especially when you think of where you are going next.

So, from the Damp Passage hike along West to the Junction (you cannot get the sword out of the rock, so do not even try), then South into Creepy Crawl, and Southwest into the Shadow Land. Here we come to another variable portion of the game. You will have to wander around in the Shadow Land until a cloaked and hooded figure appears. When that happens, the sword will suddenly materialize in your hand, and you will be able to fight.

However, since there is no way of telling when that will happen, you just have to keep moving around until it does. At least you will get a chance to practice some elementary map-making! Also, this is the most dangerous part of the game, as the figure is quite capable of killing you, too! So, best to save before you enter Shadow Land.

When the mysterious figure finally appears, attack him with your sword until he is badly wounded and cannot defend himself. At that point, get his hood. The figure will then disappear, leaving the cloak behind. Get that also. Now, you have to get out of here; and I cannot tell you exactly how, since there is no way of knowing exactly where you were when the fight started. However, if you go Eastwards, you will exit the Shadow Land at either the Creepy Crawl or the Foggy Room. From either place, go North to the Junction.

From the Junction, it is West through the Barren Area, and West again to the Cliff. Bet you just cannot wait to climb down the rope, huh? Well, pick up the bread first, then go down to the ledge. Well, well, a chest! Too bad you do not have a key to open it. In fact, there is no way for you to open it at all. But do not despair, there is a way of doing it.

Just wait around and someone will come along the top of the cliff. You may not really trust him, but tie the rope to the chest when he asks, and wait around some more. Eventually, he will return and help you back up the cliff. He will also give you a staff, which is what you are really after here. Take the staff, then go back down to the ledge, and from there, to the Cliff Base.

Now trek South to the Flathead Ocean, and do a little more waiting. Sooner or later a ship will come floating by. As soon as you see it, say: "Hello, Sailor." The man in the ship will throw something onto the beach for you. Take a look, and you will see it is a vial. It will come in handy later, so pick it up. Now comes the fun part: You have to wait for the earthquake. (Notice how you have been doing a lot of waiting around? I hope you are a patient person!)

While you are waiting, you might want to wander around a little, although you have been to most of the accessible places by now. In any case, wherever you are, once the earthquake hits, make your way to the Creepy Crawl, and from there East into the Tight Squeeze, then East again into the Crystal Grotto. Then all the way South to the Great Door, and East into the Museum Entrance.

Now, open the East door, then go North into the Museum. Look at the gold machine (it is a time machine, in case you were wondering), then set the dial to 776. Here comes the fun part: Push the machine South into the Entrance, then East into the Jewel Room. Get into the machine, and push the button. Aha! Now you are back in 776 GUE, but the time machine seems to have vanished! No matter, wait for the guards to leave, then get the ring (and *only* the ring!), then open the door, go out into the Entrance, open the North door and go North.

By golly, the machine is right there! Put the ring under the seat, turn the dial to 948, get in, and push the button. Whew, you are back in the right time period again. Get out of the machine, look under the seat (you will get the ring automatically when you do this), then back South, and South again, to the Royal Puzzle.

Okay folks, you are about to enter the absolute nastiest part of the game. You must follow the

instructions *EXACTLY* as given, or you will never get out. And, since it would be easy to make a mistake here, I strongly recommend you save the game.

1. Go Down the hole, then push the South wall. Then go East, South, East, East. Push the South wall, get the book, and push the South wall again.

2. Push the West wall twice. Then go East, South, and push the East wall.

3. Now, go straight North until you come to the marble wall, and push the East wall.

4. Now, go West, South, South, South, South, East, East, North, North, North, and push the West wall.

5. From there, go East, South, South, South, West, West, West, West, North, North, North, West, North. Push the East wall three times.

6. Now, West, West, South, South, East, East, South, and push the East wall.

7. Okay, now West, West, West, North, North, North, East, East, and push the South wall two times.

8. From there, West, South, South, East, East, North, and push the West wall two times.

9. Now, South, West, and push the North wall until it will not move any more.

10. Then West and North. Finally! You have maneuvered the ladder under the hole (which was the purpose of all this pushing and running around), and now you can just go up and out! WHEW!!!!

Okay, you have solved the Royal Puzzle and you have the book, so go North to the Museum Entrance, then open the East door and get your other stuff from the Jewel Room. Then it is back West to the Great Door, and from there back to the Junction. Now, East into the Damp Passage, and NE to the Engravings Room.

Well, we have here yet another (1) of those variable events: Sooner or later, an old man will be sleeping here. If he is not there the first time you arrive, walk around a little and return. When you finally do see him, wake him up and give him the bread. He will eat it and then make visible to you a secret door. He will then vanish.

Okay, you are getting closer to the end! Open the door, and go into the Button Room, then North to the Beam Room. Put the sword in the beam, then go back to the Button Room and push the button. Now, back North to the Beam Room and North again into the Mirror Room. There will be an opening in the Mirror, so go North one more time, and you will be inside.

Now, do not let the long and complicated descriptions scare you! It is not really as bad as you think (it is worse! Hehehehe ..just kidding!). First, raise the short pole. Then, push the white panel twice. Now, push the pine panel, and go North.

Okay, so here you are, standing a little too close for comfort to the Guardians of Zork. If I were you, I would not try going past them quite yet! Open the vial, then drink the liquid. While nothing seems to have happened, you have in fact become invisible. Now you can walk North until you come to the locked door. Knock on the door, and the Dungeon Master will open it and let you in.

All right, hang in there, you have reached the end game! Go North, then West, then North again. The Dungeon Master will be following you. Go North to the Parapet, set the dial to 4, and push the button. Now, go South, open the cell door, and step inside. The Dungeon Master will not follow you in. Once inside, you will notice a bronze door in one of the walls. However, you cannot open it yet! Something else has to be done. And it will have to be done by someone else. So, first tell the Dungeon Master to go to the Parapet. Then tell him to turn the dial to 1, and then tell him to push the button.

All right!! The magic moment has arrived! Unlock the bronze door with the key, open the door, and go South!

And there you have it, the solution to all the Zork Family. Well not really. Beyond Zork has been released, but in a different format, ie text and graphics, a new style of adventure for Infocom (but not available on the TI99/4A). Well that wraps up another well designed NEW LOOK GAMES INFO. I hope you enjoyed it and much as I did.

In coming issues, we have a look at Karate (yes now available for the TI99/4A) and many other adventures for the TI99/4A, as well as drawn maps on the adventures (see coming TNDs).

Well, until next month (cannot wait!!), see ya.
Do not forget you can write to me at 141 Beecroft Road, Beecroft 2119 or leave mail on the BBS to GAMES.

o

---

continued from page 8

After you have typed in a lot of records, or tuples into your data base, you will probably want to do something with them. There are many ways to use the data. Most likely you will want to create a "REPORT"; a print out of part or all of the data. You might want the data in the "REPORT" to be sorted by one of the fields (remember, columns) in your report. You might want everyone who lives in the (02) area code. In some databases, you are limited in how you can access your data. More sophisticated databases have what is known as a "QUERY" capability; really some way to ask "QUESTIONS" of your data base, such as "WHO ARE THE PEOPLE IN MY PHONE LIST NAMED FRED, AND WHAT ARE THEIR TELEPHONE NUMBERS?".

Of course, most databases will not simply let you enter a sentence like that one. That is the way to get data from the data base in the user's view. Usually you have to use a "QUERY LANGUAGE". This language is a semi-algebraic way of getting certain records from the data base. For instance, to get data from one popular PC data base you would type "SELECT ALL FROM MYPHONELIST WHERE NAME EQUALS FRED". After entering that "QUERY", the data base would list all the data base records with the name "FRED" in them, and incidently their phone numbers as well. Again, this is a conceptual approach. The data base might convert the conceptual version of the query into something like "select, myphonelist, name=fred". The conceptual way is considerably easier to understand.

In order to get the list of "FREDs" sorted, in some data bases you have to make another table of just people named Fred, and then have the data base sort it in alphabetical order. Some data bases will let you combine this into one step.

As mentioned above, once you have entered data into the data base you will want to create a report. Almost all data bases will let you designate, to some degree or another, how you want selected records printed on the page; the order of the fields, etc. Some will let you draw lines and boxes around fields, even. This designation is called the "REPORT". Sometimes it too is saved as part of the schema, but usually not.

As mentioned above. There are several types of data bases. The two found in the TI-99/4A and Geneve world are the relational and flat file. Before you can understand them, you should have a firm grasp of the concepts elucidated above. We will tell you the difference between the two in the next issue of Beginners Corner.

o

Tips from the Tigercub #34

by Jim Peterson, Tigercub Software, USA

Tigercub Software
156 Collingwood Ave
Columbus, OH 43213

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

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

Descriptive catalog $1, deductable from your first order.

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

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

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

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

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

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

For descriptions of these send a dollar for my catalog!

While they last, and the supply is limited, I will sell a single Texas Instruments cassette interface cable for $2 with any order for cassette software.

My sincere apologies for a serious goof in the Sort Watcher program in Tips #33. The GOSUB in line 120 should go to line 1020, not 32767! Also, in line 210 please change the 920 to 930.

Steven Shouse of TIRUG sent this improvement to the GRAPHPAGE in Tips #33 -

```
100 OPEN #1:"DSK1.GRAPHPAGE",OUTPUT ::
    PRINT #1:TAB(4);RPT$("_",75):: FOR J=57 TO 1 STEP -1
    :: J$=STR$(J)
```

The 99/4A National Assistance Group (which is a commercial enterprise, not a user's group, although they charge a fee to "join"), sells public domain programs at $3 each, but you cannot order individual programs, you have to buy a package deal.

I sell good copyrighted programs, written by myself, for $3, I let you pick and choose, even just one program if you want. I do not pretend to be a user's group (I know that Tigercub often gets misspelled as Tiger Club but I cannot help that!), and I do not charge you to "join".

The reason for these remarks is that one of the public domain programs sold by that group is listed as SAMARKAND. It may be only an odd coincidence that I wrote a random music composer entitled SONG OF SAMARKAND and put it in public domain because I did not think it was worth selling. Anyway, if you want it, here it is.

```
100 CALL CLEAR
110 REM - SONG OF SAMARKAND programmed by Jim Peterson -
    Version 3
120 RANDOMIZE
130 CALL CHAR(94,"00")
140 CALL CHAR(95,"00")
150 CALL SCREEN(11)
160 PRINT "From the Third Movement of":"":"    THE
    NEVER-ENDING SONG":"":"          by Emir Abdul
    Aziz":"":"..........................."
170 PRINT : : : : : : : : : : : : : : :
180 FOR J=1 TO 23
190 CALL
    HCHAR(12,5+J,ASC(SEG$("^THE^SONG^OF^SAMARKAND^",J,
    1)))
200 NEXT J
210 CALL HCHAR(11,6,94,23)
220 CALL HCHAR(13,6,94,23)
230 M$="187EFF42668124C3DB665A18423C5AA542817E995A001800
    24BDBD3C667E66668100243C0042187E5AA53CC3427E3C8181
    7E5AE7669924187E429924008181DBC3"
240 DIM N(30),S(21)
250 F=220
260 FOR J=0 TO 36
270 X=X+1+12*(X=12)
280 IF (X=2)+(X=5)+(X=7)+(X=10)+(X=12)THEN 310
290 Y=Y+1
300 N(Y)=INT(F*(1.059463094^J))
310 NEXT J
320 CALL HCHAR(1,1,32,320)
330 CALL VCHAR(1,31,95,96)
340 CALL HCHAR(24,1,95,64)
350 CV=2
360 K=8
370 K=K-INT(5*RND+1)+INT(5*RND+1)+2*(K>21)-2*(K<1)
380 IF (K<1)+(K>21)THEN 370
390 CALL SOUND(-999,N(K),0,N(K)*CV,0,3.75*N(K),30,-4,5)
400 X=INT(40*RND)
410 IF X>12 THEN 370
420 ON X+1 GOTO 430,490,540,580,660,730,770,850,870,
    970,990,1040,1060
430 IF INT(4*RND)<3 THEN 390
440 FOR T=K TO 20
450 CALL SOUND(-999,N(T),0)
460 NEXT T
470 K=1
480 GOTO 390
490 FOR T=K TO 1 STEP -1
500 CALL SOUND(-999,N(T),0)
510 NEXT T
520 K=T+1
530 GOTO 390
540 FOR T=K TO 1 STEP -1
550 CALL SOUND(-999,30000,30,30000,30,3.75*N(T),30,-4,0)
560 NEXT T
570 GOTO 370
580 FOR TT=K TO K-INT(5*RND+1)STEP -1
590 IF TT<2 THEN 370
600 FOR T=1 TO INT(7*RND+3)
610 CALL SOUND(-999,N(TT),0,2*N(TT),0)
620 CALL SOUND(-999,1.03*N(TT),0,2.06*N(TT),0)
630 NEXT T
640 NEXT TT
650 GOTO 370
660 FOR T=K TO K-INT(3*RND+3)STEP -1
670 IF T<2 THEN 370
680 FOR D=0 TO 15 STEP 2
690 CALL SOUND(-999,2*N(T),D,3*N(T),D,3.75*N(T),30,-4,0)
700 NEXT D
710 NEXT T
720 GOTO 370
730 FOR X=1 TO 15
740 CALL SOUND(-999,N(X),0,N(16-X),0,N(1),30,-4,5)
750 NEXT X
760 GOTO 370
770 FOR T=K TO K-INT(4*RND+1)STEP -1
780 IF T<2 THEN 370
```

```
790 CALL SOUND(100,N(T),0,2*N(T),0,3.75*N(T),30,-4,5)
800 FOR TT=N(T)TO N(T-1)STEP -10
810 CALL SOUND(-999,TT,0,2*TT,0,3.75*TT,30,-4,5)
820 NEXT TT
830 NEXT T
840 GOTO 370
850 CALL CHAR(32,SEG$(M$,2*INT(57*RND+1)-1,16))
860 GOTO 370
870 IF INT(4*RND)<3 THEN 390
880 CALL SOUND(-3000,N(K),0,2*N(K),0,3.75*N(K),30,-4,0)
890 FOR J=1 TO INT(5*RND+5)
900 S(J)=INT(21*RND+1)
910 NEXT J
920 CALL SOUND(-1,30000,30)
930 FOR T=1 TO J-1
940 CALL SOUND(-999,N(S(T)),0,N(S(T))/1.68,0,
    3.75*N(S(T)),30,-4,0)
950 NEXT T
960 GOTO 370
970 CALL CHAR(95,SEG$(M$,2*INT(57*RND+1)-1,16))
980 GOTO 370
990 IF INT(4*RND)<3 THEN 390
1000 FOR J=220 TO 660 STEP 20
1010 CALL SOUND(-999,J,0,880-J,0,3.75*N(12),30,-4,0)
1020 NEXT J
1030 GOTO 370
1040 CALL CHAR(32,"0")
1050 GOTO 390
1060 CV=CV+(CV=2)/2-0.5*(CV=1.5)
1070 GOTO 370
```

