99'er NEWS DIGEST

Focusing on the TI-99/4A Home Computer

Volume 7, Number 5 **June 1988**

Registered by Australia Post - Publication No. NBH5933

BBS Simulation

Extended BASIC Tutorial

TI BASIC Tutorial

Linking to Assembly from Extended BASIC

Implanting Assembly Programs in Extended BASIC

Console Fault Diagnosis

Programming Problems Diagnosis

Clock Construction

Copying Programs

Morning Tea and Lunch for $2

Demonstration of TELCO Terminal Emulation Program

Mini-PE RAMdisk Demonstration

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

TIsHUG (Australia) Ltd.

TIsHUG News Digest

June 1988

All correspondence to:

P.O. Box 214
Redfern, NSW 2016
Australia

The Board

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

Sub-committees

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

Regional Group Contacts

Carlingford
Chris Buttner (02) 871 7753
Central Coast
Russell Welham (043) 92 4000
Coffs Harbour
Keir Wells (066) 55 1487
Glebe
Mike Slattery (02) 692 0559
Illawarra
Bob Montgomery (042) 28 6463
Liverpool
Larry Saunders (02) 644 7377
Northern Suburbs
Dennis Norman (02) 452 3920
Sutherland
Peter Young (02) 528 8775

Membership and Subscriptions

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

TIsHUG Sydney Meeting

The next meeting will be at 9 am on 4th June at Woodstock Community Centre, Church Street, Burwood. It will be a full day with tutorials, workshops and lunch provided in the admission charge of $2.

Printed by
The University of Wollongong
Printery

TIsHUG News Digest ISSN 0819-1984

Index

Title	Description	Author	Page #
Changing colour of characters	Software hints	Nollan,Joe	22
Changing defaults	Software hints	Samouri,Mel	22
Communicators	BBS information	Mudie,Ross	5
Correction	General interest	Hope,Greg	20
Disk DSR tutorial	Software hints	McCormick,Mack	23
Disk controller compatibility	General interest	Scheidemantle,Paul	11
Disk drives and interlaces	General interest	Arnold,Tom	20
Extended BASIC tutorial	Software hints		10
Fixing disks with disk fixer	Software hints		19
Forth column	Forth forum <2>	Smyth,George L	25
From the bulletin board	Mail to all		5
Games information	General interest	Brown,Robert	13
Games information	General interest	Judd,Stephen	13
Geneve	General interest	Boyer,Jerry	8
Guide to UCSD p-System	General interest		27
Hitch hiker's guide to galaxy 3	Adventure hints	Brown,Robert	18
Hitch hiker's guide to galaxy 3	Adventure hints	Judd,Stephen	18
Holy rules of arithmetic	Software hints	Takach,Ben	2
Letter to the editor	Software hints	Meldrum,George	29
Link it #16	Software hints	Mudie,Ross	6
Myarc 9640, who cares?	General interest	Mickelson,Steve	7
Myarc mystery	General interest	Takach,Ben	29
Odds and ends	Software hints	Samouri,Mel	24
Program to type in	Diet manager		15
Program to type in	Library printer	Welham,Russell	17
Programming hint	Sprite routine		20
Publications library report	Club library	Welham,Warren	3
Regional group reports	General interest		31
Secretary's notebook	Club news	Phillips,Terry	3
TIsHUG publications library	Club library	Welham,Warren	14
TIsHUG shop	For sale	Bunbury,Bob	4
TIsHUG software column	Club software	Phillips,Terry	4
TIsHUG tutorial day	Club news	Mudie,Ross	2
Techo time	Hardware review	Paine,John	11
They're off	General interest	Trott,Geoff	1
Tips from the Tigercub #46	Software hints	Peterson,Jim	9
Using multiple UTIL1 files	Software hints	Samouri,Mel	21
Video modes for TI99/4A	General interest	Ness,Jim	12

They're off

by Geoff Trott

Welcome to all of you brave souls that decided to stay with us for one more year. I hope it is the best yet. To continue my tale of woe about the May TND, we had problems at the printery as well. Normally I take the paste up in before Wednesday of one week and pick up the magazines on the morning of the Wednesday of the following week. In line with all the other problems of last month, they had not even started printing on Wednesday when I called in to find out when I could pick them up. After much to-ing and fro-ing they were ready on the Thursday morning, so with a bit of help, they were in the hands of Australia Post by 11 am. Hopefully, members who live in Sydney received theirs on Friday, in time for the meeting. Hopefully we will not have as bad a series of problems in future months, as we should not have as much rain again this year either!

continued on page 30

TIsHUG Tutorial Day 4th June 1988

by Ross Mudie

TIsHUG will be holding a full day tutorial workshop from 9am to 4pm on Saturday 4th June 1988 at Woodstock Community Centre, Church St, Burwood.

Activities will include a Bulletin Board Service (BBS) simulation, BASIC and Extended BASIC tutorial, linking between Extended BASIC and Assembly, implanting multi part assembly programs in Extended BASIC, disk copying of Software Library disks by request, programming problems diagnosis, faulty console diagnosis and assistance with assembly of clock kits. Mini-PE systems will be on display, the club Shop will be operational and the Publications Library will be available. Morning tea and a modest lunch will be provided in the $2 registration fee.

1. BBS simulation.

There will be a computer running the TEXPAC BBS program and another connected to it via a direct cable to simulate the user's terminal. The terminal will use the new TELCO program. This will allow members who do not use the BBS to view it first hand. Advice on what equipment is needed to access the club's BBS will be available. Help with problems which existing users are experiencing will also be available.

2. BASIC and Extended BASIC Tutorial

This will be presented by Ross Mudie, who will tailor the session to the needs of the group who come along on the day. It is intended that beginners are catered for in this group. Bring your reference manual, note pad and pen.

3. Linking to Assembly from Extended BASIC.

This tutorial will be presented by Craig Sheehan using the XDP Display routine as the program example. This is an extremely effective way to lead into learning assembly. Your Editor Assembler manual, note pad and pen are recommended.

4. Implanting Multi part Assembly Programs in Extended BASIC

This is an advanced group activity which will be based on articles written by George Meldrum in November 1987 TND and Ross Mudie in March 1988 TND. The presenter for this activity is not finalised.

5. Copying of Software Library Disks.

Terry Phillips will bring the club's software library and a system for copying. There will be no charge if you supply your own, preferably initialised disks. Shop software of the day will not be available for copying and copying onto cassette tapes will not be possible due to the time required.

6. HARDWARE DEPARTMENT.

John Paine will bring test equipment to permit diagnosis of faults in consoles. John will also guide constructors of the club's latest project, the Time of Day Clock kit. Bring your own tools, solder and soldering iron for this one. Kits are available from the club shop but contact Bob Bunbury from the shop in advance to avoid disappointment.

7. Program problem diagnosis.

If you are having a particular programming problem then bring your program along on a disk or cassette and on paper if possible. Russell Welham and others will attempt to give you a helping hand.

8. TIsHUG Shop.

The shop will be open as usual on the day, but if you plan a large purchase of items such as disks etc, then please speak to Bob Bunbury in advance to avoid disappointment.

9. Food arrangements.

The usual tea and coffee will be available at morning tea and lunch when hamburgers will also be available. The $2 registration fee will cover tea, coffee or drink and hamburger. As previous participants at these days will know, it is nothing fancy but cooked on the outside BBQs in the spacious lawns of Woodstock, but it really hits the spot.

10. Mini expansion systems.

Peter Schubert will be demonstrating the AT Mini Expansion system for the TI99/4A. This great little expansion system can provide 32K memory, RS232 and PIO, controller for up to 4 DSDD disk drives, RAMdisk and operating system enhancements.

11. Publications Library.

The publications library, which contains a great deal of TI99/4A information and overseas user group magazines, will be available for your perusal and loans by library members.

With all these planned activities all that is needed is a good roll up from the membership. Country members and people who do not regularly attend the TIsHUG Sydney meetings are asked to make themselves known to Co-ordinator, Chris Buttner, so that we can help make you welcome and your visit really worthwhile.

The Holy Rule of Computer Arithmetic

and other traps for young players

by Ben Takach

Hierarchy, according to the dictionary, has nothing to do with computers nor with the evaluation of mathematical expressions. The greek translation means holy rule (rule like in government not regulation or law). No wonder we mortal sinners do transgress the holy rule every now and then, and then all hell breaks loose! Actually in every form of government, holy or otherwise, there is a certain amount of anarchy and chaos. There is a certain amount of anarchy and chaos also in the computer and calculator family with respect of the arithmetic hierarchy. The rules are by no means uniform.

Let us deal with the less obvious first. Some computers will not respond to PI unless it is DEFined before it is used the first time in a program. Some will respond only if it is entered in capital letters, others do not care. Some will work in DEGree, RADian or GRAD mode as selected by the user, others will only work in one of the three (usually radian). This can lead to some frustrating and hard to find errors when a known, good working program is transcribed to an other brand of computer. I have spent an absolutely infuriating evening attempting to debug a program, which unknown to me, on the original computer displayed the results in degree, minute, second format (DDD.mmss). An other nasty trap is the inconsistent default format (the TI-59, TI-58 and the TI-CC40 will work in all three formats, the 58 and 59 default to DEG format, the CC40 defaults to RAD).

After this excursion let us return to the holy rule of computer arithmetic. Take a respectable and reputable scientific calculator like the GE-CE92 and enter: 18-2x4. The result will be 64. Repeat the same with the TI99/4A and the result will be 10.

Of course one would never never write such ambiguous expressions unless one happens to be the Federal Treasurer.

The GE sees the expression as (18-2)x4=64, and the TI99/4A calculates it as 18-(2x4)=10. Using the TI99/4A, you can get what you wish, provided you use the brackets. If not certain of the hierarchy then use brackets, it does not mind peeling off a number of superfluous brackets. Let us solve the expression 1-45/50. Any calculator will display -0.88. The TI99/4A prints (+)0.1, e.g. -45/50+1. If the expression is keyed in the calculator in this order, then it will also evaluates it to 0.1 e.g. |+/-| 45 |/| 50 |+| 1 |=| 0.1.

Programs are written like love letters. Each reflect the style of the individual, each is great at the time of confinement, but one would not write the same stupid nonsense years later. However where is the universal critic to judge the merits of an individual programming style? If it works, and you are happy with it then it is fine! If it becomes your favorite then you will change it frequently anyway. Hence the never ending versions of many good programs. Nevertheless some sound programing rules have to be adhered to, otherwise the program is unstable. It is just a "fine weather" program which will crash when the going gets rough.

Secretary's Notebook

by Terry Phillips

A big welcome to only one new member who has joined us in the past month. He is David Corney from Lapstone. Welcome aboard David and hope we see you at some of the meetings.

Speaking of meetings, here is what is lined up for the next 3 months:

June 4 - Full day Tutorial

This big day will be one to remember. There are a host of activities planned including Linking to Assembly with Craig Sheehan, Extended BASIC for beginners to moderates with Ross Mudie, a BBS simulation using Telco, demonstration of TI Keys and Easy Writer by Larry Saunders, software copying and maybe even a demonstration of the Geneve. Lunch and refreshements will be available throughout the day for a modest fee, and as usual the shop facilities will be available. Start time: 9am.

July 2 - General interest

This will be a normal, 2pm start meeting, where if the GENEVE has not already been demonstrated at the June meeting, it should be available at this meeting. OK, so you came along to the June full day tutorial and went home with a wealth of knowledge, but now just what did the tutor mean when he said .....? This is what we call a follow up tutorial when you get the chance to ask, and hopefully receive an answer, to those nagging questions. Do not be shy. The quickest way to learn is to ask questions.

August 6 - Swap meet/market day

Got some old modules, software or hardware you do not use anymore? Chances are your unwanted gear is wanted by someone else and this is your chance to sell it, trade it or give it away. This meeting will start at 2pm so bring along your unwanted items and perhaps go home with a few extra $$s in your pocket.

Not much in the line of correspondence in the past month, but members may be interested in knowing that the Group has sent an invitation to Lou Phillips, President of Myarc, to see if he can join us in Sydney after his Brisbane Faire visit. We will keep you posted on his response. If he does come to Sydney then we would have to organise a special meeting.

Bob Hodges, telephone (02)449 4273, has written to say that he has a range of modules for sale that he no longer uses. Titles are Household Budget Management, Munchman, Alpiner, Indoor Soccer, The Attack, Parsec, Moon Mine, Star Trek, Car Wars, Music Maker, Video Graphs, Number Magic, Milliken Fractional Numbers, Addition and Subtraction 2, Scott Foresman Pyramid Puzzler Maths. Each of these at $10. At $5 he has TI Demo Module, Teach Youself BASIC and a strategy game called Airline. At $25 he has TI Artist plus 4 disks of artwork, while at $15 he has the Tunnels of Doom Module with data bases on cassette and disk. Anyone interested should of course give Bob a call.

As of this writing, we have 148 members who have renewed for another year, but I am hopeful that the next 2 weeks will see many more renewals flowing in.

I should also apologise for the bit of a mix-up with the May meeting. Those on the BBS probably knew that the meeting was our 7th Birthday event and turned up accordingly at about 12 o'clock to partake of the hot dogs and cold drinks. Unfortunately I had committed my report to the Editor prior to firming up the arrangements for this event, so there was no notice of it in last months TND. As I say, my apologies.

Do not forget, the next meeting is our first full day tutorial for this year, and with 4 rooms booked all day at Woodstock there will be plenty of room to spread out and participate in the planned activities.

Many of you are currently in the process of building clock cards, If you are having any problems then bring along your kit plus your soldering iron to the June meeting where experts will be on hand, not to do the job for you, but to give you some guidance in proper construction.

Publications Library Report

with Warren Welham

Hopefully in this month's TND, there should appear a complete list of books and TNDs in the publication library. Down the side of this list is a code number. Using this, interstate or people who cannot make it to the main monthly meeting can order books or TNDs. To order the items you desire you can either:

A)Ring me up (phone number on the inside cover of TND);

B)Leave a message on the BBS (username LIBRARY); or

C)Write to me via the post (address on the cover of TND).

If you wish to borrow the interstate and overseas newsletters, use the previous steps and just state the name of the club you want. Remember postage for these items must be paid by the member.

Is there anyone one who can speak French and is willing to translate some of the magazines from Montreal. If so please see me at the next meeting or write to me.

I would like to put the overseas newsletters in some sought of folder, so that people can borrow single issues. The problem with this is that most of them come from America and therefore use American Quarto (8.5 x 11 inches) sized paper. I cannot buy plastic envelopes for this size paper in Australia. Has anybody out there got any ideas on how to house the overseas magazines, if so please get in contact with me with your idea and if you could also let me know how much it would cost.

New Books arrived this month

Code	Title	Author
00220	TI program library 1982	T.Instrument
00221	Suggested retail price list 1982	T.Instrument
00222	Hitch Hikers Guide to the Galaxy Solution	- Infocom

New Arrivals of Overseas Publications

Group	Publications Name	Date
LA 99er Computer Group	TopIcs	Apr 88
Channel 99 Hamilton	Clubline	Mar 88
Channel 99 Hamilton	TI Focus	Apr 88
Hunter Valley 99ers user group	Hunter Valley News	Apr 88
Micropendium	-	Apr 88
North West Ohio	99'er News	Aug 87
North West Ohio	99'er News	Oct 87
North West Ohio	99'er News	Nov 87
North west Ohio	99'er News	Jan 88
North West Ohio	99'er News	Feb 88
North West Ohio	99'er News	Mar 88
Brisbane User Group	Bug Bytes	Apr 88
Central Ohio Ninety Niners Inc.	Spirit of 99	Apr 88
San Diego Computer Society	-	Mar 88
San Diego Computer Society	-	Apr 88
Club Information	-	Feb 88
Montreal	Cim 99	Apr 88
Lehigh 99'er Computer Group	-	Feb 88
Lehigh 99'er Computer Group	-	Mar 88
Lehigh 99'er Computer Group	-	Apr 88
Adelaide Computer Club	Aticc	Apr 88
Canberra Home Computer Group	Chug-a-lug	Feb 88
Edmonton 99'er Computer User Society	99'er Online	Feb 88
Edmonton 99'er Computer User Society	99'er Online	Mar 88

TIsHUG Software Column by Terry Phillips

And the winner is .....

Yes the Judges have reached their verdict and the announcement was made at the May meeting.

Overall winner is George Meldrum for his entry MERGE. This exciting utility program, primarily designed for cassette users, lets the cassette programmer MERGE programs similar to the Extended BASIC disk command of the same name. MERGE also allows the disk user to load very large cassette files that normally would not load even after CALL FILES(1). After loading, resaving to disk is possible. MERGE requires Extended BASIC and 32K memory expansion in any one of its configurations.

Congratulations George for programming an exciting piece of software.

Highly commended by the judges were the following entries:

1. A series of entertaining Extended BASIC games from Barry Gibbins. Barry's entries, TILO, WAIT-A-BIT, CATHAY, DRILLCREEK, POKER and TRIANGLE contain excellent graphics and strategy. Barry programs on only a "bare-bones" system, so all his programs will run without the benefit of memory expansion, although some do stretch the 16K console memory to its limits.
2. An excellent game, MATCHMAKER, from Tony Smith. The object of this game is to come up with matching pairs from a series of 50 hidden shapes. You play against the computer. What makes this game different is the animation of the hidden shapes. Also included is a separate program for the user to design his own animated shapes or characters. These 2 programs will run in either console or Extended BASIC and also do not require any memory expansion.
3. A FAMILY TREE data base program from Bill Longmuir. This 2 part program, introduction and main program, is as the name suggests, a program for keeping track of your family tree. With renewed interest in this area through the Bicentennial, I am sure a lot of members will be able to put this program to good use. Although it is designed to run only in Extended BASIC without memory expansion and in its current configuration is set up for cassette storage and retrieval of data, it can easily be modified for disk use and to accept more than the current limit of 50 entries.

Congratulations go to all these authors, as of course they go to all the other authors who, although they did not win a prize, took the effort to enter the contest. The main objectives of the contest have been realised, we do have software authors with exceptional talent, and we now have a good base of new and interesting software to trade with our friends, both local and overseas.

New disks received, and to be released at the June meeting will be:

DISK A189 - C99 version 4.0. This is the very latest version from Clint Pulley which is sure to be of interest to all who wish to program in this language. Is there any one out there who is dabbling in this language? MICROpendium has been running an interesting series of articles on "C" over the past few months. If you have not read these articles, then do so, and with this disk away you go. Needs 32K memory expansion and at least one disk drive.

DISK A201 - CATLIB V1.5. The latest update from Marty Kroll. This version is in memory image format and will load from a host of environments. On the disk are versions for EPSON/STAR, and OKIDATA printers plus a further version for you to set up for other brands of printers. A 106 sector documentation file is included on disk. Needs 32K memory expansion and at least one disk drive, although two are recommended. Printer optional, although if you want a printed listing, it is of course a necessity.

DISK A202 - CATLIB COMPANION V1.0. Also from Marty Kroll. The companion utility allows you to manipulate your files created with the main program and add descriptive text. Required options are the same for the main CATLIB program.

Disk numbers A190 to A200 have been reserved for software competition entries and have not yet been fully indexed but will be by the time of the June meeting.

A tape of programs will be available at the June meeting and will contain some of the competition entries from Barry Gibbins, Tony Smith and Bill Longmuir. See above for descriptions. This will also be available on disk and will be titled COMP-GAMES. The overall winning entry, MERGE, from George Meldrum will also be available on tape and disk.

Following a number of requests, I will run off some disks containing DV80 files of the club software library contents. I will also run off a few hard copies, and these will be available at the June meeting.

Do not forget that at the June meeting there will be ample time throughout the day for you to request software from the library. The complete library will be available for you to select from, with the exception that software released through the shop on that day will not be available for copying. o

TIsHUG Shop

with Bob

Great news from the SHOP this month. With the help of John Paine, Cyril Bohlsen, Leo Gorniak and Peter Schubert, we managed to supply PCBs for the clock and RGB monitor projects at the May meeting at Woodstock. Further supplies are in the pipeline! We also had a supply of the necessary part to complete the clock. $20 a set, while stocks last. I am sure we can get mmore parts, but price rises are a possibility. A persistant pesky problem! If you are having difficulty in contacting SHOP by 'phone or BBS, do not give up. Write in. Same address as the rest of the gang - PO Box 214, Redfern, NSW 2016. And, while I am thinking on the subject, thanks to both the Sutherland and Banana Coast Regional Groups for your comments on the functioning of the shop. We do try to provide a quick turnaround service to all orders. I often fail in my execution of your requests. If anyone out there has the room, the time and the inclination to take over the fun slot of our Group, Please put yourself up for nomination, now. I will gladly truck the stock, the paperwork and my half written text on shop management made easy, to your address. Before the ink dries on your nomination. I think the bugs are just about all shaken out of my system, and service is improving. Please jog me again if I am still too slack for your tastes. Next month, I intend to have a list of all the outstanding orders/requests in the shop files. Please watch out for this. Even if you do not have any orders, or wishful requests in, you may see a request you can fulfill, or an item you would like too. A timely response might then see a bulk buy being financially and practically possible. Gerry and/or Cyril will be heading the shop team for the next two monthly meetings. I will be interstate on those weekends only, so mail orders will still be OK. Phone orders will be pretty risky for the next three months. Stocktake time, and some new equipment at work will be keeping me from home. o

'I programmed it to show company profits for the next twelve months, and it turned itself off!'

The Communicators

Special Interest Group for Users of the TEXPAC BBS.
by Ross Mudie, 8th May 1988.

1. FILE CHANGES FOR REGIONAL GROUPS.

Early in May 1988 two regular columns of the BBS NEWS menu were discontinued. These were MEETINGS and BULLETIN. They have been replaced by giving each regional group a separate user name with sub-editor status. This means that each Regional group has full control over its own file in the NEWS area. File names and regional groups are as follows:

Name	Regional Group	Name	Regional Group.
BANANA	Banana Coast	CARLO	Carlingford
CEN-COAST	Central Coast	GLEBE	Glebe
ILLAWARRA	Illawarra	LIVERPOOL	Liverpool
NEPEAN	Nepean	NORTHERN	Northern Beaches
SUTHERLND	Sutherland	TISHUG	TIsHUG

The purpose of the change is twofold. It reduces the SYSOP workload and more importantly gives each regional group its own file space in the BBS where new messages are readable as soon as they are loaded.

2. BBS DEMONSTRATION AT TUTORIAL DAY, 4th June 1988.

No, the BBS is not going to picket the front door with banners! There will be a computer running the TEXPAC BBS program and another connected to it via a direct cable to simulate the user's terminal. The terminal will use the new TELCO program. This will allow members who do not use the BBS to view it first hand. Advice on what equipment is needed to access the club's BBS will be available. Help with problems which existing users are experiencing will also be available. This demonstration will allow users to view the BBS in operation from the system end, including the logging of operations which assists in resolution of problems.

3. USAGE.

The BBS maintains a log file to show who uses the BBS and when. The following information is derived from the logging file by a simple sort program.

TEXPAC USAGE 1988.	January	February	March	April
CALLS	393	329	328	302
USERS	49	61	52	59
Occupancy Hrs.Min	156.26	119.39	123.40	120.13

Invalid calls are not included in this summary.

4. PHONE NUMBER and HOURS OF OPERATION.

The BBS operates unattended 24 hours per day, 7 days per week on (02) 319 1009. It is only accessible to registered, financial users who have been issued with a user number, user name and password. Registration enquiries can be made to the Secretary or Sysop at PO Box 214, Redfern, 2016. Phone numbers are shown on page 1 of the TND magazine.

From the Bulletin Board

MAIL TO : ALL
MAIL FROM : SHANE

Greetings! Here is a short list of some other Bulletin Boards, which you might like to check out...

" LANDOVER BBS " . . .3191793
" MILLAWAYS BBS " . . .3577027
" EAGLE ONE " 7453190
" EAGLES NEST " 4510535

MAIL TO : ALL
MAIL FROM : LARRY

Has anyone have TELCO yet. If so please let me know.

Bye for now LARRY.

Will do a full demonstration of EZ-keys at the next Liverpool Regional group meeting. It will do almost everything that Super Extended BASIC cartridge does plus a lot of other things, for example, 55 program keys of up to 655 characters, Macros that can be linked into a chain, highlite, full on screen editing like Super Extended BASIC, save program every 1 to 18 minutes to disk.

Here is an example of highlite type.

```
10 CALL INIT
20 CALL LOAD(16128,2,224,38,0,2,0,8, 17, 2, 1,63,36,2,2,0,3,4,32,32,36,2, 224, 131,1 92,3,128)::
CALL LOAD(116164, 240, 240,240)
30 CALL LOAD(-31804,63)
```

That will give you a rough example of highlite but it is not nearly as good as EZ-keys highlite.

MAIL TO : ALL
MAIL FROM : LARRY

Just released: Harry Wilhelm's Beyond Video Chess. BVC is designed to enhance the Video Chess module or disk (and not replace it) in a number of subtle, but useful ways for the dedicated computer chess player. BVC will allow you to control the pieces on the board with a joystick (and if you have the TI joysticks, it allows two live opponents to use Video Chess as simply as an electronic chessboard), save and load games to disk, list out all of the moves to an Epson or compatible printer, and print out any screen in the module or disk game. In short, BVC is for serious students of the game, or those that would like to be.

BVC requires 32K, a disk system, Video Chess, and one of the following : Widget and Editor/Assembler or Mini-Memory; Horizon RAMdisk; Myarc or Corcomp disk controller; or Load interrupt switch.

The revolution continues. As from May 15th, Version 2.0 of EZ-keys+ includes a new "input" macro function that works almost exactly like the "hold" function. It also features a built in assembly disk cataloguer, an assembly screen dump available at the touch of a key, and automatic checksum checking of anything typed in from a magazine. It will allow 8K of assembly routines in addition to itself, it will not "stomp on" any other interrupt driven routines loaded with EZ-keys. You can put an Extended BASIC program in low memory and fill high memory with assembly routines (the possibilities here are staggering!). There is a good possibiltity you will be able to use it with PRbase, Funnelwriter, etc.

I will demonstrate at the July Liverpool meeting both programs.

Bye for now, LARRY.

MAIL TO : ALL
MAIL FROM : PETESAKE

Hoorah for the news article on the Geneve. My sentiments too. As a hardware developer myself, I know what it is like to deliver a new product onto the market place and how lucky I feel when the teething problems are only minor. For one person or company to produce a complex piece of hardware without outside help is almost impossible these days, as the electronics designs become so complex, especially as so much software is also incorporated within the hardware peripherals of our TI99/4A (unlike most other computers), necessitating a team of both hardware and software developers to work together. If we have the patience to wait for results and improvements a lot more can still be accomplished.

MAIL TO : ALL
MAIL FROM : PETESAKE

My first new RAMdisk peripheral is now up and running, following testing of the prototype Mini-PE RAM board. I call this the MiniRAM+ and it has the following features:

128K expandable to 512K of battery backed RAMdisk fully buffered bus design

44 way connector to console expansion port, or stackable onto Mini-PE boards

3 way power option with provision for Regulated power on board with AC/DC Plug Pack. (Can also power full Mini-PE system) Optional Real-time Clock on board

CRU address can be set to 1000, 1200, 1400, 1600, 1800, 1A00, 1C00, 1E00.

