TIsHUG NEWS DIGEST

Focusing on the TI99/4A Home Computer

Volume 11, Number 6 July, 1992

Registered by Australia Post - Publication No. NBH5933

Memo

To: All TI99/4A Users

From: TIsHUG (Australia) Limited

Subject: TI-Faire in Sydney, Australia.
On 28th and 29th November 1992,
Now only 5 months away!

The last one in Australia this century!

Sydney, New South Wales, Australia $3

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

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

TIsHUG News Digest

July 1992

All correspondence to:

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

The Board

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

Sub-committees

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

Regional Group Contacts

Central Coast
Russell Welham (043) 92 4000
Coffs Harbour
Kevin Cox (066) 53 2649
Glebe
Mike Slattery (02) 692 8162
Hunter Valley
Geoff Phillips (049) 42 8176
Illawarra
Geoff Trott (042) 29 6629
Liverpool
Larry Saunders (02) 644 7377
Northern Suburbs
Dennis Norman (02) 452 3920
Sutherland
Peter Young (02) 528 8775

Membership and Subscriptions

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

TIsHUG Sydney Meeting

The June Meeting will start at 2.00 pm on 4th of July at Ryde Infant School, Tucker Street, Ryde. The Assembly Class will run from 10.00 am to 1.00 pm for those wishing to attend.

Printed by
The University of Wollongong
Printery Services.

TIsHUG News Digest ISSN 0819-1984

Index

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

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

Editor's Comment

by Bob Relyea

I hope you are as encouraged each month as I am by the receiving of our magazine. According to Jim Peterson, it is the best one in the TI world. Even though I edit most of it, I enjoy reading through it and am always consulting back copies. I am convinced that the magazine is what has held us together for so long and I am loking forward to many more years of interesting computing and fellowship with out great group of nice guys. See ya at the meeting. o

Co-ordinator's Report

by Dick Warburton

The year is slipping by fast. It seems amazing to me that it is already June. We have only another 5 months to go to our TI Faire. It will be upon us before we know it. Things seem to be going according to plan. We have established a friendly rapport with the big club located next door to Ashfield High School. They seem very willing to help us make our function a success. We will be able to use their trestles and their club facilities. Transport and parking should be no problem. At this stage the Faire looks as if it will be really successful. Members of the Faire Committee are busy doing the tasks needed to make it a success. As the time gets nearer, we will need other helpers to give us a hand in a number of ways. If you can spare some time to help out give your name to Ian Mullins, who will certainly find you a job to do.

As we approach the time, we will need to have visitors from interstate or overseas billetted. This is your opportunity to get to know other TI99/4A users. We will need some volunteers to give us a hand with publicity. I would like some people to be willing to ring up talkback radio programs to let people know about the Faire. Others might place posters in strategic locations, or put notices on boards. We will need articles written for magazines or newspapers etc. If you can think of a way to help our Faire be more successful, let our committee know. You might be sitting on a really great idea which could help us all.

If you want to display your system or some aspect of the TI99/4A, check with Ian to see if there is space still left. Even if you do not bring your own equipment, you can arrange to help out someone else. We need volunteers for all types of jobs at the Faire; people to give information, people to show others around, volunteers to help with security, arranging equipment, setting up tables, catering. There will be so many things to do on the weekend.

It would seem sensible to offer any out of town visitors hospitality and accommodation. Surely we can billet most of our TI99/4A community if we try. We could have a raffle and draw the lucky selections out of a hat. Obviously it is difficult for those users who travel from the South Coast or Newcastle to get home and come back the next day. Who knows, you might get lucky and bring someone home who can solve all your TI99/4A problems. Seriously though, it can be a time of real sharing and getting to know other members better.

Well this has been a good week for me with the TI99/4A. I have done things which have been waiting for a long time. My wife has a 2400 baud modem, which I borrow from time to time for the bulletin board. It is incredible to download programs so fast after my trusty 300 baud machine. However it has not been working for months. I have been really frustrated trying to get on the BBS and watching the screen fill up with garbage while the modem drops out. This week I fixed it. I still do not know how but it is working. Fantastic.

I repaired two power supplies and made up an RS232 cable to connect my TI99/4A to other computers. I used a cable 20 metres long. Thanks to the kindness and thoughtfulness of John Ryan, who mailed me a program for the IBM, I can now download programs and text files successfully from my TI99/4A to my work machine. This means simply that I can continue to use my TI99/4A indefinitely for all basic work, download any difficult text files and work them over with Word Perfect if necessary. It makes my life so much easier, as I work in three locations and use a TI99/4A in two of them. I can also transfer useful simple BASIC programs and run them as well, if I need to do so. The cable is simple to make if anyone else out there is interested.

When the 80 column card comes, I will be able to set out text files more easily for transfer. I do most of my work on the TI99/4A. It meets my computing needs most of the time. My main problem lies in the fact that my work place uses IBM machines. Up till now, I have not been able to talk to them. As a result I kept different files on different systems. Now I can easily transfer what I want to the IBM system. Strangely enough, I am using my TI99/4A more and more at work. I use PRBase and Funnelwriter constantly. I still enjoy TI99/4A games, perhaps because I am basically simple. Most TI99/4A games are too hard for me. I cannot even win regularly with Connect 4. What hope would I have with a smarter machine? If I keep going, I might even become proficient on this machine. Well I had better go now, I am getting tired. I will see you at the next meeting. o

Secretary's Notebook

by Terry Phillips

The first of the winter meetings was reasonably well attended, bearing in mind that it was a cool day plus the start of the Queen's Birthday long weekend. I counted 35 members when it came to voting on the item on the agenda for the special general meeting, but a few more drifted in later in the afternoon.

The resolution to amend Clause 4 of the Club's Memorandum of Association was unanimously passed and you should now all make a pen amendment to that Clause in line with the full amendment published in the May 1992 newsdigest.

The state of membership numbers is improving with a few late renewals rolling in. There are now 7 members in arrears back to December, and I don't think now that these people will renew. 24 memberships remain to be renewed from those that expired at the end of April and I am confident that at least some of these will renew, while there are 6 memberships that expired at the end of May. Again I am confident that most of them will renew. So at the present time we still have 135 members which is more than enough to ensure a good year ahead.

New members to give a big welcome to this month are:

Mark Thornton	Mona Vale
Joe Crow	Glenfield
Tim Oakes	Beecroft

Since becoming an incorporated company it may be of interest to you to know that the club has attracted 136 new members. Sadly not all have stayed for longer than 12 months.

Speaking of departed members I was saddened to receive a letter from the Victorian State Trustees Office advising that member Eric Whelan has passed away. Eric and I corresponded frequently over the years and although he only had a basic system he was keen to provide some articles for the newsdigest.

Also in the mail, advice from Fred Cugley of Adelaide that ATTIC - the Adelaide Group - had disbanded. By my reckoning this now leaves this group, Brisbane and Perth the only operating clubs. I am not too sure about Melbourne as I have heard nothing from them in some time. Perth also may be on the way out as I have not received a newsdigest from them now for some months.

I do not know if there is much truth in this but I have just read an article that suggests supplies of floppy disks are in short supply and as a consequence it appears that prices are set to rise. It seems that IBM purchased large quantities for their bundled software and that this is creating a world wide shortage. If this be the case then it would seem prudent to get in and buy up what you can.

That's it for this month. See you at the July meeting.

o

TIsHUG Shop with Percy Harrison

Hi, this month I do not have much to write about except to advise that I still have a few coloured monitors available at $110.00 each some of which are swivel base and two are of the fixed base type.

Also I would like to thank Frank from Bayswater WA and Jim from Maryborough Qld for their kind comments about the shop and the shop articles. It is indeed gratifying to know that someone out there reads my small contribution each month.

As I have received three RAMdisks back in the last month that were either not working or had "crashed", I thought I might devote a few paragraphs writing about the RAMdisk in order to enlighten current owners and future purchasers of the RAMdisk PCB and Kit about some of the problems that they may encounter with them. Firstly I would like to make it quite clear that I am no expert on this subject nor am I an electronics "wizard", all my experience has come from building four cards ranging from 256K using 8K chips up to 1Meg using 32K chips. Fortunately I managed to complete all four without experiencing any problems whatsoever, except for a short caused by the battery holder lead touching a plated through hole which took me many hours to find and a few seconds to fix, so I did not get the opportunity to do any problem solving on these as there were no problems.

Since completing these RAMdisks I have had considerable experience with problem solving on about half a dozen cards built by other members. In the main the major problems have been caused by one of the following:

1. Dry joints, mainly due to soldering onto dirty or greasy component leads or socket pins. When handling components to be soldered try not to touch the areas that have to be soldered and take a little extra time to inspect the component for dirt or greasy residue before inserting in the board this could save you a lot of heartache and frustration when you get to the stage of "firing-up" your card.

2. Shorts caused by too much solder and/or slivers of hot solder bridging across tracks. Use solder sparingly, just enough to make a good connection. I once had a built up board returned to me which had enough solder on it to sink a ship and yet I found that there were 11 dry joints on the card simply because the component leads had been dirty and greasy. No amount of solder would have fixed the problem, the components had to be removed from the board and scraped clean before reinserting. This board, by the way, also had a short between two tracks caused by a very fine sliver of hot solder; so fine in fact that I could not see it with my naked eye.

3. Jumper wires connected to the wrong pins. This is very easy to do when you are working on the back side of the printed circuit board when the diagram shows the top side or when there are a number of connections going onto the one chip. Always take the time to double check each connection before making the solder joint and if still not sure check it a third time. Remember that IC's and many other components are very susceptible to heat and by getting it right the first time you reduce the risk of destroying the component by overheating.

4. Lastly, and by far the hardest to detect is a faulty IC or component. In the main I have found that this is the least likely cause of failure providing adequate care has been taken to ensure that overheating has not occurred during soldering and for this reason I always use sockets for IC's so that they (the IC's) do not have to be soldered in. Another benefit of using sockets is that it makes it very simple to test an IC by unplugging it and inserting another. The last board I had returned to me because it would not fire up had a faulty IN34A diode which took me two hours to find, because I looked for all the other likely problems listed above first, and about 10 minutes to fix.

All the above problems were on printed circuit boards that had never "fired-up" on completion by their owner/constructor, other boards returned to me were because of crashes or lock-ups during operation. These boards were checked out and found to have no construction faults at all. I believe that all our RAMdisks are susceptible to crashing and/or lock-ups, mine crashed twice last week and again today. There does not seem to be any pattern to this type of problem, nor do I know why it happens except at times when I start-up or close-down my system in the wrong sequence. On start-up you should always turn your PE Box on first then your other Peripherals and your Console last, the only exception to this is the printer which can be turned on or off at any time without affecting the RAMdisk. When shutting-down your system always turn the console off first and then the PE Box and other peripherals. I am afraid that RAMdisk crashes are something that we will have to live with unless it is happening several times a week in which case there may be a defect in the Card and it should be looked at.

Commercial Software

Artoons SSSD	$12
BABA Brewery Beer Labels SSSD	$10
Bride of Disk of Dinosaurs SSSD	$14
Character Set & Graphic Design Cataloguer SSSD	$6
Character Set & Graphic Design I SSSD	$12
Character Set & Graphic Design II SSSD	$10
Character Set & Graphic Design III SSSD	$14
Disk Utilities (Memorial Edition) DSSD	$11
Disk Utilities (Memorial Edition) SSSD	$12
Disk of Dinosaurs SSSD	$10
Disk of Horrors SSSD	$14
Disk of Pyrates SSSD	$12
Display Master SSSD	$15
Edu-pak Module + Book	$25
FilmLib Vers 3.0 (TI-Base) SSSD	$8
Genial Traveler SSSD	$6
GIF-Mania SSSD	$15
Legends (2 Disk Set) SSSD	$30
McPaint (5 Disk Set) DSDD	$10
McPaint (10 Disk Set) DSSD	$20
Microdex 1 SSSD	$16
Microdex II SSSD	$11
Nuts and Bolts #1 DSSD	$6
Nuts and Bolts #1 SSSD	$7
Page Pro 99 version 1.6 SSSD	$28
Page Pro Utilities SSSD	$17
Page Pro Applications #1 SSSD	$2
Page Pro Line Fonts SSSD	$9
Page Pro Medical Clipart DSDD	$10
Page Pro Medical Clipart DSSD	$13
Page Pro Templates Vol1 SSSD	$8
Page Pro Templates Vol3 SSSD	$8
Picasso Publisher Version 2.0 SSSD	$14
Picasso Publisher Support Disk SSSD	$6
Picasso Applications Disk DSSD	$2
Pix Pro SSSD	$22
Rockrunner SSSD	$15
Screen Preview SSSD	$20
Son of Disk of Dinosaurs SSSD	$12
Smart Connect SSSD	$15
Spell It! (DSDD version)	$24
Spell It! (SSSD version)	$27
Star Trek (Calender) DSDD	$14
The Missing Link Companion Disk SSSD	$2
The Ring Companion SSSD	$12
TIA Fonts and Borders I SSSD	$8
TIA Fonts and Borders II SSSD	$8
TIA Fonts and Borders III SSSD	$10
TIA Fonts and Borders IV SSSD	$8
TI Casino SSSD	$16
TI Sort SSSD	$15
Tris Module	$25
Word Processor Harrison Software SSSD	$10
X Basher SSSD	$15
XB : Bug SSSD	$22

continued on page 20

TIsHUG Software Column

by Rolf Schreiber

Before I describe the software for this month, I would like to correct an error that crept into my column last month. I inadervently called two disks by the wrong disk number in my description of their contents. Disk A90 should have been called A490, while Disk A91 should have been A491. I hope that this mistake did not cause anyone any problems whilst ordering software from Percy.

I would like to take this opportunity to pass on some information on how to avoid lock-ups with a Horizon RAMdisk on powerup. I always disable the autobooting feature before I switch the computer off, ie execute CALL AF before you switch off the power. I also switch off the computer before I switch off the PE Box. By doing these two simple things, I have reduced the number of lock-ups to perhaps less than once a year.

Software releases for July

DISK A119 is JPGRAPHICS, a graphics drawing program written by a Canadian programmer called J.P. Morin. What makes this program different from some other drawing packages is the demo option which demonstrates all the main features. For those of you following the Forth tutorials by Earl Raguse every month, this program may be of special interest, since it was written in Forth itself. This version comes with an Extended BASIC loader and also has a documentation file on the disk.

A119 Diskname: JPGRAPHICS Format: DSSD

Filename	Size	Type / Length	
FORTHSAVE	39	PROGRAM	9512
JPDOCS	93	DIS/VAR	80
JPGRAPHICS	310	DIS/FIX	128
LOAD	9	PROGRAM	2018
SOURCEJP	267	DIS/FIX	128

DISK A411 is a games disk containing two arcade games. Miner 2049er is a Donkey Kong "avoid obstacles" type of game and should amuse some of our younger gamesters. Espial is a classic "shoot-em-up" game where fast reflexes are essential. The games autoload from a menu in Extended BASIC, but there are no instructions included on the disk.

A411 Diskname: SRTY Format: SSSD

Filename	Size	Type / Length	
ESPI	33	PROGRAM	8192
ESPJ	33	PROGRAM	8188
LOAD	9	PROGRAM	1964
MINE	33	PROGRAM	8192
MINF	33	PROGRAM	8192
MING	2	PROGRAM	256

DISK A457 is another in the series of music disks from Harrison Software. This disk is called Anna Magdalena's Notebook by Johann Sebastian Bach. The program loads from Extended BASIC and you have the choice of 20 pieces of music. You can select to play 8 menuets, 6 polonaises, 1 rondo, 3 marches, 1 musette or 1 solo cembalo. Alternatively to can play all the selections in sequence.

A457 Diskname: ANMAG Format: SSSD

Filename	Size	Type / Length	
ANMAG#1	10	PROGRAM	2254
ANMAG#10	9	PROGRAM	1855
ANMAG#11	10	PROGRAM	2229
ANMAG#12	11	PROGRAM	2532
ANMAG#13	12	PROGRAM	2739
ANMAG#14	10	PROGRAM	2234
ANMAG#15	8	PROGRAM	1679
ANMAG#16	15	PROGRAM	3472
ANMAG#17	10	PROGRAM	2145
ANMAG#18	26	PROGRAM	6177
ANMAG#19	12	PROGRAM	2654
ANMAG#2	10	PROGRAM	2115
ANMAG#20	12	PROGRAM	2766
ANMAG#3	9	PROGRAM	1986
ANMAG#4	11	PROGRAM	2362
ANMAG#5	18	PROGRAM	4103
ANMAG#6	10	PROGRAM	2078
ANMAG#7	25	PROGRAM	5960
ANMAG#8	8	PROGRAM	1764
ANMAG#9	10	PROGRAM	2251
ANNMAG	5	DIS/VAR	80
LOAD	10	PROGRAM	2204
PRINTMAG	2	PROGRAM	166

DISK A492 is Microkey V1.1, a keyboard macro from Tarik Isani, who also wrote Nibbler, the disk copying utility. This utility allows you to redefine the ten control keys, <CTRL 0> to <CTRL 9>, so that pressing the CTRL key in combination with one of the numeric keys is equivalent to having typed in the sequence that the key was redefined to. For example, <CTRL 1> can be redefined to display "LIST PIO", so that every time you press <CTRL 1> it is as if you had typed in "LIST PIO" from the keyboard and pressed <enter>. Pressing <CTRL 0> at any time displays the current list of redefined key functions. The program comes with instructions which can be viewed on the screen if you hold down the "?" key (FCTN I) at startup. After the program loads, it is memory resident and can be activated at any time by pressing CTRL and the appropriate numeric key.

A492 Diskname: MICROKEY Format: SSSD

Filename	Size	Type / Length	
DELETE1	9	PROGRAM	1821
HELP	16	PROGRAM	3840
K1	3	PROGRAM	332
K2	3	PROGRAM	332
LOAD	2	PROGRAM	182
MICROKEY	26	DIS/FIX	80
READTHIS	12	DIS/VAR	80
TEXT	25	DIS/VAR	80
TEXTMICRO	27	DIS/VAR	80

DISK A493 is a disk manager program from Mike Dodd called Disk Manager 99 V2.1. The program autoloads from Extended BASIC and becomes memory resident. If you press <CTRL D> at any time, the disk manager is activated and displays the following menu selections:

1 File Commands
2 Catalog Disk
3 Rename Disk
4 Initialize Disk
5 Test Disk
6 Toggle Printer
7 Return to BASIC

The program comes with the source code, the assembled object file and a documentation file and is well worth getting if you are interested in assembly language programming and want to see how to program disk access routines. The disk was originally part of the Boston Computer Society's software library.

A493 Diskname: BCS48 Format: SSSD

Filename	Size	Type / Length	
-README	5	DIS/VAR	80
DM99-1	103	DIS/VAR	80
DM99-2	57	DIS/VAR	80
DM99/DOC	54	DIS/VAR	80
LOAD	28	PROGRAM	6835
PRINTDOCS	4	PROGRAM	538
XBDM99	19	DIS/VAR	80
XBDM99/O	84	DIS/FIX	80

continued on page 25

Techo Time

by Geoff Trott

RAMdisks

Dick was telling me at the last meeting that he hears from a lot of people who are having problems losing their ROS from their RAMdisks. I guess this is why so many chose to use an EPROM for their ROS. I tried an EPROM on a RAMdisk for the MiniPE system and found that it was worse than losing the ROS in that if anything went wrong the whole RAMdisk had to be re-loaded. The ROS in the EPROM is also rather dated so I do not recommend that for anyone. The best solution is to try to stop the loss of data somehow. For this there are some simple things you can do to minimise the possible problems and some more complicated operations in the difficult cases. First I would like to talk about why problems could arise. To do this I need to talk about the memory chips themselves.

The memory chips are static RAM chips which store the data in circuits called flip-flops. As long as power is supplied to these circuits the data remains in the memory. To ensure this we use batteries to provide the voltage when the power is turned off. As well as providing power to the chip, it must be disconnected from the rest of the external circuitry so that current is not supplied from the static RAM chip to all the other circuits to which it is connected. This is done with the Chip Enable (CE) input(s). In the case of the 64K static RAM (which holds most of the ROS) there are two CE signals, one of which is asserted when high and the other is asserted when low. The asserted low CE input (CE(L) pin 20) is used in the normal operation of the chip and is driven from a 74LS02 (the first version) or a 74LS156 (the second version). The CE(H) input (pin 26) is used to disable the chip on power loss and is connected to the power supply through the power on LED, which has a voltage drop across it. This voltage drop means that the power supply must be almost 5 volts before the chip will become active and a very small drop in voltage will cause the chip to become disabled. The importance of this can be seen if we consider what happens on turning the power on and turning it off.

