TISHUG NEWS DIGEST

Focusing on the TI99/4A Home Computer

Volume 11, Number 9 October, 1992

Registered by Australia Post - Publication No. NBH5933

TIsHUG's TI-Faire 28th and 29th November Make your arrangements to be there now!

Sydney, New South Wales, Australia $3

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

TIsHUG News Digest

October 1992

All correspondence to:

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

Membership and Subscriptions

Annual Family Dues	$35.00
Associate membership	$10.00
Overseas Airmail Dues	A$65.00
Overseas Surface Mail Dues	A$50.00

TIsHUG Sydney Meeting

The October Meeting will start at 2.00 pm on 3rd of October at Ryde Infant School, Tucker Street, Ryde. There will be an Assembler Class before the meeting starting at 10.00 am.

Printed by
The University of Wollongong
Printery Services.

Index

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

Wanted...
Members contributions to the BBS and the TIsHUG News Digest. Everyone can help, please do!

Wanted... Contact Ross Mudie,
A dual RS232 which plugs into the expansion port of the TI99/4A console. I am not sure if the original TI box-car RS232 expansion provided two RS232s or just one, otherwise a Peter Schubert mini-expansion RS232. Even just the RS232 mini board would do. phone (02) 456 2122

Editor's Comment

by Bob Relyea

By the time you read this I will be well-and-truly in the midst of my 4 week holiday in the States. I am originally from the U.S. and came to these shores as an immigrant some 20 years ago. I have not seen my father for 15 years and my nephew is getting married in Tennessee so it is a good time to combine the family reunion with the wedding reception and some good relaxation. I will be in the sunny state of Florida most of the time where they all seem to be living these days. I will be thinking of you lot as I am sipping some Root Beer while cruising through Disney World in ideal weather conditions. I even hope to talk to some American TIers, perhaps savour a bit of new software or maybe even ruffle Gary Bowser's feathers a bit. See ya at the November meeting. o

Secretary's Notebook

by Terry Phillips

First an apology for not having a column in the September TND. I was battling winter bugs and missed the August meeting and consequently did not have a great deal to write about. Very few people, from my observations, have escaped the dreaded winter bugs, however now with spring well and truly here we can look forward to sunny warm days again.

The September meeting was a bit of a disappointment for a meeting billed as a full day tutorial. When I arrived at about 10.30am there were only a handful of members present and most seemed contented to sit around having a chat. As the day wore on not a great number more turned up and there did not appear to be any organised activities. I remember I wrote recently that Buy, Swap and Sell days appear to be poorly patronised and it now appears full day tutorials are on the wane. I may be only guessing but this could be as a result of no members needing much tutoring these days. Are we down to the 'hard-core' element of members who now only like to come along to meetings for the social side of things and discuss there computing problems on a one to one basis with someone they know is fairly well versed on that particular topic that they are having problems with? Anyway October is billed as another Buy, Swap and Sell day. We will see what eventuates on that day.

There being an insufficient roll-up of Directors we were unable to have a Directors meeting - Geoff was attending a wedding in Adelaide and Rolf had difficulty in arranging transport from Wollongong, while Russell was seconded to work overtime - Dick and I however, had an informal chat on future direction of the club. It seems that there will need to be fresh ideas and new blood next year if the club is to survive in its present structure and continue servicing the needs of members as it now does. Particularly paramount will be volunteers to produce the life-line of the club, the monthly Newsdigest. Give it some thought over the next couple of months.

At the time of preparing this article, the Faire is only a couple of months away and I am pleased to report that Ian Mullins and his band of helpers have things well under control. I wonder what the turn out will be. I suspect that there will be well over 100 attendees over the 2 days. Hope we sell all those T-Shirts!

I still want to push ahead with the social night preferably on the Saturday night of the Faire but at the present time there has been very little response as to what type of venue is preferred or as to how many might want to come along. I will need to firm up on both shortly so that a booking can be made.

We have 7 new members to welcome, most of whom have come through Percy and console exchanges. It is a big welcome to the following:

Lewis Griffiths	Old Bar
Terry Reeves	Croydon Park
Joanne Mizzi	Blacktown
Reg Yates	Port Augusta (SA)
Cheryl Turner	Warialda
George Norris	Springwood
Computech Computer Centre	Canberra

Hope you all enjoy your stay with the club.

Geoff Warner a member in Western Australia and a stalwart in that club has written to tell me that TIUP is alive and well. Geoff also asked me to pass on the following:

The Shop - service and products sold are wonderful.

TND - truly a great publication with production quality the best he has seen in the TI99/4A world. He has a minor criticism with the fragmenting of articles but

continued on page 6

TIsHUG Shop with Percy Harrison

The attendance at our full day tutorial was somewhat disappointing but it was very pleasing to see some of our more recent members who attended for the first time and I hope that they felt that it was worth their while coming along.

The Club T-shirts have arrived and went on sale at the September meeting. They are available in five sizes: 14 (90cm), 16 (95cm), 18 (100cm), 20 (105cm), and 22 (110cm). We have sold out of the 22 size so anyone wanting this size should let me know as soon as possible so that we can re-order. The price has been kept to a minimum of $11.00 each which means that we will have to sell every shirt to avoid making a loss. The shirts are white Bonds made in Australia and have a map of Australia on the front with the TIsHUG Logo within the map and the words COMPUTER CLUB underneath, all in black outline. On the back is a TI99/4A computer system comprising a console, PE Box and a monitor come TV set. The comments made by those members who attended the last meeting were very favourable.

Once again I would like to remind members that I will no longer bring the club hardware to the meetings so if you need drives, consoles, PC Boards or any of the cards that fit into the PE Box please ring me prior to the meeting and I will bring the items you want with me. Both Club Disks and Commercial Disks will be available and also boxes of blank disks and a range of IC's.

PRICE LIST

Club Software Disks

See Page 3 of the August TND plus following:

A475 Clubline 99 Vol 4 No.8 SSSD	$2.00
A476 Clubline 99 Vol 5 No.5 SSSD	$2.00
A494 XB #0 Money Money SSSD	$2.00
A495 Directory SSSD	$2.00
A504 The Director SSSD	$2.00
A505 Sorting DSSD	$2.00
A506 Memory Manager SSSD	$2.00
A507 Implanting SSSD	$2.00
A508 Booklet SSSD	$2.00
TC820 Health and the Human Body SSSD	$2.00
TC830 Physics SSSD	$2.00
TC850 Chemistry SSSD	$2.00
TC860 Astronomy Disk #1 SSSD	$2.00
TC890 Teacher's Helper SSSD	$2.00
TC990 Sports (Requires XB) SSSD	$2.00
TCC9 Tigercub Collection #9 SSSD	$2.00
TCC10 Tigercub Collection #10 SSSD	$2.00

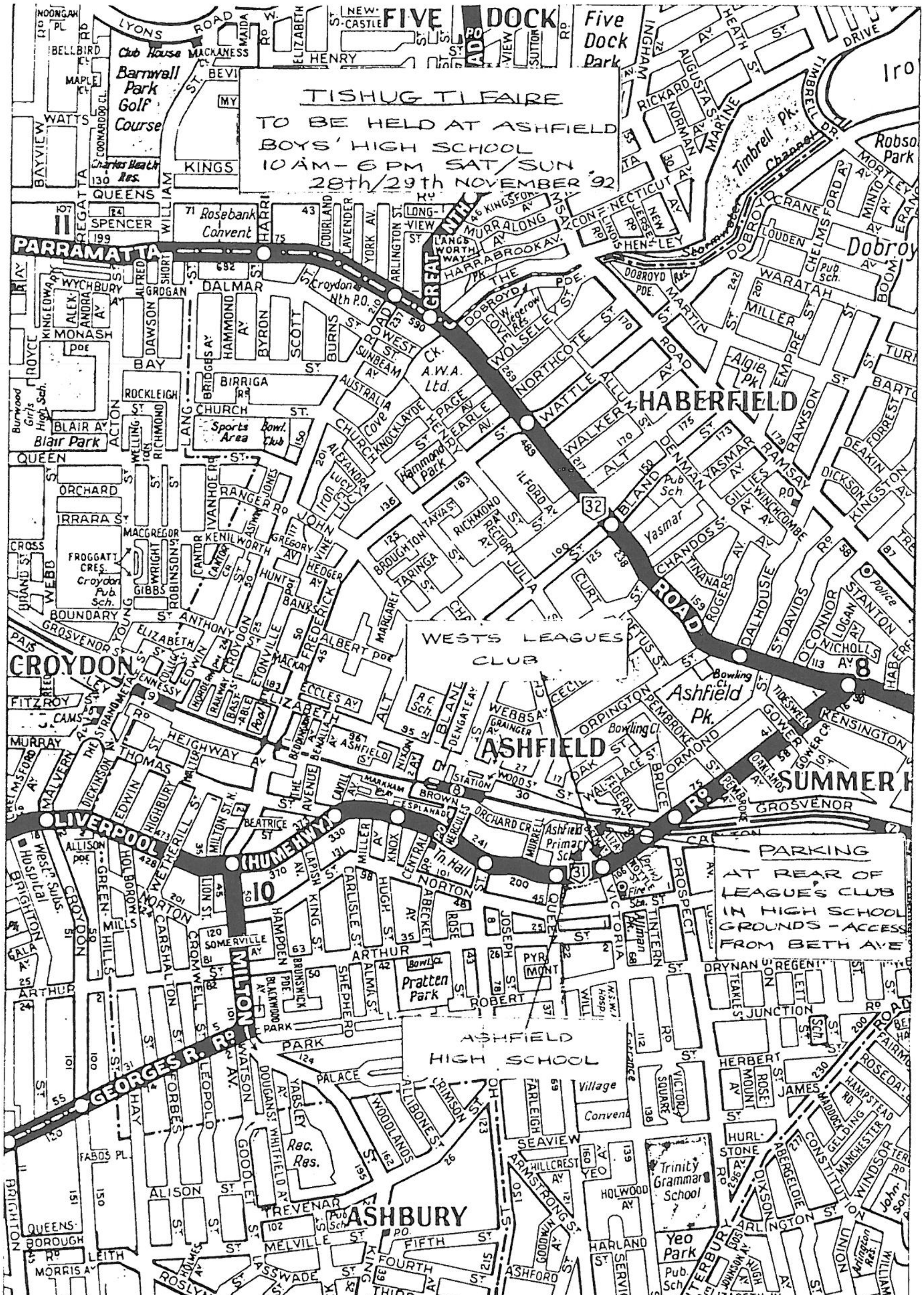
Hardware

AT Disk Control Card (DSDD Format)	$150.00
5.25 Half Height Drive Double Sided	$65.00
RS232 Card for PE Box	$100.00
Modem Card (300Bd) for PE Box	$60.00
Club T-Shirts Sizes 14,16,18,20,22	$11.00

Packaging and postage charges:

	Surface	Airmail
1 to 2 Disks	$1.90	1.90
3 to 9 Disks	$2.90	$3.60
10 to 20 Disks	$3.90	$4.80
TI Artist Plus	$3.00	$3.70
Display Master	$3.00	$3.70
TI Base	$3.00	$3.70
TI Sort	$3.00	$3.70
5.25 inch half-height drive (1.25 Kg)	refer to your local post office	

Bye for now. o



TISHUG TI FAIRE

TO BE HELD AT ASHFIELD BOYS' HIGH SCHOOL
10AM – 6PM SAT/SUN
28th/29th NOVEMBER '92

PARKING

AT REAR OF LEAGUE'S CLUB IN HIGH SCHOOL GROUNDS – ACCESS FROM BETH AVE

Shahzada Endurance Horse Ride

by Ross Mudie

Introduction

In August 1992, my trusty TI99/4A again saw service (for the fourth year) at the Shahzada Endurance Horse Ride as the event computer. The Shahzada is a 5 day, 400km event where horse and rider are pitted against the rugged terrain on the riding trails of the St Albans NSW area. The event commences at 4am daily and all riders must complete the 80km course set for each day by 5pm. Daily weather conditions recorded included temperatures between a frosty 4 degrees Celcius and a hot 23 degrees. At times hail was reported by the check point radio operators on the course, whilst strong winds and dust made conditions quite unpleasant. Some days presented four seasons in a day.

At the 1992 Shahzada, I set up two TI99/4A computers. The "main" computer managed the master data base, whilst the second computer was set up on a table in a horse float at the weighing scales. The main computer was in the back end of my communications caravan, in company with the WICEN (Wireless Institute Civil Emergency Network) amateur radio communications base station, the public address system, fax machine and my accomodation. The "Weights" computer in its horse float was immediately beside the "Ruddweigh" walk on scales for immediate input of the weight information.

The Hardware

The weights computer consisted of a TI99/4A console with in-built Extended Basic (for reliability) and inbuilt (custom wired) EPROM ROS Ramdisk/32K memory/ clock. The single stand alone disk drive was interfaced with an "original box car" disk controller and a "parallel output interface" between the disk controller expansion port and the logging printer. The screen used was a 14 inch colour TV receiver and TI modulator.

The main computer consisted of a standard TI99/4A console with Peripheral Expansion Box containing 32K memory expansion, DSDD AT disk controller, 2 disk drives, 991 sector EPROM ROS Ramdisk, RS232, Triple Tech (printer buffer and clock), Canon Bubble Jet printer (BJ10e), Wang monitor and TISHUG RGB interface.

Both systems were powered through a "Constant Voltage Transformer".

The Weights Computer

The Weights Computer was used each day to record the body weight of each horse and the combined weight of the rider with saddle and gear, etc. When each horse or rider approached the scales, the entrant number was called. Since the Ramdisk contained the NAMES data base, as soon as the entrant number was keyed in, the rider name was available on the screen. The computer operator could call out the name to ensure that the correct entrant number was being accessed. (Riders have been known to call the wrong entrant number and it is not hard to key in a wrong number when its busy).

As the rider or horse were weighed, the weights were typed straight into the computer. This overcomes problems with people's handwriting which cannot be read. The computer shows the weight history, so as the week went on each weight history grew. Many riders quickly realised the advantages available in being able to compare the current weight with the previous day's weights. To guard against loss of data in the event of a catastrophic computer failure, the printer logged every entry transaction. (Thankfully the printed log did not need to be used).

The names were transferred from the main computer by disk and the daily weight data was returned by disk. The only problem with transfer by disk was that there simply was not enough time between the names information being available for entry into the main computer and the horse being presented for weighing during the pre-ride session. Updating of the NAMES data base in the WEIGHTS computer needed to occur more regularly than was physically possible.

The weights program provided some automatic checks of the weight limit of each rider, ensuring that the appropriate class of entry was being maintained. Horse weights were monitored for not more than +/- 20kg change from pre-ride to day one and not more than +/- 10kg change between subsequent days. This provided an alert in the event of possibly excessive weight change in addition to prompting in the event of entry of a wildly invalid value.

The Main Computer

The main computer managed the event data base, which consists of files for names, veterinary data (weights and pulse, etc) in addition to the daily times and the entrants' status information. (Each entrant number has a status byte which can be used to indicate if (a) The entrant number is used, (b) If scratched, on which day, (c) If active in the event and (d) If the entrant is in one of the concurrently running single day events. (All that in an 8 bit byte occupying 274 bytes for the maximum number of 274 entrants). The use of the status bytes made sorting and printing routines much easier. There were a total of 135 entrants in this year's event.