Optional Clock and/or extra memory can be added yourself as your budget permits.

If you are interested contact me on 358 5602 or via BBS as I need to estimate the number of boards to order.

Coming soon !!

A.T. RAM+ board for the PE box.

Also including the clock, this board will come with the basic logic fitted, but no memory. Fit it yourself as you wish at 512K bytes per layer, a max of 2M bytes recommended. Yes this board has 16 socket positions plus ROS/EPROM, so 1M byte is only 2 layers of chips.

As with the Mini+ RAMdisk already mentioned, this can also have the ROS in either RAM or EPROM.

TI99/4A, the machine that keeps on growing...
Regards, PETESAKE.

Linking Extended BASIC to Assembly

16 LINK IT 16

with Ross Mudie

This program is written in response to a request from TIsHUG member, Robert Brown, who is writing an Adventure type of game. Robert has been running into problems with memory space and the slowness of Extended BASIC when comparing an input command word against a list of command words.

The purpose of this program is to allow a very quick comparison to be made between a word and a list of words which are resident in the assembly memory area. The purpose is to find if the word, which may be a command word in an adventure type of game, is valid and if it is valid, to return a number corresponding to its position in the list. The number is then used in the Extended BASIC game in either ON X GOSUB or ON X GOTO to carry out the required program flow.

The program provides two entry points which allow two lists to be tested, NOUNS and VERBS. The string is first transferred into the assembly BUFFER using STRREF and its length is checked against the length of the first string in the appropriate list. If the lengths are not equal then the next string in the list is checked for a length match. If a length match occurs, then the program compares the letters one at a time. If a perfect match is found, the program outputs a value which is the numeric position of the word in its list. If an invalid result is obtained then a value of 1 is returned which takes the first branch of the ON X GOSUB or GOTO.

```
* EXTENDED BASIC DEMO PROGRAM.
* 100 ! SAVE DSK1.LOAD
* 110 CALL INIT
* 120 CALL LOAD("DSK1.O")
* 130 INPUT "Word ? >":S$
* 140 IF LEN(S$)>15 THEN CALL SOUND(100,150,0) :: GOTO
*     130
* 150 CALL LINK("NOUN",S$,P) ! NOUN can be changed to
*     to VERB for other list
* 160 PRINT S$;P;:: IF P=1 THEN PRINT "*Invalid": :ELSE
*     PRINT : :
* 170 GOTO 130
*-------------------------------------
* CALL LINK("NOUN",S$,P)
* CALL LINK("VERB",S$,P)
* File names Source=N   Object=O
       IDT  'SEARCHmu'  by Ross Mudie, 17th April 1988.
       DEF  NOUN,VERB
NUMASG EQU  >2008
STRREF EQU  >2014
XMLLNK EQU  >2018
BUFFER BSS  16              One more than max len of S$
WS     BSS  32
SAVRTN BSS  2
B15    BYTE 15                        Sets max len of S$
       EVEN
* Register Usage.
*  R0    Element in link
*  R1    Argument in link, Len byte
*  R2    Buffer address
*  R3    Word counter
*  R4    Position in words
*  R5    Buffer pointer
*  R6    Word length
*  R7    Used for TOTQTx, where x is N or V
*  R8
*  R9
*  R10
*  R11   Return address
*
NOUN   MOV  R11,@SAVRTN              Save return address
       LWPI WS                       Register work space
       LI   R4,N1          Address to start in noun list
       MOV  @TOTQTN,R7
       JMP  START
VERB
       MOV  R11,@SAVRTN              Save return address
       LWPI WS                       Register work space
       LI   R4,V1          Address to start in Verb list
       MOV  @TOTQTV,R7
START
       MOVB @B15,@BUFFER       Specify max length of S$
       CLR  R0                          Element zero
       LI   R1,1                          Argument 1
       LI   R2,BUFFER                Where to put S$
       BLWP @STRREF                Get string from S$
       LI   R3,0                         Word counter
NEXTW  LI   R5,BUFFER    Address of BUFFER placed in R5
NEXTN
       INC  R3                           Count words
       CB   *R4,*R5       Does len of word in N's match
       JEQ  LENOK                   the one in BUFFER
       MOVB *R4,R1                        Len byte
       SRL  R1,8                    Make right justify
       INC  R1                       For the len byte
       A    R1,R4         Address of next word len byte
       C    R3,R7                        Is that all?
       JEQ  NOTFND
       JMP  NEXTN
LENOK
       INC  R5              Point to next char in BUFFER
       MOVB *R4+,R6     Put length of current data word
*                                   in R6 as a byte
       SRL  R6,8          Make it a 2 byte value (word)
NEXTL
       CB   *R4+,*R5+                Compare letters
       JNE  NEXTWD   If no match goto next word in data
       DEC  R6             Word length (letter) counter
       JNE  NEXTL     Any more letters in current word?
       MOV  R3,@>834A         Put the value of the word
       JMP  FOUND                    counter in FAC
NEXTWD
       DEC  R6                       Letter counter
       A    R6,R4  This makes R4 the address of the len
*                byte of the next word in the data area
       C    R3,R7                        Is that all?
       JEQ  NOTFND
       JMP  NEXTW
NOTFND
       LI   R1,1              This is the 1 for invalid
       MOV  R1,@>834A                Put it in the FAC
FOUND  BLWP @XMLLNK      To convert Integer to Floating
       DATA >20                  point value in FAC
       CLR  R0                          Element zero
       LI   R1,2                     Link argument 2
       BLWP @NUMASG Assign the number to the second arg
*                                          in link
       LWPI >83E0             Switch registers to GPLWS
       CLR  R0                             No errors
       MOVB R0,@>837C
       MOV  @SAVRTN,R11              Get return address
       RT
*   DATA area for game.  This is where you give all the
* words that you want to test in lists.
* You MUST ACCURATELY count the number of letters in
* each word and placing that in the byte before the
* TEXT word.
* Maximum 15 letters per word as the program is written
TOTQTN DATA 7          Describes the quantity of words
*                                  (Nouns) following.
N1     BYTE 7
       TEXT 'INVALID'
N2     BYTE 3
       TEXT 'CAT'
N3     BYTE 5
       TEXT 'MOUSE'
N4     BYTE 3
       TEXT 'COT'
```

The Myarc 9640, Who Cares?

An Editorial about the Geneve and the TI99/4A Community

by Steve Mickelson, USA

As both newsletter Editor and TI99/4A user, I have the opportunity to speak with many TI99/4A users and to read many editorials and comments in the various newsletters, databases, and BBS' about the state of our computer and where the future will lead us. I am disturbed by much of the "negativism" which seems to exist among many users about Myarc and the 9640 Geneve, much of which seems to reflect either an ignorance of the product or a totally unrealistic understanding of the financial resources of Myarc and the costs of producing a computer and software in the TI99/4A market place.

First the costs of developing computer hardware today can be prohibitive. Many Geneve critics somehow expected Myarc to introduce the Geneve and its operating system software simultaneously. Any one with any knowledge of software knows that you cannot write software until the hardware exists. And that such software will take time. Also, the fact that initial Geneve software consisted initially of TI-writer and Multiplan in 80 columns, with a modified GRAM Kracker type loader. So what is the big deal? It seems that these critics, want no compatability with TI99/4A software, if you follow their line of so called logic. Just who's camp do they support anyhow?

It was exactly one year ago that I wrote an editorial, whereby I drew an anology of the TI99/4A community being like the passengers aboard the Titanic, abandoned by its crew, (TI), and in danger of sinking from the flood of words amoung the "TI99/4A" vs "9640" camps of survivors. The only apparent alternative, seems to abandon ship and move to another computer, which is not an orphan. If we were to project ourselves 70, or so years in the future, what sort of conclusions would a future computer historion draw about the eventual fate of our community? Research would be made, relying on the opinions of the few, if any, remaining users, documents such as dusty old newsletters, and publications, such as dated editions of The Orphan Chronicles, Computer Shopper and MICROpendium would be consulted. How historically accurate a portrayal of the facts is really questionable, to this observer.

Certainly, there will be statement of the folly, or foresight of the parent company, Texas Instruments, who, after attempting to go head-to-head against Vic-20, lost a marketing battle, in which it had a little or no hope of survival. There would be some mention of how TI tried to put a stranglehold on Software development, by its use of GROMs and secret hardware architecture. And of how publication "experts" bowed to the superiority (?) of IBM PC, XT, AT, etc., as the ultimate personal computer. And how like sheep the lay user believed the so called experts, forgetting that X is an unknown quantity, and spurt is a drip under pressure, and sought only IBM or clones as the definitive example of a personal computer.

A passing line or two may describe how the TI99/8, TI99/2 and CC-40 were almost bought into production and how distributors, users and software houses were all orphaned in one bold, cruel decision by TI, to pull the plug on this fine computer for the home.

A chapter or so would be dedicated to the courageous efforts made by User Groups, commercial databases, and a handful of loyal third parties in the hardware and software area, not to mention a few publications to continue to promote and support the TI99/4A. Even TI, apparently embarrassed, by the refusal of its abandoned child to obediently roll over and die, creates a toll-free "care line" and releases some information about its turnkey, or should I say turkey hardware operating system. A public relations effort many say too little too late.

We would read of how new hardware and software were released, some, no doubt, created prior to the TI pullout, yet released with a hope of at least re-couping costs from a community of users growing more and more accustomed to fly the "Jolly Roger" and pirate everything in sight.

Another chapter would tell the tale of how undaunted, perhaps masochistic in light of its reception amongst many "survivors", company attempted to revive the community, by introducing a compatible upgrade, (to some) or downgrade (to others), which met with mixed feelings, at best, among many TI99/4A users.

We see, how some naive enough to mis-interpret Mr. Phillips' dress, (3 piece suits and a cleancut appearance), "pegs" the man in the category of TI. Since he dresses well, and acts with business like decorum, certainly he must have the Mega-bucks to blow, to hire software writers fo $30-$40,000 a pop, as TI did, for ready to use, spam-in-a-can software. When the community sees him taking a grass roots "appeal to the user for software", approach like Atari ST and Mac makers did, lynch mobs form. If only he wore a pizza stained sweat shirt and jeans, then perhaps they would be more accepting of buggy operating systems, as users had done with the O/S for the 1 Mbyte, Horizon RAMdisk! Or better yet, why not ignore the errors of Corcomp, extend Myarc's line of credit and push his fledgling company to the edge of bankrupcty, to meet the demands of users and non-users, alike, for instant software gratification? Why cannot he be as foolish as some of our expectations?

The final chapter would, using several theories, try to describe what factors lead to the sinking of the majestic TItanic.

These would include the blunder of attempting to market a device ahead of its time (TI99/4A), against a competition, (the Commodore Vic-20 and 64 line), which were no more than toys, in comparison. Still, the toy David, defeated its giant competitor, through shrewd use of marketing savvy and open invitation to software writers, as well as cheaper developmental costs. It could even be said that the User Group network postponed the inevitable demise. And how the community, used to one computer for five or so years, with its own limitations and the continued desire for cheap and easy software, spurned revival from a compatible, with recommendations from many of its own Guru's, to switch to IBM, Apple, Commodore, Atari or anybody, preferring to wallow in self pity, as an "orphan", than to see the success and succession of a new 9900-series computer.

Final conclusions may put the blame for the TItanic squarely on the shoulders of the users who instead of pulling for the common survival of the community at large, fought amongst themselves and let the pre-occupation with "Myarc bashing" and "Myarc support" destroy the community from within, like a cancer, a community, many of whom, forgot that even the "TI-creator" had problem children as the early TI99/4 and console BASIC, not to mention other software which used only cassette, thermal printers or had no method to I/O or print data. And such software was locked in GROM space. Perhaps this sleazy attitude among many users did more to push TI to abandon the TI99/8 and TI99/2, than any influence exerted by the forces in the market place.

If you do not get the drift of what I am saying: bluntly, it is that I personally cannot see why someone such as Lou Phillips, wastes his time and money on a community which has so many ingrates, who could not recognize something positive if it slapped them smack between the eyes. Are these individuals really as stupid as their remarks seem to indicate? That is to say, do they really think Lou Phillips has the backing of TI, Yamaha, or some wealthy investor syndicate; and is just too cheap to spend some of the billions of R and D bucks on his software? Are we, as users, too dim-witted to see that IBM sold a million units of its O/S2, introduced about the same time as the Geneve, before bringing out an O/S2 Operating System; and still expect Myarc, operating in a dead end market to have the finances to get software out in a shorter period of time than Big Blue? Let us be realistic in our expectations of any third party manufacturer. Perhaps TI listened to the carping from such users, and for that reason alone, pulled the plug on the home computer market.

continued on page 29

Geneve

by Jerry Boyer

part 1

Well, I finally received my new Geneve, (pronounced JIN-EVE). I had been calling around the country trying to find someone who had the Geneve in stock, but every company had the same answer, "We are only taking orders for the Geneve and you would be about number 150 or so. We will be shipping them out about 20 a week, which would mean about an 8 week delay". This was not good enough. I read an advertisement in MICROpendium from "Innovative Programming". They stated that they had the Geneve in stock. I called them immediately (800)255-2985. They said they had quite a few units in stock. I ordered it with the Enhanced Keyboard and 6 days later UPS delivered it, which seemed a miracle.

After reading the massive manual that came with it, and hooking it up as instructed, I soon discovered that nothing would work like the manual showed. I called Innovative Programming and talked to Galen A. Read, which was what Myarc had instructed me to do. Mr. Read was very courteous and attentive as I explained my problem. He said that the package must have been X-rayed and the software messed up. He stated that he would immediately send me another set, and I was to return the defective disks to them. Well, in two days they arrived in the mail. Needless to say, everything worked fine. Now that is what I call service. Mr. Read called me a day or so later to see how everything was working. I was pleased to see that he was very concerned about me and my problems. He showed what TI99/4A people are made of, no matter where you go.

The first thing you do is to transfer all of your modules to disk programs, much the same as Gram Kracker. In fact, any modules saved from Gram Kracker will work on the Geneve. The Geneve does not have any provisions for plugging in your modules. What they supply is a program called Cartridge Saver, which runs on the TI99/4A keyboard. It is very easy and very fast. I saved 40 modules to disks in less than 1 hour. I had the Widget cartridge expander which made it a little easier but the program works without the Widget as well.

The Advanced BASIC that comes with the Geneve is only a copy of Myarc's Extended BASIC II set up to run in 80 column format with a few new commands thrown in. I had a lot of trouble running my Extended BASIC programs. They would simply lock up the computer and I would have to turn it off and start all over again. This was remedied by loading my Extended BASIC program (saved to disk) and then running the Extended BASIC programs. No more lockup problems. Myarc will be sending out copies of Advanced BASIC when it is finished. The M-DOS and the My-Word programs are also not the final versions. They also will be sent out along with the Pascal interpreter that I did not receive.

This article was written with the My-Word word processor that is included. With 80 columns on screen all the time, It is a blessing to use. As I progress with my Geneve, I will try to find time to keep you all posted.

Part 2

Well I am starting to really love my Geneve computer. I have been using My-Word, the word processor, and it is really great. The fact that all 80 columns are on screen makes writing anything from just a small note to a full blown file a real pleasure. I only have a composite monitor so the letters bleed over a little but when I change the screen colours, I do not have any trouble reading the text. My-Word is almost the same as TI-Writer but with a great deal more enhancements, the best of which, I think, is that the formatter and the editor are in memory at the same time. You can go from editor to formatter and back to editor without losing the file in the editor. Also in the formatter, you can print the file that is in the buffer to screen or to the printer without saving the file to disk first. You can also view a file from a disk without disturbing the file you have in memory. This turns out to be very helpful while you are writing a file and you would like to check something on another file. At the bottom of the screen they have added a dotted line that is a copy of the tab set up line, showing where the Tabs and Margins are and there is a non blinking cursor showing where you are typing. This can be turned on or off from the Tabs command. Also in the left lower corner, you have the name of the file currently in memory. In the lower centre section you get a reading of how much memory your file is taking up in percentage used. This article used only 5%. That will give an idea of how large a file this baby will handle. Awesome. In the lower right corner the time is displayed. The Geneve has a built in time of day clock with battery.

The Geneve has two modes of operation, the TI99/4A mode and the 9640 mode. In the TI99/4A mode you have: 432K CPU RAM; 16K in Cartridge Space RAM; 64K GRAM; 32K High Speed RAM; 16K ROM. In the 9640 mode you have: 512K CPU memory; 128K VDP memory; 32K High Speed RAM; 16K ROM. As it is set up now, each mode has to be loaded separately from different disks. This is supposed to be corrected on later versions of the M-DOS systems disks. In the TI99/4A mode, you have a choice of 5 speeds. Mode 1 is slightly slower than the TI99/4A console with mode 2 a slight bit faster than the TI99/4A. Modes 3, 4 and 5 progress even faster with mode 5 being about a full 3 times faster than the TI99/4A. Most of the Extended BASIC programs have to use speed mode 1 for good control. In the 9640 mode you have only speed mode 5.

The GPL interpreter part of the TI99/4A mode loads your cartridges that you have saved to disk. In general most seem to work exactly as they did in the TI99/4A. Some do not. The ones I have that do not work are: Video Chess; Moonsweeper; Q-Bert; Buck Rogers; Pole Position; Slymoids. I am sure there are others that I have not tried. But for the most part, the Geneve operates and emulates the TI99/4A very well. I believe it to be the best thing to happen to the TI99/4A community since the TI99/4A became an orphan. I will keep you all posted as I progress with my geneve.

Part 3

Well, I hope Santa brought all the toys you wanted for Christmas, cause I am sure you are good little boys and girls. He brought me three new toys for my Geneve which I will review this month.

First, I got a disk called "JUMPBOOT" from Disk Only Software. The way this disk operates seems a miracle to me. "JUMPBOOT" is a specially formatted disk that takes advantage of the Geneve's fast multiple sector reading routines. When you first turn on the Geneve, it auto loads its operating system from DSK1. This operation took 42 secs on my machine to load the 358 sector file. With the same file transferred to the "JUMPBOOT" disk and using a Myarc disk controller card, it now loads in a mere 4 secs. When I used TI or CorComp cards the time slowed down to 7 secs loading time, still a lot faster than normal. The company claims the Geneve reaches its limit of 160Kbits/sec data transfer rate with "JUMPBOOT". The price is $15.95 plus handling charges.

The second thing I got was a "Speech Synthesizer Adapter card" from Rave 99. This card takes the board from inside the TI Speech Synthesizer and mounts it inside the PE box. The operation took only 3 minutes and it really works well. This card is a needed piece for Geneve owners since the Geneve has no place to plug in the TI Speech Synthesizer. The Geneve now talks for only $49.95 plus handling charges.

The third thing Santa brought was the "My-Art and Mouse" package from Myarc corp. My-Art is a great graphics drawing program. So far this is the first program to load directly from the M-DOS without having to use the GPL interpreter that is needed to load all the TI99/4A cartridges saved to disks. The My-Word program now loads from the GPL but they say that in the future you will be able to load it also directly from M-DOS. My-Art graphics support two modes, one is 256x212 pixels with 256 colours, and the other is 512x212 pixels with 16 colours. All the artwork is easily printed out in either single or double widths and either outlined or shaded in grey for coloured pictures. It is very easy to take control of this drawing program.

continued on page 26

Tips from the Tigercub #46

by Jim Peterson

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

Over 130 original programs in BASIC and Extended BASIC, available on cassette or disk, now reduced to just $1 each!, plus $1.50 per order for cassette or disk and post paid. Minimum order of $10. Cassette programs will not be available after my present stock of blanks is exhausted. The Handy Dandy series, and Color Programming Tutor, are no longer available on cassette. Descriptive catalogs, while they last, $1, which is deductable from your first order.

Tigercub Full Disk Collections, reduced to $5 post paid. Each of these contains either 5 or 6 of my regular catalog programs, and the remaining disk space has been filled with some of the best public domain programs of the same category. I am NOT selling public domain programs, they are a free bonus!

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

NUTS & BOLTS (No. 1), a full disk of 100 Extended BASIC utility subprograms in merge format, ready to merge into your own programs. Plus the Tigercub Menuloader, a tutorial on using subprograms, and 5 pages of documentation with an example of the use of each subprogram. Reduced to $15 post paid.

NUTS & BOLTS (No. 2), another full disk of 108 utility subprograms in merge format, all new and fully compatible with the last, and with 10 pages of documentation and examples. Also $15 post paid.

****
* NUTS & BOLTS (No. 3) is now ready, another full *
* disk of 140 new merge format utility subprograms, *
* all compatible with the previous utilities. With *
* 11 pages of documentation, $15 post paid. *
****

TIPS FROM THE TIGERCUB, a full disk containing the complete contents of this newsletter numbers 1 through 14, 50 original programs and files, reduced to $10 post paid.

TIPS FROM THE TIGERCUB VOL. 2, another diskfull, complete contents of numbers 15 through 24, over 60 files and programs, also just $10

TIPS FROM THE TIGERCUB VOL. 3, another 62 programs, tips and routines from numbers 25 through 32, $10 post paid.

TIPS FROM THE TIGERCUB VOL. 4, another 48 programs and files from numbers 33 through 41, also $10 post paid.

TIGERCUB CARE DISKS #1, #2 & #3, three full disks of text files, mostly of lessons on programming in Extended BASIC, $5 per disk post paid.

This one is explained in lines 180 to 190. I think that it will run on any Gemini printer.

```
100 DIM B(25,12),B$(25),CH$(12),L$(12)
110 GOTO 150
120 S,K,T$,C$,V,J,A,CH$(),X,X$,B$(), B(X,J),T,M,Q$,
    L$(),C,C1$,C2$,L,M$
130 CALL CLEAR :: CALL COLOR :: CALL SCREEN ::
    CALL CHAR :: CALL KEY :: CALL NUMTH
140 !@P-
150 !SEGMENTED BAR GRAPH by Jim Peterson 10/87
160 CALLCLEAR :: FOR S=1 TO 12 :: CALL COLOR(S,2,8) ::
    NEXT S :: CALL SCREEN(5) :: DISPLAY AT(3,10):
    "TIGERCUB" :: DISPLAY AT(5,6): "SEGMENTED BAR
    GRAPH"
170 CALLCHAR(95,"3C4299A1A1 99423C"):: 
    DISPLAY AT(7,12): "_ 1987" :: DISPLAY AT(9,2): "For
    free distribution but no": "price or copying fee
    may be": "charged."
180 DISPLAY AT(14,2):" Will output to a Gemini":
    "printer a horizontal bar-": "graph of up to 25
    bars, each": "segmented into up to 12"
190 DISPLAY AT(18,1):"values, with a title for": "each
    and optionally with a": "table of identification
    of": "the segment symbols."
200 DISPLAY AT(24,8):"" :: DISPLAY AT(24,8): "PRESS ANY
    KEY" :: CALL KEY(0,K,S) :: IF S=0 THEN 200
210 ON WARNING NEXT
220 DISPLAY AT(12,1)ERASE ALL: "GRAPH TITLE?" ::
    ACCEPT AT(14,1): T$ :: T$=RPT$(" ",
    17-LEN(T$)/2)&T$ :: C$=CHR$(27)
230 DISPLAY AT(16,1):"HOW MANY SEGMENTS PER BAR?" ::
    ACCEPT AT(16,27) VALIDATE(DIGIT) SIZE(2): V :: IF
    V=0 OR V>12 THEN 230
240 !@P+
250 DATA 239,229,168,251,173,175,184, 236,169,250,
    160,207
260 !@P-
270 FOR J=1 TO V :: READ A :: CH$(J)=CHR$(A) :: NEXT J
280 DISPLAY AT(3,1)ERASE ALL: "Type END when finished"
290 X=X+1 :: IF X>25 THEN 330
300 CALL NUMTH(X,X$) :: DISPLAY AT(12,1): "Title of
    "&X$&" bar?" :: ACCEPT AT(14,1): B$(X) :: IF
    B$(X)="END" OR B$(X )="end" THEN 330
310 FOR J=1 TO V :: CALL NUMTH(J,X$) ::
    DISPLAY AT(16,1): X$&" segment value?" ::
    ACCEPT AT(18,1) VALIDATE(NUMERIC): B(X,J) ::
    T=T+B(X,J) :: NEXT J
320 M=MAX(M,T):: T=0 :: GOTO 290
330 X=X-1 :: DISPLAY AT(20,1): "Print labels? Y/N" ::
    ACCEPT AT(20,19) VALIDATE("YN") SIZE(1): Q$ :: IF
    Q$="N" THEN 350
340 FOR J=1 TO V :: CALL NUMTH(J,X$) ::
    DISPLAY AT(22,1): X$&" label?" :: ACCEPT AT(24,1):
    L$(J) :: NEXT J
350 C=120/M :: C1$=C$&"B"&CHR$(1)&C$&"G"&C$&"E" ::
    C2$=C$&"B"&CHR$(3)
360 OPEN #1:"PIO",VARIABLE 255 :: PRINT #1: C$&"@" ::
    PRINT #1: C$&"E"&C$&"G"&C$&"M"&CHR$(6)
370 PRINT #1:CHR$(14)&T$&CHR$(20): "":
    RPT$(CHR$(229),70):;: :: PRINT #1: C$&"3"&CHR$(10)
380 FOR J=1 TO X :: PRINT #1 :B$(J)&C2$ :: FOR L=1 TO V
    :: M$=M$&RPT$(CH$(L), INT(B(J,L)*C+.5)) :: NEXT L
390 PRINT #1: RPT$(CHR$(232),LEN(M$)) :: PRINT #1: M$
    :: PRINT #1: M$ :: PRINT #1:
    RPT$(CHR$(231),LEN(M$))
400 M$="" :: PRINT #1: C1$; :: NEXT J :: IF Q$="N" THEN
    STOP
410 PRINT #1:"":""
420 FOR J=1 TO V :: PRINT #1: C2$&RPT$(CHR$(232),10) ::
    PRINT #1: RPT$(CH$(J),10)&C1$&"     "&L$(J) ::
    PRINT #1: C2$&RPT$(CH$(J),10) :: PRINT #1:
    RPT$(CHR$(231),10) :: NEXT J
430 !@P+
440 SUB NUMTH(N,N$):: IF FLAG=1 THEN 520 :: FLAG=1 ::
    RESTORE 480
450 GOTO 480
460 J,ONE$(),TEEN$(),TEN$(),N,N$
470 !@P-
480 DATA first,second,third,fourth, fifth,sixth,
    seventh,eighth,ninth,tenth
490 DATA eleventh,twelfth,thirteenth, fourteenth,
    fifteenth, sixteenth, seventeenth,eighteenth,
    nineteenth
500 DATA twenty,THIRTY,FORTY,FIFTY, SIXTY,SEVENTY,
    EIGHTY,NINETY
510 FOR J=1 TO 10 :: READ ONE$(J) :: NEXT J :: FOR J=1
    TO 9 :: READ TEEN$(J) :: NEXT J :: FOR J=2 TO 9 ::
    READ TEN$(J) :: NEXT J
520 IF N<11 THEN N$=ONE$(N) :: SUBEXIT
530 IF N<20 THEN N$=TEEN$(N-10) :: SUBEXIT
540 IF N/10=INT(N/10)THEN N$ = SEG$(TEN$(N/10), 1,
    LEN(TEN$(N/10))-1)&"ieth" :: SUBEXIT
550 N$=TEN$(INT(N/10))&"-"&ONE$((N/10 - INT(N/10))* 10)
560 !@P+
570 SUBEND
```