When power is turned on, the capacitors in the power supply start charging up and all the ICs start to work. It is not easy to predict what the ICs will do as they start to work and one can imagine that almost anything could happen for a very brief instant as the voltage passes through the value where the ICs start to work. They may all have their own different voltages when they do unpredictable things. It is during that time that we want the static RAM chips to be disabled so that their contents do not change. By putting the LED in series with the CE(H) input, the RAM is not enabled until the voltage is well past the value when all the other ICs are starting up and since the voltage increases relatively slowly with time, this gives time for all the unpredictable behaviour to cease before the RAM is enabled and would react to it.

When power is turned off, the voltage on the power supply decays as all the capacitors discharge. This causes the same unpredictable behaviour as each of the ICs ceases to work correctly. The LED in series with CE(H) ensures that the CE(H) input becomes not asserted and turns the RAM off well before the unpredictable behaviour starts. This all sounds great and there should be no problems. This is true on turn off, although adding larger capacitors to the power supply to slow down the rate of change of voltage would help matters. However on turn on there could be other problems related to how the ac is converted into dc. This is done with the rectifier which changes the 50 Hz ac waveform into positive pulses every 10 milliseconds. These pulses are used to charge capacitors to produce a relatively constant dc voltage which is then regulated down to 5 volts.

Consider the following scenario. On turning on the power, the first pulse causes the capacitor to charge up to 6.5 volts which then discharges to 5.8 volts before the next pulse arrives which charges the capacitor up to 8 volts followed by the third pulse which charges it up to 9 volts, as does all subsequent pulses. Assuming a discharge of .7 volts (ripple voltage) between each pulse, this would then give a reasonable raw dc supply for the RAMdisk. At start-up, this may cause the voltage out of the regulator to go up to almost 5 volts and turn everything on for a very short time and then to drop back and turn everything off before coming up to turn everything on again. This sort of operation may be the cause of some problems. It certainly was in the MiniPE RAMdisk but as the PEBox has a much larger capacity power supply, it would be less of a problem there.

The data in the RAM chips will only be changed if the CE is asserted and the WE (write enable) is also asserted. The asserted levels at the inputs for these two signals are both low which means that when the power is off they will be at their working levels. Consider the case of the power to the RAMdisk being present but the console being off. In this case all the signals to the RAMdisk will be low, including the WE signal. In this case it is important that the CE is high. This is determined by the contents of the 74LS259 chip which is an addressable latch. This could have random data stored in it on power-up, so there is a circuit consisting of a resistor, capacitor and diode to reset the contents of the latch to all zeros. This reset circuit is designed to provide a reset pulse which is longer than all the turn on uncertainties. Since this could be one or two pulses in the rectifier, this needs to be at least 20 milliseconds long. This is the capacitor labelled C1 in the schematic and this should be 10uF rather than 0.1uF. If this reset works then the RAM chip CE will be high and the chip will not be selected so that the level of the WE line does not matter. However, if strange things happen due to the power going up and down at power-up, there may be further problems here and these could be random in nature.

In this case I have developed a little circuit which cause the WE line to the RAM chips to be disconnected from the WE line coming into the RAMdisk and held high as soon as the voltage into the regulator chip starts to drop and does not connect it back again until some time after the output of the voltage regulator has been restored to 5 volts. This circuit is almost essential to the reliable operation of the MiniPE RAMdisk and has been installed in three of them with excellent results (as far as I know). PEBox RAMdisks do not normally need this final circuit as I think their power supply is better behaved.

If you are having problems with data loss in RAMdisks you need to try and find out exactly where the problem lies and whether a change in operating procedure may help. For example, in what order do you turn on and off your computer. Considering turn on first as that is the most critical time. If you turn on everything at once, there will be a race between the various power supplies and ICs as to when they will all settle into their correct operation. I would suggest that you always turn your PEBox on first as then signals like WE and all the address lines are at a defined level (low) and will not change. Then the console should be turned on. On turn off, I would suggest that you first disable the auto-boot feature of the menu by entering BASIC and doing a CALL AF command. This can also be done in Funnelweb DiskReview. Of course this means that on turn on you will need to either do a CALL AO from BASIC or a CALL MENU to get the menu up. Having disabled the auto-boot, turn off the console followed by the PEBox. You may find that it is not much trouble to leave the auto-boot off all the time and just do a CALL MENU to get the menu up. This is how I run my system and I have very few losses of ROS. In fact, I have lost the ROS in my second RAMdisk (at CRU of 1600) more often recently than that of my primary RAMdisk with all the system files on it.

continued on page 22

TI World News

compiled by Jim Peterson, Tigercub Software, USA

The Long Island UG newsletter for Jan. 1992 reproduces an ad for the ImageWise Serial Video System, in kit form, consisting of the digitizer/transmitter and the receiver/display. It is stated to capture an image in 1/60 of a second from monochrome or colour video cameras, camcorders, VCRs, etc. According to the ad, with additional software it will digitize images for display on IBM PC, Apple Macintosh, Commodore Amiga, Atari ST and other popular computer systems. A handwritten note indicates that it is compatible with the TI-99/4A with 80 column card, or the Geneve 9640. The address is Micromint, Inc., 4 Park St., Vernon CT 06066, phone (203) 875-2751. Price is not mentioned.

The same newsletter contains advertisements for TI-99/4A software to be used with the ImageWise system. These include a $9.95 disk (plus $1.50 S&H) containing the programs GRAB, SHOW and CONVERT (for use with the TI-99/4A; no mention that an 80-column card is required); Imagewise Portrait Print ($4.95 + $1.50 S&H) to print a 17"x22" poster from a digitized file; ImageWise Display Routine ($4.95 +$1.50 S&H) for a TI-99/4A equipped with the Yamaha 9938 VDP, or the Geneve 9640; ImageWise pictures in two volumes; all available from R.F.W. Enterprises, 111 Oakridge Street, Chicopee MA 01020.

And, available from Joseph M. Syzdek, 99 Highland Ave., West Springfield MA 01089-1017 for $14.95 plus $1.50 S&H, is IWD Plus for the Geneve 9640 and Imagewise Video Digitizer, to capture digitized video data over the RS232 port and display the image on a monitor in 256x212 or 512x212 resolution. It can be saved to disk in ImageWise or MyArt format and provides some editing capability.

Harrison Software is now offering a 45-minute stereo cassette of their MIDI- Master music, containing the 20 pieces from Magdalena's Notebook by J.S. Bach and Bach's Italian Concerto. Except for the last, all were produced on a CASIO CT-650 keyboard using a TI-99/4A with MIDI-Master 99. The price is $10 and the address is 5705 40th Place, Hyattsville MD 20781.

Harrison Software has announced that they are removing their assembly music disks from their software catalogue, and placed them in the public domain. User groups may distribute them as they wish. They will also be available from Tigercub Software, 156 Collingwood Ave., Columbus OH 43213 ($1.50 per disk, plus $1.50 shipping and handling if less than 8 disks are ordered).

I have a letter from Francisco T. Molina, a really isolated TI enthusiast in Argentina, who is trying to organize a local user group to be called T.I.G.R.E.S. de Argentina, which stands for Texas Instruments (99/4A) Grupo Recalcitrante y Empernido de Sobrevivientes, meaning "group of everlasting and recalcitrant survivors". He reports that his local mail service is quite unreliable but he has now found a friend in Virginia who apparently makes frequent trips to Argentina and can carry software to him.

For those who like "brain games", Asgard Software (P.O. Box 10306, Rockville MD 20849) has released TI PEI, a mahjongg game by William Reiss, and Classic Checkers by Chris Bobbitt. They load from disk and require Extended Basic and 32k; a Mouse is optional. The price of each is $14.95 plus $3 per order for S&H ($3.50 in Canada).

Rave 99 (112 Rambling Rd, Vernon CT 06066) is offering a new kit form of its Speech Synthesizer Adapter, including all parts and instructions, for $35 plus 5% S&H.

Paul Coleman (3971 S.E. Lincoln, Portland OR 97214), the author of Artist Printshop, has now released Artist Cardshop, a 2-disk package for $20 plus $1.50 S&H. This is an advanced all- assembly program to create greeting cards, using TI-Artist fonts and instances. It is also available at the same price from Comprodine (1949 Evergreen Ave., Fullerton CA 92635).

Don Shorock (P.O. Box 501, Great Bend KS 67530-0501), the author of Air Taxi, has now released Son of Air Taxi, using the same game format but with maps of Europe, Africa, South America, the West Indies, the Far East and Australia. The entire set is available for $10.

As of 1 Feb., the Clearing House BBS (614) 263-3412 held 320 text articles in 7000 sectors archived, which would print out to a very large book of TI-related articles. More articles are being added continually. These files are available for download by any user group (or individual) who becomes an associate member of the Central Ohio Ninety Niners.

According to Barry Traver in Computer Monthly, Mike Wright is compiling an encyclopedia of information on the TI-99/4A, known as "Mike's Cyc". From the looks of the March issue, Computer Monthly seems to be cutting back somewhat on its coverage of the "classic" computers.

Peron Laurent of the FANATI user group in France has released an "American version" (documentation and prompts in English) of his Drawing Master Version 1.3. This is a graphics program with pull-down windows and many advanced features including a method of avoiding "bleeding" when filling areas with colour. Some of the options listed in the windows are not yet in the program, but will be added in future versions. The program has been released as fairware and is available from Tigercub Software.

The TI-PD catalogue of Tigercub Software, including the latest supplement, now lists over 600 disks full of public domain and fairware programs. According to Asgard On-Line, the new Extended Basic from Germany, written by Winfried Winkler, will run your entire library of Extended Basic programs, without modification, up to 50% faster than the original TI version. It has also eliminated bugs that cause those multi-coloured crashes. For the programmer, it offers a fantastic array of really useful new commands including closing all files with one command, variables in GOTO and GOSUB, improved IMAGE and RND, redefining characters up to 159, assigning text to a CTRL key, and many new calls including VPEEK and VPOKE. (but I did not see any mention of a 40-column screen!). Extended Basic III is currently only available on disk, for $39.95, and requires the Mechatronics GRAM-KARTE, but a 64k cartridge version, with additional 16k of RAM and 48k of ROM, is expected to be available by the 3rd quarter of 1992 for $74.95.

Asgard no longer requires the return of the original disk in order to obtain an upgrade. Registered users can simply send a check for the required amount, non-registered owners must also enclose a photocopy of the manual cover as proof of ownership. Registered customers will be notified by mail when a product is upgraded.

Under these terms, GOFER 1.01 plus a new CLIPIX utility is available for $7.50; PIX PRO including CLIPIX for $6. Registered owners of Screen Preview can receive an enhanced version (the bug in using & and @ has been fixed) by returning their program disk or by sending a check for $2.00.

S&T Software Ltd (c/o Tim Tesch, 4346 N. 88th St., Milwaukee WI 53222) is offering the S&T MXT BBS program, which features true 40/80 column and full ANSI/ADM3A support, as well as many other advanced features. The price is $25, or $35 if the source code is wanted. To see it in action, call the Graphics Clipper BBS (414) 284-6108, the NorthSide BBS (414) 444-1309, The Orphanage BBS (918) 288-6708 or the Programmers Lair (918) 836-4325.

According to John Koloen in the February MICROpendium, the Accelerator card is on indefinite

continued on page 20

New Funnelweb Editor

Review by Geoff Trott

I have been sent a beta test version of Tony McGovern's latest update to the editor for testing with a hard disk system. The notes that came with the package are printed elsewhere in this issue, but I thought it would help if I gave some of my thoughts on this latest bit of excellent software for our very useful home computer.

Firstly I must admit that my system is not that typical as I am using an 80 column card (Mechatronics), but with luck and some effort from Garry Bowser more of you may be in the same position. Although there is a 40 column version of the editor also coming it does not have all the facilities of the 80 column version, as it lacks the features requiring the extra memory available with the 80 column cards. Although I also have a hard disk system (Myarc), this does not add as much to the system as having a RAMdisk. I happen to be basically running on a floppy disk and RAMdisk system at the moment as I need to try and recover one of my hard disks which crashed some months ago. (Oh for a bit of free time!) So I am able to test the hard disk facilities of the program but am doing all my work on floppies at the moment.

The TI-Writer package provided a first class word processing package by using the facilities of the formatter. It is true that it is a command type word processor and not a "what you see is what you get" type, but this gives a basic 40 column screen computer the ability to prepare documents to any width with relative ease. I used it for a number of years to prepare the contents of the TND, which was all in electronic form, formatted to fit our requirements, checked for errors and printed on our printer. I also happen to use Microsoft Word on a Macintosh at work so I know about all the bells and whistles that you can get in that environment, but that takes at least 1 Mbyte of memory to run while TI-Writer fits into 32 Kbytes of memory. I still enjoy using the TI99/4A and its Funnelweb environment for word processing at home.

By putting the TI-Writer editor into Funnelweb, Tony McGovern added features like the file name mail box and the ability to change case of letters with simple key strokes. This made the system so much easier to use as it became more integrated and required less typing of file names. He also improved the show directory feature along the way but the basic editor remained mainly untouched. Now he has written his own editor (based on the operation of the TI-Writer editor) which provides many more features as well as all the ones we are used to. In the process he has merged in the Euro-Writer system and expanded it to provide a configurable editor to suit almost any language with reasonable shaped characters. There is also a help facility which can provide screens of help which can also be prepared easily by the user.

Let me try and explain what is now possible and has been improved. Firstly, the editor is entered in the normal way from the menu of Funnelweb. It looks the same as the previous version at this point except for a change in the prompt line above the command line. This contains a few new entries. The new line reads:
Edit,Files,Lines,SearcH,Tabs,View,Help,STore,ReCall,Quit

Edit will return to the edit mode as before, but just pressing <enter> with nothing on the command line will return to edit mode. There are some additional ways to return to edit mode after various commands. Ctrl[1] returns to edit mode with the cursor at the point that it was when you entered command mode. This is useful with the new ability to scroll the screen while in command mode, although I am not sure how it differs from just a blank command line. It does not work after a change command. To return to where you were when you entered command mode after having completed an FS or RS, return to the edit mode and press ctrl[o]. Ctrl[2] returns to edit mode with the cursor at the top line of the current screen.

Pressing F (followed by <enter>) for Files will bring up another menu of commands. The number of these commands has also been increased and they read:
LoadF,SaveF,PrintF,LoadTemp,ShowDirectory,HardDisk,
sDPrinter,Purge
LF, SF and P work as before. Print file (PF) has some enhancements in the way of option codes which are placed in front of the device name, separated by a space. The existing option code is C for stripping out control codes. The new ones are A for appending to an existing file; M to output to a file in display fixed 128 format with MS-DOS end of line separators (<cr><lf>) and ctrl[z] end of file marker; U to output to a file in display fixed 128 format with Unix end of line separators (<lf>) and ctrl[d] end of file marker; P to send printer start up control code sequence (if these have been installed during the configuration of the editor) before sending the text; and Q to send printer reset up control code sequence (if these have been installed during the configuration of the editor) before sending the text. Load temporary file (LT) works the same as LF except that the work file name is not changed in the Funnelweb mail box. This allows other files to be merged and saves re-typing the work file name. Hard disk show directory (HD) allows a path (or disk) name to be entered (instead of just a disk number) which allows sub-directories on hard disks to be used as well as RAMdisk names such as DSKA and names of floppy disks (DSK.NAME. Note that the name must end with a period.). Once the name is entered the disk or sub-directory is catalogued as per normal. It does not give quite the same information as SD about the files but does identify emulate files and sub-directories and allows files to be marked for use and their names put in the mail boxes. Once a path is defined, it can be accessed by specifying 0 when inshow directory. Show-directory printer (DP) allows a different device from the print file (PF) printer to be used for printing the contents of the disk in show directory mode. After defining the device name, it then goes to the show directory function.

Speaking of the show directory function, this has been enhanced in several ways. There are now three files which can be marked for use. The first is the work file which is marked with <space> as before. The second is the temporary file (loaded with LT) which is marked with T. The third is the view file. Show directory now has a view file function like DiskReview (scrolling in both directions) and the viewed part of the file is stored in the VDP memory buffer and becomes marked as the B file. This is probably only an 80 column version feature. The contents of the buffer are able to be viewed inside the editor. Another nice feature is that the number of bytes unused in the editor buffer is displayed so you can see how close you are to filling the editor buffer.

Pressing L (followed by <enter>) for Lines will bring up another menu of commands. The number of commands has also been increased and they read:
Move,Copy,Delete,Show,Mark lines
The first four of these are the same as before while the last enables a line to be marked and then jumped to from anywhere in edit mode with fctn[=]. The use of these commands has been made easier by the ability to scroll the editor buffer while in command mode. This means that to find the line numbers for a move or copy of a block of lines, the lines can be displayed while constructing the command rather than remembering them. Also ctrl[m] will put the line number of the top line on the screen at the cursor position in the command line. So by using the scrolling functions and ctrl[m] it is not necessary to type in any line numbers for these commands. To show a line the same thing can be done although it is not necessary to type the S first as just a number on the command line will do it along with ctrl[2] to go to the line at the top of the screen and an empty line (or ctrl[1]) to return to where you left the edit mode.

The search functions of FS and RS have been improved by allowing the string delimiter character (/ in the old editor) to be any character other than a numeric character (0 to 9). If you wish to change or find a string of characters which include the character

"/", it is now possible to do so. Also at the end of the function there is a sound made and the editor remains in command mode. It is also possible to return to same place in the edit buffer by going to edit mode and pressing ctrl[o].

Tabs now allow up to 9 sets of tabs of which the first three are saved with the file. The particular tab set to be used is selected by number. The editor can cope with tabs from any version of TI-Writer.

View is one of the new functions which allows the contents of the view buffer to be viewed in the editor. The view keys are active for scrolling both ways. This allows viewing without going into show directory.

Help allows up to 4 screens of help to be displayed. These screens can be prepared by the user and there is a program for converting them to the correct format for the editor. As supplied by Tony, the four screens show the full character set, a quick reference for the function keys of the editor, Funnelweb entry points and Assembler language mnemonics and other useful information. The pages of help are paged with the Q and A keys.

Store and recall are very useful commands for parking the entire editor buffer into VDP memory (in less than a second) to allow some other file to be loaded and worked on. When desired the editor buffer can be recalled from the VDP memory and editing resumed. They are only available in the 80 column version because of the extra memory. It would also be nice if the contents of this buffer would remain after leaving the editor to do a format for example and was still there when the editor was re-entered. Tony might like to think about this possiblity as it could speed up preparation of documents greatly.

Quit is the last function on the main menu. This has not been changed but an additional way of leaving has been added. QQ will go directly to Funnelweb with a reminder to save the current work file first if this has not been done. This reminder is issued before a recall or purge also. A very handy warning!

If you have understood all this then you will recognise that this new editor is indeed a great improvement on the previous one in the ease of use and facilities offered, particularly in the 80 column case. I have not yet mentioned the All Characters mode which is part of this editor. When loading the editor, if you press the space bar you will receive another set of menus. The first of these allows you to choose the text editor or the program editor (eqivalent to the two editors in the Funnelweb menu). Having chosen one of these, you are then able to select to use the default 7 bit normal editor; a national 7 bit editor which just has the capability for other languages in the menus and prompts; All Characters which allows for entry of all characters with ASCII codes from 0 to 254; and TI Euro-Writer for files from Europe. For normal use the first option is all that is required. If you want access to IBM graphics for example, you can choose the All Characters mode. The help files tell you what key combinations to press to get all the non-standard characters. There are some special key presses for putting accents on vowels which are listed in the help files but there is one key press which applies to all and which I should mention. You can freeze the bottom of the screen below the line on which the cursor is placed with ctrl[f]. This allows some text to be kept on the screen for a reminder while you type in anywhere else in the document.

Well I see that I have only 11,780 bytes left in the buffer so I will conclude by saying that I think that this editor will provide serious users with much joy and save them many frustrations and much time. I would like to thank Tony McGovern for his efforts on our behalf and look forward to the release of version 5 of Funnelweb. O

Funnelweb Editor ver 5.00

Notes on beta release, by Tony McGovern

This document refers to the second set of files issued for test and comment only (no distribution of the Editor files themselves at all please) of the total rewrite of the Funnelweb editor. It retains as much compatibility with the original TI-Writer as can be managed. It will most likely form the major component of the version 5.00 revision of the Funnelweb system and is labelled as such, but the current files execute from version 4.40 in the development phase. This file is being written with the new editor.

