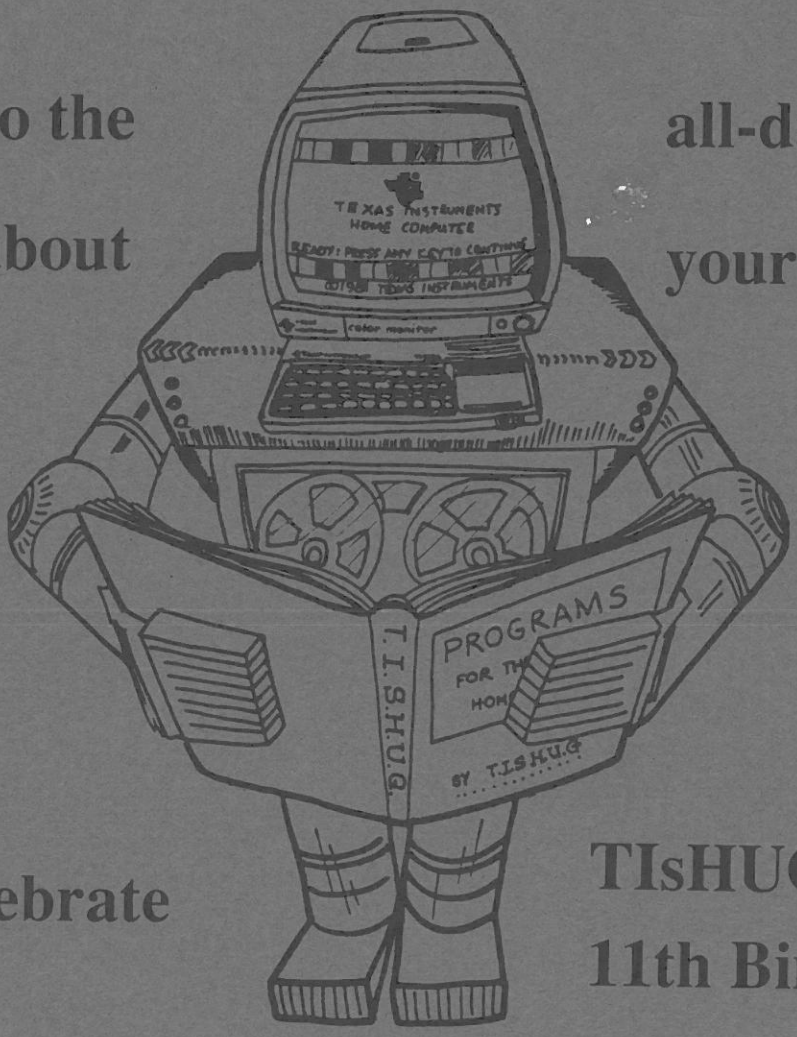
TIsHUG NEWS DIGEST

Focusing on the TI99/4A Home Computer

Volume 11, Number 4 May, 1992

Registered by Australia Post - Publication No. NBH5933

Come to the all-day tutorial to learn about your computer.



Help celebrate TIsHUG's 11th Birthday.

Sydney, New South Wales, Australia $3

We have changed our postal address. From now on please use:
PO Box 1089, Strawberry Hills NSW 2012.

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

TIsHUG News Digest

May 1992

All correspondence to:

P.O. Box 1089
Strawberry Hills, NSW 2012
Australia

The Board

Co-ordinator
Dick Warburton (02) 918 8132
Secretary
Terry Phillips (02) 797 6313
Treasurer
Geoff Trott (042) 29 6629
Directors
Rolf Schreiber (042) 85 5519
Russell Welham (043) 92 4000

Sub-committees

News Digest Editor
Bob Relyea (046) 57 1253
BBS Sysop
Ross Mudie (02) 456 2122
BBS telephone number (02) 456 4606
Merchandising
Percy Harrison (02) 808 3181
Publications Library
Russell Welham (043) 92 4000
Software library
Rolf Schreiber (042) 85 5519
Technical co-ordinator
Geoff Trott (042) 29 6629
TI-Faire co-ordinator
Dick Warburton (02) 918 8132

Regional Group Contacts

Central Coast
Russell Welham (043) 92 4000
Coffs Harbour
Kevin Cox (066) 53 2649
Glebe
Mike Slattery (02) 692 8162
Hunter Valley
Geoff Phillips (049) 42 8176
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	$35.00
Associate membership	$10.00
Overseas Airmail Dues	A$65.00
Overseas Surface Mail Dues	A$50.00

TIsHUG Sydney Meeting

The May Meeting will start at 9.30 am on 2nd of May at Ryde Infant School, Tucker Street, Ryde with a BBQ lunch if required. The Assembly Class will not run this month but will resume in June.

Printed by
The University of Wollongong
Printery Services.

TIsHUG News Digest ISSN 0819-1984

Index

Thank you to Tony Bell for his re-typing of articles which appear in this month's TND

All articles appearing in this month's issue of the TND are available as text files on disk ready for the formatter. Newsletter editors please note that, if you wish to re-print any articles, contact us, stating which articles you are interested in and giving the date of the TND. These will be dispatched to you promptly at the cost of the media plus postage.

Editor's Comment

by Bob Relyea

Just a reminder that if you put a file on the BBS for me to edit for the magazine then it must be addressed to the EDITOR. I do not have the time to go through all the other files looking for something (let alone the Telecom cost!) and Ross gives me the files on a disk every two months anyways. So, if you definitely want it in the TND and reasonably quick then it has to be addressed to the EDITOR.

I am looking forward to the all day tutorial this Saturday as, for the first time in ages, I have not been asked to do anything (barring any last minute phone calls, of course!). I can sit back and relax and learn a few things to take home and practice with. See you there. o

Co-ordinator's Report

by Dick Warburton

The TI-Faire seems to be coming together quite well. Our committee has been busy tying up all the loose ends. The venue has been secured and the basic format worked out. We are aiming to provide a climate where everyone who visits can meet and mix with everyone else. Name tags for everyone have been purchased, and we are looking closely at the social side of the TI-Faire. Fortunately the Western Suburb's Leagues Club is next door and they have been more than helpful. Meals will be cheap, showers will be available and they have promised to provide us with tables. We have been impressed with their courtesy and help.

We are approaching different club members for their help at the TI-Faire, in the type of activities listed in the inserted page in the last TND. We are trying to show as many facets of our club activities as possible, particularly the range of hardware contributions made over the years. We intend to also display the older TI99/4A gear if we can get it. If you can help us out with any exotic TI99/4A hardware, please do. We will take good care of it at the TI-Faire. We hope to have up to 20 systems running at any one time, if the power supply will take it. The range of demonstrations could be tremendous with your help. We will have games machines up and running, hopefully showing the versatile range of games available. We will display educational software, word processing, Data bases etc. We need volunteers to take shifts. Please put your name down with Ian Mullins (phone 8711514) as soon as possible. We also need to know what systems are available for the weekend. Any type of system will do. We want to show the evolution to the present day. I hope the 80 column TIM cards arrive before the TI-Faire, the ones ordered in October 1992. We can only pray. We have some surprises in store. To achieve what we want, we need some software written specially for the occasion. I will be approaching a few people to see if we can give this faire a real shake. If you have some ideas which have not yet been discussed, please see anyone on the TI-Faire committee and tell him. One of our original contributions will be the display and demonstration of our EPROM RAMdisks. If you are still having any difficulty with your's, please bring it to the tutorial day, where we will work together on them to get them going well. The EPROM section of my RAMdisk is invaluable and very reliable. I cannot say the same about one of my RAMdisks. It drove me mad trying to find why it simply would not work. I finally took the trouble to carefully examine all the address lines around the 74LS154 chip and surprise, after a little resoldering it works really well. I am hoping to increase the EPROMs on my RAMdisk as soon as possible.

Another two issues relating to the TI-Faire involve firstly approaching sponsors to help us out and to display their goods. If you know anyone who might be interested, please see one of the committee. Secondly, if you can help us out by offering to billet an out of town visitor, we would be grateful. This would obviously involve transporting our visitor about, but it is a great way to get to know overseas or interstate TI99/4A users. If you can help in any way let us know.

Well, I will see you at the next meeting; Dick Warburton. O

Treasurer's Report

by Geoff Trott

Have you renewed your subscription? If you have not, this will be the last TND you will receive. Look on the mailing label to find out when your subscription expires.

Income for March	$1680.10
Payment for March	$2147.11
Excess of expenses over income for March	$467.01

Genealogical Data Base

an Introduction by Jim Grimmond, HV99ers

This program is not intended to teach you how to go about researching your ancestors. It comes into operation after you have started to gather the information on your families and wish to put it on file. The program has been designed to create and update a genealogy file.

So you are saying "not another boring, complicated genealogy program?" No! The whole idea of this program is to get away from those complaints, make it quick internally,(ASSEMBLY) and shove in some bits and pieces not normally found in these packages. The end result is a genealogist's delight.

When you get into the program you will be amazed at how quick it is. Not only in entering data, but in retrieving it and printing it out. (IBM Lookout!).
Some of the main features are:

1. A cross reference system so you can have paternal and maternal family lines in the one (1) file (TREE).

2. In the review mode, you will be able to search for a person. When they are found, a split screen will enable you to toggle through the personal details and, by using the <FCTN/NUMBER> Keys, find that person's father, mother, children and print out reports.

3. Search by name. When the name entered is found the birth date for that name is shown on screen, if this person is the one you are looking for you can then advance into REVIEW mode for that person. If the person found is not the one you are looking for then the search can be continued.

4. Illness. This feature is a most unusual feature to find in a genealogy programme. In the documentation you will find further reference material mentioned on this little used but vital side to genealogical research. This section can be ignored if you do not have the need for it.

5. Pedigree Chart. This chart is printed on a quarto/foolscap size page and contains four (4) generations. That is the person, the person's parents, grand parents and great-grand parents. It will also show their names, birth, death dates, birth place, family line codes and their illness codes.

6. Detail Report. The detail report is best utilised by saving it to disk instead of dumping to your printer. The detail report after saving to disk can be loaded into Funnelweb and 20/30 lines of text added relating to the person on the report. The serial number printed on the detail report is intended to be used as a page number in a book which would comprise all the detail reports from your family tree. The completed book would make a nice present for your relatives.

Now, if you are a seasoned genealogist, you may wish to get started into the data base. If so, may we suggest you pull up the text file "GENE-DOCS" and print a copy. Read them through fully before you attempt to start. It is always better to have a little more knowledge than not enough.

If you are a beginner in the field of genealogy, may we suggest you pull up the text file "BEG/GENE". This text should help you get started in your research.

There are also two other text files:-

1. ABBREV/1, has quite a comprehensive list of county and country abbreviations.

2. APPEND-1,is the "Appendix of Illnesses" you will require to enter "Illness Codes" in the data base. O

Secretary's Notebook

by Terry Phillips

First this month an important notice to members. A Special General Meeting will be held on June 6 (normal meeting day) to consider and vote on an amendment to Paragraph 4 of the Club's Memorandum. Full details of the proposed amendment are given below.

The Directors of TIsHUG (Australia) Limited at their meeting on 4th April 1992 have decided unanimously to put to a Special General Meeting to be held on June 6th, a proposal to amend its Memorandum of Association by changing clause 4. to read as follows: (remove words in square brackets and add words underlined)

4. The income and the property of the Club whencesoever derived shall be applied solely towards the promotion of the objects of the Club as set forth in this Memorandum of Association and no portion thereof shall be paid or transferred directly or indirectly by way of dividend bonus or otherwise howsoever by way of profit to or amongst the members of the Club. Provided that nothing herein contained shall prevent the payment in good faith of any interest to any such member in respect of monies advanced by him or her to the Club or otherwise owing by the Club to him or her or of remuneration to any officers or servants of the Club or to any member of the Club or any other person in return for any services actually rendered to the Club. Provided further that no member of the Board of Directors or Governing Body shall be appointed to any salaried office of the Club or any office or the Club paid by fees and that no remuneration shall be given by the Club to any member of such Board of Directors or Governing Body except [provided that nothing herein contained shall be construed so as to prevent the allowance of an honorarium to any such member in respect of special honorary services rendered or] the repayment to any such member of out of pocket expenses and interest on money lent or hire of goods or rent for premises demised to the Club. Provided that the provision last aforesaid shall not apply to any payment to any railway gas electric lighting water cable or telephone company or corporation of which a member of the Board or Governing Body may be a member or to any other company in which such member shall not hold more than one hundredth (1/100) part of the capital and such member shall not be bound to account for any share of profits he may receive in respect of such payment.

The April meeting was well attended with most just standing around and having a chat and no doubt, catching up on the latest gossip. For a Buy, Swap Sell day there was not a great deal on offer though I did notice Peter Schubert off loading a few items to that inveterate buyer of just about anything, Russell Welham. I would hate to see Russell's home as I reckon he would need large extensions to accommodate all his computing gear.

Dick Cooper went home happy having had twin double sided drives installed in his system. Go for it Dick. This addition will greatly enhance your computer use.

The May meeting is an all day event with plenty of tutorials and workshops being organised. In addition there will be a lunch time BBQ so if you can, come along and join in the fun.

We only have one new member to welcome this month, and this is Margaret Scott of Carnarvon, way over there in Western Australia. Welcome aboard Margaret.

On the sick list has been Bill Walker of Hallidays Point. My informant, Frank Hall of Old Bar tells me Bill was in Taree Hospital for a while but now appears to be recovering. Hope you get well soon Bill and all the best from us all.

I think most members are interested in how the club is going membership wise, at least it would seem so as this is a frequent question that I get asked. To let you know I have done some analysis and the result is shown below:

Current Membership (Australia) 155
Current Membership (Overseas) 9

As at the day of writing this article there are 70 memberships that expire at the end of April 1992, so as can be seen this represents a significant percentage of the total membership. My anticipation is that most of the 70 will renew either at the next meeting or sometime during the month of May. We can therefore look forward to maintaining a membership base of approximately the same number as what we have now.

If you cannot remember when your membership expires, either give me a call or more simply just look at the mailing label on your newsdigest. The date of expiry is clearly shown on the first line of the label.

That is it for this month. Hope you can come along and join in the fun at the May meeting. o

Assembly Class for June

by Ross Mudie

The assembly class topic for June 1992 will be USING DSRLNK UNDER EXTENDED BASIC.

All TISHUG members are very welcome, the class will start at 10am, 6th June at Ryde Infants school, please try to arrive on time.

ALL members planning to attend this class should read pages 262 and 291 to 304 of the Editor Assembler manual before attending the class. It will be essential for class members to bring their own Editor Assembler books and note pad to this class. The people who attended the class on 4th April have received a paper copy of the Amateur Radio Log program. This provides an extended basic DSRLNK and includes SAVE and LOAD for MEMORY IMAGE files.

Memory Image is much quicker to load and save than files in display or internal format. Remember to bring your copy of the notes to the next class. Any member who was not at the meeting 4th April wanting a copy of the notes should contact me.

PLease note:

I will not be available to conduct the assembler class in May, but there will be plenty of things of interest at the ALL DAY TUTORIAL on 2nd May. o

Letter to the Editor

Dear Sir,

My name is Stanculescu Marin and I have a Texas Instruments TI-99/4A computer, but I do not have any software for it.

Please publish my name and address in your newsletter and tell everyone that I need software applications for my TI-99/4A. Any software on cassette or disk, modules or books on the TI-99/4A would all be most welcome.

Yours sincerely,

STANCULESCU MARIN
Str. CIURULEASA, Nr. 3, Sector 4,
Of.P. 7, Cod 75445 - BUCURESTI
ROMANIA

For Sale

* Myarc Geneve 9640 with Monitor
* Cable to fit Phillips CM 8833 or Commodore Monitors
* Lots of Software

PRICE: $500 + p/p

RING: Larry (07) 202 1884

TIsHUG Shop with Percy Harrison

Hi! Firstly I thought I should remind all club members of my advice given in last months TND that I will no longer be transporting the club hardware to each monthly meeting, if you want to purchase hardware and collect it at the meeting you must ring me prior to the meeting and order it so that I can have it available at the meeting. Please try to get your order in at least two days before the meeting to give me time to parcel it up. This will save me the trouble of carting heavy bags of hardware to the meeting and then having to take everything home again as quite often there are no sales of hardware made. Your co-operation in this matter would be greatly appreciated.

Also another reminder that we are still taking orders for the 80 Column Cards (TIM/SOB) at the special discount price of $165 each so if you want one and have not yet paid me for it please get you $165 to me post haste as this offer will not last indefinitely. My postal address is 3 Storey St Ryde 2112.

It is now two months since I asked members to jot down their views on the type and quality of software programs that we release through the shop and I am very disappointed to report that, to date, I have not had one reply. Am I to believe that all our club members are 100% satisfied with the material we release on disk or is it that I am expecting too much of our members by asking them to take five minutes of their time to give me an appraisal of our software quality and either post it to me or hand it to me at a club meeting?

PRICE LIST

5.25 in. DSDD Disks (Boxes of ten)	$6
5.25 in. HD Disks (Boxes of ten)	$10
3.5 in. DSDD Disks (Boxes of ten)	$10
5.25 in. DSDD Half Height Drive (New)	$65
12 Volt AC Transformer	$4
13 Volt Arlec Transformer	$12
8.5, 17 Volt Transformer	$25
60 VA Transformer	$20
MFC Printed Circuit Board	$30
MFC Kit (Disk Controller	$103
Music Kit with PCB	$65
32K Memory PC Board	$7
Eprom Ram PC Board	$45
Eprom Ramdisk Basic Kit	$35
Funnelweb Eprom Set (3 Eproms)	$36
TI Artist Eprom Set (2 Eproms)	$24
32K Static Ram IC (62256)	$10
8K Static Ram IC (6264LP)	$5
Exchange Console	$30
ROS Version 8.14	$12
TI Power Supply	$150
TI 32K Memory Card	$40
Multifunction Card	$220
RS232/PIO Card	$100
Modem PE Card (300 Bd)	$80
PE Ramdisk (192K)	$160
Printer (Serial)	$120

MODULES

*Adventureland Cassette + Book	$5
*Mission Impossible Cassette + Book	$5
*Pyramid of Doom Cassette Only	$3
*The Count Cassette + Book	$4
*Voodoo Castle Cassette + Book	$5
Adventure Module with Disk or Cassette	$15
A-Maze-ing Module Only	$5
Beginners Basic Tutor Cassette + Book	$4
Blackjack and Poker Module + Book	$8
Blasto Module + Book	$8
Bridge Bidding I,II,III Cassettes + Book	$15
Buckrogers Module + Book	$8
Carwars Module + Book	$8
Console Writer Module Only	$20
Demonstration Module Only	$5
Disk Manager 1 Module Only	$5
Disk Manager 2 Module Only	$10
Draw Poker Cassette Only	$3
Equations Module + Book	$8
Extended Basic Module + Book	$25
Hangman Module + Book	$8
Indoor Soccer Module + Book	$8
Multiplan Manual Only	$10
Parsec Module + Book	$8
Percents Module + Book	$8
TI Invaders Module + Book	$8
TI Writer Module + Manual	$25
Teach Yourself Basic Cassette + Book	$4
Terminal Emulator Module + Book	$15
Touch Typing Tutor Module Only	$5
Tunnels of Doom Module + Cassette + Book	$12
Zero Zap Module Only	$5

Note: Items marked * require the Adventure Module to run.

Packaging and postage extra on all items.

Bye for now. o

continued from page 14

POSition

One way to ask for a menu selection is to ask the user to input the first letter of his/her choice. Suppose that the options were <C>hange, <P>rint or <Q>uit. You might do this:

```
190 ACCEPT AT(10,10)SIZE(-1)
    VALIDATE("CPQ"):A$ ::
    IF A$="C" THEN 230 ELSE
    IF A$="P" THEN 340 ELSE
    IF A$="Q" THEN 980 ELSE 190
```

A simpler way would be to use POS:

```
190 ACCEPT AT(10,10)SIZE(-1)
    VALIDATE("CPQ"):A$ ::
    ON POS("CPQ",A$,1) GOTO
    230,340,960
```

If your user inputs a null, the POS function will return a value of 1 and control will transfer to the first line in the GOTO list. If this is a problem, do it this way:

```
190 ACCEPT AT(10,10)SIZE(-1)
    VALIDATE("CPQ"):A$ ::
    IF A$="" THEN 190 ELSE
    ON POS("CPQ",A$,1) GOTO
    230,340,960
```

(Source: a Tigercub program)

DIM's and SUBPROGRAM's

Will this program work?

```
10 DIM A(5)
20 CALL SETUP(A())
30 SUB SETUP(B())
40 FOR I=1 TO 10
50 B(I)=I
60 NEXT I
70 SUBEND
```

Answer: It will crash in line 50 when I=6. Once A has been DIMensioned in line 10, the process of calling SETUP in line 30 transfers the DIMension to B within SETUP. In essence, there has been a DIM B(5) inside SETUP.