The main computer allowed the Daily Event Data table to be printed out unchanged in entrant number order. The sorting routines allows printouts in riding time order for divisions of Open, Heavyweight, Lightweight, Junior or Single Day as well as all entrants. Other programs produce Veterinary certificates for the entrants which are successful and a list in order of riding time which provides the information needed to inscribe the banners.

Programming Languages

The programs described are written in TI Extended Basic with links to assembly, and the memory is pretty full, but the TI can do an amazing job.

How Did It All Perform and What About the Future?

A major failure on the 4th morning of the event occurred when a short drop out in the mains power voltage was coincident with a SAVE to the Ramdisk. Both computers were normally connected to a "Constant Voltage Transformer" which regulates the mains voltage. When the problem happened, data entry was occurring on the main computer as the weights computer was plugged in to commence the day's operations. At the moment of connection of the weights computer, the high current drawn by the de-gaussing circuit of the TV caused the mains power voltage to drop to a critically low level on the main computer. There was also a contact problem with the Ramdisk in the PE box which decided to make the system restoration a little more difficult. Three hours were lost that day in reformatting the Ramdisk, reloading back-ups from floppy disk and re-entering day 4 data that had not been backed up yet. Constant Voltage Transformers can regulate varying input voltages but they cannot handle large instantaneous loads changing on the output. Plans are being considered to run the computer off a battery back up system to prevent this type of problem in the future.

Plans are also being made to link the weights and main computer with modems, allowing a more efficient upload of new names data to the weights computer. I will be looking for a stand-alone or "box-car" RS232 with 2 RS232 ports for the weights computer, (can any one help with this?)

The weighing scales also have a RS232 output, it is hoped that next year the scales can be directly linked to the RS232 of the weights computer and the weights computer can directly upload to the main computer. (Well its something to aim for!)

continued on page 22

TI-Bits Number 20

by Jim Swedlow, CA USA

A VIRUS IN TI LAND? I DOUBT IT!

A computer virus attaches itself to a program and then hides. It moves from program to program, hidden until it starts doing whatever it does. Some virus programs display humorous and/or obscene messages. Others destroy data. By definition, a virus must exist in an electronic environment where it has programs to move among. This happens on a main frame or a hard disk. Since our TI's are mainly disk based, there should be no where for the virus to spread. A real danger, however, is a trojan horse. This little love is a program that is supposed to do one thing but actually does another - like reformat your disk or make the drive jump around.

Some simple precautions will keep you safe. When you get a new program, run it on a disk by itself. Keep all other drives empty. If anything unusual happens, shut your system off.

If you find a bad program and it came from a bulletin board, call the Sysop IMMEDIATELY. (Sysop is "bulletin boardese" for system operator.)

BOOT TRICK

Want to change the colours on your BOOT screen? This is not in the documentation, but pushing <B> changes the Background colour and <F> the Foreground. Use <FCTN 5> to save your changes.

TI vs IBM

No, this is not one of those bash the big blue monster items. Nor is it a lament for the end of the 4A. Rather, some thoughts on reality.

First, IBM compatible computers exist and are widely used. I do not want to get into the IBM v. MAC argument here. The point is that these computers are used and supported in offices and software stores. Our 4A is not. Another fact - many TI users have left the 4A and gone to other platforms. These defections have left those of us who continue to use our 4A's gun shy.

Some have suggested that TI user groups form IBM SIG's (Special Interest Groups). This has not been well received and I think wisely so. There is strong support for IBM compatibles. It would be a mistake for us to dilute our energies by diving them.

On the other hand, ignoring the real world would also be a mistake. Some of us use IBM compatibles at work. Others own one. This does not mean the end of the 4A. Our TI continues to be viable. It was not, it would have gone the way of Adam and other lesser contenders.

We cannot stick our heads in the sand and pretend that we are alone. We are not. We can, however, make sure that the TI continues to get the support it deserves. One way is to support those who live in a two computer world. From time to time, then, you will see some discussion here about working with a TI and another computer. I hope that this will be taken in the spirit that it intended. Not to wean folks away but to keep hands on those 48 keys.

MAGIC FILE MANIPULATOR

One vital necessity when you use more than one operating system is the ability to move files back and forth. If you have this need, MAGIC FILE MANIPULATOR (MAGICFM) is the program for you. You need the following:

- A TI system with disk, RS232 and 32K.
- Extended BASIC.
- A communications (modem) program that performs XMODEM file transfers for your other computer.
- A null modem cable.

The MAGICFM documentation includes pin outs for the null modem cable. Since it only uses three wires, it is easy to make.

You boot up MAGICFM on your TI and your communications program on your other computer. Then, from the other computer keyboard, you can:

- Catalog any disk on your TI.
- Delete, Protect, Unprotect or Rename any file on your TI.
- View any DV80 file.
- Transfer (via XMODEM) any file from your TI to the other computer OR from the other computer to your TI.

One of the things that make this program magic is that the file transfers work at the fantastic speed of 19,200 baud. Files fly over the null modem cable. If you think you need this program, you do. It solves any number of problems. Kudos to Ben Hatheway who wrote MAGICFM. Outstanding!

UNSUNG HEROES

The February, 1989 issue of PC Computing contained an article subtitled:

"Sometimes being great is not good enough. Witness the fate of ten products that deserved better".

Guess what one of the ten was.

"TI 99/4A: Texas Instruments' 16 bit home computer was fast, expandable, cheap and an early victim of 'dinosaur marketing.'"

EDITOR ASSEMBLER MODULE

Do you own the Editor Assembler module? Probably not. Most of us have Extended Basic but not Editor Assembler. With the price down to $10, it is a good buy. Many fine programs run with the Editor Assembler module including FUNNELWEB, ARCHIVER and DM1000. All of these programs will run under Extended Basic but they load much faster with Editor Assembler. Next time you see Editor Assembler on sale, pick up a copy. Take the book and put it on a shelf somewhere. Put the disks in your master box. You will not need either unless you start programming in assembly). You will wonder how you got along without Editor Assembler.

LOADING UTIL1 FILES IN EDITOR ASSEMBLER

When you insert the Editor Assembler module and "PRESS ANY KEY", you have two choices:

1 FOR TI BASIC
2 FOR EDITOR ASSEMBLER

Press <2> and you get FIVE choices. Only two are of use for loading programs:

3 Load and Run
5 Run Program File

"Load and Run" is for object code (Display Fixed 80) files while "Run Program File". is for Editor Assembler Program files.

If you press 5, Editor Assembler will ask you for a file name. If you just press <ENTER> without entering anything, Editor Assembler will look for and try and load a file named <UTIL1> in Drive 1 (close to the Extended Basic autoload of LOAD but you have to press two more keys). That is why FUNNELWEB has a UTIL1 and a

LOAD program - one for Extended Basic and one for Editor Assembler.

YOUR DISK WILL NOT WORK

Every now and then you get errors when you try to read from a disk. This can mean any of many things. Your drive might be dirty. You might have the wrong disk. The disk could be blown. The file could be lost. The list of horrors is long.

Here is a simple solution that you might try first. Remove the disk from the drive. Holding it by a corner, put your fingers in the center hole and turn the disk in its sleeve a few turns. Then move the media (the disk) around a little. Touch only the corner and the center. Sometimes this will restore your disk.

BAD DISKS

I purchased a box of disks at a discount house (low prices and no service!). One of the disks had bad sectors when I formatted it. I wondered what would happen, so I sent the disk back to the manufacturer asking for a replacement.

I was out about $1 (between postage and mailer) but it was worth the try. Over a month later another disk came in the mail along with a form letter explaining about quality control and such.

The only problem was that the new disk also had bad sectors. That one went into the trash and I will not buy that brand again! Who was it? I really should not say, but they are known for their film and there are not any K's in their name.

MORE DEBUGGING

I mentioned the story about the origin of "debugging" to a fellow in Canada. His response was a long and sad tale:

"Yes, I did know about 'debugging'. Fifteen years ago my company decided to do a lot of the alarms to our Central Station on a computer. This unit was six feet high. I knew NOTHING about computers except that, at that time, the office environment had to be kept within close bounds. I was chosen. They said, 'You are the most careful supervisor in Ontario'.

"I asked for air conditioning and an electronic air cleaner. 'Too expensive', they said. After again reading from the instructions given me, I showed them the pertinent info. 'H'mm', they said. Said I, 'This unit has seven filters that are to be changed each month'. Said they, 'Too expensive, just clean the old ones'.

"Said I, 'We must make grandfather disks in case we have problems'. 'Oh, we do not think so - that is just their way of making the whole project more expensive'.

Lastly, I told them that there was a week of training to be completed in Trenton and at least two of our people should attend. Said they, 'You are too valuable to lose for a whole week -- just pick out two of your people that you can afford to lose for that period'.

"I did and told them that the system would never work at all if this was how they were going to operate and that I disavowed any problem that reared it's head. $850,000 later they threw their hands in the air and told the computer people that their stupid computer was no good.

"Yes Jim, I did find bugs in the unit as well as bugs in my bosses at Head Office."

GRAPH

The other night my 13 year old just had to have graph paper for her home work. Never mind that she knew that she would need some paper for some time; she had forgotten to tell us and the need was urgent.

I remembered that I had an Extended Basic program that printed graph paper. To my surprise, I found it fairly quickly. I ran it and it worked - almost. It printed graph paper but the boxes were wider than they were long. It sufficed for her assignment but I had to fix the program.

Here it is:

```
100 ! GRAPH
110 E$=CHR$(27)
120 A$=RPT$(CHR$(128),228)
130 B$=RPT$(CHR$(255)&SEG$(A$,1,6),8)
140 B$=RPT$(B$&CHR$(255),4)
150 A$=E$&"K"&CHR$(228)&CHR$(0)&A$
160 B$=E$&"K"&CHR$(228)&CHR$(0)&B$
170 OPEN #1:"PIO.CR"
180 FOR I=1 TO 11
190 PRINT #1:E$;"@";E$;"3";CHR$(16)
200 FOR J=1 TO 8
210 PRINT #1:B$;B$;CHR$(10)
220 NEXT J
230 PRINT #1:A$;A$;E$;"3";CHR$(2)
240 NEXT I
250 PRINT #1:RPT$(CHR$(13)&CHR$(10),9)
260 PRINT #1:E$;"@"
270 CLOSE #1
```

This program will work with MOST Gemini and Epson compatible printers. There are two printer commands that can cause you problems.

The first one is <E$;"@"> (remember that E$ is defined in line 110 as CHR$(27) or Escape) which appears in lines 190 and 260. This is a reset command that tells your printer to restore its default settings. Earlier Epson compatibles (including the TI Impact Printer) do not recognize this command. If you get garbage with some "@" characters, yours does not either. These embedded reset commands cause your printer to completely loose any idea of where the top of form is so you will have to manually reset it.

The printer command that caused my problem is <E$;"3";CHR$(16)> in line 190. For the Gemini 10X and most Epson MX compatible printers, this sets line height to 16/144 (or .1111) inch. For the Star NX10, NX1000, and most Epson FX compatibles, it sets the line height to 16/216 (or .0741) inch. Hence the squat squares. I changed the line to read <E$;"3";CHR$(24)>. This set line height at 24/216 (or .1111) inch and everything worked correctly.

The difference for the <E$;"3";CHR$(2)> in line 230 is so small (.0046 inch) that it makes no difference. If you really wanted to, you could change it to <E$;"3";CHR$(3)>.

The problem with CHR$(27) "3" CHR$(n) raises its ugly head in a number of programs. Most folks wrote for the Gemini 10X/Epson MX family. This change, which came with the NX/FX lines of printers, has not received wide attention.

If your printout lines are too close together, look for this code and increase "n" by a factor of 1.5. You can, that is, if it is an Extended Basic program.

Enjoy. o

continued from page 2

accepts the difficulty maintaining continuity throughout the Newsdigest.

TI-Faire - extends his groups best wishes for its success but does not think that any will be able to attend due to distance and financial considerations.

That's all for this month. See you at the October meeting. o

TIPS to Instance Conversion

by Geoff Trott

Alf Ruggeri is trying to produce a demonstration for the TI-Faire of his Greeting Card production and would like to get some of the graphics available in TIPS in the form of instances. The TIPS program has this facility but in the process some of the resolution of the pictures gets lost. The single size instance is bigger than the TIPS printout for example and appears to lose some of its fine detail. Alf sent me a disk and some print-outs to demonstrate the problem. This seemed to be a problem of the mis-interpretation of the data and so I started to look into it. I was doing the club financial statement at the same time so I could not spend too much time on it.

If you look at the TIPS files for the data you will find that there are two files for each set of pictures. One file is reasonable small and its name ends with 'XXX'. The other one is large and its name ends with 'TXT'. They are both Internal Fixed type files, which means that their contents are stored in fixed length records and in the same format as you would find them in memory. The 'XXX' file has records of 16 bytes long while the 'TXT' file has a record length of 53 bytes. By looking at the contents of these files using DiskReview inspect command, I was able to determine that both these files store strings of characters. The 'XXX' file stores the picture name as a string of 15 characters (the string length byte takes the 16th byte) while the 'TXT' file stores large numbers of strings which are 52 characters long. Clearly, the 'TXT' file must contain the data of the picture and it is only necessary to find out the format that is used and we will be in business.

On closer inspection of the TIPS program and the data in the 'TXT' file, it became clear that each picture took up 11 of these 52 character strings. So for each of the 15 character name strings in the 'XXX' file there are 11, 52 character data strings which contain the data for that picture. The normal way to produce pictures on the screen or printer is to use a bit value of 1 for foreground or black dot and a bit value of 0 for background or no black dot. When a printer is producing the picture, it runs across the page printing 8 bits at a time so that each character sent to the printer runs its bits down the page with the most significant bit at the top and the least significant bit at the bottom. I tried interpreting the data using that scheme first and got the following for this data (ignoring the strings of all null codes and putting each character code on two lines to allow it all to fit on a single line). The data is for 5 strings of 52 characters each with the ASCII code of each character shown in two hexadecimal digits, one above the other.

```
0000000000000000000000000000000000000000000000000000
0000000001344443000000000000000000000134444300000000000
00000007FFF0002C00000000000011000 7FFF0002C0000000000
0000000FFFF0000000003788517F33E6EFFFF000000000000000
0000000FFFF111322666CCC888FF0F000FFFF000000000000000
0000000EFFFFFFFFF777333111FF0F000EC80000000000000000
000000008CCCCCCCEEEFFFFFFFFFF771113700000000000000000
0000000000000000000000888CCCEEEFFFFE000000000000000
000000000000000000000000000000000C800000000000000000
0000000000000000000000000000000000000000000000000000
```

This shows a bit of the pattern this way. If we expand out the hexadecimal into binary (leaving out the rows of zeros to save a bit of space) we get the following pattern.