These beta phase documents will describe the current form of the program and ignore any temporary alpha phase expedients. Since the alpha issue a number of bugs have come to light and been squashed. Reactions received have generally welcomed the TI Euro-Writer capability, but seem even more interested in the all-characters mode, so the interface has been updated to reflect this. Also all regular functions have now been squeezed into 128 Kbyte of VDP memory so that users of unmodified Geneve 9640 machines are not disadvantaged and no Geneve-hostile code is knowingly included. As in version 4.40, the 80 column version is dual mode and will serve as a superior 40 column editor. There is no guarantee that the features included in this issue will not change before final release.

(1) Version 4.40 features removed from version 5.00

Several features of the version 4.40 80 column Editor have been eliminated and mostly replaced by superior facilities.

(a) Changing/selecting editor type on loading to override the load-path setting was formerly done by pressing <W> or <P> as the program started to execute. Now <space> brings up a multi-part selection screen.

(b) SwapTabs has been replaced by a more comprehensive Tab setting process.

(c) The use of <W> in SD for retaining pages of <V>iew for ready reference has been dropped.

(d) Keys <ctrl-Q> and <ctrl-A> now do not force exit from command mode.

(e) RecoverEdit has been omitted. The original function of this had mostly disappeared in Funnelweb as it overwrites the text buffer on exit, unlike the original module based TI-Writer and the only remaining function was after Purge.

(2) Euro-Writer and All-Chars Capability

Texas Instruments released in Europe in 1983 (in Germany at least) a multilingual Version 2.0 of TI-Writer which supported the range of languages implicit in the TI-Writer module selection screen. We will refer to it here as Euro-Writer. Unfortunately Euro-Writer writes Tab records to file which are fatally incompatible with the original USA issue of TI-Writer. It also had a whole range of auxiliary text and character files and a new Formatter with special transliteration files for the new characters.

The new Funnelweb Editor supports both the original TI-Writer and Euro-Writer with selection at load time, either preconfigured (see later) or from the selection screen. A new mode allows entry of all characters up to ASCII value 254 (>FE>). The file loader handles all existing tab record formats (TI-Writer, Euro-Writer, Funnelweb) transparently. The user selection screen is brought up by pressing <space> as the program starts. First choice is between Word Processor and Program Editor. The next choice is from 4 options.

<1> Default 7-bit, in which no further character or command files are loaded.

<2> National 7-bit, which is standard TI-Writer, but loads national command and character files. This will be useful in languages and applications which can coexist with a modified 7-bit character set.

<3> All Characters, which loads from a different set of character and command files and allows keyboard entry of all characters >00 to >FE, for example the complete IBM character graphics set. All-Chars mode is toggled by <ctrl[,]> and indicated by a hollow arrowhead for position indicator on the bottom ruler line. Currently it is cancelled by going to Command mode, but can be reset there. This mode sets the most significant bit in any character entered. Internal buffer encoding is less efficient (reduced text buffer capacity) for Options 3 and 4 and extended Euro-Writer tab records are written out. It also means that character >FF (<fctn[V]> in this mode) is not available and is replaced by a space in the text buffer. All-Chars in word processing mode writes Euro-Writer type tab records to the disk file.

<4> TI Euro-Writer, which apart from redefined normal characters, allows entry of various modified versions of vowels, using keys <fctn[,]>, <fctn[.]>, <fctn[-]> and <ctrl[-]>. These are encoded as ASCII 128 to 167 (>80 - >A7). LoadFile now has an extensive new routine to examine and validate potential tab records of any variety. The Euro-formatter and transliteration files will be needed to handle these Euro-Writer files correctly in printing if they contain modified vowel characters. It may be that Euro-Writer users will prefer to use All-Chars mode which will mesh more easily with standard printer fonts.

The next option box allows selection (1 to 8) of the various national languages. Option 1, Australia, is the base line option for use of All-chars mode.

These modes use various auxiliary files. Euro-Writer mode loads text/command files F8TX<@ to G>E and the TI Euro-Writer files CHAR<A to G>1. All-Chars loads files F8TX<@ to G>A and CHAR@<1 to 8>. Some CHAR@x files, akin to Code Page 437 (all chars) on PC systems, are included and others may become available with your help. I realize that 8x6 textmode character patterns will not be all that wonderful on screen but it does open a new dimension for TI-Writer editing. Loading an All-chars or Euro-Writer file into the 7-bit Editor modes may corrupt the file as the MSbit is stripped from all characters. If in doubt, load into All-chars or Euro-Writer.

Whatever the character set, the characters >7F (<fctn[V]> Del, normally blank) and >FF are redefined as the solid and hollow bottom line indicator characters. The >1F pattern (the edge character in BASIC) is also always redefined as the solid mid-line for use as a distinctive screen divider in 40 column freeze mode. The cursor of the moment remains as the >1E pattern. One of the Help screens provided allows examination of and ready reference to the current character set.

The editor files must be loaded from Funnelweb under a 2 letter filename for character, language and help files to be found.

(3) New and Updated Editor Command Line Entries

The total rewrite has made it possible to introduce several new command line 2-letter entries in various categories. An important, and the most obvious change to command mode, is that text may be scrolled by line or page using the normal set of up/down scroll control keys. This allows the text to be inspected anywhere during command line entries. The new entries are specified here by their English language version.

<T > -- for Tabs now brings up a second command line which asks TABSETS (1-9)? and indicates the current setting as the default entry. The new editor allows 9 tabsets to be defined, of which the sets 1 to 3 are saved in document tab records. The current tab line is now always written into the ruler line when confirmed by <enter> and also when a new file is loaded and a tab record read in.

<V > -- for View of whatever file is currently in the scrolling view buffer in VRAM. Alternatively this can be reached from SD.

<H > -- for Help mode brings up a series of up to 4 help screens which were loaded initially, with paging between them by <Q>, <A> and exit by <ctrl[C]>. A utility for preparing your own help files is included.

<ST> -- for STore stashes the current work file and all relevant data in VDP memory. This is to permit another file to be loaded, edited and saved. Then you can do --

<RC> -- for ReCall of the STored document from storage and resume editing. This store/recall process is much faster for long files than using SF and LF and does not require any disk file space.

<QQ> -- for immediate Quit back to Funnelweb. The editor maintains a "file-edited" flag and if any text entry has been done since loading or saving the current file, a reminder to save the current work first will be issued. This warning also operates before ReCall and Purge.

<LT> -- for LoadTemporary file. A secondary workfile name is maintained during the current edit session. This may be entered directly, or marked in SD as such. This allows for inserting all or part of external files into the current workfile without disturbing the current workfile name.

<DP> -- for set showDirectoryPrinter name. This allows the device name used by <ctrl[P]> print directory in SD to be preset to something other than the PF name.

<MK> -- for MarK position in file. This sets a marker after line number entry, or else enter this with <ctrl[M]> at the current top line, which may be scrolled to any line in the workfile while still in command mode.

<HD> -- for HardDisk pathname instructs SD to do a directory using the catalog file for the pathname, using an assembly version of the standard BASIC disk catalog program. The pathname is presented for editing first.

< > -- a blank command line. On the main command line this returns to the Edit mode at the original exit point.

<number> -- from the main command line a number acts like a Show lines command. "E" for EoF is not recognized in this direct return as a letter may conflict with other commands.

Some control key presses now have new special functions in command mode and mostly were of no function before.

<ctrl[M]> now writes the current top of page line number at the cursor position on the command line in insert mode. If you must have <cr> on the command line, use <ctrl[8]> or special character mode.

<ctrl[1]> exits from command mode to the departure point from Edit mode.

<ctrl[2]> exits from command mode to the current top of page. It has the same effect as <ctrl[M]> followed by <enter>.

Some enhancements have been made to the Find and Replace string commands. Each now takes up to 3 numbers ahead of the string entry. If there are 2 numbers they are the start column and finish column for the search. For 3 numbers or 1 number the first or only number is the number of match occurrences to skip before stopping. This is similar to the Editor Assembler editor. In case you had not noticed, RS always worked like this and now FS does as well. Also when no more matches are found, BOTH FS and RS give an audible bloop and stop where they are. As a further enhancement any non-numeric character may be used as delimiter, so that /ABC/def/ or -ABC-def- or aABCadefa as RS string entry will all search for string ABC to be replaced with string def.

(4) New Edit Mode Functions

Some minor changes have been made to improve safety in editing. <ctrl[N]> in Edit mode now inserts a New line to avoid unintended deletions when NTSC/PAL toggle was intended. This also matches usage on PCs, as in Borland editors.

<ctrl[F]> now Freezes the Edit screen below the cursor line and normal editing and scrolling take place in the upper part of the screen. Entering command mode or pressing <ctrl[F]> again clears the frozen part to normal. In 80 column mode the color of the frozen part of the screen changes to the secondary color set. In 40 column mode a solid line is drawn across the screen on the line below the cursor and the screen below that remains frozen. Horizontal windowing does not shift the frozen part in 40 column mode.

<fctn[=]> effectively does a Show Line with the currently MarKed line at top of screen. It is reasonably intelligent in the face of changing workfile contents and if confused reverts to line 1.

<ctrl[O]> returns to the original line after some operations such as <fctn[=]>, RS and FS.

<ctrl[,]> has effect only in All-Chars mode. To show high bit setting is toggled in, the baseline moving marker changes to a hollow arrowhead. In this mode the MSbit of character entries is set. As the <ctrl[U]> special character function allows regular control character (ASCII code below 32) entry, any character value can now be entered into the text buffer. Patterns so displayed depend on the character set currently loaded.

<fctn[,]> in Euro-Writer mode only, modifies the normal vowel under the cursor to one with a circumflex accent. Vowels so modified must be retyped to normal form for changing the accent. Some of the modified forms may already be available in some national character files as alternative versions of regular 7-bit ASCII codes.

<fctn[.]>, <fctn[-]>, <ctrl[-]> similarly apply umlaut, grave and acute accents respectively.

(5) New Load/Save Functions

The Load/Save module now has code which performs extensive validation tests on incoming tab records from any mode into any mode. Loading and saving of text records now bypasses DSR search and goes directly to the opened DSR for improved speed. Changes under user selection are in the option codes for PrintFile.

M -- sets PF to output the file in DisFix/128 TI file format with MS-DOS end of line <cr><lf> separators and <ctrl[Z]> end of text marker.

U -- does similarly for Unix format with <lf> separators and <ctrl[D]> at end of text. M and U both cancel the L option for line numbers.

P -- if a printer start-up control code sequence has been installed this will be sent to the print device before any text records.

Q -- if a printer reset control code sequence has been installed this will send it to the print device after all text records have been output.

A -- opens the DV/80 output file in Append mode.

There is no provision for external files in the M/U formats to be read in directly and external conversion programs will be needed to produce DV/80 files first for loading by the Editor.

(6) New Show Directory Functions

The transition to SD display has been speeded up further. Three filename buffers are now displayed, showing the current workfile, the temporary loadfile and the file in the view buffer. Print Directory now uses <ctrl[P]> and goes to the device previously entered as DP device. This is initialized to the PF device. <P>rotect and <U>nprotect of files now use these more obvious keys. Pressing <T> marks the file under the cursor bar in the directory as the LoadTemp file.

Entering 0 for the disk number now reads the Internal, Relative 38 catalog pseudo-file for the pathname as configured or as last entered by HD from the command line. A directory so obtained does not indicate fractured files. File protection is indicated, but nothing can be done about it, as only DSR file level operations are available in this mode. Marking, deleting, and viewing function normally.

A fully scrolling file view function similar to that in DiskReview has been added, using <V> or <ctrl[V]> for auto-scroll. The previous one way scroll is now on <W>. The key functions have been simplified by not having auto-scroll on <ctrl[Q]> and <ctrl[A]>. This change may well be made in DiskReview too. The <space> in View mode now serves only to pause autoscrolling.

The buffer structure in VRAM matches that of DiskReview <V>iew except that it allows for 80 character lines only. The Editor entry code checks VRAM for a data structure compatible with this limited case and if it finds any, signals it by the name Buffer Recovery, but position markers are not restored. Existing View buffer contents may be reviewed by pressing <enter> at any time. Pressing <V> (or <ctrl[V]> for autoscroll) replaces the current buffer contents with the Display/80 file currently under the directory cursor bar.

As a last little addition, the SD screen also shows the number of bytes remaining in the text buffer. This includes the effects of buffer encoding and Euro-Writer and All-Chars will give a lower figure than 7-bit modes on the same file. The empty buffer value may change with future revisions.

(7) Configuration of ED

The program INSTALL/ED (object file, Funnelweb option 4 Load and Run) allows a range of initial options to be installed in ED from a DV/80 text file. CONFIG/ED is such a file and is its own documentation. Keep for reference, but a cut down version such as CON/ED will do just as well.

(8) Auxiliary files

The European character files are the same as for the TI Euro-Writer. Some All-chars files CHAR@x have been included. These have TI99/4A type characters for control characters, but otherwise match the full character sets found on PCs. All these files have a dummy 6 byte Editor Assembler program header.

The Euro command text files are prepared using the

template form such as F8TXAE/S. Follow instructions in the file carefully. This is assembled, loaded as an object file from Funnelweb Option 4, followed by FWTXMAKE. Enter the disk number first and then the identifier letters. This saves the data back to disk as a program file. If the Editor does not find the indicated file, it ignores the error but then does not set Euro-Writer mode.

Help files are prepared as a 26 line DV/80 file and converted to program file format using HELPMAKE. The help file loader on the Editor starts with HELPED and works through to HELPEG for as many files as set by INSTALL/ED. If one of the series is not found the loading continues to the next. More than enough possible candidates are included. Also needed is a screen of Formatter information.

(9) Prognostications

Please report pronto any bugs found or suggestions. One addition that would be technically easy to do, but probably not worth the code space is switchable Global Search direction. An addition that looks obvious, but would be extremely expensive in code space in the main program, is to implement direct transfer from the View buffer to the text buffer.

The new Editor may well be issued in a cut down USA-mode only version as part of the main Funnelweb package and the full version with all auxiliary and utility files as a supplementary disk. Some versions (German and Swedish) with non-English command text are included where the language files are in good shape. To use one of these you will need to have the appropriate 7-bit character files as C1 and C2 in your system.

No work has yet been done on interfacing the the Euro-Writer Formatter. For the moment, change the word at >30 in the first sector of FORMA1 from >130A to >100A so that you at least can use it with Funnelweb or Editor Assembler. Edit the drive number and language letter in the string DSK1.TXTFA in sector >0D and at >20 of sector >0E change >D800 to >9800 to disable the language selection path from the TI-Writer module.

At this time English, German and Swedish text files are included and incomplete prototypes done in French (almost complete), Italian and Dutch. Spanish has not even been looked at yet. This reflects both my limited language abilities and level of interest in Funnelweb from these parts. Suggestions for improvements in all of these are welcome. In language files where I am uncertain of the words, the entry has been left in English, otherwise they are based on the existing TI Euro-Writer files. There may seem to be a lot of files, but any individual user would need only a subset.

A first pass at a pure 40 column version is included that implements the enhanced functions that do not depend on the 9938 VDP and lots of VDP memory. Euro-Writer and a Help function yes, but All-Chars mode and SD by pathname will require a separate version that will have to reload FW from disk (as MG/MH in earlier releases of Funnelweb did and still does). Whether this special version is done, if at all, will depend on the level of serious interest. Bi-directional fully scrolling <V>iew is definitely not on. Use INSTALL/ED here also. Help files HELP4A, B, etc are loaded successively from disk.

WARNING - it seems that the ROM based ROS for HRDs by J.P. Hoddie is incompatible with 80 column adapters because it ignores TI99/4A system guidelines. Use the Bud Mills/OPA version 8.14 ROS in RAM instead, which better respects TI99/4A system rules. Also the current QUEST RD ROS has a bug in program file loading which does not agree with 40 column Help file loading.

WARNING - English speakers fooling around with the Swedish model should remember that LF in Swedish means SaveFile!! Various other little traps may exist also. o

Treasurer's Report

by Geoff Trott

Well we have received taxation exempt status and have been declared a non-profit company in the eyes of the Australian Securities Commission and so we have done well. I have also received a letter from Jim Banfield about his contributions to the TND on machine language. He is going to wait for some feedback before he writes any more, although there are still five or six already in the pipeline. If you appreciate his contributions, please drop me a line any way you want and I will see it gets to Jim. I asked at our user group meeting and George Meldrum said that he was enjoying them. That makes two readers Jim, or perhaps three with yourself! I also got a kick out of the postscript to Jim's letter. It read: "It is often said that a Scientific Paper is read by only four people: the author, the editor and the referees; sometimes by only the first two!" Perhaps this also applies to articles in the TND. I hope not but how are authors to know any different?

Income for May	$2532.30
Payment for May	$862.48
Excess of income over expenses for May	$1669.82

o

continued from page 13

Note that (A<LEN(A$)) returns 0 (or FALSE) if A is equal to or greater than LEN(A$) and -1 (or TRUE) if A is less than LEN(A$). The test and the addition can be combined into one formula:

```
140 C=C-(A<LEN(A$)) :: IF EOF(1) . . .
```

Line 130 changed to:

```
130 A=X+1 :: X=POS(A$," ",A) :: IF X THEN IF X=A THEN
130 ELSE C=C+1 :: GOTO 130
```

This suggested another change to line 130. IF X=A . . . could also be incorporated into a formula:

```
130 A=X+1 :: X=POS(A$," ",A) :: IF X THEN C=C-(X<>A) ::
GOTO 130
```

To make sure that we do not test for a word at the end of a line when execution moves from line 120 to line 140, I moved C=C-(A<LEN(A$)) to the end of line 130.

The program is now leaner and faster:

```
100 ! WORDCOUNT
110 DISPLAY AT(12,1)ERASE ALL:"File: DSK" :: ACCEPT AT
(12,10):A$ :: OPEN #1:"DSK"&A$,INPUT
120 X=0 :: LINPUT #1:A$ :: IF A$="" THEN 140 ELSE IF
ASC(A$)=46 OR ASC(A$)>127 THEN 140
130 A=X+1 :: X=POS(A$," ",A) :: IF X THEN C=C-(X<>A) ::
GOTO 130 ELSE C=C-(A<LEN(A$))
140 IF EOF(1) THEN CLOSE #1 :: DISPLAY AT(14,1):"Has
about";C;"Words" ELSE 120
```

GETTING THE BUGS OUT

Did you ever wonder why software and hardware glitches are called bugs? I have this from two sources, so it might even be true.

In the early days, computers were made of relays and vacuum tubes. The first computers took up a whole room, required an extraordinary amount of electricity and were, at best, balky. A big, big computer had a whopping 4K of memory. It could not do a quarter of what a 4A does.

These early computers were problem prone. Tubes would blow. The relays would hang up. Heat was always a worry. The relays would also trap insects. Every now and then, you see, they had to stop and get the bugs out. That is where "debugging" came from. o

To See or Not to C

by Geoff Trott

Due to lack of time this month, I will just give you a program to test the string functions that I presented last month. The program is written to allow the user to enter two strings of characters up to a length of 80 characters each. The length of each string is printed on the screen and then a message is printed to say which string is greater. Then a number is requested and the strings are compared for that number of characters. When a c99 program ends it asks if you want to run it again so that to try different strings it is only necessary to answer "y" at this time. This saves putting a loop inside the program itself.

Now to the actual program itself. It is necessary to define space for the two strings. This I have done with the "char" type declaration using two arrays of 81 characters. 81 is used to allow for 80 characters and the terminator character (NULL). Because this is done before the "main()" statement these are global variables and are accessible from any procedure. I could have put this statement inside the main procedure (after the first "{") which would have made them local variables. It makes a difference to where they are stored but in this case there is no problem with putting them in either place. The other statements before the main procedure are the two includes, one for the standard I/O definitions and the other to define names for the string procedures. Then, because the program is going to use the formatted input and output procedures (scanf and printf), their names must be declared as external for the assembler. Some assemblers for other processors assume that all undefined names are external and so do not need such statements. The TI99/4A assembler gives errors for undefined names. If you compile a C program without errors but get some undefined symbol errors in the assembly phase, you may well need to add some more names to the extern statement. On the other hand you may have mis-spelled a name!