And a little something educational -

```
100 DIM M$(100)
110 GOTO 150
120 S,J,M$(),A$,Z$,K,W$(),X,Y,ADV$,A,Q$
130 CALL CLEAR :: CALL COLOR :: CALL SCREEN ::
    CALL CHAR :: CALL KEY :: CALL ADVERB :: CALL SOUND
140 !@P-
```

continued on page 17

```
150 CALL CLEAR :: FOR S=0 TO 12 :: CALL COLOR(S,2,8) ::
    NEXT S :: CALL SCREEN(5) :: DISPLAY AT(3,2):
    "ADJECTIVE TO ADVERB V.1.3"
160 CALL CHAR(64,"3C4299A1A199423C") :: DISPLAY
    AT(5,6): " @ Tigercub Software";:; " For free
    distribution with no charge or copying fee."
170 FOR J=1 TO 100 :: READ M$(J) :: A$=A$&CHR$(J) ::
    NEXT J :: Z$=A$ :: CALL KEY(3,K,S)
180 W$(1)=" If adjective ends in Y,     change the Y to
    ILY." :: W$(2)=" If adjective ends in C,    add
    ALLY."
190 W$(3)=" If adjective ends in LL,   just add Y."
200 W$(4)=" If adjective ends in LE,   preceded by a
    consonant,    drop the E and add Y."
210 W$(5)=" If the word ends in E      preceded by a
    consonant,    preceded by a vowel, just add LY."
220 W$(6)=" This word is an exception  to the rule -
    the adverb is WHOLLY."
230 W$(7)=" If the adjective does not  end in C,E,LL or
    Y, always  just add LY."
240 W$(8)=" This is an exception to the rule.  The
    preferred adverb  form is DRYLY."
250 W$(9)=" If the adjective ends in E  preceded by a
    vowel, drop   the E and add LY ."
260 W$(10)=" If the adjective ends in E preceded by a
    consonant     other than L, add LY."
270 RANDOMIZE :: X=INT(RND*LEN(Z$)+1) ::
    Y=ASC(SEG$(Z$,X,1)) :: Z$=SEG$(Z$, 1, X-1)&SEG$(Z$,
    X+1, 255) :: IF LEN(Z$)=0 THEN Z$=A$
280 ACCEPT AT(24,1):M$(Y)
290 CALL ADVERB(M$(Y),ADV$,A)
300 DISPLAY AT(12,1): " Type the adverb form of -" ::
    DISPLAY AT(15,1): M$(Y) :: DISPLAY AT(18,10): "" ::
    ACCEPT AT(15,15) BEEP: Q$
310 IF Q$=ADV$ THEN DISPLAY AT(18,10): "CORRECT!" ::
    GOTO 240
320 CALL SOUND(100,110,5,-4,5) :: DISPLAY AT(20,1):
    W$(A): "";"" :: GOTO 300
330 !@P+
340 DATA DUE,COOL,SOLE,STOIC,FRANTIC, COMIC,ABLE,FULL,
    POOR,HANDY,SORE, SOCIAL,PENAL,SLOW,HIGH,LOW
350 !@P-
360 DATA FRISKY,PLAYFUL,HEALTHY,ROUGH, BUSY,SILLY,
    SICK,SMART,SORE,FAIR,ANGRY,
    BARE,TIRED,WISHFUL,ACTUAL
370 DATA HASTY,LONE,HECTIC,OFFICIAL, MAGIC,MAGICAL,
    MATHEMATIC,LOGIC,TRAGIC, PATHETIC,TRAUMATIC
380 DATA DRAMATIC,AUTOMATIC,AROMATIC, EQUAL,SERIAL,
    BASIC,USUAL,FAVORABLE, UNSTABLE,LEGIBLE
390 DATA HECTIC,LIVE,WARY,VISIBLE, TERRIBLE,HORRIBLE,
    VIVID,FANCY,EASY, VILE,WICKED,BLOODY,SHODDY
400 DATA NOBLE,HAPPY,LEGAL,MERRY,JOLLY, CRAZY,CASUAL,
    CAREFUL,FOOLISH,FAMOUS, GAY,GUILTY
410 DATA HOPEFUL,HATEFUL,TIMID,BRAVE, BEAUTIFUL,DRY,
    NICE,LARGE,PAINFUL, SINFUL,SORROWFUL,SIMPLE,WILLFUL
420 DATA MENTAL,MORAL,PALE,WHOLE, HUNGRY,FINAL,
    FORMAL,TRUE,AMPLE,DOUBLE
430 !@P+
440 SUB ADVERB(M$,ADV$,A) :: L=LEN(M$) ::
    E$=SEG$(M$,L,1) :: F$=SEG$(M$,L-1,2) :: G$=SEG
    $(M$,L-1,1) :: P$=SEG$(M$,1,L-1) ::
    H$=SEG$(M$,L-2,1)
450 IF ASC(SEG$(M$,1,1))<97 THEN A$="ALLY" :: I$="ILY"
    :: L$="LY" :: Y$="Y" :: V$="A EIOU" ELSE A$="ally"
    :: I$=" ily" :: L$="ly" :: Y$="y" ::
460 IF M$="WHOLE" THEN ADV$="WHOLLY" :: A=6 :: SUBEXIT
470 IF M$="DRY" THEN ADV$="DRYLY" :: A=8 :: SUBEXIT
    ELSE IF F$="LL" OR F$="ll" THEN ADV$=M$&Y$ :: A=3
    :: SUBEXIT
480 IF E$="C" OR E$="c" THEN ADV$=M$&A$ :: A=2 ::
    SUBEXIT ELSE IF E$="Y" OR E$="y" THEN ADV$=P$&I$ ::
    A=1 :: SUBEXIT
490 IF E$<>"E" AND E$<>"e" THEN 530
500 IF G$="L" OR G$="l" THEN IF POS(V$,H$,1)<>0 THEN
    ADV$=M$&L$ :: A=5 :: SUBEXIT ELSE ADV$=P$&Y$ :: A=4
    :: SUBEXIT
510 IF POS(V$,G$,1)<>0 THEN ADV$=P$&L$ :: A=9 ::
    SUBEXIT
520 IF POS(V$,SEG$(M$,L-2,1),1)=0 THEN ADV$=M$&L$ ::
    A=10 :: SUBEXIT ELSE ADV$=M$&L$ :: A=5 :: SUBEXIT
530 ADV$=M$&L$ :: A=7 :: SUBEND
```

```
100 !MOCKINGBIRD TINYGRAM by Jim Peterson.  Tap your
    tune on the 1 to 0 keys (tuned A through C)
110 !Then press any other key to hear it repeated
120 DATA 220,247,262,294,330,349,392, 440,494,523
130 FOR J=1 TO 10 :: READ N(J):: NEXT J :: J=0 :: DIM
    T(50,2)
140 CALL KEY(5,K,S):: IF S=0 THEN 140
150 ON ERROR 190
160 CALL KEY(5,K,S):: IF K=-1 THEN 160 :: K=K-(K=48)*
    10 :: T(J,1)=N(K-48) :: CALL SOUND(-999,T(J,1),0)
170 IF K=K2 THEN T(J,2)=T(J,2)+1 :: GOTO 160
180 K2=K :: J=J+1 :: GOTO 160
190 FOR X=0 TO J-1 :: CALL SOUND((T(X,2)+1)* 400,
    T(X,1), 0, T(X,1)* 1.01, 0) :: NEXT X :: J=0 ::
    GOTO 140
```

A little subprogram to add a bit of variety to your "PRESS ANY KEY" routine.

```
1 CALL CLEAR :: CALL PRESSKEY(24)
30000 SUB PRESSKEY(R)
30001 C=C+1 :: IF C=16 THEN 30002 :: DISPLAY AT(R,1):""
    :: DISPLAY AT(R,C): "PRESS ANY KEY" ::
    DISPLAY AT(R,C): "press any key" :: CALL KEY(0,K,S)
    :: IF S=0 THEN 30001 ELSE 30003
30002 C=C-1 :: IF C=0 THEN 30001 :: DISPLAY AT(R,1): ""
    :: DISPLAY AT(R,C): "PRESS ANY KEY" ::
    DISPLAY AT(R,C): "press any key" :: CALL KEY(0,K,S)
    :: IF S=0 THEN 30002
30003 DISPLAY AT(R,1):"" :: SUBEND
```

And a new way to wipe the screen -

```
1 CALL CORNERWIPE(30)
29000 SUB CORNERWIPE(CH) :: FOR T=1 TO 24 :: CALL
    HCHAR(T,3,CH,T+4) :: CALL HCHAR(25-T,32-T,CH,T) ::
    NEXT T :: CALL CLEAR :: SUBEND
```

Memory full
Jim Peterson

Extended BASIC Tutorial

Author unknown, USA

Data Writing Program

Here is a short program that allows you to write data to programs by just typing the data. By using the XLATE program, which is available on TEXPAC, you do not have to even type in this program to make it run.

```
100 ON WARNING NEXT
110 DISPLAY AT(10,1)ERASE ALL:"ENTER FIRST LINE
    NUMBER:" :: ACCEPT AT(10,25) BEEP VALIDATE(DIGIT)
    SIZE(4):LN
120 DISPLAY AT(12,1):"ENTER INCREMENT:" :: ACCEPT
    AT(12,17) BEEP SIZE(3) VALIDATE(DIGIT) :I
130 DISPLAY AT(14,1):"ENTER FILENAME:" :: ACCEPT
    AT(14,16) BEEP VALIDATE(UALPHA,DIGIT) SIZE(10):FN$
140 OPEN #1:"DSK1."&FN$,VARIABLE 163
150 DISPLAY AT(2,6)ERASE ALL:"PRESS ENTER TO END" ::
    DISPLAY AT(22,1): "ENTER A LINE OF DATA:" :: LINPUT
    "":D$
160 IF D$="" THEN 190
170 PRINT#1:CHR$(INT(LN / 256))& CHR$(LN - 256 * INT(LN
    / 256))& CHR$(147)& D$& CHR$(0)
180 LN=LN+I :: GOTO 150
190 PRINT #1:CHR$(255)&CHR$(255)
200 CLOSE #1 :: END
```

'Tell me, Mr Videogame, what's it like suddenly becoming a household name?'

Techo Time with John Paine

This month brings forth another new device from our local TI99/4A community. Peter Schubert has finished prototyping his latest peripheral, which is a product designed as an add on to the Mini-PE system and the CorComp Micro Expansion system.

The device is a stand alone RAMdisk of up to 512K bytes capacity and a built in real time clock. The following notes are taken from a release notice Peter uploaded via modem for inclusion with this article.

Attention Mini-PE owners,
the A.T. RAMdisk has arrived.

A solid state disk drive that works in a similar way to the Horizon card for the PE box. It is fully battery backed to retain data just as a floppy disk does, but with the speed of RAM.

Also incorporated in this new design is a battery backed clock, which is compatible with the TIsHUG clock software and the Johnson MENU system.

(A)dvanced (T)echnology design enables 128K to 512K bytes of RAM and the clock to fit on to one standard size Mini-PE board. You will have to see it yourself to believe it. The ROS is Horizon HRD+ compatible, includes extra CALLs, and can reside in RAM or EPROM.

The device will be available around June and can be bought in a minimum configuration with 128K bytes of RAM and no clock, and you can add the extra memory yourself as your budget allows. The price will be announced in May and orders will be taken for a limited production run.

Enquiries to Peter Schubert;
P.O. Box 28, Kings Cross, 2011
Phone (02)358 5602.

A budgetary price for the minimum system should be about A$180 and the fully configured card would be more than A$450, with the clock making up A$30 of that. Bearing in mind that the 32K chips used are priced around $20.00 each but that is not very stable, it can be seen that total costs cannot, at this stage, be predicted accurately.

The construction method allows the RAMcard to be plugged into the side port of the Corcomp Micro Expansion system as well as being fitted on top of the existing Mini-PE system. The power supply tracks will have links, which will allow power to come from an external power pack or the existing power connections within the Mini-PE system. I would recommend that an external supply be used rather than relying on the console to provide the necessary power.

Disk Controller Compatibility

by Paul E. Scheidemantle, P & A Software

From Tacoma Users Group Newsletter, November 1987. Re-typed by John Ryan of TIsHUG, 8/2/88, with additions by Ross Mudie.

One of the common questions that I am asked is: "If I get this particular Disk controller will it be compatible with one or other of the others?" Well hopefully, this article will help remove those doubts and be of help in clearing up a lot of misinformation.

All of the disk controllers listed below will initialize single or double sided diskettes, provided you have a drive or drives with these features. Next problem is compatibility between the different densities. Shown below is the basic information on each of the major controllers so that you can see which is compatible with what.

One quick note on the Ryte Data chips is that, to my knowledge, they are not compatible with any of the controllers listed below, because they require 80 track drives. You get 1440 sectors with these chips installed in your Texas Instruments disk controller, by initializing double sided single density on 80 tracks.

Texas Instruments

Initializes Single Density only, 9 sectors per track, 40 Track. This diskette can be read and written to by both Corcomp and Myarc Control cards.

CorComp and AT from Peter Schubert.

Initializes Single or Double Density, 9 sectors per track, 40 track, in single density format and 18 sectors per track, 40 track, in double density format. This diskette can be read and written to by both CorComp and Myarc control cards, or the TI control card, providing that the disk is single density format and either single or double sided (again you must have a drive to match).

Myarc

Initializes Single or Double Density 9 sectors per track, 40 track, in single density format and 16 or 18 sectors per track, 40 track, in double density format. This diskette can be read and written to by both Corcomp and Myarc Control cards, or the TI control card provided that the disk is single density format and either single or double sided (again you must have a drive to match).

* Note that if the diskette has been initialized as double density in the 16 sectors per track mode it is compatible ONLY with the Myarc controller!

Definitions

SSSD= Single Sided Single Density
SSDD= Single Sided Double Density
DSSD= Double Sided Single Density
DSDD= Double Sided Double Density
T = TI disk controller
C = Corcomp disk controller
M = Myarc disk controller
A = Peter Schubert's AT

Formats

SSSD	9	Sectors p/track	T C M A
	40	Tracks	
	360	Sectors total	
SSDD	16	sectors p/track	. . M .
	40	tracks	
	640	sectors total	
SSDD	18	sectors p/track	. C M A
	40	tracks	
	720	sectors total	
DSSD	9	sectors p/track	T C M A
	40	tracks	
	720	sectors total	
DSDD	16	sectors p/track	. . M .
	40	tracks	
	1280	sectors total	
DSDD	18	sectors p/track	. C M A
	40	tracks	
	1440	sectors total	

o

For Sale For Sale For

One Only CorComp Microexpansion System complete with Double Sided Double Density Disk Controller (supports four disk drives), 32K memory expansion, RS232 ports (two ports) and parallel printer port (PIO) as well as 240 volt power supply and manuals. This unit is still available from the US on special at around US$330 + freight, Sales Tax at 20% etc.

The asking price is only A$400.

Also, one only TI Multiplan complete with manual, cartridge and original disk.

Asking price $50.00

Contact John Paine on (02)625 6318 or 19 Janet Street, Mt Druitt, NSW 2770

Power supply for disk systems

I have a few switch mode type power supply units. These are commercial units which I have repaired and can guarantee to work reliably and efficiently. They can supply up to 4 disk drives, and come complete with plugs wired for 2 disk drives, a 240V socket, and utility outputs for a fan and +12V;-12V;+5V. The unit measures 240x110x60mm and does not need any transformer. It can be used to power a Mini-PE system or Poorman's Disk Controller with disk drives. I have a few for sale at $75. Phone (02)358 5602 or write: P.Schubert

PO Box 28, Kings Cross, NSW 2011.

For sale

RAMdisk for PE box with 256K of memory (992 SECTORS). A neat reliable board for $220. TI disk control board with manual $70. DMII module $10.

Contact P.Schubert (02)358 5602. o

TI99/4A Video Modes

by Jim Ness, USA

Way back when the TI99/4A computer was designed, it was understood that an inexpensive computer had to provide entertainment as much, or more than, computing power. So it was necessary for such a machine to have decent colour graphics capability, as well as math and text handling capabilities.

TI's response to this was the TMS9929 video display processor. This chip has the ability to display text in 8 x 8 dot size or 8 x 6 dot size (32 column vs. 40 column), or non-text graphics using 4 x 4 dot squares (multi-colour mode), or even single dot graphics (bit-map mode).

This video display processor (VDP) mode is set by an 8-bit register in the processor itself. There is no access to this register from TI BASIC or Extended BASIC, but you can set it from TI Forth, Pascal and Assembly Language.

Graphics mode

The default upon power up of the computer is Graphics mode, which is the 32 column mode we are all familiar with from BASIC. In this mode the processor keeps track of 768 squares, each of them an 8x8 dot matrix. There are 256 possible configurations of each of these squares, with the information on each configuration stored in VDP RAM. The standard TI99/4A characters are loaded upon start up of the machine (BASIC only allows characters 32 through 159 to be tampered with).

Another section of VDP RAM stores which character is currently in each of the 768 squares. And a third section keeps track of which colour each character configuration is supposed to be. Colours in Graphics mode are assigned to configurations in groups of 16 consecutive character numbers. Graphics mode does not set enough RAM aside to let you set each character to a particular colour. Remember that your BASIC program is stored in VDP RAM, so the processor has to leave 14k of VDP RAM open for that.

Text mode

Another VDP mode you are familiar with is Text, or 40 Column mode. This is the mode used for text based programs such as the Terminal Emulator, TI-Writer, Editor/Assembler, and some of the Adventure games. Use of the VDP RAM is similar to Graphics mode, except that since there are 24 lines of 40 columns, there are now 960 8x6 dot squares on the screen. Keeping track of almost 200 extra positions in VDP RAM uses up space normally used to keep track of colour sets in Graphics mode. That fact, plus the fact that colours are not as important in text programs, limits your colour choices to just picking foreground and background colours to be used for all characters.

Multicolour and Bit Map mode

Now we move on to the other two modes, Multicolour and Bit Map modes. First though, we should talk a little bit about use of the VDP memory. VDP RAM consists of 16K bytes of memory, with the beginning used by the VDP for screen functions, and the rest used for temporary storage for peripherals, and, if you do not have memory expansion, for your BASIC or Extended BASIC program. Each graphics mode uses the RAM a little differently than each of the others.

In Graphics and Multicolour modes, the first 768 bytes are used for storing the characters currently at each screen position. In Text mode, the first 960 bytes perform this function, and in Bit Map mode, the second block, starting at byte 6144, and 768 bytes long, does this.

Other sections are the colour table, which keeps track of character colours (not used in text mode, 32 bytes long in Graphics and Multicolour mode, and 6K bytes long in Bit Map mode), pattern descriptor table, which keeps track of the shapes of the characters (2k bytes in all modes except Bit Map mode, which uses 6K bytes for this function), and the sprite table (unused for Text mode, about 128 bytes in other modes). As you can see, Bit Map mode just about uses the entire 16K of VDP RAM to store its information. As you can also see, it would take a lot of work, many hours, to properly set up VDP RAM for a Bit Map mode program.

In Multicolour mode, you have control of 48 rows of 68 blocks, which can be any colour from transparent to white, but can only be blocks, no special shapes. The idea is to build large multicolour shapes from these blocks (1/4 the size of the normal Graphics mode characters). Sprites are available, also. A game like Munchman would be done in this mode.

In Bit Map mode, you have control of 24 rows of 32 columns, but, each of the 768 boxes can be independently configured. You are not limited to just a couple hundred, as in BASIC. Also, each box can have as many as 16 colours attributed to it. In other words, the end result is that you have almost ultimate control over each pixel sending the address to the VDP, is that the two processors march to a different drummer. The CPU runs at about 3MHz and the VDP at about 1MHz. The result is that you have to slow your assembly language program down for a moment to configuring this much RAM? We are talking about 15K of RAM, just to start the program. The other 1K or so is left for you to use as you please. The time factor is why you see very little on the market with fancy Bit Map mode graphics. If a software house is going to put all this time into developing something, they have to know there will be a profit somewhere down the line.

Communication with the VDP processor

There are 4 8-bit registers set aside in the memory mapped area for access to the VDP. Numbers placed into these registers in the proper manner are transferred to and from the CPU and VDP. These registers are the VDP Write Address Register, the VDP Read Data Register, the VDP Write Data Register, and the VDP Read Status Register.

The Write Address Register is used to specify to the VDP which address you are going to write to or read from. Once a number is placed properly in this register, the data currently in that address pops up in the Read Data Register for your use. Since it is an 8 bit register, and we are dealing with 16 bit addresses, only half the address is written at a time, the last byte, pause, and then the first byte. The address in question in the VDP RAM will hold a byte of information, so the data will appear in the Read Register as a whole.

The Write Data Register specifies the data that you want to have written into VDP RAM. Again, you are writing a byte of information, so the whole thing fits into the 8 bit Register. Only the address is 16 bit, so only the Write Address Register is written to twice.

The reason there has to be a pause in SYS-SCRNS. Part of the SYS-SCRNS file is devoted to FORTi's own code and bookkeeping.

Included with TI Forth is also a utility called DISK-HEAD. This utility is designed to take any formatted disk and by writing a carefully crafted be sure that the VDP catches the two byte address you are sending.

The Read Status Register contains some status information about the VDP, including coincidence of sprites and too many sprites on a single line.

By using these registers, you can put any information on the screen, byte by byte, set up the VDP for sprites, or Bit Map graphics, etc. But TI has written some subroutines into the E/A, Extended BASIC and Minimem cartridges that can be referred to in assembly language programs. These subroutines make access to the VDP even easier, and almost all programmers use them. The only problem is that the subroutines are not accessed the same for each cartridge, so if you use them, you have to write them for either the E/A and Minimem, or for the Extended BASIC cartridge.

These subroutines include VMBW, which is VDP Multiple Byte Write, VMBR (Read), VSBW (Single Byte Write), VSBR (Read), and VWTR (Write to VDP write only Register).

In each case, you put the required information into workspace registers, and then branch to the appropriate subroutine. Presto, chango, your data is transferred.

For instance, if you have no shortage of memory space, you could set aside 736 bytes for screen scrolling.

continued on page 13

Games Information

by Robert Brown and Stephen Judd

Welcome to the fourth issue of Games Information. I hope everybody is finding something of interest in this file.

During the month we discovered another test mode, (see our first issue for more details). This time it is in Hopper. When the title appears, press SHIFT[8] and it will give you the option to choose which level you start on, being from 0 to 9. Then it will continue on with the normal option of 1 or 2 players, the game will start on the selected level. If you discover another test mode we have not mentioned then please let us know.

Now let us start on this month's main subject, TI Runner. This program has been recently in the upload/download area of this BBS, so we thought we should let everybody know of the rules, and ways to help you succeed.

The Rules

To control the movement of your jogger use the joystick or the arrow keys and "Q" for fire. To pause the game press "P" and to continue press "C". By pressing FCTN[6] you can commit suicide and by pressing FCTN[8] you can give up and start again at the beginning.

In this game you must collect the keys and other assorted treasures, without being caught by the other joggers (the baddies). To do so you must find the best route around the screen collecting the treasures and at the same time avoiding the joggers. Once all the treasures have been collected by yourself a ladder will appear to take you to the next level, but if there is no other treasure left and no ladder has appeared then one or more of the joggers must have a treasure also. The way to get the treasures from the other joggers is to press fire and a hole will be created for a few seconds. If the jogger falls into this hole, then the treasure will appear above him. To get this you can safely run over the jogger and grab the treasure. Then once you have got all the treasures from the joggers you can climb the ladder to the next level.

If you are having extreme trouble with one particular screen you can skip over it by pressing FCTN[5] after the message "LOADING LEVEL X" has finished and you will be transported onto the next level simultaneously. By doing this you can get to whatever screen you like, but there is an even easier way than this.

To reach a screen quickly press SHIFT[3] while your man is flashing, and then he will disappear. You can then type in whatever level you wish to go to. For example, type 09 for level 9. This number must be between 1 and 50. Numbers below 10 must have a "0" before the level number.

Quite a few screens have some kind of meaning in them.

Now here is a competition. If you can tell us how many screens you can find which relate to anything and have some meaning, and the screen number, you will be given a special poke which turns any TI Runner program into giving unlimited misses without changing the program using a disk fixer. Even cassette users can do it. If you are interested then send a letter listing the number of screens and the screen number to:

C/- Games Information
Robert Brown
141 Beecroft Road,
Beecroft, 2119, NSW
or Send mail on the BBS to GAMES.

The winners will be announced next month, and the prize will be received in the mail.

Here is an evaluation of all the fifty screens in TI Runner. If you have trouble with a screen, GAMES will be quite happy to give you a hint, but no solution will ever be given.

	Graphics	Level of Difficulty	Pleasure	Tips, Hints or Comment
01	Fair	1 Easy	Flat	
02	Fair	1 Easy	Flat	
03	Nice	2 Easy	Flat	
04	Fair	3+ Medium	Flat	Keep one jogger out
05	Nice	3 Medium	Good	
06	V. Nice	2 Easy	Good	
07	Fair	2 Medium	Good	
08	Fair	3 Medium	Good	
09	Fair	3 Medium	Flat	
10	Nice	3 Medium	Good	
11	Nice	4 Hard	V. Good	Have joggers follow guide
12	Nice	3+ Hard	V. Good	
13	V. Nice	4+ Hard	V. Good	Hurry, jogger is coming
14	Nice	3 Medium	Good	
15	Nice	5+ Hard	Boring	Be guide but get a map
16	Nice	2 Easy	Good	
17	Nice	3+ Medium	Good	
18	Nice	2 Easy	Flat	
19	V. Nice	4 Hard	V. Good	Relax near the borders
20	V. Nice	3 Medium	Good	
21	V. Nice	5+ V. Hard	V. Good	Stay in the centre
22	V. Nice	4 Hard	V. Good	
23	V. Nice	3 Medium	V. Good	
24	V. Nice	5 V. Hard	Boring	Stay up and be patient
25	Nice	5 Hard	V. Good	
26	V. Nice	4 Hard	V. Good	
27	Excel.	3 Medium	V. Good	Company to lonely joggers
28	V. Nice	5+ V. Hard	V. Good	
29	Nice	5+ V. Hard	Excel.	
30	Excel.	5 Hard	Excel.	Open the door first.
31	Nice	3 Medium	V. Good	
32	V. Nice	3+ Medium	Excel.	
33	Excel.	5 Hard	Excel.	Mate the mutiny at stern
34	Excel.	5+ V. Hard	Excel.	Jogger eat heart first
35	V. Nice	5 Hard	Excel.	
36	V. Nice	3 Medium	V. Good	
37	Nice	5+ V. Hard	V. Good	
38	V. Nice	5+ V. Hard	????	
39	Nice	5+ V. Hard	????	
40	V. Nice	3 Medium	V. Good	
41	V. Nice	5+ V. Hard	Excel.	
42	Excel.	5+ V. Hard	????	
43	Nice	3 Medium	V. Good	
44	V. Nice	4 Hard	Excel.	
45	V. Nice	3 Medium	V. Good	
46	V. Nice	5 Hard	Excel.	Bricks will be rebuilt
47	V. Nice	3 Medium	Excel.	
48	V. Nice	5+ Excel.	????	
49	Excel.	5+ V. Hard	Excel.	Keep jogger out of hole
50	V. Nice	5+ V. Hard	Excel.	