If this has not gone over too many pages, it should be recognisable as a rather stylish 'A'. It is obviously sideways but not quite as obvious, it is actually also back to front. That is the bottom of the picture above is actually the right hand side of the picture and the top is the left hand side of the picture. This caused me to stop and reflect how

```
00000000000111100000000000000000000000001111000000000000
00000000001000010000000000000000000000010000100000000000
00000000011000010000000000000000000000110000100000000000
00000000111000010000000000000000000001110000100000000000
00000001111000010000000000000000000011110000100000000000
00000001111000100000000000000000000011110001000000000000
00000001111000000000000000000011000111100000000000000000
00000001111000000000000110001001011111000000000000000000
00000001111000000000001001011001111111000000000000000000
00000001111000000000011000011111111111000000000000000000
00000001111000000000011001111110001111000000000000000000
00000001111000000000011111111010001111000000000000000000
00000001111000000111111000110100011110000000000000000000
00000001111000111111000000110100011110000000000000000000
00000001111111100000000000110100011110000000000000000000
00000001111111111000000000110100011100000000000000000000
00000001111111111111000000110100011000000000000000000000
00000001111111111111110001101000100000000000000000000000
00000000111111111111111111010000000000000000000000000000
00000000011111111111111111100000000000000000000000000000
00000000001111111111111111111000010000000000000000000000
00000000000000001111111111111110001100000000000000000000
00000000000000000011111111111111111000000000000000000000
00000000000000000000001111111111111000000000000000000000
00000000000000000000000011111111111000000000000000000000
00000000000000000000000000001111111000000000000000000000
00000000000000000000000000000001111000000000000000000000
00000000000000000000000000000000011000000000000000000000
00000000000000000000000000000000010000000000000000000000
```

difficult it may be to rotate this and reverse it to get it to come out correctly on the printer. Fortunately the printing of the TIPS pictures is done well so this is not a problem. Then I thought I had better find out how the instance file requires the data to be organised. I did look at an instance file using DiskReview (these are display variable 80 files) and found that they contain one record which gives the size of the instance (in characters) which is then followed by the character definition data for each character as 8 decimal integers in a record separated by commas. The character definition sequence is ordered starting at the top left of the picture and scanning the picture from left to right and top to bottom. This is described in the Appendices to TI-Artist Plus. So I needed to first determine the size of the instance in characters to encompass the TIPS picture and this is 11 characters wide and 6.5 characters high (11 records of 52 characters rotated and reversed). To make sure that nothing is missed I decided that I would use instances of 11 characters wide and 7 characters high so the first record in the instance file would contain '11,7' and would be followed by 77 records, each containing 8 decimal numbers between 0 and 255 separated by commas to represent the character definition data.

Now to do the translating of the data, you must recognise that character definition is done with the pattern running horizontally for each character, 8 bits across each character, then working down the character for 8 bits. If you do not understand what I mean, I do not blame you but look in the Extended BASIC or TI-BASIC manual at the explanation of the 'CALL CHAR' subroutine. It turns out that if the data in the TIPS file is read straight into a character definition one record at a time for 7 characters whose ASCII codes are consecutive and then these are placed vertically as columns for the picture, the picture comes out the right way up and reversed. This was quite unexpectedly easier than I thought it would be and only required a very small assembler routine to which I pass the string which comes from reading one record of the 'TXT' file with 4 nulls added on the end to give 56 characters. These characters are then stored in the pattern definition table in VDP memory for the character which is also passed to the assembler routine and the following six characters (that is 56 consecutive memory locations in VDP memory). This could be done with BASIC but would involve taking each character code, getting its value (using ASC) and converting it to two hexadecimal characters and then assembling these characters into a string for input into the CHAR routine. This is not easy to do in BASIC but is very easy and quick to do in assembler.

To display the picture on the screen, it is necessary to print to the screen a rectangle of characters 11 wide and 7 high with the character codes running vertically. The first such pattern I used was the following:

```
!(/6=DKRY‘g
")07>ELSZah
#*18?FMT[bi
$+29@GNU\cj
%,3:AHOV]dk
&-4;BIPW^el
'.5<CJQX_fm
```

This worked well and I then re-defined the characters to show the picture, one column at a time. The characters were defined as a string one row at a time and printed out that way, across the screen. The character definition routine is passed the first eleven characters of the string (the top character of each column, one at a time) along with the record from the 'TXT' file. To output the data to the instance file requires output of the pattern of each character in the order of the characters in the string above. This could also be done in BASIC but would be relatively cumbersome and slow. A short assembler routine has as input the character to be worked on and as output a string. The routine reads the pattern out of VDP memory and converts the 8 bytes into decimal numbers and stores these digits in the text string separated by commas. I then modified the BASIC part of the TIPSSHOW program to do the file handling and allow the pictures in a TIPS file to be viewed on the screen and sent to an instance file if wanted.

The listing of the Extended BASIC program follows. You will note that the character string is somewhat different to the one above in an attempt to not re-define characters which may be displaying a useful message on the screen. The concept is the same however as that mentioned above. I did find one interesting point in that the character whose code is >7F was not able to be printed on the screen (using DISPLAY AT).

```
100 ! SAVE DSK1.TIPSINSTA
110 REM (C) 1992 Geoff Trott TIPSINSTA 1.0
120 DATA GDAZ
130 DATA ZZZ
140 FLZ$="!(:U\cjqx"&CHR$(130)&CHR$(137)&""");V]dkry"&
    CHR$(131)&CHR$(138)&"#*<W^elsz"&CHR$(132)&
    CHR$(139)&"$+=X_fmt{"&CHR$(133)&CHR$(140)
141 FLZ$=FLZ$&"%,>Y‘gnu¦"&CHR$(134)&CHR$(141)&
    "&-?Zahov}"&CHR$(135)&CHR$(142)&"'.@[bipw~"&
    CHR$(136)&CHR$(143)
170 CALL INIT :: CALL LOAD("DSK1.CHRPAT;O")
180 INPUT "Place image disk in drive 1 ":IGN$
190 READ FIL$ :: IF FIL$="ZZZ" THEN END
200 IF FIL$<>"GDAZ" THEN GOTO 220
210 CALL CLEAR :: CALL CHARSET :: INPUT "Enter first 4
    characters of TIPS filename ":FIL$
220 OPEN #1:"DSK1."&FIL$&"XXX",INPUT
    ,SEQUENTIAL,INTERNAL,FIXED 16
230 CALL CLEAR :: FOR I=1 TO 7 :: DISPLAY
    AT(I+4,9):SEG$(FLZ$,(I-1)+1,11):: NEXT I
240 OPEN #2:"DSK1."&FIL$&"TXT",INPUT ,INTERNAL,FIXED 53
270 RNO=0 :: DISPLAY AT(16,1):"PRESS ENTER FOR NEXT ONE"
    :: DISPLAY AT(17,7):"1 FOR INSTANCE" :: DISPLAY
    AT(18,7):"0 FOR FINISHED"
280 IF EOF(1)THEN GOTO 300
290 INPUT #1:INAM$ :: DISPLAY AT(14,2):INAM$ :: GOSUB
    320 :: RNO=RNO+1
295 GOTO 280
300 CLOSE #1 :: CLOSE #2
310 GOTO 190
320 FOR I=1 TO 11
330 INPUT #2:I$ :: I$=I$&RPT$(CHR$(0),4)::
    J$=SEG$(FLZ$,I,1):: CALL LINK("CHRPAT",J$,I$)
340 NEXT I
350 ACCEPT AT(23,27):IGN$ :: IF IGN$="0" THEN END ELSE
    IF IGN$="1" THEN GOSUB 370
360 RETURN
370 OPEN #3:"DSK1."&FIL$&STR$(RNO)&"_I",OUTPUT :: PRINT
    #3:"11,7"
380 FOR I=1 TO 77 ::
    CALL LINK("INSTAN",SEG$(FLZ$,I,1),A$):: PRINT #3:A$
    :: NEXT I
390 CLOSE #3 :: RETURN
```

The assembler routines are both in one file and a listing of them follows. They are just for Extended BASIC but could be changed to work in any environment by changing the references to the routines called by these subroutines.

```
* An Extended BASIC subroutine to put the character
* pattern into the specified characters for graphics.
* These will be done 8 characters at a time.
*
* The call will be:
*   CALL LINK("CHRPAT",C$,I$)
* where
*      C$ contains the first character to be defined
*      I$ contains 56 characters with the patterns
*       for 7 charcters.
*
* An Extended BASIC subroutine to put character
* definitions into a form for TIA+ instance files.
*
* The call will be:
*   CALL LINK("INSTAN",C$,OUT$)
* where
*      C$ contains the character whose definition is
*         to be done
*      OUT$ is the string to hold the 8 numbers
*
       DEF  CHRPAT,INSTAN
*
* Equivalences for EXTENDED BASIC routines.
*
NUMREF EQU  >200C
STRASG EQU  >2010
STRREF EQU  >2014
VSBR   EQU  >2028
VMBR   EQU  >202C
VMBW   EQU  >2024
*
FAC    EQU  >834A
STATUS EQU  >837C
*
* Start of subroutine.  Need to use new workspace
* registers, so use a context switch here!
*
CHRPAT BLWP @PROGC
       CLR  R0
       MOVB R0,@STATUS   set status no errors
       RT
*
* Start of routine with new WS
*
PROGC  DATA MYWS,START
*
START  CLR  R0
       LI   R1,1         1st argument
       LI   R2,SBUF1     buffer for string
       BLWP @STRREF      get string
       CLR  R0
       LI   R1,2         2nd argument
       LI   R2,SBUF2     buffer for string
       BLWP @STRREF      get string
       MOV  @SBUF1,R0
       ANDI R0,>7F            get character
       AI   R0,>60            add offset
       SLA  R0,3              get address
       LI   R1,SBUF2+1        data in RAM
       LI   R2,56             number of bytes
       BLWP @VMBW             tranfer bytes
       RTWP
*
* Start of subroutine.  Need to use new workspace
* registers, so use a context switch here!
*
INSTAN BLWP @PROGI
       CLR  R0
       MOVB R0,@STATUS   set status no errors
       RT
*
* Start of routine with new WS
*
PROGI  DATA MYWS,STRT
*
STRT   CLR  R0
       LI   R1,1
       LI   R2,SBUF1          get first parameter
```

continued on page 14

Assembly Class

by Ross Mudie

The Assembly Class continued on 5th September with the members participating in the writing of a program in assembly linked from Extended Basic. The program takes two time values in a single TI Extended Basic string format, performs limited validation of the numeric content and then subtracts the start time from the finish time. As the program was developed, there was quite a lot of spirited discussion on how to do the required tasks. In line program or sub-routine structure was evaluated for the amount of memory used.

At one stage an XMLLNK routine was encountered which no one was really confident on how it was used, so a test program was set up to prove if it could be successfully used. (It worked!)

The class ran out of time to complete the task, so the discussion will again be continued at the October class at 10am, 3rd October 1992.

This class will explore converting the resultant minutes into hours and minutes separated by a "." and sending the result back to Extended Basic in the input string. Quite a lot of tidying up will be required to make the program memory efficient. All TISHUG members are very welcome to attend the Assembly Language Class, just remember to bring your Editor/Assembler manual, note book, pencil and a blank (formatted) SSSD disk if you want a copy of the day's work.

A copy of the work of the class on 5th September follows for anyone who wants to join in.

```
100 ! SAVE DSK1.LOAD2
110 CALL CLEAR
120 CALL INIT
130 CALL LOAD("DSK1.O")
135 CALL CLEAR
140 S$="0442 1008"
150 DISPLAY AT(8,1):S$
160 CALL LINK("TIME",S$,T)
170 DISPLAY AT(10,1):S$,T
1000 CALL PEEK(-31926,A,B,C,D,E,F) :: PRINT A;B;C;D;E;F

* S=S  O=O

* TISHUG ASSEMBLY CLASS 5/9/92

       IDT  'TIMETEST'

       DEF  TIME

SAVRTN BSS  2
WS     BSS  32

BUFFER BSS  16


VSBW   EQU  >2020
VMBW   EQU  >2024
NUMASG EQU  >2008
STRREF EQU  >2014
XMLLNK EQU  >2018

FAC    EQU  >834A

VDPBUF DATA >0800  Used for CSN
SIXTY  DATA 60

CO     TEXT '0'
C9     TEXT '9'
B15    BYTE 15

       EVEN

*  CALL LINK("TEST",S$)

SUB1
       LI   R3,4
LOOP1  CB   *R4,@CO
       JLT  JMPEND
       CB   *R4+,@C9
       JGT  JMPEND

       DEC  R3
       JGT  LOOP1
       B    *R11

JMPEND B    @END

*  24 BYTES
*--------------------------

TIME   MOV  R11,@SAVRTN
       LWPI WS

       CLR  R0            Element
       LI   R1,1          Argument
       LI   R2,BUFFER
       MOVB @B15,@BUFFER
       BLWP @STRREF

*------------------------
*       LI   R5,2
*       LI   R4,BUFFER+1
*LOOP2  LI   R3,4
*LOOP1  CB   *R4,@CO
*       JLT  END
*       CB   *R4+,@C9
*       JGT  END

*       DEC  R3
*       JGT  LOOP1

*       INC  R4
*       DEC  R5
*       JGT  LOOP2

*  38 BYTES
*------------------------

       LI   R4,BUFFER+1
       BL   @SUB1

     INC  R4  R4 is pointing at BUFFER+6
       BL   @SUB1

*  14 BYTES

*------------------------

*      LI   R4,BUFFER+21
*      BL   @SUB1
*
*    INC  R4  R4 is pointing at BUFFER+6
*      BL   @SUB1

*  14 BYTES

*------
       LI   R0,>0800
       LI   R1,BUFFER+1
       LI   R2,2
       BLWP @VMBW

       LI   R0,>0802
       LI   R1,>2000
       BLWP @VSBW

       MOV  @VDPBUF,@FAC+12

       BLWP @XMLLNK
       DATA >11AE       CSN

       BLWP @XMLLNK
       DATA >12B8       CFI

       MOV  @FAC,R6   Hours start in R6
```

```
* MINUTES

       LI    R0,>0800
       LI    R1,BUFFER+3
       LI    R2,2
       BLWP @VMBW

       LI    R0,>0802
       LI    R1,>2000
       BLWP @VSBW

       MOV  @VDPBUF,@FAC+12

       BLWP @XMLLNK
       DATA >11AE          CSN

       BLWP @XMLLNK
       DATA >12B8          CFI

       MOV  @FAC,R8  Minutes start in R8

       MPY  @SIXTY,R6   Answer in R7
       A    R7,R8     Mins Start in R8
*------

*------
       LI    R0,>0800
       LI    R1,BUFFER+6
       LI    R2,2
       BLWP @VMBW

       LI    R0,>0802
       LI    R1,>2000
       BLWP @VSBW

       MOV  @VDPBUF,@FAC+12

       BLWP @XMLLNK
       DATA >11AE          CSN

       BLWP @XMLLNK
       DATA >12B8          CFI

       MOV  @FAC,R3   Hours FINISH in R3


* MINUTES

       LI    R0,>0800
       LI    R1,BUFFER+8
       LI    R2,2
       BLWP @VMBW

       LI    R0,>0802
       LI    R1,>2000
       BLWP @VSBW

       MOV  @VDPBUF,@FAC+12

       BLWP @XMLLNK
       DATA >11AE          CSN

       BLWP @XMLLNK
       DATA >12B8          CFI

       MOV  @FAC,R5 Minutes finish in R5

       MPY  @SIXTY,R3   Answer in R4
       A    R4,R5     Mins finish in R5
*------
       S     R8,R5

       MOV  R5,@FAC
       BLWP @XMLLNK
       DATA >20            CIF


       CLR  R0             ***
       LI   R1,2           ***
       BLWP @NUMASG        ***
```

continued on page 20

Scott Foresman Reading Series

reviewed by Charles Good, Lima, OH USA

These modules resemble PLATO software in that they present specific language arts concepts in a text format and then ask a series of questions to text the student's knowledge of the concept. Unlike PLATO software, the Scott Foresman modules make good use of music and colour graphics. The rare modules DO NOT make use of speech synthesis, unlike some of the more common cartridges in the Scott Foresman READING series. These cartridges seem to be designed for in classroom use, which may be why they were not made commonly available to the public. The "suitable age" designations below are taken from the Fall 1987 TRITON catalog which lists most of these modules.