Now look at the main procedure. It first identifies a local variable "n" as an integer. This will be a 16 bit number. There are a number of calls to the "printf" to output text on the screen. You will notice the characters "\n" appearing quite often. This is the C way of going to a new line. In fact a "\" causes the next character or up to 3 digits to be interpreted as special characters. There are a number of single letter special characters, and if 3 digits are used they are used as the ASCII code of a character (interpreted in octal) but we might leave that until later. Just note that "\n" means a new line. The "scanf" procedure is used for input and the "%s" is used to specify that the first variable is to be a string of characters. The character "%" is also a special character which is used to indicate that the following characters are to be interpreted as formatting characters. There are a number of these and will be using another one later in the program.

As the data coming in must go into the array of characters using a pointer. It so happens that the name of a variable on its own is a pointer to the first item of the array. It is the same thing as putting &str1[0], as str1[0] is an actual single character. Obviously, the operator "&" applied to a variable name returns the value of the pointer to that variable or address where that variable is stored in memory. The next statement prints out the number of the characters in the string with one line at the start and two blank lines at the end. Note the use of the "%d" to specify where the number is to be printed and the fact that it will be a decimal number. The number that will be printed will be the integer returned by the procedure strlen().

If you remember from last time, to pass an array to a procedure you must use a pointer, just as in scanf but for a different reason. C can only pass a single parameter to a procedure so if you wish to access an array it can only be done with a pointer to the array. So here also, the name of the array behaves as a pointer to the first item in the array. This is also used for all the procedure calls to the string procedures.

The break in the line is just to fit it into the magazine layout and these two lines would normally be on one line. There is no problem with breaking the line after the "," as spaces are ignored there. There would be problems with breaking a line in the middle of a text string (why? try it and see what happens), so I have put in more "printf" statements to do this later in the program.

```
/* A second program in c99 by Geoff Trott */
/* to test the string procedures          */
#include "DSK1.STDIO"
#include "DSK1.STRINGI"
extern printf(),scanf();
char str1[81], str2[81];
main()
{
  int n;
  printf("Enter a string of characters\n");
  scanf("%s", str1);
  printf("\nThis string has %d characters\n\n",
                                    strlen(str1));
  printf("Enter another string of characters\n");
  scanf("%s", str2);
  printf("\nThis string has %d characters\n\n",
                                    strlen(str2));
  if (strcmp(str1, str2) > 0) {
      printf("\nThe first string is greater than ");
      printf("the second string\n\n");
    }
  else if (strcmp(str1, str2) == 0)
      printf("\nThe two strings are identical\n\n");
    else {
      printf("\nThe second string is greater than ");
      printf("the first string\n\n");
    }
  printf("Type in a number: ");
  scanf("%d", &n);
  if (strncmp(str1, str2, n) > 0) {
    printf("\nThe first string is greater than the ");
    printf("second string for the first %d", n);
    printf(" characters\n\n");
  }
  else if (strncmp(str1, str2, n) == 0) {
      printf("\nThe two strings are identical ");
      printf("for the first %d characters\n\n", n);
    }
    else {
    printf("\nThe second string is greater than the ");
    printf("first string for the first %d", n);
    printf(" characters\n\n");
  }
  exit(0);
}
```

Now the two strings are compared and the result of the comparison output by means of the if statements. The first if statement isolates the greater than result. Note that the statements to be executed if the first if is true are enclosed in {}. Two statements were only necessary because of line length limitations and if there were only one statement the {} would not be necessary. If the greater than condition is not true then the else is executed. Here I separate equal from less than with another if. Note that the equal logical test is done with a double "=" to make it distinct from the assignment single "=". This can be a cause of error if you are not careful, as the compiler does not mind either one but they do entirely different things. The second else is for the case of not equal and not greater than so it must be for the less than.

The second part of the program asks for a number and then uses that to compare that number of characters in the two strings with similar code to before. Note the use of the "&" operator to pass a pointer to the variable "n" to scanf and the use of "%d" to place and format the value of "n" in the output.

continued on page 26

TI-Bits Number 17

by Jim Swedlow, CA USA

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

JIFFYFLIER

JIFFYFLIER is a new release from Roger Merritt, one of TI's graphics gurus. It is an elegant program that does its job quickly and simply. In other words, I liked it.

JIFFYFLIER makes one page fliers. It requires 32K, one disk drive, Extended Basic and an Epson compatible printer. It supports the colour capabilities of the Star NX1000 Rainbow. Roger points out that you will also need "lots of paper to enjoy the output".

The flier can have a border, a graphic within the flier, one line of large text and up to 40 lines of small text. You have a choice of font for the small text. The large text line can be regular or inverted characters.

The distribution disk includes lots of borders and small text fonts. There are 55 graphics for inside the flier. You can also use other CSGD "/GR" graphics.

Choices are very simple - use the <SPACE BAR> to change something and <ENTER> to accept it as displayed. The flier is shown on your screen half a page at a time. Once you are done, you can save your flier to disk and edit it with TI Writer.

The documentation is straight forward. It is only three pages long but that is all that is needed as the program is simple to operate.

Printing is very fast due to a new screen dump Assembly routine by Robby Robinson.

At $11, JIFFYFLIER is an outstanding value. If you want a copy, send your check to:

Roger Merritt
1949 Evergreen Avenue
Fullerton, CA 92635

Be sure and mention where you heard about JIFFYFLIER.

TI IN SOUTH AFRICA

EDITOR'S COMMENT: This article was written a few years back and what follows is included for historical curiosity.

I received a letter from a TI owner in South Africa. I asked him about the status of the 4A in his country. He responded:

"There were, to my knowledge, only about 650 4A's and 300 PEB's sold. There was a user group in Johannesburg of which I was a postal member. This folded about two years ago due to lack of support.

"I then transferred my membership to a group in ___ who lost interest in me when they learned I was not prepared to make pirate copies of my proprietary software. I never heard from them or of them again so I do not know if they are still operating. There were also groups in Durban and Capetown some years back but again, I do not know if they are still operating.

"At the moment I know of only one other active user. He is in Pretoria and we help one another where we can. He also has a 2 disk drive PEB with 32K RAM and a printer. For the rest, we are on our own and very isolated.

"Still, I have the 4A, I have paid for it, learned a lot from it and still intend to learn lot's more. I have some great software for it so it will be a LONG time before it gets dumped. (There is no market for it in South Africa anyway.) At work I have an IBM XT compatible but there still are thing the TI does better.

"The only popular computers on South Africa with lots of user groups are the IBM compatibles. Even the Commodore's and Atari's have lost support. Although Amigas and ST's have advertised and put on shows, etc, I have not heard of anyone in South Africa who actually has bought one.

WORDCOUNT

This short program appeared in the PUNN Wordplay. It counts the number of words in a DV80 file.

```
100 ! WORDCOUNT
110 DISPLAY AT(12,1)ERASE ALL:"File: DSK" :: ACCEPT AT
(12,10):F$ :: OPEN #1:"DSK"&F$,INPUT
120 A=1 :: LINPUT #1:A$ :: IF ASC(A$)=46 THEN 140
130 X=POS(A$," ",A) :: IF X=0 THEN 140 :: IF X=A THEN
A=X+1 :: GOTO 130 ELSE F=1 :: C=C+1 :: A=X+1 :: GOTO 130
140 C=C+F :: F=0 :: IF EOF(1)<>1 THEN 120 ELSE CLOSE
#1 :: DISPLAY AT(14,1):"Has about";C;"Words"
```

This nifty program does its job but I though it could be improved a bit.

In line 110, F$ is used only for the file name. Later on, A$ is used for the line contents. Since we do not need to know the value of F$ and A$ at the same time, I changed F$ to A$. Each additional variable slightly slows program execution. Line 140 has this code:

```
IF EOF(1)<>1 THEN 120 ELSE CLOSE . . .
```

I changed it to:

```
IF EOF(1) THEN CLOSE . . . ELSE 120
```

This eliminated the step of comparing EOF(1) to the number one. This works because EOF(1) returns 0 (or FALSE) if it is not the end of the file, 1 (or TRUE) if it is and -1 (also TRUE) if the disk is full. In an IF test, zero is FALSE and everything else is TRUE.

Line 120 could cause the program to bomb. ASC(A$) causes a fatal error when A$ is a null string. To prevent this, I added some code:

```
120 A=1 :: LINPUT #1:A$ :: IF A$="" THEN 140 ELSE IF
IF ASC(A$)=46 OR ASC(A$)>127 THEN 140
```

The test for Formatter commands (lines that begin with a period) is now performed after we have excluded null strings. Remember that ASC(A$) returns the ASCII value of the first character in A$.

This also eliminates lines that contain Editor tab and margin information (these lines start with a character that has an ASC value higher than 127).

In line 130, IF X=0 THEN 140 ELSE can be simplified. The same thing can be accomplished with IF X THEN. When X=0 execution will automatically transfer to line 140 (the next line). Also, it did not seem necessary to duplicate A=X+1. The line changed to:

```
130 A=X+1 :: X=POS(A$," ",A) :: IF X THEN IF X=A THEN
130 ELSE F=1 :: C=C+1 :: GOTO 130
```

The beginning of line 120 must be changed from A=1 to X=0.

The flag <F> in lines 130 and 140 bothered me. The program sets F as 1 every time it counts a word. The flag is used to make sure that the last word in the line is counted. I noticed that you could also tell if the last word needed to be counted if X=0 and A is less than LEN(A$).

continued on page 11

XB tips Number 18

by Jim Swedlow, CA USA

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

WHAT IS A NIBBLE, ANYWAY?

This month I am going to try and explain all of the various number words we run across. With luck, after you finish reading this, you will have some understanding of bit, byte, nibble, word, hex, binary and where -31952 really is in memory. With luck.

Computers really think in binary. In this numbering system there are two numbers, 0 and 1 (or, if you are a computer, off and on). While this works for your 4A, binary is cumbersome for humans. For example, in binary 41,576 is 1010001100011100.

Hex, or hexidecimal, has sixteen numbers from zero to F. Here are the first sixteen numbers in binary, decimal and hex:

DECIMAL	HEX	BINARY
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

The next number would be 16 or >10 or b1000 (> means hex and b means binary).

One binary number is a bit. Four bits is a nibble. So, 10 or A or 1010 takes four bits or a nibble to express.

A byte is eight bits or two nibbles. With a bit you can count from zero to one. A nibble gets you from zero to fifteen. The range of byte is:

Base	Low	High
Binary	0	11111111
Hex	0	FF
Decimal	0	255

You have probably noticed the numbers 16 and 255 when using your TI. ASCII character run from 0 to 255. There are sixteen colours (1 to 16, really 0 to 15). A string can be up to 255 characters long. And on and on.

Before tackling the next thing, a word, lets see if we can decode something. Lets take b10100 or >14. To convert either number to decimal, we need a method:

>14 is >10 plus >4
>10 is 16 and >4 is 4
16 plus 4 is 20
Hence, >14 is 20

b10100 is b10000 plus b100
b10000 is 16 and b100 is 4
16 plus 4 is 20
b10100 is 20

Further than that I cannot go in this space. A word is sixteen bits or four nibbles or two bytes. The range of a word is:

Base	Low	High
Binary	0	1111111111111111
Hex	0	FFFF
Decimal	0	65,535

But there are no negative numbers. Since we need them, we use something called twos compliment (which is way beyond the scope of this column and this writer). I can tell you, however, the impact;

Hex range	Decimal Range
0-7FFF	0 to 32,767
8000-FFFF	-32,768 to -1

Remember that >8000 is the next number after >7FFF.

Some examples:

7FFF is 32,767
8000 is -32,768
FFFF is -1
0 is 0

Confused? So was I until I worked with it for a while. These conversion rules may help:

* Any number less than or equal to 32,767 requires no conversion.

* Subtract 65536 from any number over 32,767.

* Add 65536 to any number less than zero.

This conversion process can be expressed in basic:

```
AD=AD+65536*(AD>32767)
```

If AD is the address, this returns the same number if AD is less than or equal to 32767. If AD is greater than 32767, the test returns true (-1) and a negative 65536 is added to AD. Try it on your computer.

Bottom line time. Suppose you see CALL PEEK(-31952,A,B). Where is -31952? Well, since it is less than zero, we add 65536 and get 33584 or >8330. Now you know!

FANS

I have heard good things about a PEB fan from Paul Johnson's company STATO, Inc. The fan is $15.50 plus $2.50 shipping (or $18) and is supposed to be super quiet.

Warning: it is a bit of a job to take your PEB apart to replace the fan. I have done it (to fix a broken on-off switch) and if you are good with a screwdriver, it can be done. Just do it very slowly and carefully.

STATO, Inc's address is Post Office Box 145, Townsend, MA 01469-0145. If you get one, let me know how it works.

NEXT MONTH

I have published three programs to print labels (two versions of a label program for disks and one for addresses and such). I had occasion to combine the programs (and spiff them up a bit). It will be published next month. After that, I promise, no more label programs!

Enjoy! o

XB Tips

Undocumented features of ACCEPT AT

by Ben V. Takach

No matter how familiar one is with the 99/4A machine and Extended Basic there are still surprises in store. The background to my story is buried in Bob Relyea's article 'Using The Arrow Keys in Extended Basic Programs' (page 16, May '92 issue of TND).

The Accept At statement has undocumented features, which makes this unique statement even more powerful. One may use the up and down arrows in conjunction with the Call Key statement; both of these will function exactly as the enter key.

Where would one use this feature? Assuming you have a program, which requires one or more screens full of data entries (inputs) to calculate and execute the program. If an incorrect entry is made and discovered immediately after the enter key has been pressed or a few lines further down, there is no chance to correct the mistake until one reaches the bottom of the page - if there is an option at all - to go back to the top and stepdown line by line again. Using the undocumented features one can freely move up or down the page using the two arrow keys only, and change the entries at will. It is very convenient, user friendly and professional.

The technique will work equally well within a For To - Next loop or a line by line entry. Here is how it works:

First you fill the screen - or a portion of it - with the strings relating to the requested data using Display At statements, including the name assigned to the requested data at the appropriate screen position.

Next you configure the Accept At lines, each followed by a Call Key statement, and an if then statement: if return variable = 11 then previous line. The Call Key statement need not follow the Accept At statement immediately. You may assign a new variable name to the entered data or have several assignment statements between Accept At and Call Key. However you may not command it to execute any calculation in between the two!

To be able to enjoy the total freedom of moving up or down, some of the options of the two statements (Display At and Accept At) should be included. One has already been mentioned, namely a value ought to be placed in the screen position , where Accept At will look for it, by the Display At statement, even if the value is 0. The second essential option is the size with a minus sign in the Accept At statement.

The last important feature is the On warning next line preceding the routine. This is essential to avoid the destruction of the screen whenever the letter O is pressed instead of zero etc.

Believe me it is magic if it is done properly.

Now for some program examples from my own programs. The purpose of the programs is immaterial and of no concern, thus I will not even mention the names.

```
680 DISPLAY AT(12,1):"WIDTH,FIXED DIE?";DI$;D1
690 DISPLAY AT(13,1):"WIDTH,WOV. DIE ?";DI$;D2
700 DISPLAY AT(14,1):"HEIGHT,FIX. DIE?";DI$;E1
710 DISPLAY AT(15,1):"HEIGHTMOV'G DIE?";DI$;E2
720 DISPLAY AT(16,1):"THICKN.FIXED D.?";DI$;G1
730 DISPLAY AT(17,1):"THICKN.MOV'G D.?";DI$;G2 ::
    GOSUB 2290
740 ACCEPT AT(12,22)SIZE(-5):D1 :: IF D2=0 THEN D2=D1 ::
    DISPLAY AT(13,21):D2
750 ACCEPT AT(13,22)SIZE(-5):D2 :: CALL KEY(0,X,Y)::
    IF X=11 THEN 740
760 ACCEPT AT(14,22)SIZE(-5):E1 :: CALL KEY(0,X,Y)::
    IF X=11 THEN 750
770 IF E2=0 THEN E2=E1 :: DISPLAY AT(15,21):E2
780 ACCEPT AT(15,22)SIZE(-5):E2 :: CALL KEY(0,X,Y)::
    IF X=11 THEN 760
790 ACCEPT AT(16,22)SIZE(-5):G1 :: CALL KEY(0,X,Y)::
    IF X=11 THEN 780
800 IF G2=0 THEN G2=G1 :: DISPLAY AT(17,21):G2
810 ACCEPT AT(17,22)SIZE(-5):G2 :: CALL KEY(0,X,Y)::
    IF X=11 THEN 790
```

Example 1

The above segment illustrates the line by line entry method. Lines 680 to 730 display the requests on rows 12 to 17. D1, D2, E1, E2, G1, G2 are the assigned variable names. These will be printed on the screen - initially being 0 - as zeroes on columns 22. Subroutine 2290 in line 730 contains the prompt, displayed on row 23: use up arrow to correct a previous entry. The purpose of the last 2 statements in line 740, also lines 770 and 800 are purely for convenience. The widths, heights and thicknesses are usually, but not necessarily always, identical, thus these lines save repeatedly typed in entries. Just another example of user friendliness.

Lines 740 to 810 are the data input lines. Once the program reaches 740, you may enter any numeric data and use the enter or the down arrow key to step to the next input line. You may edit the entries at will hopping up or down. The remarkably simple structure has an inherent implied protection against entering a string character. It will not accept it, and thanks to the preceding on warning next line it will not issue any warning, which would destroy the screen layout. The cursor will patiently remain on column 22 if you happen to press a letter key. Eventually you will wake up and press the appropriate key.

```
410 DISPLAY AT(1,1)ERASE ALL:"AMBIENT TEMP.";T$;B
420 DISPLAY AT(2,1):"POURING TEMP.";T$;PT
430 DISPLAY AT(3,1):"EJECT. TEMP. ";T$;ET
440 IF SR=0 THEN SR=55
450 DISPLAY AT(4,1):"SHOTRATE (shot/hour)";SR
460 DISPLAY AT(5,1):"SHOT WEIGHT     ";W$;SW
470 GOSUB 2290
480 ACCEPT AT(1,22)SIZE(-4):B :: CALL KEY(0,X,Y)::
    IF X=11 THEN 160
490 ACCEPT AT(2,22)SIZE(-4):PT :: CALL KEY(0,X,Y)::
    IF X=11 THEN 480
500 IF PT<=SO THEN GOSUB 1900 :: GOTO 490
510 ACCEPT AT(3,22)SIZE(-4):ET :: DF=ET :: CALL KEY(0,X,Y)::
    IF X=11 THEN 490
520 IF ET>=LI THEN GOSUB 1900 :: GOTO 510
530 ACCEPT AT(4,22)SIZE(-4):SR :: CALL KEY(0,X,Y)::
    IF X=11 THEN 510
540 DISPLAY AT(5,1):"SHOT WEIGHT     ";W$;SW ::
    ACCEPT AT(5,22)SIZE(-6):SW :: CALL KEY(0,X,Y):: IF X=11
    THEN 530
550 IF SW<>0 THEN 620
560 DISPLAY AT(23,1)BEEP:"WISH TO ENTER VOL. (y/n) ?" ::
    CALL KEY(3,V,ST):: IF ST<>1 THEN 560
570 IF (V=89)+(V=78)THEN 580 ELSE 560
580 DISPLAY AT(23,1):: IF V=78 THEN 540 ELSE DISPLAY AT(5,1):
    "SHOT VOLUME (cc) ";SV
590 ACCEPT AT(5,22)SIZE(-6):SV :: CALL KEY(0,X,Y)::
    IF X=11 THEN 540
600 IF SV=0 THEN 540 ELSE 610
610 SW=INT(SV*R11)/1000
620 DISPLAY AT(6,1):"DIE HEATER(kw)?              ENTER 0
    IF NONE     ";HQ :: ACCEPT AT(7,22)SIZE(-6):HQ ::
    CALL KEY(0,X,Y):: IF X=11 THEN 540
630 MT=SR*SW :: QT=(MT*CL)+((SO-ET)*CS*MT)+((PT-SO)
    *CLI*MT)+(HQ*3600)
640 QT1=QT :: DISPLAY AT(9,1):"GR.HEAT INP.IS";Q$;INT(QT1)
650 DISPLAY AT(10,1):"GR.HEAT INP.IS  (kW)";INT(QT/3600)::
    GOSUB 1910
```

Example 2

My second example is similar to the first. It illustrates the use of the technique including a few other options. Lines 410 to 470 need no explanation, these work much the same as lines 680 to 730 in the first example. The Call Key and if-then statements in line 480 provide a graceful return to the previous screen page, if the display indicates by the default values of PT and ET that the wrong selection was made earlier.

Line 510 works just as well as 480 and 490 although we have an assignment statement sandwiched between Accept At and Call key. The calculation in line 610 logically could have been calculated immediately following the Accept At statement in 590, however this would not work! The Call Key statement following it would be ineffective.

