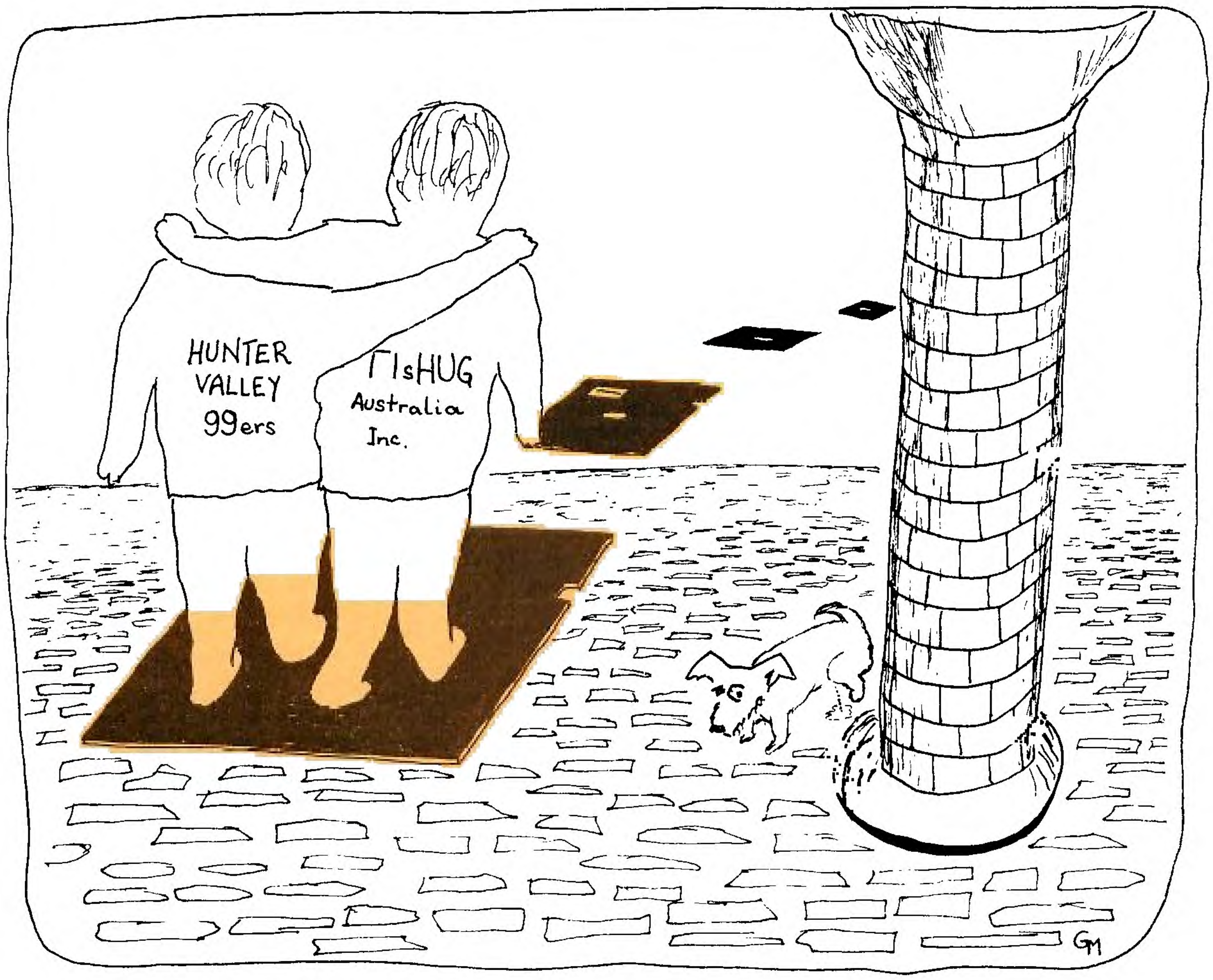
042
85424
TIsHUG

NEWS DIGEST

Focusing on the TI-99/4A Home Computer

Volume 7, Number 9 October 1988

Registered by Australia Post - Publication No. NBH5933



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

TIsHUG (Australia) Ltd.

TIsHUG News Digest

October 1988

All correspondence to:

P.O. Box 214
Redfern, NSW 2016
Australia

Membership and Subscriptions

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

Cover by George Meldrum

TIsHUG Sydney Meeting

The next meeting will be at 2 pm on 8th of October at Woodstock Community Centre, Church Street, Burwood.

Printed by
The University of Wollongong
Printery

TIsHUG News Digest ISSN 0819-1984

Index

They're off

by Geoff Trott

Stop press: I have just heard from Cyril Bohlsen that the Hunter Valley 99ers have invited TIsHUG members to their amusement day on Sunday 6th November. It will start with a car rally from Speers Point Park and finish in a games afternoon. If you want to go please give Albert Anderson a ring on (049)662602 to tell him and find out all the details.

****

Before I forget, I must say a big thanks to George Meldrum for the cover of last month's TND. Yes, that is correct, it was not Peter Schubert who did the cover, but one of his satisfied customers. George also has done this month's cover and I hope he will continue to find time for some more in the future. He was galvanised into action by my threat to have the same cover each month and instead of just complaining about it actually did something constructive. Good on you George. It was remiss of me to not acknowledge that it was his cover, but this time he has signed his work and you may have noticed the credit amongst the information on page 1.

continued on page 31

Secretary's Notebook

by Terry Phillips

This has been a very quiet month with next to no correspondence coming in and no new members to welcome either. This should start to liven up again soon as all the US groups come back on line after their summer holidays.

Unfortunately, there was no response from Inscebot in time for an announcement at the September meeting, but with a bit of luck there should be a response from them soon. It appears from my observations at least that there is tremendous interest in the TI-Base software, so the sooner it can be imported for members the better.

As you can see from the meeting information on the back page, the proposed trip to Newcastle has been put on hold, at least for the time being. I still feel that such a venture would be a very enjoyable outing and with a bit of forward planning we can all look forward to it early in 1989.

For a rundown on forthcoming activities, see Regional Group Reports.

That is it for this month. Should be more to report next time. o

Software Released to Date

by Terry Phillips

At the last meeting a couple of newer members asked for a list of what software had been issued during this year. While this has already been publicised in the individual monthly magazine issues, I thought that a recap on the topic would not go astray. So trusting the Editor has space this column is mainly aimed at newer members.

FEBRUARY, 1988:

40 COLUMN UTILITIES - Programmers assembly utility routines. Extended BASIC and 32K expansion required.
GRAPHIC LABELLER - Print out mailing labels with a picture. Extended BASIC 32K expansion and printer required.
RLE PICTURES (2 Disks) - A host of picture files for you to print out. Needs Extended BASIC, 32K expansion and a printer.
TAPE 1988/1 - Calculator, Cassette Sleeve Maker, City Attack, Fitz, Harried Housewife, Adjective/Adverb and Noun/Adjective. Also available on disk.

MARCH, 1988:

DISK A168 - Centipede, Diskcopy, Disk Doctor, Easy Designer, Disk Hacker, Christmas Card, Masscopy, Networth, Nibbler, Quick Copier, Turbo Copy, Banner, Bowling and Perplex. Mostly disk copiers but some games.
DISK A169 - A disk full of routines and utilities including Banners, Calendars, Gothic Print, Thank You notes plus a whole lot more.

Some programs on these disks require 32K expansion, while others just need Extended BASIC.

APRIL, 1988:

ROS MENU VERSION 7.3 - An updating operation system for RAMdisk users.
RLE PICTURES (3 Disks) - More pictures to print out.
CLOCK ROUTINES - Samples and demonstrations for the TIsHUG clock card designed by John Paine.

MAY, 1988:

PETERSON'S MUSIC DISK - Collection of popular songs, most with words to sing along to.
BEAXS - Latest updated version from Paolo Bagnaresi. BEAXS is an Editor/Assembler replacement and requires Extended BASIC and 32K memory expansion.
BASIC BUILDER - A utility to convert "list" files to Extended BASIC program format. Extended BASIC and 32K expansion.
BA WRITER - This is a TI-Writer substitute, also from Paolo Bagnaresi which requires Extended BASIC and 32K expansion.

JUNE, 1988:

C99 VERSION 4.0 - Latest version from the author Clint Pulley. "C" for those unsure is a programming language. Requires fully expanded system.
CATLIB VERSION 1.5 - Cataloguing library program from Marty Kroll. Has some excellent features and requires a fully expanded system.
CATLIB COMPANION - A companion disk to the main program with some added features.

COMPETITION GAMES - Tilo, Wait-a-Bit, Cathay, Triangle, Matchmaker and Matchmaker Animator and Family Tree. All require Extended BASIC.
COMPETITION WINNER - George Meldrum's MERGE utility.

JULY, 1988:

EXTENDED DISPLAY PACKAGE - Written by member Craig Sheehan. Some excellent routines on this disk from members dabbling with assembly language. Requires a full system.
DISKETTE CARETAKER - Written by member Tony Imbruglia and produces disk jackets with comments. Requires a full system.
FUNNELWEB VERSION 4.1 - Updated and enhanced version of this classic piece of software from the McGoverns. Requires full system.
DISK AID VERSION 2.0 - Utility sector reader/copier etc, from Garry Christensen. Requires full system.
GEE - A graphics programming language which requires a full system.

AUGUST, 1988:

TERR-WARE GAMES - Joker Poker, Blackjack and Wheel of Fortune. Very well programmed games which require Extended BASIC and 32K expansion.
WHEEL OF FORTUNE AND MONTE CARLO - Some more gambling games that also require Extended BASIC and 32K expansion.
TASS 2001 - TI Artist Slide Show. Enables you to display all your Artist pictures one after the other. Requires Extended BASIC and 32K expansion.
CHARACTER DESIGNER - One of the best of its type of utilities around with features not found on similar programs. Requires Extended BASIC and 32K expansion.