There you have it. A complete list of all the screens and how good they are.

Remember if you need help with any game or adventure, please leave mail for GAMES. Also if you want us to review something, drop us a line. And do not forget the GAMES competition. See you next month.o

continued from page 12

You would transfer screen lines 2 through 24 to this space in CPU RAM, and immediately transfer it back to VDP RAM, but place it into lines 1 through 23, and then blank out line 24. Like this:

```
*  Scroll Subroutine                  *
       DEF  SCROLL
       REF  VMBR,VMBW
TEMP   BSS  736
BLANKS TEXT '      <32 SPACES>                '
SCROLL LI   R0,32
       LI   R1,TEMP
       LI   R2,736
       BLWP @VMBR
       CLR  R0
       BLWP @VMBW
       LI   R0,736
       LI   R1,BLANKS
       LI   R2,32
       BLWP @VMBW
       RT
       END
```

If this routine were assembled and loaded into a BASIC program, you could have a CALL LINK("SCROLL") statement that would scroll the screen. Of course, you would not go to all that work, because a PRINT statement does the same thing, just a little slower.

o

TiSHUG PUBLICATION LIBRARY.

CODE	TITLE	AUTHOR
00001	TANTALIZING GAMES FOR YOUR TI-99/4A	HAL RENKO/SAM EDWARDS
00002	TANTALIZING GAMES FOR YOUR TI-99/4A	HAL RENKO/SAM EDWARDS
00003	PROGRAMS FOR THE TI HOME COMPUTER	Steve Davis
00004	C A REFERENCE MANUAL	Samuel P. Harbison
00005	PERSONAL/PROFESSIONAL COMPUTERS-How Can They Help You.	TEXAS INSTRUMENTS
00006	STARTING FORTH	Lee Brodie
00007	UNDERSTANDING MICROPROCESSORS.	Don L. Cannon
00008	99ER MEMORY DATA BOOK.	TEXAS INSTRUMENTS
00009	TMS9901 PROGRAMMABLE SYSTEMS INTERFACE	TEXAS INSTRUMENTS
00010	ART & GRAPHICS with Your TI-99/4A.	Thomas A. Thompson, Jr.
00011	ONLINE	STEVE LAMBERT
00012	TI-99/4A CONSOLE AND P.E SYSTEM TECHNICAL DATA	TEXAS INSTRUMENTS
00013	TECHNICAL DATA	TEXAS INSTRUMENTS
00014	TEX COMP 1986 TI BUYER'S CATALOGUE 6TH ANNIVERSARY	TEX COMP
00015	32 BASIC PROGRAMS FOR THE TI-99/4A	TOM RUGG,PHIL FELDMAN
00016	TI GAMES FOR KIDS	ROBERT F.INGALLS
00017	GETTING STARTED WITH THE TEXAS TI-99/4A	STEPHEN SHAW
00018	GETTING STARTED WITH THE TEXAS TI-99/4A	STEPHAN SHAW
00019	9900 TMS9918A/TMS9928A/TMS9929A VIDEO DISPLAY PROCESSORS.	TEXAS INSTRUMENTS
00020	ELECTRICAL ENGINEERING LIBRARY (DISKETTE SOFTWARE)	TEXAS INSTRUMENTS
00021	LEARNING TO USE THE TI-99/4A COMPUTER.	KEVIN TOWNSEND
00022	TMS9901 PROGRAMMABLE SYSTEMS INTERFACE	TEXAS INSTRUMENTS
00023	TMS9118/TMS9128/TMS9129 DATA MANUAL	TEXAS INSTRUMENTS
00024	SCHOOL DAYS FOR THE TI-99/4A	CLAIRE BAILEY PASSANTINO
00025	TUNNELS OF DOOM	TEXAS INSTRUMENTS
00026	CALCULATOR DECISION-MAKING SOURCEBOOK	TEXAS INSTRUMENTS
00027	STARTING FORTH	LEO BRODIE
00028	GET PERSONAL WITH YOUR TI-99	WILLIAM A.MANNING
00029	GET PERSONAL WITH YOUR TI-99	WILLIAM A.MANNING
00030	COMPUTER LANGUAGES AND THEIR USES	GARRY MARSHALL
00031	YOUR TI PROFESSIONAL COMPUTER:USE,APPLICATIONS & BASIC	THOMAS E.HAGRON
00032	VIDEO DISPLAY PROCESSORS PROGRAMMER'S GUIDE	TEXAS INSTRUMENTS
00033	YOUR FIRST TI-99/4A PROGRAM	RODNEY ZAKS
00034	YOUR FIRST TI-99/4A PROGRAM	RODNEY ZAKS
00035	TUNNELS OF DOOM	TEXAS INSTRUMENTS
00036	PARSEC	TEXAS INSTRUMENTS
00037	CONNECT FOUR	TEXAS INSTRUMENTS
00038	ADDITION & SUBTRACTION II	TEXAS INSTRUMENTS
00039	PERSONAL RECORD KEEPING	TEXAS INSTRUMENTS
00040	HUSTLE	TEXAS INSTRUMENTS
00041	MUNCHMAN	TEXAS INSTRUMENTS
00042	ALIEN ADDITION	TEXAS INSTRUMENTS
00043	LAWS OF ARITHMETIC	TEXAS INSTRUMENTS
00044	EQUATIONS	TEXAS INSTRUMENTS
00045	TOUCH TYPING TUTOR	TEXAS INSTRUMENTS
00046	TOUCH TYPING TUTOR	TEXAS INSTRUMENTS
00047	ADDISON-WESLEY MATH GAMES VI	TEXAS INSTRUMENTS
00048	EARLY READING	TEXAS INSTRUMENTS
00049	EARLY READING	TEXAS INSTRUMENTS
00050	VIDEO CHESS	TEXAS INSTRUMENTS
00051	TI EXTENDED BASIC	TEXAS INSTRUMENTS
00052	9900 TMS9918A/TMS9928A/TMS9929A	TEXAS INSTRUMENTS
00053	VIDEO DISPLAY PROCESSORS PROGRAMMER'S GUIDE	TEXAS INSTRUMENTS
00054	9900 FAMILY SYSTEMS DESIGN AND DATA BOOK	TEXAS INSTRUMENTS
00055	9900 TMS9918A/TMS9928A/TMS9929A	TEXAS INSTRUMENTS
00056	9900 TMS 9900 MICROPROCESSOR DATA MANUAL	TEXAS INSTRUMENTS
00057	9900 TMS 9900 MICROPROCESSOR DATA MANUAL	TEXAS INSTRUMENTS
00058	9900 TMS 9900 MICROPROCESSOR DATA MANUAL	TEXAS INSTRUMENTS
00059	THE ELEMENTRY TI-99/4A	WILLIAM B.SAUNDERS
00060	SAMS ENTERTAINMENT GAMES IN TI BASIC AND EXTENDED BASIC	KHOA TON & QUYEN TON
00061	SAMS TI-99/4A BASIC PROGRAMS	TIM ORR KNIGHTAD.LABATT
00062	TMS9902A ASYNCHRONOUS COMMUNICATIONS CONTROLLER.	TEXAS INSTRUMENTS
00063	TMS5220A DATA MANUAL	TEXAS INSTRUMENTS
00064	TI-99/4A	TENEX COMPUTER EXPRESS
00065	TMS9928/29 & TMS9128/29 INTERFACE TO COLOUR MONITORS	JOHN FAINE
00066	TMS9928/29 & TMS9128/29 INTERFACE TO COLOUR MONITORS	TEXAS INSTRUMENTS
00067	EASY C	PETE ORLIN & JOHN HEATH
00068	VIDEO DISPLAY PROCESSORS PROGRAMMER'S GUIDE	TEXAS INSTRUMENTS
00069	TMS9902A ASYNCHRONOUS COMMUNICATIONS CONTROLLER	TEXAS INSTRUMENTS
00070	BUMPER BOOK OF LISTS & REVIEWS	PETER LYNDEN
00071	UNDERSTANDING COBOL	RICHARD G.PEDDICORD
00072	BEGINNER'S BASIC TUTOR	TEXAS INTRUMENTS
00073	H.S. BASIC T.I.	UNKNOWN
00074	TI-99/4A SEVICE MANUAL 2 PARTS	TEXAS INSTRUMENTS
00075	TECHNICAL REFERENCE MANUAL	TEXAS INSTRUMENTS
00076	TMS9220A DATA MANUAL PRELIMINARY	TEXAS INSTRUMENTS
00077	ASSOC	UNKNOWN
00078	ENGINEERING JOURNAL	TEXAS INSTRUMENTS
00079	CREATIVE PROGRAMMING FOR YOUNG MINDS	UNKNOWN
00080	DIRECTIONS VOL.3 NO.1	UNKNOWN
00081	TERMINAL EMULATOR PROTOCOL MANUAL	UNKNOWN
00082	WHAT'S NEW IN COMPUTING VOL.3 NO.11	UNKNOWN
00083	WHAT'S NEW IN COMPUTING VOL.3 NO.12	UNKNOWN
00084	1984 HAYDEN CATALOGUE	UNKNOWN
00085	1984 HAYDEN CATALOGUE	UNKNOWN
00086	TEXAS MONTHLY(THE TEXAS EDISON)	T.R.REID
00087	DIRECTIONS VOL.2 NO.10	UNKNOWN
00088	DIRECTIONS VOL.2 NO.11	UNKNOWN
00089	TMS32010	TEXAS INSTRUMENTS
00090	HOME COMPUTER MAGAZINE VOL.5 NO.3	MCM
00091	TI-99/4A	TENEX COMPUTER EXPRESS
00092	OMNI ONLINE DATABASE DIRECTORY	MIKE EDELHART
00093	ADDENDUM TUTOR ASSEMBLY TUTORIAL	STEVE BARSTAD
00094	DIRECTIONS VOL.2 NO.9	UNKNOWN
00095	DIRECTIONS VOL.2 NO.4	UNKNOWN
00096	DIRECTIONS VOL.2 NO.7	UNKNOWN
00097	COMPUTE BOOKS SUMMER 1984	COMPUTE
00098	BASIC CONVERTER CHART '86	AUSTRALIAN PERSONAL COM.
00099	SOLUTIONS	TEXAS INSTRUMENTS
00100	BEGINNER'S BASIC	TEXAS INSTRUMENTS
00101	BEGINNER'S BASIC	TEXAS INSTRUMENTS
00102	BEGINNER'S BASIC	TEXAS INSTRUMENTS
00103	USER'S REFERENCE GUIDE	TEXAS INSTRUMENTS
00104	USER'S REFERENCE GUIDE	TEXAS INSTRUMENTS
00105	DIRECTIONS VOL.2 NO.5	UNKNOWN
00106	CREATIVE PROGRAMING FOR YOUNG MINDS VOLUME III	TEXAS INSTRUMENTS
00107	USER'S REFERENCE GUIDE	TEXAS INSTRUMENTS
00108	COMPUTER PLAYGROUND FOR THE TI-99/4A	MARCIA CARROZZO
00109	DISK MANAGER 1000	BRUCE CARON
00110	DIRECTIONS VOL.2 NO.12	UNKNOWN
00111	DEMONSTRATION	TEXAS INSTRUMENTS
00112	TISHUG TUTORIAL LINKING ASSEMBLY FROM EXTENDED BASIC.	ROSS MUDIE
00113	TISHUG TUTORIAL LINKING ASSEMBLY FROM EXTENDED BASIC.	ROSS MUDIE
00114	TUTORIAL INTRODUCING LINKING TO ASSEMBLY FROM X.BASIC	ROSS MUDIE
00115	WORKSHOP/TUTORIAL	TISHUG
00116	TISHUG TUTORIAL BOOKLET	TISHUG
00117	TECHNICAL DATA(FOLDER)	MIXED
00118	GAMES FOR YOUR TI-99/4A	TIM HARTNELL
00119	GAMES FOR YOUR TI-99/4A	TIM HARTNELL
00120	ADVENTURE BOOK OF HINTS	SCOTT ADAMS
00121	TI-99/4A SOFTWARE DIRECTORY	TEXAS INSTRUMENTS
00122	MYARC 128K.OS EXTENDED BASIC IV(EXTENDED BASIC II)	MYARC
00123	THE BUMPER BOOK OF LISTS VOL.2	PETER LYNDEN
00124	TISHUG TUTORIAL FILE HANDLING IN ASSEMBLY,LINKED FROM BAS	ROSS MUDIE
00125	TEX COMP 1987 UPDATE	TEX COMP
00126	I SPEAK BASIC TO MY TI-99/4A	AUBREY B.JONES,JR.
00127	FUN AND GAMES WITH YOUR TI-99/4A	STEPHEN M.NUNCY
00128	FUN AND GAMES WITH YOUR TI-99/4A	STEPHEN M.NUNCY
00129	ART AND GRAPHICS WITH YOUR TI-99/4A	THOMAS A.THOMPSON,JR.
00130	GET MORE FROM THE TI-99/4A	GARRY MARSHALL
00131	COMPUTE'S FIRST BOOK OF TI GAMES	C.REGENA
00132	GETTING STARTED WITH THE TEXAS TI-99/4A	STEPHAN SHAW
00133	SAMS BASIC TRICKS FOR THE TI-99/4A	ALLEN WYATT
00134	TI-99/4A INTERN	REINER MARTIN
00135	TURTLE TRACKS	SCHOLASTIC WIZWARE
00136	HOME COMPUTER MAGAZINE VOL.5 NO.4	MCM
00137	VIDEO-GRAPHS	TEXAS INSTRUMENTS
00138	TMS99532A/TMS99534A MODEM PRODUCTS PRELIMINARY DATA MANUA	TEXAS INSTRUMENTS
00139	EDITOR ASSEMBLER MANUAL	TEXAS INSTRUMENTS
00140	99/4 OPERATING SYSTEM	TEXAS INSTRUMENTS
00141	99/4 DISK CONTROLLER THEORY OF OPERATION	BILL HALE
00142	99/4 DISK CONTROLLER THEORY OF OPERATION	BILL HALE
00143	OEM FLOPPY DISK DRIVE TECHNICAL MANUAL VOL.3	SIEMENS
00144	RS-232 PERIPHERAL INTERFACE MANUAL	TEXAS INSTRUMENTS
00145	OEM FLOPPY DISK DRIVE TECHNICAL MANUAL VOL.3	SIEMENS
00146	HOME COMPUTER TECHNICAL DATA	TEXAS INSTRUMENTS
00147	SPEECH SYNTHESIZER	TEXAS INSTRUMENTS
00148	SPEECH SYNTHESIZER	TEXAS INSTRUMENTS
00149	RS-232 PERIPHERAL INTERFACE	TEXAS INSTRUMENTS
00150	TERRIFIC GAMES FOR THE TI-99/4A	HAL RENKO/SAM EDWARDS
00151	YOUR COMPUTER VOL.3 NO.9	FREDERIC PUBLISHING
00152	THE BEST OF 99'er VOL.1	EMERALD VALLEY
00153	COMPUTE'S GUIDE TO TI-99/4A SOUND AND GRAPHICS	RAYMOND J.HEROLD
00154	TERRIFIC GAMES FOR THE TI-99/4A	HAL RENKO/SAM EDWARDS
00155	HOW TO USE THE TI-99/4A COMPUTER	BILL BREWER & J.WILLIS
00156	CREATING ARCADE GAMES ON THE TI-99/4A	SETH McEVOY
00157	36 TI-99/4A PROGRAMS FOR HOME,SCHOOL & OFFICE	LEN TURNER
00158	THE BEST OF TI-99/4A CARTRIDGES	THOMAS BLACKADAR
00159	STIMULATING SIMULATIONS FOR THE TI-99/4A	C.W.ENGEL
00160	THE BEST TEXAS INSTRUMENTS SOFTWARE	CONSUMER GUIDE EDITORS
00161	THE BEST TEXAS INSTRUMENTS SOFTWARE	CONSUMER GUIDE EDITORS
00162	COMPUTE'S GUIDE TO EXTENDED BASIC HOME APPLICATIONS.	CHRISTOPHER FLYNN
00163	GAMES FOR THE TI-99/4A	ANDREW NELSON
00164	TI-99/4A EXTENDED BASIC REFERENCE CARD	TEXAS INSTRUMENTS
00165	HOME COMPUTER MAGAZINE VOL.4 NO.1	MCM
00166	HOME COMPUTER MAGAZINE VOL.4 NO.5	MCM
00167	HOME COMPUTER MAGAZINE VOL.5 NO.1	MCM
00168	HOME COMPUTER MAGAZINE VOL.4 NO.2	MCM
00169	HOME COMPUTER MAGAZINE VOL.5 NO.2	MCM
00170	HOME COMPUTER MAGAZINE VOL.5 NO.1	MCM
00171	HOME COMPUTER MAGAZINE VOL.5 NO.2	MCM
00172	HOME COMPUTER MAGAZINE VOL.5 NO.3	MCM
00173	HOME COMPUTER MAGAZINE VOL.4 NO.8	MCM
00174	HOME COMPUTER MAGAZINE VOL.4 NO.4	MCM
00175	HOME COMPUTER MAGAZINE VOL.4 NO.5	MCM
00176	THE BUMPER BOOK OF LISTS & REVIEWS VOL.2	PETER LYNDEN
00177	THE USERS GUIDE TO TI-99/4A COMPUTER,SOFTWARE&PERIPHERALS	CONSUMERS GUIDE EDITORS
00178	INTRODUCTION TO TI BASIC	B.ENMAN,R.ZAMORA,B.ALBRE
00179	SO YOU WANT A JOB IN COMPUTERS	COMPUTEACH INTERNATIONAL
00180	COMPUTER PROGRAMMING IN BASIC FOR EVERYONE (RADIO SHACK)	THOMAS A.DWYER
00181	STEPPING INTO COMPUTERS (BOOK 1)	MAREE McINNES
00182	FUNWELWED V4.0 OCT'87 UNIVERSAL UTILITY ENVIORMENT	TONY AND WILL MCGOVERN
00183	TI BASIC COMPUTER PROGRAMS FOR THE HOME	CHARLES D.STERNBERG
00184	MOONMINE	TEXAS INSTRUMENTS
00185	MOONMINE	TEXAS INSTRUMENTS
00186	JAWBREAKER II	TEXAS INSTRUMENTS
00187	TI INVADERS	TEXAS INSTRUMENTS
00188	TI INVADERS	TEXAS INSTRUMENTS
00189	TI INVADERS	TEXAS INSTRUMENTS
00190	MUNCHMAN	TEXAS INSTRUMENTS
00191	CAR WARS	TEXAS INSTRUMENTS
00192	CAR WARS	TEXAS INSTRUMENTS
00193	METEOR MULTIPLICATION	TEXAS INSTRUMENTS
00194	TOUCH TYPING TUTOR	TEXAS INSTRUMENTS
00195	LAWS OF ARITHMETIC	TEXAS INSTRUMENTS
00196	MEASUREMENT FORMULAS	TEXAS INSTRUMENTS
00197	PERSONAL RECORD KEEPING	TEXAS INSTRUMENTS
00198	OLDIES BUT GOODIES-GAMES II	TEXAS INSTRUMENTS
00199	COMPUTER MUSIC BOX	TEXAS INSTRUMENTS
00200	TMS9901 PROGRAMMABLE SYSTEMS INTERFACE	TEXAS INSTRUMENTS
00201	TMS9902A ASYNCHRONOUS COMMUNICATIONS CONTROLER DATA MANUA	TEXAS INSTRUMENTS
00202	TMS9901 PROGRAMMABLE SYSTEMS INTERFACE	TEXAS INSTRUMENTS
00203	TMS 9900 MICROPROCESSOR DATA MANUAL	TEXAS INSTRUMENTS
00204	TMS 9900 MICROPROCESSOR DATA MANUAL	TISHUG
00205	MEMORANDUM AND ARTICLES OF ASSOCIATION OF TIsHUG	TEXAS INSTRUMENTS
00206	EDITOR ASSEMBLER MANUAL	UNKNOWN
00207	RAM DISK OPERATING SYSTEM VER 7.1 DOCUMENT	BUD MILLS
00208	HARD+ RAMDISK CONSTRUCTION GUIDE	BUD MILLS
00209	HARD+ RAMDISK CONSTRUCTION GUIDE	BUD MILLS
00210	HARD+RAMDISK CONSTRUCTION GUIDE	BUD MILLS
00211	HARD+ RAMDISK CONSTRUCTION GUIDE	BUD MILLS
00212	HARD+ RAMDISK CONSTRUCTION GUIDE	RALPH MOLESWORTH
00213	INTRODUCTION TO ASSEMBLY LANGUAGE FOR THE TI	BUD MILLS
00214	HRD+ RAMDISK CONSTRUCTION GUIDE	BUD MILLS
00215	HRD+ RAMDISK CONSTRUCTION GUIDE	TEXAS INSTRUMENTS
00216	TEXAS INSTRUMENTS ENGINEERING JOURNAL VOL.3 NO.1	TEXAS INSTRUMENTS
00217	TEXAS INSTRUMENTS ENGINEERING JOURNAL VOL.3 NO.1	TEXAS INSTRUMENTS
00218	TEXAS INSTRUMENTS ENGINEERING JOURNAL VOL.3 NO.1	TEXAS INSTRUMENTS
00219	TI WRITER MANUAL	