I guess you will expect an explanation of the following 4 lines from 620 to 650. Well here it is; These should not even be published here, because they are irrelevant to this report. I have left them there to prove that the program segment is part of a real dinky-di program, and that it was not written for the sake of an example.

```
130 DIM Q$(32),V(32),H$(5),W$(5)
140 Q$(0)="MACH.REPL.COST ($)" :: Q$(1)="MACH.REPL.TIME ($)"
150 Q$(2)="INTEREST RATE (%) " :: Q$(3)="SPACE ($/UNIT SQ.)"
160 Q$(4)="AREA USED (UNITS) " :: Q$(5)="INSURANCE ($/y)    "
170 Q$(6)="MISC.FIX.CST.($/y)" :: Q$(7)="HOURS WORKED (h/y)"
180 Q$(8)="HYDR.FLUID ($/y)   " :: Q$(9)="LUBRICANTS ($/y)  "
190 Q$(10)="ELECTRICITY ($/y) "
200 Q$(11)="FUEL/no melt/($/Y)" :: Q$(12)="WATER ($/y)         "
210 Q$(13)="DIE SPRAY ($/y)    " :: Q$(14)="CONSUMABLES ($/y) "
220 Q$(15)="WAGE RATE (gr.$/h)" :: Q$(16)="LABOUR%(1 M.=100%)"
230 Q$(17)="OVERHEAD(%on wage)" :: Q$(18)="DIE COST ($)       "
240 Q$(19)="DIE LIFE  (shots) " :: Q$(22)="REJECT RATE (%)    "
250 Q$(20)="DIE CAVITIES (No) " :: Q$(21)="CASTINGS/Y (No)    "
260 Q$(23)="DIE MAINT.($ tot.)" :: Q$(24)="ALLOY COST ($/kg) "
270 Q$(25)="TRIMMED WGHT. (kg)" :: Q$(26)="METAL LOSSESS  (%)"
280 Q$(27)="HANDLING ($/cast.)" :: Q$(28)="TRIMMING ($/cast.)"
290 Q$(29)="FINISH   ($/cast.)" :: Q$(30)="SHOT RATE (Shot/h)"
300 Q$(31)="MELT. COSTS ($/kg)"
310 H$(1)="DATE          " :: H$(2)="JOB NAME/No." ::
    H$(3)="ORDER No.    " :: H$(4
    )="MACH.TYPE/No"
320 H$(5)="CUSTOMER     " :: GOSUB 1040 :: W$(1)=DA$
330 DISPLAY AT(9,1)ERASE ALL:"Casting Cost & Mach.
    Load    Estimation Ver.3 (1992
    )":"by B.V.Takach,Aug.1987"
331 ON WARNING NEXT
340 DISPLAY AT(22,1):"PRINTOUT ? (Y/N)" :: CALL KEY(3,K,S)
    :: IF S=0 THEN 340
350 IF K=78 THEN P=0 :: DISPLAY AT(22,1): ::
    GOTO 390 ELSE IF K=89 THEN P=1 ELSE 340
360 IF P$="" THEN P$="PIO" :: DISPLAY AT(22,1):
370 DISPLAY AT(14,1):"PRINT DEV.No ";P$ ::
    ACCEPT AT(14,14)SIZE(-7):P$ :: PR$=P$&",OUTPUT"
380 FOR I=1 TO 5 :: DISPLAY AT(I+14,1):H$(I)&" "&W$(I)::
    NEXT I :: FOR I=1 TO 5
    :: ACCEPT AT(I+14,14)SIZE(-14):W$(I):: NEXT I
390 !DATA INPUT SECTION
391 S=15 :: X=0 :: L=4 :: GOTO 393
392 S=31 :: X=16 :: L=4
393 CALL CLEAR :: DISPLAY AT(1,1):"CASTING
     COST & MACH.LOAD PGM================
    ============" :: DISPLAY AT(22,1):"use UP-ARROW key to
    to return to the previous line."
394 FOR CL=3 TO 21 :: DISPLAY AT(CL,1): :: NEXT CL
395 J=L :: FOR I=X+0 TO S
400 DISPLAY AT(J,1):Q$(I)&" ";V(I);:: J=J+1 ::
    NEXT I :: J=L
405 FOR I=X+0 TO S
406 ACCEPT AT(J,21)SIZE(-6):V(I)
409 CALL KEY(0,KX,SX):: IF KX=11 THEN I=I-1
    :: J=J-1 :: GOTO 406
410 !IF V(I)=-1 THEN V(I)=V(I)*-1 :: I=I-1 ::
    J=J-1 :: GOTO 406
420 J=J+1 :: NEXT I
421 DISPLAY AT(22,1): :: DISPLAY AT(23,1):
    :: DISPLAY AT(22,1):"ANY CORRECTION ? (y/n)"
422 CALL KEY(3,K,ST):: IF ST<>1 THEN 422
423 IF (K=78)*(X=0)THEN 392
424 IF (K=89)*(X=0)THEN 391
425 IF (K=89)*(X=16)THEN 392
426 IF (K<>89)*(K<>78)THEN 422
```

Example 3

The last example shows the application of the technique in a for-to-next loop. Lines 130 to 320 have been reprinted to aid better understanding of the routine. Line 331 is the useful line which eliminates the built-in self-destruct screen feature induced by a misplaced finger or a thumb or two. Line 380 is a typical unforgiving loop, where the said misplaced thumb can do irrecoverable damage; only function 4 followed by RUN would lead to salvation (left in to teach the user to be more careful!).

390 - 426 is the multi-page data input routine. Each for CL=3 to 21 loop will gather in one page of inputs until all 31 questions have been answered.

The purists may argue that the functions in line 393 could have been accomplished more elegantly. Indeed they could have been! One could have left the heading (line 1) standing and deleted the lines 2 - 23 by a for delete = 2 to 23 :: Display At (delete) :: next delete line. Alas the gain would have been nil! On the other hand the call clear is clearly less defendable against an erase all in the first Display At statement. I could have easily edited it out of the line, but again it may as well be left there for someone who may be inspired to create something more useful and much more beautiful.

The REM-ed out line 410 on the other hand is yet another story. Before this hidden gem of the Accept At potential was discovered - through discussions at one of our monthly club meetings - the program used the -1 correction routine. Line 410 did the (clumsy) trick. Upon the discovery of previously entered incorrect data, one would type in -1, which would then place the cursor on the previous line. At the same time the -1 entry had to be converted to get rid of the minus sign. The multiplication with -1 accomplished it. Lines 421 to 426 give the final chance to make any changes to the page full of entries.

Well, if you have an application program where a mistaken key push would cause a re-run, try this so far unpublished Extended Basic feature.

Finally, before I sign off, do you know the difference between Call Clear or Erase All and the routine:

```
FOR CLEAR= 1 TO 24 :: DISPLAY AT(CLEAR,1) :: NEXT
CLEAR ?
```

Well, this is another story. You may read all about it next time! o

Letter to the Editor

Dear Bob,

I was glad to see that you published my Reformatter+ program. I think people will find it quite useful.

However, the listing was printed from a program which contained assembly imbedded with ALSAVE, and therefore cannot run as listed!

The easiest fix is to delete line 100, add ,@ to the end of line 130, and change line 860 to-

```
860 DISPLAY AT(14,1)SIZE(-LEN(A$)):@$ :: CF=CF+1 :: IF
POS(@$,"-",1)=0 AND CF=1 THEN 860 ELSE A$=@$
```

In this case, if you want to hyphenate less than the maximum number of characters, you must erase the others. If you want to avoid that, imbed Bruce Harrison's STRACC routine, as mentioned in the documentation.

Best Wishes, Jim Peterson

The STRACC routine is published elsewhere in this issue for those of you who require it as suggested in Jim's letter. o

Tigercub Programmable Calculator

6-memories, 6-windows, 34-functions, 14-digits

by Jim Peterson, Tigercub Software, USA

I always wanted a calculator with more than one memory, and a window to display the contents of each one. The computer has plenty of memory, and the monitor screen has plenty of room for windows, so I wrote a 6-memory 6-window calculator.

Recently I decided to go back and upgrade that old program. By the time I got through I had a 6-memory 6-window 34-function programmable calculator with many other features.

It was necessary to write this program to accept either numeric or alphabetic data and then sort it out. For this reason, it does not respond as instantly as a calculator. However, I think it does some things that few if any calculators can do.

When you boot this program, the screen displays 6 memory areas marked U through Z, and you are asked if you want to label them. That will help a great deal in keeping track of what you are using them for. The computer will force you to unlock the alpha lock and label them in lower case, which will make them stand out nicely in inverse video.

Next you are asked if you want a 14-digit display. Unlike 8-bit PCs, the TI99/4A calculates to 14 digits of accuracy, but normally rounds them off to 10 digits for screen display. This option will display the full 14 digits, if it is not more than 9,999,999,999 or less than -9,999,999,999.

You are required to depress the alpha lock again to answer this prompt, and it must stay depressed thereafter.

Then you are asked if you want to use conventional or straight line mode. Conventional mode is much like you would use with an ordinary calculator - you must press Enter after you input each value, but not after each function, e.g. 77 (Enter) + 81 (Enter) = .

In straight line mode you simply type 77+81= (Enter), which is a bit faster but the computer then pauses for a few seconds to decipher the input before giving the answer.

If you want to enter large numbers in exponential notation, you must use the conventional mode.

To switch from one mode to the other, just enter J. The mode you are in is displayed in the upper right of the screen. Entering JJ will clear the memory labels and irretrievably clear all memories; Q will terminate the program.

If you use the = sign, the result is simply displayed on screen, but if you use a memory name (U through Z) the result is placed in that memory and displayed in its window. For instance, 77+81X puts 158 in memory and in window X.

You can also enter a memory name to calculate with the value it contains: e.g. U (Enter) + 81 (Enter) V adds 81 to the value from U and puts the result in V. W+XY adds the values in W and X and puts the result in Y. U+UU would double the value of U.

To poke a value into a memory, just enter a value and a memory name, such as 77 (Enter) U, or in straight line 77U.

The four basic functions are:

+ (located on Shift =) for adding,
- (located on Shift /) for subtracting,
/ for dividing and
* (located on Shift 8) for multiplication.

All that shifting is a nuisance, especially if you are using one hand to keep track in a column of figures. To make it easier, you can use P (plus) for addition, M (minus) for subtraction, D (divide) for division, and T (times) for multiplication. The correct symbols will still appear on screen.

Other available functions are:

^ (power)
% (percent) and
R (root)

For example:

10 (Enter) ^ 2 (Enter) = will give you 100, which is 10 to the power of 2.
10 (Enter) % 100 (Enter) = gives 10 which is 10% of 100.
3 (Enter) R 64 (Enter) = gives the 3rd root of 64, or 4.

FCTN U, which is the '_' symbol, will give you the value of pi. Therefore, _ (Enter) * 10 (Enter) = multiplies pi by 10.

When you enter a problem the name of the function you used, such as "addition" is highlighted in inverse video at the bottom of the screen, so you will know if you made a mistake.

With this calculator, you can even enter a series of calculations. In conventional mode 67 (Enter) + 33 (Enter) / 2 (Enter) * 5 (Enter) U or in straight line mode 67+33/2*5U (Enter) will add 33 to 67, display the result, divide by 2, display the result, multiply by 5 and put the value in U. You are limited only by the line length of 28 characters.

But I said this calculator has 34 functions. Where are the other 26? TI BASIC has a few other maths functions, and in Appendix K of the Extended BASIC manual you will find the algorithms for 20 advanced math functions. I have no idea what those do, but I programmed them into my calculator. Here they are -

CTRL A atn
CTRL B cosine
FCTN A exponent
FCTN B log
CTRL E sine in radians
CTRL F tangent
CTRL G secant
CTRL H cosecant
CTRL I cotangent
CTRL J inverse sine
CTRL K inverse cosine
CTRL L inverse secant
CTRL M inverse cosecant
CTRL N inverse cotangent
CTRL O hyperbolic sine
CTRL P hyperbolic cosine
CTRL Q hyperbolic tangent
CTRL R hyperbolic secant
FCTN F hyperbolic cosecant
CTRL T hyperbolic cotangent
CTRL U inverse hyperbolic sine
CTRL V inverse hyperbolic cosine
FCTN G inverse hyperbolic tangent
CTRL X inverse hyperbolic secant
CTRL Y inverse hyperbolic cosecant
FCTN W inverse hyperbolic cotangent

To use one of these, enter a value, then a FCTN or CTRL and = or a memory name. 8 (Enter) FCTN A U will put the exponent of 8 in memory U. Be warned that entering invalid values in some of these will cause a numeric overflow or underflow and, since I have turned off ON WARNING to avoid spoiling the screen display, you will not be informed.

You do not have any use for those? Well then, you can reprogram them for any functions you do need. They are in lines 760 through 1080. Be sure to use A for the value being input, C for the result. C=A-.1*A+.06*A will return the value of A minus a 10% discount, plus a

6% sales tax. If you need additional variables, put their values in your memories and reference them in your equation, using M(1) through M(6) for memories U through Z. C=A*M(1)/M(6) will multiply A by the value in U and divide it by the value in Z. You can write multiple statement equations, even multiple-line equations. Use J as a loop counter, @ for an internal variable; if you need other internal variables, use some that are not in the prescan list in line 110, and add them to that list. When you type the name you want displayed, use lower-case letters and use FCTN C rather than the space bar for spacing. You can easily customize this calculator with a couple of dozen formulas for whatever field you are working in.

Sometimes you might want to total the values in all memories. Just enter & to total and display.

To clear all the memories, enter C. To clear memory X, for instance, enter CX (in either mode).

E is the oops! key. Enter E to restore the last previous values in all memories, or EU, for instance, to restore the last previous value in U.

Sometimes you may just want to add up a series of numbers. Enter A= if you just want the totals displayed, or AU, for instance, to accumulate the total in U. You are now in cumulative mode and each value you enter will be added to the total. Enter Q to get out of this mode. C and E are not active in this mode, and you cannot enter memory names.

If you want a hard copy of your work, enter FCTN 0 and the memory labels, names and values will be output to your printer. If you selected the 14-digit option, the printout will also be in 14-digit format.

To save your work, enter I to save all memories, or IU, for instance, to save a specific one. You will be prompted for a disk drive/filename. To retrieve the data, enter O for all memories or OV, for instance, for a specific one.

To access the base conversion mode, enter B. You will be able to convert any number from/to any base from 2 to 36. To escape this mode, enter 0 for the value to be converted.

You find it difficult to remember all those commands? At the prompt for the first value or command, just press FCTN 7 for a Help screen. Thanks to Karl Romstedt for that one.

I said this was a programmable calculator, and I was not just referring to the fact that you could reprogram those 26 functions listed above. This calculator lets you enter an equation; the program then rewrites itself while it is running, and uses the equation to solve whatever values you give it.

To get into programming mode, enter #. You will be prompted to enter a formula. This must be in the form of a valid Extended BASIC statement, using A for the value to be determined and B through F, as many as you need, for the values you will be prompted to input. All math functions are supported, eg A=B^C-INT(SQR(C)). The program pauses to tokenize your input and then prompts you for values to use for, in this instance, A, B and C. You are then prompted for a memory name in which to store and display the answer.

Remember the mathematical heirarchy - if you want to add or subtract before multiplying or dividing, use parentheses - A=(B-C)*D/(E-F).

If your formula is not a valid Extended BASIC statement, it will be rejected. If it is valid but incorrect for its purpose, it will give erroneous results; for instance, if you use X as a variable name, you will not be prompted for its value, which will be 0. To exit the programmed formula, enter 0 at all prompts.

Finally, this calculator contains a programmable iterative calculator to solve such difficult problems as A=B^B-SQR(B), where A is the known value. These can only be solved by trial and error.

To access this mode, enter @. You will be prompted for a formula, which must be in the A=B format. The computer pauses to write the equation into itself, prompts you for a value of A, and goes through a series of trial and error calculations which are displayed on screen. Then you are prompted for a memory to receive the result. To exit this mode, give A a value of 0.

I hope you find this program useful. I am releasing it to the public domain; I will not even bother to put a fairware donation request on it. However, if you do find it useful, would you spend 19 cents for a postcard to tell me so? The few remaining TI99/4A programmers are getting tired of releasing programs and never hearing a word about them. One of these days they may just decide to only release them to each other.

Assembly Class

by Ross Mudie

The assembly class held in June only had 3 students and the scheduled subject of USING DSRLNK was replaced by an informal look at a DSR PEEKER, a program used for conversion Binary - Hex - Decimal as well as some assembly techniques.

The next assembly class will be at 10am, 4th July, at Ryde Infants school, when another attempt will be made to cover DSRLNK.

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

Multiplan Tips

by Steve Zimmerman, Pug Peripheral, PA, USA

This month's column will cover some of the problems you can get yourself into in Multiplan by using the FORmat commands. These can be an aid to help you display what you need to show, but can also have some pitfalls for the unwary. Let's take a hypothetical spreadsheet (I just made it up!). This sheet will project figures into future years, using percentage increases. We will display figures an integers, using the FORmat command.

This is where the pitfall comes in. By using the FORmat command to "simplify" the display into whole numbers, a discrepancy (or series of discrepancies) is introduced into the worksheet. When a number is displayed as an integer using the FORmat command, x 0000...1 to x.49999...round down to x. and x.50000...0 to x.99999.. round up to x+1 -BUT ONLY ON THE DISPLAY! The ACTUAL number is still stored in that cell, and will be used when that cell is referenced in calculations! The same type of error can happen when numbers are assigned a fixed number of decimals.

Multiplan will round the number of decimals displayed according to the same principle. Thus, if your display is rounded off, but the actual values are not, and the rounded are then processed in formulas, you may see results which clearly "do not add up"! Still, the computer is "right" - the problem is operator error.

To see this in action, we will now set up a spreadsheet. Begin in R2C1 with the number 10. Set the formula in R2C2 to RC[-1]*1.039. Copy this right 12 cells. Now move down to R4C1. Enter the formula =R[-2]C. Copy this right 13 cells. Now FORmat R4 to display Integer values (key F,C,R4,tab,tab,<enter>). We know that these two cells above them, but display as integers rather than decimals.

continued on page 22

Programming Tips and Reviews

by Stephen Shaw, England

DISK DRIVE XB AUTO-LOAD

Back in Issue 32, page 24, I reported to you a "bug" in the system such that if you permitted your system to auto-load a program from disk when selecting XB, you lost the use of randomize. In Issue 32 I gave a short XB program to cure it. No response from anyone! I tested the bug out on my console and found that the bug was there and the fix cured it! However I am now using a different (cosmetically older) console, and find the bug is NOT there.

Disk owners... can you do a little test and report the results please? Type in this program:

```
100 RANDOMIZE
110 FOR T=1 TO 5
120 PRINT INT(10*RND);
130 NEXT T
140 RUN "DSK1.LOAD"
```

and save this little program onto a new disk in drive one as "LOAD". Now reboot XB from the title screen so that your LOAD program is auto-loaded and watch the result. Do you keep getting a repeating pattern of five numbers or is each group of five numbers different? Please let me know, together with the serial numbers of your console (on the base) and your XB module. These typically are in the form ATA0583 - goods manufactured later in Italy often omit serial numbers, so advise "no serial/Italian" or whatever. In issue 32 there is a simple XB cure, but Bruce -who is an assembly programmer- found a similar problem occurring with some ramdisk operating systems booting machine code, so he wrote the following two routines, both intended to be used with XB. These are available on disk from the disk library ready assembled on a utility disk.