READING TRAIL
Suitable for ages 8-12, this cartridge teaches about the characters, setting, and points of view in stories. Famous characters from the Wizard of Oz and a separate story about fishing are used to illustrate specific points.

READING POWER
Suitable for ages 8-12, this module teaches research skills involving the dictionary, encyclopedia, and library card catalog. Specifically the student learns how to find information that is organized alphabetically in these kinds of reference materials. A detective story called "The Lion's Charm" with colour animation and music is used in some of these activities.

READING RAINBOWS
This is one of the rarer of the "rare" READING modules. It has been listed in very few catalogs over the years. It teaches how things are alike, parts and wholes, and sizes. Speech synthesis is used effectively. My first grade daughter whipped through this in a very short time, so I assume it is designed for first grade (age 6).

READING WONDERS
Another of the more rare modules, READING WONDERS teaches the student to distinguish between various types of fiction and non fiction. Several colourful stories are used to illustrate what is and is not historical fiction, modern realistic fiction, science fiction, biography, autobiography, and information articles. I suspect that this is probably for ages 11-13

READING ADVENTURES
This uses a variety of stories to teach, within a paragraph, main and supporting details, drawing conclusions, and sequential relationships. I have seen this one mentioned, but not described in catalogs. It looks like about ages 8-10, but I am not sure.

READING CHEERS
I would have guessed this was for 2nd grade, but my 1987 TRITON catalog says ages 8-12 (2nd grade is age 7-8). The module teaches root words with endings (lazy and lazily), contractions, and compound words.

All of the above "rare" modules are c1983. To complete the record I will briefly describe below the more commonly available 1982 Scott Foresman titles in the READING series.

READING ON
Some nicely illustrated science fiction stories illustrate the use of maps, schedules, graphs, and why and how people use them. For ages 8-9

READING FUN
There is minimal use of speech synthesis in this 2nd grade level module. Four colourful stories illustrate problems and how people solve them, why things happen, and how characters feel.

READING ROUND UP
Four stories based on an "American Wild West" theme

continued on page 18

Crazy Extended BASIC

by Wesley R. Richardson, OH USA

The purpose of this article is to describe how an Extended BASIC (XB) program is stored on disk and how a program can have line numbers out of sequence, or even have hidden lines, yet still run properly. The intent is inform programmers so they can attempt to restore programs which have been altered.

The program CRAZY-XB1 is a very simple program which prints 'LINE 40' 'LINE 50' and so on to the screen. The listing for CRAZY-XB2 shows how the program can be altered to have descending line numbers. Note that line number 7 is for two different instructions. The listing for CRAZY-XB3 would appear that only line 10 is in the program, yet when CRAZY-XB3 is run, it will function exactly like -XB1 and -XB2. These listings are how the program would appear after you typed LIST.

```
 10 REM CRAZY-XB1
 20 REM WESLEY R. RICHARDSON, FEB 1990
 30 REM NORTHCOAST 99ER'S, CLEVELAND, OH
 40 PRINT "LINE 40"
 50 PRINT "LINE 50"
 60 PRINT "LINE 60"
 70 PRINT "LINE 70"
 80 PRINT "LINE 80"
 90 PRINT "LINE 90"
100 END

        ----------

 10 REM CRAZY-XB2
 20 REM WESLEY R. RICHARDSON, FEB 1990
 30 REM NORTHCOAST 99ER'S, CLEVELAND, OH
  9 PRINT "LINE 40"
  8 PRINT "LINE 50"
  7 PRINT "LINE 60"
  7 PRINT "LINE 70"
  6 PRINT "LINE 80"
  5 PRINT "LINE 90"
100 END
        ---------

 10 REM CRAZY-XB3

        -----------
```

To understand how these programs work, we must first look at the Extended BASIC representation for the program. If you refer to the CRAZY-XB1 ASCII code sector listing (below), you will see that the lines are listed in reverse order. The disk sector listing has line 90, then 80 and so on, ending with the CRAZY-XB1 statement.

Note that if you edit a line or add a line, then that line gets moved to the beginning of the file. If line 40 is edited, then it will be in the file (and located in memory) before line 90. The line number table in memory enables the computer to run the program in the correct sequence, even if the processor has to expend some time chasing around all over memory to do so!

If you edit a program and simply save the program, the lines as listed on the screen will be in proper order, but internally they will be quite mixed. If you have a program, for example PROGNAME1, in which you have made several changes, the lines can be re-ordered by the following steps:

```
1) SAVE "DSK1.PROGNAME2",MERGE
2) NEW
3) MERGE "DSK1.PROGNAME2"
4) SAVE "DSK1.PROGNAME3"
```

I suggest using different filenames in case you make an error, then you can recover using the original file. When creating a program, do all of you debugging and modifications and when your program is finished, then use the MERGE routine to organize the internal program lines.

Now that we understand the BASIC lines can be out of order in the file, how do we modify the line numbers? If you refer to the CRAZY-XB1 hex code sector listing, we will see how XB keeps track of the line numbers. In the first row, locate the 0064, that is line 100. Also in the first row is 005A (hex 50=decimal 80+ hex A=decimal 10), that is line 90. We can see the old line numbers in hexadecimal and decimal.

OLD LINE #		NEW LINE #	
HEX	DEC	HEX	DEC
0064 =	100		
005A =	90	0005 =	5
0050 =	80	0006 =	6
0046 =	70	0007 =	7
003C =	60	0007 =	7
0032 =	50	0008 =	8
0028 =	40	0009 =	9
001E =	30		
0014 =	20		
000A =	10		

Using a sector editor, I changed the old line number hex values to those indicated under new. If you examine rows one, two and three in the CRAZY-XB2 hex code sector listing, you will see these changes. But wait, how can the program still work? Extended BASIC executes instructions according to memory location, not to line numbers (the actual line numbers found in the line number table are not important in running a program, their place in the table is important). When we list the CRAZY-XB2 program, it appears on the screen as I listed it previously. If you try to edit the program by typing 10 then FCTN X, you will be able to see lines 10, 20 and 30, but when you go to line 9, the old line 40, XB will tell you "LINE NOT FOUND." The program will still run correctly.

If we make one more change, we can hide some lines. By changing the sector row one value of 0064 for line 100 to a value like 0001, you will produce CRAZY-XB3. Now only line 10 can be viewed when listed, but the program still works fine.

Line numbers in Extended Basic range from 1 to 32767, or hex 0001 to 7FFF. If we change the line number to a value in the range of 8000 to FFFF, it will cause a BREAK in the program when that line is executed. For example, if the program reached the line number 83E8, the line number would then have the value of 8000 subtracted, leaving 03E8 and the message "BREAKPOINT IN 1000" would be displayed.

In the hex code sector listing for CRAZY-XB1, in lines 1 to 3, there are 2 byte or four digit numbers such as 373B, 373E, 374A, and 3756, after each line number. These refer to the memory location for the Extended Basic instruction. The difference between adjacent values is the number of bytes used for the Extended Basic instruction. The format for each instruction is XXYYYY..YYY00. The XX is the number of bytes used for the instruction, not including the 00.

Since the maximum value which can be represented is FF, the longest line length in XB is 255 bytes. Depending upon the statements which you use, this 255 byte length can have different ASCII lengths which you see when entering an Extended Basic program. The Extended Basic statements are stored in token format, for example PRINT is 9CC7. The word PRINT takes 5 ASCII bytes, but to Extended Basic, only requires 2 bytes to store 9CC7.

Some information such as the text contained in print statements is in the same format when saved to disk. For example the characters LINE 50 are stored on disk in the readable form as shown in line 7 of the ASCII code sector listing for CRAZY-XB1.

The third format which Extended Basic uses on disk for program files is for CALL statements. Memory must be reserved for variables and CALL statements. One way to find the tokens for each of the XB commands is to write a program using each of the commands on a separate line, and then look at the hex codes using a sector editor. Be sure to use the MERGE technique listed above if you wish to keep the sequence of lines in order when the program is saved to disk.

As I indicated earlier, I do not agree with using hidden instructions in Extended Basic programs. If you encounter one of the modified programs, perhaps now you will have some idea about how they were modified and the meaning of the values of an Extended Basic program stored on disk.

CRAZY-XB1 - ASCII CODE SECTOR LISTING

```
=======================================
 I N  E    7 0  . .  . .  . L  I N  E
 6 0  . .  . .  . L  I N  E    5 0  . .
 R T  H C  O A  S T    9  9 E  R '  S ,
   C  L E  V E  L A  N D  ,    O H  . !
 A R  D S  O N  ,    F E  B    1 9  9 0
 B 1     .  . .  . .  . .  . .  . .  . .
```

CRAZY-XB1 - HEX CODE SECTOR LISTING

```
=======================================
 002B 3739 3712 37D7 0064 373B 005A 373E
 0050 374A 0046 3756 003C 3762 0032 376E
 0028 377A 001E 3786 0014 37AA 000A 37CC
 028B 000B 9CC7 074C 494E 4520 3930 000B
 9CC7 074C 494E 4520 3830 000B 9CC7 074C
 494E 4520 3730 000B 9CC7 074C 494E 4520
 3630 000B 9CC7 074C 494E 4520 3530 000B
 9CC7 074C 494E 4520 3430 0023 9A20 4E4F
 5254 4843 4F41 5354 2039 3945 5227 532C
 2043 4C45 5645 4C41 4E44 2C20 4F48 0021
 9A20 5745 534C 4559 2052 2E20 5249 4348
 4152 4453 4F4E 2C20 4645 4220 3139 3930
 000C 9A20 4352 415A 592D 5842 3100 AA3F
 FF11 0300 0000 0600 01C3 5241 5A59 2D58
 4231 2000 0000 0000 0100 0000 0000 0000
 0000 0000 0028 0000 0000 0000 0000 0000
```

CRAZY-XB2 - ASCII CODE SECTOR LISTING

```
=======================================
 I N  E    7 0  . .  . .  . L  I N  E
 6 0  . .  . .  . L  I N  E    5 0  . .
 R T  H C  O A  S T    9  9 E  R '  S ,
   C  L E  V E  L A  N D  ,    O H  . !
 A R  D S  O N  ,    F E  B    1 9  9 0
 B 2     .  . .  . .  . .  . .  . .  . .
```

CRAZY-XB2 - HEX CODE SECTOR LISTING

```
=======================================
 002B 3739 3712 37D7 0064 373B 0005 373E
 0006 374A 0007 3756 0007 3762 0008 376E
 0009 377A 001E 3786 0014 37AA 000A 37CC
 028B 000B 9CC7 074C 494E 4520 3930 000B
 9CC7 074C 494E 4520 3830 000B 9CC7 074C
 494E 4520 3730 000B 9CC7 074C 494E 4520
 3630 000B 9CC7 074C 494E 4520 3530 000B
 9CC7 074C 494E 4520 3430 0023 9A20 4E4F
 5254 4843 4F41 5354 2039 3945 5227 532C
 2043 4C45 5645 4C41 4E44 2C20 4F48 0021
 9A20 5745 534C 4559 2052 2E20 5249 4348
 4152 4453 4F4E 2C20 4645 4220 3139 3930
 000C 9A20 4352 415A 592D 5842 3200 AA3F
 FF11 0300 0000 0600 01C3 5241 5A59 2D58
 4232 2000 0000 0000 0100 0000 0000 0000
 0000 0000 0028 0000 0000 0000 0000 0000
```

CRAZY-XB3 - ASCII CODE SECTOR LISTING

```
=======================================
 I N  E    7 0  . .  . .  . L  I N  E
 6 0  . .  . .  . L  I N  E    5 0  . .
 R T  H C  O A  S T    9  9 E  R '  S ,
   C  L E  V E  L A  N D  ,    O H  . !
 A R  D S  O N  ,    F E  B    1 9  9 0
 B 3     .  . .  . .  . .  . .  . .  . .
```

CRAZY-XB3 - HEX CODE SECTOR LISTING

```
=======================================
 002B 3739 3712 37D7 0001 373B 0005 373E
 0006 374A 0007 3756 0007 3762 0008 376E
 0009 377A 001E 3786 0014 37AA 000A 37CC
 028B 000B 9CC7 074C 494E 4520 3930 000B
 9CC7 074C 494E 4520 3830 000B 9CC7 074C
 494E 4520 3730 000B 9CC7 074C 494E 4520
 3630 000B 9CC7 074C 494E 4520 3530 000B
 9CC7 074C 494E 4520 3430 0023 9A20 4E4F
 5254 4843 4F41 5354 2039 3945 5227 532C
 2043 4C45 5645 4C41 4E44 2C20 4F48 0021
 9A20 5745 534C 4559 2052 2E20 5249 4348
 4152 4453 4F4E 2C20 4645 4220 3139 3930
 000C 9A20 4352 415A 592D 5842 3300 AA3F
 FF11 0300 0000 0600 01C3 5241 5A59 2D58
 4233 2000 0000 0000 0100 0000 0000 0000
 0000 0000 0028 0000 0000 0000 0000 0000
```

o

I Wish ...

by Jim Peterson, Tigercub Software, USA

I wish that someone would write a rapid disk copier, like Rediskit, that would allow me to specify the drive I wanted to copy to each time, so that I could be able to load a disk into one drive while the disk in another drive was being written to.

Even better, I wish someone would write an even faster disk copier that would read in as much as possible and then write it to two or more drives.

I wish that someone would figure out how to put four more tone generators under the hood of the TI-99/4A, with software to access them.

I wish that someone would write a LINK to assembly to store strings in the expansion memory; the limitation to console memory in Extended Basic is one of the worst, although least-known, faults of the TI.

I wish that someone would write a library of links to assembly, to do the many things that Extended Basic cannot do or cannot do fast enough.

I wish that the anonymous genius who created the Ernie and Bert program would share his secret with us. He seems to have achieved better sound quality, in far less memory, than Sound F/X.

I wish that someone would remap the keyboard to simulate the Dvorak typewriter.

I wish that someone would write a tutorial on programming in assembly in very simple language that I can understand, using three-letter words to replace such intimidating terms as "least significant byte" and "floating point accumulator".

I wish that someone would write a music composing program. A person with a good knowledge of both music theory and programming should be able to do so, because music is basically mathematical in concept - sounds must vibrate a certain number of times per second in order to be recognized as musical tones, and collections of those tones must be arranged within certain parameters, definable by music theory, in order to sound musical. It should be possible to randomly produce phrases within those parameters and allow the user to select a phrase to be further randomly developed, until a complete melody emerges.