TND001	DEC'82	TND073	SEP'86
TND002	APRIL'82	TND074	APRIL'86
TND003	NOV'82	TND075	JAN/FEB'86
TND004	APRIL'83	TND076	NOV'85
TND005	OCT'82	TND077	NOV'85
TND006	AUG'83	TND078	NOV'85
TND007	DEC'83	TND079	DEC'85
TND008	JAN/FEB'83	TND081	OCT'85
TND009	JAN/FEB'84	TND082	SEP'84
TND010	MARCH'84	TND083	JULY'84
TND011	JUNE'83	TND084	JULY'84
TND012	JULY'83	TND085	JUNE'85
TND013	SEP'83	TND086	DEC'85
TND014	VOL.1 NO.3 (NO DATE)	TND087	APRIL'87
TND015	VOL.1 NO.4 (NO DATE)	TND088	APRIL'87
TND016	VOL.1 NO.8 (NO DATE)	TND089	MAY'87
TND017	VOL.1 NO.7 (NO DATE)	TND090	MAY'87
TND018	VOL.1 NO.6 (NO DATE)	TND091	JUNE'85
TND019	VOL.1 NO.5 (NO DATE)	TND092	MAY'85
TND020	APRIL'82	TND093	DEC'85
TND021	MARCH'82	TND094	MAY'84
TND022	JAN/FEB'82	TND095	JUNE'87
TND023	DEC'81	TND096	JUNE'87
TND024	MAY'82	TND097	JULY'87
TND025	VOL.1 NO.1 (NO DATE)	TND098	JULY'87
TND026	JUNE'82	TND099	JULY'87
TND027	AUG'82	TND100	AUG'87
TND028	VOL.1 NO.2 (NO DATE)	TND101	AUG'87
TND029	JULY'84	TND102	AUG'87
TND030	JAN/FEB'85	TND103	SEP'87
TND031	DEC'84	TND104	SEP'87
TND032	NOV'84	TND105	SEP'87
TND033	OCT'84	TND106	SEP'84
TND034	SEP'84	TND107	JAN/FEB'85
TND035	AUG'84	TND108	NOV'84
TND036	JULY'84	TND109	DEC'84
TND037	JUNE'84	TND110	AUG'84
TND038	MAY'84	TND111	MAY'84
TND039	APRIL'85	TND112	JUNE'84
TND040	APRIL'85	TND113	OCT'84
TND041	MARCH'85	TND114	APRIL'84
TND042	MARCH'85	TND115	MARCH'84
TND043	JAN/FEB'87	TND116	JAN/FEB'84
TND044	DEC'86	TND117	DEC'83
TND045	APRIL'84	TND118	DEC'84
TND046	NOV'86	TND119	OCT'87
TND047	NOV'86	TND120	OCT'87
TND048	MARCH'84	TND121	NOV'87
TND049	AUG'85	TND122	NOV'87
TND050	SEP'85	TND123	NOV'87
TND051	JUNE'84	TND124	NOV'87
TND052	OCT'85	TND125	DEC'87
TND053	SEP'85	TND126	DEC'87
TND054	AUG'86	TND127	DEC'87
TND055	AUG'86	TND128	DEC'87
TND056	JULY'86	TND129	OCT'84
TND057	MARCH'86	TND130	SEP'84
TND058	JUNE'86	TND131	JULY'84
TND059	JUNE'86	TND132	JUNE'84
TND060	MAY'86	TND133	OCT'85
TND061	MAY'86	TND134	JAN/FEB'88
TND062	MAY'86	TND135	JAN/FEB'88
TND063	JAN/FEB'86	TND136	JAN/FEB'88
TND064	MARCH'86	TND137	JAN/FEB'88
TND065	APRIL'86	TND138	MARCH'88
TND066	APRIL'86	TND139	MARCH'88
TND067	OCT'85	TND140	MARCH'88
TND068	MAY'85	TND141	MARCH'88
TND069	MAY'85	TND142	APRIL'88
TND070	OCT'86	TND143	APRIL'88
TND071	OCT'86	TND144	APRIL'88
TND072	SEP'86		

```
1000 REM DIET MANAGEMENT PRO
GRAM
1010 DEF RD=INT(VAR*10^DP+.5
)/10^DP
1020 DEF DPL=POS(STR$(ABS(VA
R))&".",".",1)-1
1030 CALL CLEAR
1040 PRINT "SELECT DESIRED A
CTION": :
1050 PRINT "TYPE"
1060 PRINT "   P FOR PLAN"
1070 PRINT "   R FOR RECORDI
NG PROGRESS"
1080 PRINT "   D TO DISPLAY
PLAN"
1090 PRINT "   C TO CHANGE P
LAN"
1100 GOSUB 3540
1110 IF CD=80 THEN 1190
1120 IF CD=82 THEN 2170
1130 IF CD=68 THEN 2070
1140 IF CD=67 THEN 1160
1150 GOTO 1100
1160 CM=1
1170 GOTO 2070
1180 REM PLANNING SECTION
1190 GOSUB 5920
1200 CM=0
1210 PRINT "TYPE": :
1220 INPUT "NAME ":N$
1230 IF CM THEN 5500
1240 INPUT "SEX ":SX$
1250 IF (SX$="M")+(SX$="F")=
0 THEN 1240
1260 IF CM THEN 5500
1270 INPUT "HEIGHT IN INCHES
 ":H
1280 IF CM THEN 5500
1290 INPUT "AGE ":AGE
1300 IF CM THEN 5500
1310 INPUT "PRESENT WEIGHT "
:SW
1320 IF CM THEN 5500
1330 INPUT "DESIRED WEIGHT "
:FW
1340 IF CM THEN 5500
1350 INPUT "START DATE OF DI
ET   (M/D/Y)  ":SD$
1360 IF CM THEN 5500
1370 INPUT "DIET COMPLETION
DATE (M/D/Y)  ":FD$
1380 PRINT
1390 PRINT "IF ERROR PRESS E
, ELSE C TO CONTINUE."
1400 GOSUB 3540
1410 IF CD=69 THEN 1420 ELSE
 1470
1420 E=1
1430 IF CM THEN 1440 ELSE 55
10
1440 LATCH=1
1450 IT=0
1460 GOTO 5510
1470 IF CD=67 THEN 1490
1480 GOTO 1400
1490 CALL CLEAR
1500 PRINT "THE FOLLOWING SE
CTION RE-   QUESTS INFORMATI
ON ABOUT    YOUR DAILY PHYSI
CAL ACTIVITYLEVEL.": :
1510 PRINT "FIVE CATEGORIES
OF ACTIVITY WILL BE USED TO
DETERMINE   YOUR DAILY CALOR
IE USAGE.": :
1520 PRINT "   THESE ARE..."
: :
1530 PRINT "          *SLEEPI
NG"
1540 PRINT "          *SITTIN
G"
1550 PRINT "          *STANDI
NG"
1560 PRINT "          *WALKIN
G"
1570 PRINT "          *EXERCI
SING": :
1580 PRINT "ESTIMATE THE HOU
RS SPENT    EACH DAY FOR EAC
H ACTIVITY. THE TOTAL MUST B
E 24 HOURS.": :
1590 PRINT "PRESS ANY KEY TO
 PROCEDE."
1600 GOSUB 3540
1610 CALL CLEAR
1620 IF CM THEN 5500
1630 INPUT "SLEEPING ":SLH
1640 IF CM THEN 5500
1650 INPUT "SITTING ":SIH
1660 IF CM THEN 5500
1670 INPUT "STANDING ":STH
1680 IF CM THEN 5500
1690 INPUT "WALKING ":WH
1700 IF CM THEN 5500
1710 INPUT "EXERCISING ":EXH
1720 TAH=SLH+SIH+STH+WH+EXH
1730 IF TAH=24 THEN 1780
1740 PRINT
1750 PRINT "TOTAL MUST EQUAL
 24 HOURS,  TRY AGAIN.": :
1760 IT=8
1770 GOTO 5500
1780 CALL CLEAR
1790 CM=0
1800 PRINT "DATA INPUT COMPL
ETED": :
1810 PRINT "YOUR DIET PLAN W
ILL NOW BE  PREPARED, ONE MO
MENT PLEASE."
1820 REM  TOTAL CALORIC INTA
KE
1830 W=SW
1840 GOSUB 3580
1850 GOSUB 3610
1860 BMR=MR*BA
1870 GOSUB 3740
1880 REM NET INTAKE TO REACH
 DESIRED WEIGHT
1890 D$=SD$
1900 GOSUB 3820
1910 SJDN=JDN
1920 SDN=0
1930 D$=FD$
1940 GOSUB 3820
1950 FDN=JDN-SJDN
1960 DR=(SW-FW)/FDN
1970 LC=DR*3500
1980 C=NLC-LC
1990 WR=7*DR
2000 REM PREPARE DISPLAY DAT
A
2010 GOSUB 4390
2020 REM OUTPUT
2030 GOSUB 3950
2040 GOSUB 4000
2050 GOTO 4080
2060 REM DISPLAY PLAN
2070 GOSUB 5920
2080 IF N$<>"" THEN 2100
2090 INPUT "TYPE NAME ":N$
2100 GOSUB 3950
2110 GOSUB 4040
2120 IF CM THEN 5510
2130 GOSUB 4390
2140 GOSUB 4080
2150 REM DIET MONITORING
2160 REM INPUT
2170 GOSUB 5920
2180 IF N$<>"" THEN 2200
2190 INPUT "TYPE NAME ":N$
2200 INPUT "TODAY'S DATE (M/
D/Y)          ":TD$
2210 INPUT "TODAY'S WEIGHT "
:APW
2220 PRINT : :"IF ERROR PRES
S E, ELSE ANY  KEY"
2230 GOSUB 3540
2240 IF CD=69 THEN 2170
2250 GOSUB 3950
2260 GOSUB 4040
2270 PVW=SW
2280 GOSUB 4950
2290 IF EOF(2)THEN 2310
2300 GOSUB 5000
2310 CLOSE #2
2320 REM DATE TO DAY
2330 D$=TD$
2340 GOSUB 3820
2350 TDN=JDN-SJDN
2360 ADR=SW-APW
2370 IF TDN=0 THEN 2390
2380 ADR=(SW-APW)/TDN
2390 B=1
2400 IF ADR=0 THEN 2420
2410 B=0
2420 LOSE=1
2430 IF SW>FW THEN 2450
2440 LOSE=0
2450 P=0
2460 IF TDN<FDN THEN 2480
2470 P=1
2480 DPW=SW-TDN*DR
2490 DP=0
2500 VAR=DPW
2510 RDPW=RD
2520 WV=APW-RDPW
2530 W=APW
2540 GOSUB 3580
2550 GOSUB 3610
2560 BMR=MR*BA
2570 GOSUB 3740
2580 ANLC=NLC
2590 Q$=" CAN'T"
2600 IF (P=1)+(APW>FW)+(LOSE
=1)=-3 THEN 2700
2610 IF (P=1)+(APW<FW)+(LOSE
=0)=-3 THEN 2700
2620 AC=NLC
2630 IF P THEN 2670
2640 FDR=(APW-FW)/(FDN-TDN)
2650 ALC=FDR*3500
2660 AC=ANLC-ALC
2670 VAR=AC
2680 RAC=RD
2690 Q$=STR$(RAC)
2700 AWR=7*ADR
2710 REM FORECAST FUTURE PER
FORMANCE
2720 FFW=APW
2730 IF P=1 THEN 2750
2740 FFW=APW-ADR*(FDN-TDN)
2750 FFD$="???  "
2760 IF B THEN 2860
2770 IF (APW>=SW)+(LOSE=1)=-
2 THEN 2860
2780 IF (APW<=SW)+(LOSE=0)=-
2 THEN 2860
2790 FFDN=(APW-FW)/ADR+TDN
2800 DP=0
2810 VAR=FFDN
2820 FFDN=RD
2830 DN=FFDN+SJDN
2840 GOSUB 5030
2850 FFD$=D$
2860 WC=APW-PVW
2870 TR=WV-PVWV
2880 GOSUB 4950
2890 GOSUB 5170
2900 VAR=APW
2910 RAPW=RD
2920 VAR=AC
2930 RAC=RD
2940 TRAC=DPL
2950 VAR=ANLC
2960 RANLC=RD
2970 TRANLC=DPL
2980 VAR=FFW
2990 RFFW=RD
3000 TRFFW=DPL
3010 DP=1
3020 VAR=WR
3030 RWR=RD
3040 TRWR=DPL
3050 VAR=AWR
3060 RAWR=RD
```

```
3070 TRAWR=DPL
3080 VAR=PVW
3090 TPVW=DPL
3100 VAR=APW
3110 TAPW=DPL
3120 VAR=RDPW
3130 TRDPW=DPL
3140 VAR=WC
3150 TWC=DPL
3160 VAR=WV
3170 TWV=DPL
3180 VAR=TR
3190 TTR=DPL
3200 VAR=FW
3210 TFW=DPL
3220 IF TDN<FDN THEN 3270
3230 CALL CLEAR
3240 PRINT "IT IS NOW TIME T
O DEVELOP A NEW PLAN."
3250 FOR LOOP=1 TO 750
3260 NEXT LOOP
3270 REM DEVELOP PROG COMMEN
TS
3280 IF LOSE THEN 3350
3290 IF TR<0 THEN 3380
3300 IF TR<>0 THEN 3360
3310 IF WV=0 THEN 3360
3320 IF WV>0 THEN 3340
3330 IF LOSE THEN 3360 ELSE
3380
3340 IF LOSE THEN 3380 ELSE
3360
3350 IF TR>0 THEN 3380 ELSE
3300
3360 M$="GOOD"
3370 GOTO 3390
3380 M$="BAD"
3390 IF LOSE THEN 3410
3400 IF WC<=0 THEN 3440 ELSE
 3420
3410 IF WC>=0 THEN 3440
3420 O$="GOOD"
3430 GOTO 3450
3440 O$="BAD"
3450 IF LOSE THEN 3470
3460 IF WV>=0 THEN 3480 ELSE
 3500
3470 IF WV>0 THEN 3500
3480 P$="GOOD"
3490 GOTO 3510
3500 P$="BAD"
3510 REM DISPLAY
3520 GOSUB 5210
3530 REM CALL KEY
3540 CALL KEY(0,CD,ST)
3550 IF (ST=-1)+(ST=0)=-1 TH
EN 3540
3560 RETURN
3570 REM BODY AREA
3580 BA=(W/2.2)^.425*(H*2.54
)^.725*.007184
3590 RETURN
3600 REM BMR LOOK-UP
3610 IF SX$="M" THEN 3640
3620 RESTORE 3650
3630 GOTO 3670
3640 RESTORE 3660
3650 DATA 49,42,35,35,31
3660 DATA 49,44,38,36,33
3670 AF=5
3680 IF AGE<AF THEN 3720
3690 AF=AF*2
3700 READ MR
3710 GOTO 3680
3720 RETURN
3730 REM CALORIC USAGE
3740 SLC=SLH*BMR
3750 SIC=SIH*.52*W
3760 STC=STH*1.04*W
3770 WAC=WH*1.56*W
3780 EXC=EXH*2.08*W
3790 NLC=SLC+SIC+STC+WAC+EXC
3800 RETURN
3810 REM DATE TO DAY
3820 P1=POS(D$,"/",1)
3830 P2=POS(D$,"/",4)
3840 MO=VAL(SEG$(D$,1,P1-1))
3850 DAY=VAL(SEG$(D$,P1+1,P2
-P1-1))
3860 YR=VAL(SEG$(D$,P2+1,LEN
(D$)-P2))
3870 YRPR=YR
3880 MOPR=MO+1
3890 IF MO>2 THEN 3920
3900 YRPR=YR-1
3910 MOPR=MO+13
3920 JDN=INT(365.25*YRPR)+IN
T(30.6001*MOPR)+DAY+1720982
3930 RETURN
3940 REM OPEN FILE
3950 FN$=SEG$(N$,1,6)&"PLAN"
3960 PRINT : :"ACCESS DIET P
LAN FILE"
3970 OPEN #1:"DSK1."&FN$,INT
ERNAL,VARIABLE 240
3980 RETURN
3990 REM DATA OUTPUT
4000 PRINT #1:N$,AGE,H,SX$,S
W,FW,SD$,FD$,DR,WR,FDN,C,SJD
N,SLH,SIH,STH,WH,EXH,TAH,SLC
,SIC,STC,WAC,EXC,NLC
4010 CLOSE #1
4020 RETURN
4030 REM DATA INPUT
4040 INPUT #1:N$,AGE,H,SX$,S
W,FW,SD$,FD$,DR,WR,FDN,C,SJD
N,SLH,SIH,STH,WH,EXH,TAH,SLC
,SIC,STC,WAC,EXC,NLC
4050 CLOSE #1
4060 RETURN
4070 REM DSPLY PLAN-PAGE 1
4080 CALL CLEAR
4090 IF F=0 THEN 4110
4100 OPEN #F:"RS232"
4110 PRINT #F:"DIET PLAN FOR
 ";N$: :
4120 PRINT #F:"START DATE";T
AB(19);SD$
4130 PRINT #F:"FINISH DATE";
TAB(19);FD$: :
4140 PRINT #F:"INITIAL WEIGH
T";TAB(19);SW
4150 PRINT #F:"TARGET WEIGHT
";TAB(19);FW: :
4160 PRINT #F:"DAILY CALORIE
S";TAB(18);RC
4170 PRINT #F:"TO ACHIEVE GO
AL": :
4180 PRINT #F:"DAILY WGT LOS
S";TAB(20-TDR);RDR;TAB(25);"
LBS"
4190 PRINT #F:"WKLY WGT LOSS
";TAB(20-TWR);RWR;TAB(25);"L
BS": :
4200 IF F=0 THEN 4220
4210 CLOSE #F
4220 PG=1
4230 OPG=2
4240 PRINT "PRESS"
4250 PRINT "  SPACE TO SEE P
AGE";OPG
4260 PRINT "  R TO RECORD WG
T CHANGE"
4270 PRINT "  C TO CHANGE PL
AN"
4280 PRINT "  E TO EXIT": :
4290 PRINT TAB(18);"PAGE";PG
;"OF 2"
4300 GOSUB 3540
4310 IF CD=32 THEN 4360
4320 IF CD=82 THEN 2170
4330 IF CD=67 THEN 5510
4340 IF CD=69 THEN 4370
4350 GOTO 4300
4360 IF PG=1 THEN 4760 ELSE
4080
4370 END
4380 REM ROUND OFF & LOC DEC
 PT
4390 DP=0
4400 VAR=C
4410 RC=RD
4420 DP=1
4430 VAR=DR
4440 RDR=RD
4450 TDR=DPL
4460 VAR=WR
4470 RWR=RD
4480 TWR=DPL
4490 VAR=SLH
4500 TSLH=DPL
4510 VAR=SIH
4520 TSIH=DPL
4530 VAR=STH
4540 TSTH=DPL
4550 VAR=WH
4560 TWH=DPL
4570 VAR=EXH
4580 TEXH=DPL
4590 VAR=TAH
4600 TTAH=DPL
4610 DP=0
4620 VAR=SLC
4630 RSLC=RD
4640 VAR=SIC
4650 RSIC=RD
4660 VAR=STC
4670 RSTC=RD
4680 VAR=WAC
4690 RWAC=RD
4700 VAR=EXC
4710 REXC=RD
4720 VAR=NLC
4730 RNLC=RD
4740 RETURN
4750 REM DISP ACT LIST-PG 2

4760 CALL CLEAR
4770 IF F=0 THEN 4790
4780 OPEN #F:"RS232"
4790 PRINT #F:TAB(5);"DAILY
ACTIVITY LEVELS": : :
4800 PRINT #F:" ACTIVITY
HOURS  CALORIES"
4810 PRINT #F:" _________
_____  _________": :
4820 PRINT #F:" SLEEPING";TA
B(16-TSLH);SLH;TAB(26-LEN(ST
R$(RSLC)));RSLC
4830 PRINT #F:" SITTING";TAB
(16-TSIH);SIH;TAB(26-LEN(STR
$(RSIC)));RSIC
4840 PRINT #F:" STANDING";TA
B(16-TSTH);STH;TAB(26-LEN(ST
R$(RSTC)));RSTC
4850 PRINT #F:" WALKING";TAB
(16-TWH);WH;TAB(26-LEN(STR$(
RWAC)));RWAC
4860 PRINT #F:" EXERCISING";
TAB(16-TEXH);EXH;TAB(26-LEN(
STR$(REXC)));REXC
4870 PRINT #F:TAB(15);"_____"
;TAB(23);"_____"
4880 PRINT #F:"   TOTAL";TAB
(16-TTAH);TAH;TAB(26-LEN(STR
$(RNLC)));RNLC: : :
4890 IF F=0 THEN 4910
4900 CLOSE #F
4910 PG=2
4920 OPG=1
4930 GOTO 4240
4940 REM OPEN PROG FILE
4950 FN$=SEG$(N$,1,3)&"PROGR
ES"
4960 PRINT : :"ACCESS PROGRE
SS FILE"
4970 OPEN #2:"DSK1."&FN$,INT
ERNAL,VARIABLE 32
4980 RETURN
4990 REM PROG DATA INPUT
5000 INPUT #2:PVW,PVWV,PD$
5010 RETURN
5020 REM DAY TO DATE
5030 DN=DN-1720982
```

```
5040 YRPR=INT((DN-122.1)/365
.25)
5050 MOPR=INT((DN-INT(365.25
*YRPR))/30.6001)
5060 DAY=DN-INT(365.25*YRPR)
-INT(30.6001*MOPR)
5070 MO=MOPR-1
5080 IF MOPR<14 THEN 5100
5090 MO=MOPR-13
5100 YR=YRPR
5110 IF MO>2 THEN 5130
5120 YR=YRPR+1
5130 S$="/"
5140 D$=STR$(MO)&S$&STR$(DAY
)&S$&STR$(YR)
5150 RETURN
5160 REM PROG OUTPUT
5170 PRINT #2:APW,WV,TD$
5180 CLOSE #2
5190 RETURN
5200 REM PROG DISPLAY
5210 CALL CLEAR
5220 IF F=0 THEN 5240
5230 OPEN #F:"RS232"
5240 PRINT #F:TAB(4);"DIET P
ROGRESS FOR ";N$: :
5250 PRINT #F:"WEIGHT STATUS
"
5260 PRINT #F:"  PREV  (";PD
$;")";TAB(23-TPVW);PVW
5270 PRINT #F:"  TODAY (";TD
$;")";TAB(23-TAPW);APW
5280 PRINT #F:"  PLANNED";TA
B(23-TRDPW);RDPW
5290 PRINT #F:"  CHANGE";TAB
(23-TWC);WC;TAB(25);O$
5300 PRINT #F:"  VAR FROM PL
AN";TAB(23-TWV);WV;TAB(25);P
$
5310 PRINT #F:"  TREND";TAB(
23-TTR);TR;TAB(25);M$: :
5320 PRINT #F:"DAILY CALORIE
S"
5330 PRINT #F:"  TO MEET GOA
L";TAB(24-LEN(Q$));Q$
5340 PRINT #F:"  TOTAL USED"
;TAB(23-TRANLC);RANLC: :
5350 PRINT #F:"AT PRESENT";T
AB(16-TRFFW);RFFW;"ON";TAB(2
9-LEN(FD$));FD$
5360 PRINT #F:"RATE YOU"
5370 PRINT #F:"WILL WEIGH";T
AB(16-TFW);FW;"ON";TAB(29-LE
N(FFD$));FFD$: :
5380 PRINT #F:"PLANNED WKLY
LOSS";TAB(22-TRWR);RWR
5390 PRINT #F:"ACTUAL  WKLY
LOSS";TAB(22-TRAWR);RAWR: :
5400 PRINT "PRESS R TO REVIE
W PLAN, C TOCHANGE PLAN OR E
 TO EXIT."
5410 IF F=0 THEN 5430
5420 CLOSE #F
5430 GOSUB 3540
5440 IF CD=82 THEN 2070
5450 IF CD=69 THEN 5480
5460 IF CD=67 THEN 5510
5470 GOTO 5430
5480 END
5490 REM PARAM CHANGER
5500 ON IT GOTO 5550,5570,55
90,5610,5630,5650,5670,5750,
5770,5790,5810,5830
5510 CM=1
5520 CALL CLEAR
5530 PRINT "NAME=";N$;TAB(26
);"Y/N"
5540 GOSUB 5870
5550 PRINT "SEX=";SX$;TAB(26
);"Y/N"
5560 GOSUB 5870
5570 PRINT "HEIGHT=";H;TAB(2
6);"Y/N"
5580 GOSUB 5870
5590 PRINT "AGE=";AGE;TAB(26
);"Y/N"
5600 GOSUB 5870
5610 PRINT "PRESENT WEIGHT="
;SW;TAB(26);"Y/N"
5620 GOSUB 5870
5630 PRINT "DESIRED WEIGHT="
;FW;TAB(26);"Y/N"
5640 GOSUB 5870
5650 PRINT "START DATE=";SD$
;TAB(26);"Y/N"
5660 GOSUB 5870
5670 PRINT "COMPLETION DATE=
";FD$;TAB(26);"Y/N"
5680 GOSUB 5870
5690 IF E THEN 5700 ELSE 575
0
5700 E=0
5710 IF LATCH THEN 5740
5720 CM=0
5730 GOTO 1490
5740 LATCH=0
5750 PRINT "SLEEP HOURS=";SL
H;TAB(26);"Y/N"
5760 GOSUB 5870
5770 PRINT "SIT HOURS=";SIH;
TAB(26);"Y/N"
5780 GOSUB 5870
5790 PRINT "STAND HOURS=";ST
H;TAB(26);"Y/N"
5800 GOSUB 5870
5810 PRINT "WALK HOURS=";WH;
TAB(26);"Y/N"
5820 GOSUB 5870
5830 PRINT "EXERCISE HOURS="
;EXH;TAB(26);"Y/N"
5840 GOSUB 5870
5850 IT=0
5860 GOTO 1720
5870 GOSUB 3530
5880 IT=IT+1
5890 IF CD=78 THEN 5910
5900 RETURN
5910 ON IT GOTO 1220,1240,12
70,1290,1310,1330,1350,1370,
1630,1650,1670,1690,1710
5920 CALL CLEAR
5930 PRINT "WHERE DO YOU WAN
T THE DATA?"
5940 PRINT "   P FOR PRINTER
               S FOR SCREEN"
5950 GOSUB 3540
5960 F=1
5970 IF CD=80 THEN 5990
5980 F=0
5990 CALL CLEAR
6000 RETURN
```

* * *

```
100 ! LIBRARY PRINT PROGRAM
110 ! BY RUSSELL WELHAM
120 ! MAR 1988.
130 DISPLAY AT(8,1)ERASE ALL
:"PRINT:?":TAB(8);"0) END":T
AB(8);"1) BOOKS":TAB(8);"2)
TNDS"
140 ACCEPT AT(8,7)SIZE(-1)VA
LIDATE(DIGIT):N :: ON N+1 GO
TO 150,160,170,140,140,140,1
40,140,140,140
150 STOP
160 N$="LIB/BOOKS" :: GOTO 1
80
170 N$="LIB/TNDS"
180 DISPLAY AT(14,8):"DSK1"
:: ACCEPT AT(14,11)SIZE(-1)V
ALIDATE(DIGIT):D$ :: DISPLAY
 AT(14,12):".";N$
190 N$="DSK"&D$&"."&N$
200 DISPLAY AT(16,1):"PRINT
FROM WHICH PAGE? 1" :: ACCEP
T AT(16,24)SIZE(-1)VALIDATE(
DIGIT):P
210 I,X=0 :: CALL CHAR(127,"
FFFFFFFFFFFFFFFF"):: CALL HC
HAR(18,1,127,32)
220 OPEN #1:N$,DISPLAY ,FIXE
D 216
230 OPEN #2:"PIO",DISPLAY ,V
ARIABLE 96
240 GOSUB 310
250 IF EOF(1)THEN GOSUB 370
:: CLOSE #1 :: CLOSE #2 :: G
OTO 130
260 LINPUT #1:A$ :: X=X+1
270 IF I+1<P THEN IF X=75 TH
EN X=0 :: I=I+1 :: GOTO 250
ELSE 250
280 B$="| "&SEG$(A$,87,6)&"|
 "&SEG$(A$,6,57)&" | "&SEG$(
A$,63,24)&" |" :: DISPLAY AT
(19,1):B$
290 PRINT #2:B$
300 IF X=75 THEN X=0 :: GOSU
B 370 :: GOTO 240 ELSE 250
310 PRINT #2:CHR$(27);CHR$(1
3);"P";CHR$(27);CHR$(10);CHR
$(27);CHR$(10);TAB(35);"TIsH
UG PUBLICATION LIBRARY."
320 PRINT #2:CHR$(27);CHR$(1
0);TAB(30);RPT$("_",27)
330 PRINT #2:RPT$("_",96):"|
 CODE  | TITLE
                | AUTHOR
                |"
340 PRINT #2:CHR$(27);CHR$(1
0);CHR$(27);CHR$(10);RPT$("_
",96)
350 PRINT #2:CHR$(27);CHR$(3
0);CHR$(8);CHR$(27);CHR$(10)
360 RETURN
370 PRINT #2:CHR$(27);CHR$(1
0);CHR$(27);CHR$(10);RPT$("
",96):: I=I+1 :: PRINT #2:""
:TAB(40);"PAGE";I;".":"\ \"
:: RETURN
```

continued from page 6

```
N5      BYTE 8
        TEXT 'RECEIVER'
N6      BYTE 8
        TEXT 'COMPUTER'
N7      BYTE 14
        TEXT 'DRUNKEN SAILOR'
        EVEN
TOTQTV DATA 8  quantity of verbs
V1      BYTE 7
        TEXT 'INVALID'
V2      BYTE 2
        TEXT 'GO'
V3      BYTE 3
        TEXT 'GET'
V4      BYTE 4
        TEXT 'DROP'
V5      BYTE 6
        TEXT 'PICKUP'
V6      BYTE 4
        TEXT 'TAKE'
V7      BYTE 4
        TEXT 'KILL'
V8      BYTE 4
        TEXT 'SWIM'
        EVEN
        END
```

*----------

Hitch Hiker's Guide to the Galaxy part 3

by Stephen Judd and Robert Brown

Zaphod

You come out of the dark to find that you are now Zaphod Beeblebrox, the President of the Universe. In fact, you are on your way to steal the Heart of Gold (with a little help from Trillian). As your speedboat zooms towards its destination, search the seat carefully and you will find seat fluff and a key. The key opens the toolbox, but you do not need to do that now. Just make sure you take the box; you might be needing it later. Now, if you continue on your present course, you will never make it between the cliffs and the spire (or maybe you know that already). The trick is to make the autopilot do the hard work, so steer the boat towards the rocky spire.

The spire gets closer, closer, closer and then, at last the autopilot wakes up, just in time, and steers you to safety! Whew, that was a close one. OK, now you can stand up and go North to the Dais, where the dedication ceremonies will be held.

Wait around, enjoying the cheers of the crowd (read the banner if you like), until Trillian appears. She will jump out of the crowd, and hold a gun to one of your heads. The guards are a little hesitant about what to do, so now is your chance: tell them not to shoot.

After a few moments, they will drop their rifles into a pile, just what you have been waiting for. Tell Trillian to shoot the rifles. As the weapons disappear, you and Trillian make a break for the HOG! You made it!! But, everything seems to be getting.. dark.

OK. Now you should have collected the four fluffs, the ultra-plasmic awl, the paint chipper, the nutrimat computer interface, and the tool box. After you have done the last scenario (whichever one that is), do not go back to the Bridge. Pick up the interface, and go to the Nutrimat. It is tea time!

Open the panel on the Nutrimat, remove the circuit board, and replace it with the interface. Now, touch the pad. With a clearer idea of just what it is you want, the Nutrimat begins to have some problems. Its own limited circuitry cannot handle it (well, it is just a dumb machine, after all), so it ties into the main shipboard computer.

Do not spend time here watching the Nutrimat go through its gyrations. Head for the bridge, and plug the large plug into the large receptacle. The moment is almost here. The HOG has arrived at the legendary lost planet of Magrathea, and the natives are not friendly. In fact, they are sending up a bunch of missiles to vaporize the HOG (hmmm, they really ARE NOT friendly!). Now, push the switch on the spare drive. Wow! Talk about improbabilities! The missiles have turned into a giant sperm whales!