MACHINE CODE PROGRAMMERS- CIF & CFI

I received an enquiry about using CFI in a machine code program to be used from XB, and was unable to find any source code using it. The second program below uses it and works!

```
* ASSEMBLY SUBROUTINE "SEED"
* FOR USE with EXTENDED BASIC PROGRAMS
* SEEDS RANDOM NUMBERS
* THIS SEEDS RANDOM NUMBER PROCESS AND REPORTS KEY
  PRESSED INTO VARIABLE IN XB * BEHAVES LIKE A
  "CALL KEY" LOOP
* I.E.  "SEED" WILL KEEP LOOPING ITSELF UNTIL A KEY
  IS STRUCK
* CALL LINK("SEED",K) SEEDS THE RANDOMIZE PROCESS
* use as CALL LINK("SEED",K) :: RANDOMIZE
* AND REPORTS THE KEY STRUCK BY THE USER  INTO  THE
  XB VARIABLE K
* AFTER  THIS LINK, RANDOMIZE WILL WORK REGARDLESS
  OF HOW PROGRAM STARTED
* CODE BY BRUCE HARRISON
* RELEASED TO PUBLIC DOMAIN
* 18 AUG 1991
NUMASG EQU  >2008        NUMERIC ASSIGNMENT VECTOR
XMLLNK EQU  >2018        XML LINKAGE VECTOR
KSCAN  EQU  >201C        KEYBOARD SCAN VECTOR
KEYADR EQU  >8374        KEY-UNIT LOCATION
KEYVAL EQU  >8375        KEY VALUE BYTE
FAC    EQU  >834A        FLOATING POINT ACCUMULATOR
CIF    EQU  >20          CONVERT INTEGER TO FLOATING POINT
CFI    EQU  >12B8        CONVERT FLOATING POINT TO INTEGER
NUMREF EQU  >200C        NUMERIC REFERENCE VECTOR
STATUS EQU  >837C        GPL STATUS BYTE
       DEF  SEED         DEFINE ENTRY POINT
SEED
       LWPI WS           LOAD OUR WORKSPACE
       MOV  @>8378,R10   TAKE THE VDP INTERRUPT TIMER INTO R10
       ANDI R10,>0001    MASK OFF ALL BUT THE LOWEST BIT
       CLR  @KEYADR      CLEAR KEY-UNIT
KEYIN  MOVB @>83D7,@>83C1 TAKE THE SCREEN TIMEOUT'S LOW BYTE INTO SEED + 1
       CLR  @STATUS      CLEAR GPL STATUS
       BLWP @KSCAN       SCAN KEYBOARD
       LIMI 2            ALLOW INTERRUPTS
       LIMI 0            DISALLOW INTERRUPTS
       CB   @ANYKEY,@STATUS HAS A KEY BEEN STRUCK?
       JNE  KEYIN        IF NOT, GO BACK
       XOR  @>83C0,R10   NOW XOR SO LOW BIT OF R10 IS LOW BIT TAKEN ABOVE
       MOV  R10,@>83C0   PUT R10 AT SEED
       MOVB @>8379,@>83C0 PUT BYTE FROM VDP INTERRUPT INTO HIGH BYTE OF SEED
       CLR  R0           CLEAR R0 FOR NUMBER ASSIGN
       LI   R1,1         FIRST PARAMETER TO PASS
       MOV  @KEYADR,@FAC PLACE KEY'S ASCII VALUE AT FAC
       BLWP @XMLLNK      USE XML LINKAGE
       DATA CIF          TO CONVERT INTEGER TO FLOATING POINT NUMBER
       BLWP @NUMASG      ASSIGN NUMBER TO PARAMETER
       LWPI >83E0        LOAD GPL WORKSPACE
       CLR  @STATUS      CLEAR GPL STATUS BYTE
       B    @>006A       RETURN TO GPL INTERPRETER
WS     BSS  32           OUR OWN WORKSPACE
ANYKEY BYTE >20          SPACE CHARACTER ASCII
       END
```

The following program uses both CFI and CIF and works. The comments are very useful too. Note that these listings are for use with the XB module, and amendments are required for use with EdAs or MiniMem - notably you do not need some of the EQUates.

The source code below is "standalone" and does not require the above code, they are alternates!

```
* QUICK RANDOM
* MAKES RANDOM NUMBERS QUICKLY
* USE WITH EXTENDED BASIC
* TWO CALL LINKS ARE INCLUDED
* CALL LINK("SEED",K) :: CALL LINK("RKWIK",1,10,B)
* randomizes AND sets variable B from 1 to 10.
* CALL  LINK("SEED",K)  ACTS LIKE A CALL KEY LOOP,
  BUT
* SETS A RANDOM VALUE IN RANDOM NUMBER SEED
* AND REPORTS THE KEY VALUE INTO A VARIABLE (K)
* AFTER  SEED  HAS  BEEN  PERFORMED,  CALL  LINK
  ("RKWIK",LOW,HIGH,VAR) WILL WORK
* GIVE LINK THREE PARAMETERS:
* FIRST THE LOWEST INTEGER IN DESIRED RANGE
* SECOND THE HIGHEST INTEGER IN DESIRED RANGE
* THIRD  THE  VARIABLE  INTO WHICH NUMBER IS TO BE
  ASSIGNED
* LIMITS FOR LOW AND HIGH ARE (-32768 AND +32767)
* SO LONG AS "SEED" HAS  BEEN  USED,  RANDOMIZE  IS
  UNNECESSARY
* TO GET RANDOM NUMBERS FROM RKWIK
* IF  RND  IS  USED,  RANDOMIZE MUST BE DONE AFTER
  "SEED" AND BEFORE RND IS USED
* CODE BY BRUCE HARRISON
* RELEASED TO PUBLIC DOMAIN 30 AUGUST 1991
NUMASG EQU  >2008        NUMERIC ASSIGNMENT VECTOR
XMLLNK EQU  >2018        XML LINKAGE VECTOR
KEYADR EQU  >8374        KEY-UNIT ADDRESS
KEYVAL EQU  >8375        KEY VALUE ADDRESS
KSCAN  EQU  >201C        KEYBOARD SCANNING VECTOR
FAC    EQU  >834A        FLOATING POINT ACCUMULATOR
CIF    EQU  >20          CONVERT INTEGER TO FLOATING POINT
CFI    EQU  >12B8        CONVERT FLOATING POINT TO INTEGER
NUMREF EQU  >200C        NUMERIC VARIABLE REFERENCE
STATUS EQU  >837C        GPL STATUS BYTE
       DEF  SEED,RKWIK
SEED
       LWPI WS           LOAD OUR WORKSPACE
       MOV  @>8378,R10   TAKE THE VDP INTERRUPT TIMER INTO R10
       ANDI R10,>0001    MASK OFF ALL BUT THE LOWEST BIT
       CLR  @KEYADR      CLEAR KEY-UNIT
KEYIN  MOVB @>83D7,@>83C1 TAKE THE SCREEN TIMEOUT'S LOW BYTE INTO SEED + 1
       CLR  @STATUS      CLEAR GPL STATUS
       BLWP @KSCAN       SCAN KEYBOARD
       LIMI 2            ALLOW INTERRUPTS
       LIMI 0            DISALLOW INTERRUPTS
       CB   @ANYKEY,@STATUS HAS A KEY BEEN STRUCK?
       JNE  KEYIN        IF NOT, GO BACK
       XOR  @>83C0,R10   NOW XOR SO LOW BIT OF R10 IS LOW BIT TAKEN ABOVE
       MOV  R10,@>83C0   PUT R10 AT SEED
       MOVB @>8379,@>83C0 PUT BYTE FROM VDP INTERRUPT INTO HIGH BYTE OF SEED
       CLR  R0           CLEAR R0 FOR NUMBER ASSIGN
       LI   R1,1         FIRST PARAMETER TO PASS
       MOV  @KEYADR,@FAC PLACE KEY'S ASCII VALUE AT FAC
       BLWP @XMLLNK      USE XML LINKAGE
       DATA CIF          TO CONVERT INTEGER TO FLOATING POINT NUMBER
       BLWP @NUMASG      ASSIGN NUMBER TO PARAMETER
       LWPI >83E0        LOAD GPL WORKSPACE
       CLR  @STATUS      CLEAR GPL STATUS BYTE
       B    @>006A       RETURN TO GPL INTERPRETER
RKWIK
       LWPI WS           LOAD OUR OWN WORKSPACE
       CLR  R0           CLEAR R0, NOT ARRAY VARIABLE
       LI   R1,1         SET FOR FIRST PARAMETER
       BLWP @NUMREF      GET FIRST PARAMETER (LOW END OF DESIRED RANGE)
       BLWP @XMLLNK      USE XML LINKAGE
       DATA CFI          TO CONVERT VARIABLE TO INTEGER
       MOV  @FAC,R12     R12 HAS LOW NUMBER
       INC  R1           POINT TO SECOND PARAMETER
       BLWP @NUMREF      GET SECOND PARAMETER (HIGH END OF DESIRED RANGE)
       BLWP @XMLLNK      USE XML VECTOR
       DATA CFI          TO CONVERT TO INTEGER NUMBER
       MOV  @FAC,R13     R13 HAS HIGH NUMBER
```

```
        INC  R13          INCREMENT TO INCLUDE BOTH ENDS
        S    R12,R13      SUBTRACT LOW LIMIT FROM HIGH LIMIT
        LI   R4,28645     PUT A BIG NUMBER IN R4
        MPY  @>83C0,R4    MULTIPLY BY THE RANDOM NUMBER SEED
        AI   R5,31417     ADD A BIG NUMBER TO RESULT IN R5
        MOV  R5,@>83C0    PLACE THAT BACK AT SEED LOCATION
        CLR  R4           CLEAR R4 SO NUMBER IS RIGHT JUSTIFIED IN R4-R5 PAIR
        DIV  R13,R4       DIVIDE BY THE RANGE +1
        A    R12,R5       ADD THE LOWER LIMIT TO REMAINDER FROM INTEGER DIVISION
        MOV  R5,@FAC      MOVE THAT NUMBER TO FAC
        BLWP @XMLLNK      USE XML LINKAGE
        DATA CIF          TO CONVERT TO FLOATING POINT FORMAT
        INC  R1           POINT AT THIRD PARAMETER (VARIABLE FOR RANDOM NUMBER)
        BLWP @NUMASG      ASSIGN THE VALUE TO THE VARIABLE
        LWPI >83E0        LOAD UP GPL WORKSPACE
        CLR  @STATUS      CLEAR STATUS BYTE
        B    @>006A       RETURN TO GPL INTERPRETER
WS      BSS  32           OUR OWN WORKSPACE
ANYKEY  BYTE >20          THE SPACE CHARACTER VALUE
        END
```

==========

Pre-review: SCUDBUSTER, Harrison Software.

I have a pre-release copy of this simple shoot-em game and can report as follows-

Enemy missiles fall from above on random courses and you must shoot them down with your missile from screen bottom left. You have one missile to hit each enemy missile, miss and they hit and you get negative points, shoot them down for points. The aiming system is unique- you use your joystick to move a dot around the screen and your defence missile is launched at the dot. Unlike "Barrage" your missile does not explode at that fixed point, but keeps on going. Thus you do not have to target a specific future location of the enemy missile, merely aim for your defence missile's flight path to cross the flight path of the attack missile so the two missiles coincide.

Bruce Harrison is a first rate machine code programmer, and the quality of programming can be seen in the highest speed of the defence missile! You have an unlimited number of lives- missing a missile produces negative points! Missiles become mildly faster as the score increases until you hit the target score and end the game.

An interesting touch is that from pressing fire to the time your missile arrives at the marked point is virtually a constant time. This makes for interesting strategy- you can move the aiming dot to far screen right to make your defence missile speed really quickly even if the attack missile is at screen left- just so long as the missiles make contact with each other.

The present version is as mentioned pre-release and lacks the final polish, but appears at first sight to be a simple but playable shoot-em-down game.

Harrison Software can be contacted at:

5705 40th Place, Hyattsville, MD, USA, 20781.

In accordance with your committees views, this is neither a recommendation or otherwise of any supplier, which may only be expressed by the committee, but only a software review as permitted.

==========

Update on the scan of George in last issue- and the cartoon scan in this issue, timed to coincide with a new series on BBC TV- Star Trek, The Next Generation.

The scans have been done by Ray Kazmer of Sunnyvale, California, and the procedure now used is as follows:

Scan using IBM PC clone and clean up using FIRST PUBLISHER. Version 3 allows perfect ratio on the PC! Save in MAC format.

Transfer to TI99/4A using PC-TRANSFER, which despite a habit of scrambling pictures most of the time, is said to be faster than using a direct wire hook up with comms programs on both machines.

And once on the TI, the MAC format pics can be printed using PIX PRO and transferred to other formats such as TI Artist if size permits. O

continued from page 6

hold. The Chicago Fair will be on October 31 this year, at the same Elk Grove Holiday Inn site. Al Beard is working on a 2-pass assembler, called T-Assembler, for the Geneve. Myarc has been catching up on repairs, but it is not known whether they will resume production of the HFDC and Geneve. Comprodine has released a "Color Banner Maker" for the Star NX-1000 Rainbow and other compatible colour printers.

Mark Wacholtz has formed a new TI software company called Media Ware Software (2141 NW 64th Ave., Suite 15, Sunrise FL 33313-3950). Their offerings include Page Pro pictures of mythological beasts, a routine to print labels designed through TI-Artist, programs to convert CSGD to Instance format and TI-Artist fonts from Extended Basic, and Page Pro border fonts.

A committee of vendors at Fest West '92 proposed a set of standards to define equipment requirements for new hardware and software. These will be finalized at the Lima Fair. The proposals are as follows:

Level #1 is defined as TI-99/4A console, 32k memory expansion, cassette, and E/A 5 loader (E/A, Supercart, TI Writer, Multiplan, etc.) Level #2 is Level #1 plus RS232, DSSD disk drive and controller. Level #3 is Level #2 plus at least 128k of CPU RAM bankable at the >6000 space. And Level #4 is Level #3 plus 9938/58 VDP with 192k VDP RAM. (I do not get it- is Level #1 for those with 32k installed in the console, but no P-box? And why the jump to DSSD for Level #2?)

Harrison Software will release two more disks of MIDI music at the TICOFF show. They are the Two-Part and Three-Part Inventions by J.S. Bach. These files have been written so that they use the Piano voice on either Casio or Yamaha keyboards, and are in SNF format so that they can be easily modified to any voice.

Reviews of Fest West '92 state that ESD still did not have an operating prototype of their IDE controller but that they anticipate an April 15, 1992 release date. There is a mention of a sneak peek at version 3.0 of Midi Master, which is obviously still not in production. It is reported that the price of version 3.0 will be substantialy higher, but those who bought the earlier version were promised the upgrade at no additional cost. Western Horizon Technologies announced delivery and pricing of a new version of Digi-Port software and hardware, to be shipped through Bud Mills Services.

OPA announced a new EPROM for the Geneve that automatically boots it into TI mode without a disk. They also announced a new EPROM based ROS 9 series, rather than RAM based, for the Horizon 3000 Ramdisks.

The Taylor Company, newly founded by Chris Taylor, demonstrated the "aTI", "xTI" and "mTI", described as being respectively an advanced, expanded and multimedia version of the TI-99/A; no further description of what they are, but they are apparently based on RAMBO. Taylor also announced that his company is developing a concept to be known as "Concept 99", and has written core modules of a drawing program, a bit map font program, a dbase manager, a chess program, a scheduler, a word processor, and language tutor programs. To be marketable, they must be completed, tested, and have manuals written. In order to know where to concentrate his resources, he asks us to let him know for which program we will be willing to make a deposit. There is no mention of what the price might be, or what hardware will be needed to run the programs. O

continued from page 3

Packaging and postage charges:

	Surface	Airmail
1 to 2 Disks	$2.00	$2.00
3 to 9 Disks	$3.00	$3.80
10 to 20 Disks	$4.00	$5.00

O

TI-Base Tutorial #17

by Martin Smoley, North Coast 99ers USA


```
*
*                3LBLSWD/C
CLOSE ALL
USE TNAMES
  MOVE
 SET LSPACE=800
 CLEAR LOCAL
     SET HEADING OFF
     SET RECNUM OFF
  LOCAL WD N 3
 REPLACE WD WITH 1
WHILE .NOT. (EOF)
 WHILE (.NOT.(EOF)).AND.(WD<3)
 DISPLAY WD
   MOVE
  IF .NOT. (EOF)
   REPLACE WD WITH WD + 1
   ENDIF
  ENDWHILE
 DISPLAY WD
  DOCASE
   CASE WD = 3
    DO 3ACROSS
    BREAK
   CASE WD = 2
    DO 2ACROSS
    BREAK
   CASE WD = 1
    DO 1ACROSS
    BREAK
  ENDCASE
 REPLACE WD WITH 0
DISPLAY WD
ENDWHILE
     SET HEADING ON
     SET RECNUM ON
 CLOSE ALL
RETURN Copyright Martin A. Smoley 1990
*
```

This month I cannot seem to find the time to write a wordy tutorial, so I threw this together. I find it much easier to whip up some CFs then to write a long explanation of some programming function. I have had several questions, over time, about printing two or three labels across. Because there are many new users of TI-Base I am going to take another shot at it. This set of CFs will print three, two or one across with some modifications.

Actually, this set of CFs prints three, two, or one label across at the end of the Db, depending on where the (EOF) falls, but I will get into that later. Let's start at the top with 3LBLSWD. This is the main CF. It drives the other three CFs. As you can see I am using my favourite Db, TNAMES. You should recall from last month that record zero in TNAMES contains some heading data. Because I do not want to make a label of that, the first command after the USE is MOVE. This gets me to the first real name and address. This set of CFs will need some extra room for storage so I am expanding the local space from the normal 256 to 800. The next phrase (CLEAR LOCAL) is required by TI-Base to register the SET LSPACE=800. Without the CLEAR LOCAL, LSPACE will remain set at 256. This task should be done as close to the beginning of your CF as possible, because CLEAR LOCAL also does what it says. When invoked it throws away everything you previously had in every local.

```
*                3ACROSS/C
MOVE -2
LOCAL L1A C 35
REPLACE L1A WITH "          ";
  | "   Exp. Date: " | XP
LOCAL L2A C 35
REPLACE L2A WITH TRIM(FN) | " ";
        | MI | TRIM(LN)
LOCAL L3A C 35
REPLACE L3A WITH SA
LOCAL L4A C 35
REPLACE L4A WITH TRIM(CT) | ", ";
        | ST | ". " | ZP
 MOVE
LOCAL L1B C 35
REPLACE L1B WITH "          ";
  | "   Exp. Date: " | XP
LOCAL L2B C 35
REPLACE L2B WITH TRIM(FN) | " ";
        | MI | TRIM(LN)
LOCAL L3B C 35
REPLACE L3B WITH SA
LOCAL L4B C 35
REPLACE L4B WITH TRIM(CT) | ", ";
        | ST | ". " | ZP
 MOVE
LOCAL L1C C 35
REPLACE L1C WITH "          ";
  | "   Exp. Date: " | XP
LOCAL L2C C 35
REPLACE L2C WITH TRIM(FN) | " ";
        | MI | TRIM(LN)
LOCAL L3C C 35
REPLACE L3C WITH SA
LOCAL L4C C 35
REPLACE L4C WITH TRIM(CT) | ", ";
        | ST | ". " | ZP
PRINT (f)
PRINT L1A,L1B,L1C,(LF)
PRINT L2A,L2B,L2C
PRINT L3A,L3B,L3C
PRINT L4A,L4B,L4C,(LF)
RETURN Copyright Martin A. Smoley 1990
*

*                2ACROSS/C
MOVE -1
LOCAL L1A C 35
REPLACE L1A WITH "          ";
  | "   Exp. Date: " | XP
LOCAL L2A C 35
REPLACE L2A WITH TRIM(FN) | " ";
        | MI | TRIM(LN)
LOCAL L3A C 35
REPLACE L3A WITH SA
LOCAL L4A C 35
REPLACE L4A WITH TRIM(CT) | ", ";
        | ST | ". " | ZP
 MOVE
LOCAL L1B C 35
REPLACE L1B WITH "          ";
  | "   Exp. Date: " | XP
LOCAL L2B C 35
REPLACE L2B WITH TRIM(FN) | " ";
        | MI | TRIM(LN)
LOCAL L3B C 35
REPLACE L3B WITH SA
LOCAL L4B C 35
REPLACE L4B WITH TRIM(CT) | ", ";
        | ST | ". " | ZP
PRINT (f)
PRINT L1A,L1B,(LF)
PRINT L2A,L2B
PRINT L3A,L3B
PRINT L4A,L4B,(LF)
RETURN Copyright Martin A. Smoley 1990
*
*                1ACROSS/C
LOCAL L1A C 35
REPLACE L1A WITH "          ";
  | "   Exp. Date: " | XP
LOCAL L2A C 35
REPLACE L2A WITH TRIM(FN) | " ";
        | MI | TRIM(LN)
```

```
LOCAL L3A C 35
REPLACE L3A WITH SA
LOCAL L4A C 35
REPLACE L4A WITH TRIM(CT) | ", ";
        | ST | ". " | ZP
PRINT (f)
PRINT L1A,(LF)
PRINT L2A
PRINT L3A
PRINT L4A,(LF)
RETURN Copyright Martin A. Smoley 1990
*
```

So if you have something in TEMP1, TEMP2, etc. and you do a CLEAR LOCAL, poof, it is all gone. The rest of this CF is very confusing. I understand what I am doing, but I still get confused every time I think about it for more then a short time. I know that some of you will look at this section and not be confused, but for those of you who are confused, I wanted you to know that you are not alone.