If you are trying to exchange newsletters and are using the listings of user groups published by Texas Instruments and by others, you are finding that they are way out of date! Send me a disk and some return postage - or just send $1.50 - and I will send you my address list of about 140 groups I exchange with. It is updated every month from return addresses on newsletters I receive.

For those of us who are still struggling along with one disk drive, this routine will transfer any number of D/V80 files, totalling up to about 42 sectors, from one disk to another in one pass, and will optionally save under changed names.

```
100 DIM M$(2000),F$(25),C$(25):: CALL CLEAR ::
    T$=CHR$(1)
110 DISPLAY AT(8,6):"TIGERCUB FILEMOVER" ::
    DISPLAY AT(15,1):"PRESS ENTER WHEN FINISHED"
120 F=F+1 :: IF F>25 THEN 130 ::
    DISPLAY AT(12,1):"FILENAME? DSK"&T$ ::
    ACCEPT AT(12,14)SIZE(-12)BEEP:F$(F):: IF F$(F)<>T$
    THEN 120
130 F=F-1 :: FOR J=1 TO F :: ON ERROR 260 ::
    OPEN #1:"DSK"&F$(J),INPUT ::
    DISPLAY AT(12,1):"READING "&SEG$(F$(J),3,255)
140 X=X+1 :: LINPUT #1:M$(X) :: C=C+LEN(M$(X))
150 IF C>10000 THEN DISPLAY AT(20,1):"INSUFFICIENT
    MEMORY FOR "&SEG$(F$(J),3,255):: GOTO 190
160 IF EOF(1)<>1 THEN 140
170 X=X+1 :: M$(X)=T$ :: CLOSE #1
180 W=W+1 :: NEXT J
190 X=0 :: DISPLAY AT(15,1):"" ::
    DISPLAY AT(12,1):"INSERT COPY DISK AND
    PRESS":"ENTER"
200 CALL KEY(0,K,ST):: IF ST=0 THEN 200 ::
    DISPLAY AT(13,1):""
210 FOR J=1 TO W :: IF F$(J)=CHR$(2)THEN 230
220 DISPLAY AT(12,1):"FILENAME? DSK"&F$(J)::
    ACCEPT AT(12,14)SIZE(-12)BEEP:C$(J)
230 NEXT J :: FOR J=1 TO W :: IF F$(J)=CHR$(2)THEN 250
    :: OPEN #1:"DSK"&C$(J),OUTPUT ::
    DISPLAY AT(12,1):"SAVING "&SEG$(C$(J),3,255)
240 X=X+1 :: IF M$(X)<>T$ THEN PRINT #1:M$(X):: GOTO 240
    ELSE CLOSE #1
250 NEXT J :: END
260 ON ERROR STOP :: DISPLAY AT(22,1):"CANNOT OPEN
    "&SEG$(F$(J),3,255):: F$(J)=CHR$(2):: RETURN 180
```

Here is a very ingenious idea published in the Corpus Christi User Group newsletter by H. Macdonald. He could not find the author or newsletter which gave him the idea, so if you know, tell me and I will print due credit.

I have modified it a bit. This short routine will load quickly and enable you to bypass loading and running the Menu Loader program on a disk when you already know the filename of the program you want to run.

Save the Menu Loader under the filename MENULOADER and save this routine under the filename LOAD. Be sure to save it before you try it, because it erases itself!

```
100 CALL INIT :: CALL LOAD(-31806,16)::
    DISPLAY AT(12,1)ERASE ALL:"RUN MENULOADER? (Y/N)"
110 CALL KEY(3,K,S):: IF S=0 THEN 110 ELSE IF K=78 THEN
    130 ELSE DISPLAY AT(12,1)ERASE ALL:"LOADING
    MENULOADER" :: RUN "DSK1.MENULOADER"
130 CALL CLEAR :: CALL LOAD(-31952,55,215,55,215):: END
```

Here is one with a bit of a surprise at the end. Key the v,A in line 190 as FCTN[V], CTRL[,], CTRL[A].

```
100 CALL CLEAR :: CALL SCREEN(16)
110 DATA 80C0A09088445269,0000000000007EB1,
    0103050911224A96,0000000101010100,21409C2A492A1CC0,
    9999336600001824
120 DATA 8482395492543903,0000000000808080,
    E0B09880E7702010,18244281423C0000,0F190307E1020408,
    000000FF80808080
130 DATA 000F13E620221D00,0CFB34670A22DC00,814224FF,
    30DF2CC641443B00,00F0C86F0447B87F,000000FF01F901F9
140 DATA 80FF808686808686,00FF006666006666,
    00FF003F3F3F3F3F,01FF01F9F9F9F9F9,8086868086868093,
    00666600666600FF
150 DATA 006666006666600E6,3F3F3F3F3F3F3F3F,
    F9F9F9F9F9F9F9F9,0000000E01C3AE2,9380FF,FF00FF,
    E600FF0007080807
160 DATA 3F00FF00FF1988FF,F901FF00FF8744FF,
    1F09090FF3198AFC
170 FOR CH=96 TO 129 :: READ CH$ :: CALL CHAR(CH,CH$)::
    NEXT CH
180 DISPLAY AT(1,14)ERASE ALL:"`ab" ::
    DISPLAY AT(2,13):"cdefg" :: DISPLAY AT(3,14):"hij"
    :: DISPLAY AT(4,12):"klmnopq"
190 DISPLAY AT(5,12):"rsssstu" ::
    DISPLAY AT(6,12):"vwwwxyz{" ::
    DISPLAY AT(7,12):"|}}}~v,A" ::
    DISPLAY AT(9,12):"TIGERCUB"
200 DISPLAY AT(11,12):"SOFTWARE" ::
    DISPLAY AT(13,7):"156 COLLINGWOOD AVE." ::
    DISPLAY AT(15,7):" COLUMBUS OH 43213" ::
    CALL HIGHCHAR
210 GOTO 210
220 SUB HIGHCHAR :: FOR CH=32 TO 129 ::
    CALL CHARPAT(CH,CH$)::
    X$=SEG$(CH$,3,12)&SEG$(CH$,13,4)::
    CALL CHAR(CH,X$):: NEXT CH :: SUBEND
```

Thanks to Ramon Martinez in the Orange County User Group news letter - a double NEXT is accepted if the pre-scan is turned off.

```
100 J=1
110 !@P-
120 FOR J=1 TO 100 :: IF J/10<>INT(J/10)THEN NEXT J ELSE
    PRINT J :: NEXT J
```

A computer without a program is like a car without gas. If everyone who filled up at a self-service pump drove away without paying, how soon would all the gas stations be closed?

Memory full! Jim Peterson

---

continued from page 24

Tulin	Model	HDS	CYL	RWC	WP	Notes
	TL 226	4	640	640	640	Wedge servo - do not use on IBM XT controller
	TL 238	4	640			
	TL 240	6	640	640	640	
	TL 258	6	640			
	TL 326	4	640			
	TL 340	6	640			

Myarc Hard - Floppy Disk Controller Card

by Lou Amadio

(With extracts from the Myarc Hard and Floppy Disk Controller manual)

The introduction of the Myarc Hard and Floppy Disk Controller Card (HFDCC) has been the most significant hardware release since the introduction of the Horizon RAMdisk. The card was first released in 1988 and has undergone some modifications to the DSR during that time.

If you look at the specifications of the HFDCC you can be forgiven for wondering if Myarc has overdone it a bit. According to the manual:

> The Myarc HFDC is a powerful and versatile disk controller that will allow you to use up to three hard drives at a time with up to 134 Mb capacity on each drive. It will also allow you to control up to 4 floppy drives (40 or 80 track) which may be single or double sided and either 5-1/4 or 3-1/2 inch size or any combination of these sizes. The floppy drives may be any standard capacity up to 1.44Mb. By adding a second floppy disk controller, it is possible to control a further 4 drives for a total of 8 floppies.

In addition to the above, the card has been designed to allow Streaming Tape backup facilities. The software for this option has yet to be released.

Floppy Emulation

The HFDCC has the ability to emulate a floppy drive in a number of ways:

1) "DSK1." emulation allows the use of software that looks for a file in floppy drive #1.

2) "DSK." emulation allows the use of software that looks for a particular disk name, for example "DSK.TIMP" for Multiplan.

3) DSK1 file emulation: All of the files on a floppy disk are "cloned" onto the hard disk. Reading and writing to this area of the hard disk is similar to accessing a physical floppy drive.

The HFDCC must be used in a TI PE Box. It is also advisable to replace the standard plastic clamshell with a metal TI unit to aid in cooling the on-board voltage regulators. (See relevant article TND March 1989

Selecting a Hard Drive

Drives from XT or AT computers or their clones are suitable for the Myarc HFDCC. What to buy is generally a matter of what is available at the best price. Second hand 10 Mb drives from XT computers are very cheap, but require a lot more capacity from the power supply:

Drive Type	Max Amps (+12V)
5-1/4 in Full Height	5.0A
5-1/4 in Half Height	2.5A
3-1/2 in Half Height	1.5A

Warning: The PEB does not provide sufficient power for a hard disk. However, a low power hard disk may be powered from the PEB provided that the power supply is modified (see Micropendium, March 1988)

Connecting the Drive

Before connecting anything, examine your hard drive and write down any information on the drive, particularly the serial number, model number and any indication of the number of heads and number of cylinders. This information is required during the formatting process.

Hard disks have a provision for 2 ribbon cables: one for data (20 wires) and the other for control signals (34 wires). Apart from the high rotational velocity (approximately 3000 RPM), it is the separate data cable that contributes to the high speed access of a hard disk system.

Configuring the HFDCC

Ensure that the DIP switch settings on the HFDCC are correctly set: one is for configuring floppy drives and the other sets the Communications Register Unit (CRU) address that the card will occupy. The address, if changed from the default of 1100, must not clash with any other device CRU.

Configuring the Hard Disk

As with floppies, hard disks must be set up to respond to a drive identifier (eg WSD1 for the #1 hard disk) and only the last hard drive (if there is more than one) should have the terminating resistor pack. The 34 conductor cable that connects to the hard drive(s) must not be shared with any floppy drives.

Using the Hard Drive

The Myarc manual has step by step instructions on how to format and use the hard disk system. The only other advice is to ensure that you have a means of backing up your data as hard disks can crash in a big way without warning.