I wish that someone would write a really complete tutorial article on the various types of disk files and the means of accessing them.

If those folks who like to brag about the lightning speed of their low level languages are actually writing programs in those languages, I wish they would share them with us.

I wish that my new Star NX1020R printer had, among its many character sets, those handy graphics characters that were accessible in ASCII 225-254 in the Star emulation of the old Gemini 10X and SG-10.

I wish that the suppliers of products for the TI, and the developers of new products, would advertise in MICRO-pendium so we could find out what they have to offer!

continued on page 18

Programming Music part 3

by Jim Peterson, Tigercub Software, USA

In Part 1 of this series, I showed you the simple routine to set up a musical scale, and showed you how easy it was to merge in various routines to create different effects in single-note music. In Part 2 I showed you how to key in single-note melodies from sheet music. Now, we will get into 3-part harmony.

But first, there are a few more things I should have told you about reading music. You will often see curved lines arching over two or more notes. If the notes are not all the same, ignore those lines - they call for phrasing which you cannot really accomplish. But, if the line curves over two or three of the same note, you will get a better effect if you add all their duration values together and program them as a single note. For instance, if your chart gives a whole note a value of 8 and a half-note a value of 4, and the music has a curved line over a whole note followed by a half-note, just program one note with a duration of 12.

You may find a heavy black bar at the beginning of a measure, with a colon to its right, and somewhere later in the music will be a heavy bar with a colon at its left. This means that the notes between those bars are to be played through twice - and naturally you will want to save time by programming them in a GOSUB as I showed you in Part 2. It can get more complicated than that, but generally you can follow the lyrics to decipher what to do.

Rather rarely, you may find three notes, usually joined together, with a 3 above them. These are called a triplet, and all three of them are to be played, with the same duration for each, in the length of time it would normally take to play one of them. These can create a problem under any method of music programming. The best method is to divide the duration of the note by three and write individual CALL SOUNDs in your music, rather than a GOSUB to a routine, to handle those notes.

Now, let's get on to 3-part harmony. It is just the same as keying in single note music, except that you must also give frequency values to B and C - and, as before, you have to give those values only when they change.

So, load the SCALE routine from the first lesson, and key in this bit of music to experiment with. Notice that I found three repeating phrases and put them in subroutines in 500, 600 and 700 to make this shorter.

```
110 GOSUB 500 :: T=4 :: A=15 :: B=11 :: C=9 :: GOSUB 100
0 :: T=8 :: A=18 :: GOSUB 1000 :: T=2 :: A,B,C=0 :: GOSU
B 1000 :: T=2 :: A=23 :: B=18 :: C=15 :: GOSUB 1000 :: G
OSUB 600
120 T=2 :: A=21 :: B=18 :: C=15 :: GOSUB 1000 :: A=23 ::
GOSUB 1000 :: T=12 :: A=20 :: B=16 :: C=11 :: GOSUB 1000
130 T=2 :: A,B,C=0 :: GOSUB 1000 :: GOSUB 500 :: T=4 ::
A=21 :: B=16 :: C=13 :: GOSUB 1000 :: T=10 :: A=25 :: GO
SUB 1000
140 T=2 :: A=28 :: GOSUB 1000 :: GOSUB 600
150 T=2 :: A=27 :: B=23 :: C=18 :: GOSUB 1000 :: A=30 ::
GOSUB 1000 :: T=10 :: A=28 :: B=23 :: C=20 :: GOSUB 1000
160 T=2 :: A,B,C=0 :: GOSUB 1000 :: T=3 :: A=28 :: B=23
:: C=20 :: GOSUB 1000 :: T=1 :: A=27 :: GOSUB 1000 :: GO
SUB 700
170 T=6 :: A=25 :: B=21 :: C=9 :: GOSUB 1000 :: T=2 :: A
=23 :: B=18 :: C=15 :: GOSUB 1000
180 T=10 :: A=20 :: B=16 :: C=11 :: GOSUB 1000 :: T=2 ::
 A,B,C=0 :: GOSUB 1000
190 T=3 :: A=28 :: B=23 :: C=20 :: GOSUB 1000 :: T=1 ::
A=27 :: GOSUB 1000 :: GOSUB 700
200 T=4 :: A=25 :: B=21 :: C=16 :: GOSUB 1000 :: A=21 ::
 B=18 :: C=15 :: GOSUB 1000
210 T=14 :: A=20 :: B=16 :: C=11 :: GOSUB 1000 :: T=2 ::
A,B,C=0 :: GOSUB 1000 :: STOP
500 T=2 :: A=23 :: B=20 :: C=16 :: GOSUB 1000 :: A=28 ::
GOSUB 1000 :: A=27 :: GOSUB 1000 :: A=28 :: GOSUB 1000
:: A=27 :: GOSUB 1000
510 A=28 :: GOSUB 1000 :: A=23 :: B=20 :: C=16 :: GOSUB
1000 :: A=20 :: B=16 :: C=11 :: GOSUB 1000 :: A=16 :: B=
11 :: C=8 :: GOSUB 1000 :: RETURN
600 T=2 :: A=27 :: B=23 :: C=18 :: GOSUB 1000 :: A=23 ::
B=18 :: C=15 :: GOSUB 1000 :: A=21 :: GOSUB 1000 :: A=2
3 :: GOSUB 1000
610 A=27 :: GOSUB 1000 :: A=23 :: GOSUB 1000 :: RETURN
700 T=4 :: A=27 :: B=21 :: C=16 :: GOSUB 1000 :: T=8 ::
A=25 :: GOSUB 1000 :: T=3 :: A=27 :: B=23 :: C=18 :: GOS
UB 1000
710 T=1 :: A=21 :: GOSUB 1000 :: T=4 :: A=25 :: B=21 ::
C=16 :: GOSUB 1000 :: T=8 :: A=23 :: B=20 :: C=16 :: GOS
UB 1000
720 T=3 :: A=25 :: B=21 :: C=16 :: GOSUB 1000 :: T=1 ::
A=23 :: GOSUB 1000 :: T=2 :: A=23 :: B=18 :: C=15 :: GOS
UB 1000
730 A=21 :: GOSUB 1000 :: A=20 :: GOSUB 1000 :: A=21 ::
GOSUB 1000 :: RETURN
```

Save that under the filename ROSES, clear the memory with NEW, and key this in -

```
1000 CALL SOUND(D*T,N(A),V1,N(B),V2,N(C),V3):: RETURN
```

Save that by-

```
SAVE DSK1.PLAIN3,MERGE .
```

Load ROSES again and merge it in by MERGE DSK1.PLAIN3 . Add a line -

105 D=200 and RUN it.

Sounds rather raw and harsh, does it not? Try changing that line 105 to -

```
105 D=200 :: V2=5 :: V3=8
```

Try it again. Sound better? The first time, all 3 voices were being played at the loudest volume. Usually computer music will sound better if the harmony notes are given a lower volume. Experiment and find the volumes you like best. Is the music too slow for you? Just change the value of D. Is it not in your singing key? Just change the value of F in line 100, as I showed you before.

But, does the music still have too strong a beat for your taste? Clear the memory again and key this in -

```
1000 CALL SOUND(-4250,N(A+Z),V1,N(B+Z),V2,N(C+Z),V3):: G
OSUB 1010 :: RETURN
1010 FOR W=1 TO T*D :: NEXT W :: RETURN
```

Save that as NEG3,MERGE because it uses negative duration for 3 voices. Then load ROSES again and merge it in. This time, try line 105 with D=50 and with V2 and V3 as you wish. Sound smoother?

In lines 110, 130, 160, 180 and 210 of ROSES, you will find A,B,C=0 . That makes all three voices silent, because in line 100 N(0) is given a frequency of 40000 which is above the range of human hearing. This is how I programmed those silent pauses, the "rests" which were written in the music.

On a piano or guitar, the strings continue to vibrate during a rest, so that the sound gradually fades out. However, the electronically generated tones of a computer stop very suddenly. That is why I often add the duration of the rest to the duration of the preceding note, and play it right on through. Some people think that it does not sound right, so here is another solution. Clear memory again and key this in -

```
2000 FOR W=2 TO 8 STEP 8 :: CALL SOUND(-999,N(A+Z),V1+W,
N(B+Z),V2+W,N(C+Z),V3+W):: GOSUB 2010:: NEXT W :: RETURN
2010 FOR Y=1 TO T*D/4 :: NEXT Y :: RETURN
```

Save that as REST,MERGE. Load ROSES again, merge in SCALE and NEG3 (this will not work well with PLAIN3) and merge in REST. Now go to lines 110, 130, 160, 180 and 210, delete the A,B,C=0 :: and change the GOSUB 1000 after it to GOSUB 2000. Add line 105, run it and see if you like that better. Anyway, keep it for now because we will use it again.

You will probably want to have the music play through more than once. Just add :: FOR J=1 TO 4 to the end of line 105 (if you want it to play 4 times) and change the end of line 210 to read NEXT J :: STOP .

I said that you could change the key of the music just by changing the value of F in line 100. There is also a way to change it while the music is playing. After the FOR J=1 TO 4 in 105 put:

```
:: Z=Z-(J=2)*3-(J=3)*1+(J=4)*4
```

That is somewhat complicated but it just means to play the second time three whole keys higher, the third time one key higher still (I know the *1 is unnecessary!) and drop back 4 keys for the 4th time, so you can take it from there and modify it as you wish. If you want to use that routine with silent rests, change the GOSUB after each rest to 3000 instead of 1000, and add this line -

```
3000 CALL SOUND(-4250,N(A),V1,N(B),V2,N(C),V3):: GOSUB 1
010 :: RETURN
```

This tune happens to end in a rest, which is unusual. If you key in another tune and it seems to end too abruptly, just after that NEXT J and before the STOP, put in a long duration such as T=12 and a GOSUB 2000 to that REST routine to fade out more slowly.

Now, when you are keying in your own tunes, the notes on your sheet music will usually have two or three of those little eggs on the stem. It is best to use the upper one for A, the next one for B, and the lower one for C; the computer could care less, but you will find it easier to keep track of what you are doing. If there are less than three, just go directly below to the bass clef and find a note there. If you still do not have enough, you can always use 0 to make that voice silent. Or, you can usually just let the previous note continue. If your sheet music has guitar chords - those little square grids with dots on them - above the staff, they will give you some help - if there is no guitar chord above the note you are working on, the chord has not changed and it is safe to use the previous harmony notes.

There are many other CALL SOUND routines you can use for different effects. This is similar to the one that Bill Knecht used for his hymns - I call it VIBRA.

```
105 D=1 :: V1=1 :: V2=5 :: V3=11
1000 FOR J=1 TO T*D :: CALL SOUND(-99,N(A),V1,N(B),V2,N(
C),V3):: CALL SOUND(-99,N(A)*1.01,V1,N(B),V2,N(C),V3)::
NEXT J :: RETURN
```

This one I call WUBBA, for no good reason -

```
105 D=1 :: V1=1 :: V2=5 :: V3=11
1000 FOR J=1 TO T*D :: CALL SOUND(-99,N(A),V1,N(B),V2,N(
C),V3):: CALL SOUND(-99,N(A)*1.01,V1,N(B),V3,N(C),V2)::
NEXT J :: RETURN
```

And this one I call TREM -

```
105 D=1 :: V1=1 :: V2=5 :: V3=11
1000 FOR J=1 TO T*D :: CALL SOUND(-999,N(A),V2,N(B),V2,N
(C)*1.01,V3):: CALL SOUND(-999,N(A)*1.01,V1,N(B),V2,N(C)
,V3):: NEXT J :: RETURN
```

I included line 105 in those, to merge in the duration and volumes along with the sound routine. Change the value of D to suit yourself, even in decimal increments such as D=1.5 .

It is easy to play a song repeatedly but with a different effect each time. Merge in VIBRA and change its line number to 1010. You can do this by typing 1000 and FCTN X, Enter, FCTN 8 to bring it back, type over the line number, and Enter. Merge in WUBBA and change it to line 1020 in the same way, then TREM and change it to line 1030. Add :: FOR R=1 TO 3 to the end of line 105. Put in a new line 1000-

```
1000 ON R GOSUB 1010,1020,1030 :: RETURN
```

And change the end of line 210 to NEXT R :: STOP.

Next time - more different effects, and autochording. O

continued from page 8

```
       BLWP @STRREF
       MOV  @SBUF1,R0
       ANDI R0,>7F              get character
       AI   R0,>60              add offset
       SLA  R0,3                get address
       LI   R1,SBUF4
       LI   R2,8
       BLWP @VMBR               read in definition
       LI   R3,>2C00            code for ","
       LI   R4,SBUF4            input data
       LI   R5,8                data count
       LI   R6,0                output byte count
       LI   R7,SBUF3+1          output buffer
STR00
       CLR  R0                  start with zero
       LI   R2,>3000            ASCII for zero
       MOVB *R4+,R0             get next byte
       LI   R1,>6400            100
       CB   R0,R1
       JHE  STRHU               at least 100
       LI   R1,>0A00            10
       CB   R0,R1
       JHE  STRTE               at least 10
       JMP  STRUN               only units
STRHU
       AI   R2,>0100            increment
       SB   R1,R0
       CB   R0,R1
       JHE  STRHU               loop until less than 100
       MOVB R2,*R7+             store byte
       INC  R6                  byte count
       LI   R2,>3000
       LI   R1,>0A00
       CB   R0,R1
       JL   STR01
STRTE
       AI   R2,>0100            increment
       SB   R1,R0
       CB   R0,R1
       JHE  STRTE               loop until less than 10
STR01
       MOVB R2,*R7+             store byte
       INC  R6                  byte count
       LI   R2,>3000
STRUN
       AB   R0,R2
       MOVB R2,*R7+
       INC  R6
       DEC  R5
       JLE  STR02
       MOVB R3,*R7+             put in comma
       INC  R6
       JMP  STR00
STR02
       SWPB R6
       MOVB R6,@SBUF3
       LI   R0,0
       LI   R1,2
       LI   R2,SBUF3
       BLWP @STRASG
       RTWP
*
*
*
MYWS   BSS  32
SBUF1  BYTE 1,0
SBUF2  BYTE 56,0
SBUF4  BSS  56
SBUF3  BYTE 80,0
       BSS  80
       END
```

o

XHi - Graphics for 9938

by Jan Alexandersson, Sweden

A new fantastic program for DIJIT AVPC, Geneve or Mechatronic with video processor 9938 has come from:

Alexander Hulpke
Sadowastrasse 68
D-5600 WUPPERTAL 1
West Germany

You can order it from the address above by sending a suitable fairware donation (I recommend at least 20 DM) and 7 DM for disk and postage. The program gives you the possibility to use a large number of CALL LINK to get high resolution graphic within an Extended Basic program. The manual on the disk for XHI version 3.6 from 29 September 1989 has 20 pages. The content of the disk:

XHI	The program XHI
XHI/T	Source code
XHIDOX	Archived manuals
XHIDOC	English manual (in XHIDOX)
XHIDEU	German manual (in XHIDOX)
COLDEF	Redefine colours
LOAD	XHI-loader with SysTex
CHARA1	Character set for HARDCOPY
HCSETUP	Printer Setup for HARDCOPY
HCLOAD	Extended Basic-loader for HARDCOPY
YLOAD	EXTENDED BASIC EA5 Loader
HARDCOPY	Version 1.2, EA5
HIRESDEMO	Demo-program with graphic
UHR	Demo, analog clock
FIXSTERNE	Demo, star constellations
KUGEL	Simple Ray-tracing program
KUGEL;PAR	Example of start values
KUGEL;PIC	Complete picture

A program which resembles XHI but works with CALL LINK for Text2 is called X80. It is now in test version 0.91 from 11 August, 1989.