When we get to the End Of the File (EOF), there may only be one or two labels to be printed, not three. "I know you think (Who cares!), but this is a programming tutorial." Therefore, after TIB does a complete print of 3 labels across, I want to see how many names are left before the EOF. "Without following my CF, think about that for a minute." OK, let's go. The big, or outside WHILE is simple. It keeps looping until it finds the EOF. The big WHILE keeps the whole CF going. The little WHILE loop is looking ahead for the EOF. WD is the local I chose to hold the number of labels across or to the EOF. Because TIB is already looking at a record we will start with 1, hence REPLACE WD WITH 1. If we make 2 MOVEs, that will mean we have encompassed 3 records, the one we are looking at and two more. The WHILE reads, WHILE (.NOT.(EOF)).AND.(WD<3). This means that the largest WD can be and still go through this WHILE is 2, but if WD equals 2 going in it will still be 3 coming out.

What I am actually after is the knowledge that we have 3 records and that WD equals 3. So let's say WD equals 1 as we move into the WHILE. "I DISPLAY WD so I can see what it is." The MOVE tries to take us to the next record. If there is another record, we will not hit the EOF and 1 will be added to WD. WD now equals 2. As we hit the ENDWHILE we loop back to the WHILE statement. Because WD equals 2 we go through the WHILE one more time. Inside the WHILE we MOVE to the next record and if we have not hit the EOF, one more is added to WD. WD now equals 3 so the WHILE will not execute again. Instead we drop down to the DOCASE. "I used DOCASE instead of IFs because I needed some practice with DOCASE." Because WD equals 3 the first CASE will DO 3ACROSS. Because we have MOVEd twice in the process of getting here the first thing to do in 3ACROSS is back up 2 MOVEs to where we started. The rest of 3ACROSS merely sets up locals, gathers up the names and addresses and prints the labels, three across. The line near the bottom "PRINT (f)" should be removed. I use it to set my printer to Condensed Mode. That way I can test this CF on regular paper. I do not have three across labels. After the print we return the the DOCASE and fall through to the ENDCASE where WD is set to 0. The ENDWHILE sends us back to the first WHILE and because we have not hit the EOF we go through again. This time we are still looking at the last record that we printed, not the first record to be printed, that is why we set WD to 0 instead of 1. This means that while we are inside the WHILE we will go forward three MOVES, and while printing we will drop back two. That keeps us in the grouping of three records we wanted.

Now let's say we printed thousands of labels (three across), and we approach the EOF. We finished a print and there are two records left. We hit the WHILE with WD set at 0. The first MOVE moves us off the old group of three to the new group and does not really mean anything, but WD is incremented to 1. Remember we had 2, that is the one we are looking at and only one more. WD now says 1. We go around one more time. We move to the next record, but we have not smashed into the EOF, so the IF statement adds 1 to WD, which now says 2. We go around one more time even though there are no more records. We enter the WHILE and try to MOVE. Now we see the EOF. This shuts off the IF and 1 is not added to WD. WD remains at 2. Therefore, when we go through the DOCASE, CASE 2 will be taken and 2ACROSS will be run. 2ACROSS will back up only one MOVE and two labels will be printed. As TIB goes back to the main CF we have hit the EOF and no more moves will be made. This EOF finder will work with three, two, or one record remaining between the last print and the EOF. I left several DISPLAY WD statements in the CF to show where I had trouble. I actually took several more out to save page space. o

continued from page 5

If your RAMdisk loses data, you need to determine whether the problem is just the loss of ROS or whether all data is lost. If more than the ROS is being lost, the problem may well be to do with the batteries. If you have rechargeable batteries, it is possible for one of them to fail and cause problems which do not show up if you measure the voltage after it has been in the PEBox for a few minutes on power. If you lose more than the ROS, suspect the batteries. I run on non-rechargeable batteries which last for years. I have seen re-chargeable batteries which have emitted fumes or liquid which have attacked the tinned tracks of the PCB.

If you have a RAMdisk with 256K RAM chips and your batteries are going flat quickly, there may be a problem with the choice of manufacturer of the 74LS154 chips. The 256K chips do not have the second CE input (pin 26) that the 64K chips have so that the 256K CE inputs must have a resistor to the battery supply so that when the power is removed the CE inputs of the RAM chips are held at a high voltage and they are disconnected from the external circuitry. These CE inputs come from outputs of 4 line to 16 line decoder chips 74LS154. These chips can be a problem if they take current into their output when the power supply is 0 volts. ICs will normally do this unless they have special output circuits called open collector outputs. It turns out that most 74LS154 chips appear to behave in the way that we want, that is they do not let current into their outputs when power is removed, but I have found one that did and another one that had at least one output that did not behave. In these cases, not only will the extra current through the resistor on the CE input cause the batteries to discharge quickly, but the contents of the RAM are susceptible to being corrupted as CE will be low and WR will also be low. This can be tested by measuring the voltage across the resistors themselves when the power is off. This should be 0 volts indicating no current flow.

If you are just having problems with the ROS, you might try large capacitors on the 5 volt power supply and the battery power supply as well as the reset circuitry for the 74LS259. If you are still having problems then perhaps you should consider adding my little circuit to the WE line.

For further reading on my tussles with RAMdisks see TND Volume 9 Number 2 page 6, TND Volume 9 Number 3 page 29, TND Volume 10 Number 1 page 5, TND Volume 10 Number 2 page 5 and TND Volume 10 Number 7 page 19. o

continued from page 18

Now the fun begins. Move down to R6C1, enter the value 2, and Copy Right 13 cells. Next, move to R8C1, and enter the formula =R[-4]C*R[-2]C. Copy this right 13 cells. Now FORmat R8 just like R4, above. Now you can see some anomalies. In R8C5 10*2=23! In R8C7 13*2=25! And so on.....

For more fun move down to R10C1, and enter the formula =R[-2]C (you can do this by keying in =, moving the cursor up two cells, and hitting <enter>.) Copy this right 13 cells also. Since R10 values are equal to R8 values, you can now see what is actually in R8!

continued on page 24

STRACC routine for Reformatter+

by Bruce Harrison

```
        LIST
*    STRACC FOR JIM PETERSON
*    FOR SPECIAL FORM ACCEPT AT
*    25 MAR 1990
       DEF  ACCSTR
* CALL LINK("ACCSTR",ROW,COL,SIZE,STR.VAR)
* IF SIZE NOT NEEDED, SET AT ZERO
* SUBROUTINE WILL ALLOW 28 OR LESS
* DEPENDING ON ROW POSITION
* WILL AUTOMATICALLY DETECT AND ADJUST
* FOR 40 COLUMN MODE IF IN USE
KSCAN  EQU  >201C
VSBW   EQU  >2020
VMBW   EQU  >2024
VMBR   EQU  >202C
VSBR   EQU  >2028
STRREF EQU  >2014
STRASG EQU  >2010
NUMREF EQU  >200C
*
KEYADR EQU  >8374
KEYVAL EQU  >8375
GPLWS  EQU  >83E0
WS     EQU  >20BA
FAC    EQU  >834A
STATUS EQU  >837C
SCRMO  EQU  >83D4
GLRT   BSS  >2
CURORG DATA 0
ENDOC  DATA 0
SAV11  BSS  >14
STRING BSS  29
NOKEY  BYTE >FF
OFFSET BYTE >60
*
*
NUMASK DATA >0030
DECTEN DATA >000A
MRKR   DATA 0
ROW    DATA 0
COL    DATA 0
MAXLEN DATA 0
STRLEN BYTE 29
MAXMAX DATA 30
SCRLIM DATA 0
WID40  DATA 40
SCRWID DATA 32
GETOUT BYTE >BC
ENTERV BYTE >0D
LEFTV  BYTE >08
RITEV  BYTE >09
DELKEY BYTE 3
INSKEY BYTE 4
INSFLG DATA 0
GRMO   BYTE >E0
TEXMO  BYTE >F0
SPACE  BYTE >80
EDGE   BYTE >7F
ANYKEY BYTE >20
STOR1  DATA >0000
STOR2  DATA >0000
BLNKLN TEXT '                                '
       TEXT '                                '
ONOFF  DATA >0201
FORTY  DATA >0028
ALTKEY BYTE 0
       LIST
ACCSTR MOV  R11,@GLRT    MOV GPL REG 11
       LWPI WS           LOAD USER WS
       CLR  @KEYADR      MAKE SURE KEY-UNIT IS ZERO
       CB   @SCRMO,@TEXMO SEE IF WE'RE IN 40 CHAR MODE
       JNE  KEYRTN       IF NOT, LEAVE SCRWID AT 32
       MOV  @WID40,@SCRWID MAKE WIDTH 40
KEYRTN
       CLR  @STATUS      CLEAR GPL STATUS BYTE
       BLWP @GPLLNK      USE GPLLNK
       DATA >34          FOR BEEP TONE
       CLR  R0           NOT AN ARRAY
       LI   R1,1         FIRST PARAMETER
       CLR  @STATUS      CLEAR STATUS
       BLWP @NUMREF      GET NUMBER
       MOV  @FAC,R4      PLACE IN R4
       ANDI R4,>00FF     MASK OFF FP
       DEC  R4           INDEX ROW TO ZERO
       MOV  R4,@ROW      PLACE AT ROW
       INC  R1           NEXT PARAMETER
       CLR  @STATUS
       BLWP @NUMREF      GET PARAMETER
       MOV  @FAC,R4      MOV IT TO R4
       ANDI R4,>00FF     MASK OFF FP
       INC  R4           SET FOR ACCAT COLUMN 1, SCREEN COL 3
       MOV  R4,@COL      PLACE AT COL
       INC  R1           THIRD PARAMETER
       CLR  @STATUS
       BLWP @NUMREF      GET SIZE FOR STRING
       MOV  @FAC,R4      MOVE TO R4
       ANDI R4,>00FF     MASK TO INTEGER
       JNE  ACCITG       IF NOT ZERO, JUMP
       MOV  @MAXMAX,R4   ELSE DEFAULT
       JMP  ACCIT        THEN JMP OVER
ACCITG A    @COL,R4
       C    R4,@MAXMAX   CHECK FOR MAX
       JLT  ACCIT
       MOV  @MAXMAX,R4   ALLOWABLE (28)
ACCIT  S    @COL,R4
ACCITH MOV  R4,@MAXLEN   MAX LEN WILL NOT GO INTO SCREEN EDGE
       MOV  @ROW,R5      PREP FOR DISPLAY
       CLR  R6           OF STRING
       MPY  @SCRWID,R5   ON SCREEN
       MOV  R6,R0        AT ROW
       A    @COL,R0      AND COLUMN SPECIFIED
       MOV  R0,@CURORG   SAVE THAT SCREEN POSITION
       MOV  R0,@ENDOC    AND MAKE THAT ALSO LAST POSITION
       MOV  R0,R9        MOVE R0 TO R9
       A    @MAXLEN,R9   ADD MAXIMUM LENGTH
       DEC  R9           DECREMENT BY ONE
       MOV  R9,@SCRLIM   SET AS LIMIT FOR INPUT CHARACTERS
       LI   R1,4         NOW SET UP FOR
       CLR  R0           FOURTH PARAMETER
       LI   R2,STRING    AT LOCATION STRING
       MOVB @MAXLEN+1,@STRING  WITH MAXIMUM LENGTH SET
       CLR  @STATUS
       BLWP @STRREF      GET THE STRING
       MOVB @STRING,R2   TAKE LENGTH BYTE
       JEQ  KYNOF        IF LENGTH ZERO, JUMP
ACCITF SRL  R2,8         MAKE R2=STRING LENGTH
       LI   R9,STRING+1  SET R9 AT FIRST BYTE OF STRING
       MOV  @CURORG,R0   PUT CURORG IN R0
ACCITC MOVB *R9+,R1      MOVE ONE BYTE TO R1
       AB   @OFFSET,R1   ADD BASIC OFFSET
       BLWP @VSBW        WRITE TO SCREEN
       INC  R0           MOVE TO NEXT SCREEN POSITION
       DEC  R2           DECREMENT R2
       JNE  ACCITC       IF NOT ZERO, JUMP BACK FOR ANOTHER
       MOV  @CURORG,R0   ELSE SET UP AT ORIGIN
KYNO   BL   @KEYACC      AND ACCEPT A KEYSTROKE
       CB   @KEYVAL,@ENTERV  WAS ENTER KEY STRUCK?
       JNE  KYNOA        IF NOT, JUMP
       MOVB @ALTKEY,R1   ELSE WRITE BYTE BACK
       BLWP @VSBW        TO SCREEN
       B    @GOBACK      AND GET OUT, LEAVING DEFAULT STRING IN PLACE
KYNOA  MOVB @KEYVAL,@ALTKEY SET STRUCK KEY IN ALTKEY
KYNOF  MOVB @SPACE,R1    MOVE OFFSET SPACE CHAR TO R1
       MOV  @MAXLEN,R4   SET R4 TO MAX LENGTH
       MOV  @CURORG,R0   SET R0 TO ORIGIN
KYNOB  BLWP @VSBW        WRITE A SPACE TO SCREEN
       INC  R0           MOVE TO NEXT CHARACTER POSITION
       DEC  R4           DECREMENT COUNT
       JNE  KYNOB        IF NOT ZERO, WRITE AGAIN
       MOV  @CURORG,R0   FINISHED, BACK TO ORIGIN
       MOVB @STRING,R4   CHECK STRING LENGTH
       JEQ  KYN1         IF ZERO, JUMP AHEAD
KYNOG  MOVB @ALTKEY,R1   ELSE PUT STRUCK KEY
       AB   @OFFSET,R1   ON SCREEN AT ORIGIN
       BLWP @VSBW
       INC  R0           MOVE TO NEXT SPOT
       MOV  R0,@ENDOC    MAKE ENDOC=THAT SPOT
KYN1   BL   @KEYACC      GO GET A KEYSTROKE
       CB   @KEYVAL,@ENTERV IS THAT ENTER KEY?
       JNE  CMP0C        IF NOT, JUMP AHEAD
       MOVB @ALTKEY,R1   ELSE WRITE ALTKEY TO R1
       BLWP @VSBW        AND PUT IT ON SCREEN
       MOV  @ENDOC,R2    NOW MOVE ENDOC TO R2
       S    @CURORG,R2   AND SUBTRACT CURORG
       JEQ  KYN1C        IF THAT'S ZERO, JUMP
       MOV  R2,R4        ELSE PUT R2 IN R4
       MOV  @CURORG,R0   AND GET BACK TO ORIGIN
       LI   R9,STRING+1  SET R9 TO STRING CONTENT
KYN1B  BLWP @VSBR        READ CHARACTER FROM SCREEN
       SB   @OFFSET,R1   SUBTRACT BASIC OFFSET
       MOVB R1,*R9+      MOVE THAT BYTE TO BYTE POINTED BY R9, AND INCREMENT R9
       INC  R0           INCREMENT R0
       DEC  R4           DECREMENT COUNTER
       JNE  KYN1B        IF NOT ZERO, BACK
       CB   R1,@ANYKEY   SEE IF LAST BYTE WAS SPACE
       JNE  KYN1C        IF NOT, JUMP
       DEC  R2           ELSE DECREASE COUNT BY ONE
KYN1C  SWPB R2           SWAP BYTES
       MOVB R2,@STRING   MOVE NUMBER OF CHARACTERS TO STRING
       JMP  PUTSTR       JMP TO STRING ASSIGNMENT
CMP0C  CB   @KEYVAL,@INSKEY   ARE WE ENTERING INSERT MODE
       JNE  CMP9         IF NOT, JUMP AHEAD
       C    R0,@ENDOC    ELSE, CHECK WHETHER WE'RE AT ENDOC
       JEQ  CMP0B        IF SO, JUMP
CMP0A  INC  @INSFLG      ELSE SET FLAG FOR INSERT MODE
CMP0B  MOVB @ALTKEY,R1   MOVE ALTERNATE CHAR TO R1
       BLWP @VSBW        WRITE TO SCREEN
       JMP  KYN1         GO BACK FOR NEW KEYSTROKE
CMP9   CB   @KEYVAL,@DELKEY HAS DELETE BEEN STRUCK?
       JNE  CMP1         IF NOT JUMP AHEAD
       C    R0,@ENDOC    ARE WE AT END OF ENTRY
       JLT  CMP1A        IF LESS, JUMP
       JEQ  CMP1A        IF EQUAL, JUMP
       MOVB @ALTKEY,R1   ELSE WRITE CURRENT CHARACTER
       BLWP @VSBW        TO SCREEN
       CLR  @INSFLG      CLEAR INSERT FLAG
       JMP  KYN1         GO BACK FOR NEXT KEY
CMP1A  CLR  @INSFLG      CLEAR INSERT FLAG
       B    @DELCHR      DELETE CHARACTER AT CURSOR POSITION
CMP1   CB   @KEYVAL,@GETOUT HAS FUNCTION-0 BEEN STRUCK
       JEQ  GOBACK       IF SO, ABORT PROCESS
       CB   @KEYVAL,@LEFTV HAS LEFT ARROW BEEN STRUCK
       JNE  CMP2         IF NO, NEXT COMPARE
CURLFT MOVB @ALTKEY,R1   ELSE WRITE CHARACTER TO SCREEN
```

```
        BLWP @VSBW
        DEC  R0             MOVE BACK ONE SPACE
        C    R0,@CURORG     ARE WE AT ORIGIN?
        JLT  CMP2A          IF LESS, JUMP
        B    @KI2A          ELSE BRANCH TO SPECIAL INPUT ROUTINE
CMP2A   INC  R0             CANCEL LEFT MOVE
        B    @KYN1          LOOK FOR ANOTHER KEY
CMP2    CB   @KEYVAL,@RITEV    RIGHT ARROW KEY
        JNE  CMP3           NO, JUMP AHEAD
CURRIT  MOVB @ALTKEY,R1     ELSE WRITE CHARACTER
        BLWP @VSBW          TO SCREEN
        INC  R0             MOVE TO NEXT POSITION
        C    R0,@SCRLIM     IS THAT LIMIT?
        JGT  CMP4A          IF GREATER, JUMP
CMP4    C    R0,@ENDOC      END OF ENTRY?
        JGT  CMP4A          IF GREATER, JUMP
        JMP  CMP5           ELSE NEXT COMPARE
CMP4A   DEC  R0             CANCEL MOVE
        B    @KYN1          GO GET NEXT KEYSTROKE
CMP5    B    @KI2A          TO SPECIAL KEY-IN
CMP3    MOV  @INSFLG,R7     ARE WE IN INSERT MODE?
        JEQ  CMP8           IF NOT, NEXT COMPARE
        B    @INSERT        ELSE INSERT KEYSTROKE
CMP8    MOVB @KEYVAL,R1     MOVE KEYSTROKE TO R1
        AB   @OFFSET,R1     ADD BASIC OFFSET
        BLWP @VSBW          WRITE TO SCREEN
        INC  R0             MOVE NEXT SPOT
        C    R0,@SCRLIM     IS IT LIMIT?
        JLT  CMP6           JUMP IF LESS
        JEQ  CMP6           OR EQUAL
        MOV  R0,@ENDOC      PUT R0 AT ENDOC
        DEC  R0             DECREMENT SPOT
        JMP  CMP7           JMP AHEAD
CMP6    C    R0,@ENDOC      IS THIS END OF ENTRY?
        JLT  CMP7           IF LESS, JMP
        MOV  R0,@ENDOC      ELSE PUT R0 AT ENDOC
CMP7    B    @KYN1          AND GO GET NEXT KEYSTROKE
PUTSTR  CLR  R0             CLEAR R0, NOT ARRAY
        LI   R1,4           SET FOURTH PARAMETER
        LI   R2,STRING      R2 POINTS TO LOCATION OF STRING
        CLR  @STATUS        CLEAR GPL STATUS
        BLWP @STRASG        ASSIGN THE STRING VALUE
GOBACK  CLR  @STATUS        CLEAR GPL STATUS
        CLR  @INSFLG        CLEAR OUR INSERT FLAG
        LWPI GPLWS          LOAD GPL WORKSPACE
        B    @>006A         BRANCH TO GPL INTERPRETER
KEYACC  BLWP @VSBR          READ CHARACTER AT R0 POSITION
        MOVB R1,@ALTKEY     STASH AT ALTKEY
        LI   R1,>1E00+>6000 PUT CURSOR CHAR IN R1
        BLWP @VSBW          WRITE CURSOR TO SCREEN
        LI   R4,0200        PRESET R4 FOR CURSOR ON PERIOD
KI2     CLR  @STATUS        CLEAR THE GPL STATUS BYTE
        BLWP @KSCAN         SCAN THE KEYBOARD
        LIMI 2              ENABLE INTERRUPTS
        LIMI 0              VERY BRIEFLY
        DEC  R4             DECREMENT R4, THEN CHECK FOR R4=0
        JEQ  CHNG           AND JUMP TO CHNG IF R4=0
        CB   @ANYKEY,@STATUS  CHECK TO SEE IF A KEY HAS
        JNE  KI2            BEEN STRUCK.  IF NOT, GO BACK TO KI2
        RT                  IF KEY STRUCK, RETURN TO CALLING PGM
CHNG    CI   R1,>1E00+>6000 SEE WHETHER CURSOR IS IN R1
        JEQ  L1             IF SO, JUMP TO L1
        LI   R1,>1E00+>6000 OTHERWISE, PUT CURSOR IN R1
        BLWP @VSBW          WRITE CURSOR TO SCREEN
        MOVB @ONOFF,R4      PUT >0200 IN R4
        JMP  KI2            GO BACK TO KI2
L1      MOVB @ALTKEY,R1     PUT THE SCREEN'S PREVIOUS CHAR IN R1
        MOVB @ONOFF+1,R4    LOAD R4 WITH >0100
        BLWP @VSBW          WRITE THE CHARACTER TO SCREEN
        JMP  KI2            GO BACK TO KI2
KI2A    BLWP @VSBR          READ CHARACTER FROM SCREEN
        MOVB R1,@ALTKEY     MOVE TO ALTKEY
        LI   R1,>1E00+>6000 PUT CURSOR
        BLWP @VSBW          ON SCREEN
        LI   R5,>0080       LOAD SPECIAL DELAY COUNT
        CLR  @INSFLG        CLEAR INSERT FLAG
KI2B    CLR  @STATUS        CLEAR GPL STATUS
        BLWP @KSCAN         SCAN KEYBOARD
        CB   @KEYVAL,@NOKEY NO KEY STRUCK
        JEQ  KI2C           IF SO, JUMP
        LIMI 2              SET INTERRUPTS
        LIMI 0              CLEAR INTERRUPTS
        DEC  R5             DECREMENT COUNTER
        JNE  KI2B           NOT ZERO, BACK
        CB   @KEYVAL,@LEFTV IS LEFT ARROW HELD DOWN
        JLT  KI2C           IF LESS, JUMP
        CB   @KEYVAL,@RITEV RIGHT ARROW
        JGT  KI2C           IF GREATER, JUMP
        MOVB @KEYVAL,R5     ELSE MOVE KEY TO R5
        SWPB R5             IN LOW BYTE
        AI   R5,-8          ZERO-BASE VALUE
        SLA  R5,1           DOUBLE IT
        MOV  @CURTBL(R5),R4 MOVE TABLE VALUE TO R4
        B    *R4            BRANCH TO LOCATION IN R4
KI2C    MOVB @ALTKEY,R1     MOVE CHARACTER TO R1
        BLWP @VSBW          WRITE TO SCREEN
        B    @KYN1          GO BACK FOR NEW KEY
CURTBL  DATA CURLFT,CURRIT
INSERT  C    R0,@SCRLIM     ARE WE AT LIMIT?
        JLT  INSO           IF LESS, GO ON
        CLR  @INSFLG        ELSE NO INSERT
        B    @CMP3          GO BACK
INSO    MOV  R0,@STOR1      SAVE CURRENT POSITION
        MOV  @SCRLIM,R0     MOVE LIMIT TO R0
        DEC  R0             DECREMENT BY ONE
        BLWP @VSBR          READ THE CHARACTER
INS3    INC  R0             JMP AHEAD
        BLWP @VSBW          WRITE CHARACTER
        DEC  R0             MOVE BACK ONE
        C    R0,@STOR1      ARE WE AT ORIGINAL POSITION?
        JEQ  INS1           IF SO, MOVE ON
        DEC  R0             DEC AGAIN
        C    R0,@STOR1      IS THIS ORIGINAL POSITION
        JEQ  INS1           IF SO MOVE ON
        BLWP @VSBR          WRITE CHARACTER
        JMP  INS3           JMP BACK FOR ANOTHER
INS1    MOVB @KEYVAL,R1     PUT CURRENT KEY IN R1
        AB   @OFFSET,R1     ADD BASIC OFFSET
INS2    BLWP @VSBW          WRITE TO SCREEN
        INC  R0             MOVE TO NEXT SPOT
        MOVB @ALTKEY,R1     SET ALTKEY IN R1
        BLWP @VSBW          WRITE TO SCREEN
        C    @ENDOC,@SCRLIM IS END AT LIMIT
        JGT  INS4           IF GREATER, JMP
        INC  @ENDOC         ELSE INCREMENT END OF ENTRY
INS4    B    @KYN1          GO BACK FOR NEXT KEY
DELCHR  MOV  R0,@STOR1      SAVE CURRENT POSITION
DEL1    INC  R0             MOVE TO NEXT POSITION
        C    R0,@SCRLIM     IS THAT THE LIMIT
        JGT  DELEX          IF GREATER, JMP
        BLWP @VSBR          ELSE READ CHARACTER
        DEC  R0             AND DECREMENT POSITION
        BLWP @VSBW          WRITE CHARACTER
        INC  R0             MOVE AHEAD
        JMP  DEL1           DO IT AGAIN
DELEX   MOVB @SPACE,R1      BEYOND LIMIT, PUT SPACE
        DEC  R0             ON PREVIOUS SPOT
        BLWP @VSBW          HERE
        MOV  @STOR1,R0      NOW GET OLD POSITION BACK
        DEC  @ENDOC         DECREMENT END OF INPUT
        B    @KYN1          BACK FOR NEXT KEY
*  GENERAL PURPOSE GPL LINK
*  FOR USE UNDER EXTENDED BASIC
*  ON TI-99/4A ONLY!
*  USED HERE ONLY TO SUPPLY BEEP SOUND!
GR4     EQU  GPLWS+8
GR6     EQU  GPLWS+12
STKPNT  EQU  >8373
LDGADD  EQU  >60
XTAB27  EQU  >200E
GETSTK  EQU  >166C

GPLLNK  DATA GLNKWS
        DATA GLINK1
RTNAD   DATA XMLRTN
GXMLAD  DATA >176C
        DATA >50
GLNKWS  EQU  $->18
        BSS  >08
GLINK1  MOV  *R11,@GR4
        MOV  *R14+,@GR6
        MOV  @XTAB27,R12
        MOV  R9,@XTAB27
        LWPI GPLWS
        BL   *R4
        MOV  @GXMLAD,@>8302(R4)
        INCT @STKPNT
        B    @LDGADD
XMLRTN  MOV  @GETSTK,R4
        BL   *R4
        LWPI GLNKWS
        MOV  R12,@XTAB27
        RTWP
        END
```