SEPTEMBER, 1988:

HIGH RES GRAPHICS - Design your own high resolution graphics. Also contains several demonstration files. Needs Extended BASIC and 32K expansion.
COMIC ANIMATOR - Do your own or just watch the demonstration files supplied. Extended BASIC and 32K expansion.
CARFAX ABBEY - Is the main program on the disk and it is an adventure of the Tunnels of Doom type. This main program requires Extended BASIC and 32K expansion. The disk also contains 3 more game programs, Boxes, Snakes and Ladders and Taskforce which require Extended BASIC only.
J P HODDIE GAMES DISK - Some well programmed assembly and Extended BASIC games mainly of the space shooting variety.

To the end of September, 34 disks have been issued. o

For Sale

Peripheral Expansion Box with two DSDD slimline disk drives, 32K Memory expansion card, TI disk controller, TI RS232 card, with all documentation. All for $800.
Microsoft Multiplan complete for $50
TI-Writer complete for $50
Ring Mike Shami (02)798 5127 for details and negotiations.

TIsHUG Software Column by Terry Phillips

Here is a list of the new disks added to the software library over the past few weeks.

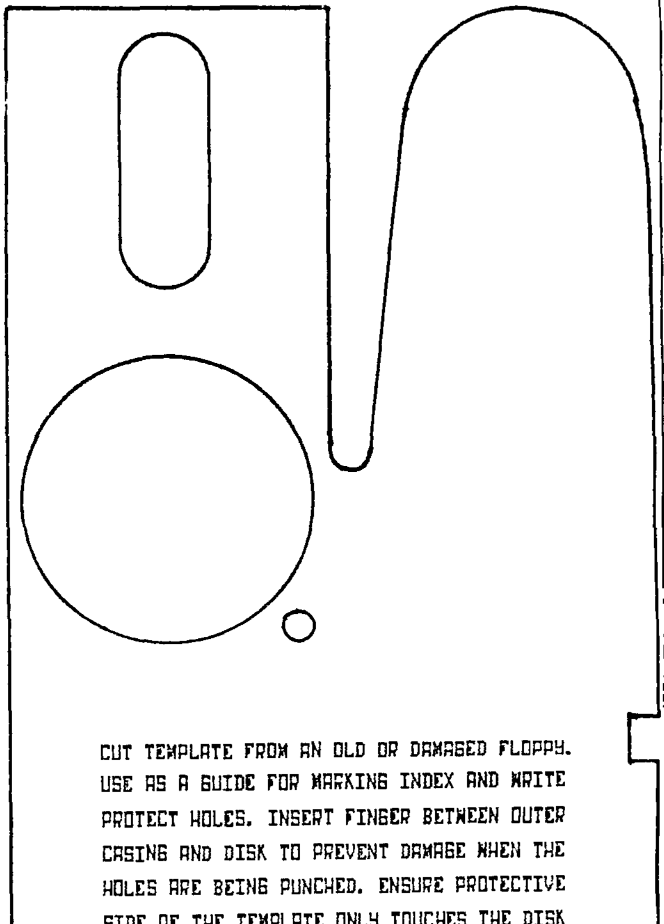
LIBRARY DISK A259 - ASSEMBLY GAMES: Astroblitz, Buzzard Bait, Cannonball, Cave Creatures, Compu-Car, Connect 4, ET Adventure at Sea, Angler Dangler, Galaxia and Willy Worm. All require 32K expansion and Editor/Assembler module or equivalent.
LIBRARY DISK A260 - The main program is an assembly game entitled Hopscotch, which is based on the Q-Bert type games. This jumping little fellow is a kangaroo who utters a 4 letter expletive when donged by one of the bouncing balls. Very good graphics and animation. A secondary program on the disk is WINDOS, a type of disk manager, but written entirely in German. Requires 32K and Editor/Assembler module.
LIBRARY DISK A261 - ASSEMBLY GAMES: Mathcatcher which appears incomplete but has excellent animation of a skate-boarder, Mission X a bomb and shoot everything in sight game, St Nick which is one for Christmas when you can help Santa gather up the toys, Star Trap which is only Star Wars under a different name, TI Mazog which is a sort of 3D Pacman but very difficult to play. 32K and Editor/Assembler module required.
LIBRARY DISK A262 - BITS AND PIECES: Perfect Push a difficult and frustrating game where the object is to assemble a rocket ship in the correct order, Hitchhikers Guide a D/V80 file that contains the solution to the adventure, an Extended BASIC and assembly version of the rapid loader for Infocom adventures which not only allows for much faster loading but alterations to screen and text colors to suit your taste. 32K and Editor/Assembler required.
LIBRARY DISK A263 - LEATHER GODDESSES OF PHOBOS a big Infocom adventure too large to run even in a fully expanded system without a super space module.
LIBRARY DISK A264 - LURKING HORROR: Another Infocom adventure where the object is to find and destroy the hideous creature. Like a Stephen King novel. 32K and Extended BASIC.
LIBRARY DISK A265 - SPELLBREAKER: Another Infocom adventure and the final chapter in the Enchanter and Sorcerer series. 32K and Extended BASIC.
LIBRARY DISK A266 - SIDEWRITER: An excellent utility for printing Multiplan spreadsheets and TI Writer files on their sides. Highly recommended. 32K and Editor/Assembler required.
LIBRARY DISK A267 - TUNNELS OF DOOM: Finally, the module is available on disk but appears only to load through the loaders supplied with Funnelweb. Apparently no Extended BASIC or Image type loaders can handle it.
LIBRARY DISK A268 - BIG TEXAS ADVENTURE: a spy type adventure with some 250 locations. Appears playable but fairly slow in set up and play.
LIBRARY DISK A269 - ADVENTURES: Incredible Hulk, Buckaroo Banzai, Adult Adventure (Warning - contains words and scenarios that may be offensive to some people), First Days In Eden, Doors to Eden, Discovery at June Lake, On The Loose, Lost Gold. The disk also contains solutions to Deadline, Infidel and Eye Witness. All these adventures run out of the Adventure Module or equivalent disk version.

As I will be on holidays over the next school holiday period, which runs into the October meeting date, there will be no specific disks issued at that meeting. To make up for this there will be a double issue at the November meeting and titles will be advised in the November TND.

At the Board meeting held on 3rd September it was agreed that the Group should act as an intermediary for Freeware/Shareware authors. What was decided was that the Group, through shop sales would collect on behalf of authors their suggested donation, that is, if an author suggests sending him $10, then that amount would be collected on top of the normal $5 disk fee. It should be noted that it is NOT compulsory for any member to pay the suggested authors fee, and if that person declines then the disk will be available for the normal $5 cost. On the other hand a lot of software authors suggest you only send the donation if you intend using the program, and of course you would not know this until you had it home and tried it out. Therefore, if you were to pick up a disk at one meeting, for the normal fee, take it home and find that it is exactly what you want, then the author's donation can be given at the next meeting. The idea behind all this is to encourage software authors to continue writing programs that will ultimately benefit the entire TI99/4A community. Give it some thought, any feedback would be welcome.

That is it for this month, but just to remind you that the standing offer of your selection of any of the programs in the library is still open, and I am encouraged by the number of members who have taken advantage of this. o

A Guide to Doubling Your Disk Storage for Single Sided Drives


CUT TEMPLATE FROM AN OLD OR DAMAGED FLOPPY.
USE AS A GUIDE FOR MARKING INDEX AND WRITE PROTECT HOLES. INSERT FINGER BETWEEN OUTER CASING AND DISK TO PREVENT DAMAGE WHEN THE HOLES ARE BEING PUNCHED. ENSURE PROTECTIVE SIDE OF THE TEMPLATE ONLY TOUCHES THE DISK

THE ORIGINAL HV99 FLOPPY FLIPPER FINGER

The Communicators

Special Interest Group for
Users of the TEXPAC BBS.
by Ross Mudie, 4th September 1988.

1. BBS CONSOLE FAILURE.

On 23rd August 1988 the BBS console failed at about 8am. The spare NTSC console had to be picked up and was promptly installed, allowing the BBS to be back in operation at 2.45pm on the same day. The cause of the problem was the failure of the TMS9901 chip in the console. A socket was installed for the replacement IC and the VDP oscillator choke replaced as a precaution. The felt oiling pad was removed from the GROM socket which was cleaned out, first with a cloth on a blunt knife blade and then with freon to remove any remaining particles of dirt.

2. CONTROL AND SPECIAL KEYS AND KEY SEQUENCES.

There are a number of users who are still unaware of the control functions available with files. To pause a file listing use <CTRL> S once only and the BBS will pause after the end of the current line. Remember that 80 column lines will occupy two 40 column screen lines so the BBS may seem to take two lines to stop listing. Use <CTRL> Q to restart the listing. To Escape from a file listing press E once only. The BBS will exit from the listing at the end of the current line.

After listing or Escaping from a file the BBS sends the prompt:
"[M]ain[#], [N]ews menu or News # >"
At this point the following actions are possible:

(a) M will give the Main Menu.
(b) M +a number will go directly to that numbered item in the Main Menu, e.g., M2 will go directly to the program download. The valid number is range 1 - 9.
(c) N will give the News Menu.
(d) A number will go directly to listing the News item corrosponding to that number in the News Menu, providing that the number is in the current range of items in the News Menu.

To correct a typing error into the BBS use CTRL[H] which is a destructive backspace. This also works in the non-echo mode used for passwords. CTRL[H] stops at the left hand end of the line and characters are deleted in the BBS regardless of what your terminal program leaves on your screen.

Full details of how to use the BBS are to be found in the file BBS_HELP in the NEWS menu.

3. CONTRIBUTING MATERIAL FOR THE BBS.

If you wish to contribute material for the BBS, then prepare your material as a text file and send it as mail to SYSOP on the BBS. Anyone wanting to contribute on a regular basis should contact the SYSOP so that a Sub-Editor name can be allocated. Sub-Editors can load their file into the NEWS menu without the need for any SYSOP action. Any BBS member can place programs in the Upload/Download Room for others to download. Programs are generally left in this area for a month for others to download and may be deleted by the uploader, SYSOP or the GAMES Sub-Editors.