PS. Rolf, all I need now to enjoy the benefits of a hard disk system is a HFDCC.

Lou

Quick Hard Disk Back-up

by Garry Christensen, Brisbane

It is important to back up your hard drive on a regular basis but I am sure you are aware of the time that it takes. This program is not a replacement for a proper back-up but may suffice for the more regular ones.

If you are like me, you will not have used half of the hard disk. The other half is just sitting there (spinning actually) blank.

The BACKUP program will copy, sector for sector, the contents of the disk onto the second, unused half. If necessary you can recover the sectors and write them onto to first half.

I have tried to anticipate all situations that may occur when backing up your hard drive. The program checks that less than half of the disk has been used and issues an error if some of the contents will be erased.

At this stage, this program is designed for those drives with a clean bit-map. Sectors that have been marked as unusable may prevent this program from being used.

Warning: Performing a restore will over write the contents of the first half of the disk and any files created or edited since the back-up will be lost. Performing a restore without first backing the disk up will result in the loss of all data on the disk.

Instructions for use.

The operation of this program is as simple as following the on-screen prompts. You will first be requested to enter the drive number.

After this is done, select the function required, back-up or restore. This is done with FCTN[1] and FCTN[2] keys respectively to prevent the possibility of key-bounce or the speed of the Geneve 9640 from causing the selection of the wrong function.

If restore is selected you will be asked to type the word RESTORE to ensure that this is the desired procedure.

At this stage, a warning message is issued and you are required to press a key to continue. After a short delay for the calculation of parameters, the program proceeds with the task at hand.

Both the current sector being read and one being written to are displayed as well as the last sector to be used. When this sector is reached, the procedure is complete.

This program is distributed under the concepts of Fairware. Monetary donations are not necessary however letters from users or donations of software will be appreciated. Of course, if you feel the desire to throw your wealth at me I could put it to productive use. O

Adjust Emulate File Pointer

by Garry Christensen, Brisbane

This utility allows you to change the file that is active for the DSK1 emulate function of the Myarc HFDCC.

Extended BASIC

It can be used in several ways. The first is to execute the Extended BASIC program called 'EM'. Here you will be asked to input the device number, path name, and filename. An entry may look like this:

1.STORE.UTIL.GENERAL

This file must be an emulate file that has been established using MDM5. While there is error checking incorporated, errors are not reported. You are returned to the title screen regardless of whether the operation was successful.

The pointer to an emulate file can be cancelled by giving the device number only. This results in no emulate file being marked as active on the hard disk.

1

Entering the above name would result in the emulate file pointer on hard drive number 1 being cleared.

Editor Assembler

You will find a file called "EM/S1" included on this disk. This is the source code for this program that can be permanently set for one specific file.

Near the beginning of the program you will see the following lines:

```
NAME TEXT ' '
     BYTE 0
     EVEN
```

After the word TEXT and in between the inverted commas, insert the device number, path name, and filename as before. Assemble this code and the resultant object code can be used any time to set the file active.

You may wish to generate several of these programs for the more commonly used emulate files. Once again, the device number alone will de-activate all emulate files on the hard disk.

While all care has been taken, the author can accept no responsibility for data lost while using this program. No bugs have been located but be aware that this program bypasses all protection by using direct sector access. Ensure that your hard disk has been backed up onto floppy disk.

The source code for these programs can be found in Bug-Bytes, June 1989. O

Backing up the Hard Disk

by Ben Takach

Life is certainly much easier since the hard disk is part of my TI99/4A system. It works fast, it is convenient, saves a lot of diskettes and most important of all, I have instant access to every file.

Like other Myarc products released in the past, the hard disk control card and disk manager is in need of some polish, revision and update. The find option of the disk manager is somewhat primitive, thus one has to remember and type in the exact file name, but it will locate the requested file quite fast.

One of the less well understood feature of the hard disk system is the backup procedure. It is not a big job to produce a backup copy of a floppy disk. Backing up a 1440 sector DSDD diskette may be completed in 10 to 12 minutes, including verified formatting. Such disk may hold up to 360 kbyte of data. Backing up a hard disk with 10 Mbyte of information stored under 30 to 50 sub-directories would take at the above rate about 8 hours. This would be more acceptable if the backup process would be automatic. Unfortunately one has to sit there and constantly change diskettes. This is the paradox of the super fast electronic information transfer. One can save a lot of time whilst producing, gathering and transferring vast amount of data, and then has to waste a full working day to protect it.

This becomes all the more evident for the owners of the Myarc hard disk control card. The announced streamer tape backup is not yet supported by the so far released software or firmware. Using the resident diskette backup routine is painfully slow and it eats up a lot of disk space. The system manual makes no mention of the formatting details, however it appears the density is less than 18 sectors/track. Further, if one encounters a read or write error half way through the backup process then there is no manual option to find and exclude the offending file. One has to abort the process with several hours down the drain.

Although the file by file backup option is more time consuming, one may save those several hours in the long run by being able to monitor the process, and an eventually corrupted track or file is easily pinpointed. The file by file backup is best combined with the technique of archiving to save diskette space.

This method proved to be the best of the many unfavourable choices, as I found out after many wasted hours and frustration. I still had to spend 8 long hours at the keyboard, but managed to backup 8 Mbyte of data on 11 floppies. This method has one unavoidable disadvantage. One does not and cannot backup the directory structure. Thus the complete directory structure has to be manually recreated after a hard drive failure. Creating 40 to 50 directories is also a lengthy task.

How does it work?

I used The Extended BASIC version of ARCIII. This program does not recognize the hard drive, thus one has to first transfer the files to a valid drive. I used 2 drives (drive 1 and 3) and a RAMdisk (drive 2). The MDM files (the hard disk control program files) are saved to DSK1. directory of the hard disk except its LOAD program. The ARCIII_XB program was renamed LOAD and saved to disk and placed in drive 1 (the MDM program files may also be saved to this disk again minus the LOAD program). The RAMdisk (drive 2) was used for source and target disk whenever space permitted it, else drive 3 was the nominated target drive. CALL MDM from console BASIC loaded the disk manager and all the files of one directory were copied to the RAM disk. I selected Extended BASIC option after quitting out of MDM, thus ARCIII was booted at once. The archived file was subsequently copied to the backup disk in drive 3.

continued on page 26

Hard Disk Data

from Sparta PCBoard Hard Disk Conference, updated by Bob Arnold of Random Access BBS.

This hard drive data is supplied as a guide only. Drive makers can and quite frequently do change drive models and specifications without notice. The author assumes no responsibility for the use of this information in any form.

Abbreviations: HDS - Heads
CYL - Cylinders (Tracks)
RWC - Reduced Current Write Compensation (0 indicates RWC on all the time)
WP - Write Pre-Compensation - Usually 0 or number of CYL's

Where specifications are unknown the box is left blank.

Atasi	Model	HDS	CYL	RWC	WP	Notes
	AT3020	3	635	0	0	
	AT3033	5	635	0	0	
	AT3046	7	635	0	0	
	AT3051	7	733	0	0	
	AT3085	8	1024	0	0	

Bull	Model	HDS	CYL	RWC	WP	Notes
	D530	3	987	987	987	
	D550	5	987	987	987	
	D570	7	987	987	987	
	D585	7	1166	1166	1166	

CDC	Model	HDS	CYL	RWC	WP	Notes
Wren I	9415-5-21	3	697	697	0	
	9415-5-36	5	697	697	0	
Wren II	9415-5-48	5	925	925	0	Drive connector J1 pin 2 may need to be disabled when used on an AT
	9415-5-57	6	925	925	0	
	9415-5-67	7	925	925	0	
	9415-5-77	8	925	925		
	9415-5-86	9	925	925	0	
	9415-5-25	4	615	615	300	
	9415-5-38	5	733	733	733	
	9420-5-30	3	989	989	0	
	9420-5-51	5	989	989	0	

CMI	Model	HDS	CYL	RWC	WP	Notes
	CM 3426	4	612	612	612	
	CM 5205	2	256	256	256	Wedge servo - do not use on IBM XT controller
	CM 5410	4	256	256	256	
	CM 5616	6	256	256	256	
	CM 6426	4	615	256	256	
	CM 6426S	4	640	256	256	
	CM 6640	6	640	640	256	

Fujitsu	Model	HDS	CYL	RWC	WP	Notes
	M2230AS	2	320	320	320	
	M2233AS	4	320	320	320	
	M2234AS	6	320	320	320	
	M2235AS	8	320	320	320	
	M2230AT	2	320	320	320	
	M2233AT	4	320	320	320	
	M2241AS	4	754	754	754	
	M2242AS	7	754	754	754	
	M2243AS	11	754	754	754	

Hitachi	Model	HDS	CYL	RWC	WP	Notes
	DK511-3	5	699		300	
	DK511-5	7	699		300	
	DK511-8	10	823		400	

IBM	Model	HDS	CYL	RWC	WP	Notes
	20 Meg	4	615		128	
	30 Meg	5	733		300	

IMI	Model	HDS	CYL	RWC	WP	Notes
	5006H	2	306	306	214	
	5012H	4	306	306	214	
	5018H	6	306	306	214	

Irwin/Olivetti	HDS	CYL	RWC	WP	Notes
416	2	819			
510	2	628			+- HD/TAPE
516	2	819			+- HD/TAPE
HD561	4	180	128	180	

LaPine	Model	HDS	CYL	RWC	WP	Notes
	Titan20	4	615			

Maxtor	Model	HDS	CYL	RWC	WP	Notes
	XT-1065	7	918	918	918	When attaching the XT-1065 & XT-2085 to the IBM AT, you may need to disable drive connector J1 pin 2.
	XT-1085	8	1024	1024	1024	
	XT-1105	11	918	918	918	
	XT-1140	15	918	918	918	
	XT-2085	7	1224	1224	1224	
	XT-2140	11	1224	1224	1224	
	XT-2190	15	1224	1224	1224	

Micropolis Model	HDS	CYL	RWC	WP	Notes
1302	3	830	830	400	
1303	5	830	830	400	
1304	6	830	830	400	
1323	4	1024	1024	1024	
1323A	5	1024	1024	1024	
1324	6	1024	1024	1024	
1324A	7	1024	1024	1024	
1325	8	1024	1024	1024	
1353	4	1024			
1353A	5	1024			
1354	6	1024			
1354A	7	1024			

MicroScience Model	HDS	CYL	RWC	WP	Notes
HH 325	4	615			
HH 612	4	306	306	306	
HH 725	4	612	612	612	
HH 1050	5	1024			

MiniScribe Model	HDS	CYL	RWC	WP	Notes
MS 1006	2	306	153	0	
MS 1012	4	306	154	0	

	Model	HDS	CYL	RWC	WP	Notes
	MS 2006	2	306	306	0	
	MS 2012	4	306	306	0	
	MS 3012	2	612	612	128	
	MS 3212	2	612	612	128	
	MS 3412	4	306	306	128	
	MS 3425	4	612	612	128	
	MS 4010	2	480	480	0	
	MS 4020	4	480	480	0	
	MS 6032	3	1024	1024	512	
	MS 6053	5	1024	1024	512	
	MS 6074	7	1024	1024		
	MS 6085	8	1024	1024	512	
	MS 8212	2	615	615		
	MS 8425	4	615	615		

Mitsubishi	Model	HDS	CYL	RWC	WP	Notes
	MR 522	4	612		300	
	MR 533	3	971			

NEC	Model	HDS	CYL	RWC	WP	Notes
	5124	4	310	310	128	
	5126	4	612	612	128	
	5146	8	615	615	128	

Newbury	Model	HDS	CYL	RWC	WP	Notes
	NDR 320	4	615			
	NDR 340	8	615			
	NDR 1085	8	1024			
	NDR 1105	11	918			
	NDR 1140	15	918			
	NDR 2190	15	1024			

Priam/Vertex	Model	HDS	CYL	RWC	WP	Notes
	V130	3	987	987	987	
	ID40 (V150)	5	987	987	987	
	ID60 (V170)	7	987	987	987	
	V185	7	1166	1166	1166	

Quantum	Model	HDS	CYL	RWC	WP	Notes
	Q520	4	512	256	256	
	Q530	6	512	256	256	Wedge servo - do not use on IBM XT controller
	Q540	8	512	256	256	

Rodime	Model	HDS	CYL	RWC	WP	Notes
	RO 101	2	192	96	0	
	RO 102	4	192	96	0	
	RO 103	6	192	96	0	
	RO 104	8	192	96	0	
	RO 201	2	320	132	0	
	RO 202	4	320	132	0	
	RO 203	6	320	132	0	
	RO 204	8	320	132	0	
	RO 201E	2	640	640	0	
	RO 202E	4	640	640	0	
	RO 203E	6	640	640	0	
	RO 204E	8	640	640	0	