Note that a STOP is not needed at the end of line 20. Your 4A will not execute a SUBPROGRAM unless it is CALLed.

Enjoy! o

TIsHUG Software Column by Rolf Schreiber

This month is an all day tutorial, and the software library will be available for people to browse through, and select disks for copying. There will be a small charge to cover the cost of copying any disks which are not currently on sale in the shop. You will first need to see Percy and pay him the money for the number of disks you want from the library. He will issue you with a receipt, which you will need to present when you have selected the disks that you want copied. Do not forget that you also must have the correct number of disks for the software to be copied onto. These disks do not have to be previously formatted, nor do they have to be blank; the track copier we use first formats the disks in all cases.

If there is time, I will be available to answer (or at least attempt to!) any questions about software, as well as demonstrating some of the current range of commercial software.

Software releases for May

DISK A485 is a letterhead and address list creating program from B. Barnardh, who used to be a member of the Brisbane based TIBUG User Group several years ago. There is also an invoice/quote form generating facility, but I have not tested the programs sufficiently well to tell you much more than this. There are no instructions on the disk, but the programs are all menu driven and it should not be too difficult to work out if you have a need for this type of software. A disk catalogue is shown below:

A485 Diskname: YEO Format: SSSD

Filename	Size	Type / Length	
ADD	5	PROGRAM	914
ADD/LIST	24	PROGRAM	5868
ADDRESS	101	INT/FIX	120
BLANK	12	PROGRAM	2658
CREATE	3	PROGRAM	336
INVO/QUOTE	21	PROGRAM	4944
L/H	2	INT/VAR	250
LETTERHEAD	7	PROGRAM	1498
LOAD	5	PROGRAM	1003
S	2	INT/VAR	192
YEO	34	PROGRAM	8202

DISK A486 is a very comprehensive genealogy data base called Genealogy Record Keeping V1.12, by Allen Wright of the Hunter Valley 99'ers User Group. The data base is written in assembly language as a DIS/FIX 80 object file. The program is loaded by the Editor/Assembler option 3 loader, or the Funnelweb equivalent. The file to load is GENEALOGY, which then loads the other files as modules. The data base is menu driven and comes with extensive documentation and help files, which were written by Jim Grimmond. A more detailed description is the subject of another article in this month's TND.

A486 Diskname: GENEALOGY Format: DSSD

Filename	Size	Type / Length	
-READ-ME-	17	DIS/VAR	80
ABBREV/1	24	DIS/VAR	80
APPEND-1	77	DIS/VAR	80
BEG-GENE	42	DIS/VAR	80
GENE/4GEN	21	DIS/FIX	80
GENE/DOC-1	105	DIS/VAR	80
GENE/INDE	6	DIS/FIX	80
GENE/PRNT	34	DIS/FIX	80
GENEALOGY	66	DIS/FIX	80

DISK A487 is an 80 column card high resolution graphics utility called XHI, from Alexander Hulpke in Germany. XHI loads from Extended BASIC and requires the 9938 VDP chip, which means that either the AVDP or the Mechatronics 80 column card must be present. The program also runs on the Geneve in GPL mode. Once the object file has been loaded into memory, using the XB loader supplied, you can run several demonstration programs, or else start programming yourself. All the advanced graphics features offered by the 9938 video chip are made available in the Extended BASIC environment by linking to routines in XHI. In this way XHI is similar to The Missing Link, but it must be remembered that XHI is designed to display your graphics in 80 columns, while TML is limited to 40 columns. It is also important that the diskname is 'XHI' or else the demonstration programs won't run. There are extensive documentation files on the disk (in English), as well as the source code (with comments in German). I ran XHI on the Geneve, and it performed very well. If you own an 80 column card then this disk will open the doors to programming with high resolution graphics, and greatly simplifies seemingly difficult tasks, such as loading high resolution pictures into an Extended BASIC program.

A487 Diskname: XHI Format: DSDD

Filename	Size	Type / Length	
-README	8	DIS/VAR	80
CHARA1	5	PROGRAM	1024
COLDEF	9	PROGRAM	2046
FIXSTERNE	18	PROGRAM	4274
HARDCOPY	22	PROGRAM	5242
HCLOAD	2	PROGRAM	154
HCSETUP	10	PROGRAM	2067
HIRESDEMO	24	PROGRAM	5749
KUGEL	22	PROGRAM	5316
KUGEL;PAR	5	DIS/VAR	80
KUGEL;PIC	29	DIS/FIX	128
LOAD	44	PROGRAM	10877
SAMPLE-DOC	30	DIS/VAR	80
UHR	5	PROGRAM	873
X-EARTH	32	PROGRAM	7892
XHI	84	DIS/FIX	80
XHI/T	167	DIS/VAR	80
XHIDOC	220	DIS/VAR	80
YLOAD	8	DIS/FIX	80

DISK A488 is a disk full of assorted utilities from Tom Freeman of the LA99'ers User Group. One of the most useful programs on this disk is the CHECKSUM program, now extensively used by MICROpendium for all their program listings. When I bought this package from the USA, it was accompanied by about 20 pages of printed documentation. I would like some volunteers to type up this information, so that it may be reprinted in the TND in the near future. Failing this, if there is sufficient interest (at least 5 enquiries), we will photocopy the documentation for $5.00. Please direct all your enquiries to Percy!

A488 Diskname: UTIL/TF/LA Format: SSSD

Filename	Size	Type / Length	
/READ/THIS	3	DIS/VAR	80
ALLKEY	11	DIS/VAR	80
ASL/CL	14	PROGRAM	3273
CHECK/CL	15	PROGRAM	3534
CHECK/O	10	DIS/FIX	80
CHECKSUM	5	PROGRAM	946
CL/ASL	12	PROGRAM	2618
DATAMERGE	6	DIS/VAR	163
DATAMERGE1	11	DIS/VAR	163
DATAMERGE2	17	DIS/VAR	163
DISKTAPE/O	6	DIS/FIX	80
DISKTAPE/P	4	PROGRAM	664
HIDDEN	3	DIS/VAR	163
KBRD/FRMTR	12	DIS/VAR	80
MAKE/DATA	6	PROGRAM	1100
MAKE/DATA1	6	PROGRAM	1110
MAKE/DATA2	6	PROGRAM	1039
ORIGINS	6	PROGRAM	1123
PRINTMERGE	4	PROGRAM	579
QUADCOL	11	PROGRAM	2482

QUADCOL/AL	15	PROGRAM	3467
READ/DV80	6	PROGRAM	1121
SECTOR/O	18	DIS/FIX	80
SIDEWYS/CH	9	PROGRAM	1950
SIDEWYS/SC	9	PROGRAM	1929
SWAYS/AL	21	PROGRAM	4867
SWAYS/NODT	5	PROGRAM	867
SWYS1/NODT	5	PROGRAM	784
SWYS2/NODT	5	PROGRAM	784
VARLISTER	9	PROGRAM	1851
XBTOKENS	24	DIS/VAR	80

DISK A489 is FONTART 1, Ver 1.0, (c) 1987 by P&A Software. The author is Paul E. Scheidemantle from Michigan in USA. This disk includes 11 files of fonts for TI-Artist, in six different font styles. If you use these files Paul suggests that you send him $5.00. His address details are in the README file on the disk.

A489 Diskname: FONTART1 Format: SSSD

Filename	Size	Type / Length
--README--	4	DIS/VAR 80
3DLG/A1_F	42	DIS/VAR 80
3DLG/A2_F	28	DIS/VAR 80
3DLG/B1_F	40	DIS/VAR 80
3DLG/B2_F	26	DIS/VAR 80
3DLG/C1_F	44	DIS/VAR 80
3DLG/C2_F	29	DIS/VAR 80
LCHR/F_F	37	DIS/VAR 80
OLDE/LL_F	32	DIS/VAR 80
OLDE/LN_F	18	DIS/VAR 80

Tigercub Software Releases

As you are probably aware by now, Jim Peterson is the most productive programmer in the entire TI-99/4A world, and that has been achieved against some very stiff competition. Besides writing a huge number of articles that have helped innumerable beginners to programming, he has collected a vast library of public domain software for the TI-99/4A, meticulously sorted into categories and ready to run. This month I am releasing three disks from Jim's Tigercub collection. I hope that you enjoy them.

TCC-6 is the sixth disk in the Tigercub Collection. As with all the others in the series, the disk is packed full of educational software and games. The Tigercub menu loader allows to load each program by selecting its number from the list below:

1 Can of Worms
2 Rithmatik
3 Long Division Cryptos
4 Nimbo
5 Glunk
6 100%
7 Addition Magician
8 Arithmagraphs
9 Bagels
10 Digitron
11 Four In A Row
12 Going Home
13 Gomoku
14 IQ Math
15 Tigercub Math Puzzle
16 Mawari
17 Multiplication Madness
18 One Check
19 One To Five
20 Othello
21 Sphinx

TC-1145 is a telecommunications aids disk which also contains two games which you can play using a modem. The disk is menu driven and contains the following items:

1 Encoding and Decoding Program
2 Mail Prep.
3 Phonemake
4 PRBase Modem Utilities
5 Battle of the Sea Lords
6 Off-Line Source Mail
7 TE II Log-On File Generator
8 TEX-THELLO for modem
9 XPREP V1.1
10 40 Column BBS Mail Prep.

The PHONEMAKE program was designed to be used in conjunction with Stuart Olson's Mass Transfer V4.1.

The PRBase modem utility allows for the packing and unpacking of PRBase data files into a format suitable for transmission by modem, using the X-MODEM protocol.

TC-1210 is devoted entirely to graphics printing routines. The programs on this disk have been written by various authors and are either shareware or public domain. On booting the disk, the menu displays the following items:

1 TI-Artist Instance to XB
2 TI-Artist Picture Loader
3 Printer Graphics Expander
4 Graphics Dump to TI-Writer Formatter
5 Screen Dump in Assembly
6 TI-Artist Instance Utility
7 TI-Artist Instance Printer V2.0
8 DIS/FIX 128 to DIS/VAR 80 Converter
9 MAX-RLE Loader
10 Snowflake (CHAR Pattern Designer)
11 Snowplay (Snowflake viewer)
12 Reverse RLE pictures
13 MAX-RLE Extended BASIC Loader
14 TI-Artist Instance to CALL LINK Compiler
15 Inverse Subprogram
16 TI-Artist Squeezer

The following descriptions have been mainly derived from information contained within each program.

128TO80 is an assembly language routine from Bud Wright in the USA. The program converts text and graphics from a DIS/FIX 128 format to a DIS/VAR 80 format. This conversion is essential when you download or transfer files using the XMODEM protocol, which transmit data in 128 byte blocks and write to disk in DIS/FIX 128 format, which cannot be loaded into Funnelweb. This is only V1.0 and the program has no error trapping, which is awkward if either your source or destination files do not exist.

ART>XB is a TI-Artist Instance to Extended BASIC program converter from The Smart Programmers, with additional print/dump/merge options by Lucie Dorais of the Ottawa User Group. This program will show the Instance by using Extended BASIC CHARs from 143 down to 32, until all are used. You can then dump the Instance to a printer by pressing 'D' (allow time to load the dump object code).

ARTP/LOAD is a TI-Artist picture loader by Jürgen Switalski from USA. This fairware program allows you to load TI-Artist pictures in a running Extended BASIC program.

DEFPRINT, by Steve Vickers of the Canadian OTTAWA User Group, is a printer graphic encoder which returns the ASCII values and corresponding characters for any graphic design created using the built in CHAR editor. The program can handle a maximum of 31 bit-columns.

DUMP/TIW is a TI-Writer bit-image printer graphics utility by Richard Mitchell, from the Super 99 Monthly. To use this utility, save it in MERGE format then merge it into any program. It will dump any screen of text and/or graphics into a D/V 80 disk file which can be printed through the TI-Writer formatter mode on any Epson/Gemini type printer.

continued on page 15

Check your renewal date. Do not miss out.

Games Information

by Robert Brown

[Editor's Note: This is the first of a series, the second of which was put in last month. The two articles had to be printed in reverse order because the first one was received in a corrupted state and I had to wait for another copy before printing it. I trust that this did not inconvenience anybody.]

Welcome to yet ANOTHER Games Information article. I have not written an articles since the September (1991) TND, as I have been busy with University essays and exams, but since these have all finished, I am now free (well, to a certain extent) to write some more informative, well read (I hope) articles.

Since I have not written any articles for a while, this article will be a BUMPER one!! Way back in the May TND of 1989, I wrote some hints to solve Zork I, made by Infocom. Today, I will give you the solution. If you have not solved it by now, you will never solve it! (Sorry !#@?! - well it is true!!!).

You begin West of the house, and your first chore is to get inside. So, go South and East. Open the window and enter the house (you are in the kitchen), then go West into the living room. Get the lamp, then move the rug, revealing the trap door. Open the trap door, turn on the lamp, and go down. At this point, some mysterious person will shut the door on you, do not worry about that for now.

Okay, so here you are in the cellar. It is time to pick up your first treasure, so go South, then East to the Gallery. Get the painting, then continue North to the Studio. Go up the chimney (you can only fit with the lamp and the painting), and you will be in the kitchen again. Now, go upstairs to the attic, and get the knife and rope. Come back down and go into the living room.

Open the case and put the painting inside. Then, drop the knife and get the sword. Open the trap door again, and return to the cellar. Again, the door is shut by someone (you will never find out who is doing this, but it does not matter). Now you are back in the cellar, and since we are coming to one of the more dangerous parts of the game, you might want to do a save here - well you should be saving it after every 20 or so moves, just in case you STUFF UP!!.

Gripping your trusty sword, head North into the Troll room. There is a nasty troll here with a bloody axe, and the only way past is to kill him. So, do just that: "Kill Troll With Sword." It will most likely take more than one attempt, so keep at it, and eventually he will disappear in a cloud of black smoke. Now, drop the sword, because you really do not need it any more, and it will hinder you in carrying other, more important items.

Having dispatched the troll, you move along East, East (into the Round room), then SE and East. You are now in the Dome room. It is a long way down, and too far to jump, so here is where the rope comes in handy. Tie the rope to the railing, then climb down the rope. You will be in the Torch room. Leave the torch for now; you will be coming back this way again later.

From the Torch room, go South, then East and get the coffin. Return West, then continue South to the Altar. There is no way you are going to get down that hole with the coffin, and even the program will tell you that you have not got a prayer. That is a hint, folks: Just "Pray," and you will find yourself in the forest again. Since it is daylight out, save energy and turn off the lamp. Now, head along South, then North (I know, but it works!) to the clearing, then East to the Canyon View. Climb all the way down to the bottom, then go North to Rainbow's End. Drop the coffin and open it. Inside is a jewelled sceptre. Get that, and wave it. The rainbow will become solid (you will need to cross over from the other side later). Now, "Look." You should see a pot of gold. Get that, and the coffin.

After that, go SW, then all the way back up to Canyon View. From there, it is NW to the clearing, and then West to the window. Once in the kitchen, open the bag and get the garlic (nothing else, just the garlic). Go on into the living room, and put your treasures in the case. Now, sit down and take a breather, because you are about to do a lot of travelling!

Once again, open the trap door, turn on the lamp, and go down. Watch carefully, and you will notice that this time, the door does not close! Whoever was doing it before must have gotten bored. Anyway, you are on your way to the dam, so move along North, East, North, NE, and East. You are now on top of the dam.

From there, go North to the Lobby. Pick up the matches, then go either North or East (does not matter) into the Maintenance room. Get the wrench and the screwdriver, then push the yellow button. Now, return to the dam, and you will see that the green bubble is lit. Turn the bolt with the wrench, then drop the wrench. You have opened the dam, and you will be coming back this way again to reap the fruits of your labours. However, right now, you are on your way to Hades, so let's get going!

Go South, then down into the Loud room. Leave the platinum bar for now; you will get it later. Head West into the Round room, then SE and East (hmmmm, have you not been here before???). Again, climb down the rope. This time, get the torch. At this point, you can turn off your lamp, the torch will provide light so long as you have it.

Now, continue straight South, getting the bell, then the book and candles from the altar. Go down the hole to the cave, then down again to the entrance to Hades. Your candles will have blown out by this time, but do not worry about it. Okay, here is where you have to be careful. First, ring the bell. It will become red hot and you will drop it. You will also drop the candles. Stay calm, and do the following, all in one command: Get the candles, light match, light candles with match (necessary, because of the torch, and DO NOT use the torch, or you will vapourise the candles!!). Okay, strange things happened when you lit the candles, now read the book. Whew! The demons have been exorcised! Drop the book, then go South and get the crystal skull. Now, back North, then up to the cave, then North to the Mirror room. By the way, better put out the candles. Rub the mirror, and you will now be in another Mirror room (this one is North of the dam, as the other one is South).

Now, go North, then West, then North, then West into the Squeaky room (well, I told you you would be doing a lot of travelling this time!). Make sure you have the garlic with you, then go North into the Bat room. So long as you have the garlic, he will not bother you. There is a jade figurine here, but leave it for now. You will pick it up on your way out.

Go East to the Shaft room. Put the torch into the basket, then turn on your lamp and head North to the Smelly room, then down to the Gas Room (best not to carry any open flames here!!!). Now, you are about to enter a small maze, so follow these directions CAREFULLY!!! East, Northeast, Southeast, Southwest, down, and you should be at the top of the ladder (if you are not, may God have mercy on your soul!). From there, go down to the Ladder Bottom, and then South to the Dead End for the coal. Get that, then return to the ladder top. From there, go up, North, East, South, North, and you will be back in the Gas room. Go up, then South to the Shaft room again.

Put the coal in the basket, and lower the basket. Now, guess what? You have to go back through the coal mine again! So, make your way to the Ladder Bottom, but this time, go West to the Timber room. Ignore the broken timber (not useful for anything), and drop all

but the screwdriver. Now, you can squeeze through the crack to the West.

And here you are in the Drafty Room, which is also at the bottom of the shaft. There is the basket, so get the coal and the torch, and move South into the Machine room. Open the lid, put the coal in the machine, close the lid, and turn the switch with the screwdriver. Drop the screwdriver, open the lid, and get the diamond (well, no one ever said Zork was an EASY game!). Now, go back North, and put the torch and the diamond in the basket. Squeeze back East into the Timber room. Get the skull, lamp, and garlic. You will not be needing the matches and the candles any more, so you can leave them. (They were insurance in case the thief came along and stole the torch before you could get the diamond). Now, head East again to the Ladder Bottom, and from there up and through the coal mine (you know the way now!), to the Gas room. Pick up the bracelet, then continue on up and South to the Shaft room. Get the torch and the diamond from the basket, turn off the lamp, then go West into the Bat room. Get the jade, then go South, East, South to the Slide room.

Now, here is a fast way back to the cellar: Just go down the slide! Wheeeee! Then it is up to the living room (remember, the trap door is open now), and all the treasures go into the case.

Turn on the lamp, and return to the cellar. From there, it is North (ah, deja vu!), then East, North, NE to Reservoir South. Now that the sluice gates are open, you can head North, picking up the trunk of jewels, North again to Reservoir North, getting the air pump, and North one more time, getting the crystal trident. After that, go all the way back South again to Reservoir South, then East to the dam, and then East once more to the Dam Base.

Here you find a little pile of folded plastic; guess what it is? Right, it is an inflatable boat! So, inflate it with the pump, then drop the pump, then get inside the boat, say "Launch," and you are floating off along the Frigid River. Now, just keep waiting until you see the buoy. Get that, then "East" to the beach. Get out of the boat, then get the shovel and move on to the Sandy Cave to the NE. You might want to save the game at this point, since you have to dig here until the scarab turns up, and I am sure you do not want to get buried alive (it is been known to happen!). Okay, drop the shovel and get the scarab, then go back SW. Drop the buoy and open it, inside is an emerald. Get that, then continue South to the Aragain Falls.

Here you can cross the rainbow, which brings you to the End of The Rainbow (that was obvious). Turn off the lamp, then go SW to the Canyon Bottom. From there, make your way back to the living room, and put all the treasures in the case. Your collection is quite impressive by now, but you are not finished yet. Go East twice, then North twice. Climb up the tree and get the egg. Climb down again, and go South, East, and back to the living room. However, this time, you do not put the treasure in the case.