4. BBS MEMBERSHIP.

Access to the BBS is limited to BBS members only. TIsHUG members wishing to join the BBS should contact the Secretary; membership rates are shown on page 1 of the TND magazine. o

From the Bulletin Board

MAIL TO : ALL
MAIL FROM : DIGIT
I have tried to use PRbase from a RAMdisk, and while it loads, I am not able to access the data disk drive either 1 or 2. It gives an error message. Does anyone know the reason why, and more importantly, how to fix it?

MAIL TO : ALL
MAIL FROM : SARA
KEVIN COX is in need of the Documentation for Super Extended BASIC. We believe that they were put out by J & K H Software.
Contact Kevin on BBS via SARA or call (066)52 3649. Any information will result in a disk swap.

MAIL TO : ALL
MAIL FROM : LARRY
I have just received a copy of Wheel of Fortune program from my friend Jerry in the USA. It is a super program that makes the TIsHUG program look 10th rate. It's wheel looks and works like the TV show with bankrupt, free spin, lose a turn. It has random speed with the computer slowing down and stopping the wheel. Two or three players can play. It has three rounds with the winner playing the bonus round that has a time limit to solve it. This program will be demonstrated with two other programs at least at the next Liverpool regional meeting. Bye for now Larry.

MAIL TO : ALL
MAIL FROM : PETESAKE
SUBJECT: HV99ers FAMILY WEEKEND OUTING.
I would like to suggest a change to the proposed date of October 1 for the outing. As this is during the school holidays I will be unable to attend as I have already made plans with the family. I have asked other members and found some in a similar predicament as family plans were made some time before. I would suggest the main topics be on Sunday with overnight accomodation details made available to those that travel up on Saturday. Perhaps a Saturday afternoon tour of the vineyards could be organised followed by a dinner in the evening at the local RSL or such.
Perhaps our Secretary could pass this suggestion on to the HV99 organiser of this event.

MAIL TO : ALL
MAIL FROM : GOWFAR
Hi to all. I read in a recent TND about the adventure game called "LEGENDS" and how it is a good buy. I do not know how many people actually have the game, but I bought it and can thoroughly recommend it. You have to be patient, at first, as it takes quite some time to build your characters up to a stage where they are strong enough to venture round the whole island, but once you get up to that stage, you find more and more that is new. At this stage, I have not completed the game but if anybody would like to know more or like some help with it, let me know and I will see if I can help.

See the file SCIFI_BBS for info on the SCI-FI BBS. The SCIFI-BBS file is now a sub-editor file. The date line in the first line of the file will show the date of the last update by Greg who is SYSOP of SCI-FI BBS.
Ross Mudie, SYSOP TEXPAC, 14/8/88.

MAIL TO : ALL
MAIL FROM : GOWFAR
Hi to all, yet again. Just another short note to let you know that if you are planning on doing a bit of downloading from TEXPAC and have wondered whether to get the SWORDS program or not but cannot decide, ring SCI-FI BBS (have a look at the file of the same name for details on how to ring in) and have a look at it there. I have been running the program as a play over the modem one for a few years now. So, play it and if you like it, download it from here (to play only on computer at home, not modem, I believe). Rgs, Greg.

---

o

Extended Display Package

a review by its author, Craig Sheehan

Extended Display Package is a utility designed to improve and enhance screen handling from Extended BASIC. This has been achieved with a number of assembly language links that provide many features. These include more graphics characters and colour sets, a forty column display, windowing, enhanced display and accept commands, scrolling, HI-Res drawing as well as a screen dump option. The package is available from the club shop on either a double sided or flippy disk for the usual media fee. Included on the disk is the utility, a demonstration program and the document files.

At forty two pages, the manual for XDP takes over an hour to print out. To do this, use the "Print Docs" option of the demonstration program and answer the few simple prompts. For best results, it should be printed on an Epson compatible printer, with an 11" page length and 1/6" line pitch. This will ensure that the perforation is skipped at the end of each page.

The first part in this series of articles will concentrate on the DISPLY command (note: no 'A'). It has all the features of the original Extended BASIC command, but also has many extra features. Before you can type in the example below, the package must be loaded:

1. From Extended BASIC, insert the disk in drive one.
2. Type in: RUN "DSK1.XDP" and press enter.
3. Type in: CALL LINK("NEW") and press enter.

The package is now loaded. A word of caution: once XDP is loaded, do not type in "NEW". If you wish to delete a program from the memory, enter CALL LINK("NEW") instead so that the utility does not need to be reloaded again.

After XDP is loaded, type in the following program, which will display text of two different colours on the screen.

```
100 CALL LINK("XDP")
110 FOR CHAR=32 TO 127
120 CALL LINK("CHRPAT",CHAR,HEX$)
130 CALL LINK("CHAR",CHAR+96,HEX$)
140 NEXT CHAR
150 FOR COLSET=1 TO 12
160 CALL LINK("COLOR",COLSET,9,1,COLSET+12,3,1)
170 NEXT COLSET
180 CALL LINK("SCREEN",2)
190 CALL LINK("DISPLY",1,1,".WWUsing XDP, text of
    different colours can be displayed .CR(2)Red text
    .OF(96) .CR Green text .CRGreen, .OF(0)Red")
200 GOTO 200
```

If you wish to, you could continue on forever printing coloured messages. If you think the commands used appear similar to those normally used in Extended BASIC, you will have no trouble in using XDP. Basically this program copies the pattern of ASCII characters 32 to 127 into codes 128 to 223. This gives two entire character sets, which both can be given a separate colour. In describing this program, I will detail the sections that are similar to Extended BASIC first and then deal with code peculiar to XDP. Lines 110 to 140 copy the normal character set into a set of higher ASCII codes. This is done by placing the pattern of character code CHAR into HEX$, and then writing that pattern into a code that is 96 higher than the original code. The colours of the two sets are defined in lines 150 to 170. Colour sets 1 to 12 hold the normal character set whilst sets 13 to 24 have the copy of that character set. Line 180 simply sets the screen colour to black.

So far so good. Line 100 is important to any XDP program. The XDP link initialises the package and makes changes to the memory's structure so that XDP commands can take effect. Once this link has been executed, Extended BASIC's screen handling commands no longer have the desired effect and should not be used. The DISPLY command on line 190 can initially be viewed as being the same as DISPLAY AT(1,1). In this regard it is. As with DISPLAY AT, the text in the string will be displayed from the screen position specified, with the exception of the specification comma

A specification command is almost the same as TI-Writer's formatter commands. They are preceeded by a period and consist of a two letter code which may or may not have arguments following them. These specification commands can be placed any where in the text, and illegal commands will be ignored and printed on the screen. The first command reads '.WW'. This stands for 'Word Wrap' and tells the DISPLY command to shift words so that they are not chopped in half at the edge of the screen.

After the word wrap command is some text that is printed normally, except that words will not be chopped in half at the end of a line. The next command is '.CR(2)'. This means carriage return, and will fill the rest of the screen line with spaces before setting the next display position to the first column of the next row. The two in brackets means simply to do this twice. If no argument is supplied, then the program will assume that it only needs to do this once.

Now for the part that selects the colours. There is nothing magic about this and the principle is quite simple. When we copied the normal character set, we copied it into the character codes that were ninety six above the original character. So if 96 is added to the ASCII code of each character to be displayed, we are in fact displaying this second character set, and by giving this second character set a different colour, we get the effect of being able to select between two different coloured fonts. All the '.OF(96)' does is to add 96 to the ASCII code of each character displayed thereafter. To return to the normal characters, simply set the offset to zero with '.OF(0)'. The offset command was originally included so that ASCII codes like 239 that have no character on the keyboard could be displayed with the DISPLY command, but has been used for other applications since, such as for different coloured fonts and for displaying a string of upper case characters as lower case or vice versa.

* * * * *

After that colourful introduction to XDP, we will look at some of the other features of DISPLY. With Extended BASIC's DISPLAY command, it is possible to insert variables between the text like so:

```
DISPLAY AT(1,1):NAME$;" HAS";SCORE;"POINTS."
```

Unfortunately this not allowed with XDP's DISPLY command, or indeed with any string operation. A possible solution is to use the string concatenator ("&") to join the variables together as a string, with a STR$ to convert the numeric variable "SCORE" into a string. Recoded as described above, the equivalent XDP command becomes:

```
CALL LINK("DISPLY",1,1,NAME$&" HAS "&STR$(SCORE)&
    " POINTS.")
```

This is rather messy and does have a major draw back. If NAME$ was very long, say 240 characters plus, the total length of the combined string would exceed the maximum 255 character limit that a string is allowed to have. The result would be that all the characters beyond the 255th would be chopped off the end of the string, never to be seen again.

The solution that is offered by the DISPLY command is the use of the specification command '.VA'. This will cause the next variable in the variable list to be retrieved and be displayed on the screen. Using this new specification command, the above example becomes:

```
100 CALL LINK("XDP")
110 CALL LINK("DISPLY",1,1,".VA HAS.VA POINTS.",NAME$,S
    CORE)
120 GOTO 120
```

Note that the order that the variables are used is the same as the order that they appear in the variable list. When displaying strings with this specification command, any other commands in the string from the variable list will be ignored, even if valid.

continued on page 22

Memory Ramblings

by Geoff Trott

Having used a RAMdisk for a few months, I decided that it would be nice if I could store a bit more on the RAMdisk. This seems to be the usual way, start with something small and soon you need more and more. At about the same time, perhaps coincidentally, Lou asked if I wished to join with him and buy some 32K byte static RAM chips at a good price. In the end Rolf came in as well and we bought about 16 chips each for quite a bit of money total. Having the chips I then decided to build the RAMdisk on a Horizon type card which Lou had previously populated with diodes, resistors, capacitors and sockets. I obtained a copy of the instructions from the Library and set to work.