RMS	Model	HDS	CYL	RWC	WP	Notes
	RMS 503	2	153	77	77	
	RMS 506	4	153	77	77	
	RMS 512	8	153	77	77	

Seagate	Model	HDS	CYL	RWC	WP	Notes
	ST 206	2	306	128	128	
	ST 212	4	306			
	ST 213	2	612			
	ST 225	4	615	615	300	
	ST 238	4	615	615	300	Use RLL controller
	ST 251	6	820		128	
	ST 277	6	820		820	Use RLL controller
	ST 406	2	306	128	128	
	ST 412	4	306	128	128	
	ST 419	6	306	128	128	
	ST 506	4	153	128	128	Unbuffered seek
	ST 706	2	306	128	128	
	ST4026	4	615	615	615	
	ST4038	5	733	733	733	
	ST4051	5	977	977	977	
	ST4096	9	1024	1024	1024	

Shugart	Model	HDS	CYL	RWC	WP	Notes
	SA 604	4	160			
	SA 606	6	160			
	SA 612	4	306			
	SA 712	4	320	128	128	

Syquest	Model	HDS	CYL	RWC	WP	Notes
	SQ 306 RD	2	306	306	306	
	SQ 312 RD	2	615	615	615	Remove jumper W3
	SQ 325 F	4	612	612	612	
	SQ 338 F	6	612	612	612	

Tandon	Model	HDS	CYL	RWC	WP	Notes
	TM 252	4	306	306	306	
	TM 262	4	615	615	615	
	TM 362	4	615			
	TAN 501	2	306	128	153	
	TAN 502	4	306	128	153	
	TAN 503	6	306	128	153	
	TM 602S	4	153	128	153	
	TM 603S	6	153	128	153	
	TM 603SE	6	230	128	128	
	TM 702AT	4	615	615	615	
	TM 703	5	695	695	695	
	TM 703AT	5	733	733	733	
	TM 705	5	962			
	TM 755	5	981	981	981	

Toshiba	Model	HDS	CYL	RWC	WP	Notes
	MK53FA	5	830	830	512	Drive connector J1 pin 2 may need to be disabled for AT
	MK54FA	7	830	830	512	
	MK56FA	10	830	830	512	

continued on page 20

HardMaster

Distributed by Asgard Software.

by Col Christensen, Brisbane

HardMaster, Version 2.0, is a sector manager for use with the Myarc Hard-floppy Disk Controller card and comes on disk as memory image files with filenames of HM and HN. It loads quickly using Editor Assembler option 5 type loaders. Obviously to use the program, one needs an expanded TI99/4A system or 9640 Geneve, the Myarc Hard Floppy Disk Controller Card and at least one hard drive. A printer is useful but not necessary.

On loading the program, the user is given the option of which hard drive number to access. Does the sound of having more than one hard drive make your mouth water? After that number is input and checked to be valid and available, a list of command options is displayed as shown below. At the completion of any command, this list of options will be reprinted on the screen making it unnecessary to have to remember the commands and the parameters each needs.

```
EDit s   PRint s s    QUad s   DIrectory
FInd s s w/$  PAck s b   AScii s s   MAp
HDrv n   FDrv n   OutDev   TRee   FB b   EXit
```

The postscript, "s", stands for sector, the "w" for word of 2 bytes, the "b" for byte, the "$" for ASCII string and the "n" for number.

Command inputs all must begin with a 2 letter command name and followed, in most cases by parameter numbers in hexadecimal but do not use the ">" sign before the numbers. The command and each parameter must be separated by one space character.

All output to the screen or printer uses the hexadecimal format with one exception. That is in the record length of fixed or variable files (for example, D/V 80). We are used to seeing them as decimal numbers in all common directory listings and for convenience, this has been maintained.

A status line is displayed at the bottom of the screen during most phases of operation of the program. The status line shows which drive has been selected, the disk name, the disk size and the sector number being accessed. The status line is continually being updated as different sectors are being read.

The running of some functions can be interrupted by pressing a key to pause and press again to resume. Those functions with a pause facility also have been provided with the opportunity for the user to BREAK altogether. To do this press FCTN[4], FCTN[9] or CTRL[C]. If output is going through the RS232 card and you press FCTN[4], the DSR routine in the RS232 will detect this keypress, perform a break, and give an error code which will appear on the screen as "Device error". In this case the printer file is not closed and information may still remain in the printer buffer. The next time the printer is used, the first part of the printout may not be as it should be. This is no fault of HardMaster and can be prevented by BREAKing a printout with the FCTN[9] or CTRL[C] combination that HardMaster looks for.

By the way, most commands in this program are used to read the contents of disk sectors and they cannot change any of the contents of a disk unless you intend it that way. Only two commands allow changes to the disk but, before this happens though, you must consciously change the default input from N to Y.

The Commands

HD and FD options allow the user to change the drive number you wish to access. Hard drive number input allows 1 to 3 while Floppy drive allows for drives 1 to 9. If you have a DSK1 emulation file on the hard disk, this can be accessed as FD.1 and it would be assumed that the first mechanical floppy drive would be configured to DSK some other number. Horizon RAMdisks set to any drive number up to 9 and any CRU address can be accessed.

The EDit s command is the most versatile and will be the most used. When you suffix the ED command with a sector number and press the <ENTER> key, the contents of the sector will be displayed in hexadecimal on the screen together with instructions on key usage. The CTRL key is used in combination with W to Window from hexadecimal to ASCII, with B to go Back one sector, with N for Next sector higher and with C for esCape to command mode.

Although you are not told on the screen, the FCTN[5], FCTN[6], FCTN[4] and FCTN[9] are also active as these keys on the Geneve are so easy to use. In EDit you can skim the cursor over the sector display with the FCTN and arrow keys or you can type over the contents on either the hexadecimal or ASCII window. On the hexadecimal window, you are allowed to type only hexadecimal digits.

The only ways out of this command are CTRL[C], the escape keystroke or FCTN[9], the BACK key combination. If you have previously typed over any of the sector contents on the screen purposely or accidentally, you will be given the option to save or not to save the changes to disk.

Sector number inputs when using the hard drive can be from one to five digits up to FFFFF and for the floppy drives up to 4 digits. In either case trying to access a sector number greater than the disk size as shown on the status line at the bottom of the screen will result in an error message.

PRint s s will print out to the output device name in memory the contents of the sectors from the start sector to the end sector entered as parameters. You get a printout of both the hexadecimal contents and also its ASCII representation. Of course some values in hexadecimal have no ASCII counterpart so a dot is printed instead.

OutDevice allows you to change the output device name from the default of PIO. This command needs no parameters. After typing OD, and pressing <ENTER>, you are asked to type in the name of the device to which you want the output to go. This can be through the RS232 or to a floppy or RAMdisk file. Output cannot go to the hard drive as the Peripheral Access Block used for the printer differs from the requirements of the hard drive.

QUad is based on the Forth system of displaying 4 sectors at a time on the screen. You can step through sectors quite quickly this way if you are searching for some known visual characteristic of a disk sector. Because the contents of four sectors will not fit onto a forty column screen, horizontal windowing is employed. The same CTRL keys as for EDit are active and FCTN[4], FCTN[5], FCTN[6] and FCTN[9] are too.

The DIrectory report contains information to include: the filename; the number of sectors that the file occupies; file type abbreviated to 3 characters; length of record if applicable (the only numeral in the program shown in decimal notation); the sector number that the file descriptor record (FDR) is stored on; and the sector clusters where the actual file is stored on the disk. If the file on the disk is fractured, more than one cluster is shown.

Directory is a multi-purpose routine. It automatically categorizes your disk into one of three types. In all cases you will first have to indicate whether you want the output to the screen or to the printer. If you have changed the default from PIO to a disk filename, then that is where the output will go. What the routine does in each of the three circumstances varies as shown below.

1. Hard drive. You will have to type a path name to read a directory on the hard drive. The hard drive caters for a root directory as well as sub-directories, each of which may have pointers to more sub-directories. To access the root directory, just type the path name, ROOT. To access a sub-directory, such as for example, ADV, type the full path name (for example, TI.GAMES.ADV) placing a period between each, but no period at the end even though the last entry is a sub-directory. Including the disk name in the path name is not required.

2. Floppy drive formatted by Myarc Disk Manager 5. This format, a little like the hard drive, has a root directory and can have also up to three sub-directories. When using the command, DI, with this type of disk, HardMaster checks the disk and if there are sub-directories as well as the root directory, presents an option list from which to choose. If no sub-directories are found, then the root directory is automatically processed.

3. Floppy drive initialized TI-style. When you want a directory of this type of disk, the program goes straight on to show the directory.

FInd s s w/$ - a search of the disk is made from the starting sector, the first "s", to the ending sector, the second "s", for the 2-byte word or the ASCII string of any length that is entered. The two byte word or string is looked for at both even and odd addresses and the search can even detect a match formed by bridging the last of one sector and the beginning of the next. Output of results can be directed either to screen or printer.

TRee needs no parameters and produces a list of directories, emits a beep, and then a full list of directories and their associated filenames with the sector numbers that contain their DDRs and FDRs. If you only want the directory names, then the beep indicates when to break out of the routine. Output can be directed to the screen or printer. The printout is tabbed over to start with a left margin of 10 by using the printer code of <27>"1"<10>. This margin makes the printout more suitable for filing in a ring binder or similar folder.

AScii s s is the printout of the sectors specified in your input. The two parameters following the AS command represent the starting and ending sectors to be accessed. Only the ASCII representable characters are printed in 4 lines of 64 characters. Byte values below 32 and above 127 are printed as spaces. Maybe one day you cannot access a text data file in its entirety but you can print out most of the accessible sectors and so recover on paper or disk much of the file. The disk file can be edited with TI-Writer.

MAp directs to the output device the sector allocation table or bit-map of the disk in the current drive. With big differences in the bit-maps of hard drives and floppy drives, the program automatically selects the form of printout suitable for the particular kind of drive.

The floppy drive printout will show each sector on the disk and whether it is marked off as used or free. One thing to be aware of in interpreting the printout is that the bytes on sector 0 that represent used and free sectors are bit reversed from the order of the sector usage on the disk. For example, the byte 3F (00111111) on sector 0 will indicate the following series of sectors to be used(X) and free(-) with XXXXXX--.

It is not practicable to show the sector allocation table of a hard drive as individual sectors because of the immense amount of paper and time needed to print it. The printout has been confined to listing the hexadecimal bytes of the bit-map plotted against the sector numbers that they represent. Remember in interpreting the table to take into account the number of sectors your hard drive allocates for each AU. Not like the TI99/4A bit-map, each byte representing 8 AUs is not bit-reversed.

Note that the MAp procedure for the hard drive begins by reading the bit-map sectors from >1F backwards towards sector 1 until it comes to a sector containing non-zeros. It then assumes, well the author does anyway, that this is the last of the used bit-map and prints out the allocation table from sector 1 onwards until the end of this sector. Very good in theory but unfortunately your hard drive may have contained bad sectors when it was formatted, and parts of the bit-map towards the logical end of the drive accordingly could have been marked off as used. If MAp prints out the allocation table past the obvious end of used bit-map, then it can be interrupted with FCTN[9] or CTRL[C].

PAck s b will not have a great deal of use but can be handy when re-writing a dud directory by first packing the sector with zeros. (I avoided the use of a command name, Fill, because Fl is already used for FInd.) "s" is the sector which has to be packed and b is the byte for each of the >100 places. Before the command is executed to disk, you must confirm your intention by changing the default option of N to Y.

FB b allows a change of screen colours. Foreground and Background colours will be set by the 2 nybble parameter suffixed to the command. The default colours are F4 with F the white foreground and 4 the blue background.

EXit is provided as an "out" because the normal quit key has been disabled. If you entered HardMaster from Funnelweb then this program keeps Funnelweb's colours and on exit, it returns to Funnelweb. This feature makes it a suitable program to run through Funnelweb's load option screens. o

continued from page 22

One can judge the size of the archived file; archiving saves roughly 50% of disk space. In my case the files filling 30238 sectors of hard disk space were backed up on 11 diskettes, the archived files using up 14692 sectors. To sum up, it was a tedious work, I sure will be glad when Myarc gets around to release the software support for the streamer tape backup. o

continued from page 16

Smoley Martin A. 0713831			
01/21/88	800.11	800.22	0713831
02/29/88	200.11	200.22	0713831
06/17/88	1000.11	1000.22	0713831
08/03/88	1200.11	1200.22	0713831
09/11/88	3200.44	3200.88	TOTAL
Aardvark Grant E. 0717851			
06/06/88	600.11	600.22	0717851
08/27/88	300.11	300.22	0717851
09/11/88	900.22	900.44	TOTAL
Jones Quincy W. 0820871			
03/03/88	400.11	400.22	0820871
05/12/88	900.11	900.22	0820871
09/11/88	1300.22	1300.44	TOTAL
Whitman Raymond (Slim) A. 0921861			
12/30/87	500.11	500.22	0921861
03/01/88	1100.11	1100.22	0921861
04/22/88	700.11	700.22	0921861
09/11/88	2300.33	2300.66	TOTAL

o

The Hard Disk Drive

a guided tour, by Garry Christensen, Brisbane

At regular intervals a product comes along that has the potential of changing the way everybody uses their TI99/4A. I remember the upheaval that occurred when Horizon introduced the RAMdisk. A similar situation is upon us with the Myarc Hard/Floppy Disk Controller Card.