1. GRAPHIC MODES FOR 9938

Of the ten different graphic modes in the video processor 9938 are six completely new and there are also the usual four graphic modes from the 99/4A.

Text1-2 or Graphic1 should be chosen if you mostly need text. These three modes are the only ones where the character set is the same for the whole screen. Text2 should be used for all new programs when much text is shown i.e. for word processing, spread sheets and data bases. Graphic1 could be the wright thing for start screens and menus where you get big fat characters if 32 per row is enough. You can create graphic with some difficulty in Graphic1 with CALL CHAR or similar.

Graphic6-7 should be used if you mostly want to use graphic. Graphic6 has more pixels (512x212) on the screen if 16 (out of 512) colours is enough. If your priority is colours then you choose Graphic7 which has 256 colours simultaneously but less pixels (256x212). Graphic6-7 can with some difficulty be used for text. Remember that the result will only pixels so there is no way to read the text with CALL GCHAR or similar in assembler.

Graphic2 has the screen divided in three parts each with its own character sets. Each character can have several colours because each pixel row has its own data similar to CALL COLOR. Graphic2 is only interesting if you also want to use it with a 99/4A. Graphic3 is only good for old programs written in Graphic2 where you want to change to multicoloured sprites (mode2) without any need to change the rest of the code. Perhaps something for Parsec. Graphic4 is only useful for old 99/4A-programs written for Graphic2 where you want to change to true bitmap with unique colour for each pixels without the need to change so much of the code. Perhaps something for TI-Artist.

Graphic5 can only use four colours but it needs only 32 kbytes which can be used if you want to change between several pictures which is stored in VDP-RAM at the same time.

2. SPRITE MODES FOR 9938

Sprite mode 2 is used for Graphic3-7 which give you multicoloured sprites (max 32) where each pixel row has its own attribute as colour, collision, priority and early clock. You can have 8 sprites at the same pixel row before anything vanish. With Magnify 3 or 4 you have the possibility to use 16 different attributes because a sprite with four characters has 16 rows. As with Graphic2 you cannot use auto-motion of sprites which means the the program itself must do the move.

3. COLOURS WITH 9938

The four ordinary graphic modes which you have in the 99/4A get with a 9938 the possibility to redefine the colours as long as you use 16 out of 512 colours for Graphic1-2 and 2 out of 512 for Text1. Graphic6 has also 16 out of 512 colours for both pixels and sprites. Graphic7 has 256 out of 256 colours for pixels and 16 out of 16 colours for sprites. These 16 sprite colours are not the same as the standard 99/4A colours but are the same as the default colours in Geneve Myart. The 512 colours are created by setting the three basic colours red, green and blue to a value 0-7. Graphic7 gets 256 colours by setting blue in only 4 levels. Normal colours in 99/4A:

No	Colour	Red	Green	Blue
0	Transparent	0	0	0
1	Black	0	0	0
2	Green	1	6	1
3	Light green	3	7	3
4	Dark blue	1	1	7
5	Blue	2	3	7
6	Dark red	5	1	1
7	Cyan	2	6	7
8	Red	7	1	1
9	Light red	7	3	3
10	Dark yellow	6	6	1
11	Light yellow	6	6	4
12	Dark green	1	4	1
13	Magenta	6	2	5
14	Grey	5	5	5
15	White	7	7	7

Colour 0 can only be altered if the screen colour is 0 or VDP register R8 bit 5 i set to 1. In Basic and XHI will these 16 colours be numbered 1-16 (corresponds to 0-15 in the table).

4. TO LOAD XHI

The program XHI is loaded with CALL INIT::CALL LOAD("DSK1.XHI"). You get a number of CALL LINK for Graphic6 and Graphic7. You can always jump between the usual Extended Basic -screen (Graphic1) and the high resolution screen (Graphic6 or Graphic7). You can write both to the Extended Basic-screen and the XHI-screen. Both is in memory but only one is displayed. With the XHI-screen active you can still use all Extended Basic-commands to write and read the Extended Basic-screen even though you do not see it on the screen. The XHI-screen is shown without any disturbances also when executing program lines. You can use SIN without any problem which is difficult with 99/4A and Graphic2. XHI-LINK can only be used when the XHI-screen is active. CALL SCREEN and CALL MAGNIFY work on the Extended Basic - and XHI-screen at the same time because they only write to VDP registers.

5. CALL LINK WITH XHI

All XHI commands can also be executed with logic operators IMP, AND, OR, XOR and NOT. You can add a Myart picture to the previous picture with OR. High resolution is called by CALL LINK("PLOT", VCOOR,HCOOR,COLOR) as follows:

a.Graphic modes

HICLR	Graphic6 + clear screen
HIRES	Graphic6
CLR256	Graphic7 + clear screen
MOD256	Graphic7
NORMAL	Graphic1
COLMIX	redefine colour
COLRES	reset colours

b.Graphic

BACK	screen colour (out of 256)
also CALL SCREEN (only 16 colours)	
DCOL	colour for plot
PLOT	write pixel
CLS	clear/colour the pixel area
GPIX	read pixel
CIRCLE	write circle
LINE	draw line
FILL	fills an area
DRAWTO	continue line (turtle)
SAVE	save Myart-picture
LOAD	load Myart-picture
ARTLES	TI-Artist to Myart-screen

A stand-alone program for XHI can print Myart pictures on a printer. This HARDCOPY version 1.2 can also be used without a 9938-processor.

c.Window

VIPORT	create window for graphic
FILSCR	clear/colour window
COPY	copy window
XPAND	expand the window horizontally
YPAND	expand the window vertically
REDUCX	reduce the window horizontally
REDUCY	reduce the window vertically

d.Text

PRINT	writes a text string
CWIDTH	choose pixel width for characters

With CWIDTH 6 you can have 80 char/row and with CWIDTH 5 you get 102 char/row. In normal BASIC you have the width 5 and pixel 2-6 is used. If you create your own characters with the width 4 (+ 1 pixel space) then you could have 102 characters/row.

e.Sprites

SPRITE	activates sprite
LOCATE	move sprite
SCOL	alter multicoloured sprite
PATTER	sprite-character
DELSPR	delete sprite
also CALL MAGNIFY works	
MOTION	interrupt motion of sprite
STOP	stops sprite

The author has plans to include PEEKV, POKEV, COLOR HARDCOPY and circle with correct proportions in G6.

6. PROBLEMS

G6 CIRCLE has wrong proportions because the pixels are more dense in the X-direction than in the Y-direction.

G7 CLR256 makes all pixels black so you must also use CLS. Transparent appears to be missing among the 256 colours so you must choose the same colour for CLS and BACK.

MOTION of sprite with DIJIT AVPC EPROM version 1 will destroy the 16 characters of the first program line. You must put a REM with at least 16 characters in the first line. I use:

```
100 !123456789ABCDEF
```

The line must be high up in the CPU-RAM which you are sure to get if you:

```
SAVE DSK1.MERGE,MERGE
NEW
MERGE DSK1.MERGE
```

The new EPROM PWRUP2A works without this problem.

7. PROGRAMS FOR XHI

```
100 ! LOAD MYART GRAPHIC6
110 ! JAN ALEXANDERSSON
120 ! SWEDEN
130 ! 1989-08-19
140 ! Extended Basic + 9938 + XHI
150 CALL CLEAR
160 INPUT "FILE ":FILE$
170 IF FILE$="" THEN END
180 CALL LINK("HICLR")
190 ON ERROR 220
200 CALL LINK("LOAD",FILE$)
210 CALL KEY(5,K,S):: IF S<1 THEN 210
220 CALL LINK("NORMAL")
230 CALL LINK("COLRES")
240 GOTO 160
```

```
100 ! LOAD MYART GRAPHIC7
110 ! JAN ALEXANDERSSON
120 ! SWEDEN
130 ! 1989-07-28
140 ! Extended Basic + 9938 + XHI
150 CALL CLEAR
160 INPUT "FILE ":FILE$
170 IF FILE$="" THEN END
180 CALL LINK("CLR256")
190 ON ERROR 260
200 OPEN #1:FILE$,FIXED 128,
INPUT :: LINPUT #1:COLOR$ :: CLOSE #1
210 COLOR=ASC(COLOR$)
220 CALL LINK("BACK",COLOR)
230 CALL LINK("CLS",COLOR)
240 CALL LINK("LOAD",FILE$)
250 CALL KEY(5,K,S):: IF S<1 THEN 250
260 CALL LINK("NORMAL")
270 GOTO 160
```

```
100 REM XHI ARTLES G6
110 INPUT "TIARTIST FILE ":F$
120 INPUT "MYART G6 FILE ":M$
130 CALL LINK("HICLR")
140 ON ERROR 180
150 CALL LINK("CLS",16)
160 CALL LINK("ARTLES",F$)
170 CALL LINK("SAVE",M$)
180 CALL LINK("NORMAL")
```

```
100 REM XHI FILSCR G7
110 CALL LINK("CLR256")
120 ON ERROR 230
130 CALL LINK("BACK",255)
135 CALL LINK("CLS",255)
160 CALL LINK("VIPORT",10,50,100,100)
170 CALL LINK("FILSCR",7)
180 CALL LINK("VIPORT",10,150,100,200)
181 CALL LINK("FILSCR",14)
220 CALL KEY(5,K,S):: IF S<1 THEN 220
230 CALL LINK("NORMAL")
```

```
100 !123456789ABCDEF
110 REM XHI MOTION G6
120 CALL SPRITE(#1,64,1,50,50,10,10)
130 CALL LINK("HICLR")
140 ON ERROR 210
150 CALL SCREEN(16)
160 ATTRIB$="08080808080808080808080808080808"
170 CALL LINK("SPRITE",1,70,ATTRIB$,50,50)
180 CALL MAGNIFY(2)
190 CALL LINK("MOTION",1)
200 CALL KEY(5,K,S):: IF S<1 THEN 200
210 CALL LINK("NORMAL")
```

o

A Look at the PRK Module

by R.A. Green, Ontario Canada

The Personal Record Keeping module (PRK) and its companion the Personal Report Generator module (PRG) provide an easy to use a data base system. However, it would often be useful to be able to process a data base with a BASIC program. The PRK module provides you with this facility!

With the PRK module plugged in, you will find that TI BASIC has several new subroutines which your programs can CALL. These are:

CALL P(...) - reserve VDP RAM for PRK data base loading;

CALL A(...) - facility similiar to Extended BASIC's ACCEPT AT statement;

CALL D(...) - facility similiar to Extended BASIC's DISPLAY AT statement;

CALL G(...) - get fields from a PRK data base record or put fields into a PRK data base record;

CALL H(...) - get the number of records in a data base or set the number of records in a data base;

CALL L(...) - load a PRK data base into VDP RAM from disk or cassette;

CALL S(...) - save a PRK data base from VDP RAM to disk or cassette.

While not all features of these subroutines are known, with what will be described later and a little experimentation you can make good use of them to write TI BASIC programs to process your own data bases.

The "A" and "D" subroutines provided by the PRK module can be freely used in any TI BASIC program to provide a facility similiar to Extended BASIC's ACCEPT AT and DISPLAY AT statements. These two subroutines are fully described in Volume 1 Number 4 of the 99er Magazine, page 72. A short description of each is given below.

Display At

The BASIC CALL statement to invoke this subroutine is:

```
CALL D(R,C,L,VALUE, ... )
```

where:

R - is the row number for the display,
C - is the column number for the display,
L - is the length of the field into which the value is to be displayed,
VALUE - is a numeric or string constant or variable that is the value to be displayed.

The four parameters may be repeated as many times as desired to do several displays in a single statememt. A single display is restricted to a single row on the screen.

Accept At

The BASIC CALL statement to invoke this subroutine is:

```
CALL A(R,C,L,F,VAR,MIN,MAX)
```

where:

R - is the row number where the accepted data will appear,
C - is the column number where the accepted data will appear
L - is the length of the field into which the entered value is to appear,
F - is a function value set by the accept at subroutine to indicate how the input request was ended; the value will be one of:
1 ENTER key pressed,
2 CLEAR key pressed,
3 AID key pressed,
4 REDO key pressed,
5 PROC'D key pressed,
6 BEGIN key pressed,
7 BACK key pressed,
VAR - in the numeric or string variable into which the accepted value is placed,
MIN - is the minimum value acceptable for numeric input,
MAX - is the maximum value acceptable for numeric input.

Processing a PRK data base

Your TI BASIC program can process a PRK data base. By process, I mean that you can read records from, update records in or add records to the data base. First, however, you must use the PRK module to create and save the data base or a model (i.e. empty) data base.

Next, when using your TI BASIC program to process the data base you must have the PRK module plugged in. You select TI BASIC from the master title screen, and in BASIC command mode you enter:

```
CALL FILES(n)        (for disk users only)
NEW
CALL P(size)
OLD program-name
```

where:

n - is the number of disk files your program requires (minimum of 1),
size - is the amount of VDP RAM to reserve for the PRK data base.

Then you can run your TI BASIC program.

The value for "size" requires some experimentation. Your BASIC program and the PRK data base must share the VDP RAM. A large value for "size" means that you can have only a small program. A small value for "size" means that only a small data base can be processed. A value of about 8000 is a good starting point; you can adjust this up or down depending upon the size of your program and data base.

You may find that it is convienent to segment your processing of a data base into three separate TI BASIC programs.

First, a program to "convert" your data base into an ordinary BASIC file. This program could be very small thus allowing a large data base.

Second, a program to process the BASIC file, generating an updated BASIC file. This program can be completely independent from PRK. In fact, it could be written in any language available on the TI99/4A.

Third, a program to "convert" the updated BASIC file back to a PRK data base. This program, like the first one, could be very small allowing a large data base.

Reading a PRK data base

In order to process records or fields from a PRK data base, your BASIC program must first "load" the data base into VDP RAM. This is done via the BASIC statement:

```
CALL L(file-name,X)
```

Where "file-name" is a string variable or constant that is the name of the PRK data base (for example, "DSK1.MYDB", "CS1", etc.). And where "X" is a numeric variable whose use is unknown. It may be that X is set by the L subroutine to indicate error conditions.

Once the data base is loaded, your program can determine the number of records or "pages" in the data base via:

```
CALL H(1,6,0,NOREC)
```

Where the first parameter, "1", tells the H subroutine that you are requesting the number of records in the data base which it will set into the numeric variable, "NOREC". The use of the second and third parameters, six and zero, is unknown.

Your program can obtain the value of a field in the data base via:

```
CALL G(1,RN,FN,C,variable)
```

Where the first parameter, "1", indicates a request to the "G" subroutine to obtain data from a field in a record. "RN" is a numeric variable or constant that is the number of the record from which the data is to be obtained. "FN" is a numeric vairable or constant that is the number of the field from which data is to be obtained. The field numbers are as shown by the PRK module when defining a data base. "C" is a numeric variable set by the "G" subroutine. It is set to zero if the field in the record contains data. "Variable" is a numeric or string variable into which the value of the field will be set by the "G" subroutine. Whether "variable" is a string or numeric variable depends upon how the field was defined when the data base was created with the PRK module.