I had to remove a few of the sockets for chips which were not used and then solder together the chips which were on top of other chips. The worst of these are the 3 74LS138 chips which are near the edge connector. I also had a bit of trouble with one connection to address line A11. I eventually used a hole under U14, which caused a bit of a problem with the resistor soldered between pin 28 and pin 20. There is of course one of these resistors on each memory chip to ensure that it enters power down mode when the power is turned off. The remainder was straight forward except that I am using 3 alkaline AA batteries which will eventually need replacing, so they are attached by wires through holes drilled in the board. I have included a diode from the + end of the battery to the PCB to ensure that no charging takes place.

After checking the connections I tried it out. Using the CFG program I had confusing results. I first tried just 3 chips and it seemed to be all OK. I was using version 7.1 in my existing RAMdisk but version 7.3 in the new one. I tried a few more chips and then problems arose. Using Minimem I discovered that the chips were not being accessed either on read or write and looking carefully at the board I found that I had left a wire off. There was one chip without any meaningful connections. A quick go with the soldering iron and then back to try it.

Then I ran into a frustrating time when I must have decided to load CFG from my other RAMdisk and CFG V7.1 hangs with lines at the top of the screen and an open parenthesis where the size of the RAMdisk appears. This makes one think that the RAMdisk memory is not working. Back to Easybug and setting CRU addresses and reading memory. There has to be a better way I say to myself. I did not have the MEGTEST memory test program, but did have the original Horizon RAMdisk memory test. This is an Extended BASIC program with CALL LINKs to two assembler routines. These looked quite short in their uncompressed D/F 80 format, so I decided to disassemble them. One was called FILL and the other CHECK, so their operation was easy to anticipate.

FILL was called once for each bit pattern to be used while CHECK was called once for each chip to be tested. Parameters were passed via CALL LOADs to fixed places in memory. Upon examination of the disassembled code, it was quite easy to see what was going on and how it would have to be modified. The main difference was that the original routine only sent out 8 bits of CRU information whereas the bigger RAMdisk needs at least 10 bits for 1Mbyte and 9 for 512K, which I was proposing to have. In the end I used 12 bits as this allows for future expansion. The other change was to allow for the difference in the accessing of the 8K DSR area. In the large RAMdisk only 6K of the 8K chip is actually accessible. The other 2K resides in the first 32K chip (U17), and to keep compatibility with the existing ROS is in the 4th rack of that chip. The first 3 racks do not get used, but to check the memory I wanted to check all of that chip. This means that the check of U11 (the 8K DSR chip) only checks the first 6K (writing and reading) while the rest of the checking of the memory exercises all the address space in each chip.

Having made these few changes I then changed the Extended BASIC program to reflect a little more closely the actual chips being tested and after debugging it all found that U14 had rather a lot of errors. I had broken a track while putting in that wire mentioned earlier, so it was back to the iron again. After that all ran well although I had one chip giving me 2 errors on some of the tests. I am not sure if that would be a problem in practice, but I put another chip in its place and all was well. I only had one layer of chips but had more than a DSDD disk to play with.

I decided make the RAMdisk into 3 disks. On the first I put my system with menu and the programs I use most often. On the second I put Multiplan and on the third I put the dictionaries for the spelling checker, thinking that I might use the checker if its speed was reasonable this way. The easiest way to access the second and third drive is via the drive name. Multiplan looks for a disk with a name TIMP, and I changed the spelling checker to look for a disk named DICT. This may all sound very easy, and it did to me when I read about it in the documentation. There are a few traps for the beginners which you may like to know.

1. Only the RAMdisk with the smallest CRU address can be used in this way. There is no way (that I know of) to get at the third disk on the second RAMdisk.

2. The T for toggle the disks from the menu only worked when my RAMdisk had a CRU address of >1400 (ie not >1000) and had the lowest CRU address of the two RAMdisks. When the CRU address was set to >1000, T from the menu did not work but CALL TD from BASIC worked in both cases. I have not tried the various DM1000 versions to see if they will copy to a disk name rather than number, but to copy files to all disks the first time requires the ability to toggle the disks.

3. The use of disk name to access a file on the RAMdisk requires that the RAMdisk CRU be >1000, otherwise the physical disk drives are accessed until an error occurs, probably because I do not have 4 disk drives.

4. There seems to be some problem with using the second RAMdisk with some programs. FUNNELWEB has strange problems with its directory function (AID) which causes odd control character codes to fill up the screen. It seems that the space character changes to some other rather random character. The version of DM1000 supplied with FUNNELWEB also seems to have problems with the second RAMdisk. I also have problems copying files to any RAMdisk with DM1000 V3.5 stopping with an error when it tries to copy a file with the name of the disk which it then finds is too big. Restarting the copy again works but it is annoying.

5. I tried using a disk name of D instead of DICT and everything stopped working. Most confusing I found this, as I had to load in the whole system again. That was the only change I had made and sure enough, when I used a different name (DICT) the problems disappeared. I did not try any other short disk names. I was trying to use FUNNELWEB at the time.

But enough of all this bad news. At least it all worked for a week or so until I decided that I wanted to start to do some c99 programming and so decided to add a few more chips to the RAMdisk to allow the c compiler to be on the disk along with some of the support files. Something went wrong and I am without my RAMdisk at the moment. Wait for the next installment.

On another slightly related topic, Rolf brought me a Minimem which he thought was losing its battery power and which he wanted to enhance along the following lines. He suggested that if the RAMs and ROM were replaced by an 8K RAM, and an Editor/Assembler GROM was installed as a switchable option to the Minimem GROM, he would have a Supercart/Minimem combined cartridge. As I was doing the necessary surgery, and with all these 32K byte chips in front of me I thought that an improvement would be to put in a 32K chip which could be switched in in 4 lumps and so allow much more flexibility. Well to cut a long story short, that is what we have done, and the more I think about it the more useful I think it would be. Losing the ROM from Minimem only seems to lose the ability to load programs and probably some routines, Easybug is still there. This means that I could load diagnostic programs into that memory and have a selection of these as well as Easybug available for diagnosing faults in systems.

continued on page 7

Jenny's Younger Set

Dear Jenny,
I have improved my score on Draw Poker from $23,000 to $30,000. Is this a Younger Set Hall of Fame score? Here is a program for those who like Dr. Who. It is a quiz from the third Dr. series.

```
100 REM 3rd Dr QUIZ by Vincent Maker
110 CALL CLEAR :: REM This is in Extended BASIC
120 PRINT "Q.1 Who was the Doctor's (Jon Pertwee)
    assistant during the invasion of AXOS?": : :
130 RIGHT=0 :: WRONG=0
140 PRINT :"Was it A) Liz Shaw                B) Harry
    Sullivan          C) Jo Grant              D)
    Michael Yates": :
150 PRINT :"Choose a letter-": :
160 CALL KEY(3,K,S)
170 IF S=0 THEN 160
180 IF K=67 THEN RIGHT=1 ELSE WRONG=1
190 IF K=67 THEN PRINT "right" ELSE PRINT "wrong"
200 PRINT : : :"Q.2 What was the Doctor's car called?"
210 PRINT "Was it A) Bessie                 B)
    Masterpiece            C) Fred
                        D) Joe": :
220 PRINT :"Choose a letter-": :
230 CALL KEY(3,L,J)
240 IF J=0 THEN 230
250 IF L=65 THEN RIGHT=RIGHT+1 ELSE WRONG=WRONG+1
260 IF L=65 THEN PRINT "right" ELSE PRINT "wrong"
270 PRINT : : :"Q.3 Who was the Queen of Peladon during
    the Doctor's second visit there?": :
280 PRINT "Was it A) Mary                   B)
    Elizabeth              C) Thalira
                        D) Galaxy": :
290 PRINT :"Choose a letter-": :
300 CALL KEY(3,A,B)
310 IF B=0 THEN 300
320 IF A=67 THEN RIGHT=RIGHT+1 ELSE WRONG=WRONG+1
330 IF A=67 THEN PRINT "right" ELSE PRINT "wrong"
340 PRINT : : :"Q.4 Who was in charge of Global
    Chemicals?": :
350 PRINT "Was it A) Thomas Stevens         B)
    General Carrington     C) Michael Yates
                  D) Lethbridge-Stuart": :
360 PRINT :"Choose a letter-": :
370 CALL KEY(3,X,Y)
380 IF Y=0 THEN 370
390 IF X=65 THEN RIGHT=RIGHT+1 ELSE WRONG=WRONG+1
400 IF X=65 THEN PRINT "right" ELSE PRINT "wrong"
410 IF RIGHT=4 THEN AS$="Perfect score, four out of
    four"
420 IF RIGHT=3 THEN AS$="Very good, three out of four"
430 IF RIGHT=2 THEN AS$="Fair score, two out of four"
440 IF RIGHT=1 THEN AS$="Poor score, one out of four"
450 IF RIGHT=0 THEN AS$="Very poor score, zero out of
    four"
460 PRINT : : :AS$: : :
470 INPUT "Want to play again (Y/N)? ":XY$
480 IF XY$<>"N" THEN 100
490 PRINT : : : : :"Thank you for playing"
500 END
```

I also have two programs here, one for Extended BASIC and one for TI-BASIC users.

```
100 REM By Vincent Maker
110 CALL CLEAR
120 PRINT "In this program you must pick the 3 numbers
    (0 to 9). If you like a time limit can be imposed"
130 INPUT "Set a time limit (Y/N)? ":A$
140 T=0
150 IF A$="N" THEN 170
160 INPUT "Time limit (in goes)? ":T
170 RANDOMIZE
180 A=INT(10*RND)
190 B=INT(10*RND)
200 C=INT(10*RND)
210 INPUT "Number one? ":D
220 INPUT "Number two? ":E
230 INPUT "Number three? ":F
240 IF A=D THEN 260
250 GOTO 270
260 RIGHT=1
270 IF B=E THEN 290
280 GOTO 300
290 RIGHT=RIGHT+1
300 IF C=F THEN 320
310 GOTO 330
320 RIGHT=RIGHT+1
330 PRINT "You have ";RIGHT;" correct so far"
340 T=T-1
350 IF T=0 THEN 410
360 INPUT "Continue? ":A$
370 IF A$="NO" THEN 410
380 IF A$="N" THEN 410
390 RIGHT=0
400 GOTO 210
410 PRINT : : :"Thank you for playing": : :"Have a nice
    day/night."
420 END
```

```
100 REM By Vincent Maker
110 PRINT "You are found on the Wizard's property, he
    demands an excuse."
120 INPUT "Well? ":A$
130 PRINT "Sorry, you are committed for trial -"
140 INPUT "How do you plead (G or N)? ":PLEA$
150 IF PLEA$="G" THEN 210 ELSE INPUT "Did you intend
    harm there (Y/N)? ":A$
160 IF A$="Y" THEN 210
170 PRINT "You are found guilty of tresspass.  The
    penalty is 2 years in goal/ja il."
180 INPUT "Do you wish to appeal (Y/N)? ":Y$
190 IF Y$<>"Y" THEN PRINT "See you in 2 years pal." ::
    STOP
200 PRINT "The judge is kind, you get 6 weeks!" :: STOP
210 PRINT "You will be executed tomorrow." :: STOP
220 END
```

I hope you like them, Vincent Maker

Well Vincent, thankyou for all those programs. I am sorry that they did not make the last issue, but I did not receive them in time. You are the only one who has submitted a score for Draw Poker, so I guess you are the holder of the record in the Hall of Fame. I hope you other younger setters get some fun out of Vincent's programs. I found them a bit of fun! Now to the first request to our resident adventure games expert, Crocodile Jones.