Turn on the lamp, and go down (once again!) into the cellar, and North to the Troll room. Now, you are about to enter a maze, so follow the directions very carefully! West (this brings you into the maze), South, East, up, and you find several items here. Take only the coins and the key, and be careful not to touch the skeleton! From here, go SW, East, South, SE, and you will be in the Cyclops room. The Cyclops is not friendly, but you can deal with him effortlessly: Just type in "Ulysses" (or "Odysseus," if you prefer). Old One-Eye will tear out of there right through the wall! In fact, he will create a passage eastward from that room right into the living room!

However, you do not want to go that way yet! Instead, go upstairs, and you will be in the Treasure Room, the thief's secret lair. Now, give him the egg, and go back downstairs, then East to the living room. Deposit the coins in the case, then get the knife (the thief needs a little time to open the egg). Okay, go back West to the Cyclops room. Again, at this point, saving is highly recommended; the thief will not be easy to kill!

So, head upstairs and use the nasty knife to kill the thief. Once he is dead, all treasures in the room will be visible. This includes the egg, a silver chalice, and anything he may have stolen from you before. Get everything, then follow these directions: Down, NW, South, West, up, down, NE, and you will be in the Grating Room. Unlock and open the grate (watch out for falling leaves), then go up. You will be in a clearing. From there, go South and climb the tree again. Wind up the canary That is inside the egg. A songbird will come by and drop a bauble for you. Climb down again and get the bauble, then return to the living room. Put all the treasures in the case, making sure you REMOVE THE CANARY from the egg and put it in the case separately! You are almost finished. Just one more trip to make!

Now, for the last time, enter the cellar and go North. From the Troll Room, go East until you come again to the Loud Room. Type in: "Echo," and you will now be able to get the bar. So, grab it and return to the living room. Once you place it in the case, you will get a message. Follow the advice of that message, and you will get a map. Take that, and return to the place where this all started, the mailbox West of the house. You should have no trouble getting to the barrow from there. Of course, once you enter the barrow....

You did not really think it would end there, did you? Not when there is still Zork II and Zork III (also Beyond Zork - but not available on the TI) waiting for you up ahead! Ah, but it is too late; you cannot turn back now! You will just have to grit your teeth and follow through to the end (with a little help, of course). See you in Zork II! The solution to Zork II we be revealed in next months article (see last month's issue). If you are really desperate for some help, just just give me a call (see information at the bottom of the bottom of the article).

Since this is a BUMPER ISSUE, I will continue with another solution to an Infocom adventure - this adventure being WITNESS. I gave some hints and a map of this adventure way back in the December 1989 TND, so I thought it would be about time to give you ADVENTURE LOVERS the solution.... finally!!

Here goes....

Okay, you start South of the house, where you just picked up a matchbook. Go North twice to the front door and ring the bell. Phong will let you in. Then just try to go East, and Phong will lead you to the Living Room, where Monica and Mr Linder are.

Now, wait (get used to doing that, because there is a lot of waiting in this one), and Linder will eventually take you to his office. Sit down in the wooden chair, and Linder will hand you a note. Read it, as it will help waste some time. Now, just do anything (but stay seated!) to make time pass. Show the matchbook to Linder for an interesting reaction, if you like. In any case, you just have to keep waiting.

Eventually, Monica will come in briefly to announce she is going to the movies. This is not what you are waiting for, however! So, keep on waiting, and finally, the murder will occur. Linder will be shot while you sit there, and you cannot stop it from happening. Read the description carefully at the moment the shot is fired. There is something odd about it. In fact, the whole thing is a setup.

The first thing to do is stand up, then push the button. Instead of ringing to summon the butler, it causes a strange click to be heard from the clock. At this point, Phong will enter the room. Tell him you want the keys, and he will hand them over to you. Now, examine the clock. Keyhole seems a little strange, does

not it? The doorbell rings while you are doing this, so as Phong goes to answer the door, examine the keyhole.

I will bet you are getting some ideas already! However, you will need to have the powder analyzed, and Duffy has not arrived yet, so wait around until he does. Then get the powder analyzed (you can ignore Stiles, he is only a red herring). While That is being done, examine the window (you cannot open the clock yet, it is the one key you do not have). The green wire seems suspicious, so get it for future reference.

Now, go West into the Hallway, then North twice, and open the Butler's door. Go West into the room, and read the mystery book (by the way, you can drop the telegram and note, they are not important). A gun receipt is used as a bookmark. The purchaser's name is obviously phoney, but hang on to the receipt anyway.

Okay, from the Butler's Room, go East twice to Monica's room, then unlock and open the back door. Go East into the Backyard, then South twice to the office path. Aha, a muddy gun! No fingerprints, alas, but you might want to take it along with you, just in case. Now, go West into the Side Yard.

Hmm, more footprints here, but they are not quite the same as the ones on the office path. In fact, it looks like someone was standing here for awhile. Wonder who it might have been? (No, not Sergeant Duffy!). Anyway, go West again to the driveway, then North and East into the Garage.

Unlock and open both the garage door and the workshop door, then go East into the Workshop. The place looks like an electrician's paradise, and there is not much you can do here; but, there are spools of wire hanging around. Could it be...? Examine spool, and you have established a link of sorts between this place and the study. The green wire is obviously from this room. Now, all you need is the person who put it there.

You now stand there waiting for Monica. Just keep waiting; she will arrive (saying "Wait for Monica" is easiest. It will take a while, so if you want to hunt down Phong and ask him about the gun receipt, you have time). When she does get there, she will fiddle briefly with the junction box (very suspicious!) before noticing. Now, wait until she leaves, then follow her. You MUST use directions here, just saying "Follow Monica" will not work.

Follow her all the way to her room, and wait for her to come back out of the bathroom. When she returns, ask her about Mr Linder. Her response will establish the motive. Now, wait some more, and she will eventually leave the room. Follow her again, this time to the office.

As soon as you get in there, handcuff her. Somewhere along the way, Sargeant. Duffy will have left with the body, so you cannot arrest her until he comes back. In the meantime, you have to find some very important evidence. So, first search Monica for the key. When you get it, unlock the clock and open it. She is already removed the gun, but you can search her for that, also.

Now, just wait until Duffy returns, and arrest her for the murder. And that should be about it. By the way, if you try leaving her and waiting in the office (so you can find the gun in the clock), you will find that, however hard you try, you will not be able to handcuff her (which is necessary so she can be searched). So, you will just have to wait and follow her.

Well, That is it!! Did you not always want to solve a murder!!

This is the end of this BUMPER ISSUE of GAMES INFORMATION. I hope you have enjoyed it, and some people actually found it useful!!

Coming up early in 1992, I have solutions to more Infocom adventures, such as Suspect, Starcross, Suspended, Seastalker (all the S's) and finally a Z - Zork II. I will also give reviews of Microsurgeon (for all those budding surgeons) and how to design screens for one of my favourite games - TI Runner. If I also get my act together, and post to the Editor some maps for Scot Adams/Infocom Adventures, you will have these too.

As usual, if you have any questions, I can be contacted in the following ways...

1. By the BBS, Username: Games
   Password: Not telling!!!

   For our Overseas Readers, the BBS is TEXPAC.. Phone (02) 456-4602. You must put a country code of "62" in front of the number, as with the phone numbers below!

2. By Phone...
   (02) 743-3019 Home
   (02) 516-2399 Work (Until about 7pm)

3. By Post... NEW ADDRESS

   46 Llewellyn Street
   Rhodes 2138
   New South Wales
   A u s t r a l i a

NOW you DO NOT have any excuses for not getting in touch with me!! This article is Copyright By Robert Brown - All Rights Reserved

Just a Short Note:

(From the Author) "This article consists of over 79 sectors and 3700 words. It only took me about 2 hours (MAX), so why do OTHER people not get to it, and write some articles, it is not very hard!"

- Overheard one day, when the author was boasting about his writing talents (or lack of them!!). O

continued from page 21

Line 930 and sub-routine 8670 will accomplish just that. If on the other hand the EPROM code is 65535 (65535 = >FFFF) then there is no more EPROM code left. Line 935 will send an appropriate message through sub-routine 8700 and sends the program to sub-routine 7000. Otherwise we will reach 940 and the decoded and completed line will be printed out. Line 950 will increment the line count and the byte address count. Line 965 has the same purpose as line 935 as well as an error trap if the EPROM program happens to be shorter than anticipated. Line 970 branches to sub-routine 7000, which could well have been made into a sub-program. This program segment handles the printout of the definition tables.

Finally, line 990 will close the opened files and ends the program. The printout sub-routine starts at line 1000 and first finds the appropriate op-code mnemonic then sub-routine 3100 will perform the actual printout routine of the disassembled program line. Actually these two sub-routines should have been combined into one. Alas, none of us is perfect, hindsight is 20-20 vision! It makes no difference, it works!

```
100 REM SAVE DSK1.PRINTFILE
105 DIM RC$(105)
110 OPEN #1:"DSK1.BAE17-80",INPUT ,DISPLAY ,FIXED 80
120 OPEN #2:"PIO",DISPLAY ,VARIABLE 110
130 FOR I=1 TO 104
140 INPUT #1:RC$(I)
150 PRINT #2:I;TAB(5);RC$(I)
160 NEXT I
170 CLOSE #1
180 CLOSE #2
190 END
```

continued on page 10

Techo Time

with Geoff Trott

MiniPE Disk Controllers

If you have a MiniPE system with a disk controller then you may be having similar problems to me and some other systems that I have been looking at recently. The MiniPE system is great because it is so portable and neat. However it has a few minor problems which cause more problems as time goes on. The main problem is, I believe, the power supply. The MinPE system gets its 5 volt power from the console. This causes the power supply in the console to run a bit warmer but otherwise it is able to cope with the extra demand.

The console power supply is normally set up to give 5.25 volts at its output. This then goes to the motherboard and makes its way to the connector at the right hand side of the console. The motherboard takes about 1.5 Amps so there is some voltage drop by the time the voltage gets to the edge of the board. There is only one contact on the connector for the 5 volt supply whereas there are 4 contacts for the ground. TI only use the 5 volts for the Speech Synthesizer and for that the one contact is sufficient. The disk controller board uses about 400 ma and so the single contact can be a problem if any dirt builds up on the contact (over time). The disk controller chip (or the 9901) seems to be very sensitive to its power voltage and becomes unreliable in operation if the voltage is less than 4.85. In order to get to the 2793 IC, the current has to pass through the contact on the motherboard and socket on the MiniPE lower board, cross to the right hand side at the front of the bottom board of the MiniPE system, up through another connector and then to the back of the disk controller board. There can easily be a drop of 0.15 to 0.2 volts in this path so that it does not take much resistance in the contacts to cause the voltage at the vital chips to fall below their reliable operating point.

Peter Schubert, when designing the system, recognised that the power supply would be a problem and made provision to put in an external power supply in the form of a three terminal regulator (7805) on the board. This is the best solution to the problem as it makes sure that the chips get a reliable power supply independent of the state of the plug-in contact and it takes the load off the console power supply making the console run cooler. This is also something to consider if you are going to install a TIM and SOB board set. Of course, there are ways of making sure that the potential losses in voltage are minimised, if you decide to keep the power from the console, by putting in wiring in parallel in the console and MiniPE system.

To install a local power supply you will need a 9 volt dc plug pack power supply capable of 400 ma, 7805 3 terminal voltage regulator, 100 uF 16 volt capacitor (tantalum best), rectifier diode (1 amp), Aluminium for heat sink, wire and plugs and sockets for the plug pack supply. If you look at the back right hand corner of the disk controller board, you will see an area of tinned copper with a hole (3 mm) in it. This is where the 7805 is bolted to the board with the heat sink under it. To the left are the 3 holes for the 7805 leads to go through the board and be soldered. To their left are two holes labelled with "-" and "+". These are where the power from the plug pack comes in. Solder wires from here to a plug/socket that can be mounted on the cover of the MiniPE system and into which the plug pack power supply can plug. Polarity of supply is important here. The diode (to protect against reverse polarity) goes just to the left of the "+" hole pointing to the back to a hole close to the edge of the board (anode at the front and cathode at the back). This leaves two holes to the right at the back (going clockwise) for the capacitor with the left hole positive and the right hole negative polarity. To disconnect this supply from the rest of the system, a link needs to be cut under the board. This link is in the middle of the board between the 74LS245 and 74LS32 using the wire of a lead of a decoupling capacitor. This wire comes through the board and loops back into another hole. It is simply a matter of cutting this loop of wire under the board. Probably the hardest part of the whole exercise is fitting the heatsink and power plug on the cover. It is best if you can attach the heatsink to the cover to get the heat away from the regulator as quickly as possible.

As I have explained before, there is a hardware bug in the disk controller (same one as in the AT card) which allows the disk controller chip to perform functions when it is not supposed to be selected. This is a particularly bad bug if you have a RAMdisk as well, but it could be a problem at any time. It seems to mainly cause a track on a disk (9 or 18 consecutive sectors) to become un-readable. The solution for the MiniPE disk controller is quite simple and so I would suggest that you do it next time you have the opportunity. You need a 74LS04 and a few pieces of wire. Take the 74LS04 and bend out pins 1, 3, 4, 5, 6, 8, 9, 10, 11, 12 and 13. Bending the capacitor out of the way, solder pins 2, 7 and 14 to the corresponding pins of the 74LS32 as you put the 74LS04 on top of the 74LS32 (between a 74LS245 and a 74LS30 in the middle of the board). Solder a wire from pin 1 of the 74LS04 to the end of the 2.2 kohm resistor connected to pin 4 of the 74LS20 about 1.5 cm away. Turn the board over to work on the bottom of the 74LS32. Note carefully where pin 1 of the 74LS32 is on the bottom of the board. Connect a wire from pin 1 of the 74LS32 to pin 11 of the 74LS32. locate the track that runs from pin 11 of the 74LS32 to a hole through the board close to the 74LS245. Cut this track and solder a wire from the hole through the board on the other side of the cut from pin 11, to pin 3 of the 74LS32. What this has done is to use a spare gate in the 74LS32 (pins 1, 2, 3) and insert it in series with the output from pin 11 (to select the 2793) to allow another signal (pin 4 74LS20, disk controller selected), suitably inverted by the 74LS04, to be combined in to produce a new output on pin 3 of the 74LS32 (to select the 2793). O

continued from page 9

```
10 CALL CLEAR :: DISPLAY AT(4,1):"HEX NO.?" :: ACCEPT
   AT(4,10):HEX$ :: GOSUB 6000
20 DISPLAY AT(5,9):DEC
30 DISPLAY AT(16,1):"PRESS A KEY TO CONTINUE OR
   X TO END"
35 CALL KEY(0,RV,SV):: IF SV<>1 THEN 35
40 IF RV=ASC("X")OR RV=ASC("x")THEN END ELSE 10
6000 REM HEX TO DECIMAL CONVERSION
6010 DEC=0
6020 L=LEN(HEX$)-1
6030 FOR I=1 TO LEN(HEX$)
6040 C=ASC(SEG$(HEX$,I,1))
6050 C=C-48
6060 IF C<10 THEN 6080
6070 C=C-7
6080 IF (C<0)+(C>15)THEN 6130
6090 DEC=DEC+C*(16 L)
6100 L=L-1
6110 NEXT I
6120 RETURN
6130 PRINT : :"ILLEGAL CHAR (";SEG$(HEX$,I,1);") IN":
     "HEX VALUE ";HEX$
6140 REM
6150 RETURN
```

Conclusions:

This tutorial is intended to help the reader to better understand the file handling techniques applicable to the TI99/4A, as well as to demonstrate the possibilities of working with hexadecimal code in an Extended BASIC program. Segments of this program and the associated file handling program may be used in other programs if the need arises. These may also be converted to short stand alone programs. One such example is given below, the stand alone hexadecimal to decimal converter program. The print file program is a further example. This 11 line BASIC program could be

continued on page 12

To See or Not to C

by Geoff Trott

The shop has a set of c99 disks for sale and so Rolf suggested I should do a series of articles on c to give you some idea of how you could use c to write programs. The first question that you might ask is why use c when Extended BASIC allows any program to be written and we are all familiar with BASIC. There are potentially two answers to this question. The first is that programs written in c run much faster than those written in Extended BASIC. The second is that c is a language that is slowly spreading through the programming community as the preferred language for writing a whole range of programs from scientific ones to real time control programs and systems programs like compilers.

c is not a very easy program to learn to use from scratch. It is normally taught to people who are already familiar with a structured language like Pascal. The strength of c as well as one of its problems is its ability to do all the things that assembler allows you to do. This means that the compiler allows statements that would cause an error from a Pascal compiler which can make it quite difficult to debug a program. Nevertheless, I think that the increase in speed makes c a very useful language to use on the TI99/4A when the same program written in BASIC would take hours to run. It is also an interesting language which is relatively standard and useful on a range of other computers. We have a Canadian, Clint Pulley, to thank for our current compiler and Tony McGovern's Funnelweb for making the building of c99 programs relatively painless.

Let me first introduce you to the contents of the three disk set which you can get from the shop. c99a - the first disk contains:

An information file which is reproduced here -

```
-README      10 d 80
```

The REL4 compiler program files -

```
C99C         33 Prog
C99D         33 Prog
C99E         33 Prog
```

An example Funnelweb Scriptloader file

```
C99SAVIT      3 d 80
```

The files which can be used with an "include" statement. Some contain code, others contain definitions.

```
BITRTN       16 d 80
BITWRT        8 d 80
CONIO         2 d 80
CONV;C        6 d 80
FLOAT;C      30 d 80
FLOATI        2 d 80
GRF1RF        3 d 80
RANDOM;C      4 d 80
SOUNDLIB      2 d 80
SPEECHLIB     2 d 80
STDIO         3 d 80
STRINGFNS    11 d 80
STRINGI       2 d 80
TCIO;C       18 d 80
TCIOI         6 d 80
```

Files which are part of the library, containing assembler code loaded in Editor Assembler option 3 mode.

```
C99PFF        2*D 80
C99PFI        3*D 80
C99PFI;O      2 D 80 Funnelweb version - use your version
CFIO         11 D 80
CSUP         14 D 80
FPRINTF       5 D 80
FSCANF        6 D 80
GRF1         14*D 80
PRINTF       13 D 80
SCANF        15 D 80
SOUND        10*D 80
SPEECH        4*D 80
SPRINTF       5 D 80
SSCANF        5 D 80
STRINGS      13 D 80
```

c99b - the second disk contains:

Some more Library type files for debugging and making program files.

```
CLOAD        11*D 80
DEBUG        32*D 80
FMSAVE        6*D 80 use your versions of these
FWSAVE        6*D 80    provided in Funnelweb
SAVE         13*D 80
SBUGC        45*D 80
```

Source files, both program and utility type files. Good examples of programs.

```
AR;C         43 d 80
CHRINT;C      8 d 80
CRULD;C      10 d 80
DIMTST;C      5 d 80
HEXSTR;C      9 d 80
INTCHR;C      7 d 80
MKDICT;C     11 d 80
OPT;C        16 d 80
PRSET;C      12 d 80
RNDTST;C      4 d 80
RUNOFF;C     45 d 80
SECTOR;C     57 d 80
SIEVE;C       5 d 80
SOUNDS;C      2 d 80
STRING;C     10 d 80
```

c99c - the third disk contains :

Documentation files.

```
AR;DOC       22 d 80
BITDOC       12 d 80
C99MAN1      47 d 80
C99MAN2      53 d 80
C99MAN3      36 d 80
C99SPECS     32 d 80
FLOATDOC     30 d 80
FMTIODOC      7 d 80
GRF1DOC      19 d 80
PRINTDOC      2 d 80
RUNOFFDOC    16 d 80
SPEECHDOC    37 d 80
STRINGDOC    10 d 80
TCIODOC      27 d 80
```

The first thing to do is to copy the compiler files (C99C, C99D, C99E) from the first disk onto your Funnelweb disk (or preferably into your RAMdisk with Funnelweb). These are loaded by the file CP on the Assembler menu and when the compiler finishes a return is made back to Funnelweb. The operation of writing and running a program with c99 involves a number of steps. The first step is to write the program using the Program Editor. This program is normally named with a ;C or /C on the end of the name. Then the compiler is run with the ;C file as input and a file with the same name but ;S on the end as the output file. This file is an assembler source file and so now Funnelweb can help the process. If the ;S file is "marked" as the work file, when the assembler is started it will have that file name as source file and as object file the same name with a ;O on the end. Then a scriptloader file can be prepared (using the program editor) to perform the loading function of the object files of the program along with any library files needed. An example of this is the file C99SAVIT on the first disk which follows:

```
* Script-Load for C99 files from hard disk

      AUTO
      FIND
      FILE "DSK.LIB.C99PFI;O"
      FILE "DSK2.;O"
      FILE "DSK.LIB.CSUP"
      FILE "DSK.LIB.CFIO"
      FILE "DSK.LIB.STRINGS"
      FILE "DSK.LIB.C99PFF"
      FILE "DSK.LIB.FMSAVE"
      LAST SAVE
```

Unfortunately, this is a file suitable for an earlier version of Funnelweb but the concepts are the same for the latest version. Firstly, comments are preceeded by an "*" in column 1. Commands then follow such as:

AUTO - autostart the program at the name following LAST.