I do not think that anyone who has seen this device in action has not said "I must get one of those", even if it has been qualified with "one day".

Myarc did well with the manual that accompanies the device, giving instructions for the use of MDM5, interfacing with BASIC, hardware specifications and low level access. It is this final point that I wish to concentrate on here.

Unlike the documentation that accompanied much of the TI99/4A equipment, Myarc have provided considerable detail about accessing the hard disk with assembly language. The manual lists the parameters required and the addresses to be used when utilizing the on-board sub-programs. It also details the layout of the data in the important sectors on the hard drive. Whilst this data is supplied, it must be considered as the 'minimum necessary'. If you intend to make any use of the data, there is some investigation left to be done.

Loading The Sectors.

The first thing that has to be done before examining the sectors is to load them into memory. This is not as difficult as it might first seem because we have at our disposal a wonderful program called SuperBug II, by Edgar L. Dohmann. Load SBUG (or another comparable debugging program) by using the Editor Assembler 'Load and Run' option. Do not use option 5 because there are no utilities loaded into memory with this option.

Dump the REF/DEF table by using the command "D 3F80,3FFF". Find the name DSRLNK and note the word that immediately follows it. This is the entry address for the routine.

For the following section, refer to page 46 of the HFDCC manual. Using the "M" command load these values into memory starting at >834A: >0000, >8101, >F000, >0000, >0000, >0000, >3000.

The first word is used to return the sector number actually read from a floppy disk and is not used for hard disk. The next byte is the unit number (hard drive 1, in this case) with the first bit set to 1. This indicates that the data from the disk is to go directly into CPU RAM. The following byte is the read/write flag, >01 to read and >00 to write.

The next word is the address that the sector is to be loaded into. CPU memory is specified in the previous bytes so the sector will load into CPU RAM starting at >F000.

The sector number is next. The highest sector number that can be represented with a 2 byte word is 65535. My 20Mb drive has 78720 sectors so there is clearly a problem when you have a drive bigger than about 15Mb. Two words are used to specify the sector number. Consider the number being right justified in 32 bits eg. >00013380. The least significant word is at address >8350 and the most significant at >8352.

For the first access that we are going to do, both are zero. If however you want sector >20, the words would be >0020,>0000.

>8354 is unused at this time and >8356 points to the VDP address of the op-code that we wish to use. For floppy disk, the op-code for sector access is >10 and for the hard disk it is >20. A length byte must be included, obviously >01.

After setting >8356 to >3000, exit the change memory function of SBUG. Re-enter the same function with the VDP address, "M 3000V". This time each byte must be entered separately, >01 first then >20.

Almost finished. All that is left is to execute the DSRLNK routine. Check the REF/DEF table again and find the address for the DSRLNK routine. The value that I get is >2120. Once again we need to change some memory, this time in CPU. "M E000" will get us started. This address is to be the start of a very small program to branch to the DSRLNK. Enter these values:

>0420,>2120,>000A

When the above 3 words are dis-assembled they produce:

```
BLWP @DSRLNK
DATA >000A
```

If your address for the DSRLNK routine is different, use it. Lastly set a breakpoint at >E006 so that the computer will not go wandering off on its own. Set the workspace to >E800, the program counter to >E000 and the status byte to >0000. Use the "R"function for this.

Before going any further, check that you have done everything correctly. Is it OK? Good, now enter "E" to execute. You will see the HFDCC come on, a flash from the hard drive then SBUG will tell you that it has encountered a breakpoint.

Dump >100 bytes from >F000 with "D F000,F100". Surprise, there is the data from sector 0 of your hard disk.

Play around with this a little before going on. Reset the values using a new sector number, set the breakpoint and the program execution parameters and go again. Look at the results. You have at your disposal a primitive sector editor because to rewrite the sector you only have to change the read/write flag at >834D to >00.

Before you change too much, it may be a good idea to know what you are looking at. Here is a dump of part of my sector 0.

```
4844 2020 2020 2020 2020 99C0 2020
013A 131D 13A5 B24F 1107 0020 021A
0087 0072 007E 007C 006C 0081 0035
0000 0000 0000 0000 0000 0000 0000
```

Before I go any further I must mention something called an Allocation Unit (AU). An allocation unit is a block of consecutive sectors. There may be up to 16 sectors in one allocation unit. The disk is considered as a series of allocation units, the size of each being determined by the size of the disk.

All pointers are one word so the maximum number of allocation unit's on the disk is >FFFF. If it has a large storage capacity, there is a larger number of sectors per allocation unit.

Allocation units are only used to find a file. All directory descriptor records (DDR) and file descriptor records (FDR) are located on the first sector of an allocation unit. The remainder of the allocation unit is rarely used. All files start at the first sector in an allocation unit and use consecutive sectors, unless the file is fractured.

Volume Information Block and Directory Descriptor Records.

OK, let us go through sector 0. Refer to page 62 in the manual for this section.

The first 10 bytes are set aside for the name of the hard disk, that much will be obvious. The next word is the total number of allocation units on the disk. Remember that the number of sectors on the disk is this number multiplied by the number of sectors per allocation unit (more about that in a minute).

Byte 12 is the number of sectors per track. The following three bytes are a mystery. The manual says that they should be "WIN" but this is not the case. They have been used to store some data necessary for the operation of the disk.

Bytes 16 and 17 are also formatting parameters. These are listed in the manual however only one part is of interest to us. The first nybble is the number of sectors per allocation unit minus 1.

Look at the example above. You will see that the word is >131D. The first 1 means that I have 2 sectors per allocation unit (1+1). All pointers and values must be multiplied by 2. If you have a calculator that will do hexadecimal arithmetic, get it out now. If you do not, here is a great chance to practice on paper.

You will see that I have >99C0 AUs on the disk and 2 sectors per AU. A total of >13380 sectors (78720 decimal).

The next two words contain the date and time of creation. The time is first. Write the value down in binary notation. The first 5 bits are the number of hours (24 hour clock), the next 6 are the minutes and the final 5 is the seconds divided by 2. When you catalog your disk you will see that the seconds are always even.

Do the same with the date. The first 7 represent the year, the next 4 are the month and the final 5, the day.

In the printout of my sector 0 the words >13A5,>B24F are the time and date. Breaking them into binary gives 0001 0011 1010 0101, 1011 0010 0100 1111. When split they look like this:

```
00010    2 Hours
011101  29 Minutes
00101   10 Seconds (x2)

1011001 89 Year
0010     2 Month
01111   15 Date
```

Byte 22 is the number of files in this directory and byte 23 is the number of sub-directories. I doubt that this needs any further description.

There is no room in this sector to store the pointers to all the files that may be under this directory. Another sector is used just to store these pointers, called the file descriptor index record. The word starting at byte 24 is the AU of this list. Remember that this number must by multiplied by the number of sectors per AU to get the physical sector number.

Check your disk and have a look at the FDR. It should look something like this:

```
026D 0267 01B4 01B5 01B6 01B7 01B8
01B9 0085 0215 0216 0217 0218 0219
0086 0021 01BB 0000 0000 0000 0000
```

Each word is the AU of the file descriptor record for each file. We will come to FDRs shortly. All entries are sorted into alphabetical order. When a new file is added, it must be placed in the correct place in this chain of pointers. The 128th entry in this record always points back to the DDR that it belongs to.

As you are aware, the hard disk can emulate DSK1 in a number of ways. One method is called DSK1 file emulation. In this method, the entire floppy disk is copied sector by sector and is used on the hard disk just as it would be on a floppy. The word at byte number 26 is the pointer to the emulate file that is active. The actual value given is the AU of the file descriptor record. If it is >0000, there is no file active. The demonstration program deals with this word.

Finally there is the list of pointers to the DDRs of all the sub-directories. Each pointer gives the AU of the DDR. In the example of the volume information block above, the first entry points to the first directory, and so on until a 0 is reached. These entries are also sorted in alphabetical order.

The DDRs are the same format as sector 0 (the volume information block) except that "DIR" is present and the emulate pointer is replaced with a pointer to the parent DDR. The parent DDR is the directory that contains this directory. Following this path of pointers will lead back to the VIB.

Sector Usage - the Bit Map.

All disks must keep record of which sectors are used. The hard disk is no different. The bit map starts on sector 1 and is quite simple to follow. Each bit represents an allocation unit. If the bit is 1, the AU is used. The bits are taken in order from most significant to least significant. To calculate the actual sector number represented there is a little mathematics to do.

Take the sector number that you are looking at and subtract 1. Multiply this number by >100 and add the number of the byte. Multiply that by 8. The result is the AU represented by the first bit in that byte. By means of an example, on sector 6 the >50th byte in the sector may be >80.

(((6-1)*>100)+>50)*8 = >2280

In this way you can calculate the position of any used or unused sector on the disk. Look for the end of your bit map, calculate the sector number and have a look to see if you are right.

An extra feature that the hard disk has over and above the floppy drives is a separate bit map for the DDRs and FDRs. On my disk, all the first sector has been set aside for DDRs and FDRs. The files do not begin until sector >1000. When a DDR or FDR is created, it is put in this area and the sector marked off in sector 1. Unless the disk has a large number of files, all the descriptor records are at the beginning.

File Descriptor Record.

If you have followed through this far you will be able to track down any file descriptor record on the hard disk. The FDR will direct you to the file itself. Refer to page 64 in the manual.

The following is an example of one of the FDRs on my hard disk.

```
5359 5354 454D 2020 2020 0000 3900
05A0 0000 0000 61BC B266 61BC B266
4649 0000 0000 02D0 0020 0000 0894
0B63 0000 0000 0000 0000 0000 0000
```

The first 10 bytes are the name of the file. Bytes 10 and 11 give the record length if it is greater than 255 bytes. At this stage there are no programs for the TI99/4A that will take advantage of this, however MDOS for the Geneve 9640 does have support for this type of file.

Byte 12 describes the file attributes. I will not be listing all the functions of the bits here because they are well described in the manual. Please note that in the manual, bit 0 is the least significant bit. The example above indicates that the file is a program, protected, has been modified since the last back-up and is a DSK1 emulate file.

Byte 13 indicates the number of records per sector.

Bytes 14 and 15 tell you the number of sectors allocated to the file. A very large file may have more that 65535 allocated to it so the most significant nybble is stored in the most significant nybble of the extended information area. This is the MSN of byte 38. Place this value in front of the value in bytes 14 and 15 to give the true file length.

The value of byte 38 is 0 and the length is listed as >05A0. The total length in sectors is >005A0.

The end of file offset is located at byte 16. Variable length and program files may not necessarily end at the end of a sector. This byte indicates how many of the bytes in the final sector are to be included in this file. There will often be garbage in the remaining unused bytes. In the above example the offset is 0 so the file does end at the end of the sector.

The logical record length in byte 17 is set to the record length for fixed length records, the maximum length of a record for variable length records and 0 for program files and the data files that make use of extended record lengths. In the case of DV80 files, this byte is set to 80.

Bytes 18 and 19 and the second nybble in byte 38 indicate the number of sectors actually used in this file. If you open a file and specify its size, that number of sectors is allocated for its use. You may not have written to all the sectors at that stage. The value derived from this part of the FDR indicates the highest sector written to for variable length files and the highest record written to in the case of fixed length files.

The next 4 words are the date and time that the file was created and last updated. I have covered the encoding of time and date already.

Bytes 28 and 29 are always "FI".

On a hard disk, fractured files are quite common. Long files may be split into many fragments. I once assembled a large source code and sent the list file to disk. The assembler wrote a small portion of object code then some list file, then some more object code and so on. The resultant object and list files were tightly interwoven.

To allow for the case where there is not enough space for pointers to all the parts of the file in one FDR, the controller creates another FDR in which to continue the table. This is called an offspring and the original is the parent.

Bytes 30 and 31 are the pointer to the AU of the parent FDR. If this is the parent, this value is 0. Bytes 32 and 33 point to the AU of the offspring FDR. This chain of parent and offspring FDRs can continue until the file is stored or the disk is filled.