Dear Jenny,
I have received the following letter:

Dear Crocodile Jones,
I am having problems with adventure 7 (Mystery Fun House). How do you get into the Fun House? Do you need money? If so, could you advise me on how to get that as well? Thankyou,
Christopher of Winston Hills

Well Christopher, you need to get the dollar that is in the grate. do this by chewing the gum and putting it on the stick (to stick) and using it to get the coin (with stick). Buy your ticket and wear your shoes. Then "GO FUN". Thanks for the letter,
yours faithfully, Crocodile Jones

continued from page 6

The perfect place for a good disk diagnostic when(?) I write one. A very good idea Rolf!
The switching can be done with 2 single pole double throw switches (toggles) but this means 3 such switches (one for the GROMs) and a push button (reset) are needed. It is not easy to fit all that in a cartridge case. Another way is to do the RAM switching with a 74LS74 and a push button with leds to show which one is in use. This reduces the hardware to 2 push buttons and one toggle (or slide), with room for 2 LEDs.

The Gramulator

by Mark Van Copperhole, USA

At last! A direct equivalent for the popular but out of production Gram Kracker has been designed by an engineer in Massachusetts. It is called the Gramulator.

A wire wrapped prototype was demonstrated to the MAGNETIC Users Group in Andover, MA at their September meeting and to the Boston Computer Society TI99/4A Users Group at their November meeting. It performed flawlessly at both meetings. The Gramulator offers virtually all of the features of the Gram Kracker but it is targeted to cost less.

No production Gramulators have been built yet. To go from a prototype to a production model requires an investment of about $1000. As with anything else the more that can be made on one batch, the cheaper they will be. You are invited to respond to this if you would consider purchasing this product. Technical questions are welcome. Please write to:

Mark Van Coppenole
52 Audobon Road
Haverhill, MA 01830
(617) 372-0336

FEATURES:

The Gramulator simulates 64K of GRAM and 16K of RAM (in two 8K banks at >6000->7FFF).

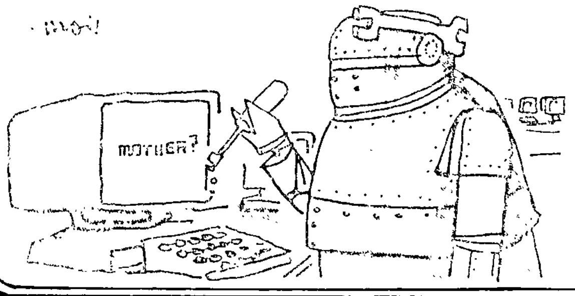
1) You can customize the built-in TI99/4A operating system in GROM 0 AND TI BASIC in GROMs 1 and 2.
2) You can backup your GROM and ROM cartridges to disk to protect your investment and reduce wear on the cartridge port. All TI, Atarisoft and Parker Brothers cartridges can be backed up. (Does not work with MBX.)
3) Acts as a "Super Space" cartridge allowing you to run programs requiring RAM at >6000->7FFF (including MYARC'S XB11).
4) Allows you to use a customized GROM 0, 1, or 2, while a cartridge is in the slot. One applicatiom is that you can use your own character set with a cartridge like TI-Writer.
5) Capable of loading user written GPL code.
6) A total of 80K of memory with lithium batteery backup.

The software needed to LOAD and SAVE GRAM and ROM will be built-in for instant access. A memory editor, which will be supplied on disk, will allow you to alter and save any program loaded into the built-in GRAM and RAM. User documentation and technical information will be included.

Memory expansion and a disk drive are required to take full advantage of the gramulator.

Eye Witness report by Walt Howe:

1. I saw the demonstration at the Boston meeting and was very impressed. Mark has designed the Gramulator to take advantage of inexpensive, readily available components that should help keep the price down.
2. One improvement over the Gram Kracker will be an external, easily accessed battery for quick replacement.
3. If you are at all interested in this, drop Mark a note. Without good evidence of user support, it will never be built. Make copies of this and pass it around on bulletin boards and hand it out at user group meetings. This project should really be supported. o



Multiplan Madness

by Tom Arnold, Hamilton

Yes I remember that I was going to talk to you about formulas in Multiplan, specially the IF-THEN statement. However time constraints and the fact that I am not all too clear about them yet has lead me to delay this topic till later. I want to talk to you about using spread sheets, in particular Multiplan, as Text Editors. Why would I want to use Multiplan as a word processor? Simply because in certain applications it can be very useful. I actually use a spread sheet at work quite often and I never work with numbers! It is very useful in keeping lists of things, in particular lists that you might want to up date from time to time. I assume that you would want to sort these lists. How about a telephone list, an address list of your club members, a list of topics on the TI99/4A, an index to a book, any index for that matter. For these applications Multiplan is a very powerful tool. Let me explain.

At work I have to write job practices, generally I type them out. All these practices need indexes. I used to write every topic out by hand, then number each letter in alphabetical order. For example under "P", prince comes before punk and punk before pyke. Then I would write out all the topics in order for the typist to retype. This took a long time, especially if there were many items.

Along comes the computer and I immediately found an easier way. I type out two columns, the topic in column #1 and the page number in column #2. These are entered in the order that I come across them in the book. After I am done I "sort" the first column and I have an instant index. Very simple and the typist does not need to retype it either!

How do you do this in Multiplan? First, do not type Titles or any other text you would want on the printed page, enter this later. Assuming we are entering an index. Type in the names (topics) in column #1. After each entry move the arrow keys right and then enter the number or other references you want. Repeat this until all entries are done. Now sort the spread sheet by selecting SORT. First you will be prompted the column you wish to sort by, in our case enter (1). Control A will move you to the second field where the row selection will be prompted. The default is between rows #1 and #255. This will be your normal selection. However you can use this to sort partial lists. This could be useful if you want to sort one part, by one column and the other part by another column. Which brings us to the last prompt (via CTRL[A]). You are given the choice of ascending order (<) or decending order (>).

I suggest you try this little experiment. List 5 names in column 1, list the numbers 5 to 1 in column 2, list the letters c,f,d,s,f in column 3, and then the numbers 1,34,76,45 in column 4. Now sort using a different column each time. See how the rows follow each other and the order changes. You may also sort using different columns, for example you could sort a long list by column #1 and then the top 20 entries could be sorted by column #2 to achieve a different order.

As for uses, how about starting by making a combination of phone list and address list. Then sort by name, phone numbers, addresses, printing out a new list each time. These would then be handy references. For example, you could isolate all your friends who live in Australia whose names begin with "P" and whose phone numbers are in area code "2456". I would think that you would find 10 or 20 anywhay! Just kidding!!!

One more point about using Multiplan to make columns of items. You will not accidently format them into one jumbled mass! I forgot to mention that all titles etc. should be added after all your sorting is done. Hope this had been helpful. o

Character Set Animation

Author unknown

In this article we will learn how to create complex computer animation using the same techniques that filmmakers use in creating hand drawn animation like cartoons. We will create a 24-character representation of a galloping horse. If that is not enough, we will place several of them on the screen at once. The result will be a whole stable of horses all galloping in tempo - and not a SPRITE in sight!

CorComp (or AT) makes it possible

One of the most useful features of the new CorComp Commands is the ablity to move large blocks of memory at one time with the MOVEM command. Until CorComp added this much needed command, the only way you could move data from one location to another was by using FOR NEXT loops moving one byte at a time. With this new command, however, an entirely new form of computer animation is available to the TI99/4A owner - Character Set Animation.

What is Character Set Animation?

Character Set Animation is one of the most common forms of computer graphics. It works much like film animation in that a set of pictures are flashed into view in such a way that it looks as if the picture is moving. In the early computer versions of this type of animation, the entire "movie" was stored as a series of frames, all in sequence in memory. To create the effect of motion, the computer would simply move the location of the screen area to begin at the next "frame" of the movie. Sort of like looking at the same landscape from different windows in the same building. It is a lot easier to change the "window" than to move the building!

The TI99/4A computer cannot use this type of animation because it uses a "memory mapped" video chip, the TMS9929A. This means that to create the same effect, we would probably have to move the "building" brick-by-brick. This means tedious byte-by-byte actions that are way too slow in BASIC to ever create the sense of motion. Luckily, CorComp has changed all that with their MOVEM command.

Redefine the characters

To create the same sense of motion on the TI99/4A, we will need to constantly redefine the displayable character set. In this example, we will use the lowercase letters (character sets 9-12 in TI-BASIC). We can just print the characters onto the screen, then redefine them by placing new data into the character definition tables in VDP memory. We will let the computer do the rest!