After accepting the congratulations of Ford, Zaphod, and Trillian (who conveniently disappear into the sauna again), return to the Nutrimat, where you will find, at last, a cup of REAL tea. Get the cup (you will drop the No Tea), but do not drink it!! Bring it to the Bridge. Drop the real tea (you will automatically pick up the No Tea). Remove the dangly bit from the tea substitute, and put it in the real tea. You have one more little trip to make. First, however, drop everything you are carrying except the Babel Fish and the Aunt's Thing (yes, you have it again, you just cannot get rid of it).

Push the switch on the Drive. After a short stay in the dark, you will find yourself in the whale's tummy (it may, however, take more than one try to get here, but you will make it eventually). There is a flowerpot here! Get the pot, and put it in the Aunt's Thing. Now, wait around (you really do not have a choice), and you will be in the dark again.

Ah, back on the HOG at last. If you take inventory, you will notice you do not have the Aunt's Thing. Do not panic! It will, as always, turn up. In the meantime, go around picking up the various fluffs. The Zaphod fluff, along with the tool box, will be by the hatch. Trillian's, of course, is in her handbag, and Ford's is on the satchel, and the last one is in the pocket of your gown (unless you took it out earlier and dropped it somewhere).

The Aunt's Thing has reappeared by now, so go up to the Bridge. Take the flowerpot, plant all four fluffs, drop the pot, and wait awhile. When you see a tiny sprout has formed, take the pot into the sauna. When you emerge, a changed man, you will also have a changed plant.

However, there is another problem! The HOG has landed on Magrathea, but Eddie, overprotective as usual, has jammed the hatch shut. And, he is not going to open it, no matter how long it takes him to check for dangers on the planet (which will be quite a few years).

You are almost ready! First, eat the fruit from the plant (mmm, tasty!). You have a vision, and pay close attention to it. The vision shows you what tool Marvin will need to open the hatch. This varies from game to game, and there is no way to know which one it is until you eat the fruit. That is also why you have to collect all those tools. Get the tool that you saw in the vision. If it happens to be one you have not seen yet, then you will find it in Marvin's pantry.

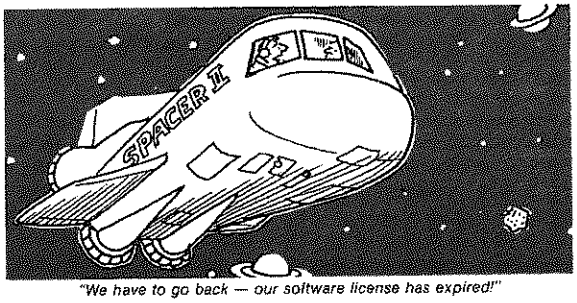
The trick now is to find Marvin, and he is in his pantry, behind the screening door. First, get the real tea. You automatically drop the No Tea. But, you do not have your common sense anymore, so, pick up the No Tea! Now, you have both Tea and No Tea at the same time!!

Go to the Screening Door. Open it. The Door, impressed by your being able to have both Tea and No Tea will let you through! However, WAIT!!! Do not go through the door yet! If you set foot in the pantry, you will be overwhelmed by depression! So, that magic moment has arrived, the moment you have been waiting for ever since you left Earth, drink the real tea!! (Ahhhhhh, good to the last drop!)

All right! Now you can go into the Pantry (yay)! Marvin will be there, sulking as usual. Tell him to fix the hatch. Marvin will grumble, but he will agree to do it, and he will tell you to meet him at the Hatch Access Space, with the proper tool, in twelve moves. As you already have the tool (thanks to the fruit), you can go directly to the Access space (drop everything but the tool and the Fish), and wait for Marvin. When he arrives and asks for the tool, give it to him. Marvin will fiddle briefly, and the hatch will slide open.

Go out to the Hatch, and then down the Hatch. Wow! You have now set foot on the legendary lost planet of Magrathea, and, ..that is it. Well not really. There is a sequel, but not available on the TI99/4A. o

CHOCKY AND TICTACMAN



"We have to go back — our software license has expired!"

Fixing Blown Disks with the Navarone Disk Fixer

Author unknown, USA

Note: This is an adaptation of a file that originally appeared on the TI-Source BBS (author unknown). While the original was informative and useful, I found, while actually using the Navarone Disk Fixer cartridge, a few additional short cuts to repairing disks with blown sectors 0 and/or 1, and I thought I would present this updated version in order to include these new tips. Howard Massey.

Did you ever try to catalog a disk and find out the Disk Controller thinks the disk is not initialized? But you know better! What do you usually do with the problem disk? Most people just delete the file giving them the problem, and that will work, but it also gets rid of that file forever! A better solution is to use Disk Fixer by Navarone Industries.

The Disk Fixer enables one to examine and change the contents of any disk on a sector by sector basis. I think it is worth its forty dollar list price. It is available from some TI retailers, or directly from Navarone Industries.

Here is the process to fix a blown disk.

First acquire a Disk Fixer from a friend or buy one, it is worth it. Before you start, have a blank, initialized disk handy, as you will certainly be needing it during the course of the "surgery". Now we have to determine if the problem with your bad disk lies with either Sector 0 or Sector 1, or both. If it is a "blown" sector, which means the sector is physically OK, but the information contained in it has somehow been partially or totally erased, no problem, Disk Fixer can fix it!

Sector 0 contains the information concerning the disk name and number of sectors used on the disk. If this sector is completely blown, attempting to catalog your disk will always result in a "DISK NOT INITIALIZED" error message, even though your data is actually present on the disk. If it is partially blown, attempting to catalog the disk may show you used sector information that is obviously in error (as when more or less than 358 sectors are used or available for a SS/SD disk). The Disk Fixer can easily confirm for you if the problem is in your Sector 0.

Put the Disk Fixer cartridge in the slot, turn on your computer and select 2 from the master menu screen. Put your bad disk in drive 1, and then type in the following:

R 0,1 <ENTER>. This command means Read Sector 0 from drive 1.

If you get an error message, you know that Sector 0 is blown. The easiest way to fix this is to copy a good Sector 0 from another disk to the blown disk. Here is how to do that:

1) Insert a good disk in drive 1.
2) Read Sector 0 of that disk: R 0,1 <ENTER>
3) Put the blown disk in drive 1.
4) Write good Sector 0 to disk: W 0,1 <ENTER>

If you now leave the Disk Fixer, and use a program to catalog the bad disk, you will see that the diskname and the used/free information is the same as the good disk, but do not let that alarm you - we did that to fool the Disk Controller into letting us at least catalog the bad disk, even inaccurately!

The next step is to find out if Sector 1 has been damaged also. The easiest way is to simply look at your catalog and see if all the files you believed were on the disk are still there. If you are sure that they are, and that none are missing, then you have, almost, fixed your disk. If this is the case, then all you now need to do is to make a copy of your bad disk, file by file, to a blank, initialized disk. The reason you must do this is that since the Sector 0 information on the "fixed" disk is inaccurate, the first time you try to add or delete a file to or from this bad disk, you will run into serious trouble! So put a write-protect tab on it, and back it up onto a fresh initialized disk, one file at a time.

Do not use a bulk copying program, either bit map or sector by sector, as the object disk will then end up with the same inaccurate Sector 0 data. By copying a file at a time (using, for example the "C" command in the DM1000 File Copy/Rename/Delete/Change option), your controller will create a new, completely accurate Sector 0 on your back-up disk. When you have done that, you can either file away the original for your archives or simply erase and re-use it.

On the other hand, suppose the catalog shows that some or all of the files you believed to be on that disk are missing? Have no fear, they are probably still there, they just are not showing up to your disk controller because Sector 1 has been blown as well. In order to fix up this sector, we need to know precisely what files are actually on the disk. We can obtain a current catalog of the files present by using Disk Fixer to peek into sectors >2 to >21 (that is sectors 2 through 33 in decimal). As the Disk Fixer manual points out, TI DOS uses Sector 1 for a Directory Link Map, essentially a "road map" for the controller to look up the various File Directories. Even if this map has been damaged or erased, the directories themselves as well as the actual file data probably still exists on the other sectors, so it is no tragedy. TI DOS uses sectors >2 through >21 to store the actual File Directories, containing the file names, types, and sector locations. The data itself is stored on sectors >22 on up (and the very first file you write onto a new disk is always stored on Sector >22, so that is a good way to quickly determine if a disk is actually blank or not). Therefore, all we really need to do is read sectors >2 through >21 in order to get a list of the file names on our bad disk.

When Disk Fixer reads a sector, the data is displayed on your screen in hexadecimal, which you must then convert to ASCII in order to actually read the information in English. Fortunately, the "M" command allows us to inspect the buffer RAM and in doing so automatically does this hexadecimal to ASCII conversion. So here is how we get a list of the files on our bad disk:

1) Type R 2,1 <ENTER>. The screen will fill with hexadecimal code. There is no need to see it all, so instead of pressing the space bar after the first lot, press any other key to abort.
2) Now type M 1000V,100CV <ENTER>. On the right side of your screen, you will see the first three lines of Sector 2 in ASCII. These lines will contain the file name. Get a piece of paper and write this name down, along with the sector number, as follows.
   Sector 2: {filename}
   (If at any time you want to see the whole screen and not just the first three lines in ASCII, type
   M 1000V,10FFV You can press the space bar to stop the scrolling).
3) Now just type R <ENTER>. Disk Fixer will automatically advance to the next sector (Sector 3). Follow the same procedure as in step 2 above and make sure you write down on a piece of paper the sector number and the filename.
4) Continue as above until eventually you come to a sector filled with >E5 data in every address. This usually means you have reached the end of your search. If you still think there are more files missing, keep looking until you reach Sector >22, which will definitely contain only file data. If you are ever unsure of which sector Disk Fixer is currently reading, use the "C" command in order to find out the current sector.

At the end of this procedure, you should now have a complete list of all the files on your disk, along with their sector numbers. The next step is to sort this list into alphabetical order. If you are unsure about how to sort them when dealing with funny characters, etc., TI DOS sorts by lower to higher ASCII values. These values can be found on your TI BASIC reference card, or in the manual. Suppose, for example, you ended up with the following list of files on your bad disk:

continued on page 20

Disk Drives and Interlaces

by Tom Arnold, Channel 99ers

One of the objects of this article is to point out the importance of attending our monthly meetings. It is here that you will pick up information that is very useful to you. At the last meeting Clint Pulley was showing me a few new programs that he had for Geneve. Included was a new version of the Myarc Disk Manager. We started discussing formatting of disks, a standard, very common procedure, or so I thought. It turns out that the Myarc Disk Manager can set the interlace at different levels. The idea of interlace is to spread the sectors out to allow the computer to store the data and move the head from track to track. This ability to set the interlace can only be done on the Myarc Disk Controller (see at the end of this article, ED). Clint pointed out that the disk controller reads the disk at a certain speed but can only retrieve a few sectors at a time. It then lets the disk spin another revolution before reading some more. The extra revolutions cause the process to slow down. By spacing the data in nearly consecutive sectors the disk controller is reading all the time.

For comparison, the TI controller lays down the sectors in the order 075318642 which means any sector must be read in four passes. This is necessary because the TI controller can only handle the data at this speed. Apparently the Myarc controller reads the data and puts it directly into CPU RAM. It also puts some data into a special buffer RAM chip on the controller card. This allows for greatly increased access speed. By giving the disk the proper interlace you can achieve optimum read and write speed. The interlace is usually set at 3 for double sided double density and 2 for double sided single density. The normal TI interlace is 5. You can experiment by formatting the disks at different interlaces. By watching the verification speed you can determine the best interlace. When the verification slows down you have lowered the interlace number one number too many. The disk is spinning one revolution too many on each read! See the manual that comes with the Myarc disk controller for further details. It is listed under "Format".

One extra benefit that Clint told me about was that I could format my BOOT disk for the Geneve at interlace "1". I tried this and could not believe it. The Geneve normally boots up in about 24 or 30 seconds. With the disk interlaced at 1, the boot time was 8.5 seconds!! This was a real revelation. I had been planning on buying a "fast booting disk" from Disk Only Software. Now, thanks to Clint's expertise I do not have to.

Correction

from Greg Hope, TIsHUG

A correction with a regard to the above article from Greg Hope. This article claims that the Myarc disk controller is the only one that allows interlace setting. Sorry, but the author is wrong!! I was one of the original people, in the Australian TI community, to get a DS/DD controller from CorComp. To my certain knowledge, the CorComp controller has always had interlace setting and it was the first company to come out with a DS/DD controller for the TI. I hope that clears up that misconception. o

Programming Hint

Short Sprite Routine

This is a simple routine for all you Extended BASIC game programmers. This one line program is used to find the position of a sprite and then start a second sprite moving in the direction of the first, so that it will intercept it.

```
100 CALL POSITION(#2,R,C,#1,Y,X) :: CALL MOTION
    (#2,(Y-R)*.49, (X-C)*.49) :: GOTO 100
```

A good book on sprites is available from: Millers Graphics, 1475 W. Cypress Ave. San Dimas, CA. 91773. It is the Smart Programming Guide to Sprites o

continued from page 19

Sector #	Filename
2	CAT
3	HELLO
4	ART
5	PLOT
6	LOGO
7	LOAD
8	PICTURES
9	LINES1
A	LOGO2
B	DUMP
C	LINES2

If we sort them, we end up with this list.

Filename	Sector number
ART	4
CAT	2
DUMP	B
HELLO	3
LINES1	9
LINES2	C
LOAD	7
LOGO	6
LOGO2	A
PICTURES	8
PLOT	5

Now we are ready to fix Sector 1. Start by reading Sector 1 from your blank, initialized disk, since this will contain all zeroes in all addresses:

1) Put the blank disk in drive 1 and type in the following: R 1,1 <ENTER>.
2) Put the bad disk in drive 1 and type: W 1,1 <ENTER>. We have now written a completely blank Sector 1 to our bad disk. This is important, since the controller must find all zeroes following the last File Directory entry.
3) The next step is to alter this data in the Disk Fixer's buffer and then rewrite it back onto the bad disk. Get out your sorted list of files (as above), and type the following: A 0 <ENTER>. Now all you need to do is to type in the sorted list sector numbers, in order, pressing the space bar, not <ENTER> after each entry. So if we were using the list above, here would be the routine:

```
type: 4 <SPACE>
      2 <SPACE>
      B <SPACE>
      3 <SPACE>
      9 <SPACE>
      C <SPACE>
      7 <SPACE>
      6 <SPACE>
      A <SPACE>
      8 <SPACE>
      5 <SPACE>
      <ENTER>
```

After each <SPACE>, Disk Fixer will advance one line, just type in each number, one at a time, with a <SPACE> in between each entry and be careful not to hit <ENTER> until after the last entry. If you make a mistake, do not worry. Just press <ENTER>, retype A 0, and Disk Fixer will let you start all over.

4) The last step is to write this altered data onto Sector 1 of your bad disk. Make sure the bad disk is still in drive 1, type W 1,1 <ENTER>, and you are finished, except for the necessary step, as above, of copying the bad disk, file by file, onto your blank disk. As before, do not just put the "fixed" disk back into your active file, because the first time you try to add or delete a file, you will be in for problems.

And that is about all there is to it. Happy fixing! o

For Sale

I have a very good selection of cartridge software surplus to my needs. Titles such as Tunnels of Doom, SpadXIII Mk.2, Alpiner, Parsec, Moon Mine, Catwars, Music Maker and many more.

Please contact me on (02) 449 4273.

Using multiple UTIL1 Files

Mel Samouri, USA

If anyone else out there has a problem with all of those "UTIL1" files, here is my solution to a couple of them. First some background. I got into this because of a problem with my RAMdisk. With a good RAMdisk you can put a lot of programs onto the RAMdisk that normally would take up several SSSD disks or even SSDD disks. The RAMdisk can hold more than a DSDD disk and (to me) is so much easier to use. TI-Writer editor loads in 1/10 of the time from a RAMdisk than from a normal floppy disk as one example, and the Dragonslayer spelling checker runs in about 1/3 the normal time as it is heavily disk dependent. (In case you have not guessed, I would not trade mine for anything, except possibly a CRAY1 supercomputer). But, back to the problem. It seems that most of the software writers name their assembly language files "UTIL1", as that is the default name for the TI-Writer utility files. This is not a problem unless you want to put more than one program on a disk and you cannot because they have the same name.

I will discuss here the modification for Funnelweb Farm's Funlwriter (V3.1) and Paul Charlton's FastTerm.

For Funlwriter change the following program lines as I have done. The line 160 merely displays the name of the program and not the file name. Line 240 actually contains the first file name to run. In this case, I have renamed the files from "UTIL1" and "UTIL2" to "SPEL1" and "SPEL2" respectively.

```
160 OP$(0)="SPELLCHECK" ! OPTION #4
240 A$="DSK1.SPEL1" :: K=2 :: GOTO 290 ! OPTION #4
```

This was pretty easy as you can see. I will also put in a plug for Funlwriter as it has some excellent capabilities, such as the ability to act as an Editor/Assembler emulator, and will run most assembly programs. The only one I have found that it will not run is Demon Attack. Small price to pay to be able to get away from the E/A cart.

For FastTerm, it is not quite so easy, but is not impossible. Please read the working notes at the end before attempting this.

1. Load the file FAST-LOAD into an editor, preferably the E/A editor but TI-Writer will do.
2. In line 6, at the TAB 48 position are the two hexadecimal digits that define the disk drive number, TAB 53 is where the two hexadecimal digits for the first character (originally 55, I changed it to 54), TAB 56 is where the four hexadecimal digits for the second and third characters (originally 5449, I changed it to 4552), TAB 61 is the position of the last change for line 6. It must be changed from 7 to 8! This is the checksum tag, the 7 tells the loader to perform checksum error detection and the 8 tells it not to. As we have altered this line, the checksum is no longer valid and if the 8 is not used, you will get a "DATA ERROR" message when the file is loaded.
3. In line 7, at TAB 6 are the last two hexadecimal digits for the fourth character (originally 4C, I changed it to 4D). At TAB 8 are the two hexadecimal digits for the fifth and final character (I did not change this one), and serve a special purpose as when the program is loading, this digit is incremented to load the second file. The last change is at TAB 61, changed from 7 to 8 for the reasons stated above.
4. Save the file back to disk and rename the files (I used TERM1 and TERM2) and there you have it. A customized version.

Working Notes

1. Perform this **only** on a "working" copy and not on your master! It is easy to start again on a working copy if you make a mistake, but, on your only copy; well..
2. The file **must** be saved as DISPLAY/FIXED 80 or it will not load. If using the E/A editor, when it asks if you want Variable 80, answer no! If using TI-Writer, use the following program to convert the file from DIS/VAR 80 to DIS/FIX 80. Failure to convert the file to DIS/FIX 80 will also cause the dreaded DATA ERROR. (Of course you can get DIS/FIX 80 files out of TI-Writer by putting "F " in front of the file name when doing a PF. ED)

```
100 CALL CLEAR
110 PRINT "READING"
120 DIM A$(25)
130 OPEN #1:"DSKn.filename",DISPLAY, VARIABLE 80
140 FOR I=1 TO 25
150 LINPUT #1:A$(I)
160 NEXT I
```

the first character (originally 55, I changed it to 54), TAB 56 is where the four hexadecimal digits for the second and third characters (originally 5449, I changed it to 4552), TAB 61 is the position of the last change for line 6. 3. The two files UTIL1 and UTIL2 **must** be sequentially named, for example, TERM1 and TERM2, or TERMA and TERMB, as the loader will load the first program then add 1 to the last character of the file name and perform another load of this filename. The program is broken into two parts because of a file size limitation of the loader.

If any of you are considering purchasing a RAMdisk I will give you my experiences so far. I looked at the Horizon RAMdisk and the MYARC 512K card with RAMdisk/Print spooler and finally opted for the MYARC for several reasons. The two Horizon cards I used had problems with dropping data and causing lockups. I will give them the benefit of the doubt and admit that both cards were built as kits and I did not assemble them, so I cannot be sure that there was not some error made in assembly or just unlucky glitches peculiar to those two cards, but I felt that it just was not worth taking the chance, so I purchased the MYARC 512K system. Unlike the Horizon system, the MYARC does not have the battery backup system, but does have a external power supply connection that takes any 9V at 500MA (the ATARI game power packs work great). I initially had a problem with occasionally turning on my system and finding my RAMdisk erased, but this was not the fault of card but of occasionally heavy surges in my AC line voltage. I fixed that with a large capacitor (20,000mfd) across the output of the power pack and have not lost a "bit" of data since. (Pun intended!) I have also noted that CorComp has now come out with a RAMdisk card but I have not seen one first hand. There seems to be a division into two camps lately over the two manufacturers, CorComp and Myarc. Randy, in the Rag, is very "anti-Myarc" and "pro-CorComp". I have two complete systems, one is my "home" system and one travels with me as I am on the road a lot.

My home system consists of the Myarc 512K card, TI RS232, Myarc disk controller, and the CorComp Triple Tech card, 2 SSDD drives, Gemini-10X printer and Smith-Corona L1000 LQ printer.

My travel system consists of the CorComp Micro Expansion System (32K, Controller, RS232), 2 DSDD drives. So I can say that I have seen both sides of the street. Both of these systems have pluses and minuses.

On the minus side for the Myarc system, is some incompatibility with the copy protection of some software, such as Companion and Advanced Diagnostics, but on the plus side is customer support, as any questions or correspondence I have had with them, I have always received a prompt reply.

On the plus side for the CorComp is the software compatibility with all software I have tried, but this is rather heavily (to me) offset by their minus side, customer support. I have written to them 5 times and (in desperation) called them 3 times and as yet have not received **any(!)** support from them whatsoever, (do they know how to write?), and why is it that "the guy who can answer that question is not here today and I do not know when he will be back", (does he even work there?), and they will not return long distance calls even if you offer to reverse the charges. For 6 months now I have been trying, which is too long to keep any customer "satisfied", but who knows, I may hear from them someday.. (But I am not holding my breath! Would I spend the $500.00 on CorComp hardware again? Not unless their customer support is **proven(!)** to be vastly improved (anyone have any pull with them? Randy?).

I hope, as always, that this is of use to someone else besides me. If you have you have any questions or comments please let me know.

Mad Mel, bags packed and gone again.

o

Changing the Colour of Characters

by Joe Nollan, Tacoma User's Group

Whilst looking over some of the news letters from other clubs, I came across this short machine language routine that changes the character set colours. Credit for this routine goes to Harry Wilhelm and the TWIN TIers USERS GROUP. Some of you are all ready ahead of me and thinking that it can be done with a CALL COLOR statement,and you are right, however by using this routine the changes are permenant! This means that even after the program stops running.

Key in this two liner :-

```
100 CALL INIT :: CALL LOAD(16128, 2, 224, 38, 0, 2, 0,
      8, 17, 2, 1, 63, 36, 2, 2, 0, 3, 4, 32, 32, 36,
      2, 224, 131, 192, 3, 128)
110 CALL LOAD(16164,240,240,240):: CALL LOAD(-31804,63)
```

List the program and then run it. You will now see that the numbers and arithmetic operators are white. As Harry pointed out this makes it easier to distinguish between zero and the letter "O", as does the letter "I" and numeral 1. You can turn it off by typing CALL LOAD(-31804,0) and turn it on by typing CALL LOAD(-31804,63). As a program debugging tool I think it is great. You can now load and list your programs to check for errors. Be sure to turn the routine off before running your program because it will not allow any other colour changes. As the program is written it affects character sets 2, 3 and 4 but by editing the first CALL LOAD statement it is possible to change the colour of any character set. The eighth number after the address (16128) identifies the number of the first set to be changed and has a 15 offset which means that you subtract 15 from the number to find the first set. In this case the 17 represents set 2 (17-15=2). If the 17 is changed to 15 the program will change sets 0, 1 and 2. The eighth number after the set identifier is the number of sets to change and in this case is a 3. Change this 3 to a 15 and all the character sets (0 through 14) will be changed. The second CALL LOAD statement represents the list of colours. In the original program this list contained 3 values (one for each set) and each was a 240. We now want to change 15 sets so we need to add 12 additional 240's for a total of 15. The values in the list (240 in the example) represent the foreground and background colours and can be determined by this colour formula:

((F-1))+(B-1).

In the example the foreground is white (16) and the background is transparent (1) yielding:

((16-1))+(1-1) = (15)+0 = 240

To change a set to a dark green (13) on magenta (14) the result is given by:

((13-1))+(14-1) = ((12))+13 = 205

While the routine is running, a single CALL LOAD can change a character set. In our routine the list of 15 numbers is loaded begining at 16164 such that 16164 has the value for set 0 and 16165 has the value of set 1 and so on. To determine the load address for a single value add the set number to 16164.

For example to change the colour of set 8 to dark red (7) on gray (15) use the statement:-

CALL LOAD(16172,110) where 16172 equals 16164 plus the set number and 110 equals the result from the colour formula above. To sum things up, the routine can be keyed in with or without line numbers. When it is running it has total control of the colours. It can be turned on and off, and when it is off the CALL COLOR statments will affect the character colours normally. A single character set can be changed, or any combination of sets can be changed at once. Hint:

Remember the turn off statement, (CALL LOAD(-31804,0)) because it is possible to make all the characters invisible (transparent)!

Now comes your time to experiment and feel free to contact me.

Joe Nollan
908 E. 35th Street.
TACOMA, WA 98404 USA.
(206) 572-4680

PS ((F-1))+(B-1) is explained by:- Foreground colour minus one multiplied by sixteen, all added to Background colour minus one. For example, ((16-1))+(1-1) which equals 240. In technical terms, it is the hexadecimal value of a two digit byte, Foreground first digit and Background second digit using E/A colour chart.

Re-typed by John Ryan of TIsHUG, 17th February 1988.

NOTE: This programming idea uses the User Interrupt routine and there is more information on this in the article published last month in TND.

o

Changing Defaults for TI-Writer, Multiplan and Fast Term

by Mel Samouri, USA

Here are some instructions for changing the defaults for the TI-Writer formatter printer, the Multiplan printer and data drive, and Fast Term data drive(s) and filenames.

TI-Writer defaults

This is an instruction listing for changing the printer defaults for output from the formatter. All numbers are in **hexadecimal**.

1. Format a blank, single sided, single density disk. (Do not use a disk that has just had files erased, actually format the disk.)
2. Copy the file FORMA1 onto the blank disk.
3. Using a sector editor, such as Disk+aid, read sector 29.
4. At bytes 22 through 53 enter the name of your output device (the ASCII edit mode is easiest for this).
5. Copy the file back onto your working disk and presto, no more having to enter the name of your printer again (unless you have two printers like me, and then just put in the one you use most, or is the longest to enter, like the RS232/2.BA=1200.DA=8.CR).

Multiplan defaults

1. Format a blank, single sided, single density disk. (Do not use a disk that has just had files erased, actually format the disk.)
2. Copy the file MPINTR onto the blank disk.
3. Using a sector editor, such as Disk+aid, read sector 21.
4. At bytes 5F enter the DATA drive number (for example 2). (The ASCII edit mode is easiest for this.)
5. At bytes C4 through EB enter the name of your output device (the ASCII edit mode is easiest for this).
6. Copy the file back onto your working disk and presto, no more having to enter the name of your printer again (unless you have two printers like me, and then just put in the one you use most, or is the longest to enter, like the RS232/2.BA=1200.DA=8.CR).