FIND - no longer used.

FILE - specifies the files to be loaded. Must be in quotes, single or double, without spaces within the quotes.

LAST - the end of the file with the following name used as the link name for starting the program.

In the example above, for loading a c99 program and then saving it as a memory image program file, the first file to be loaded is the Funnelweb C99PFI file to insert SLOAD and SFIRST at the start of the program. The next file is the main program (plus any sub-programs if needed) which would have its name inserted before the ;O. Then follows the files from the library, CSUP is always required, CFIO for file I/O, STRINGS for string routines, followed by C99PFF to insert SLAST, ready for FMSAVE which is a Funnelweb version of SAVE. Finally comes the LAST directive followed by the starting point SAVE which would start the save program. I suggest that you read the Funnelweb documentation on Scriptloader in FWDEC/SCLL to find out more about this process. In particular, the latest version of Funnelweb allows the Scriptload file to be used to control the assembly process as well as the loading process.

Let me finish this first article by giving you a simple program you can try this all out with and then ask me why you cannot get it to work! The simplest program is one which prints out a message of some sort. For my example program I will get it to print out "Hello Geoff". The first thing to do is to go into the Program Editor which is item one on the assembler menu of Funnelweb. Then type in the following lines:

```
/* A first program in c99 by Geoff Trott */
/*                                       */
#include "DSK1.STDIO"
extern printf();
main()
{
  printf("Hello Geoff");
  exit(0);
}
```

This program is then saved using SF as DSK1.FIRSTC;C for example. On exit from the editor, choose the c compiler from the menu (item 4). This will load and ask two questions whose default answers are n for no so that pressing enter twice will then move on to ask for the input file name. Type in DSK1.FIRSTC;C being careful to use upper case. Press enter and then type in the output file name as DSK1.FIRSTC;S. The compiler will then process the input file and the name "main" will appear. If all is well there will be no errors and you will receive some information about the number of lines processed (29 in my case with 16 global symbols and 0 local symbols). Of course, you must have the file named STDIO on the disk in DSK1 for this to happen. If you have a second drive you can change the disk number in the "#include" statement in the program to suit. At the end of the compiler phase, you must press n to return to Funnelweb. Then, if you have version 4.40 of Funnelweb, there are two ways to proceed.

First, enter the program editor again and type in the following commands:

```
AUTO
ASSM
FILE "DSK1.FIRSTC;O"
STOP
FILE "DSK1.CSUP"
FILE "DSK1.CFIO"
FILE "DSK1.PRINTF"
LAST START
```

This should be saved with a SF as DSK1.LOADFC and can then be used to assemble and load the program (as long as CSUP, CFIO and PRINTF are all on the disk in drive 1). To do all this, exit from the editor and choose loaders from the menu (item 3). Then choose Scriptloader (item 5) and enter the file name DSK1.LOADFC. Answer "Y" to the assembler question "Y" or enter to the load question and watch the processing happen. This is quite a bit of hassle to get such a simple program running but it provides a framework for any subsequent program with just simple changes to the names.

If you are not using Funnelweb 4.40, remove the ASSM and STOP commands from the above file and start the assembler by hand. First mark the file DSK1.FIRSTC;S by using FCTN[7] from the Funnelweb menu and then enter the assembler. If you enter the assembler first, you can still mark the file with FCTN[7] but you then need to use the right arrow FCTN[D] to get the marked names to appear. Both source and object file names appear and so you just need to press enter several times and the assembler will run. When it has finished, run Scriptloader as above and answer "N" to the assembler question and the files will just be loaded and started. More next month. O

continued from page 22

```
470 IF I=C THEN A$="CHDATA "&A$ ELSE A$="       "&A$
480 FOR R=0 TO 1 :: PRINT #R:A$: :: NEXT R
490 NEXT I
500 H$=STR$(768+8*C)
510 FOR R=0 TO 1 :: PRINT #R:" ":"* ENTRY POINT FOR PROGRAM":" ":PN$;TAB(8);"LWP
I WS":" " :: NEXT R
520 FOR R=0 TO 1 :: PRINT #R:"* DEFINE CHARACTERS":" ":TAB(8);"LI   R0,"&H$&RPT$
(" ",8-LEN(H$))&"  " :: NEXT R
530 FOR R=0 TO 1 :: PRINT #R:"       LI   R1,CHDATA"
540 H$=STR$(8*(160-C))
550 PRINT #R:"       LI   R2,"&H$&RPT$(" ",10-LEN(H$))
560 PRINT #R:"       BLWP @VMBW"
570 PRINT #R:" ":"* SKIP OVER NEXT DATA":" ":"       B    @SHOWIT":" " :: NEXT R
580 FOR R=0 TO 1 :: PRINT #R:"* DATA FOR CHARACTER DISPLAY":" " :: NEXT R
590 AC$="DSDATA BYTE " :: FOR P=1 TO LEN(W$):: O=ASC(SEG$(W$,P,1)):: IF O<30 THE
N O=O+128
600 AC$=AC$&STR$(O)&"," :: IF P/8=INT(P/8)OR P=LEN(W$)THEN AC$=SEG$(AC$,1,LEN(AC
$)-1)ELSE 620
610 FOR R=0 TO 1 :: PRINT #R:AC$ :: NEXT R :: AC$="       BYTE "
620 NEXT P :: FOR R=0 TO 1 :: PRINT #R:"       EVEN":"" :: NEXT R
630 FOR R=0 TO 1 :: PRINT #R:"* MORE NECESSARY DATA":" ":"DEC32  DATA 32":"WIDTH
  DATA "&STR$(W):"HEIGHT DATA "&STR$(H):" " :: NEXT R
640 FOR R=0 TO 1 :: PRINT #R:"* R0 = SCREEN POSITION TO WRITE":"* R1 = CHARACTER
 TO WRITE"
650 PRINT #R:"* R2 = ADDRESS OF CHARACTER TO":"*       WRITE":" " :: NEXT R
660 FOR R=0 TO 1 :: PRINT #R:"* R3 = WIDTH":"* R4 = HEIGHT":"* R5 = CHARACTER CO
UNTER":" " :: NEXT R
670 FOR R=0 TO 1 :: PRINT #R:"* R6 = COLUMN COUNTER":" " :: NEXT R
680 FOR R=0 TO 1 :: PRINT #R:"* R7 = ROW POSITION":"* R8 = COL POSITION":"* R9 =
 32 MULTIPLIER":"* R10 = SCREEN POSITION":" " :: NEXT R
690 FOR R=0 TO 1 :: PRINT #R:"* CALCULATE SIZE OF INSTANCE":" ":"SHOWIT MOV  @WI
DTH,R3"
700 PRINT #R:"       MOV  @HEIGHT,R4":"       MPY R3,R4":" " :: NEXT R
710 FOR R=0 TO 1 :: PRINT #R:"* GET ROW FROM XB":" ":"       CLR  R0":"       LI
   R1,1":"       BLWP @NUMREF":" " :: NEXT R
720 FOR R=0 TO 1 :: PRINT #R:"       BLWP @XMLLNK":"       DATA CFI":" ":"
 MOV  @FAC,R7":" " :: NEXT R
730 FOR R=0 TO 1 :: PRINT #R:"* GET COL FROM XB":" ":"       CLR  R0":"       LI
   R1,2":"       BLWP @NUMREF":" " :: NEXT R
740 FOR R=0 TO 1 :: PRINT #R:"       BLWP @XMLLNK":"       DATA CFI":" ":"
 MOV  @FAC,R8":" " :: NEXT R
750 FOR R=0 TO 1 :: PRINT #R:"* CALCULATE SCREEN POSITION":" ":"       DEC  R7":
"       DEC  R8":"       MOV  @DEC32,R9"
760 PRINT #R:"       MPY  R7,R9":"       A    R8,R10":"       INCT R10":"
MOV  R10,R0":" " :: NEXT R
770 FOR R=0 TO 1 :: PRINT #R:"* START AT FIRST COLUMN":" ":"       CLR  R6":" "
:: NEXT R
780 FOR R=0 TO 1 :: PRINT #R:"* SET ADDRESS OF TEXT":" ":"       LI   R2,DSDATA"
:" " :: NEXT R
790 FOR R=0 TO 1 :: PRINT #R:"* SET COLUMN COUNTER TO ZERO":" ":"       CLR  R6"
:" " :: NEXT R
800 FOR R=0 TO 1 :: PRINT #R:"* PUT CHARACTER IN R1 AND THEN":"*  PRINT IT":" "
:: NEXT R
810 FOR R=0 TO 1 :: PRINT #R:"PRINT  MOVB *R2+,R1":"       AI   R1,>6000":"
  BLWP @VSWB":" " :: NEXT R
820 FOR R=0 TO 1 :: PRINT #R:"       INC  R6":"       C    R6,R3":"       JNE  A
GAIN":"      AI    R0,32"
830 PRINT #R:"       S    @WIDTH,R0":"       LI   R6,0":" " :: NEXT R
840 FOR R=0 TO 1 :: PRINT #R:"* PREPARE TO DO IT AGAIN":" ":"AGAIN  INC  R0":"
     DEC  R5"
850 PRINT #R:"       CI   R5,0":"       JNE  PRINT":" " :: NEXT R
860 FOR R=0 TO 1 :: PRINT #R:"* RETURN TO XB":" ":"RETURN LWPI GPLWS":"       B
   @BASIC":" " :: NEXT R
870 FOR R=0 TO 1 :: PRINT #R:"* END OF SOURCE CODE":" ":"       END":" " :: NEXT
 R
880 STOP
890 PRINT "INSTANCE TOO LARGE:":" ":"SORRY ABOUT THAT!" :: STOP
```

O

continued from page 10

condensed to 3 lines if written in Extended BASIC. It will print out a sequential line number and the associated records line by line. It will also run in conjunction with an 80 column printer. In this case the line number will be on its own followed by the record printed on the next line. O

Check your renewal date. Do not miss out.

TI-Bits Number 15

by Jim Swedlow, CA USA

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

TIGERCUB

I do not usually promote commercial software in this column but this time I am making an exception. Jim Peterson has been a major contributor to the TI community over the years. His material is first rate. Jim has helped many 4A owners (including yours truly).

There are 130 programs in the Tigercub catalogue for $1.00 each (plus $1.50 per order for media). Catalogues are $1, which is refundable from the first order.

There are four disks in the Tips from Tigercub series at $10 each. The three Nuts and Bolts are $15 each. These prices are postpaid.

All of these disks are full of ideas, programs, suggestions and tutorials. You will NOT be disappointed. Novice or experienced, his programs will be of use.

He can be reached at:

Tigercub Software
156 Collingwood Avenue
Columbus, OH 43213

AVATEX 1200hc MODEM

I picked up one of these 1200 baud modems. It is a good modem that works well with the TI. The cable is straight through with pins 2 and 3 switched:

```
TI--MODEM

 1-----1
 2-----3
 3-----2
 5-----5
 6-----6
 7-----7
 8-----8
20----20
```

Set the DIP switches as suggested in the manual.

This information applies only to the 1200hc. Other Avatex modems require different cables.

OUCH!

Every once in a while I goof and someone catches me. This time it was in TI BITS Number 14 in which I said that you cannot chain Include File (.IF) commands in TI Writer.

That is what the manual says and I believed the written word. Not so, says UGOC Pres Bob Harper. He is correct. You can end each file with a .IF command for the next file. I still prefer, however, a master file that has all of the .IF commands.

FUNNELWEB Version 4.10 is out. It has some nice new features. The configuration program has been completely rewritten. It uses overlapping windows (just like that computer we do not talk about). You can change more items including four menu choices on each main menu (in version 4.0 you could only change one each).

The McGoverns have added some new keys to the TI Writer Editor. <CTRL Q> and <CTRL A> perform a ROLL UP and ROLL DOWN. <CTRL ;> changes whatever is under the cursor to lower case and <CTRL :> to uppercase.

If you start typing with ALPHA LOCK on, just move your cursor back to the first letter and press <CTRL ;>. The letter will be changed to lower case. Better yet, it is a repeater, so if you keep those keys down, everything you typed will be converted. Very nice.

TELCO 2.1 is also out. Among other things, it now supports Compuserve B transfer protocol and most parallel and serial printer ports. One feature (not new) that it has is the ability to send an initialization string to your modem. The distribution version has this string:

ATS0=0!

This turns off your modem's auto answer feature.

The above string and the following information apply to Hayes modems and true compatibles. Check your modem's handbook before doing anything.

You may want to add to this initialization string. Here is what I use:

ATS0=0 S7=60 S11=55 V1 X4!

What does all this mean? Glad you asked.

AT tells your modem that you want to talk to it (as in ATtention).

S0=0 turns off auto answer. Very handy if you have call waiting.

S7=60 changes how long your modem waits for a carrier from the other modem (in seconds). The default is 30 seconds.

S11=55 sets the time between tones when you are dialling (in milliseconds). The default is 70 so this is faster.

V1 tells your modem to display its messages in words (rather than code numbers).

X4 tells your modem to use its full message set rather than an abbreviated one.

! is a carriage return. This tells the modem that it is the end of the command string.

You may find that you needs vary (these work with my Avatex 1200hc) but there are things you can do to control your modem. Experiment.

FILE NAMES: The following applies to TI Controllers ONLY. No guarantees about MYARC or CORCOMP.

The Disk Controller book says that TI file names can contain any character between ASC 32 and 95 except space and period. Having seen other characters used, I decided to test this. I wrote a simple program to open a file, print something, close the file, open it again, read the text, close it and then delete the file:

```
100 FOR I=0 TO 255 :: ON ERROR 190
110 OPEN #1:"DSK1."&CHR$(I)
120 PRINT #1:STR$(I)
130 CLOSE #1
140 OPEN #1:"DSK1."&CHR$(I)
150 INPUT #1:A$
160 IF A$<>STR$(I) THEN PRINT
    "BAD READ IN";I
170 CLOSE #1:DELETE
180 NEXT I :: STOP
190 ON ERROR 210
200 CLOSE #1:DELETE
210 PRINT "FILE ERROR IN";I
220 RETURN 180
```

continued on page 19

Renew Now!

XB tips Number 16

by Jim Swedlow, CA USA

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

ON BACK-UPS AND FLIPPIES

If you already back-up your disks faithfully and have decided about flippies, skip this. Otherwise, read on.

BACK-UP's are essential. The first thing you do when you get a new program or disk is back it up. Do not run it, do not modify it, do not catalog it, do not list it - back it up.

Why? Simply put, disks go bad and disk drives eat disks. If your only copy goes bad, it could take days to weeks to get a replacement, depending on the source. Ever read the 'warranty' that comes with some software?

So, make your back-up first. If you buy a program, keep the master (with the maker's label) with your back-ups and use a working copy for every day.

Keep your back-ups and masters in a separate disk box away from your computer. That way you will not use them by mistake.

If a disk does go bad, make sure that your hardware is working by using another disk BEFORE using your master or back-up. Otherwise, you could destroy both copies!

Updating back-ups is vital. If you wrote the program, you probably will revise it more than once. If you get a single program, you will normally add it to an exiting disk. If you do not update your back-up, you will lose your new program if something goes wrong with your working disk.

A FLIPPY is a single sided disk that has been modified to act like two single sided disks. By adding three holes, you can put your disk in your drive upside down and record on the back. Instructions for making flippies can be found in the August, 1984 ROM.

Some folks, like Craig Miller, recommend against flippies. They argue that a disk is designed to turn one way and bad things happen when you flip it over and make it turn the opposite direction.

Others use flippies and claim that they have had no problems. One approach is to use the front as a working copy of one disk and the back as the back-up of another disk.

I compromise by only using flippies for back-up copies, thus reducing the number of back-up disks I need. I do not, however, use them for working disks. I have not had any problems.

DISK COPIERS

There are two types of disk copiers: file-by-file and sector-by-sector. You need both.

FILE-BY-FILE copiers read your disk one file at a time and then write that file to the new disk. DISK MANAGER II, the module that came with the TI disk controller card, is a file copier that takes about three weeks to back-up a disk.

A much better file copier is DISK MANAGER 1000. This freeware item includes a sector copier and other features. It makes DISK MANAGER II obsolete.

SECTOR-BY-SECTOR copiers copy your disk without regard to the file content. Instead, they read the master sector by sector and then write that information to the new disk, sector by sector.

Some sector copiers allow you to use or ignore the bit map. This is a table in sector zero that tells the disk controller which sectors are used and which are free. Using the bit map shortens copy time.

There are many good sector copiers. Freeware items include DISK MANAGER 1000 and MSCOPY. You can buy NIBBLER and TURBO COPY. TURBO COPY is the fastest copier I have found.

WHY BOTH? While sector copiers are faster, file copiers are more versatile and can repair some disk problems.

If the bit map is bad, copying the disk with a file copier will result in a good bit map on the new disk.

Another problem is a result of the way the disk controller saves files on a disk. Say that you are writing a letter and save it to disk three times: after a third is written, after two-thirds and when you are done. Each time the file will be larger.

If there is not enough space to write the new, larger file where the old, smaller file had been, the disk controller will save what it can where the file had been and the rest on another part of the disk.

The result is what Craig Miller calls a 'fractured file'. It takes longer to read a fractured file because the disk drive head must jump around more. This increases the probability of a read failure and of harm to your disk.

Millers Graphics' ADVANCED DIAGNOSTICS will tell you if you have fractured files. If you have done a lot of saving and changing on a disk, the odds are that you do.

The only way to unfracture a file is to copy your disk with a file copier.

AVAILABILITY: Both MSCOPY and DISK MANAGER 1000 are in our library as freeware.

Send freeware authors the price they request. Freeware is the best source of new TI products right now. It will dry up, however, without our support.

ASC AND SEG$

A common coding for determining the ASCII value of the first character of a string is:

```
ASC(SEG$(A$,1,1))
```

This can be simplified by omitting the SEG$ function:

```
ASC(A$)
```

ASC always returns the ASCII value of the first character of the string. You will get an error if A$ is a null string (if A$="").

Suppose you want to lop off the first four characters of a string. You might do this:

```
A$=SEG$(A$,5,LEN(A$)-4)
```

Our 4A's, however, do not compare the third value in the SEG$ function (the length of the new string) to the length of the old string. Therefore, this works just as well:

```
A$=SEG$(A$,5,255)
```

Use 255 as that is the maximum length of a string variable. If the length of A$ is less than 5 - even if it is zero the new A$ will be a null string (but NO error).

continued on page 4

Contributions to TND

by J.E. Banfield

Because my TI99/4A is in use for up to 14 hours per day testing printed circuit boards (another story) and since my disk format may not be compatible with the group's systems, I must rely on the good services of Geoff Trott to find some volunteer to type up and edit my contributions to TIsHUG from the crude typescript supplied.

Since readers (including myself) tire rapidly of long articles, I propose several sets of contributions each in serial form:

A. Why I write in machine code and how others can do the same.

B. How to cope with the TI99/4A system including the GPL code and DSR interfacing.

C. Description of the PE box which I constructed and of the cards which reside in it.

D. Machine code subroutines to service the peripherals, together with logic diagrams for the same.

Why I write in Machine Code

by J.E. Banfield

This article summarises my experiences with computing.

In about 1960, our University (New England) was considering the purchase of its first (mainframe) computer. One possibility was the Ferranti Sirrius. Ferranti Ltd., a British firm later taken over by ICT and then by ICL, had built, in conjunction with Manchester University, the Atlas machine, then the most powerful computer of the day. A high level language, the Autocode, was available to users.