A lot of Data

To do this will require a great deal of data. In the example program below, we have chosen a 6 by 4 "frame" or 24 characters per frame. This means that each frame requires 192 bytes of data. Since we will be using only five frames, we will need a total of 960 bytes to animate our picture. As you can see, it takes quite a bit of data to create computer animation!

We will not be able to store the data in the standard TI99/4A character strings either ("001FAC446688F100"). Since we will be "poking" data directly into the VDP character table, we will have to express our character definitions as decimal values (>FF=256, >10=16, etc.).

We also need a place to store all this data. In our example we use the expansion memory starting at >A000 (40960). If you do not have the 32K memory, you can locate your data in some free area of VDP memory. Try around >2000 (8192). If you want to move the data area to some other portion of expansion memory, simply change the value in line 500 (CPU=NNNNN).

So here is what we do

Here is a quick summary of the program given below. You can use the ideas found here to create your own "computer movies".

1) Load all needed character data into free memory area.
2) Clean out original character data, so we can start with a "blank screen".
3) Put up the characters on the screen (they are blank, so we cannot see them yet!)
4) Set up a loop that places each "frame" of data into the character set table (this causes us to "see" the frame on the screen).

In our version below, you have the added ability to control the speed of the horses' gallop.

NOTE: The program below operates only in TI-BASIC. If you want to run this program in Extended BASIC, you need to change all the CorComp Command statements to match the Extended BASIC versions.

```
100 REM ***************
110 REM *             *
120 REM *  GALLOPING  *
130 REM *    HORSE    *
140 REM *             *
150 REM *  ANIMATION  *
160 REM *    DEMO     *
170 REM *             *
180 REM ***************
190 REM
200 REM      04/85
210 REM
220 REM   SUBFILE99
230 REM
240 REM *************
250 REM *SET UP AREA*
260 REM *************
270 REM
280 CALL CLEAR
290 REM
300 FOR X=1 TO 8
310 CALL COLOR(X,16,6)
320 NEXT X
330 CALL SCREEN(6)
340 REM
350 A$="abcdef"
360 B$="ghijkl"
370 C$="mnopqr"
380 D$="stuvwx"
390 SP=25
400 REM
410 CALL VPOKE(6,96,"GALLOPING HORSE DEMO")
420 REM
430 REM ***********
440 REM *POKE DATA*
450 REM ***********
460 REM
470 CALL SOUND(150,1400,0)
480 CALL VPOKE(388,96,"LOADING DATA INTO EXP MEM")
490 CALL VPOKE(326,96,"ONE MOMENT PLEASE....")
500 CPU=40960
510 FOR X=0 TO 959
520 READ B
530 ADDR=CPU+X
540 CALL MPOKE(ADDR,0,B)
550 NEXT X
560 REM
570 REM *****************
580 REM *CLEAN OUT CHARS*
590 REM *****************
600 REM
610 VDP=1544
620 FOR X=0 TO 191
630 ADDR=VDP+X
640 CALL VPOKE(ADDR,0,0)
650 NEXT X
660 REM
670 REM *BEEP WHEN DONE*
680 REM
690 CALL SOUND(150,1400,0)
700 CALL VPOKE(738,96,"PRESS ANY KEY TO START DEMO")
710 CALL KEY(0,K,S)
720 IF S=0 THEN 710
730 REM
740 REM *CLEAR VDP AREA*
750 REM
760 CALL HCHAR(2,1,32,736)
770 REM
780 REM *****************
790 REM *PUT UP BLANK SET*
800 REM *****************
```

```
820 FOR Y=1 TO 3
830 FOR X=1 TO 25 STEP 8
840 Z=((192*Y)+X)-64
850 A=Z
860 B=A+32
870 C=A+64
880 D=A+96
890 CALL VPOKE(A,96,A$;B,96,B$;C,96,C$; D,96,D$)
900 NEXT X
910 NEXT Y
920 REM
930 REM ****************
940 REM *DISPLAY HORSES*
950 REM ****************
960 REM
970 CALL SOUND(150,1400,0)
980 CALL VPOKE(736,96,"<S>=SLOWER, <D>=FASTER,
   <Q>=QUIT")
990 CALL MOVEM(3,40960,1544,192)
1000 GOSUB 1130
1010 CALL MOVEM(3,41152,1544,192)
1020 GOSUB 1130
1030 CALL MOVEM(3,41344,1544,192)
1040 GOSUB 1130
1050 CALL MOVEM(3,41536,1544,192)
1060 GOSUB 1130
1070 CALL MOVEM(3,41728,1544,192)
1080 GOSUB 1130
1090 GOTO 990
1100 REM
1110 REM *CHECK KEYBOARD*
1120 REM
1130 CALL KEY(0,K,S)
1140 IF S=0 THEN 1230
1150 IF K<>83 THEN 1190
1160 IF SP>50 THEN 1230
1170 SP=SP+4
1180 GOTO 1230
1190 IF K<>68 THEN 1220
1200 IF SP<1 THEN 1230
1210 SP=SP-4
1220 IF K=81 THEN 1260
1230 FOR X=1 TO SP
1240 NEXT X
1250 RETURN
1260 CALL CLEAR
1270 STOP
1280 REM
1290 REM *****************
1300 REM *HORSE CHAR DATA*
1310 REM *****************
1320 REM
1330 REM
1340 REM *FRAME 1 DATA*
1350 REM
1360 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1370 DATA 0,0,0,0,0,0,0,0,0,0,0,1,1,6,15
1380 DATA 0,0,6,118,155,127,247,231,0,0,0,0,128,192,
   32,136
1390 DATA 0,0,0,3,14,29,5,0,0,0,3,252,188,14,11,11
1400 DATA 0,0,255,31,0,24,31,157,59,15,254,192,0,0,0,
   240
1410 DATA 135,14,60,124,12,8,8,56,196,204,48,0,0,0,0, 0
1420 DATA 0,0,0,0,0,0,0,0,15,5,126,127,97,99,103,99
1430 DATA 207,223,243,224,192,192,128,128,252,191,
   223,127,3,3,1,1
1440 DATA 112,248,254,142,252,248,128,128,0,0,0,0,0,
   0,0,0
1450 DATA 1,1,0,0,0,0,0,0,193,128,0,0,0,0,0,0
1460 DATA 192,224,112,24,28,0,0,0,1,1,1,0,0,0,0,0
1470 DATA 128,192,224,48,56,0,0,0,0,0,0,0,0,0,0
1480 REM
1490 REM *FRAME 2 DATA*
1500 REM
1510 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1520 DATA 0,0,0,0,0,0,0,0,0,0,1,5,8,59,87
1530 DATA 0,8,12,190,121,248,252,191,0,0,0,0,0,128,
   32,208
1540 DATA 0,0,3,7,15,29,0,0,0,7,252,208,151,31,31,15
1550 DATA 0,255,32,32,108,255,255,255,255,1,0,0,4,28,
   252,184
1560 DATA 187,184,48,16,16,16,48,240,144,192,0,0,0,0,
   0,0
1570 DATA 0,0,0,0,0,0,0,0,15,15,62,60,56,236,204,198
1580 DATA 255,143,0,0,0,0,0,0,248,255,31,60,48,112,
1590 DATA 28,254,6,12,56,48,0,0,0,0,0,0,0,0,0,0
1600 DATA 0,0,0,0,0,0,0,0,220,216,192,96,112,0,0,0
1610 DATA 1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0
1620 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1630 REM
1640 REM *FRAME 3 DATA*
1650 REM
1660 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1670 DATA 0,0,0,0,0,0,0,0,0,0,0,0,1,0,63
1680 DATA 0,0,2,3,62,239,223,127,0,0,0,0,128,192,32, 16
1690 DATA 0,0,0,3,15,63,10,0,0,0,3,252,200,12,31,31
1700 DATA 0,0,255,24,48,124,56,255,23,255,128,0,0,56,
   126,255
1710 DATA 247,238,252,20,8,24,16,16,196,108,48,0,0,0,
   0,0
1720 DATA 0,0,0,0,0,1,3,6,31,31,61,121,243,195,1,0
1730 DATA 191,224,224,192,128,0,128,96,254,254,7,7,
   14,12,24,112
1740 DATA 56,7,255,0,0,0,0,0,0,128,192,192,192,192,
   192,128
1750 DATA 12,24,24,56,24,0,0,0,0,0,0,0,0,0,0
1760 DATA 51,62,0,0,0,0,0,0,192,0,0,0,0,0,0,0
1770 DATA 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1780 REM
1790 REM *FRAME 4 DATA*
1800 REM
1810 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1820 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,1
1830 DATA 0,0,6,118,155,127,247,231,0,0,0,0,0,0,128, 32
1840 DATA 0,0,0,3,7,10,0,0,0,0,3,255,204,141,15,14
1850 DATA 0,0,252,127,16,239,63,111,1,11,29,224,0,12,
   142,220
1860 DATA 127,255,254,122,30,12,4,68,8,252,204,32,0,
   0,0,0
1870 DATA 0,0,0,0,0,0,1,3,13,15,15,63,248,224,128,0
1880 DATA 159,127,248,56,24,24,56,28,252,255,63,1,1,
   0,112,63
1890 DATA 124,196,243,255,199,192,192,192,0,0,0,128,
   192,192,96;48
1900 DATA 118,60,0,0,0,0,0,0,0,0,0,0,0,0,0
1910 DATA 6,1,0,0,0,0,0,0,0,128,192,192,0,0,0,0
1920 DATA 0,0,0,0,0,0,0,0,24,0,0,0,0,0,0
1930 REM
1940 REM *FRAME 5 DATA*
1950 REM
1960 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1970 DATA 0,0,0,0,0,0,0,0,0,0,6,15,30,28,61
1980 DATA 48,56,254,231,227,255,253,255,0,0,0,0,128,
   32,144,232
1990 DATA 0,1,3,7,13,0,0,0,3,255,232,78,142,15,7,15
2000 DATA 255,124,80,24,60,247,239,159,255,63,12,0,0,
   0,241,4
2010 DATA 252,28,56,56,24,16,208,112,216,64,0,0,0,0,
   0,0
2020 DATA 0,0,1,1,3,3,118,60,60,255,240,128,0,0,0,0
2030 DATA 112,240,112,48,56,24,12,6,255,1,12,6,3,1,0, 0
2040 DATA 48,240,120,56,124,198,3,1,0,0,0,0,0,0,0,128
2050 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2060 DATA 3,1,0,0,0,0,0,0,0,192,224,0,0,0,0,0
2070 DATA 0,0,0,0,0,0,0,0,192,112,56,0,0,0,0,0
2080 REM
```

o

For Sale

MAIL TO : ALL
MAIL FROM : SHOP

For sale

1 TI disk drive for sale, no power supply. Please get in touch with Steven Carr on (02)608 3564 between 4:30 and 10:30 pm on weekdays or between 10:00am and 10:00pm on weekends or leave a message via TEXPAC BBS.