Example. Read field 1 and 3 of all records.

```
100 CALL L("DSK1.MYDB",X)
110 CALL H(1,6,0,NOREC)
120 FOR RN=1 TO NOREC
130     CALL G(1,RN,1,FIELD1$)
140     CALL G(1,RN,3,FIELD3$)
150     PRINT FIELD1$,FIELD3$
160 NEXT RN
```

Writing a PRK data base

In order to write or update records or fields in a PRK data base, your BASIC program must first "load" the data base into VDP RAM. The data base may be full or a model (i.e. empty). This is done via the BASIC statement:

```
CALL L(file-name,X)
```

Where "file-name" is a string variable or constant that is the name of the PRK data base (for example, "DSK1.MYDB", "CS1", etc.). And where "X" is a numeric variable whose use is unknown.

Once the data base is loaded, your BASIC program call fill values into the fields of the records via:

```
CALL G(0,RN,FN,C,value)
```

Where the first parameter, zero, indicates a request to the "G" subroutine to set data into a field in a record. "RN" is a numeric variable or constant that is the number of the record into which the data is to be placed. "FN" is a numeric vairable or constant that is the number of the field into which data is to be placed. The field numbers are as shown by the PRK module when defining a data base. "C" is a numeric variable whose use is unknown. "Value" is a numeric or string variable or constant that is the value to be set into the field by the "G" subroutine. A field can be cleared to blanks or zero via:

```
CALL G(2,RN,FN,C,X)
```

Where the first parameter, two, tells the "G" subroutine to clear field number FN in record RN. The use, if any, of "C" and "X", in this case is unknown.

After your program has filled in or updated all desired fields and records, it must tell PRK how many records there now are in the updated data base via:

```
CALL H(0,6,0,NOREC)
```

Where the first parameter, zero, tells the H subroutine that you are setting the number of records in the data base to "NOREC". The use of the second and third parameters, six and zero, is unknown.

Finally, your program must save the updated data base via:

```
CALL S(file-name,X)
```

Where "file-name" is a string variable or constant that is the name of the PRK data base (for example, "DSK1.MYDB", "CS1", etc.) and where "X" is a numeric variable whose use is unknown. o

Treasurer's Report

by Geoff Trott

I have received a few letters recently which you may be interested in. The first one was from Jim Banfield of Armidale with the latest instalment of his interesting series on machine language. I am enjoying reading them as I type them in for Jim. I hope there are others also receiving some inspiration from them. Jim has sent in some interesting pictures of his system which we will publish soon.

The other letter was from Tony McGovern with the latest in the beta test version of the new editor. I know that many people I have spoken to are waiting for the release of this editor. Even if you do not have an 80 column card, the 40 column version will have quite a few of the enhancements. Tony is waiting for version 5.00 of Funnelweb before releasing the editor. I wonder if we can invite him to the TI-Faire and have a world release of version 5.00 of Funnelweb? That would be worth the price of admission on its own!

Income for August	$1431.19
Payments for July	$ 877.65
Excess of income over expenses for August	$553.54

o

continued from page 12

I wish that printers were made so that they would continually shift either the print head or the ribbon up and down, so that we could use up the entire width of the ribbon instead of that narrow strip down the center. Considering their outrageous prices for ribbon cartridges, they owe us that much!

I wish that the local stores would get together and set up a computer information service so I could use my modem to find out what stores have what I am looking for, and what their price is.

I wish that manufacturers of electronic equipment would stop labeling all the controls in raised black letters on a black background.

I wish most of all that Texas Instruments had continued developing our computer, that we now had a TI-99/9Z! o

continued from page 10

are used. Concepts taught are figures of speech, word meaning, and idioms. The module is designed for ages 9-10.

READING FLIGHT

For ages 11-12. A neat story about an archaeological dig on a south seas island called Bolo Island teaches classifying, summarizing, and outlining information. o

Beginning Forth - part 20

by Earl Raguse, UGOC, CA USA

Well, here we are at Beginning Forth 20, you should no longer be classed as beginners, so I guess it is time for me to sign off with a couple of special words.

The first of these is DUMP, a very useful word which lets you look at memory contents and print them out if so desired. The DUMP is in both HEX and ASCII side by side. The easiest way to find out is to just try DUMP. All you need on the stack is the address to start at and the number of memory locations to dump.

To find the address of a word in memory one uses ' (tick) and NFA to use with DUMP, you may guess as to how many memory locations to display to see the complete definition. As it happens Forth has a word for it. Just like the Greeks, Forth usually has a "word" for it. That word is -FIND, see TIFM APP D p7, which not only finds the PFA of a word, but its length. I have written another word LOC (LOCate) using -FIND, which is shown on Screen #77. I think its pretty self explanatory, especially, if you read up on -FIND.

Type in and load Screen #77 and you wilL be encouraged to enter a word to locate, if the word is found in the dictionary, you should see the following, assuming you entered LOC EXPECT

```
               Found    EXPECT
               Length    143
               PFA=     -21420
               NFA=     -21430

            Press Q to dump it
              else press any
```

If a word is not found, Forth will say so.

There is nothing particularly innovative going on here, except maybe the word | (I do not know its name, lets call it 'post'). Forth uses it | (post), not as a word, but to indicate the bottom of the stack. Just in case you have forgotten, the word ;S is what enables you to look at the stack (parameter, that is) without disturbing its contents. If you do not use this word you are missing out on a great tool, even better than DUMP.

Back to |, in this case I use it to line up text. Its even better when you have several lines of text and you want to move them toward the center of the screen instead of on the left margin. Later you will see | redefined, for essentially the same purpose, but a completely different word. The flexibility of being able to define it on the spot is the reason I do not incorporate it in my UFW's.

One of the advantages of being a newsletter librarian, (for UGOC) is access to all the exchange newsletters from other groups. One excellent newsletter is The Computer Voice from our sister group in San Diego. A gleaning from the Feb 86 issue shows how to get instant reverse video for use in prompts etc. The words were originated by Michal Jaegermann, from Edmonton in Alberta, Canada. You have heard of him before, remember his word PRINTS? The following Screens #78 and #79 were included in an article by Lutz Winkler, whom you have also seen quoted in this column. Is it not nice that there are so many smart people who are 99/4A owners?

I do not pretend to know all of what is going on with Screens #78 and #79; but the word ROLL is widely known by advanced Forth programmers. It is in effect a programmable ROT which allows one to access further down in the stack. It is related to ROT in much the same way that PICK is related to OVER. PICK allows one to dig down into the stack further than OVER. Many Forth programmers turn up their noses at words like PICK, they say it promotes sloppy work.

Now we have other fish to fry. Looking at Screen #78, FX is a word I can understand, its just the words that use it that I do not fully comprehend. FX in effect is the reverser; when you XOR a number with FFFF (in hex of course), all the 1's become 0's and vice-versa. What REV does, I think, is to take four numbers off the Parameter Stack, hereinafter the 'stack' and push them on the Return Stack, hereinafter the 'Rstack' or simply R. It then takes them off the Rstack one at a time, does an FX while returning them to the stack for use by XCHAR. XCHAR in turn is used by XFORM which is just a DO...LOOP covering the ASCII character set from decimal codes 34 to 127. Do not ask me why they excluded 33 the !, but they did, consequently ! does not work. I have not tried changing the word yet, there may be a good reason for it, but rest assured I will try it sooner or later.

But I digress, notice that XFORM uses CHARPAT and CHAR which have identical meanings as in Extended Basic. REV was discussed, and 5 ROLL just digs out the character number which is buried under the four XORed numbers on the stack, 80 + adds an offset of 128 (decimal) to the character number so that the standard character set is still intact. Notice also, that XFORM is executed in the immediate mode.

From here on is where my ignorance begins, the words RV, RS, RLIT, and R" are a little too advanced for me. RLIT replaces SLIT and R" replaces ." in resident Forth. I am not sure about the functions of RV (ReverseVideo) and RS. I have the feeling that the RS refers to STATE, and that these are 'STATE Smart' words. STATE is the address of the compiler status flag. I am not yet well enough informed to make use of it in words that I write.

It does not follow that one understands how a Forth word works to use it. How many Extended Basic programmers know how ACCEPT AT works internally?

On Screen #79, I have added the words PRMPT and .M to show how to use R". They are not a part of the Reverse Video essentials. OH YES!, before I forget, you must have -GRAPH loaded in order to have words like CHAR and CHARPAT available, else you must have a disk with all the necessary Screens #57-#64 from the original Forth System disk plus all the referenced screens on it when you try to load Screen #79.

Well, onward and upward, my next offering is Screens #80 and #81. Have you ever been programming away and then had to stop to dig up a reference to find the ASCII code for Q or W for instance? If so then ASC or AS are the words for you. If you have Screen #80 loaded and type either AS or ASC, you will get a complete ASCII table on the screen. Of course you must exit from the EDIT mode to do that (ie FCTN 9), and I strongly advise doing a FLUSH (or F;) before using ASC or AS,

```
SCR 77
 \ Word Location and Dump EGR 12 87
   0 VARIABLE LEN \ length
 : | CR CR 15 SPACES ;
 : PROMPT 5 12 AT ." Enter LOC and a word to find" ;
 : LOC CLS ( word) -FIND DUP CLS  0=
    IF 15 8  AT        ." Not Found " ABORT THEN
    IF 15 8  AT        ." Found   " OVER NFA ID.
    DUP LEN !        | ." Length  " .
    DUP              | ." PFA =  " .
    NFA DUP          | ." NFA =  " .
    12 16 AT           ." press Q to dump it"
    14 18 AT           ." else press any" THEN
 KEY 81 = IF LEN @ CLS DUMP ELSE PROMPT THEN ; CLS PROMPT




SCR 78
 \ Reverser Video #1   Michal Jaegermann
 \ Via Lutz Winkler and The Computer Voice
   BASE->R  57 CLOAD CHAR HEX
 : ROLL DUP 1 = IF DROP ELSE DUP 1 DO SWAP
```

```
     R> R> ROT >R >R >R LOOP 1 DO
     R> R> R> ROT ROT >R >R SWAP LOOP THEN ;
: FX FFFF XOR ;
: REV >R >R >R >R   R> FX R> FX R> FX R> FX ;
: XCHAR  DUP CHARPAT REV 5 ROLL 80 + CHAR ;
: XFORM 7F 22 DO I XCHAR LOOP ; XFORM
: RV BEGIN KEY DUP 1F > WHILE 80 OR EMIT8
     REPEAT DROP ;
: RS COUNT OVER + SWAP DO I C@ 80 OR
     EMIT8 LOOP ;
     -->

```

else you may lose something you had on the CRT. The words AS and ASC do the same thing in a different way, try them and see which you like.

Sometimes you may want a printed form of the ASCII table; sure, they are all over, but when you want one, where are they? Screen 81 takes care of that. I had originally thought that SWCH ASC UNSWCH would print out the list, but the printer ignores all my | (post) words which nicely locate text on the CRT, but the printer does not understand them. Try it and see, you get the data but not the format. Screen #81 uses only printer decipherable words SPACE, SPACES (essentially Tabs) and CR's to get the job done. In Forth there is always 3 or 4 ways to skin a cat.

Its been fun. See you at the meetings!

May the FORTH be with U, Always!

```
SCR 79
\ Reverser Video #2
: RLIT 22 STATE IF COMPILE SLIT
     WORD HERE C@ 1+ =CELLS ALLOT
     ELSE WORD HERE THEN ; IMMEDIATE
: R" [COMPILE] RLIT STATE IF COMPILE
     RS ELSE RV THEN ; IMMEDIATE
R->BASE DECIMAL
: PRMPT CLS
     12 6 AT R" THIS IS A PROMPT"
     14 8 AT R" DO SOMETHING" ;
: .M 10 10 AT R" WHAT'S GOING ON HERE? "
      4 12 AT R" Its rather heady stuff in case you "
      3 14 AT R" are not aware of how clever this is,"
      2 16 AT R"
Do not ask me how it works, but it does " ;


SCR 80
\ ASCII CHARACTER DISPLAY EGR 12 17 87
  0 VARIABLE CH  0 VARIABLE ROW  0 VARIABLE COL
: HD CLS 7 0 AT ." ASCII CHARACTERS BY NUMBER" CR CR ;
: .CL 22 2 DO COL @ I AT CH @ DUP EMIT SPACE . 1 CH +! LOOP
: AS HD 5 0 DO I 8 * 2 + COL ! I 20 * 32 + CH ! .CL LOOP ;
: | ROW @ AT ;
: ASC HD 2 ROW ! 52 32 DO
   I 00 + DUP  3 | .  6 | EMIT
   I 20 + DUP 10 | . 13 | EMIT
   I 40 + DUP 17 | . 20 | EMIT
   I 60 + DUP 24 | . 28 | EMIT
   I 80 + DUP 32 | . 36 | EMIT CR PAUSE 1 ROW +! LOOP QUIT
: .M CLS 10 12 AT ." PRINTS ASCII TABLE "
        10 14 AT ." just type ASC or AS" QUIT ;
.M


SCR 81
\ ASCII CHARACTERS PRINTED EGR 12 17 87
: S SPACES ;  : SP SPACE ;
: HD 14 S ." ASCII CHARACTERS BY NUMBER" CR CR ;
: PASCII  SWCH HD 64 32 DO
  10 S I 00 + DUP EMIT SP .
  10 S I 32 + DUP EMIT SP .
  10 S I 64 + DUP EMIT SP . CR LOOP
  FF UNSWCH ;

  CLS 11 12 AT ." ASCII TABLE TO PIO"
       12 16 AT ." <PASCII> Does It" QUIT
```

We have finally come to the end of this excellent series. We hope that you have enjoyed it and will find yourself browsing through these tutorials from time to time for years to come. o

The Home Computer

by Jim Peterson, Tigercub Software, USA

Can you stand a few more words from the last surviving advocate of the HOME computer?

And what is a HOME computer? It is a computer designed to be used in the home, to do whatever someone might do in the home that can be done better with the aid of a computer.

AND - the HOME computer is designed to be used by a person who has no particular interest in computers, who regards them as just another electronic tool to be used to make life easier or more enjoyable. Also, that person is probably just a bit intimidated by computers.

A person who is not interested in computers? Well, that eliminates everyone who is reading this, but read on anyway.

Now, what percentage of VCR owners have never learned to program their VCR? How many do not know what some of the buttons on their cable TV remote unit do? How many housewives are failing to take advantage of half the pushbuttons on their microwave, or their washing machine? I do not think anyone has the answer to those questions, but I am sure that the percentages are very large!

Many people who buy a new appliance NEVER read the manual. They learn some of its features by experimenting, and never use the rest. Most other people read the manual one time, file it away with the warranty or lose it, and operate the appliance based on what they remember from that one reading. Of course, there are an increasing number of people who are incapable of reading the manual at all, and very few people who are capable of writing a manual that anyone can understand!