We were considering the smallest machine of the range, the Sirrius. The memory (dynamic) consisted of coils of nickel wire housed in drawers of the console. Piezomagnetic transducers (Nickel is piezomagnetic) pulsed ultrasonic waves with digital codes into one end of the coil to be received at the other end and re-transmitted with or without modification. Amazing but it worked. (Another early machine, the CSIRAC which was Melbourne's first computer, used a similar system with mercury liquid delay lines.)

Two Ferranti sales representatives came to Armidale to give a two day course on the Ferranti Sirrius Autocode. The course was brilliant. One representative, Dr. Cliff Bellamy, ultimately became the first Professor of Computing Science at Monash University and he maintained a Sirrius in his Department until at least the mid 1970's. The first day of the course dealt with the Autocode language; more of that later. In the second day we wrote our trial programs connected with our research work and sent them to Ferranti Ltd. in Sydney to be run free of charge! It is a tribute to the Autocode and to the course that our (at least my) programs ran perfectly first go. In fact my program formed the basis of a scientific paper printed in the Journal of the Chemical Society (London) in 1964. You cannot do that with modern high level languages.

The Sirrius Autocode language was elegantly simple, the definitive manual ("Ferranti Sirrius Computer, Description of the Autocode", Ferranti Ltd., June 1963) sets forth the descriptions, rules and syntax in only 13 pages of open typescript. Each statement (line), which may or may not be numbered (with a 5) for example), defines a single operation. For example

```
n1 = n1 + n2.
```

Variables were integers, n1 etc., or floating point numbers, v27 or v(n5). The only difficult thing was formatting numbers for output, for example:

```
PRINT v3,4025
```

defining digits before and after the decimal point in a table. There were no DO (FOR ... NEXT) loops. You set an integer as a counter, say

```
n5 = 7
```

each loop decrements the counter,

```
n5 = n5 - 1
```

and then executes a conditional jump:

```
-> 3, n5>1
```

3 referring to a labeled statement. Only carelessness caused a program to jam; usually in an endless loop.

Modern programmers think of Autocode as akin to machine code. Certainly you must break each procedure into simple steps for both. There is no such thing as a free lunch; which I was soon to discover.

When the University of New England eventually bought a computer, it was an IBM 1620 Model 2 for which the normal coding was a primitive version of FORTRAN, said to make translation of complex mathematical formulae simple and easy. Ha! The version of FORTRAN supplied had a rigid and inconvenient input format in contrast to the free format input of the Autocode. The same program that I wrote on my second day of Autocode took several weeks of frustration to get up and running. Maybe I do not like FORTRAN or vice versa. I soon decided to learn Assembler, only to find that I had to understand machine code first; so why not cut out the middle man? Why not?

Within three days I had written and debugged my first simple machine code program: it took free format input of Autocode format and translated it into the rigid input required by the stock 1620 programs; for example the Eigenvalue Program for Real Symmetric Matrices, which I needed to use. (I later wrote my own in machine code.)

Other machine code programs followed. Debugging sessions at the console (you cannot do that with modern mainframes) were greatly facilitated by using and knowing machine code. I was hooked.

By about 1971, IBM had given notice that it would no longer support the 1620, at least in this country (I have heard this story recently, was it on October 28th 1983?) and so the University of New England bought an ICL mainframe. However I was able to retain the use of the 1620 until 1985 by taking over the maintenance myself. Use of machine code was of great help in locating crook circuit boards.

One of my first tasks on taking over the care of the 1620 was to interface a Data Products line printer to replace the IBM printer previously on hire. This introduced me to 74 series TTL and again machine code was vital in the interfacing.

Eventually, when shortage of space finally put an end to the 1620, I had to use the DEC 2020 system and I found that the assembler, Macro 10, was excellent. I temporarily abandoned machine code until the TI99/4A.

I bought the TI99/4A mainly because it was cheap ($180 new at the time) because I wanted an intelligent keyboard for control of a long word length computer that I was designing and building. I tried every trick that I knew and some others beside to hack the system and enter and run machine code from BASIC without success; the GPL language and GROM system saw to that. I did not and still do not have an Extended BASIC module. Finally a friend lent me his MiniMemory module and I was off. O

continued from page 6

MAX-RLE is a program which allows the viewing of RLE (Run Length Encoded) graphics in DIS/FIX 128 or DIS/VAR 80 format, and their conversion into the GraphX or TI-Artist format. The program is fully described elsewhere in this issue of the TND.

INSTOLINK is a TI-Artist Instance to CALL LINK Compiler by Bud Wright which allows the conversion of Instances to runable assembly source code. To use in your Extended BASIC program, just load the object code (after it has been compiled and assembled) and do CALL LINKs to see your loaded screens. Very fast displays. O

Using the Arrow Keys

in Extended BASIC Programs

by Bob Relyea and Ross Mudie

I (Bob) would like to describe another phase of the on-going saga of adding to my Grade Standardising program. This is the program that inputs a student's name and their grade on a particular assignment/test. The program then totals it all up with the mean (average) and everything and then standardises the marks to whatever mean and standard deviation that you require. The original version requires that input be done as a 'single entry', that is, only one name and one mark be seen on the screen at any one time. I wanted to develop a method of using a full page display so I could see the whole class of names and marks at once and then move the cursor up and down the screen, if required, to make any corrections or additions. With the help of Ross (the genius!) Mudie with the 'arrow' part of it I finally managed to get a satisfactory result. This article will be an attempt to explain how it was all done and we hope that you pick up a few programming clues in the process- I did!

The program that follows all of this is an extract from the much larger program, but it contains the entire section under discussion and is a program in its own right. It has been saved to disk (List "DSK1.STD") from a working program so you will find it to be accurate. I will go down through it line by line explaining anything off the beaten path, and for the sake of the beginner I will explain some of the easier parts as well.

The first part that I want to comment on is line 850 of the program. There looks like a lot of 'wasted' spaces being used. Actually it is to make 'up' and 'down' symbols to indicate up and down keys. When that line comes up in the program you see:

Up ↑ and Down ↓ keys are active.

It makes the presentation more interesting!

The next point of interest is line 920. I wanted my program set up so that after telling the computer in line 830 how many marks I intended to enter that it would automatically number the 'page' from 1 to N and put a full stop after each one. I also wanted the numbers after 9 to be 'outdented' one space so that the numbering would look better. To achieve all this I used the STR$ function which changes numbers (1 to N) and uses them as if they were letters. I called these numbers-converted-to-letters NUM$ and I used the concatenation '&' to include the '.' after each number. Then in lines 930 to 950 are the instructions to print the first 9 numbers down the screen. Notice in line 940 how I had the program do the listing. I used the DISPLAY AT(ROW,COLUMN) command as this avoids the scrolling that you get with the PRINT statement. For the specified ROW I put in I+3. I already had used the first three lines with titles, etc so I wanted it to start numbering from row 4. Since the first I is 1 then I+3 equals 4 and it starts numbering right where I want it to. Now notice in line 930 I give instructions to number from 1 to 9 in this way. If my N is greater than 9 then it skips to line 970 and continues the same way except for one little change. Notice in line 990 that I change the COLUMN number from 2 to 1. This enables all numbers from 10 onwards which are all TWO digits to start numbering from one position further to the left so that the right sides of the numbers all line up with the ones from 1 to 9 which are all one digit. Pretty neat, huh?

The purpose of the L=2 in line 1010 will be elaborated on later but if you note lines 1030 and 1050, the ACCEPT AT(L+1,4) places the cursor bang on the 3rd Row which is where "1." was written. Everytime the programs goes back to line 1020 to run it all through again 'L' is increased by one (L=L+1) so the cursor automatically goes to the next line for input, and so on.

You can see from lines 1030 and 1050 that there are two bits of information that need to be inputted into each row, the student's name, SN$(I), and the mark M(I). That they are on the same row can be seen from the L+1 in each row position in the ACCEPT AT statement. The two column positions that the cursor positions itself at to accept this information are column 4 and column 25. I wanted to be able to use the up and down arrow keys so that I could move up and down the screen and make any corrections. This is where Ross helped me out. After lines 1030 and 1050 I have a CALL KEY statement followed by- If K=11. The eleven (11) is the code for the UP arrow key. Keep in mind that the first input is on column 4 and the second one on column 25. Line 1030 has you on column 4 so to go 'back' from here means to go to the previous row, column 25. Therefore, line 1040 also has L=L-1 to go back to the previous row and it also has instructions to go to line 1050 which places the cursor right where you want it! Now, following line 1050 is another CALL KEY statement following the routine outlines about but it is not followed by any L=L-1. This is because line 1050 has you on column 25 and to go 'back' from here means to go to column 4 of same line. Hence, the row number is the same so there is no need to have a L=L-1 which goes to the previous row. Now, there is one point that I did not explain about line 1040- the I=I-1. As is typical in any program, line 1020 has FOR I=1 to N coupled with a Next I in line 1080, so the computer recycles through it all automatically until it is all finished. Everytime you push ENTER after the end of each row (that is, after column 25's input), one 'I' is marked off the computers counter until you reach N, and then it goes on to the next part of the program, which in this case is line 1090. The problem arises when you use the arrow key to go back to previous rows and then use the DOWN arrow key or the ENTER key to return to where you were. On the way back every ENTER that you push tallies up another 'I' on the computers counter. So, if you then continue to input data you will find that your program will stop a line or two short, depending on how many lines you had to back up to make your correction. To make a long story short, it also means that you cannot input all the data that you wanted because there is no way that you can get the computer to continue on accepting data once you have reached 'N'. To counter this you must included a I=I-1 to take the computers tally back one 'I' everytime you back up a row so that when you press ENTER to go back down the screen the 'count' is back on schedule. I found out about this the hard way!

Now, why was the L=2 used in line 1010. It is tied up with the problem I had of working out what to do when I had a class list of names that exceeded the number of rows on a single screen. I wanted the program to accept one page of data at a time and when the bottom line was finished on the first page it would automatically erase page one, number the second page starting from the top with the next number of the list. The only way you can do this that I know of is to select a variable that is not being used, such as L, and let everything be based on it. When you get within a line of the bottom of the page, the program automatically pauses and displays "Press any key to continue..." on the bottom. How do you get it to stop? Simple! Just look at line 1070 and you see an interesting use of the INT function. INT stands for integer and means 'exact' number, that is, without a fraction. So, when I gets up to 20, I/20 would be 20/20 and works out to be an integer so the program pauses as instructed. After you make a key press the program goes to line 1108 and repeats it all for the next 20 numbers with the line numbering starting from the top because of the L=2 feature.

```
800 CALL CLEAR
810 INPUT "FILENAME? ":FN$
820 CALL CLEAR
830 INPUT "HOW MANY MARKS DO YOU WISH TO ENTER? ":N
840 CALL CLEAR
850 DISPLAY AT(3,1):"    ^          |          Up | and Down V
Keys are          active"
860 DISPLAY AT(15,1):"Press any key to continue.."
870 CALL KEY(0,K,S):: IF S=0 THEN 870
880 CALL CLEAR
```

continued on page 20

Reformatting

by Jim Peterson, Tigercub Software, USA

With the establishment of the Clearinghouse BBS, newsletter editors will have available more articles on disk, rather than having to photocopy them or retype them from other newsletters. This will make it easier for them to reformat articles to their own requirements but they will have to know how to do so.

I am by no means an expert on this subject, but I will offer a few ideas. It seems to me that the most practical column widths are 28, 40, 60 and 80. (We use a column width of 0 to 55, or 56 characters in the TND, Ed.)

I have always believed that BASIC and Extended BASIC program listings should be published in 28-column width, exactly as they appear on the screen. This makes it much easier to key them in accurately, especially when the listing contains strings of blank spaces or long strings of hex codes. For that reason, years ago I wrote a program to reformat listed programs accurately into 28-column width. Nowadays, the Super Extended BASIC module will do that much more easily. Since my Tips From The Tigercub newsletters consisted mostly of program listings, I always published them in four columns of 28-character width. However, that is too narrow for primarily text articles, requiring too much hyphenation or creating too many gaps.

The 40-column width is perhaps best of all, because it can be printed in two columns of elite font or three columns of condensed font. The 60-column width can be printed in two columns of condensed font.

The 80-column width is suitable for regular D/V 80 files in pica font, but this is somewhat wasteful of space; pica is considerably larger than most printed material. It is possible to write routines to reformat and print D/V 80 text in even longer lines, up to 160 characters long in Epson condensed elite, but lines of more than 80 characters are difficult to scan.

Page Pro printing may require other widths, but I know nothing about that; I really do not consider the oversized crowded characters of Page Pro to be suitable for newsletters.

The first step in reformatting should be to separate any program listing portions from the rest of the text and reformat them separately, if at all. For instance, if you load the article THISN-THAT into the Funnelweb Editor and find that lines 200 to 300 of a 400-line article are a program listing, do this:-

```
FCTN 9, SF, 200 300 DSK1.PROGRAM then
FCTN 9, LF, 1 199 DSK1.THISN-THAT then
FCTN 9, LF, E 301 400 DSK1.THISN-THAT and
FCTN 9, SF, DSK1.TEXT.
```

An alternative way of achieving the same effect is to first save the program listing, delete those lines from the buffer and save the result. For example:

```
FCTN 9, SF, 200 300 DSK1.PROGRAM then
FCTN 9, D, 200 300 then
FCTN 9, SF, DSK1.TEXT.
```

The next step is to make sure that the title and each paragraph, as well as any line that nothing should be added onto, ends in a carriage return. The carriage return (CR, ASCII 13) in the Funnelweb Editor looks like a little square C above and to the left of a little upside down L.

If the CRs are missing and you have a text file with indented paragraphs and centered headers (blank spaces before the title, etc.), this tinygram program will add the carriage returns.

```
100 DISPLAY AT(3,4)ERASE ALL
:"CARRIAGE RETURN ADDER":"":
"  This tinygram program wil
l add carriage returns to any
 text file which has centere
d"
110 DISPLAY AT(8,1):"headers
and indented para    graphs.
"
120 DISPLAY AT(12,1):"Input
filename?":"DSK" :: ACCEPT A
T(13,4):IF$
130 DISPLAY AT(15,1):"Output
 filename?":"DSK" :: ACCEPT
AT(16,4):OF$
140 DISPLAY AT(18,1):"Put bl
ank lines between    paragr
aphs? Y/N" :: ACCEPT AT(19,1
7)SIZE(1)VALIDATE("YNyn"):Q$
150 OPEN #1:"DSK"&IF$,INPUT
:: OPEN #2:"DSK"&OF$,OUTPUT
:: C$=CHR$(13)
160 IF EOF(1)THEN 190 :: LIN
PUT #1:M$ :: IF Q$="Y" OR Q$
="y" THEN 180
170 IF M$="" THEN PRINT #2:C
$:M$;:: GOTO 160 ELSE IF ASC
(M$)<33 THEN PRINT #2:C$:M$;
:: GOTO 160 ELSE PRINT #2:""
:M$;:: GOTO 160
180 IF M$="" OR M$=" " THEN
PRINT #2:C$ :: GOTO 160 ELSE
 IF ASC(M$)<33 THEN PRINT #2
:C$:C$:M$;:: GOTO 160 ELSE P
RINT #2:"":M$;:: GOTO 160
190 PRINT #2:CHR$(13):: CLOS
E #1 :: CLOSE #2
```

Another way to add carriage returns is to type CTRL U to get the underline cursor, then go through the text with FCTN 4 and FCTN 6 and the arrow keys, typing SHIFT M wherever you need a carriage return. Now, here is a tip:

FCTN 9, S, 1 to get to the beginning of the file, then FCTN 9, RS, / CTRL U SHIFT M CTRL U / FCTN W CTRL U SHIFT M CTRL U / (slash, carriage return, slash, tilde, carriage return, slash), to replace each carriage return with a tilde followed by a carriage return. Why? We will get to that! Hit Enter, then A for All.

In order to use the Funnelweb Formatter to reformat a file, it is necessary to print it back to disk - and when you do that, the Formatter can play some very nasty tricks! Any & symbol in the text will simply disappear! An @ in the text will disappear but cause the word following it to be repeated again and again on following lines. An asterisk (SHIFT 8) followed by two or more digits will disappear, along with the first two digits. A caret sign (SHIFT 6) becomes a space, while a full stop at the beginning of a line will cause the entire line to disappear! (and since this article now contains a & and a @, you have problems already!)

So, to be on the safe side, get back to the top of your text, go into RS again with /&/\/ (the \ symbol is FCTN Z). If you jump to the end of the file, there were no ampersands, so you will not have to restore them after reformatting. You might do the same thing with the @, just in case. The asterisk bug is very unlikely to occur in a text file, and the problem with carets is also unlikely (except in this article!), but you might scan down through the first character of each line to be sure that a decimal number in the text has not placed a period there. If so, the best solution is to insert a 0 before the period.

There is a better way to prevent the Formatter from garbling your text. I have never understood why Texas Instruments used characters that might appear in text as control characters, when other useless characters were available - and I have often wondered why the McGoverns did not do something about it. But you can, if you have John Birdwell's Disk Utilities (DSKU). Copy your

Funnelweb to a new disk, just in case. Boot up DSKU and select 1. File utilities, then 5. Find string. The filename is FO. Select H for hex. At the prompt for a string, type 2A23214026 and for replacement string type 7C2321605C. Select R for replace, then FCTN W, hit Enter twice to accept defaults and it is done. From now on, if you want to underline a word, use FCTN Z instead of &, if you want to double strike a word use FCTN C instead of @ and if you want to input a value from a data file use FCTN A instead of an asterisk. You will still have to watch out for those periods and the caret sign - I do not know the fix for those.

Now, to get Funnelweb to reformat the text, open a new line 1 with FCTN 8 and type in .LM0;RM79;IN0;FI<cr> where <cr> is a carriage return (CTRL 8). The 79 would give you 80-column text; you can substitute any number, 1 less than the actual column width you want (because the computer is counting from a left margin of 0, not 1). If you wanted a pre-set left margin you could change that 0 and adjust the RM figure accordingly. If the text does not have paragraph indentations and you want to add them, change the 0 after IN to whatever number of spaces you want to indent; you can also increase the amount of indentation that way. And, if you want Funnelweb to insert additional blank spaces in the text in order to justify the right margin (line it up evenly), add ;AD after the FI. Now save the file to disk, then go to the Funnelweb Formatter.

Accept that same default filename; instead of the printer option, enter 'DSK1.' and a different filename, accept all the defaults and the file will be printed back to the disk under that new filename.

Return to the Editor and load that new filename. You will find that your text has been reformatted to the desired width, but every line now ends in a line feed (a little L to the upper left of a little F); your carriage returns have also been converted to line feeds. And there are now three lines at the top containing nothing but a blank followed by a line feed, groups of similar lines throughout the text, probably a long series of such lines at the end and sometimes a few lines in the text containing a few dashes followed by a line feed. Some of them also contain a form feed! You will have to go through the text with FCTN 4, FCTN 6 and the arrow keys, deleting those lines with FCTN 3.

Now, to get rid of those line feeds - FCTN 9, PF, C DSK1.newfilename. It is best to always use a new filename for each step in the process so that if you make a mistake - as you will occasionally! - you do not have to go all the way back to start over.

Load that new file and you will find that printing to disk with the C option has apparently stripped out all the line feeds. Actually, it changed them to the space character (ASCII 32), which could cause problem in multiple-column printing or concantenation. If you want to get rid of them, SF the file to disk, LF it back and PF it to get rid of the tab line.

You might find it easier to just use this handy dandy routine to delete the line feed lines and line feeds -

```
100 DISPLAY AT(12,1)ERASE AL
L:"Input filename?":"DSK" ::
 ACCEPT AT(13,4):IF$ :: OPEN
 #1:"DSK"&IF$
110 DISPLAY AT(15,1):"Output
 filename?":"DSK" :: ACCEPT
AT(16,4):OF$ :: OPEN #2:"DSK
"&OF$
120 A$=CHR$(32)&CHR$(10):: B
$=CHR$(12)&CHR$(10)
130 IF EOF(1)THEN 150 :: LIN
PUT #1:M$
140 IF M$=A$ OR M$=B$ THEN 1
30 ELSE IF SEG$(M$,LEN(M$),1
)=CHR$(10)THEN PRINT #2:SEG$
(M$,1,LEN(M$)-1):: GOTO 130
ELSE PRINT #2:M$ :: GOTO 130
150 CLOSE #1 :: CLOSE #2
```

But, now the carriage returns you worked so hard to put in have also disappeared! Not to worry. FCTN 9, S, 1, FCTN 9, RS, / FCTN W / CTRL U SHIFT M CTRL U /, Enter, A for All and those tildes will magically turn into carriage returns! Occasionally one will appear at the beginning of a blank line instead of at the end of the preceding line, in which case you will have to delete it and put it where it belongs with CTRL U SHIFT M. Why did I use FCTN W, the tilde? Just because I have never seen it used for anything else in a text file (oops! except in this article!) - I could have used FCTN A, FCTN Z, FCTN C, etc. In fact, if you want to preserve any CTRL U type printer codes in the text, you could RS them back and forth in the same way. After CTRL U to get the underline cursor, FCTN R is the ASCII 27 escape code, SHIFT 2 is 0 and SHIFT A is 1, etc.