MAIL TO : ALL
MAIL FROM : ABEL

For sale

Console and power supply, PE box, 32K memory expansion, disk drive SSSD

Extended BASIC module, DM2 module, TI Logol, Touch Typing Tutor.

(044)21 4274 after hours. Make an offer.

Software tips #2

from Stephen Shaw, England

Here are some interesting routines by Dr Roy T Tamashiro, for Extended BASIC plus 32K.

Flip.

Lines 100 to 150 are a demonstration program. The Flip routine is in lines 30100 onwards and may be saved as a merge file

```
100 CALL FLIP
110 CALL CLEAR
120 PRINT "FLIP DEMO": : :
130 FOR I=33 TO 126 :: PRINT CHR$(I); :: NEXT I ::
    PRINT
140 CALL LINK("FLIP")
150 INPUT "Press enter":R$ :: GOTO 110
160 !
170 !
30100 SUB FLIP
30110 ! 1986 R TAMASHIRO
30120 ! COMPUTER BRIDGE
30130 ! NOVEMBER 1986
30140 ! XB + 32K
30150 !
30160 CALLINIT :: CALL LOAD(16376,70, 76,73,80,32,32,
    36,244) :: CALL LOAD(8196,63,248)
30170 CALL LOAD(9460,2,0,4,0,2,5,37,72, 2,2,0,2,2,4,0,
    4,2,1,40,56,4,32,32,44)
30180 CALL LOAD(9484,192,96,40,56,6,5, 213,65,6,193,6,
    5,213,65,5,192,2,128,6,240,21,5,6,4)
30190 CALL LOAD(9508,22,239,2,37,0,16, 16,234,2,2, 2,
    240,2,1,37,64,2,0,4,0,4,32,32,36,4,96,0, 112)
30200 PRINT "FLIP LOADED"
30210 SUBEND
30220 END
```

Upside down demonstration
(-Merge FLIP into it!!!-)

```
100 CALL FLIP ! Merge it in!
110 DATA 1,"UPSIDE DOWN"
120 DATA 3,"by Roy Tamashiro "
130 DATA 5,"from Computer Bridge"
140 DATA 6,"November 1986"
150 DATA 15,"After a while, everything"
160 DATA 16,"gets corrected..."
170 DATA 18,"It will wait for you to"
180 DATA 19,"press the ENTER key"
190 DATA 99,"DUMMY DATA"
200 RESTORE :: CALL UPSIDE
30000 GOTO 200
31500 SUB UPSIDE
31510 DIM W$(24) :: FOR I=1 TO 24 :: W$(I)="" :: NEXT I
31520 READ R :: IF R<25 THEN READ W$(R) :: GOTO 31520
31530 CALL LINK("FLIP") :: CALL CLEAR :: FOR I=1 TO 24
31540 DISPLAY AT(24-I,14-LEN(W$(I)/2): W$(I)
31550 NEXT I :: CALL DELAY
31560 FOR I=1 TO 24 :: CALL HCHAR(I,1,32,32) :: DISPLAY
    AT(I,14-LEN(W$(I)/2): W$(I) :: NEXT I :: CALL DELAY
    :: CALL LINK("FLIP")
31570 DISPLAY AT(24,1):"PRESS ENTER" :: CALL KEY(0,K,S)
    :: IF K<>13 THEN 31570
31580 CALL CLEAR :: SUBEND
31590 SUB DELAY :: FOR D=1 TO 1000 :: NEXT D :: SUBEND
31600 END
```

=======

Q. How do I redefine the CURSOR in a running Extended BASIC program?

A. The answer to this query is on page 57 of TI*MES Issue 8, and comes from Jim Peterson of Tigercub Software. It requires 32K memory expansion.

```
100 ! Define character N with the definition you
    require.  Now proceed:
110 CALL CHARPAT(N,A$) :: FOR J=1 TO 16 STEP 2 ::
    H$=SEG$(A$,J,2) :: CALL HEXDEC(H$,D) :: T=T+1 ::
    H(T)=D :: NEXT J
120 CALL INIT :: CALL LOAD(8196,63,248)
130 CALL LOAD(16376,67,85,82,83,79,82, 48,8)
140 CALL LOAD(12288,H(1),H(2),H(3),H(4), H(5),H(6),
    H(7),H(8))
150 CALL LOAD(12296,2,0,3,240,2,1,48,0, 2,2,0,8,4,32,
    32,36,4,91)
160 CALL LINK("CURSOR") !program here -------- then:
20000 SUB HEXDEC(H$,D) :: N=1 :: DEC=0
20001 FOR J=1 TO LEN(H$) :: A$=SEG$(H$,LEN(H$)-J+1,1)
    :: IF ASC(A$)>58 THEN HT=ASC(A$)-55 ELSE HT=VAL(A$)
20002 DEC=DEC+N*HT :: N=N :: NEXT J
20003 IF DEC<=32768 THEN D=DEC ELSE D=-(65536-DEC)
20004 SUBEND
20005 END
```

========================

TI Editor/Assembler bug

If you wish to use PIO as an output device for a LIST, you have to put a full stop after it to make it work (eg PIO.) due to an error in the ASSM2 file. The error lies in the DSRLNK routine which the assembler uses. It does not use the same routine that your programs call! The assembler internally uses a modified VSBR routine which returns data to R0 instead of the usual R1, but whoever wrote the internal DSRLNK forget this! Consequently, the assembler looks at R1, and the program uses an incorrect name length when scanning for the device name! The period causes the routine to exit, and "catches" the error.

Want to fix it? If you still have the original ASSM2 file, use a sector editor to look for the code 04 20 AA BE 00 00 D1 C1. It is usually in the 8th sector of the file. Change that last C1 into a C0 and the Assembler will work as it is supposed to! No need to add a full stop after PIO now!

Courtesy Danny Michael (Dump and Neatlist author), SHOALS 99ERS.

================

Will we see Multiplan on disk, like TI Writer? Nope. Open up your Multiplan module and check out those chips, then look at the size of the disk files they load into your 32K ram and VDP ram. That comes to about 66K. Now squeeze it into 32K? Nope. Stuck with the module until you get your 9640 (Geneve) anyway!

==================

OPERATING SYSTEM BUG

Seems a little late in the day to be discovering system bugs, but here we are with one I have not so far seen reported.

Type and run this:

```
100 CALL KEY(3,A,B) :: PRINT B:B:B :: CALL JOYST(1,X,Y)
    :: GOTO 100
```

Now hold down a key on the right side of the joystick, hold it down continuously. What you get is a lot of +1's. Every key scan treats the continuously held down key as new.

Now hold down a key on the left side of the keyboard. You may get an odd +1, the odds are you will not. You will get a lot of -1's.

Now change the CALL JOYST to scan JOYST(2.., and try again to see if there is any difference.

These results are not what you might expect, and could well cause problems, especially if you use a CALL KEY to scan for the fire button and check its status as well. This could explain why I have problems using the unreleased module game LASSO and also explain why it is so unfinished?

This may be a hardware bug, the same as the alpha lock/joystick problem, so easily fixed with a suitable diode and a cut pcb track, or it could be a firmware bug. It is constant in every Extended BASIC module I have, including Myarc Extended BASIC.

==================================

FORTH NOTE

Would you like to input a floating point number from the keyboard? It is not too obvious how to do it and thanks go to George Sloane of the Aloha User Group in Hawaii for this one:

```
: ENTFP ( --- f ) PAD 1+ DUP 20 EXPECT
    LEN SWAP 1- C! VAL FAC> ;
```

Explanation:

The address of PAD+1 is placed on the stack, this is because VAL, perhaps contrary to what the Manual suggests, starts reading the entry at PAD+1. If you just used PAD, you would lose the first character, either the most significant digit, or perhaps a decimal point or a minus sign.

20 EXPECT allows you to enter up to 20 digits more than you may need but so what!

VAL then converts the string to a floating point number.

FAC> brings the floating point number on to the stack.

If you want an integer number output, you could use FAC->S instead.

CALL LOADs

In response to requests, here is a list I have found useful. Remember, you must do a CALL INIT before you can use CALL LOAD!

CALL LOAD(-31961,51) :: END ..return to title screen
CALL PEEK(-28672,A)........IF A=0 then NO speech synth attached.
CALL LOAD(-31888,63,255) then NEW up another 250 odd bytes of memory from CALL FILES(1))
CALL LOAD(-31888,55,215) then NEW three drives! NB: Attempted disk usage after turning all drives off like this can be fatal!
CALL LOAD(-31931,0) ..delete Extended BASIC protection
CALL PEEK(-31863,A) ..in a running program, if A=231 then expansion memory is fitted.
CALL LOAD(-31961,149) :: END ..reset console and load DSK1.LOAD
CALL PEEK(-31952,A,B) ..Pointer to starting address of line number table.
CALL PEEK(-31950,A,B) ..pointer to end address of line number table.
CALL PEEK(-31954,A,B) ..current line being referenced in line number table. The line number table is composed of four bytes for each line: two bytes hold the line number and two bytes hold the address.
CALL LOAD(-31806,16) ..disable quit key.
CALL LOAD(-31878,N) ..all sprites over N stop. Can be used to make several sprites start moving simultaneously, This LOAD is over-ridden by a subsequent CALL SPRITE or CALL MOTION.
CALL LOAD(-31806,64) will disable all sprite motion until it is reset with CALL LOAD(-31806,0) or (-31806,16). If you wish to disable sprite motion and quit key, use CALL LOAD(-31806,80).