The average home computer buyer, knowing nothing about computers, can easily be convinced that he needs 640k a RAM, a hard drive, a mouse, and who knows what else. He needs all those things like he needs a hole in the head, and he is completely baffled by the technical jargon in the manuals that come with the machine.

His computer probably comes bundled with an assortment of "free" software that is alleged to be worth more than the machine itself. It is probably excellent software - but each program comes with a thick manual, hopefully written in intelligible English, which must be studied before the program can be used.

Big programs like that are fine for the workplace, where a worker becomes familiar with a program and remembers how to use it because he uses it every day. For the typical home computer user, they are totally impractical.

So, what is a HOME computer? It is a computer with no more memory than is needed to do the job, practically automatic in operation (i.e., with built-in disk operating system!), with one disk drive, and with an adequate supply of short simple programs to do what needs to be done at the moment and no more, so simple that they can be operated by reading on-screen instructions and prompts.

I happen to own such a computer. It is called the Texas Instruments TI-99/4A HOME Computer. o

continued from page 10

```
END    CLR  R0
       MOVB R0,@>837C
       LWPI >83E0
       MOV  @SAVRTN,R11
       B    *R11

       END
```

o

Quad density Disks

by Jan Alexandersson, Sweden

A disk stores data as sectors each with 256 bytes. Such a sector is the smallest amount of data that you can write to a disk, so the computer writes always a whole sector at a time. You sometimes write to a data file with PRINT #1:A,B,C which is less than 256 bytes. The computer will store it in a buffer in RAM until it can write the whole sector. Do not forget to close the file with CLOSE #1 because there may still be data in the buffer.

DISK HEADER

There are two special sectors number 0 and 1 on a disk for management of all files on the disk. Sector one has pointers, sorted by filename in alphabetic order, which points to the sector with the file header. This file header shows file name, file structure (DIS/VAR 80, INT/FIX 128, PROGRAM etc.) and which sectors that contain data belonging to the file. Sector zero has general information about the disk as number of sides, tracks/ side, sectors/track and a table of which sectors are occupied so the computer can know which sectors it can use for a new file. TI made the table in a way that it can only hold 1600 sectors. A SS/SD 90 kbytes disk uses 360 sectors and a 360 kbytes DS/DD disk uses 1440 sectors.

If you have a disk with more than 400 kbytes (1600 sectors) then there is no space for all sectors. This is the reason why a 512 kbytes Myarc RAM disk cannot use more than 400 kbytes. The remaining 112 kbytes is used as expansion RAM 32 kbytes, printer buffer and working memory for Myarc Extended Basic II. Horizon RAM disk has solved the problem by using two disk numbers each with 360 kbytes (1440 sectors) on each card. Corcomp 512 kbytes RAM disk has 32 kbytes expansion memory but the remaining 480 kbytes is used as a RAM disk. This can be managed by the use of an additional sector for marking of used sectors. An empty RAM disk will then have 3 used sectors (0-2).

QUAD DENSITY (5.25 inch)

There is a Myarc disk controller (with DS/QD EPROM) and a Myarc hard disk controller HFDC. HFDC EPROM H6 is not OK because SAVE of a file from Basic or TI-Writer will destroy the disk (DM V works despite of this). You must have EPROM H10 or H11 for quad density. Both types of disk controller can use disks with 720 kbytes DS/QD (double sided/quad density). This mean that there are 2880 sectors. Myarc has solved the problem with sector zero by letting each bit mark two sectors at the same time (1 allocation unit is 2 sectors) which means that you cannot use less than two sectors for file header and two sectors for data. The shortest possible file is then four sectors. A program will despite of this write and read single sectors of 256 bytes. A disk drive for quad density can read single density and double density but only write quad density.

The file headers are mainly stored on sectors 2-33 to speed up the search of files. This means that a normal SS/SD or DS/DD can have 32 file header for fast access. Higher numbers is used if there are more than 32 files. The computer will also store data sectors on sectors 2-33 if there are less than 32 files but not until all other sectors are used.

A DS/QD disk with a 2 sector file header can only have 16 file headers on sector 2-33 with fast access. My Myarc HFDC card cannot store any data sectors on sector 2-33 when I have a few long files. This often happens when you have archived files.

DISK MANAGER FOR 720 KBYTES DS/QD

There are some disk manager suitable for 720 kbytes:

- Funnelweb 4.30 Quick Directory
- Funnelweb 4.30 Show Directory 40 or 80 columns
- Funnelweb 4.30 Disk Review 40 or 80 columns
- Myarc CALL DIR
- Myarc DM III
- Myarc DM V for HFDC
- Disk Utilities

You cannot use DM1000 for quad density because it will misunderstood sector 0 and that one allocation unit is two sectors. Hard Master will show the total number sectors in a wrong way for QD-disks.

The well working disk managers all show the same number of free and used sectors. This corresponds to the marked allocation units in sector zero. The file length is shown in different ways:

DS/QD	Header	Data sectors	Minimum
FW QD	1	used	2
FW SD	1	used	2
FW DR	2	even number	4
CALL DIR	2	used	3
DM III	2	even number	4
DM V	2	even number	4
DSKU	2	even number	4

Older versions of Funnelweb may differ from this. Funnelweb QD and SD shows how much space is needed for copying to a smaller disk. CALL DIR shows the number of sectors that cannot be used to increase the file. If you have an odd number of data sectors then there is one free sector which only can be used by that particular file. DM V, DSKU and FW DR show the number of sectors that cannot be used by other files. None of the disk managers has any knowledge of that sector 2-33 cannot be used for data but only for file header.

TI-Writer behaves strange because the shortest possible file has two data sectors when you save it to a DS/QD disk. If this file is copied to a smaller disk then it will have a total length of 3 sectors. Three data sectors can despite this be used with TI-Writer on a QD disk.

You can also use 3.5 inch DS/DD 720 kbytes in the same way as 5.25 inch DS/QD.

HIGH DENSITY (3.5 inch)

Myarc HFDC is prepared for DS/HD 1.44 Mbytes 3.5 inch drives but the software is not ready. I have not seen any information about how the allocation units will be organized. I suspect that it will have an allocation unit of 4 sectors. This would mean that the shortest file is 8 sectors long.

HARD DISK

My 20 Mbytes hard disk has an allocation unit of 2 sectors so the files will have two sectors for file header and an even number of data sectors. The shortest file will be 4 sectors. Both CALL DIR and DM V will show 3 sectors for the shortest file in this case. This is rather strange because a 20 Mbytes hard disk is similar to a DS/QD 720 kbytes disk. A hard disk can have maximum >FFFF allocation units and a 20 Mbytes hard disk uses >99C0.

A 40 Mbytes hard disk has 4 sectors per allocation so the smallest file will be 8 sectors. Myarc DM V will only show 4 sectors file header and used data sectors. The unused but occupied data sectors will not be shown so the disk manager will show 5 sectors for the smallest file that occupies 8 sectors.

I have not seen any information about an 80 Mbytes hard disk but I think that it will have 8 sectors per allocation unit. The smallest file will be 16 sectors which will be shown as 9 sectors by the disk manager.

continued on page 22

Writing in Machine Code

by J.E. Banfield

Discussion on the MacroT De-Compiler
(Refer TIsHUG News Digest, Vol. 11, No. 3, p 14)

If only an S or s type address is specified, the result of an arithmetic or logical operation goes to that address.

Logical Operations for the TMS9900

In logical operations, only the corresponding bits of the destination and source operands are considered; there are no carries. The following table is adapted from the Digital Corporation PDP10 System Reference Manual and the Fairchild 74181 data sheets. In this table, D and S refer to words and bytes at the corresponding address.

9900 Opcode	Logical expression 74181	MacroT Mnemonic	Result
Destination data			0 0 1 1
Source data			0 1 0 1
04C	Logic 0	SETZS	0 0 0 0
024	D.S	ANDI	0 0 0 1
4, 5	D.S	ANDC, ANDCB	0 0 1 0
28	D + S	XOR	0 1 1 0
E, F	D + S	IOR, IORB	0 1 1 1
054	S	SETC	1 0 1 0
070	Logic 1	SETOS	1 1 1 1

SETZS	Sets the source to zero.
ANDI	AND immediate, sets the source bits to 1 if both source and immediate operand bits are both 1, otherwise sets the corresponding bit to 0.
ANDC	AND the destination bits with the complement of the corresponding source bits; namely, if the source bit is 0 and the destination bit is 1, the result bit is 1; otherwise the result bit is 0.
XOR	Exclusive OR sets the destination to 0 if both the source and destination bits are the same (either both 1 or both 0); otherwise the destination bit is 1.
IOR	Inclusive OR sets the destination bit to 1 if either or both of the source and destination bits are 1; otherwise set it to 0.
SETCS	Set the source to the complement of itself: namely, if the bit is a 1, change it to 0; if it is a 0 then change it to 1.
SETOS	Set the source bits to 1's.

Status bits in the 9900 processor are set according to the results; see the TMS9900 Microprocessor Data Manual.

Arithmetic Operations

Additions

In these operations, carries between each binary digit are executed.

A	ADD	add words
B	ADDB	add byte
058	AOS	add one to the source data
05C	ATS	add two to the source data

For example:

```
Destination    0  0  1  1
Source         0  1  0  1
                  1  1
----------------------------
     ADD       1  0  0  0

Source         0  0  1  1
 one           0  0  0  1
                     1  1
----------------------------
     AOS       0  1  0  0
```

Subtractions and equivalent operations

Most processors, probably also the TMS9900, perform subtractions by adding the 2's complement of the (in this case source) operand to the destination operand. Thus if the source contains the hexadecimal number 5:

```
                                  0  1  0  1
     the complement is            1  0  1  0
     and add 1 to the complement  1  0  1  1
                                  ----------
                                  1  0  1  1 = B
```

and B is the code for -5 on a 4 bit computer.

060 SOS Subtract 1 from source is equivalent to the addition of minus 1 to the source data:

```
Source data  0 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1
minus 1      1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
           1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
             -------------------------------
             0 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0
```

carry out is true.

In practice, the complement of the source data is added to the destination data with an input carry (add one) true for SUB and SUBB.

Care is needed to interpret the result of ST3 (carry) and ST4 (overflow) in these operations. I find it best to try out test calculations and use instructions 19, SKIPNO (skip if no overflow), 18, SKIPC (skip on carry) or 17, SKIPNC (skip on no carry) to test the outcome. The alternative is to read the manual and attempt to interpret it; quite a job.

Thus the opcodes involving subtraction are:

6	SUB	subtract the source word from the destination word
7	SUBB	subtract the source byte from the destination byte
060	SOS	subtract one from the source data
064	STS	subtract two from the source data
050	MOVNS	move the negative of the source data to the source address (i.e. subtract the source data from zero)
074	MOVMS	move the magnitude of the source data to the source

This last instruction does a negate (or subtraction from zero) if the original data was negative. o

continued from page 21

A hard disk with more than 7 heads will not work with Myarc HFDC according to Richard Twyning, EAR july 90. There are only 3 address lines for head select. Small hard disks will use pin 2 for write precompensation, medium size will not use it and large size will use it as the most significant bit for head select. Most HFDCs cannot use the third address contact for WDS3 according to Asgard Reflections vol 2, no 3.

REFERENCES

MG Advanced Diagnostics
Asgard Hard Master
Jan Alexandersson: HFDC review
Jan Alexandersson: Sector Editor review o

continued from page 4

A minor software bug caused an undesired change to the status byte for two of the single day entrants. This problem was discovered and fixed on the third day of the event.

Over the period of the week long event, the TI99/4A computers handled all the event data. At the end of the week, a 52 page book of data printouts and ride reports was produced and one Bubble Jet printer ink cartridge was emptied. The photocopying task for the 170 copies of the book took 11 hours. The 1992 SHAHZADA officially finished at 5pm on the Friday and the results books were ready before the 10am presentation on the next day.

The TI99/4A is a very small computer by comparison to today's personal computers, but it is still very capable of producing the goods for many applications. If anyone would like to have a look at the results book or see some photos from the event, ask me at the October or November meetings. o

Regional Group Reports

Meeting Summary For OCTOBER

Banana Coast	11/10/92	Sawtell
Central Coast	10/10/92	Saratoga
Glebe	08/10/92	Glebe
Hunter Valley	10/10/92	
Illawarra	12/10/92	Keiraville
Liverpool	09/10/92	
Northern Suburbs	22/10/92	
Sutherland	16/10/92	Jannali

BANANA COAST Regional Group
(Coffs Harbour Environs)

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

CENTRAL COAST Regional Group

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

GLEBE Regional Group

Regular meetings are normally on the Thursday evening following the first Saturday of the month, at 8pm at 43 Boyce Street, Glebe. Contact Mike Slattery, (02) 692 8162.

HUNTER VALLEY Regional Group

The meetings are usually held on the second Saturday of each month at members homes starting at 3:15 pm. Check the location with Geoff Phillips on (049) 428 176. Note that after 9:00 pm this number is used for the ZZAP BBS which includes TI-99 information. Geoff.

ILLAWARRA Regional Group

Regular meetings are normally held on the second Monday of each month after the TIsHUG Sydney meeting, except January, at 7.30pm, at the home of Geoff & Heather Trott, 20 Robsons Road, Keiraville. A variety of activities accompany our meetings, including Word Processing, Spreadsheets and hardware repairs. Contact Lou Amadio on (042) 28 4906 for more information.

LIVERPOOL Regional Group

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

NORTHERN SUBURBS Regional Group

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

Dick Warburton.

SUTHERLAND Regional Group

The winter weather kept a few members away for the August regional meeting, but a good time was had by those in attendance. Most of our time was spent in road testing an old TI program called Mailing List. This was done minus the reference manual.

It is an Extended Basic program which was written with two print options, RS232 and the TI thermal printer. So with a few minor changes to the code and a bit of fiddling with the tab settings, everything worked fine.

Regular meetings are held on the third Friday of each month at the home of Peter Young, 51 Jannali Avenue, Jannali at 7.30pm. Peter Young .CE2

TIsHUG in Sydney

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

OCTOBER MEETING - 3rd OCTOBER

This will be a BUY, SWAP and SELL Day, the last one of the year. Bring along your little-used or unused gear and you might be surprised what somebody else is looking for. I (Bob), for one, am looking for a reliable 80-column card.

****

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

November	11 October
December	15 November

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

****

TIsHUG Meetings for Sydney

October

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

November

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

December

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

TI-Faire

1. Volunteers are still required to assist with the manning of stands that are understaffed.

2. Twenty (20) old tablecloths or bedsheets are required to cover tables for TI Faire stands. All contributions will be gratefully received at the October/November TIsHUG meetings.

3. Can any member loan items for display at the "TI UNIQUE ITEM" stand?

4. Contributions for the "TI SALE OF SURPLUS HARDWARE/SOFTWARE" stand are required.

5. PETER SCHUBERT requires the loan of a MINI RAMDISK for his display of "AUSTRALIAN DESIGNED HARDWARE" at the TI FAIRE. Can anyone help?

PLEASE CONTACT IAN MULLINS 871-1514

TIsHUG "T" Shirts for Sale

TIsHUG "T" SHIRTS are now on sale at the "SHOP". They range in size from 14-22 and are BONDS RIBBED NECK AND SLEEVES in white with black logos front and back, 65% polyester and 35% cotton. PRICE $11. A BARGAIN!