Rather than go through all that, maybe you would rather just throw the Funnelweb Formatter away and use my little handy dandy Text Reformatter in good old primitive Extended BASIC. It's slow, but it avoids all those extra steps and all those pitfalls. The text must have carriage returns.

```
100 CALL CLEAR :: CALL SCREE
N(5):: FOR SET=0 TO 12 :: CA
LL COLOR(SET,2,16):: NEXT SE
T :: CR$=CHR$(13)
110 DISPLAY AT(2,7):"TEXT RE
FORMATTER"":" by Jim
 Peterson"
120 DISPLAY AT(6,1):"Input f
ilename?":"DSK" :: ACCEPT AT
(7,4)BEEP:IF$ :: OPEN #1:"DS
K"&IF$,INPUT
130 DISPLAY AT(8,1):"Output
filename?":"DSK" :: ACCEPT A
T(9,4)BEEP:OF$ :: OPEN #2:"D
SK"&OF$,OUTPUT
140 DISPLAY AT(11,1):"Presen
t line length?" :: ACCEPT AT
(11,22)SIZE(2)VALIDATE(DIGIT
):LL
150 DISPLAY AT(13,1):"Reform
at to what length?" :: ACCEP
T AT(13,26)SIZE(2)VALIDATE(D
IGIT):R
160 IF R=LL THEN 140 ELSE CA
LL CLEAR
170 IF EOF(1)THEN 290 :: LIN
PUT #1:M$ :: M$=P$&M$ :: P$=
"" :: IF R>LL THEN 230
180 L=LEN(M$)+(POS(M$,CR$,1)
<>0):: IF L<=R AND POS(M$,CR
$,1)<>0 THEN PRINT #2:M$ ::
GOTO 170 ELSE IF L<R THEN P$
=M$&" " :: GOTO 170
190 C$=SEG$(M$,1,R):: CALL L
ASTPOS(C$," ",P)
200 IF P<>0 THEN 210 ELSE PR
INT #2:C$ :: M$=SEG$(M$,R+1,
255):: GOTO 180
210 GOSUB 300 :: GOTO 180
220 GOSUB 310 :: GOTO 180
230 IF POS(M$,CR$,1)<>0 AND
LEN(M$)<=R+1 THEN PRINT #2:M
$ :: GOTO 170
240 IF LEN(M$)<R THEN P$=M$&
" " :: GOTO 170
250 C$=SEG$(M$,1,R):: CALL L
ASTPOS(C$," ",P):: IF P=0 TH
EN PRINT #2:C$ :: M$=SEG$(M$
,R+1,255):: GOTO 230
260 IF P=R THEN PRINT #2:SEG
$(M$,1,P-1):: M$=SEG$(M$,R+1
,255):: GOTO 230
270 GOSUB 300 :: GOTO 230
280 GOSUB 310 :: GOTO 230
290 PRINT #2:P$ :: CLOSE #1
:: CLOSE #2 :: STOP
300 IF SEG$(M$,R+1,1)=" " TH
EN PRINT #2:SEG$(M$,1,R):: M
$=SEG$(M$,R+2,155):: RETURN
```

```
310 PRINT #2:SEG$(M$,1,P-1):
: M$=SEG$(M$,P+1,255):: RETU
RN
320 SUB LASTPOS(A$,B$,Y):: X
,Y=0
330 X=POS(A$,B$,X+1):: IF X>
0 THEN Y=X :: GOTO 330
340 SUBEND
```

I have also written a Formatter+ program which will even reformat text which does not have carriage returns if header lines and paragraphs are idented. It will also optionally allow you to hyphenate any word which breaks after the second character and optionally right justifies the text. It is too long to list here, but will be available on the Clearinghouse BBS and in my TI-PD catalog.

The next-to-last thing to do when reformatting (but, from my reading of many newsletters, often omitted!) is FCTN 9, S, 1, FCTN 9, RS - // (replace a hyphen followed by a blank with a null, to fix hyphenated words that have ended up in the middle of a line). Do not, repeat do not do this until you have restored the carriage returns!! Enter, but this time to do not use A for All; use Y(es) and N(o) to go through the text, deleting unwanted hyphens but leaving those at ends of lines or elsewhere if they belong.

And, last of all, PLEASE, PF rather than SF, to print the file back to disk rather than saving it back, to get rid of that pestiferous tab line!

Now, as to reformatting program listings - preferably, do not! A 28-column listing combined with 40-column text is not going to waste much space.

Remember that program listings are printed so that people can key them in and run them and the least mistake in reformatting them will usually result in garbage.

Assembly source code and c99 source code (and probably most any other language) MUST NOT be reformatted! However, anything beyond the 25th character of assembly source code is just a comment, usually preceded by an asterisk and so is anything after an asterisk in the first column. These do not affect the program. If a comment exceeds your desired line length, it is safe to open a new line below it and retype the comment there, preceded by an asterisk.

BASIC and Extended BASIC programs can be reformatted, but the method described above is not reliable. If you must reformat them, I think that the following program will do a foolproof job. Again, first you must put carriage returns at the end of each program line and the only practical way is to get the CTRL U underline cursor and go through inserting them with SHIFT M. For those of you who are not programmers, I had better emphasize that I am talking about numbered program lines, not the numbered lines of text that appear in the Editor.

```
100 DISPLAY AT(3,6)ERASE ALL
:"PROGRAM RELISTER":"":" Wi
ll reformat a LISTed XBas
ic program from any lineleng
th to any other length."
110 DISPLAY AT(8,1):" Each
program line (not file li
ne) must end in a carriag
e return."
120 DISPLAY AT(12,1):"Input
filename?":"DSK" :: ACCEPT A
T(13,4):IF$ :: DISPLAY AT(15
,1):"Output filename?":"DSK"
 :: ACCEPT AT(16,4):OF$
130 DISPLAY AT(18,1):"Presen
t line length?" :: ACCEPT AT
(18,22)SIZE(2)VALIDATE(DIGIT
):A
140 DISPLAY AT(20,1):"Reform
at to what length?" :: ACCEP
T AT(20,26)SIZE(2)VALIDATE(D
IGIT):X :: IF X=A THEN 130
150 OPEN #1:"DSK"&IF$,INPUT
:: OPEN #2:"DSK"&OF$,OUTPUT
:: IF X<A THEN 230
160 IF EOF(1)THEN 270 :: LIN
PUT #1:M$ :: L=LEN(M$):: IF
POS(M$,CHR$(13),1)=0 THEN 18
0
170 IF P+L<X+1 THEN PRINT #2
:M$ :: P=0 :: GOTO 160 ELSE
PRINT #2:SEG$(M$,1,X-P)&CHR$
(13):SEG$(M$,X-P+1,255):: P=
0 :: GOTO 160
180 IF L<A THEN M$=M$&RPT$("
",A-L):: L=A
190 IF P=0 THEN PRINT #2:M$;
:: P=L :: GOTO 160
200 IF P+L<X THEN PRINT #2:M
$;:: P=P+L :: GOTO 160
210 IF P+L=X THEN PRINT #2:M
$&CHR$(13):: P=0 :: GOTO 160
220 PRINT #2:SEG$(M$,1,X-P)&
CHR$(13):SEG$(M$,X-P+1,255);
:: P=LEN(SEG$(M$,X-P+1,255))
:: GOTO 160
230 IF EOF(1)THEN 270 :: LIN
PUT #1:M$
240 L=LEN(M$):: IF L+P>X THE
N PRINT #2:SEG$(M$,1,X-P)&CH
R$(13):: M$=SEG$(M$,X-P+1,25
5):: P=0 :: GOTO 240
250 IF M$=CHR$(13)THEN 230
260 IF POS(M$,CHR$(13),1)<>0
 THEN PRINT #2:M$ :: P=0 ::
GOTO 230 ELSE PRINT #2:M$;::
 P=LEN(M$):: GOTO 230
270 CLOSE #1 :: CLOSE #2
```

Another way of reformatting a program listing is to use Curtis Alan Provance's remarkable Textloader to convert the listing to program format and then listing it to disk in the desired format, using Super Extended Basic. However, Textloader can introduce some bugs, so be sure to test the program before you list it and be prepared to fix the bugs.

Now that you have gone to all that work, are you going to print your article through the Funnelweb Formatter and perhaps garble it again? Remember what I told you about the &, @, *, ^ and the leading period! Program listings usually contain those characters. If you have not modified your FO file and you do not replace and transliterate, you will print garbage! o

continued from page 13

Note line 170: CLOSE #1:DELETE. The DELETE command causes your disk controller to delete the file after it is closed. This was necessary as your TI will only allow 127 files per disk and if I did not delete the files, the limit would have been reached.

So what were the unacceptable file names? Everything over 127 bombed out as did 0, 32 (space) and 46 (period). Everything else worked, including lower case.

The TI Manual recommends against using lower case letters in a file name. You can, but there is a danger of saving a file as "DSK1.myfile", trying to read it as "DSK1.MYFILE" and not finding it. These are two different names to your disk controller.

The point is that everything below 128 (except 0, 32 and 46) can be used in a file name (with a TI controller, anyway). Enjoy. o

Renew Now!

Sorting part 5

by Ron Brubaker, USA

An introduction to the shell sort was the subject of last month's article. This article is intended to show that the shell sort is amendable to all of the variations shown previously for the bubble sort. First of all, consider the following program which utilises the same sorting routine shown at the end of last month's article but to sort a different set of data.

```
10 REM *** READ IN A LIST OF RANDOMLY ORDERED NAMES ***
20 REM
30 DIM A$(15),R(15),P(15)
40 READ N
50 FOR I=1 TO N
60 P(I)=I
70 READ A$(I),R(I)
80 PRINT R(I);
90 NEXT I
100 DATA 15
110 DATA "WASHINGTON, GEORGE",1,"JEFFERSON, THOMAS",3
120 DATA "FORD, GERALD",37,"KENNEDY, JOHN",34
130 DATA "FILLMORE, MILLARD",13,"ARTHUR, CHESTER",21
140 DATA "ADAMS, JOHN Q.",6,"LINCOLN, ABRAHAM",16
150 DATA "ROOSEVELT, FRANKLIN",31,"REAGAN, RONALD",39
160 DATA "CARTER, JAMES",38,"WILSON, WOODROW",27
170 DATA "MONROE, JAMES",5,"ROOSEVELT, THEODORE",25
180 DATA "ADAMS, JOHN",2
190 REM
200 REM ***** SHELL SORT ROUTINE - NUMERIC *****
210 REM
220 L=(2*INT(LOG(N)/LOG(2)))-1
230 L=INT(L/2)
240 IF L<1 THEN 390
250 FOR J=1 TO L
260 FOR I=J+L TO N STEP L
270 K=I
280 T=R(I)
290 IF R(I-L)<=T THEN 330
300 R(I)=R(I-L)
310 I=I-L
320 IF I>L THEN 290
330 R(I)=T
340 I=K
350 NEXT I
360 NEXT J
370 GOTO 230
380 REM
390 REM *** ROUTINE TO PRINT SORTED LIST OF NUMBERS ***
400 REM
410 PRINT
420 FOR I=1 TO N
430 PRINT R(I);
440 NEXT I
450 END
```

In this form the program will sort the list of presidents in the order in which they held office. If an alphabetic sort is preferred simply change the middle section as follows:

```
10 REM *** READ IN A LIST OF RANDOMLY ORDERED NAMES ***
20 REM
30 DIM A$(15),R(15),P(15)
40 READ N
50 FOR I=1 TO N
60 P(I)=I
70 READ A$(I),R(I)
80 PRINT R(I);
90 NEXT I
100 DATA 15
110 DATA "EDMUND, BARTON     ",1,"WATSON, JOHN      ",3
120 DATA "HOLT, HAROLD      ",17,"HUGHES, WILLIAM      ",7
130 DATA "CHIFLEY, BEN      ",16,"DEAKIN, ALFRED      ",2
140 DATA "FRASER, MALCOLM   ",22,"MENZIES, ROBERT     ",12
150 DATA "GORTON, JOHN      ",19,"WHITLAM, GOUGH      ",21
160 DATA "REID, GEORGE      ",4,"CURTIN, JOHN      ",14
170 DATA "McEWEN, JOHN      ",18,"SCULLIN, JAMES      ",9
180 DATA "McMAHON, WILLIAM",20
190 REM
200 REM ***** SHELL SORT ROUTINE - ALPHABETIC *****
210 REM
220 L=(2*INT(LOG(N)/LOG(2)))-1
230 L=INT(L/2)
240 IF L<1 THEN 390
250 FOR J=1 TO L
260 FOR I=J+L TO N STEP L
270 K=I
280 T$=A$(I)
290 IF A$(I-L)<=T$ THEN 330
300 A$(I)=A$(I-L)
310 I=I-L
320 IF I>L THEN 290
330 A$(I)=T$
340 I=K
350 NEXT I
360 NEXT J
370 GOTO 230
380 REM
390 REM *** ROUTINE TO PRINT SORTED LIST OF DATA ***
400 REM
410 PRINT
420 FOR I=1 TO N
430 PRINT A$(I);
440 NEXT I
450 END
```

As you will notice from running this program, the list of Australian Prime Ministers will be sorted in alphabetic order. As a school teacher it is interesting to note that there is always a student in every class that knows America's first president (George Washington, who live 200 years ago), but it is rare to find a student who knows Australia's first Prime Minister (Edmund Barton, who lived less than 100 years ago!). Sometimes I wonder where our priorities are!

If you prefer to use a pointer sort then it is possible to write a short program for numeric and string variables. Perhaps this could be done as an exercise?

Next month will probably be the last in this series. It will introduce a sorting algorithm called Quicksort II which is one of the finest sorting methods. o

continued from page 16

```
890 DISPLAY AT(1,1):"FILENAME: "&FN$ :: DISPLAY AT(2,5):"STUDENT'S
NAME" :: DISPLAY AT(2,25):"MARK"
900 DISPLAY AT(3,1):"------------------------------"
910 FOR I=1 TO N
920 NUM$=STR$(I)&"."
930 IF I>9 THEN 970
940 DISPLAY AT(I+3,2):NUM$
950 NEXT I
960 GOTO 1010
970 FOR I=10 TO N
980 NUM$=STR$(I)&"."
985 IF I>20 THEN 1010
990 DISPLAY AT(I+3,1):NUM$
1000 NEXT I
1010 L=2
1020 FOR I=1 TO N :: L=L+1
1025 ON WARNING NEXT
1030 ACCEPT AT(L+1,4)VALIDATE(UALPHA,".")BEEP SIZE(-20):SN$(I)
1040 CALL KEY(0,K,S):: IF K=11 THEN L=L-1 :: I=I-1 :: GOTO 1050
1045 ON WARNING NEXT
1050 ACCEPT AT(L+1,25)VALIDATE(DIGIT,".")BEEP SIZE(-4):M(I)
1060 CALL KEY(0,K,S):: IF K=11 THEN GOTO 1030
1070 IF I/20=INT(I/20)THEN GOTO 1090
1080 NEXT I
1090 DISPLAY AT(24,1):"Press any key to continue.."
1100 CALL KEY(0,K,S):: IF S=0 THEN 1100 :: CALL CLEAR
1108 L=2
1110 FOR I=21 TO N :: L=L+1 :: NUM$=STR$(I)&"." :: DISPLAY AT
(L-1,1):NUM$ :: NEXT I
1115 L=2
1116 FOR I=21 TO N :: L=L+1
1118 ON WARNING NEXT
1120 ACCEPT AT(L-1,4)VALIDATE(UALPHA,".")BEEP SIZE(-20):SN$(I)
1123 CALL KEY(0,K,S) :: IF K=11 THEN L=L-1 :: I=I-1 GOTO 1130
1125 ON WARNING NEXT
1130 ACCEPT AT(L-1,25)VALIDATE(DIGIT,".")BEEP SIZE(-4):M(I)
1135 CALL KEY(0,K,S) :: IF K=11 THEN 1120
1140 NEXT I
```

And that is about it folks. Hope you enjoyed it and learned something from it. Try typing it in and seeing how it works. Perhaps you can incorporate it into one of your own programs. If you have not done much programming, it is not too late to start! o

Decoding EPROM files

File Handling Techniques part 5

by Ben Takach

If you followed the sequence of events from the previous month's article then you may make the remark that the sub-routine in lines 7000-7280 is not repeated at all! It is only used once, thus, it ought to be part of the main program. Indeed it should be! This was one convenient method of combining two stand alone programs in one. I have to print out more definition tables than complete programs, therefore it is used as a stand alone program with a few lines preceeding the sub-routine. Combining the two was much easier by adopting this concept, than if it had have been part of the main program. The program produces 32 pages of printout and the length of the definition tables is 11 pages. It takes the computer almost 2 hours to complete the task.

The first page of the completed program and the first definition page is reproduced for better understanding. Do I hear you saying "The cursed thing is in the German language!". Well TI does not care, the assignment came from a German Company so it had to be in the German language.

The program structure:

Computer programs to solve a particular problem written by different programmers, will have different structures. There are many ways to write a program. The purists may argue on the merit of one or the other solution, however if the program runs well enough then it is fine! The program run time will be influenced somewhat by its architecture. Likewise the memory requirement will also be dependent on the programming concept. In the case of small programs it may only be a second run time difference between two alternate programming concepts. It is hardly worth worrying about. Nor should you go out of your way to save memory. Large complicated programs on the other hand may not run at all due to lack of memory space, if inappropriate concepts are applied.

One has to be very careful in such cases, because every byte saved will increase the chances of success. This program falls into that category. The problems are:

* Two large files,
* Several dimensioned arrays, two of them large,
* A long list of numeric and string variables,
* Three open files, and
* Heavy going from start to finish.

Several memory saving steps have been applied throughout to ensure that the limited memory space of the TI99/4A is conserved where it is possible. These are:

* Although A$(110) is declared in the DIM statement, it is not filled. The 80 byte long String is passed on to a String variable as soon as it is inputted from the disk file, and A$(n) is reset (made to be a null string).
* Numeric variables are reused whenever it is possible.
* Variable names have been kept short.
* Leaving behind any pending RETURN addresses have been carefully avoided.
* All repeated sequences are in sub-routines.
* Printout is line by line, results are not stored in memory.

The program run time is very long. Due to the very tight fit, garbage collecting trips are frequent. This will delay program execution. The long execution time may be appreciated by the fact that the EPSON LQ 1000 parallel printer, running through a print buffer, will have to wait after printing a line for about 2 seconds before it receives the next print command. (The short end of page delay in line 3140 will affect the execution speed of the program at the end of each page only).

Oh, just one last hint, try to design and write your program in such a way that the completed segment may be run at any time during the programming process. By this method errors and omissions will show up at a time when corrections are easier than debugging the finished article.

Now we will step through the program and explore its architecture in detail. There is not much to talk about in the first few lines up to line number 120. Line 1 is my standard REM line. One does not need to remember the name of the program and it is very convenient to save it many times during programming. Data entry in lines 40, 50 and 60 and again in 130 to 180 gives a professional touch to the program. All the questions appear first, then the information is entered one by one. The entire field is in full display, one can concentrate on the correct response.

It would also be easy to include an OOPS facility between 180 and 190, which could display the message, say at line 23; Any corrections (Y/N)? If the reply is yes then execution would return to 180. Lines 190-220 will be used to assign hexadecimal values to the processor instruction set mnemonics. This could have been handled in many other ways (e.g. by a DATA and READ statement). Considering the memory usage, the adopted method is the fastest and uses the least amount of memory.

Line 850 opens the printer, sub-routine 3000 will print the heading and sub-routine 3200 will print the column headings. Labels and label addresses cannot be extracted from the EPROM code directly. These are only identified in the definition file. Searching for these would make the program execution much longer. It is easier to look it up in the definition file (labels are always the last four records in this file) and enter it through the keyboard. The labels and their addresses are entered in lines 860 through 866. Line 866 will look after the essential Hex to decimal conversion. The actual conversion is executed by sub-routine 6000. Line 870 reads the data, which will be used by sub-routine 5000, the decimal to hex conversion routine.