continued from page 22

Multiplan also has an integer function like that of Basic. We will look at that now. Move down to R12C1 and enter this formula: =INT(R[-2]C)- by keying =, INT, (up arrow, up arrow, <enter>. Copy this right 13 cells (yes AGAIN!). Now observe the differences between R8 and R12! R12(using the INT function) has values just like those you would get in BASIC - everything to the right of the decimal point has been dropped, not rounding up or down! This is quite a significant difference when doing calculations.

Of course, the same errors can appear any time you use FORmat to display numbers in a specific format and then use arithmetical operators on them. Always use caution in setting up a worksheet to make sure of the type of numbers you will be working with and be discrete in your formatting to avoid this type of problem. Remember, a spreadsheet is just a tool - it is up to the user to make sure that the tool is used properly and that the implication of operator decisions are fully understood.

So much for Steve Zimmerman's article. Quite a scary story, but take heart: Multiplan can do things right. Steve is right that the indiscriminate use of the FORmat options CAN MAKE THINGS LOOK RIGHT WITHOUT BEING RIGHT.

continued on page 27

Beginning Forth - part 17

by Earl Raguse, UGOC, CA USA

MORE ON FILE HANDLING and SORTING

Last time I showed how to sort a file which was written on a screen, but I did not say how to rewrite the results back to another screen, because I did not know. However, I knew I could figure out how to write the sorted data back to the screen it came from or any other for that matter. That actually turned out to be fairly easy.

It occurred to me that since there is the word SCOPY to move screens around, why could the same techniques not be used to write a file back to the source disk using FLUSH. The trouble was that FLUSH is a resident word so its definition is not shown on a screen. SCOPY on TI System Screen #39 was not too helpful either, but it also used BLOCK which I had used in GETFILE last month, to copy the data screen to the NAMELIST file. It occurred to me I could also use BLOCK to get the address of the disk buffer for the screen I wanted to write to. Then all that would be necessary was to move the NAMELIST file to the disk buffer address, then FLUSH it to disk. It works! The word is WRITESCReen or WS on Screen #55. Although I gave it to you last time I did not mention it because supposedly I had not invented it yet. WRITESCReen permits us to write the sorted data to any screen we choose. Because of the way BLOCK works if we specify any screen number followed by BLOCK, that screen will be loaded and its address in the disk buffer space will be put on the stack. Then after clearing that buffer with CB (ClearBuffer), note that because 64 is added to the BLOCK address, the first line is not cleared so any title you may have there is still intact; we then bytewise move our file to that address using CMOVE, mark it UPDATEd and FLUSH it to disk.

It seems that we now have complete system for creating screen files using the EDITOR, transferring them to memory, for sorting or whatever and writing them back to disk were they can be further modified using the EDITOR, or accessed by TIWriter etc. via WRITE-V80 discussed previously.

As you know its easy to output screens to your printer. But what if you want get rid of the line and screen numbers? Although not discussed last time, I also included the word TYPESCRN abbreviated to TS, shown on Screen #55 to do just that. Also discussed briefly last time is MAKEFILE abbreviated to MF for direct access to the EDITOR.

This is still a primitive file handling program. The file is limited to one screen of 15 records, no provisions are made for displaying the complete record to the CRT other than with the EDITOR.

No provisions are made for selecting the sort field, and records are limited to 64 bytes, but all the necessary principles are illustrated. Repairing these deficiencies requires major modifications which would make the main principles of the procedure more obscure and difficult to follow and would not well serve the intended purpose of instruction on the use of Forth.

Longer files would require multiple screens and the accompanying housekeeping tasks of computing record numbers, counting screens etc. Longer records reduces the number of records per screen and a universal format is probably impractical unless one arbitrarily used two lines per record and wasted the unused space and perhaps memory.

Also if the file gets long, it will not fit in memory all at one time, then it would be necessary to load only the sortfield plus the record number index into memory, then sort that along with the record number index. Then one could print the file by serial access from disk by using the record number index.

Further, by this time one should be thinking of using another sorting technique, like the Heap Sort or Shell-Metzner Algorithm which would be much faster on large files. Then too there is the problem of how to display the file on the CRT and how to print it on the printer, its hard to see how one could satisfy everybody or maybe anybody.

It seems that a more complex format needs to be customised by the user, hence he must securely grasp the principles herein to apply to his special needs. No sorting program is of value unless the sort field length and location can be defined by the user.

This is easy to implement using variables instead of numerical constants, but the program must ask for inputs every time or allow a choice of custom or default configurations. The latter is easy also since Forth requires one to initiate all variables with some value, generally zero, but any default could be used if one could figure out what the user might want.

There are so many tradeoffs between allowing easy operation with no choice to near infinite flexibility but having all the nuisance questions to be answered each time. One possible solution is to create a file of the configuration parameters and save them to disk using the techniques described in BFORTH #15. For me, the best answer is to make the program fit the problem, not massage the problem to fit the program. This however, requires more skill than many users have. An ancient dilemma, maybe that is why I am not a professional programmer. May the FORTH be with you. o

continued from page 4

Tigercub Software Releases

TCC-8 is another in Jim Peterson's Tigercub Collection. As with all the other disks, it is menu driven from Extended BASIC and contains the following items.

1 Rookie Outfielder
2 Pillbox and Tanks
3 Mice in a Maze
4 Fireflies (BASIC only)
5 Runaway Pig
6 Gleep Shoot, Zook Shoot
7 Antares
8 Aussie Fighter
9 Balloons
10 Beetle Walk
11 Bridge Guard
12 Chicken
13 Indy 500
14 Invasion
15 Lily Padder
16 Patscram Mission
17 Patscram Instructions

The catalogue of files on this disk is shown below:

TCC-8 Diskname: TIGERCUB Format: SSSD

Filename	Size	Type / Length	
*TC-101	18	PROGRAM	4180
*TC-37	12	PROGRAM	2622
*TC-65	22	PROGRAM	5368
*TC-94	19	PROGRAM	4378
*TCX-1034	17	PROGRAM	3960
*TCX-1093	21	PROGRAM	5025
ANTARES	21	PROGRAM	4885
AUSSIE	32	PROGRAM	7922
BALLOONS	15	PROGRAM	3471
BEETLE	32	PROGRAM	7935
BRIDGE	29	PROGRAM	6982
CHICKEN	17	PROGRAM	3894
INDY500	11	PROGRAM	2427
INVASION	13	PROGRAM	3065
LILYPAD	20	ROGRAM	4819
LOAD	7	PROGRAM	1374
PATSCRAM	41	PROGRAM	10239
PATSCRAM/I	7	PROGRAM	1518

o

Writing in Machine Code

Branching, Signed and Unsigned Numbers

by J.E. Banfield

In program execution it is often necessary to transfer processor control from one part of the program to another by replacing the current contents of the program counter (which normally points to be executed at .+2) by something else. TI classifies all such instructions as branches, comparable to meeting a branch when climbing up a tree.

Some such instructions always select one side of a branch fork, hardly branching, and I prefer to designate these instructions as jumps, which are always unconditional and replace the current contents of the program counter with the contents of the source address. Instructions with codes 068 and 044 are examples.

Other branches, namely instructions 10 to 1C differ in two respects, they may be conditional and they modify rather than replace the current contents of the program counter. I prefer to call such instructions skips.

The instruction 10 01 is an unconditional skip with a count of one. Accordingly, the next instruction is skipped and the processing resumes after that.

The instruction 10 00 always skips zero instructions. It is a no op, an instruction which does nothing. It is useful to allow a short delay, for example, in accessing the Video Processor chip which often requires a 2 micro-second delay.

The instruction 10 0C always skips the next twelve instructions. In each case, the count of the number of instructions to be skipped is given in the second byte of the instruction and this is interpreted by the processor as a signed number.

Signed and Unsigned Numbers

In its regularly perverse ways, Texas Instruments, in its 9900 Microprocessor Data Book, has an unusual use for "logical". Normally, as in data sheets for the 74LS181 arithmetic and logical unit (ALU), logical refers to those processes where carries are not propagated between adjacent bits, for example, AND, IOR, XOR, etc.

Thus, rather than use the term "logical greater than" (9900 Manual page 21), I prefer the term "unsigned number" for those in which the leftmost bit is considered the most significant bit of a positive number and to use the term "signed number" for those in which the leftmost bit is interpreted as the sign, thus I use "unsigned number greater than".

Consider the 4 bit numbers 0 to F. Regarded as unsigned numbers, the order of increasing magnitude is:
0 1 2 3 4 5 6 7 8 9 A B C D E F

However, as a signed number, the leftmost bit is zero (positive sign) for numbers 0 to 7 and is 1 (negative sign) for numbers 8 to F. Thus F has the value -1, because if we add one to F (in binary):

```
  1 1 1 1      =F

  0 0 0 1      =1
c 1 1 1
  0 0 0 0      =0
```

the result is zero (with a carry true). Similarly, C has the value -4 since:

```
  1 1 0 0      =C

  0 1 0 0      =4
c 1
  0 0 0 0      =0
```

Accordingly, the ascending order of 4 bit signed numbers is:

```
 8  9  A  B  C  D  E  F  0  1  2  3  4  5  6  7
-8 -7 -6 -5 -4 -3 -2 -1  0  1  2  3  4  5  6  7
```

Skipping Backwards

The 9900 processor interprets the second byte in a skip instruction as a signed number. If the leftmost bit of the second byte is a 1, the number of skipped instructions is negative. Twice the absolute number of the count is subtracted from the current program counter and we execute an instruction at a lower address, namely a skip backwards. If the count is -1 that is FF is the second byte of the skip instruction, we skip back to the same skip instruction and the processor may cycle indefinitely. We will make use of this to simulate a stop instruction in a later contribution by 10 FF. This is an unconditional skip (back) to the same address. Unless the console is fitted with a reset of load button, the only thing then is to switch off the power and start again.

The instruction, 10 FC causes the processor to skip back 4 instructions (to 3 before the instruction itself) unconditionally, since an 8 bit binary number (1 1 1 1 1 1 0 0 or FC) needs four additions of one to become zero and thus equals -4.

Conditional Skips

This is a complicated area not particularly clearly defined in the 9900 Manual (everything is there but it is not easy to follow). At this stage we will consider a simple example and leave the hard part until later.

In the console ROM (in fact in the KSCAN routine) you will find the instruction:
M037E C0 82
which (see prior article) interprets as move the contents of accumulator 2 to accumulator 2. At first sight one might consider the programmer to have lost his marbles with a senseless NO OP. But this is not so since the status bits in the CPU are set according to the number moved. The next instruction:
M0380 16 14
translates as SKIPNE 14. That is, skip if the number moved to Ac2 is not equal to zero. The count is 20 (decimal) and so if Ac2 was not zero, processing resumes at address M03AA rather than at M0382. A calculator with hexadecimal arithmetic is handy! o

continued from page 12

So you can now see how to use these string procedures and how to call them. Along the way I have introduced printf() and scanf() which are two very useful procedures for doing input and output to your program. I suggest you try and think up some more tests you might run for these procedures and even some tests for the remaining string procedures which I hope to cover next time. I must warn you that if you use the procedures originally circulated with c99 version 4, you may find some strange results. The versions that I have amended and presented to you last month should always do the correct thing.

Before I stop I must give you the scriptload file I used for this example. Remember that I am advocating that you use Funnelweb for processing the c99 source files after the compiler phase. This will allow the assembly and loading to be done with very little input from you. It will still take a while to get all the files from disk so that it will be much quicker if you can arrange to use a RAMdisk for all the files. I think that the purpose of all the files is obvious from their names, given that I have called my version of the program "SECONDC;C" and I stored the scriptload file as "SECONDC;T".

```
* Script for string test program
AUTO
FILE "DSK1.CSUP"
FILE "DSK1.SCANF"
FILE "DSK1.PRINTF"
FILE "DSK1.STRINGS"
FILE "DSK1.SECONDC;O"
LAST START.
```

o

15, 16

Regional Group Reports

Meeting Summary For JULY

Banana Coast	12/07/92 Sawtell
Central Coast	11/07/92 Saratoga
Glebe	09/07/92 Glebe
Hunter Valley	11/07/92?Boolaroo
Illawarra	13/07/92 Keiraville
Liverpool	10/07/92
Northern Suburbs	23/07/92
Sutherland	17/07/92 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 held regularily at the Boolaroo Ambulance Station. All welcome. Please contact Geoff Phillips on (049) 428 176 for details.

ILLAWARRA Regional Group

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

LIVERPOOL Regional Group

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

NORTHERN SUBURBS Regional Group

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

Dick Warburton.

SUTHERLAND REGIONAL REPORT

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

Peter young
Regional Co-ordinator

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.

JULY MEETING - 4TH JULY

This will be the second BUY SWAP and SELL day of the year. They used to be very popular but have not attracted much attention as of late. Let's have a good look around the house for any unused gear that may be of help to somebody else. Any new software from abroad may be demonstrated as well. See you there!

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

August	12 July
September	9 August

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

July

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

August

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

September

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

October

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

November

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

December

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

continued from page 24

The secret is to not only have things look right but also to be right is a Multiplan function called ROUND. ROUND puts you in complete control of numeric values, not only as to the number of decimals to display (or none if you key in "0" for number of decimals) but unlike the FORmat option, ROUND actually adjusts values in memory to your exact specifications!

Once you have keyed in Steve's example as shown above with all its peculiar results, go back to R4C1 and key in the new formula ROUND(R[-2]C,0) and then press <enter>. Again copy this right 13 cells. Recalc and see the changes.

The moral of the story, Multiplan has more power and more user control than is evident at first sight. For more information look up the ROUND function on page 180 of the Multiplan manual.

Yes, ROUND is a very useful function and whenever your results do not seem to "add up", go back and examine your formulas to see whether ROUND would not be appropriate in the right place. O