Fast Term

1. Format a blank, single sided, single density disk. (Do not use a disk that has just had files erased, actually format the disk.)
2. Copy file UTIL1 onto the new disk.
3. Read sector 36 in the ASCII display mode (if possible).
4. Bytes AE through BC are the Xmodem/TEII file transfer name. (e.g. DSK2.SENDFILE)
5. Bytes D2 through E0 are the autolog file name. (e.g. DSK2.LOG)
6. Copy the file UTIL1 back onto your working copy and you are set to go.

*Notice*Notice*Notice* Perform the above procedures only on working copies and not on the master copies in case you make a mistake.

For advanced users of Disk+aid.

You can forego the file copy method by using the ASCII search method, but be careful, because if you are searching for the string "RS232" to find the printer default, it will stop on an earlier sector than you want. At single density, it will stop 9 sectors too soon assuming that the file is not fractured. The byte locations will be your only guide in this case. When they match, you have the right sector.

For the data drive number, search for the string "DSK1." and make sure the byte numbers match! This is the preferred search string as both modifications are in the same sector.

Hope somebody can use this. Mad Mel
On the road again.

o

Disk DSR Tutorial

by Mack McCormick, USA

This is a tutorial for the advanced assembly language programmer. The subject by request is disk DSR routines at the disk ROM and direct access to the controller chip level. In reviewing the information I have on the subject, I find that the entire scope is covered in five books for a total of about 2,000 pages. Obviously, I must limit my discussion. I will break this into several sub-series.

The disk DSR is developed on three levels:

Level 1 - Basic disk functions. Sector Read/Write, head control, drive selections, track formatting and buffer allocation

Level 2 - The "file" concept. Each file is accessible by its name and an offset of a 256 byte block relative to the beginning of the file.

Level 3 - Extension to the user level. Fixed or variable length records or files.

One other level which you will not find documented is direct access to the controller chip in the controller card.

I intend to confine my discussion to level 1 and chip level routines. Due to length, this tutorial will discuss sector I/O. This will be followed in subsequent tutorials by formatting, direct file access, and buffer allocation (e.g. CALL FILES).

There are three different controller chips contained in the three different controller cards on the market (TI, CorComp, MYARC). The chips are all made by Western Digital. They are the WD 1771, WD 2793, and the WD 1770 respectively. Once again I will limit this tutorial to the TI controller card and its chip. Everything in this tutorial will pertain to all three cards, except direct access to the controller chip and its associated commands.

First let us review the TI controller card features and ROM. As you know it can control up to 3 DS/SD drives. There are 40 tracks per drive and 9 sectors per track. Each sector is 256 bytes in length. Track 0 is closest to the outside and track 39 nearest the center of the disk. There is a built in DSR ROM which contains 6 level one routines, which may be executed by branching to them. These will accomplish almost all we need to do, except things like track I/O, Volume Information Block update and others. To get at these routines you must access the Floppy Disk Controller (FDC) chip directly. To accomplish this we need to know how the FDC chip accesses the drive and build from there.

These are some of the features of the WD 1771 chip. Automatic track seek with verification, in the read/write mode single/multiple sector read/write with automatic sector seek. Writes entire track for formatting. Programmable track to track step times. There are six registers.

Data shift register - Assembles serial data from the disk read and transfers during write.

Data Register - 8 bit holding register during read/write operations. During a seek command it contains the desired track position.

Track Register - 8 bit register that contains the track number of the current read/write head position. Incremented by one as the head steps in toward track 39 and decremented by one towards track 0. Contents are compared with the disk track number in the ID field during read, write and verify.

Sector Register - 8 bit register for holding the desired sector position. Contents compared with the disk sector ID field during read and write operations.

Command Register - 8 bit register for the command to be executed.

Status Register - 8 bit register to hold drive status.

Much more on these registers as the tutorials progress.

There are eleven commands available:

Type	Command	Bits 7 6 5 4 3 2 1 0
I	Restore	0 0 0 0 h V rlr0
I	Seek	0 0 0 1 h V rlr0
I	Step	0 0 1 u h V rlr0
I	Step In	0 1 0 u h V rlr0
I	Step Out	0 1 1 u h V rlr0
II	Read Command	1 0 0 m b E 0 0
II	Write Command	1 0 1 m b E ala2
III	Read Address	1 1 0 0 0 E 0 0
III	Read Track	1 1 1 0 0 1 0 s
III	Write Track	1 1 1 1 0 1 0 0
IV	Force Interrupt	1 1 0 1 I3I2I1I0

Plug in the appropriate values by type command:

Type I

h = Head load flag. 1-beginning; 2-not beginning.
V =Verify. 1-verify on last track; 0-no verify.
rlr0 = Stepping motor rate. 0 0 - 6ms; 1 0 - 10ms; 1 1 - 20ms.
u = Update flag. 1-update track register. 0-no update.

Note: Head step times are based on the 1 MHz clock contained in the controller card.

Type II

m = multiple record. 0-single; 1-multiple.
b = Block length flag. 1-IBM format (256 Byte).
Other flags only if need to know.
E = Enable head load and 10 msec delay
1-delay; 0-head already loaded no delay.
ala0 = Data Address Mark 00->FB(Data Mark)

Type III

s = Synchronize Flag. 0-Single density.

Type IV (interrupt condition flags)

I0 = 1, not ready to transisition.
I1 = 1, Ready to not ready transition
I2 = 1, Index Pulse
I3 = 1, Immediate Interrupt

This all seems confusing now but before it is all over you should have a better understanding.

Head loading means the read/write heads are placed in contact with the disk (the click you hear when the drive activates) and data may be transfered. The head stays loaded until a command is received to unload or until a timeout occurs (2 disk revolutions).

I suppose this is the best place to cover the disk format. Have you ever wondered what is in between the data fields (256 bytes), well here it is. Stick with this series and we will write a program to directly look at that data with a track read command.

Number of Bytes	What is there
12	Index Gap. >FF
6	Sync >00
* Sector begins here. Repeat 9 times *	
1	ID Single density >FE
1	Track Address >00->27
1	Side >00
1	Sector Address >00->08
1	Sector Len >01
2	Cycle Redundancy Check >F7
11	Data Separator >FF
6	Sync >00
1	Data Address Mark >FB
256	File Data
2	CRC >F7
* Sector ends here *	
36	Data Separator >FF
240	End of track fill >FF

From this you can see there are 3177 bytes per track but only 2304 are used for actual data bytes.

So far we have covered the basic background. We will discuss what each of these do as we proceed.

There are three ways to perform a sector I/O. You may use the DSRLNK, access the disk ROM without DSRLNK, or access the controller chip directly. Let us examine the first two methods.

Sector I/O is commonly referred to as subprogram 10. All arguments for the I/O are passed through the FAC block in CPU RAM (>834A). Here is how it maps out.

>834A-4B {Address of actual sector accessed when complete.}
>834C Disk drive 1,2, or 3.
>834D Read/Write 0=write; <>0=read
>834E-4F VDP Buffer Address (256 byte size).
>8350-51 Sector Number. Error codes returned at >8350 after operation. 0=no error; 1=error.

I have included another file to demonstrate a straight forward way to read a sector and write it back out to the disk using DSRLNK and direct ROM access. It should be documented well enough for you to follow.

```
*****************************************
*                                       *
*   Sector I/O routine demo using       *
*              DSRLNK                   *
*   Accompanies Sector I/O tutorial     *
*    by Mack McCormick 74206,1522       *
*                                       *
*****************************************
      DEF  SECTOR
      REF  VMBW,VMBR,DSRLNK
PABI  DATA >0110                    subprogram 10
CPUBUF BSS 256                        CPU buffer
SECTOR LI  R0,>F80                  address of PAB
      LI   R1,PABI                          PAB
      LI   R2,2                       two bytes
      BLWP @VMBW                write PAB to VDP
      LI   R1,>0101
      MOV  R1,@>834C          disk drive 1, <>0=read
      LI   R1,>1000
      MOV  R1,@>834E         VDP buffer start address
*                              at least 256 bytes
      CLR  R1
      MOV  R1,@>8350               look at sector 0
      LI   R1,>F80
      MOV  R1,@>8356        point to the PAB at >8356
      BLWP @DSRLNK                access the disk
      DATA >A                use disk DSR subroutines
* Normally you would check for errors at >8350 here.
* You could also check >834A for actual sector read.
*-------------------------------*
*   Put it up on the screen     *
*-------------------------------*
      LI   R0,>1000                VDP buffer address
      LI   R1,CPUBUF               CPU buffer address
      LI   R2,256                 move 256 bytes down
      BLWP @VMBR
* This would be the place to manipulate data before
* writing it back up
      CLR  R0                              sit position 0
      BLWP @VMBW          write up screen image table
*-------------------------------*
*    write back out to disk     *
*-------------------------------*
      LI   R1,>0100                        disk 1, write
      MOV  R1,@>834C
      BLWP @DSRLNK                  write it back out
      DATA >A
      JMP  $          you would exit the program here
      END
* You can see how easy it is to write a sector copier
* just from this short code.  Add a few whistles and
* bells and you have a first class product.
*****************************************
*            Second Example             *
*    Sector I/O routine demo using      *
*         direct ROM access             *
*   Accompanies sector I/O tutorial     *
*     by Mack McCormick 74206,1522      *
*                                       *
*****************************************
      DEF  SECTOR
      REF  VMBW,VMBR,GPLWS
SUBR  DATA >0110                    subprogram 10
CPUBUF BSS 256                        CPU buffer
MYREG BSS >20                        my workspace
SECTOR LWPI GPLWS
      LI   R1,>0101
      MOV  R1,@>834C          disk drive 1, <>0=read
      LI   R1,>1000
      MOV  R1,@>834E         VDP buffer start address
*                              at least 256 bytes
      CLR  R1
      MOV  R1,@>8350          look at sector 0
      LI   R12,>1100          set CRU register to base
*                        address of >1100 disk DSR ROM
      SBO  0         page in the disk DSR ROM to >4000
* Of course you could eliminate the next five
* instructions and manually scan the DSR ROM for the
* word which immediately proceeds >0110 and loaded R9
* with that value which is >56DC in the case of the
* CorComp card and BL directly to it.  I scanned the
* link table so this program could be used with other
* DSR subroutines and with all controller cards.
      LI   R9,>4000          beginning of disk DSR ROM
NEXT  C    *R9+,@SUBR      search link table for entry
*                                          point
      JNE  NEXT
      AI   R9,-4           subtract 4 for entry point
      MOV  *R9,R9         get the entry point address
      BL   *R9                   branch to the routine
* Normally you would check for errors at >8350 here.
* You could also check >834A for actual sector read.
*-------------------------------*
*   PUT IT UP ON THE SCREEN     *
*-------------------------------*
      NOP
      NOP           are required here because the DSR
*                         routine INCT's the RT address
      LI   R0,>1000                VDP buffer address
      LI   R1,CPUBUF               CPU buffer address
      LI   R2,256                  move 256 byte down
      BLWP @VMBR
* This would be the place to manipulate data before
* writing it back up.
      CLR  R0                              sit position 0
      BLWP @VMBW        write up to screen image table
*-------------------------------*
*   Write back out to disk      *
*-------------------------------*
      LI   R1,>0100                        disk 1, write
      MOV  R1,@>834C
      BL   *R9
      SBZ  0                          page out disk DSR
      JMP  $          you would exit the program here
      END
```

o

Odds and Ends

by Mel Samouri, USA

The bit map and sector copy functions of DM1000 work like this.

Bit Map - Only the sectors mapped in the disk directory as being used are copied. This means that fractured files will be un-fractured (if possible) on the copy disk and only those sectors are copied. This is best as it does not copy unused (blank) sectors so it is a little quicker.

Sector - This allows you to "clone" a disk (some software companies use renumbered sectors as protection or purposely fracture files) or part of a disk (blown directory) and if the defaults are used, will make an exact duplicate of the disk. If you are not fairly familiar with how the disk is set up, **use the defaults!** If you do not, you could get only part of your programs copied, as it does not use the sector map. (For double density an extended bit map is used, they are not all in sectors 0 to 20, if you have a lot of files on the disk.

You should, if possible, perform what is called "optimizing" on file disks that you use frequently or delete programs/add programs to, as the files become fractured (part here, part there, etc.), and this can cause added loading times, worn drives and even (worst case) the dreaded data loss. You can tell this is happening if you hear the drive head "seesawing" back and forth when you load your program/file.

I forgot to add that "optimizing" is performed by using the file copy (option 1 of DM1000), onto a clean disk. Use the Sweep Disk option on the original disk (**not the copy disk!**), and then copy the files from the copy disk onto the original disk and presto, no fractured files!

This sounds like a lot of work, but it is all part of file maintenance. The guys (or gals as the case may be) who work for the Source do this regularly, as does anyone who has a hard disk or drum system.

Another small tip that I have not seen suggested is that you make up a set of disks with your favorite programs on them and **put them away!** The only time they should be pulled out is to make another copy. This is called the **master** disk system. If you do this and a disk goes bad or is damaged, all is not lost. Just copy the the Master to get another "working" copy and you are back up and running.

Lots of luck!

o

The Forth Column

Forth Forum <2>

This month I will introduce the essence of Forth programming, writing user defined words. To do that, I will introduce a few words (: . + * ;) and explain word structure. Appendix D of the TI Forth manual contains all of the system defined words available to the programmer. To the right of each word's stack manipulation representation is the file name and the screen number the word can be found on. To use that particular word, that file must first be loaded into the system. If you followed last month's directions, which explained how to create a more efficiently loading system disk, most of these options are already in memory. The only extensions not loaded are: FLOAT; ASSEMBLER; 64SUPPORT; and CRU. If you find a need for any of these options, insert your original Forth system disk and enter '45 LOAD' for FLOAT, '75 LOAD' for ASSEMBLER, '22 LOAD' for 64SUPPORT, and/or '88 LOAD' for CRU.

One exception to the above is the primitives. The primitives are words that are written in assembler and are marked 'RESIDENT' in the appendix. The words in the files are written in Forth and can be examined on your original Forth system disk by editing the screen that contains the words of your choice as explained above.

A person named J. Volk has written numerous programs which can be found in the club library. One of Volk's screens is called 'My Favorite Words'. I looked the words over and had problems understanding the entire routine of his word ROLL, so I decided to rewrite it. Having no success, I decided to shelve the problem until some other time when I might have a need for more frustration. Speaking with Marshall Linker after last month's Fairfax meeting, I was shown how sometimes the most obvious operators can be easily overlooked. Marshall had already written the word using recursion. He also wrote a number of other words using the same idea, which should be presented in a forth coming issue of MANNERS newsletter.

Starting this month, I will present the tutorial, and follow with more advanced ideas, thoughts, opinions, and whatever I come across I think the reader of this column will enjoy.

The Stack

The Forth programmer uses 'words' when programming, which are actually subroutines that perform certain tasks. 'CLS' is a word in Forth which performs the task of clearing the screen, similar to the 'CALL CLEAR' subprogram in BASIC. 'MON' is a word which returns the system to the title screen, the same as pressing FCTN[=] (QUIT) in BASIC. Most words, however, manipulate or leave values on the stack. The stack, sometimes referred to as the parameter stack, is a string of integers accessed in a first-in, last-out arrangement. I think of it in terms of children's building blocks stacked up vertically with a number painted on each one. You can stack them up, one on top of another, and remove them one at a time, from the top down only. Actually, there are ways of accessing numbers other than at the top of the stack, but we will look into that some other time. For now, think of the stack as blocks stacked up vertically.

Putting a number on the stack could not be much easier. Type the number and press [ENTER]. Try it with the number '4' after booting your system. You should receive an 'ok' and a carriage return. Whenever Forth outputs 'ok', that is the system's way of telling you that the instruction you entered has been performed successfully. By pressing '4' and [ENTER], you have told Forth to take the number you input and place it on the stack.

. (n ---)

The period is the first word we need to learn. This word manipulates the stack in that it removes the value at the top of the stack and prints it on the screen. Since you placed a '4' on the stack, the best way to try this word is to go ahead and execute it. Press '.' and then [ENTER]. You should see the number 4 printed on your screen, followed by 'ok' and a carriage return. Next to the word '.' above is the word's stack manipulation. Before execution of this word there is a number on the stack (we will get to the explanation of what happens when there is nothing on the stack later), represented by the 'n'. The word referenced is represented by the three dashes, and the stack results are to the right of this. Therefore, (n ---) means that the number that was on the stack before execution is no longer there after the word has been executed.

Words do not have to entered individually. Often it is more convenient to enter a string of words all at once. Try entering this: '1 2 3 4'. Now, to print the stack we can use '.' four times. Enter '. . . .', being sure that you have a space between each of the periods. Your display should read '4 3 2 1 ok'. Keep in mind that since the number 4 was the last integer entered, it will be the first number to be output.

This brings us to the point where we can begin defining our own words. Suppose that you had a need to print the four numbers on the top of the stack several times in a program. Instead of using four periods each time, it would be more convenient if you could define a word which would perform this task each time referenced. The two words you need to know are ':' and ';'. The colon alerts the system that the next word it reads should be placed in the dictionary, and its 'definition' will continue until the word ';' is executed. So go ahead and enter this definition.

: PFOUR . . . . ;

When the word PFOUR is executed, it will perform the task of printing the top four numbers on the stack to the screen. Let us try it. Enter '4 3 2 1 PFOUR'. You should see '1 2 3 4 ok' on the screen.

+ (n1 n2 --- n1+n2)

The plus sign is used to add the two numbers on the top of the stack. The result of this addition is placed on the top of the stack in place of the two numbers it just added, as shown in the stack manipulation representation to the right of the word above. To try the word out, first place two numbers on the stack; enter '5 13'. Now that two numbers are on the stack, we can add them together by entering '+'. The sum of those two numbers is now on top of the stack and can be printed out by entering '.'. Your system should print '18' to the screen and give you an 'ok'. Of course we could have accomplished the same task by entering everything at once, '5 13 + .' and pressing [ENTER].

Forth uses postfix operations (usually referred to as Reverse Polish Notation or RPN), as opposed to the infix operators we are normally accustomed to. Those with Hewlett Packard calculators already understand this notation. Other calculators will have you enter '5 + 13 =' to add those two numbers. In RPN you must place the numbers on the stack first and then input the operator, '5 13 + ='. As I said earlier, '.' is used to print the answer. '=' provides a logical comparison operator in Forth as we shall learn in several months. If you have trouble conceptualizing RPN, do not bother with it and just think of the words in terms of their definitions, such as '+' (n1 n2 --- n1+n2) meaning that the two numbers at the top of the stack are added, with the sum taking their place.

* (n1 n2 --- n1*n2)

If you understood '+', this word is self-explanatory. This word removes the two numbers on the top of the stack, multiplies them, and returns the result to the top of the stack. Entering '5 13' places these two numbers on top of the stack. '*' multiplies 5 and 13 and places the result on top of the stack in place of the 5 and 13 and '.' prints the result, '65'. Again, it would have been easier to input '5 13 * .' and press [ENTER], as we will do from now on. RPN is not really difficult to understand, it is simply another way of doing things. The more you work with this system, the easier it will become for you. At this point we can begin to practice writing words. One routine you may want is to double the value of a

number. You can name this routine just about whatever you want, with only a few limitations. One limitation is that the word you choose must not have any spaces within it. Spaces are used by the Forth system as delimiters, so that it can tell where a word's spelling begins and ends (logical!). If you want to use more than one word in your word you may concatenate by using a slash (/), or dash (-), or any other printable character. For example, if you wanted to name a word 'DOUBLE NUMBER', you could use 'DOUBLE/NUMBER' or 'DOUBLE_NUMBER', and so on. Personally I prefer the '/' because I do not have to find the shift or function key to enter it.

Another limitation is that your word must be less than 32 characters long. This is not much of a limitation if you consider that you could create the word 'DOUBLE_THE_NUMBER_ON_THE_STACK'. This is descriptive, but time consuming. Most programmers will simply define such a routine as 'DOUBLE', or as I prefer, 'DBL'. Whatever your choice, make it descriptive of the function, and make it convenient.

Since we know that the new word's definition will start with a colon and end with a semicolon and be named 'DBL', we can write down this much.

```
: DBL ;
```

Now all that is needed is to put the definition in there. Since we double numbers by multiplying them by two, the definition would be.

```
2 *
```

This completes our definition-> `: DBL 2 * ;`

The stack manipulation would be DBL (n --- 2*n). If you entered this definition you can use it by placing a number on the stack and executing the word. '13 DBL .' will write the number 26 on the screen, as this word DBL places a 2 on the stack, on top of the number you entered before DBL, multiplies the two numbers, places the answer on the top of the stack, and prints the answer on the screen.

Play around with these few words and try to become familiar with word structure, as we will continue next month with a bunch of new words to learn. The more you work with this system, the sooner everything will all seem to fit together.

Last month I offered a new system disk for TI Forth which included dark blue letters on a light blue background, and said that if this choice was not yours, you could change it. On screen 3, line 6, the set '71 7 VWTR' determines these colours. VWTR stands for Video Write To Register, which is a word that writes a byte to a particular register, but do not worry about all that yet.

Type this in.

```
: X 255 18 DO I DUP 7 KEY DROP VWTR . ?TERMINAL IF
  LEAVE THEN LOOP ;
```

Now enter 'X'. The number displayed will be the value of the byte that, when written into VWTR number 7, will produce the displayed screen and character colours. Continually press any key until you come across the colour combination you most enjoy. If you make your decision before the entire cycle of colours is complete, enter FCTN[4] (CLEAR). If you want to see the set again, after it has completed its cycle, enter 'X' again.

Now that your decision is made, enter 'FORGET X 3 EDIT'. This erases the word X from your dictionary and lists screen number 3 to your display, with the cursor in the upper left hand corner. Use the arrow keys to place the cursor over the '71' and change this to the value of your choice. To write this to your disk, enter FCTN[9] (BACK), then enter 'FLUSH'. The next time you boot your system disk, your own colours will be used.

Next month I will introduce words that manipulate the position of values on the stack, talk about the system disk, and (perhaps) present a loader program for your programs and the programs in the club library.

The Forum

Whenever you use DO ... LOOP in conjunction with R> and >R, chances are that you may get yourself in trouble. With this knowledge I was still able to lock up my computer on numerous occasions in my attempt to rewrite ROLL, J. Volk's word which 'rolls' a specified stack position to the top of the stack. Supposedly, this is not good programming practice, but you can go one of two ways on it. You can follow 'good' programming practices, which should lead to good habits when you become more familiar with what you are doing and begin writing more extensive programs. Or, you can take the route whereby you use whatever you can to accomplish your purpose. If nothing else, this is an example of recursion and should be understood by the budding Forth programmer. Here is Marshall Linker's word.

```
: ROLL ( n --- ) ( rotate n-1st number to the top of
  the stack ) -DUP IF 1- SWAP >R MYSELF R> SWAP THEN ;
```

By the way, Marshall used ENDIF, but BASIC being the first language I learned, I feel more comfortable using THEN. Choose for yourself.

If you do not examine this word's makeup, you may be confused when you try to use it. Just as Forth convention places 'column' before 'row', we must also keep in mind that Forth convention assumes the number 0 as the first number in the numbering system. If you have trouble with this, think in terms of how many numbers below the top of the stack the number you wish to 'roll' is. For example, if the stack values are: ¦ 5 4 3 2 1 0 (the symbol '¦' is used to indicate the bottom of the stack), and you wish to roll the number 4 to the top of the stack, since it is the fourth number below 0, you would use '4 ROLL', leaving the stack: ¦ 5 3 2 1 0 4 .

Try this word out and experiment with the stack. Remember, if you are using the Forth system disk I described, or if you load -DUMP, you can use the word .S to examine the stack.

Faster Forth

Forth is not as fast as assembler, but in large applications comes very close. It utilizes a technique called 'threaded code' which makes a list of addresses which point to executable instructions. These instructions are not assembler instructions but are primitive routines written in assembler which define the kernel of Forth. This inner interpreter which interprets this list of addresses consumes up to half of the run time of the primative words. Although Forth does not use registers, since it is a stack based system, the machine instructions on which the primitives are written for the host computer do use registers to maintain the stacks and perform various arithmetic operations.

One way to get around the speed degradation would be to design a computer which assimilates a true stack based system, embodying the ideas inherent in the Forth language. Metaforth Computer Systems has built a computer with this idea in mind. It has a single board processor, using bipolar logic, and uses HMOS (High-performance Metal Oxide Semiconductor) RAM chips for hardware stacks. Using a 10 MHz clock, the production model will run programs over 100 times faster than a Z80. The design lends itself well to LSI design, so we may see something of this in the not too distant future.

Well, that wraps it up for this month. As always, if you have a contribution to the column, write to me or leave me a message on one of the TIBBS boards.

George L. Smyth
3017 Sylvan Drive
Falls Church, VA, 22042 o

continued from page 8

There is a neat feature called zoom which magnifies sections of the picture all the way down so that 12 pixels fill the screen. You control the amount of zoom with the Mouse buttons. Another plus feature is that you are able to control the speed of the drawing pencil, which they call the icon, using the 10 number keys. With 10 different speeds even a shaky hand like mine can draw very intricate artwork. I was impressed. The only minus feature I could find was that I could not use any artwork from the other graphics programs like: TI-Artist; GRAPHX; CSGD. They simply will not load. Maybe some smart programmer will correct this in the future as I (like many others) have a large collection of really beautiful artwork including many RLE pictures. The price of this Mouse and My-Art package is $125.00 plus handling charges.

Its time to play with my toys. Bye for now. o

A Guide to the UCSD p-System for the TI99/4A

Author unknown, USA

Introduction

When TI discontinued the machine in October, 1983, few of the over two million TI99/4A owners were using the p-System. If all TI99/4A owners felt like orphans, we p-System users were doubly so. But however isolated we seem, there are advantages. The p-System on the TI99/4A is a full implementation of a "serious" operating system and as such we are able to do much more with it than TI-BASIC. Also, we can get help from any p-System user, regardless of their hardware. Finally, when the day comes when our machines either die or are outgrown, our programs will be portable to virtually any other machine on the market.

The purpose of this guide is to orient the TI99/4A user to the p-System in general, and point out the special considerations about its implementation on our machine. It is hoped that this guide will be of value to all who are interested, from the serious programmer to those who are considering purchasing the p-System

Overview of the p-System

The p-System is an operating system, a set of programs to run the computer and allow us to write and run programs. It is not a language. Any language might be available under the p-System. While only Pascal is currently available to us, other users of the p-System can use BASIC, FORTRAN, and Modula-2.

This concept is confusing to many TI99/4A users, because TI Console and Extended BASIC have the operating system built in, they include text editors, disk management programs, etc., all in the ROM containing BASIC. In the p-System, one addresses all these functions separately, and most are on disk.

The essential part of the p-System for the TI99/4A, is the p-Code card (or the old p-Code peripheral). The p-Code card contains some of the system programs in ROM and GROM. You must also have the 32K memory expansion. There is currently no way to use the extra 96K RAM available in the 128K card manufactured by Foundation. A disk is not absolutely necessary, as programs can be loaded and run from cassette. However, I do not know of any cassette programs, nor do I know of anyone who has used a cassette with the p-System.