The real action begins with line 900, or more precisely it all happens in lines 900 through 966. First the line and byte number counter is set to 0, then the EPROM code records are input sequentially. The length of each record is, as explained earlier, 80 characters and contains 16 words of EPROM code. The record is assigned to E$ and the record (A$(I+1)) is reset to avoid the otherwise inevitable memory full error. The 16 EPROM words are decoded one at a time through the nested FOR - NEXT loop in line 910.

The EPROM code is in hexadecimal format and the finished program listing, including the line numbers, also have to be in hexadecimal format. The program generates the line numbers and the byte addresses D1$ and D2$ respectively. These have to be converted to hexadecimal through sub-routine 5000. The line numbers as well as the byte address are in a 4 digit format. Anything less has to be padded using an appropriate number of leading zeros. Lines 5110-5130 take care of the padding. D3$ on the other hand has to be converted to decimal, because it will be used as a pointer to search for the appropriate definition record. Sub-routine 6000 does the conversion.

Lines 915 to 920 in conjunction with sub-routine 8660 and 8640 test for a label code coincidence and include the label information in the program line if any have been found. This is handled by two further nested FOR - NEXT loops. Line 922 inputs the definition record indicated by the EPROM code. Line 930 is needed to cater for an anomaly. Operand code 0000 is the address of the first input terminal. The code 0000 is assigned to the NOP instruction. In other words if the most significant nybble is 0 then the next 3 zeros will not imply Input terminal 1, but simply a no operation program step.

continued on page 9

TIA/Link

by Barry Traver, PA USA, from MICROpendium

The program TIA/LINK will write for you the assembly source code for a CALL LINK that will put a TI-Artist Instance on the screen at any desired location!

In normal graphics mode (the usual mode in TI BASIC or TI Extended BASIC), you only have at the most 16 character sets to work with (normally only 14 in Extended Basic, because the other space is usually reserved for sprite information), so you do have a limited number of characters that can be redefined. Why not use bit-mapped graphics? Well, that can be done from Extended BASIC, but it is very complicated and difficult to do. If you want to play with bit-mapped graphics in Extended Basic, I recommend that you purchase from Texaments Harry Wilhelm's The Missing Link, which is a rather amazing extension of Extended BASIC specifically in that direction. Graphics in Extended Basic is normally achieved by redefinition of characters. That is what we will still be doing. Extended Basic, however, has two important drawbacks.

1...It takes a long time to redefine a whole bunch of characters and put them on the screen, and;

2...Sometimes you run out of characters to redefine. To that, we may add a third drawback;

3...Creating a graphic can be a tedious thing to do, especially for those of us who have perhaps little art talent. Let's see if we can overcome these drawbacks, one by one.

The obvious answer to the slow speed of redefining characters and creating the screen display is to put those operations into assembly. We have done that before, but with TIA/LINK we will be doing it with the emphasis on pictures rather than text. Since we will be staying with normal graphics mode, there is no complete answer to the problem of running out of characters, but TIA/LINK will let you make use of character sets 15 and 16 in Extended Basic (that is 16 additional characters), so you will not "run out of ink" as fast as you would otherwise. That leaves one other drawback to overcome:

There are not extensive graphics libraries for the graphics mode in Extended Basic, but we now have in abundance such libraries for other graphics formats. The standard format for graphics is the TI-Artist Instance. There are lots of icons, or pictures either already in TI-Artist format or able easily to be converted into TI-Artist Instance format. With all the programs available, why not liven up your Extended Basic programs? The CALL LINK("GRAFIC",ROW,COL) created by TIA/LINK is an easy way to do this. The passed parameters (ROW, and COL) permit the placing of the graphic at any desired screen location, i.e. at any ROW and COLumn. As is normally the case with Extended Basic, it can be more efficient to reuse characters when possible, if the same 8x8 character pattern appears more than once in the picture. In general, you ought always to (re)use the space character -CHR$(32)- whenever possible. With V 1.3 of TIA/LINK, you do have the choice of whether or not to reuse characters other than the space character.

TIA/LINK does run faster if you reuse only the space character, but the number of characters that can be redefined is limited, so I recommend that you ordinarily choose the other option: it may allow you to use TIA/LINK with certain Instances that are too large for you to use. When TIA/LINK is at work redefining characters, it starts at CHR$(159) (the last character in character set 16) and works backwards. This means that 33 characters (not merely 17 characters) can be redefined before it starts redefining any "regular" characters (i.e.,from ASCII 33 to 126) in your character set. If you are willing to give up lower case (from ASCII 97 to 122), that will give you another 30 characters or so to work with for the picture. This means that you can put a TI-Artist Instance on the screen that may measure, say, eight rows by eight columns (requiring up to 64 redefined characters, if we assume that none can be reused). Larger TI-Artist Instances may require some creativity, if you want to have text on the screen at the same time. If you want a large picture plus text, you may need to redefine separately some lesser-used characters that have lower ASCII numbers. For example, you may need to use "#,$,/,&,+,<,=.>, or @".

Suppose you find that your TI-Artist Instance was large enough to require use of characters from ASCII 159 to 87. That means you have "lost" your capital letters from Z back to W. Suppose, however, that you want to place on the screen text that includes the capital letters W and Y. After you display your graphic,you could add the following statements to your Extended Basic code before displaying your text to solve the problem:

```
100 CALL CHARPAT(87,A$) ! Get definition for W".
110 CALL CHAR(35,A$) ! Redefine # as W.
120 CALL CHARPAT(89,A$) ! Get definition for Y.
130 CALL CHAR(36,A$) ! Redefine $ as Y.
```

The graphic itself could be placed on the screen in this way:

```
150 CALL LINK("GRAFIC",9,11)
```

Where 9 and 12 represent the ROW and COLumn where you want the top left corner of the graphic to be placed. It does not matter whether you put up the text first then the graphic, or the other way around. After you are finished with your graphics display in your Extended Basic program, you may need to restore character definitions that have been changed. You can use either FONTALS or VDP/SAVER with normal character definitions to create an assembly routine that will quickly restore your character sets back the way they were before some of the characters were redefined. The DV80 program SAMPLE_I is to be used with TIA/LINK, as a sample Picture?

This is an abbreviated version of the docs about this program, for a complete description see pages 14 & 15 in MICROpendium HR.

```
90 !TIA/LINK,By Barry Traver, From MICROpendium July,1991Page 14
100 ! COPYRIGHT (C) 1991 by Barry Traver, 835 Green Valley Drive, Philadelphia,
PA 19128 (phone: 215/483-1379) — ALL RIGHTS RESERVED!
110 ! TIA/LINK by Barry Traver
120 DIM DEF$(160)
130 DISPLAY AT(1,1)ERASE ALL:"TIA/LINK, Version 1.3":" MICROpendium edition":"
(C) COPYRIGHT 1991":"    BY BARRY A. TRAVER": :
140 DISPLAY AT(7,1):"TI-ARTIST INSTANCE FILE?":" DSK" :: DISPLAY AT(12,1):"A/L S
OURCE CODE FILE?":" DSK"
150 DISPLAY AT(17,1):"CALL LINK NAME?":" GRAFIC" :: DISPLAY AT(22,1):"REUSE ONLY
 SPACE CHARACTER?":" Y"
160 ACCEPT AT(8,2)SIZE(-27)BEEP:IN$
170 ON ERROR 180 :: OPEN #2:IN$,INPUT :: ON ERROR STOP :: GOTO 200
180 ON ERROR 190 :: CLOSE #2 :: ON ERROR STOP
190 RETURN 140
200 ACCEPT AT(13,2)SIZE(-27)BEEP:OUT$
210 ON ERROR 220 :: OPEN #1:OUT$,OUTPUT :: ON ERROR STOP :: GOTO 240
220 ON ERROR 230 :: CLOSE #1 :: ON ERROR STOP
230 RETURN 180
240 ACCEPT AT(18,2)SIZE(-6)BEEP:PN$ :: PN$=PN$&RPT$(" ",6-LEN(PN$))
250 ACCEPT AT(23,2)VALIDATE("YN")SIZE(-1)BEEP:F$ :: IF F$="Y" THEN F=1 ELSE F=0
260 W$="" :: INPUT #2:W,H :: C=160 :: DISPLAY AT(22,1):"USING CHARACTER":""
270 IF EOF(2)THEN 360 ELSE LINPUT #2:I$
280 IF I$="0,0,0,0,0,0,0,0" THEN W$=W$&CHR$(32):: DISPLAY AT(22,16):32 :: GOTO 2
70
290 IF F THEN 320 ELSE I=160
300 IF I$=DEF$(I)THEN W$=W$&CHR$(I):: DISPLAY AT(22,16):I :: GOTO 270
310 I=I-1 :: IF I>=C THEN 300
320 C=C-1 :: IF C=32 THEN 890 ELSE DEF$(C)=I$
330 W$=W$&CHR$(C)
340 DISPLAY AT(22,16):C
350 GOTO 270
360 CLOSE #2 :: CALL CLEAR
370 FOR I=0 TO 12 :: CALL COLOR(I,16,1):: NEXT I :: CALL SCREEN(5)
380 FOR R=0 TO 1 :: PRINT #R:"* SOURCE CODE CREATED BY TIA/LINK, A PROGRAM":"*
 COPYRIGHT (C) 1991 BY BARRY A. TRAVER,"
390 PRINT #R:"*    835 GREEN VALLEY DRIVE,":"*    PHILADELPHIA, PA 19128":"*
   (PHONE: 215/483-1379)":" " :: NEXT R
400 FOR R=0 TO 1 :: PRINT #R:"* DEFINE ENTRY POINT":" ":"       "&"DEF  "&PN$:"
" :: NEXT R
410 FOR R=0 TO 1 :: PRINT #R:"* XB EQUATES":" ":"BASIC  EQU  >006A":"CFI    EQU
 >12B8":"FAC    EQU  >834A":"GPL WS EQU  >83E0"
420 PRINT #R:"NUMREF EQU  >200C":"VMBW   EQU  >2024":"VSBW   EQU  >2020":"XMLLNK
 EQU  >2018":" " :: NEXT R
430 FOR R=0 TO 1 :: PRINT #R:"* SET UP WORKSPACE":" ":"WS     BSS  32":" " :: NE
XT R
440 FOR R=0 TO 1 :: PRINT #R:"* CHARACTER DEFINITIONS":" " :: NEXT R
450 FOR I=C TO 159
460 A$="BYTE "&DEF$(I)
```

continued on page 12

Riemann Sphere Graphic Program

More on The Missing Link (TML)

by Stephen Shaw, England

This program illustrates how to project a graphic onto a sphere. The projector has a wide angle lens and is at the South pole of the sphere (which of course is translucent). This location leads to some squeezing of the image as it nears the South pole, which could be regarded as being "at infinity".

The graphic must have coordinates within the range +2 to -2 in both directions if the image is to appear entirely within a polar view otherwise the image will go round the sphere! The far pole has a value of infinity so do not worry about going past it... - larger graphics can easily be scaled down!

The program presented here produces two types of graphic- one is a set of nested squares and the other is a spiral starting from the North pole, which has location 0,0.

A further variation shown here is to have the globe slightly transparent, such that you can see the plot on the "other side" as well as the side facing you- to make these less confusing, the sample plots are done with lines on the face and with spaced dots on the back.

If only the face is to be viewed then you can easily plot with just dots and forget about lines- and you do not even have to draw the circle either.

This program is for Extended Basic plus The Missing Link but can be easily transferred to any other language that allows use of pixel graphics.

The original program was written 9.3.91 by R Castle-Smith in Rocky Mountain Basic for output to a plotter. The conversion for the TI is by Stephen Shaw.

The original program was published in Fractal Report #16 (8/91) cost Two Pounds, or a 6-issue subscription for Ten Pounds, from Reeves Telecom. Labs. Ltd., West Towan House, Porthtowan, TRURO, Cornwall, TR4 8AX.

This program allows you to view the globe from any direction. The direction is given by three inputs, which are the angle to the centre of the sphere from each of the three planes (left-right, up-down, towards-away) passing through the centre. The values of these three inputs are relative rather than absolute- an input of 1,1,0 is the same view as 2,2,0.

The three values are in the variables A(1), A(2), and A(3). If the first two are kept at zero, and A(3) is varied from +1 to -1 for successive pictures, you will see a view from one pole passing to the equator to the other pole.

```
100 ! RIEMANN SPHERE R CASTLE-SMITH 3/91 for TI by S
    SHAW 8/91
110 ! EX BAS + THE MISSING LINK
120 CALL LINK("CLEAR")
130 RANDOMIZE
140 IF RND<.5 THEN SP$="SP" ELSE SP$="SQ"
150 A(1)=3*RND*RND :: A(2)=3*RND*RND :: A(3)=4*RND-4*
    RND :: IF A(1)+A(2)+ABS(A(3 ))<.01 THEN 150
160 IF A(1)>2 OR A(2)>2 THEN 150
170 FOR T=1 TO 3 :: CALL SHORT(A(T)):: NEXT T
180 IF RND<.5 THEN PLOBAC=0 ELSE PLOBAC=1
190 REM
200 REM DERIVE TRANSFORM
210 REM
220 IF A(1)=0 AND A(2)=0 THEN B(1),C(2)=1 :: B(2),B(3)
    ,C (1),C(3)=0 :: GOTO 290
230 M=SQR(A(1)*A(1)+A(2)*A(2))
240 B(1)=-A(2)/M :: B(2)=A(1)/M :: B(3)=0
250 N=SQR(M*M+A(3)*A(3))
260 C(1)=-A(1)*A(3)/(M*N)
270 C(2)=-A(2)*A(3)/(M*N)
280 C(3)=M/N
290 ! DRAW CIRCLE!!!
300 CALL LINK("PRINT",1,1,SP$&" "&"X="&STR$(A(1))&"Y="&
    STR$(A(2))&" Z="&STR$(A( 3)))
310CALL LINK("PRINT",180,1,"PLOBAC="&STR$(PLOBAC)&"(0=
NO PLOT ON BACK)")
320 !
330 !
340 CALL LINK("CIRCLE",101,101,70)
350 !
360 REM GENERATE PATTERN
370 !
380 ! SPIRAL
390 IF SP$="SQ" THEN 480
400 FOR I=1 TO 5000 STEP 4 :: R=I/1000 :: T=I/10 :: XC
    =R *COS(T):: YC=R*SIN(T)
410 GOSUB 630
420 NEXT I
430 !CALL LINK("DUMP")
440 CALL LINK("PRINT",9,1,"!*!")
450 CALL KEY(5,K,S):: IF S<>1 THEN 450
460 RUN
470 RUN
480 ! SQUARE
490 !
500 FOR SQUARE=.1 TO 4 STEP .2
510 ! FOUR SIDES, 1 TO 4
520 YC=-SQUARE :: FOR XC=-SQUARE TO SQUARE STEP SQUARE/
10 :: GOSUB 630 :: NEXT C
530 XC=SQUARE :: FOR YC=-SQUARE TO SQUARE STEP SQUARE/
    10 :: GO SUB 630 :: NEXT YC 540 YC=SQUARE :: FOR
    XC= SQUARE TO -SQUARE STEP -SQUARE/10 :: GOSUB 630
    :: NEXT C
550 XC=-SQUARE :: FOR YC=SQUARE TO -SQUARE STEP -
    SQUARE/ 10 :: GOSUB 630 :: NEXT YC
560 PENUP=1
570 NEXT SQUARE
580 CALL LINK("PRINT",9,12,"*!*")
590 CALL KEY(5,Q,W):: IF W<>1 THEN 590
600 ! CALL LINK("DUMP")
610 CALL LINK("CLEAR")
620 RUN
630 MD=XC*XC+YC*YC
640 XR=2*XC/(1+MD)
650 YR=2*YC/(1+MD)
660 ZR=(MD-1)/(1+MD)
670 !
680 BACK=0
690 IF A(1)*XR+A(2)*YR+A(3)*ZR<0 THEN BACK=1
700 IF BACK=1 AND PLOBAC=0 THEN PENUP=1 :: GOTO 750
710 XV=B(1)*XR+B(2)*YR+B(3)*ZR
720 YV=C(1)*XR+C(2)*YR+C(3)*ZR
730 IF BACK=1 THEN PENUP=1
740 IF PENUP=1 THEN OLDX=XV :: OLDY=YV :: CALL LINE(OL
DY,OLDX,YV,XV):: PENUP=0 ELSE CALL LINE(OLDY,OLDX,(YV),
(XV)):: OLDX=XV :: OLDY=YV
750 RETURN
760 STOP
770 SUB LINE(OLDX,OLDY,X,Y)
780 IF OLDX=0 AND OLDY=0 THEN SUBEXIT
790 OLX=OLDX+101 :: X=X+101 :: OLY=OLDY+101 ::
Y=Y+101 :: CALL LINK( "LINE",OLX,OLY,X,Y):: SUBEND
800 SUB SHORT(X)
810 S$=STR$(X):: S$=SEG$(S$,1,5):: X=VAL(S$)
820 SUBEND
```

You will quickly see that by varying the viewing angles to produce several pictures and then putting them together with COMIC SHOW 4 (available from the disk library) you can have an animated tour around your global graphic.

To transfer a TI Artist format graphic to a form that can be transformed with this type of routine, the easiest route seems to be to locate the graphic at the top left of a TI Artist picture, then load the picture using Myarc Extended Basic.

Myarc XB is the only pixel based graphic program I know that will tell you if a specific pixel is "on" or "off". If you know this information by scanning the graphic, you can set up a disk file containing the X,Y coordinates for each ON pixel, then read that as an input to the above transform program, with scaling and positioning as required.

This little program requires Myarc Extended Basic, which in turn requires the Myarc Ram Card. It makes use of a unique feature of Myarc XB, the ability to scan the screen and determine if a pixel is on or off. This is by means of a modified CALL GCHAR.

The program is used to create a DV80 data file to be read by another program, or converted into a mergeable set of DATA lines after adding line numbers and data statements, commas etc, which can of course be printed (after amendment).

```
100 CALL INIT
110 CALL LOAD("DSK1.MYARTIST") ! [from disk library]
120 CALL GRAPHICS(3)           !  set bit map mode
130 CALL LINK("LOAD","DSK1.PIC") !load pic-do not add _P
140 REM
150 OPEN #1: "RD.DATA"         ! RD is optional device
160 FOR ROW=1 TO 20            ! name for Myarc ramdisk
170 FOR COL=1 TO 160           ! approx size of overscan
180 CALL GCHAR(ROW,COL,VAR)    ! myarc xb variant form
190 IF VAR=1 THEN PRINT #1: ROW;COL
200 NEXT COL
210 NEXT ROW
220 CLOSE #1
230 END
```

The object code utility MYARTIST is available from the disk library.

The data is saved by the above program in a DV80 file which can be edited by TI Writer or read by your graphic drawing program. The data file looks like this in TI Writer- part only-

```
6  1
6  2
6  3
6  4
6  5
6  6
6  7
```

and so on and so on and so on...

Of course, it makes life easier if we prepare a MERGE file directly after scanning the image, and the following program will do this, collecting ten pixel coordinates per program DATA line...

```
100 CALL GRAPHICS(3)   ! MYARC XB ONLY
110 CALL INIT :: CALL LOAD("DSK2.MYARTIST")
120 CALL LINK("LOAD","DSK1.PIC")
130 OPEN #1: "RD.DATA", DISPLAY ,VARIABLE 163 ! will
    create loadable file
140 L=1 :: CT=0
150 REM
160 FOR ROW=1 TO 20     ! approx size of image
170 FOR COL=1 TO 160
180 CALL GCHAR(ROW,COL,VAR)
190 CALL WRITE(0,20,2,STR$(ROW)&" "&STR$(COL)&" "&
    STR$(VAR)&"   ")
200 IF VAR<>1 THEN 360
210 CT=CT+1
220 IF CT=1 THEN A$=CHR$(0)&CHR$(L)&CHR$(147)
230 IF CT=11 THEN A$=A$&CHR$(0) :: PRINT #1: A$ :: L=L
    +1 :: CT=1 :: A$="" :: GOTO 220
240 IF CT>1 THEN A$=A$&CHR$(179)
250 N$=STR$(ROW) :: LN=LEN(N$)
260 A$=A$&CHR$(200)&CHR$(LN)
270 FOR I=1 TO LN
280 A$=A$&CHR$(ASC(SEG$(N$,I,1)))
290 NEXT I
300 A$=A$&CHR$(179)
310 N$=STR$(COL) :: LN=LEN(N$)
320 A$=A$&CHR$(200)&CHR$(LN)
330 FOR I=1 TO LN
340 A$=A$&CHR$(ASC(SEG$(N$,I,1)))
350 NEXT I
360 NEXT COL :: NEXT ROW
370 A$=A$&CHR$(0) :: PRINT #1: A$
380 A$=CHR$(255)&CHR$(255)
390 PRINT #1: A$ :: CLOSE #1
```

The merge data file will load directly using MERGE RD.DATA and it starts off like this...

```
1 DATA 1,132,1,133,1,134,2,132,2,133,2,134,3,132,3,133,3,134,4,132
2 DATA 4,133,4,134,5,132,5,133,5,134,6,1,6,2,6,3,6,4,6,5
3 DATA 6,6,6,7,6,8,6,9,6,15,6,16,6,17,6,18,6,19,6,20
4 DATA 6,21,6,47,6,48,6,49,6,50,6,51,6,52,6,53,6,59,6,60
5 DATA 6,61,6,62,6,63,6,64,6,65,6,69,6,70,6,71,6,72,6,73
6 DATA 6,74,6,75,6,79,6,80,6,81,6,82,6,83,6,87,6,88,6,89
7 DATA 6,90,6,91,6,92,6,93,6,97,6,98,6,99,6,100,6,101,6,107
```

and so on.