Where there are more than 1 sector per allocation unit, the offspring FDRs may be placed on any unused sector within the AU. If this is the case, the most significant nybble of byte 39 points to the sector number within the AU of the parent FDR and the least significant nybble points to the sector of the offspring FDR.

The above example is the parent FDR and it has no offspring.

The word starting at byte 34 gives the total number of AUs used for the FDRs for this file and the next word is the pointer to the file descriptor index record. This record contains pointers to all the FDRs in this directory and the pointer to it allows the user to follow the path back to directory. The example shows that the file descriptor index record is at AU >0020.

The function of the bytes in the extended information area have already been covered.

Up to this point, there have been few differences from the FDRs on a floppy disk where the functions are common. The pointers to the file are different. The hard disk points to the AU at the start of the file fragment and points to the last AU used. These two pointers are called a cluster.

The clusters are 2 words each and occupy the rest of the sector from byte 38. The chain of clusters is ended by a 0. Only one cluster is shown in the example. This file is continuous from AU number >0894 to >0B63.

With the information above you should be able to load any sector from the hard disk and to follow any valid path from sector 0 to the file that you are looking for. Are there any uses for this information?

You may wish to write a sector editor for the hard disk or modify one that is used for floppies. A sector editor for both the hard and floppy disks is presently under development in Australia and will be available soon. here may be other simple utilities that could be useful. There is one such program listed below as a demonstration of the information that I have discussed.

Demonstration - Change Emulate File Pointer.

It is not unusual to have several DSK1 emulate files on the hard disk. Only one can be active at one time. This program is loaded from Extended BASIC and will change the active emulate file without going through MDM5.

The device number, path name and filename are passed to the assembly program from Extended BASIC, the file is located and the AU of the FDR is written into the pointer in sector 0.

In Extended BASIC the following program can be used.

```
100 CALL CLEAR
110 CALL INIT
120 CALL LOAD("DSK1.EM/0")
130 INPUT "ENTER #.Pn.Fn":A$
140 CALL LINK("START",A$)
```

In simple terms the assembly portion can be listed as follows:

Get filename.
Get number of sectors per AU.
Get list of directory pointers.
Point to next name in the path name.
Find the DDR.
Repeat the above steps until the filename is reached.
Find the FDR.
Check that it is an emulate file.
Load sector 0, change pointer and write sector 0.

The routine to find the DDR or FDR is simply:
Get list of pointers.
Get first pointer.
Get sector.
Look for match of names.
Next pointer and repeat until found or no more pointers.

Please note that the format of the string passed to the program is important. An example may be "1.TEMP.BLANK". The "1" indicates which hard drive is to be used. Entering only the device number (eg "1") will result in the pointer being cleared. No DSK1 file emulation will be active.

There is error trapping, however it is not reported. I have attempted to keep the size to a minimum. In keeping with this philosophy, I have given the DSR1NK routine a "short back and sides" that would make the toughest drill sergeant smile.

Source code listed in Bug-Bytes, June 1989, starting on page 9.

Making your own Dictionaries

by Geoff Trott

This month I will only have time to try and explain the program which appeared in the last TND. Next month (with luck) I will give the source of the string procedures used in the program and explain how I have used them. I am glad that Craig has started a series on c99 as he is explaining it much better than I will so I commend those articles to you if you cannot follow what I am saying. Perhaps between the two of us you might see some light!

The first two statements are "#include" statements. These are special statements as they do not need a ";" at the end. They tell the compiler where to go to get a source file, like the ".IF" in TI-Writer or "COPY" in Assembler. The first of these is part of every C language and defines the defaults for the standard I/O library routines. The second file contains the definitions of the string procedures which I have compiled and made into a library. The next lines down to "main()" define variables which become global variables and so available to all procedures used in this program. More than one variable of a particular type can be declared in each statement and they may also be intialized to particular values with an "=" operation. There are a number of integer variables declared but the most interesting variables are the character variables. "c" is a single character variable, and "*p3" indicates that "p3" is a pointer to a character. I shall return to pointers in a moment but it can be considered as a variable which stores the address of a variable. All the rest are single dimension character arrays of various sizes, some of them initialized. For example, "at[2]="@"" defines a two character array which is initialized to the string "@". The string "@" is not quite the same as the character '@', as a string is always terminated by the null character ('\000') and so will have two characters '@' and '\000' rather than just '@'. Note the use of the single and double quotes. A back slash followed by a number is a way of specifying a non printing character using the octal number system. The number can have a maximum of three digits with the first one of three having a maximum value of 3 with the others having a maximum value of 7 (octal number system). So "ffcr" will contain the 3 characters: form feed, carriage return, null. "al" contains the 26 capital letters followed by a null, "fname" contains the name of a file (DSK2.) followed by a null and "nc" contains just a null.

That took a long time and we have not yet entered the program! However you need to understand the variables that will be used in the program and how to set up your own. The main program starts with "main() {" and ends with the matching "}" just before the procedure "putwd()". Notice that the "{" is actually on the next line but it does not have to be. C is free format so you can put many statements on one line or one statement on many lines, as I did with the declarations. It is good practice to try and set programs out neatly so they are easy to follow using indenting and by in general only putting one statement on a line. I have not done this exactly, but hopefully only where it makes sense to deviate from this ideal. The main program starts with some more variable declarations which now become local to the main program and will not be available to any procedures. Some integers and pointers to character variables are defined. Then the processing begins.

The first thing that happens is that a file is opened for writing the output and the single character "*" is written to it using "fwrite", a function in the standard library. Then the sector byte counter and the line byte counter are initialized. The line byte counter determines when there are enough characters to almost fill the D/V 80 record while the sector byte counter is used to determine when a sector is full. "puts" outputs a message to the screen and the "\n" character moves the cursor to the start of the next line on the screen. The program then enters a "for" loop, to read in 26 files, one for each letter of the alphabet, and process them to the output file. This loop does most of the processing.

The first thing that happens is that the current letter of the alphabet is concatenated onto the end on the filename in "fname" and this name is shown on the screen as well as used to open a file. Then the current word "wore" is initialized to be the letter of the alphabet, a pointer is made equal to the address of "sl", which holds the record to be written in the output file. The first record of the file is then read into "buff", and the number of characters in "buff" determined. "sl" is loaded with the first character of "buff" followed by "*" and this is output. Then follows the house keeping for "sect" to keep track of how many bytes are left in the current sector. "fwrite" outputs the number of characters specified, whereas if "buff" has only one byte in it there are no words for this letter of the alphabet and a null byte is written using "fputs". This is done using the "if" clause which involves 3 statements inside the "{ }" if the "if" clause is true. Note the double "==" in the test as the single "=" is used for assignment of values. If there are words in the input file starting with this letter of the alphabet, they are processed by the "while" loop which ends just before the "fclose" statement, which closes the input file. Following that the end of letter record is written ("@") and the next value of the "for" loop is started. When the "for" loop is finished, two records are written consisting of "formfeed" and "carriage return" characters. Finally the output file is closed.

All the work of getting the next word is done by the code in the while loop while the work of outputting the words or parts of words is done by the procedure "putwd". Getting the words is complicated by the format I have used when adding and changing words in the letter files. Writing words is similarly complicated by the format needed by the dictionary files. To get a word from the input buffer the program looks for a space which defines the end of the first word in the line. Subsequent spaces define the additions to the end of that word. Each word is sent to "putwd" as it is isolated. After one record is handled the next one is read. If it is not empty the number of characters is determined and the "while" loop continued. If there are no more records in the file the length is set to zero, the output buffer is written and variables are set up for the next file.

Procedure "putwd" has no parameters passed to it as it uses the global variables defined in the first few lines. Some local variables are defined, including some pointers, and then the first statement is an unusual "for" loop which executes a blank statement a number of times. As can be seen, there are multiple statements in the "for" clause. The statements are separated by commas and those before the first semicolon are executed first and only once. These initialize the pointers "p1" and "p2" to be the addresses of the buffers "wore" (containing the last word) and "word" (containing the next word) respectively and sets the counter "n" to zero. Between the two semicolons is the test for continuation of the loop which in this case is that the character pointed to by "p1" is the same as the character pointed to by "p2". After the second semicolon are the statements executed at the end of each loop, in this case both pointers are incremented and the counter is incremented. So this complex statement finishes when the two words differ from each other with the pointers at the characters which are different and "n" with the number of characters which are the same. Then there is a diagnostic check on the input to make sure that the words are in alphabetical order and output of a message if they are not. Then a null is stored at the next character in "sl" using the pointer "p3" just in case the number of characters to be written would make the record larger than 80 characters or require more than the 256 characters in a sector and then the current record would be written. continued on page 35

From the Bulletin Board

MAIL TO : ALL
MAIL FROM : TI-ARTIST

This is a new file in the news menu to help people with problems that they might be having with different graphics programs they might use. Also from time to time I will put items of interest concerning graphics programs etc. Also how to animate your picture files. An interest group has been set up and is held after each TIsHUG meeting to give demonstrations on the latest graphics programs from America. If you need help leave a message for me TI-ARTIST and I will try and figure it out.

TI-ARTIST.

MAIL TO : ALL
MAIL FROM : LIVERPOOL

One error in SPLIT/MKII. Should read
150 L=L+1 :: LINPUT #1:A$ :: B$(L)=A$

Bye for now LARRY

MAIL TO : ALL
MAIL FROM : RLE#1

I would just like to inform everyone that RLE#1 is a new sub-editor. In this file, I have put a picture of Bugs Bunny. To look at it, you need to capture it (ie log it to disk) and then go into TI-Writer and edit out the first line ie uploaded by..... Then re-save it and it will load through Max RLE loader. If you do not have this loader, I will upload it for you in the next few days.

By the way, you cannot view the RLE#1 file with TE2.

Robert Brown (a person on many talents)

MAIL TO : ALL
MAIL FROM : SHOP

For Sale For Sale For Sale
======== ======== ========

Urgent sale required. Dick Smith Dataphone II 300 baud modem, extra long cable to connect to phone. Must sell to get a 1200 baud modem to assist with work for the WIA (Wireless Institue of Aust) VK2 (NSW) Division. A steal at only $50.00

Please leave message on this BBS or call me at home on (02)608 3564 after 7:00 pm weeknights till 10:30 pm or all day on the weekends if you can get me

Thanks to you all: Steven Carr TIsHUG Shop Manager

MAIL TO : ALL
MAIL FROM : ROSCO

I am thinking of selling my system as I may need to downgrade to IBM. I am now a full time Tech Teacher and need an IBM compatible. My system consists of:

2 computers (1 silver 1 grey), PE box, 2 Disk drives, 720K RAM, 32K, RS232, Multiplan, Disk Manager 2, Editor Assembler, Extended BASIC, Plato, 200 plus disks of software, a modem (300 baud) and heaps of documentation, magazines etc ---- Offers please --- Phone me on (02)637 6772 at home

---

MAIL TO : ALL
MAIL FROM : SHERLOCK

For Sale .....

TI 32K Card 50.00
TE2 module plus book 30.00
Extended BASIC module plus book 30.00
TI-Writer module plus book 30.00
Editor Assembler module plus book 25.00
Technico 9900 development board including EPROM Programmer RS232C interface plus documentation

---

For Sale

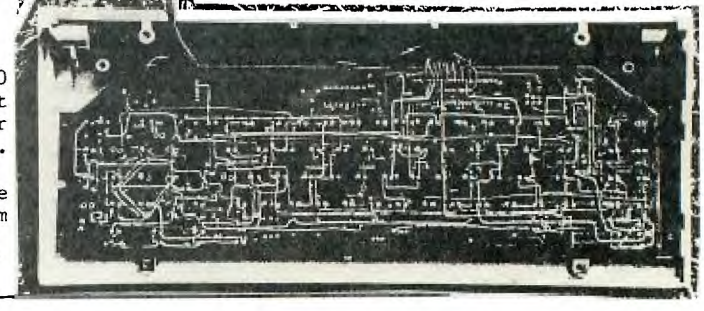
TI99/4A silver console with UHF modulator and inbuilt 32K; PE box and interface; 184K RAMdisk; TI RS232 card; TI disk controller with Disk Manager 2; Double sided disk drive; Speech Synthesizer; Cassette tape player; 2 joysticks; Naverone cartridge expander.