The p-Code card includes, in GROM, the p-Code interpreter. This program accepts the user and system programs in p-Code and executes them by interpreting them into 9900 machine instructions.

Anyone serious about the p-System will have to have at least one and preferably two disk drives. The p-System will support double sided disk drives. If you are going to make use of the p-System for writing programs, you will find that with a single drive setup your drive had better be double sided. Two drives are better and two double sided drives are best. I see little use for a third drive. A printer is certainly useful for the serious p-System user and can be supported through the RS232 (and PIO) card.

There are several programs that are important parts of the p-System. One is the Pascal compiler, the program that takes Pascal programs and translates them to p-Code, so the p-Code card can run them. This program comes on disk and is necessary to write your own programs but not to run someone else's programs.

Another disk contains two important programs, the Editor and the Filer. The Editor is used to edit documents and programs. You only need this to write your own programs, although it can even, to some extent, be used for word processing. The Filer manages the disks. Virtually any user of the p-System needs this program. Also on this disk are a number of utility programs which are of much value to most users.

The last disk contains the Assembler and Linker, both of which are necessary to do assembly language programming.

Learning to use the p-System

The TI manuals that come with the system components are pretty good, as manuals go, but are not designed to teach the basic use of the system. There are many books about the UCSD p-System and Pascal. Two very useful books are: "Introduction to the UCSD p-System" by Grant and Butah (Sybex), and "The UCSD Pascal Handbook" by Clark and Koehler (Reward Books). The former is an excellent introduction to the use of the p-System. This guide will describe the unique aspects of our implementation, by comparing the TI99/4A system to that described in this book. The latter book is a relatively technical guide to UCSD Pascal with very good examples. There are a number of good books that teach Pascal. A discussion of them is beyond the scope of this guide.

USUS (the UCSD p-System users' Group), is a good source of help in the use of the system. Many of its members can easily be reached via the MUSUS SIG on Compuserve (PCS-55). The beauty of the p-System is that it, at least in theory, acts exactly the same on all systems. Thus, you can get help from any knowledgeable p-System user. MUSUS is a good place to find such people.

The Filer

The Filer works pretty much as described by Grant and Butah. The major exceptions are that some of the prompts are shorter and the TI99/4A only displays 40 columns at once. To see the whole 80 column display one uses the FCTN[7] and FCTN[8] keys (screen left and right).

It should be noted that disks may be physically in the drives, but cannot be addressed by their volume names unless they were present at startup of the p-System, or at reinitialization, I(nitialize command, or by invoking the V(ols command of the Filer. This command does not just list the on-line diskettes, but formally recognizes them so that they can be accessed by volume name in user programs. If you want to use the Filer but did not have it in a disk drive at start up, then you have a problem. You cannot invoke the Filer V(ols command to get the system to recognize the disk - because you cannot run the Filer. The solution to this catch-22 is to use the system I command and restart the system. On restart, the p-System looks anew at the disk drives, and will then recognize the diskettes loaded in them, and know where to find the Filer, Editor, etc.

The Z(ero command sets up a new (or zeros an old disk) after it has been formatted by the DFORMAT utility program (or the Disk Manager Cartridge). Single sided disks have 180 (512 byte) blocks, double sided disks are twice that size. Using duplicate directories consumes 4 extra blocks but will often avert disaster when disk failure occurs.

The Editor

The Editor for the TI99/4A is that which Grant and Butah call the "Screen Oriented Editor", which is the Editor that is discussed at length in their book. In general, the Editor works just like that of Grant and Butah with a few differences that could probably be best described as bugs. The "S" (Same) option on the "F" (Find) command does not work. The copy buffer seems to be emptied at times for reasons that are not obvious. After repeated insertions and deletions of a given page on screen, often the screen image is not correct. The actual text is different. As a result, it is best to use the "V" (Verify) command when in doubt. It re-displays the current screen page.

The TI99/4A's small RAM size allows only a little more than 12K characters in the text buffer. If you need to break down a larger file the AUTOPSY program in the USUS library can be of help. Programs can be written in several parts and strung together with the Pascal compiler {$I} (include file) command.

Compiler

The TI99/4A Pascal compiler is a full Version IV.0 implementation. It seems to be identical to the compiler described by Clark and Koehler. In its TI version, the Compiler appears to be in four parts:

1) the start-up code;
2) the code to process the Declarations,
3) the code to process the body of procedures, and
4) the code to produce p-Codes.

Parts 2 and 3 are swapped in and out for each procedure in the program, making compilation a good time for a coffee break. The swap parameters described by Grant and Butah seem to be inoperative on the TI. I suspect that the compiler is set for maximum swapping to accommodate the TI's limited RAM. Making a listing causes the already slow compiler to run much slower still, so one should be sparing in using this function. Also, if errors are found by the compiler it may not make a listing, even if the user continues compilation to the end. It may even hang with the error:
STACK OVERFLOW*REBOOT

At this point, one must turn the console off and on again to restart.

The TI99/4A version of Pascal includes a number of extensions that allow the use of the special capabilities of the machine. These extensions are in the form of procedures included in the "TEXAS INSTRUMENTS UNITS", which are in the system LIBRARY. These include the routines to change screen color, use graphic modes (bit map, multi-color, etc.), sprites, sound and speech. The use of these routines will allow the sophisticated user to do in a high level language what otherwise was previously only possible in Assembly. The new TI Forth possesses these features as well. Please note, however, that the use of these procedures will make the program machine dependent, and thus not portable. One cannot, for example expect a program using sprites to run properly on an Apple, because that machine lacks the hardware to produce them. To keep a program portable, one should avoid using the facilities in the TEXAS INSTRUMENTS UNITS.

Among the routines included in these UNITS, I have only used the procedures for random number generation and screen color definition. These procedures work as specified.

For program development, it is desirable to be able to quickly go from compiler to editor, so it is good to have both programs on one disk or both on different disks on different drives (in a multi-drive system). Also, put the file SYSTEM.PASCAL on the ROOT volume so that meaningful error messages are displayed.

Utilities

There are a number of valuable programs included with the Editor and Filer disk. This guide will discuss the most useful of these programs.

DFORMAT formats new disks (or reformats old ones). It serves the same function as the Disk Manager module in this function. It can format double sided disks but usually fails (I was told by TI that they knew of this bug) but works fine on single sided disks. For double sided disks, one can always use the Disk Manager 2 module if available. The Z(ero command in the Filer must still be used in either case before a disk is usable.

MARKDUPDIR and COPYDUPDIR are useful in dealing with duplicate directories on disk, whose use is highly recomended.

MODRS232 alters the device definition associated with REMIN: and REMOUT: (these use the same definition) and PRINTER:. TI has kindly included the source code for these programs. If you use a printer whose definition is different from the default (RS232/2.BA=9600.DA=7.PA=0.EC is probably not what you are using), you can alter the source code to make the printer definition without user prompting and run it as SYSTEM.STARTUP. The rules for describing the RS232 or PIO communication setup are the same as used by BASIC.

RECOVER is useful if both directories are blown (or you foolishly used only single directories). It can help restore lost files. PATCH will allow repair of individual files. You have to be a very expert user to use PATCH, however.

SETLTYPE allows one to force a program to reside in RAM rather than VDPRAM. Its stated purpose is to declare whether a program is in p-Code or native code (machine language). Machine language programs must reside in RAM. p-Code programs may be in RAM or VDPRAM as the p-System selects (usually in VDPRAM). The system views both VDPRAM and RAM as one large space to be filled with programs and data. It will store data down from the high end of RAM and programs up from the low end of VDPRAM until RAM or VDPRAM is filled. Then it will overflow into the other space (i.e. programs will overflow in RAM or data into VDPRAM). One can use SETLTYPE to force a program or segment to be in RAM. If this is done it will run somewhat faster, but may impinge on (and thereby limit) the area for data. As mentioned above, assembly language programs must be put in RAM with SETLTYPE.

Using the p-System

Several notes can be made about using the p-System on the TI99/4A. First, you cannot use the TI LOGO cartridge on a system with the p-Code card in place. The p-System will work fine. LOGO will, in certain circumstances, lose control to the p-Code card, destroying what you were doing with LOGO. If your p-Code card has an on/off switch, all will be well if you turn the card off before using LOGO.

When a TI99/4A is powered up with the p-Code card in place (and turned on) the p-System initializes and then displays the p-System greeting. The p-System remains active until the user stops it. Thereafter it will not restart without using the technique below (or turning off the system). To get from the p-System to the BASIC PIO communication setup are the same as used by BASIC.

RECOVER is useful if both directories are blown (or you foolishly used only single directories). It can help restore lost files. PATCH will allow repair of individual files. You have to From console or Extended BASIC, execute the following command.

CALL LOAD(14586,0,0)

and then type FCTN[=] (Quit) or the BASIC command 'BYE'. The p-System will then operate as if you had turned the system off and then on again. Some programs and cartridges seem to have the same effect as the CALL LOAD listed above. Exiting TI-Writer, Multiplan, or the Companion word processor will lead to restart of the p-System. Likewise, some of the commands in the Disk Manager will reset the p-System so that it will become active again upon leaving this module. Speed is an issue of interest to all potential users of the p-System. If computing speed is the reason for the TI BASIC programmer to consider the p-System, the following facts apply. Because the p-System is interpreted, it is relatively slow compared to compiled Pascals (such as Turbo Pascal on MSDOS and CP/M systems). Compared to other machines, the p-System on the TI99/4A is quite slow. I believe it is the slowest implementation of the p-System around. TI BASIC, however, can be benchmarked with a sundial, so it is easy for the p-System to beat it. Number crunching programs in Pascal can run up to 60 times faster than their BASIC counterparts, especially if put in RAM (see SETLTYPE). Programs doing mostly screen I/O can be slower than BASIC, however. Disk I/O seems a little faster with the p-System, perhaps because of the simpler file system, leading to less searches for the next block etc. I have no experience with Forth, but know that it too is faster than BASIC. The programmer desiring speed above all else should compare the p-System to Forth.

On running the V(ols command in the Filer program, you will notice that there is a device #14 called 'OS:'. If you do not have a disk in drive one (the usual root device), one will find that the 'PREFIX' and 'ROOT' devices are defined as 'OS'. This device includes files that are in ROM and are used for booting the system. You can read and examine these files but not, of course, modify them. To change the PREFIX definition, use the Filer P(refix command. To change the ROOT definition you must use the system I(nitialize (warm start) command.

Compatibility with other Machines

The p-System on the TI99/4A is very compatible with other p-System implementations. Source code developed on the TI99/4A can and does run without modification on most other machines, if you avoid using the TEXAS INSTRUMENTS UNITS included with the Pascal Compiler. The TI99/4A is less able to run programs from other machines due to its relatively small RAM size. If the program is properly segmented, however, there will be no problem. The Pascal compiler, for instance, is TI99 blocks long, 49.5K bytes. It runs on the TI99/4A with a lot of disk swapping. The only other problem in running program from other machines on

the TI is the presence of a 40 (rather than 80) column display. The TI will display all 80 columns with the Screen Right and Screen Left keys, but you will probably want to modify the display (if possible) to fit the screen. In the file SCREENOPS.CODE on device #14 (see above) is a description of the TI screen size. This file can be read by a program. This file tells the program in the case of the TI that the screen is 40 columns. Clever programs will adapt themselves to the size of the screen.

A larger problem is physically transferring programs to and from the TI99/4A. The TI p-System lacks any functional communications program at this time. Files can be exported from the TI99/4A easily by using the Filer to T(rans to REMOUT:.

There is good but not complete compatibility with other Pascals. I have programs that I wrote on the TI p-System that I ported to an MSDOS (IBM PC or compatible) machine and converted to Turbo Pascal. Little modification was necessary.

Software for the TI p-System

I know of no commercial software for the TI p-System. I suspect that if enough interest could be shown, then the whole spectrum of existing p-System software would be made available.

USUS maintains a large library of programs, most of which are not machine dependent. These volumes are available to USUS members for a small fee, in TI format. Most programs have not been modified to run on the TI. If you modify a USUS program to run on the TI99/4A, you are requested to send a copy back to the USUS TI distributor.

Wish List

The p-System on the TI 99/4A is powerful but remains largely undeveloped. What follows are my sugestions for software developments which would greatly aid the use of the system.

FORTRAN: could someone port this compiler to the TI99/4A? There would surely be great interest in this language on the TI99/4A. Likewise, many would welcome Modula-2.

Terminal Emulation: There have been programmers who have had terminal emulator programs "almost working". This is probably the highest priority for the p-System. A terminal emulator program would allow porting of other programs to the TI99/4A without re-keying the source code.

A good Data Base: there are no high quality data base programs available for the TI99 in any form. Several excellent programs are available on the p-System. Could one be ported to the TI99/4A?

Help!

The information in this guide is essentially all from my own meager experiences. Like most p-System users on the TI99/4A, I doubt that I have ever laid eyes on another user. This guide has been written, therefore, in a vacuum. Its purpose is to catalyze discussion, debate, and the creation of a better guide to the TI p-System.

To improve on this guide, I need the experiences of other users, both the sophisticated and the novice.

This guide could be improved with the following information. Is there anybody out there who can help with machine language programming on the system? How about how to change character sets? Anybody have any suggestions or other tricks for use for the p-System? I solicit your comments, suggestions, and criticisms. If you feel the urge, trash this guide and write your own. We all will benefit by the interchange of information and ideas. o

continued from page 7

While the 9640 may not be the key to the survival of our ever shrinking community, the negative non constructive criticisms against the clone or its small, (small compared to TI), manufacturer and unhealthy bickering amongst various camps of users will only drive more nails into the TI99/4A coffin, you better believe it! So the next time you have negative words about Myarc or any other supporter of the TI99/4A orphan, keep them to yourself or save them for those "other" computers and help promote a positive attitude, which is the key to your orphan's continued survival. o

Myarc Mystery

by Ben Takach

On one of those rainy evenings my TI99/4A and I were having a bit of a bash. I was running one of my often used programs out of the Myarc RAMdisk when suddenly the program went crazy. I mean it behaved rather oddly. It responded normally to the inputs, accepted the keyed in values, seemingly executed the calculations, it even printed out the strings preceding the results, but the screen was absolutely blank where the results should have been. I have listed the program, reset the machine, displayed the RAMdisk directory, shifted the RAMdisk from one drive to the other, then run the program again from the disk this time. You guessed it, no results. I have switched off the computer, removed the battery backup supply from the RAMdisk, then switched the system on again. The RAMdisk was not recognised, which is fine, it was not initialised. After partitioning, I was able to load it with files and run it, catalog it, but it still did not display the values of any numeric variable, not even a 0 (zero)! Strings were fine. At this time, not having a single clue to the source of the evil, I BYE'd out of the sordid mess and tried some simple keyboard tests, booted up through the TI Extended BASIC cartridge, which I have not used for ages.

I entered in command mode: R1=234 :: R2=123 :: PRINT R1/R2 then ENTER. The cursor obediently jumped down a line, but there was nothing to see there. Just then I remembered the magic Myarc words, the self diagnosis routine "CALL RDTEST". Well, I was greeted with the well known unfriendly TI99/4A burp and the SYNTAX ERROR message. Out with Myarc and in with the old standby, the TI 32k Card. Which off course looks so simple on paper! Did you ever worked out how a well organised station can turn into an unholy mess in 10 seconds just because you have decided to lift the top cover of the PE box? Just about then Mrs. T. entered with a cup of coffee, sporting her "well I am off to bed, you can rot here with your precious toy for all I care" look, searching for a few square inches of flat surface to put the cup down, and uttered one of those remarks which landed many a good husband behind bars for manslaughter: "Why are you wrecking it now?". The machine behaved perfectly with the TI card. I was looking at the many chips of the Myarc card itching to go for the phone, but alas midnight is no time for a serious chat. Then as a last defiant act I plugged the Myarc card in again. The darn thing has worked like a charm ever since.

The moral to the story? Just because I chickened out at the midnight hour, not you nor I will ever know what John Paine would have said if he heard his telephone ring. o

Letters to the Editor

from George Meldrum, IRG

In the May issue of the TND, a rather tedious method of moving large cassette programs to disk was suggested. I would like to recommend a quicker and easier way. The hint is use the MERGE program. The MERGE program was primarily designed to merge cassette tape programs in a similar way as the MERGE command works with programs stored on disk. It can however be used for transferring programs from cassette to disk by the following method.

Load MERGE - >CALL INIT::CALL LOAD("DSK1.MERGE")

Then repeat these steps for each cassette to disk transfer -

1) Clear memory - >NEW
2) Load cassette - >CALL LINK("MERGE")
3) Save to disk - >SAVE DSK1.filename

The MERGE program will load any size Extended BASIC cassette program - including those extra large cassette programs that will not load even after a CALL FILES(1) command. In fact it is not necessary to use the CALL FILES command at all when using the MERGE program. Assuming you have not executed a CALL FILES command then programs saved to disk are done so under the normal CALL FILES(3) default. A word of warning though, watch out that you type 'NEW' each time before loading, else you may end up with more than you bargained for! o

continued from page 1

I received a letter from the directors with praise for our efforts on the May News Digest. We, the workers, thank you for your kind words and will try to maintain the standards we have set ourselves. I hope that we will continue to please you all, and that some of you will be pleased enough to try out your word processing skills and send us a letter telling us what you like (or dislike) about our efforts. It is rather lonely sometimes sitting at the keyboard wondering if anyone is going to read this tripe, I mean these words of wisdom. The letter from the directors stated that some copies had blank pages. Unfortunately copies like this seem to creep through. Apart from checking every copy for this problem, which I do not have time to do and then to get the TND to you all in a reasonable time, all I can offer is that anyone with a copy with blank pages should return it to me and I will replace it for them. It did happen a few months ago, but those copies were sent to people in my area by chance. I assume the problem occurs at the end of the print run and should only be a very few copies. As to the quality of the print this month, I think that it was a rushed job with originals which were printed with a cloth ribbon on my printer, which looked dark enough but did not duplicate as well as the carbon ribbon we usually use. You will notice that the headings came out as well as usual, as they were done with a laser printer. By the way, does anyone like (hate) the headings? It would be nice to be told if we are on the right track, or do we assume that "no news is good news"? I have the printer back now with the correct print wheel, and it is working better but still not 100%. After this edition is finished I shall have to send it in for further surgery. We are also trying to get the size of print increased by not reducing as much. One impediment to this is that we use a preprinted paste up sheet which has the borders on it. If you look at a page of the April TND, you wil notice that the margins at the top and bottom are much less than those at the side. We will try, this month, removing the area at the bottom which is only used for the page numbers, and ask for less reduction as a consequence. The page numbers will appear in the top part. Tell us if you find that better. Perhaps we should run a competition for the design of a new layout page, as we will need to print some more of those in a few months.

****

I was unable to attend the last Sydney meeting (the captain of my tennis team required my presence) and neither was anyone from our group, so that I do not have the overseas magazines to summarise for you. If you wish me to continue to do this, perhaps you should write and tell me. This has left a bit of room for other ramblings, so I thought I might take up some space and tell you what we are trying to do as far as the quality of the TND. In the process I will explain the steps involved in producing the TND and perhaps help any prospective and current authors prepare articles in such a way that they are easily incorporated into the magazine. We are trying to produce a magazine which is consistent in its presentation and with as few errors as possible. Consistency means that we need to produce all the material on the same printer using the same daisy wheel and font and with the same pitch, and probably edited by just one person. Of course we take a few liberties with the headings, and there are a few other problem areas which I will mention as I go along. We have standardised on two columns of 55 characters in each, at 12 characters to the inch and justified on both sides. As long as the words are not too long, this gives a reasonable use of the area without too many spaces in the line. The problem areas are when programs are printed. For the programs to type in part, we have printed these as you would see them on the screen, namely 28 characters wide and in three columns on the page. We probably should use 10 characters to the inch for these as there is plenty of room and this may make them a bit more readable. BASIC programs elsewhere in the magazine are formatted to try to make them more readable by indenting the line numbers. In order to keep the consistancy, almost all files need to be edited.

It is almost unknown for a file to be printed without some form of massaging to make it suit my standards. I try to remove all spelling errors and replace all unnecessary abbreviations such as & and don't etc. I also refer to the computer as TI99/4A in full and use Extended BASIC for XB etc. I have quite a few preferences like that which you may object to if you wish, but I try to be consistent in what I do. I also try and reduce the use of unnecessary capital letters, as these tend to be less readable than using upper and lower case. One of the hardest contributers to edit is Jim Peterson, whose articles contain very few errors (except for American spellings), but they are all formatted to 32 columns and contain a mixture of text and programs. Because of the narrow column, he uses hyphenation which has its own problems when reformatting. Let me explain what I have to look out for when editing one of his files.

Since I want a particular column width and right justified, I must use the formatter. Also I need to use word wrap mode in the editor to make sure that nothing is lost off the end of a line when inserting characters. The first thing I do when I have the file in the editor buffer is to go through the file, putting in end of paragraph characters at appropriate points to suit the text. Then I use the tab command of the editor to set the right margin of the editor to 39 which gives 40 characters on the screen. Then I go through the file reformatting all the paragraphs to the 40 characters, and putting in the formatting commands to control the margin and indent to make the result like the original. Then the hyphens need to be removed, and after formatting they are actually "- " in the middle of the word. These can be removed using the replace string command of the editor, but if any hyphens are still at the end of a line, these need to be treated separately. To remove these I use the insert character command just before the first half of the word, then move the cursor down one line and to the left to give enough room on the line for the whole word and then reformat. Then the characters in the middle of the word can be removed and the paragraph can be reformatted.

The program segments are more of a problem. Each program line is treated as a paragraph, but there may be an unwanted space every 28 characters. These need to be removed where they are in the middle of a word or number, but spaces may need to be inserted at suitable points (after commas say) to allow the line to be broken just before column 54.

Once all the paragraphs have been reformatted, I return to the top to put in the format commands and to check for any corrections as I read the file. The first format command is ".LMO;RM54;FI;AD;IN+5" followed immediately by a <cr> (if a space follows the number the command will not work. The title and author are usually centred (.CE2) with the title underlined using &. For the program I like to do a ".LM4;IN-4;NA" to make the the line numbers stand out and to turn off the adjust to avoid introducing extra spaces. Any tables that will fit without problems are left as they are by using ".NF" before and ".FI;AD" after the table. When I have done all I can see, it is time to print it and see what it looks like. The formatter puts 2 spaces after every "." and does not print "*12", an asterisk followed by two digits. Other characters which cause problems, particularly in programs, are & and the carat. In assembler programs @ is a problem, while in Forth programs "." at the start of a line can be a problem. Forth screens need to be 64 characters wide, so we have decided to photo reduce them first to fit them into the 55 character width of our columns. Unless I am very lucky, there is some problem which shows up in the printing and has to be corrected.

All of this has taken up to 3 hours for one file and easier ones will take about 15 minutes. The printout shows other problems and it is easy to miss something on the screen which stands out once it is printed. While I am doing this, Rolf Schreiber my layout man extraordinaire, is pasting up articles as fast as I give them to him. One of our last big problems is to make a cover. Anyone have any ideas for covers? Help!

****

continued on page 31

Regional Group Reports

Meeting summary.

Banana Coast	12/6/88	Sawtell
Carlingford	15/6/88	???
Central Coast	11/6/88	Toukley
Glebe	9/6/88	Glebe
Illawarra	20/6/88	Keiraville
Liverpool	17/6/88	Moorebank
Northern Suburbs	23/6/88	Mona Vale
Sutherland	17/6/88	???

BANANA COAST Regional Group
(Coffs Harbour area)

For information on meetings of the Banana Coast group, contact Keir Wells at 9 Tamarind Drive, Bellingen, telephone (066) 55 1487.

CARLINGFORD Regional Group.

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

CENTRAL COAST Regional Group.

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

GLEBE Regional Group.

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

The activities at these regional meetings are rather informal and include looking at new hardware, hardware repairs, looking at new software and having a general chat.

ILLAWARRA Regional Group.

Regular meetings are normally on the third Monday of each month, except January, at 7.30pm, Keiraville Public School, Gipps Rd, Keiraville, opposite the Keiraville shopping centre. Contact Bob Montgomery on (042) 28 6463 for more information.

LIVERPOOL Regional Group

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

Last meeting was at Ross Hardy's house. We turned off the pre-scan in one very big program to show the difference in speed when it started up. John demonstrated some RAMdisk tips.

Meetings coming up.

June 17th 1988, at Hans Zecevic's house,
33 Malinya Crescent, Moorebank, 600 8716

This will be a demonstration of TI-keys and EZ-keys and, if it arrives in time, a demonstration of the new Star NX1000 Printer. If I can work out the problems, I will do a demonstration of the GRAM-RAM but at the present time I have not got it to work at all with my Computer. I had a power problem. It is drawing too much power from the PE box and crashing my other RAMcard.

I also had a meltdown of my main transformer. That is now fixed.

NORTHERN SUBURBS Regional Group.

Hello, just a reminder from the Northern Beach's Regional Group that we meet monthly on the fourth Thursday of the month at various locations. We have a lot of fun trying different things. We welcome any visitors. Our next meetings will be held as follows:-
Thursday 23rd June at Craig Sheehan's home, 21 Suzanne Rd, Mona Vale. Topic: Using PRbase.

Future meetings will be advertised at least two months ahead. See you at the next meeting. Contact Dennis Norman on (02) 452 3920 or Dick Warburton on (02) 918 8132 for further information.

SUTHERLAND Regional Group.

Any persons interested in joining the Sutherland Group on the third Friday of each month, are more than welcome. The format of the meetings are very informal as more often than not the conversation digresses onto matters purely social rather than related to computerisation. The supper is not bad either.

Meetings are held on the third Friday of each month. Group co-ordinator is Peter Young, (02) 528 8775. BBS Contact is Gary Wilson, user name VK2YGW on this BBS.

TIsHUG in Sydney

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

Meetings planned this year. Please note the October date on your planner calendar.

4/6/88 9am, Full day Tutorial.
2/7/88
6/8/88
3/9/88 2pm, Swap meet.
1/10/88 9am, Full day Tutorial.

All TIsHUG Regional groups are invited to submit items for this column. Send details as mail to SYSOP on the BBS.

The information presented here is obtained from the BBS. Some Regional groups are not advising of their meetings, which makes the maintenance of this file on the BBS very difficult.

continued from page 30

I have included for you the list of books and magazines supplied by Warren and Russell Welham. I am a little unhappy at not being able to edit it into a better form but it was supplied to me as a program which produced the printout. Perhaps the next time we do this I will have time to make it look a bit better for you. I have also included the program written by Russell to print out the listing. Unfortunately it resets the printer to 10 characters per inch and sends out two "\" characters at the end of each page which pauses the printer buffer I use to make life bearable. My printer also does not like backing up the paper either, but apart from that an interesting program for you to learn from.

****

Congratulations to George Meldrum from our own Illawarra Regional Group for taking out the grand software competition prize. We have known for some time of George's talents, as he has provided excellent quality software to help our group members obtain the most from their systems, particularly those without disks. It sounds like the competition attracted some very good software which makes his win all the better. Carry on computing, George!