The data is a record of all pixels which are ON in the format row,column. The program that reads this data can relocate the image by varying the values... this little program will give you some idea. The data can be used by any language and is available for use by the Riemann projection program.

```
1 REM USE MERGED DATA FILE TO PRINT GRAPHIC EVERY WHICH
  WAY ALL AT ONCE
2 ! THIS LISTING FOR MYARC XB BUT ANY PIXEL ADDRESSABLE
  LANGUAGE CAN BE USED
3 !
4 ! DATA FILE TO BE MERGED INTO PROGRAM
5 !
100 CALL GRAPHICS(3)
110 READ A,B
120 CALL POINT(1,A,B) :: CALL POINT(1,180-A,240-B) ::
    CALL POINT(1,180-A-50,B+60) :: CALL POINT
    (1,180-B,A+10) :: CALL POINT(1,A+60,220-B)
130 IF A>400 THEN 500
140 GOTO 110
150 ! TEST IN LINE 130 REFERS TO END OF DATA MARKER
    INSERTED AS LAST 2 ITEMS

500 GOTO 500
```

o

Appending to D/V 80 Files

by Wesley R. Richardson, OH USA

The Extended Basic program, DV80APPEND. will append a Display Variable 80 (DV80) format file to the end of an existing DV80 file. One purpose of this utility program is to combine files which are too large for them both to be loaded into the TI-Writer Editor.

Since the TI-Writer Editor uses the last line of the file to store the margin and TAB settings, you may wish to load the appended to file using the Editor/Assembler Editor and then save the file back to disk to remove the TAB settings line. The only caution using DV80APPEND is that if the first character of a line has an ASCII value greater than 127, then that line will not be included in the output file.

The program works best with the input file on one disk drive and the output file on another disk drive because the disk is accessed for each input and output line.

```
100 REM DV80APPEND
110 REM WESLEY R. RICHARDSON, DECEMBER, 1990
120 REM NORTHCOAST 99ERS, CLEVELAND, OH
130 PRINT "DSKX. FILE TO ADD?"
140 INPUT "  ":F1$
150 PRINT "DSKX. APPEND TO FILE?"
160 INPUT "  ":F2$
170 OPEN #1:F1$,INPUT
180 PRINT "READING ";F1$
190 OPEN #2:F2$,APPEND
200 LINPUT #1:W$
210 IF EOF(1)THEN 260
220 IF ASC(W$)>127 THEN 260
230 N=N+1
240 PRINT #2:W$
250 GOTO 200
260 PRINT "CLOSING ";F1$
270 CLOSE #1
280 PRINT "CLOSING ";F2$
290 CLOSE #2
300 PRINT N;"LINES ADDED"
310 END
```

o

MAX-RLE

User Documentation File

Objectives of this file are as follows:
(1) Brief explaination of RLE files as used by CompuServe,
(2) Equiptment required to load and run MAX-RLE,
(3) Downloading RLE files from CompuServe,
(4) Loading and running MAX-RLE.

(1) What is an RLE?

RLE is an abbreviation for Run Length Encoded. These file located in various locations on CompuServe are files containing encoded represention of high resolution images displayable on the screen of the user's computer.

RLE is mostly intended for users operating with CompServe's "VIDTEX" Terminal Emulator. However many computers (The TI-99/4A included) do not have a "VIDTEX" Emulator available. Here is where the need for an offline display program comes into being. MAX-RLE and its antecedants and decendants are intended to display these files to your screen offline which have been saved to your disk while online.

(2) What do I need to run MAX-RLE?

MAX-RLE requires the following hardware and software to run.

(a) TI-99/4A Console (of course)
(b) Monitor or TV (ditto)
(c) TI-99/4A Expansion System (or equivalent)
(d) 32K Memory Expansion (or more)
(e) RS232/PIO Interface (optional)
(f) Disk Drive(s) and Controller
(g) Editor/Assembler Module (PHM 3055)

(3) How do I get these "RLE" files?

RLE files are located in several areas on CompuServe, among these are our own beloved TIFORUM data libraries, the PICSIG, ARTFORUM and the CB simulator area.

MAX-RLE was developed with Paul Charlton's Fast-Term Terminal Emulator as the emulator used in capturing files and is currently the emulator suggested for use in downloading RLE files. For specific information on Fast Term's operations see the documentation for Fast Term. For general information on downloading from CompuServe, examine the information here in the FORUM by typing GO TINEWS at the main function prompt while using CompuServe. Files downloaded from the data libraries should be done with the XMODEM protocol. Files from the CB simulator area should be captured in your buffer and saved to disk just as a message would be. XMODEM will save to disk in DISPLAY/FIXED 128 format and those with the buffer in DISPLAY/VARIABLE 80 format. The chief difference is that XMODEM format saves the file in exactly the same format as CIS and the file may be re-uploaded at a later date, or even transmitted to a bulletin board which supports XMODEM. Buffer saves cannot be retransmitted to CIS as they will lack certain header information identifying them as RLE. This information is stripped out by Fast Term on downloading.

(4) Ok, How do I use MAX-RLE?

To load MAX-RLE perform the following steps with the system turned on and the Editor/Assembler cartridge in your cartridge port.

Place the disk containing MAX-RLE in one of your drives.

Press any key to get the console's selection screen, press 2 to select Editor/Assembler and select option 3 for Load and Run.

At the Filename prompt type in DSKn.filename

where 'n' is the number of the drive containing MAX-RLE and 'filename' is the name you gave MAX-RLE on your disk.

Press Enter again to get the Program Name prompt. Type in START

From the title screen you have 3 options.

(a) Exit to the TI-99/4A Title screen by pressing FCTN and = at the same time
(b) Catalog a disk by typing in DSKn.
(c) View a file. MAX-RLE supports the translation of four types of files to your screen:
DIS/FIX 128 RLEs
DIS/VAR 80 RLEs
GRAPHX
TI-ARTIST V2.0

To do this just type in the files name, i.e. DSK1.LOGO. TI-ARTIST files used with MAX-RLE do not require you to type in the _C nor the _P ending seen in the catalog. MAX-RLE will automatically load both if present.

With the image on the screen you have three options available.

(a) Return to the MAX-RLE title screen by pressing Enter.

(b) Print screen on your printer. Press P. You will be given the MAX-RLE default of a parallel printer output for EPSON/GEMINI printers. If you are using a serial printer press enter and then enter your printer's description.

(c) Save the image on your screen in one of the four above mentioned formats. If you press S, you will be given the MAX-RLE default of GRAPHX format to save to.

If you wish to save the image in a different format then press the space bar to get the prompts for the other formats. Type in the device and filename you wish to save to i.e. DSKn.MYPICT

Notes

MAX-RLE supports the Horizon RAMdisk HD command, at the MAX-RLE title screen prompt.

Uploading files converted to RLE format to CompuServe. Read the help information available online in the data library by typing H. RLE files for uploading to CIS should be saved in D/F 128 format and uploaded using Fast-Term. While uploading select XMODEM (Modem 7) from CIS's upload options and upload in RLE format. This will automatically mark the title as RLE file type for identification by CompuServe. In addition when the time finally comes the we TI'ers get a "VIDTEX" compatible emulator we will be able to view the images on line.

Final Note

In a conversation with Travis, he commented to me that although he could not have the first RLE Translator for the TI-99/4A on CIS (in reference to my 99RLE), he was determined to have the BEST. Well, I will agree with him on that Travis has written the BEST. Thanks from all of us Travis. o

continued from page 26

Lastly, but maybe not the least, I have included some words that I used during the development of the above. RN reads the name from the PAB and puts it on the PAD where you can print it with PP. The word .N does this all in one fell swoop, and .NL examines the PAB for the length byte and prints it. All little things one has to do when trouble shooting.

C U next time. May the FORTH be with U. o

Beginning Forth - part 15

by Earl Raguse, UGOC, CA USA

DISK FILE ACCESS

As I previously promised, this time we will create and access files on disk which are not screens. Screens #32 through #36 do just that. Screens #32 and #33 basically follows the TIFM Ch 8.

I strongly recommend that you read this chapter even though it is not necessary to use what is in this article. This chapter is a little confusing to me, but I am sure it is clear and concise to the already well informed. At least TI does provide some examples even if they do have errors.

I have basically followed the example on page 8 (the 2nd page 8 that is), the major difference (from the example) is that I chose Display Variable 80 files so that they could be read by TI Writer. I initalized the buffer BUFR with BLANKS, as explained last time, to insure that there was no garbage in it. I also defined a word PABSET so that I would have control of when it was executed instead of execution on loading of the screen.

Screen #33 defines FILSET to open a file called DSK2.FILE. Then the words GNAM (GetNAMe), NLEN (NameLENgth), WNAM (WriteNAMe) and, on Screen #34, XNAM (fiXNAMe) which invokes the other three words to overwrite the dummy name FILE in the PAB to whatever one wished. OPNFIL (OPeNFILe) calls FILSET and XNAM and OPN (OPeN) Forth word to open a file.

Screen #34 then gets down to the mechanics of using the file. The variable *** is defined and initialized to contain *** which is subsequently used by DONE? and EOF? to check to see if the last input was *** to signify the end of the file. TT is simply short for -TRAILING TYPE.

The word .DD (printData Disk) prompts you to put your data disk in drive #2. If you have only one drive, you will have to change that as well as the DSK2 on line 2 of Screen #33.

```
SCR #32
 0  ( FILE ACCESS #1 EGR 12 30 86 ) FG IT : IT ;
 1  0 VARIABLE BUFR 78 ALLOT ( My PAB-BUF       )
 2  BUFR 80 BLANKS           ( Initialize it   )
 3  PABS @ 10 +              ( PAB-ADDR Pointer )
 4  BUFR                     ( PAB-BUF  Pointer )
 5  5200                     ( PAB-VBUF Pointer )
 6  FILE V80STUF             ( Name the file   )
 7  V80STUF                  ( Execute the name )
 8  : PABSET                 ( Define word     )
 9    SET-PAB                ( Begin the PAB   )
10    SQNTL ( *)             ( Sequential file )
11    DSPLY ( *)             ( Printable data  )
12    VRBL                   ( Variable record )
13    80 REC-LEN ;           ( 80 max rec length)
14  ( * default but incl for completeness     )
15  -->
```

WRTREC (WRiteRECord) actually gets your input and stores it on the PAD using GSTR$, which is a UFW. It is fairly complex, so I will not explain, see me after class (like my teacher used to say). The words, PAD BUFR 80 CMOVE 80 WRT, get 80 characters from PAD and CMOVE (move) them to BUFR. The words 80 WRT writes them to the file we opened.

WRTLOOP (WRiTeLOOP) is simply an endless loop which invokes WRTREC until we write *** that is detected by DONE? which in turn CLSE's (CLoSEs) the file, clears the PAD and escapes the loop.

WRITEFILE orchestrates it all and should need no explanation.

RDREC (ReaDRECord), on Screen #36, is another infinite loop that uses a new Forth word RD, to permit us to see what was written to file. 80 RD BUFR causes 80 characters to be read from the open file and moved to BUFR. DUP EOF? compares it, using CSTR$, with *** and if equal, escapes the loop. If not, 80 TT prints it to the screen. MYSELF goes back for more indefinitely.

READFILE just does the orchestration like WRITEFILE

Screen #36 also has WF and RF, so that the word .M puts a prompt on the screen and like TI Writer lets you abbreviate commands to two letters.

```
SCR #33
 0  ( FILE ACCESS #2 EGR 12 30 86 )
 1  : FILSET  V80STUF PABSET  ( set the PAB      )
 2    F-D" DSK2.FILE"         ( file descriptor )
 3    UPDT ;                  ( with dummy file )
 4  : GNAM CLS 5 5 AT         ( get real name   )
 5    ." INPUT FILE NAME 10 " ( prompt input    )
 6    ." CHAR MAX" 16 10 AT   ( locate cursor   )
 7    10 GSTR$ ;              ( get file name   )
 8    HEX                     ( better for VDP  )
 9  : NLEN 9 + A 0 DO DUP I - C@ ( get length   )
10    20 = IF ELSE I LEAVE THEN  ( of the file  )
11    LOOP SWAP DROP A SWAP - ;  ( name         )
12  : WNAM  PAD               ( get name at PAD )
13    PAB-ADDR @ F + A VMBW   ( and move to PAB )
14    PAD NLEN 5 + PAB-ADDR @ ( file name length)
15    9 + VSBW ; DECIMAL -->  ( to PAB byte 9   )

SCR #34
 0  ( FILE ACCESS #3  EGR 12 30 86 )
 1  : XNAM GNAM WNAM ;                ( fix name  )
 2  : CPAD PAD 80 BLANKS ;            ( clear PAD )
 3  : .M  CLS 4 3 AT                  ( print     )
 4       ." Write File = WF  "        ( the       )
 5       ." Read File = RF "  ;       ( menu      )
 6  : OPNFIL FILSET XNAM OPN ;        ( open file )
 7      0 VARIABLE *** 2 ALLOT        ( def varbl )
 8      *** 4 BLANKS *** !" ***"      ( initilize )
 9  : TT ( a n --) -TRAILING TYPE ;   ( type stuff)
10  : .DD  3 1 AT ." Put data disk "  ( prompt    )
11           ." in drive number 2 "   ( drive #   )
12       13 3 AT ." PRESS ANY KEY"    ( proceed   )
13               KEY DROP CLS ;       ( do it     )
14  -->
15

SCR #35
 0  ( FILE ACCESS #4  EGR 12 30 86 )
 1  : .P CLS                          ( print prompt)
 2    3 5 AT ." Enter a data string"
 3           ." 80 char max "
 4    8 7 AT ." Enter *** to quit"
 5           0 12 AT ;                ( accept here )
 6  : WRTREC 80 GSTR$ PAD BUFR 80     ( get string  )
 7      CMOVE  80 WRT ;               ( write record)
 8  : DONE? *** 3 CSTR$               ( check ***   )
 9      IF CLSE CPAD .M QUIT          ( close file  )
10      ELSE WRTREC THEN ;            ( write record)
11  : WRTLOOP .P PAD DONE? MYSELF ; ( the loop     )
12  : WRITEFILE CLS .DD OPNFIL WRTLOOP ; ( open file)
13  : EOF? 3 *** CSTR$ IF CLSE CPAD ( check ***    )
14         DROP ." EOF" QUIT THEN ; ( if so EOF    )
15  -->

SCR #36
 0  ( FILE ACCESS #5 EGR  2 14 87 )
 1  : RDREC  80 RD BUFR DUP EOF?  ( check for EOF )
 2           80 TT CR MYSELF ;    ( no, print rec )
 3  : READFILE CLS .DD OPNFIL     ( open read file)
 4           CLS 1 4 AT RDREC ;   ( read a record )
 5  : WF WRITEFILE ;              ( hate to type  )
 6  : RF READFILE ;               ( abbreviations )
 7    .M                          ( print menu    )
 8
 9              ( These words are for the curious )
10            ( only they do not effect the file )
11  : RN PAB-ADDR @ 10 + PAD      ( read the file )
12    15 VMBR ;                   ( name, wrt pad )
13  : .N CPAD RN PAD 15 TT ;      ( prt file name )
14  : .NL PAB-ADDR @ 9 + VSBR . ; ( prt name lngth)
15
```

continued on page 25

Regional Group Reports

Meeting Summary For MAY

Banana Coast	10/05/92 Sawtell
Central Coast	09/05/92 Saratoga
Glebe	07/05/92 Glebe
Hunter Valley	09/05/92?Boolaroo
Illawarra	11/05/92 Keiraville
Liverpool	08/05/92
Northern Suburbs	28/05/92
Sutherland	15/05/92 Jannali

BANANA COAST Regional Group
(Coffs Harbour Environs)

We never miss meeting at Kerry Harrison's residence 15 Scarba St. Coffs Harbour, 2 pm second Sunday of the month. Visitors are most welcome. Contact Kerry 52 3736, Kevin 53 2649, Rex 51 2485 or John 54 1451.

CENTRAL COAST Regional Group

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

GLEBE Regional Group

Regular meetings are normally on the Thursday evening following the first Saturday of the month, at 8pm at 107 Arundel St. Glebe. Contact Mike Slattery, (02) 692 8162.

HUNTER VALLEY Regional Group

The meetings are held regularily at the Boolaroo Ambulance Station. All welcome. Please contact Geoff Phillips on (049) 428 176 for details.

ILLAWARRA Regional Group

Regular meetings are normally held on the second Monday of each month after the TIsHUG Sydney meeting, except January, at 7.30pm, at the home of Geoff & Heather Trott, 20 Robsons Road, Keiraville. A variety of activities accompany our meetings, including Word Processing, Spreadsheets and hardware repairs. At the April meeting we had a go at fixing Bruce's 3 1/2" drive as well as other hardware problems. Contact Lou Amadio on (042) 28 4906 for more information.

LIVERPOOL Regional Group

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

NORTHERN SUBURBS Regional Group

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

Come and join in our fun. Dick Warburton.

SUTHERLAND REGIONAL REPORT

A small group of four members, attended the March meeting at Jannali. As has been the case with many of the recent meetings, most of the time was spent in "test running" the various graphics programs available.

This time it was my new version of TI Artist Fonts and Borders, volumes I,II,III. The programs are a combination of instances and fonts which are largely designed for producing Title Pages. The graphics are quite good but it proved to be quite time consuming setting up Borders using the instances provided. A lot of guess work is involved.

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

Peter young
Regional Co-ordinator

Check your renewal date. Do not miss out.

TIsHUG in Sydney

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

MAY MEETING - 2ND MAY

This is the first All Day Tutorial session. Many things were discussed and planned at the April meeting, including a session for utilising speech on the TI. Look elsewhere in this issue for other planned tutorials. Come along and join in and see what you can learn!

****

The cut-off dates for submitting articles to the Editor for the TND via the BBS or otherwise are:

June	10 May
July	14 June

These dates are all Sundays and there is no guarantee that they will make the magazine unless they are uploaded by 6:00pm, at the latest. Longer articles should be to hand well before the aboves dates to ensure there is time to edit them.

****

TIsHUG Meetings for Sydney

May

The first all day tutorial session. Your chance to learn about using software, writing programs or understanding hardware. We can provide anything that you want but you must tell us what you would like and at what level you would like it.

June

A Special General Meeting to consider changes to our Memorandum of Association. New software and hardware to be demonstrated. Watch this space for more details.

July

The second buy, swap and sell day. This one is on the first Saturday of the school holidays but plan to take the day off and see what is about.

August

New software and hardware to be demonstrated. Watch this space for more details.

September

The second all day tutorial session. Your chance to learn about using software, writing programs or understanding hardware. We can provide anything that you want but you must tell us what you would like and at what level you would like it.

October

The third buy, swap and sell day. This one is in the middle of the school holidays but plan to take the day off and see what is about.

November

The TI-Faire will be a few weeks after this meeting so it may be taken up with the organizational requirements of this big day. New software and hardware to be demonstrated. Watch this space for more details. Time to think about nominating for positions on the board. I am sure there will be some vacancies this year!

December

The Annual General Meeting followed by some festive eats and drinks. There will probably be a bit of celebration after the TI-Faire, if we are all still friendly after the event. Make sure that you attend and give your support to all the workers in the club. o