12 modules with books including Extended BASIC, MiniMemory, TE2, Multiplan; approximately 40 disks and 20 tapes of software

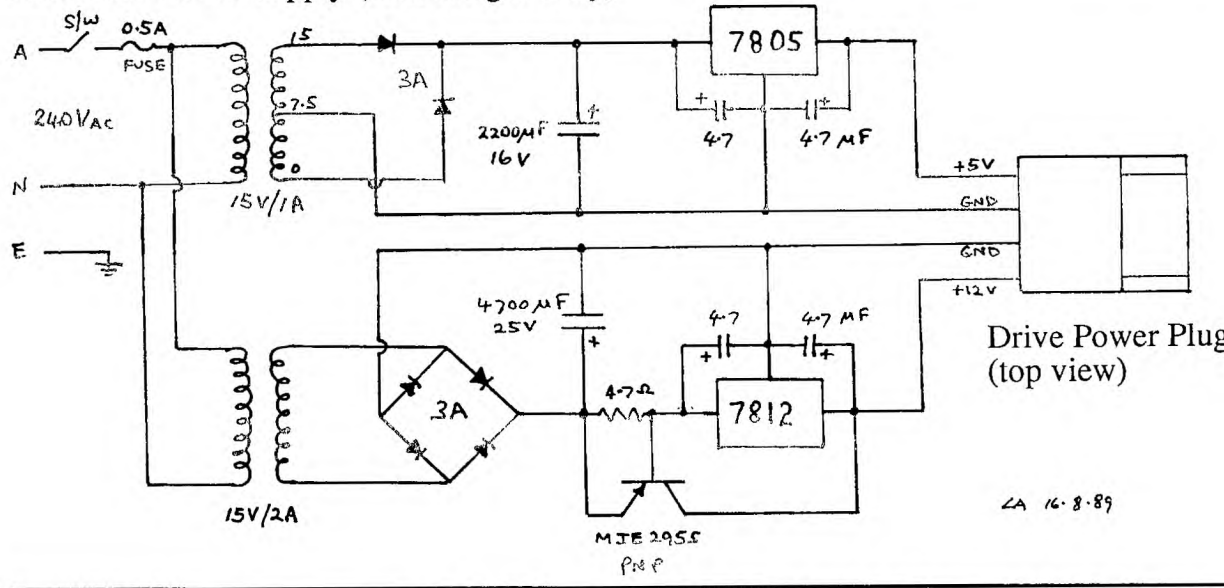
All club magazines from February 1983 and many other informative books.

All this and more for $1000 or nearest offer. Phone Alistair on (02)745 3543.

Photograph of Modified IBM Keyboard



Hard Disk Power Supply (Half Height Only)



Transfer Programs between Cassette and Disk

by Eric-Paul Rebel, Netherlands

Normally OLD DSK1.PROGRAM; SAVE CS1 will work. But what when the file is too large and even CALL FILES(1) will not help?

After loading an INT/VAR 254 from disk you can type in:

CALL LOAD(-31888,63,255).

This will act like CALL FILES(0) and sometimes the program can be saved. If not, it is really too big too fit on cassette.

But how can you get a file from cassette on your disk that will not load because it is too big?

Simply by using the DSKBUF program. How? Type in:

```
RUN "DSKx.DSKBUF/EXB"   (only ones)
CALL LINK("SAVE")      (this will save the disk buffer)
OLD CS1                  (the program has to load now)
CALL LINK("LOAD")    (this will restore the disk buffer)
SAVE DSKx.PROGRAM         (or any name you like)
```

```
        DEF  SAVE
        DEF  LOAD

VMBW    EQU  >2024
VMBR    EQU  >202C

SAVE    MOV  @VDPTOP,R0
        MOV  R0,@SAVTOP
        LI   R1,DSKBUF
        LI   R2,>4000
        S    R0,R2
        BLWP @VMBR
        LI   R0,>3FFF
        MOV  R0,@VDPTOP
        RT

LOAD    MOV  @SAVTOP,R0
        MOV  R0,@VDPTOP
        LI   R1,DSKBUF
        LI   R2,>4000
        S    R0,R2
        BLWP @VMBW
        RT

SAVTOP BSS 2

DSKBUF END
```

Eric-Paul Rebel
Merelstraat 27
1223 NR HILVERSUM
The Netherlands

TI99/4A News

by Chris Bobbitt, from Asgard News

Asgard release Page Pro 99 at US$24.95 plus US$4.75 Airmail delivery.

Page Pro 99 composes a 66 line page full of text, graphics and lines. There are no formatters, no cryptic commands or functions, just a "what-you-is-what-you-get" screen, and the ability to paste in pictures and type text.

It will allow up to 28 pictures of any size or shape anywhere on screen. It will let you type text in a complete large font and a small font of your choice (both with upper, lower case, numbers and symbols) and draw lines anywhere you need them. It will create almost anything (maps, newsletters, etc). Has three densities, from "rough draft" up to reproduction quality.

It will type in any direction (up, down, etc), can load on pictures and get rid of them at your option, can read in a text file and paste it into the page, save it as a text file, load in new fonts etc.

It comes with a collection of artwork and fonts, as well as utilities to convert TI-Artist fonts and instances into Page Pro 99 format, as well as make 2 column justified text for use in Page Pro 99. Arrived 6/6/1989.

Genial Macintosh to TI99/4A program

Genial Computerware is reportedly working on a program that will allow you to move Macintosh pictures to a TI99/4A. They placed a sample of the program up on US BBS networks recently

---

First-Base words

First Base is reportedly entering the beta-testing stage of development.

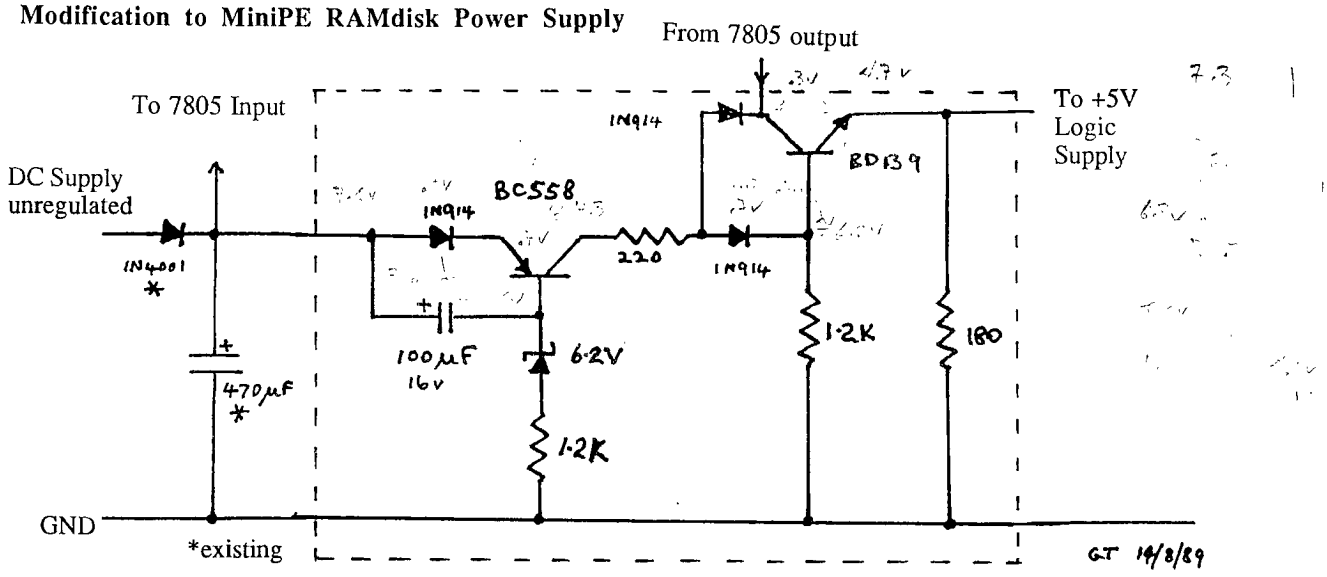
GIF for the TI99/4A

GIF graphics transfer protocol for the TI99/4A is in development for converting MAX/RLE from black and white to colour.

Computer Fix Aust has BIZTEL smart modems on sale that have Auto dial, Auto Answer, Auto Baud protocols (2400, 1200, 300). Price only $299. Will work with Telco, Mass Transfer, Fast Term, etc.

Note for Telco users, Telco Auto dial will change modem to Baud rate that the Auto dial is set for. So you can set four Auto dial programs to dial the BBS at 300, 1200, 1275, 2400 rate. Just call up auto dial and press 1, 2, 3 or 4 and it will do the rest.

Modification to MiniPE RAMdisk Power Supply



Forth to you too! Session 1

Author unknown

Introduction

According to our source there are quite a few people out there who got the TI-Forth disk and documentation when TI made them available to user groups. But not very many do much with it. Why? Well, the TI manual is not a tutorial, it assumes that you know something about Forth. Though packed with useful information there are no "HOW TO" instructions for the beginner. We will try to get you started from the very beginning. Hopefully we will strike a happy medium, somewhere in between teaching and providing information that is useful to you.

What is Forth?

There was much hype when it was made available, some of it was much TIL (Threaded Interpretive Language) and it will be hard for you to believe that there is no GOTO command. If that is hard to swallow, there is more. It uses RPN or post-fix notation (RPN = Reverse Polish Notation). In other words, it is not 2 + 2 that equals 4 but 2 2 + .

We will find out more as we go along, for now let us just say that Forth is very powerful, quite a bit faster than BASIC, compact, but perhaps more difficult to learn than BASIC. As a matter of fact, knowing BASIC may make it harder on you, because you will be thinking BASIC until you get the hang of Forth.

Getting started.

Before you do anything with your Forth disk, get out a Disk Manager and make a backup copy. Do all your work and experimenting with this copy unless you are prepared to get a new Forth disk. Now plug in the Editor Assembler, opt for 3 (LOAD and RUN) and enter DSK1.FORTH. After a moment the screen shows "BOOTING..." which is soon replaced by a menu. These are the LOAD options. For right now you need to concern yourself with only 2 of them: the normal or the 64 column editors. Your choice will depend on several factors: 1) your eyesight, 2) your monitor, and 3) how well you have adapted to using 'windows'. So jump right in and enter -64SUPPORT. After your disk drive is through you will see a tiny 'ok', meaning the 64 column editor has been booted. To see what your screen will look like type 34 EDIT <enter>. If you can read what is displayed on your screen, you will want to stay with -64SUPPORT. If it is hard on your eyes, settle for the 40 column editor. To get an idea what it looks like, hit FCTN[9](ESCAPE), then enter TEXT COLD. Forth will re-boot and when it is done, enter -EDITOR. (From now on, 'enter' will mean to type in the word followed by the ENTER key.) Again enter 34 EDIT to see what your 40 column editor looks like.

Programming in Forth consists of editing SCREENS, such as that number 34 screen you called up for editing. But we are not ready for that, yet. Hit ESCAPE (FCTN[9]) and enter FLUSH and do this: Make yourself an overlay strip so you can edit easily. Keys and their functions are explained on page 5, Chapter 3, of the TI-Forth manual. Now here is another thing you might want to find out right now: a display colour that suits you. Since you are still in the so-called 'interactive' mode of Forth (no program is running) you can enter this ditty :

```
: SEE 252 22 DO I DUP . 7 VWTR KEY 2 = IF ABORT ENDIF
    LOOP ;
```

After you get the 'ok', type SEE.

Do not worry if you cannot read anything, at times the foreground and back ground colours match and there is nothing to be read, keep hitting any key and the colours will change. When you see a combination which gives you a good screen display, write down the last number (bottom of the screen) and continue to step through the loop (or exit via FCTN[4]).

You have accomplished 2 things:

1) you know which editor you will want to use
2) you have chosen a screen colour. o

Conversion Tables

by D.N. Harris

This short program is for the user of a second hand motor car with the speedometer calibrated in miles rather than kilometers. Changing the range in line 110 will give various ranges, and I have arranged for the miles to be rounded up if over .5 kilometres.

```
1 REM D.N.HARRIS
2 REM 5TH AUGUST 1989
3 REM YES I DID GET AN URGE TO PROGRAM
4 REM JUST WHEN I WAS ABOUT TO GO TO
5 REM THE MEETING!
100 DEF KILO(MILE)=MILE*1760*3*12*2.54/100000+.5
101 OPEN #1:"PIO"
102 PRINT #1:"MILES";TAB(40);"KILOMETRES-TO NEAREST
    WHOLE ROUNDED UP"
103 PRINT #1:RPT$("_",80)
110 FOR MILE=0 TO 120 STEP 2
118 MILE$=STR$(MILE)
119 KILO$=STR$(INT(KILO(MILE)))
120 PRINT #1:TAB(4-LEN(MILE$)); MILE$;
    "                                   ";
    TAB(40-LEN(KILO$)); KILO$
130 NEXT MILE
```

This program prints quite reasonably aligned tables. It also shows the way DEF can be used without confusing reference to the term "function", so for example in compiling a table of temperatures one could DEF FAHR(CENT)=9*CENT/5+32.5, the .5 being added so as to round up. If one wanted to round down one would make the constant 31.5, or for truncation one leaves it at 32. In this case one could look at the whole group of numbers resulting if one got rid of the string handling functions, or if one wanted to round to 2 decimals the constant becomes 32.005 and it is necessary to multiply by 100 before taking the integral value, then one can divide by 100, then one can convert to strings, and allow a TAB value for a LEN of up to 5 including the decimal point. Some people would say a swear word and print ragged tables, but having struggled through a Tech course in Computing, I like to get the output looking professional and the documentation *clear and* possible even for me to understand in the wrong mood when tired! A few lines of program will in this case print a foolscap page of output. When planning to print tables it is a good idea to think of the range that will be printed, as foolscap allows about 66 lines, so 1 to 60 step 1 or 0 to 120 step 2 will fit a foolscap page. If the printer starts to put ink all over the platen maybe you made a wrong guess about the interval! Printing tables justifies a computer, but if all you want is to play games or handle documents, a games machine or a dedicated word processor would do the job better. o

continued from page 34

TI Focus, June '89: PRESS to have 80 to 90 revisions on original concept and will be available in about August; Review of Page Pro 99 which supports graphics and text in columnar form; use of REM in BASIC; report on the Lima Multi Users Show; review of Dijit 80 column card and a BASIC "Magic Nines" program .

TopIcs, LA 99ers, June '89: TI-Sort (for TI-Base) can handle 99,999 records; Art Green has released V4.2 of TI-Writer clone; Inscebot are working on a hard disk version of TI-Base and TI-Artist Plus; hard disk user tips; historical information on the TI99/4A computer; using "RUN" within Extended BASIC programs; Two Way Print utility; controlling spaces after the period in TI-Writer by using the " " character; Genial Traveller Diskazines overview and Beginning Forth #13. o

continued from page 1

John Hagart also asked if anyone has a Logo module for sale and Logo programs on disk. He is helping out at his local primary school, supervising children doing lessons on the school's computers. He would like to be able to practice at home on his TI99/4A. TI-Logo is quite a good version of Logo and I have found the manual that comes with it to be quite good as a guide to using Logo. There are a set of programs called the Logo Curriculum Guide which come with a manual and several disks of programs designed for the young with lots of graphics and other features. I am sure that some of the things that can be done with TI-Logo would be very difficult to do on other machines. Can anyone help John? If you can, write to him at 8 Griffin St., Gordonvale, Nth Qld 4865; phone (070)56 1196.

****

More from the continuing saga of the hard disk drive that has corrupt sectors. I received a disk from Garry Christensen of Brisbane (thanks Garry) a few weeks ago with all the articles which are in this issue and which appeared in Bug-Bytes first, along with the program Hard Master by his father Colin. By the way, I hope Colin is better now and recovering from his heart attack. Back to the subject at hand. Hard master allowed me to make the directories on the hard disk whole again and even allowed them to be listed out so that I now know the names of all the files on the disk. Unfortunately one of the corrupt sectors is used by the bit map and so Myarc Disk Manager 5 still will not run on that disk. This means that there is no easy way to copy files off the hard disk as there is no other program which will allow files to be copied. I could copy all files except program files by writing a program to treat them as data files, but this would be a lengthy process. Program files have me beaten at the moment. If they are BASIC then they could be OLDed and then SAVEd but if they are anything else there is no utility which I know of which will do it easily. Does anyone have any ideas? Meanwhile John Vandermey has his controller card back at last and has kindly left his disk with me for more time to see if I can get any more from it.

****

The editor's system is now virtually complete with the purchase of a monitor and RS232 card. The system consists of console, Mechatronics 80 column card and its power supply, Commodore 1084S video monitor which plugs directly into the 80 column card (or Geneve), PE box with 32K byte memory expansion, TI-RS232 card, 256K bytes Printer Buffer, Brother HR-15 daisy-wheel printer, Myarc Hard and Floppy Disk Controller card, 20M byte hard disk, double sided double density floppy drive (courtesy John Paine) and power supply for hard disk. I am very pleased with the monitor which gives a very clear picture on 80 columns and did not require and extra circuitry or even modified cabling. The monitor came with two cables, one for the Amiga and one for a PC. The PC cable was perfect for the Mechatronics card. If someone were looking for a dual purpose monitor this one would seem to provide all the facilities required. With all this great equipment at the editor's disposal I am hoping that someone will volunteer to take on the job next year to give me some time to do other things. I can always hope I suppose!

****

Lou is always asking me difficult questions about hardware and good ways for members to expand their systems. He has done an excellent job of putting his 2 way expansion system into a neat enclosure which he will be showing at the meetings. The difficult questions relate to what would be best to choose as the two cards to install in the expansion slots, given that there are only 2 slots and they take PE box cards. The ideal mix would have 32K memory (if not already in the console), a disk drive, a printer port (RS232 or PIO), a RAMdisk and perhaps a serial port for a modem. If that could be done in two cards then you would have a rather neat and powerful system. The obvious way to go would be an AT Multifunction card and a RAMdisk card. The not so obvious way would be to use a TI disk controller card (more readily available?) with a RAMdisk card and 32K memory inside the console or attached to the RAMdisk card and a simple PIO port also attached to the RAMdisk card. What do you think? If you like these ideas or wish to inject some other ideas, talk or write to Lou so he can gauge which way most people would like to go and if you get the chance, have a look at his (Mark 1) enclosure. I am sure that you will be impressed at what he has done. o

Newsletter Update

by Lou Amadio

Local Newsletters

ATICC, July '89: Old committee re-elected unopposed; seven part article on "The Making of a Portable" by Jan Janowski; "Sprites in G", an article on how to program in advanced G; Extended BASIC Sprites; Redefining the cursor; Multi-Keyboard Scanning; an RGB monitor kit from Colin Cartwright and two pages of useful function key overlay strips.

Melbourne Times, April '89: Glossary of computer acronyms; Geneve Notes, autoexec and menu files; Peeping into Pascal by Peter Gleed; The Power of AND (part 4) by Craig Miller; Reminder of TI Fair, October 14th; Multiplan tutorial, Costing for Profit by Peter Gleed.

TIUP, July '89: BASIC program to produce wrapping paper; general information on the Zork adventures; beginners notes on handling cassette players and designing graphics; using the Editor Assembler cartridge; configuring Funnelweb and how to wire up a pair of joysticks.

Bug Bytes, June '89: Reminder of the program competition which has been organized as part of the Melbourne Faire; The Hard Disk, a Guided Tour; using a mouse with MyArt; documentation on Disk-Edit by Col Christensen; Sound and Assembly and Direct Keyboard Access. Bug Bytes, July '89: Two new fairware programs from the Christensens: one to back up a hard disk on to itself and the other to facilitate the transfer of programs from disk to cassette; article on Sound in Assembly; Cassette Tips and Hints and the Funnelweb Report by Tony McGovern.

Overseas Newsletters

TIdbits (Mid South 99ers), July '89: New EPROM for the Mechatronics 80 column card from Barry Boone fixes some bugs; new commands for My-Word; upgrade for 99 FORTRAN; cSHELL99 (written in C) provides a GEOS like shell for the TI99/4A; Sign Shop (similar to Print Shop) by Peter Hodie; future speculation on the second anniversary of the introduction of the Geneve includes a 20 MHz version with a minimum of 2 Mb RAM; Z80 card for the PE Box; Texaments to supply a high speed sort utility for TI-Base owners; Geneve Notes by B Miller.

CIM 99, July '89: A lot of useful information written in French!

ROM, June '89: BASIC file handling; BASIC program to check the linearity and convergence of monitors and TVs; assembly language Boot Tracking; new high resolution 80 column graphics program from Germany; RAGTIW word processing program with new editor and formatter commands.

Northern NJ 99ers, July '89: Telco Terminal Emulator V2.1 reviewed; upgrade for TI disk controller recognizes a fourth drive and head step times; upgrade for TI RS232 adds TP and SIO (for RS232 defaults); review of text scanner and use of special BASIC commands resident in the PRK module.

Spirit of 99, July '89: Problems in keeping new members; review of Page Pro 99 for combining text and pictures; review of Jiffy Card to produce quality greeting cards; TI-Writer V4.2 has a number of useful improvements over the original version; Legends II to be released soon and several TI-Base Tutorials.

continued on page 33

Regional Group Reports

Meeting summary.

Banana Coast	10/09/89	Sawtell
Carlingford	20/09/89	Carlingford
Central Coast	9/09/89	Toukley
Glebe	7/09/89	Glebe
Illawarra	18/09/89	Keiraville
Liverpool	8/09/89	
Northern Suburbs	28/09/89	
Sutherland	15/09/89	Jannali

BANANA COAST Regional Group
(Coffs Harbour area)

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

CARLINGFORD Regional Group.

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

CENTRAL COAST Regional Group.

Regular meetings are normally held on the second Sat rday of each month, 6.30pm at the Toukley Tennis Club hall, Header St, Toukley. Contact Russell Welham (043)92 4000

GLEBE Regional Group.

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

ILLAWARRA Regional Group.

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

LIVERPOOL Regional Group

Regular meeting date is the Friday following the TIsHUG Sydney meeting at 7.30 pm. Contact Larry Saunders (02)644 7377 (home) or (02)708 5916 (work) for more information.

At next Liverpool meeting will be using the Page Pro 99 program. This program is the most outstanding Desk Top Publisher ever made for the TI99/4A computer. Any one into TI-Artist or Word Processing be prepared to have your socks blowing to the noon. (It is fantastic.) Press is due any time now. Legends II (expanded to a third disk which slowed the release) will be here any time now.

NORTHERN SUBURBS Regional Group.

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

Come and join in our fun. Dick Warburton.

SUTHERLAND Regional Group.

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

TIsHUG in Sydney

Monthly meetings start promptly at 2pm (except for full day tutorials) on the first Saturday of the month that is not part of a long weekend. They are held at the Woodstock Community Centre, Church street, Burwood. Regular items include news from the directors, the publications library, the shop, and demonstrations of monthly software.

Unfortunately, a couple of the planned activities for the August meeting had to cancelled due to problems at the venue. However these will scheduled for later months. Many apologies for any members inconvenienced because of this. Meetings for the next two months are:

September 2 - Ross Mudie will demonstrate how to exchange programs an mail between two computers via modems. First a TI99/4A will be connected to a TI99/4A, followed by a TI99/4A to IBM connection. The first part of series of lectures on using TI-Writer's transliterates will be presented. In addition, the TI-Artist SIG will meet.

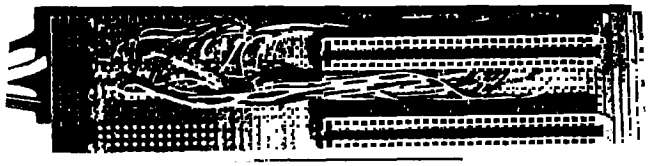
October 7 - What have the various special interest groups achieved over the last six months? When do they meet? How can you become involved? All of these questions will be answered with presentations from each of the SIGs. The TI-Artist SIG will meet and the final lecture in the transliterate series will be presented. The remaining meetings for 1989 will be held on November 4 (Full day workshop) and December 2.

Craig Sheehan (Meeting coordinator). o

For Sale

One only 512K byte RAMdisk for a MiniPE system including clock for $300. Phone Les Andrews on (02)319 2572

Photograph of 2-way interface connectors



continued from page 30

This is determined by the number of characters left in "word" after differing from "wore" with 1 added for the length byte. If the record is written, the variables and pointers are reset. Then the count of characters which were the same (plus 128) is stored in the output string using the pointer "p3" and the remainder of "word" is copied to the output string. The variables and pointers are adjusted accordingly and then the contents of "word" are copied into "wore".

Note that there are two lines commented out within "/* */" which are debugging lines I have left in. These are ignored by the compiler.

If you are confused about pointers then you are not alone. They are very useful but require a good understanding to not make mistakes. An asterisk in front of a pointer means the value of the variable to which the pointer is pointing. An ampersand in front of a variable means the pointer to that variable. All array names are in fact pointers and if you want to return a value from a procedure to the main program (other than a single integer) you must do it using a pointer. Pointers are associated with variable types so that when arithmetic is performed it is done according to the data type. For example pointers to character variables are incremented by 1 while pointers to integers are incremented by 2 since characters occupy only one byte while integers occupy two bytes of memory. C also allows pointers to be incremented before or after their values are used. It is a difficult concept to explain in a short space but I shall continue with more about pointers next month (or perhaps the one after) when I return to look at the string functions. o