CALL LOAD can also be used for SOUND and SPEECH. These entail long articles and have been covered in previous TI*MES.

SPRITE SAMPLER:

First a program using CALL LOAD(-31878),

```
100 CALL CLEAR :: CALL INIT :: CALL SCREEN(2) ::
    R=2*PI/28
110 FOR I=28 TO 1 STEP -1
120 CALL SPRITE(#I,46,16,96,128,10*COS(I*R),
    10*SIN(I*R)) :: CALL LOAD(-31878,0) :: NEXT I
130 GOTO 130
```

or using -31806:

```
100 CALL CLEAR :: CALL INIT :: CALL SCREEN(2) ::
    R=2*PI/28
110 CALL LOAD(-31806,80)
120 FOR I=1 TO 28
130 CALL SPRITE(#I,46,16,96,128,10*COS(I*R),
    10*SIN(I*R)) :: NEXT I
140 CALL LOAD(-31806,16)
150 GOTO 150                compare the results!
```

And nothing to to with CALL LOAD but related to the sprite sample:

```
100 R=2*PI/28 :: CALL CLEAR :: CALL SCREEN(2)
110 FOR I=1 TO 28 :: CALL
    SPRITE(#I,46,16,96,128,COS(I*R),SIN(I*R))
120 NEXT I
130 GOTO 110
```

From Tips from the Tigercub Number 8:

Original:

```
100 IF X=1 THEN Y=7 ELSE IF X=2 THEN Y=33 ELSE IF X=3
    THEN Y=19 ELSE IF X=4 THEN Y=21.
```

Better:

```
100 Y=VAL(SEG$("07331921",X*2-1,2))
```

o

Card Games

by Werner Kanitz

Poker, Gin! Christmas, Easter?? Not really ever heard of RAM cards. Some time ago several members of our little fraternity started playing the RAMcard game. These little beasties allow the owner to store heaps of stuff on them. This could be programs or data, which becomes readily accessible at RAM speed (just think, instant phone books).

The apparent prerequisite is a disk controller card. However I believe that it is possible to drive these critters as stand alone units i.e., run programs off the RAMcards without a disk controller. It is however beneficial to have a mate with disk drives so the card can be set up and loaded.

Well, since my collection of computers and paraphenalia met the prerequisites to handle a RAMcard they put the hard word on me to, if not buy one, at least buy the bits and glue one together myself. (They jest. I flunked cut and paste in pre school.) Besides I have heard of people going dyslexic and requiring eye crutches as a result of looking at the little pointy bits where the glue goes.

Nonetheless, the lad weakened and bought a circuit board with the intention of eventually buying the bits to fill the holes. At that time the holes were to be filled with 8K byte crisps. I did not buy any then (they would not accept monopoly money as legal tender). Since then, as things are wont to do, the buggers have more than trebled in price, but this is marginally offset by the fact that their storage capacity has quadrupled. (This fails to alleviate the monopoly money economy prevalent in my financial circles).

To cut a long story short it seems that not the entire board needs to be filled to work. It is possible to do things in small stages and gradually build up to the desired capacity. In an attempt to convince myself that the exercise is worth the effort, I borrowed a completed card off my visually impaired mate Karel Kuit who has just recently had to acquire eye crutches. He assembled his own RAMcard (but then again my wife needs glasses and she has never built a RAMcard so perhaps the relationship is not as strong as first suspected).

Now, the RAMcard is a solid state disk drive which is battery backed so none of the goodies become demagnetised and fall off when the power is cut. Further there are no mechanical bits flapping up and down and spinning around, so acessing data on the RAMcard is marginally(?) quicker than a normal disk drive. Its almost like having your data and programs resident in the computer.

The first thing I did was to remove the batteries to clean the existing programs and data off the card (nice bastard). Still, it was replaced with the latest ROS (RAM Operating System), a RAM resident Editor Assembler, a RAM resident TI-Writer, a file display utility, a DM-1000, a Disk Editor and a Quick copier to mention a few. Practically the only cartidge required is Extended BASIC.

The speed at which these programs are acessed is really quite amazing and really puts this little toy into a different class. I am therefore resolved that I shall get some of those crisps to glue into the holes on my card as soon as I can. I only wish that monopoly money were legal tender, perhaps I can get away with using a latex cheque!

o

Review of 4A/TALK

by Scott Darling, Sysop of GENIE, USA

This file was on Compuserve, and is from the TI-Forum.

```
:--------------------------------------:
:    4A/Talk from DataBioTics, Inc.    :
:--------------------------------------:
: Performance.......................B  :
: Ease of Use.......................B+ :
: Documentation.....................A+ :
: Value.............................A- :
: Final Grade.......................A- :
:--------------------------------------:
```

What is 4A/Talk? 4A/Talk is a terminal emulator type of program. It requires Console, 32K card, Disk system, RS232, Modem, Extended BASIC module, E/A module, or Mini-Memory module, TI or CorComp Disk Controller (unsure of Myarc compatibility). Optional is a printer. This program has TE2, Xmodem, and ASCII capabilities. It also includes two unique features not found on ANY other terminal program. A Disk Cataloger and a Delete file option from online mode! This Disk catalog also has the option to highlight a file on the disk so the name can be used for up or downloading.

Performance:
The program worked flawlessly in all phases of operation. It could not be crashed by the user in any manner that I tried. The only complaint is that the program is so large, the Buffer space is only 8K. This 8K is used for ALL phases of transfers, But there is a graphical representation on screen to show how large the buffer is at all times in ASCII mode. But on other hand the 8K is also used for TEII and XM(transfers, so this gives the drives a break!

Ease of Use:
The program has an onscreen help listing (CTRL[7]) that can be brought up at any time! Every prompt is in menu choices, so you have a way of backing out of an incorrect keystroke. There is a configure file to set up a default file and an auto-dialer file to list your most frequently called numbers and access them using a smart modem! There is also a Disk Catalog function that allows you to read any disk on screen. Also included is a Delete file function, from an on or offline mode.

Documentation:
Is FANTASTIC! I could give this program to anyone and they could make a connection on the first try! The documentation covers every aspect of the program and then some! The documentation alone is worth the price. There is nothing left to the imagination!

Value:
If you do a lot of ASCII reading, this program may be cumbersome! But for TE2 or Xmodem it will be a worthwhile investment.

Final Grade:
I downrated this program solely on 8K buffer for ASCII. I realize because of the program size it was impossible to have a larger buffer. Even Fast-Term would have only 11K buffer if both files (TE2 and Xmodem) were loaded. Also I thought that there should be a mode of operation to dump the buffer when it is full. At least it does not erase the buffer like Pterm! Another inconvenience is using ASCII uploading. The program only has a line at a time option. There is no provision for send-all of the data automatically!

Ordering Info:
DataBioTics, Inc.
PO Box 1194
Palos Verdes Estates, CA 90274

The Bull Board 2
Software Hardware
P.O. Box 307
Upper Marlboro, MD 20772
Price: $19.95 plus $2.50 (S H)

The TI/PC Connection

by Rolf Schreiber

This module from CorComp is a welcome addition to the utilities available for the TI99/4A. It effectively allows the two way transfer of ASCII (ie data or text) files between an IBM PC (and/or compatible/clone) formatted disk and a TI99/4A formatted disk

Briefly, the following actions are possible:
1) Read the directory of a PC or a TI99/4A formatted disk (SSSD, DSSD or DSDD at 18 sectors/track only) by selecting either a '1' or a '2' from the menu.
2) Copy a file between a PC and a TI99/4A formatted disk (in either direction) by typing a 'C' in the command field and executing the command by pressing FCTN[6].
3) Rename a file on either a PC or a TI99/4A formatted disk by typing an 'R' in the command field and executing the command by pressing FCTN[6].
4) Delete a file on either a PC or a TI99/4A formatted disk by typing a 'D' in the command field and executing the command by pressing FCTN[6].
5) Change the file protection of a file on a TI99/4A formatted disk by typing either a 'U' (to unprotect) or a 'P' (to protect) in the command field and executing the command by pressing FCTN[6].

Points to note are:
1) You need a CorComp Disk (or AT) Controller card for the program to work.
2) You need a 2 drive setup: drive #1 for the PC formatted disk and drive #2 for the TI99/4A formatted disk. Since PC disks are usually formatted at least 360K, drive #1 must be a DSDD drive. Drive #2 may be a RAMdisk.
3) You can escape to the menu at any time by pressing FCTN[9].
4) The TI99/4A file must be in DIS/VAR 80 format and must be unprotected.
5) The sector size of the PC formatted disk appears to be 128 bytes.

Points I did not like:
1) The program does not allow you to format a disk, either in PC format or in TI99/4A format.
2) You cannot transfer a TI99/4A file in DIS/FIX 80 format; it must first be converted to DIS/VAR 80 form using either TI-Writer or Editor/Assembler.
3) The TI99/4A text file must must not be write protected, otherwise the program ignores the command and returns to the menu.
4) The program does not work with a MYARC disk controller card.
5) The program does not allow you to view a file on the screen, or to send the directory to a printer. I presume that memory space in the cartridge was a limiting factor.
6) The size of a PC ASCII file that can be converted is limited, and in my opinion, far too small.

Points I liked:
1) It allows a two way transfer of text or ASCII data between a PC readable disk and a TI99/4A formatted one.
2) It opens up another avenue of obtaining and using information and increases our machine's capabilities in a very important direction.

Possible applications:
1) Transferring your data files to an PC without the need to retype the information, if you feel the need to upgrade your computer.
2) Transferring information more easily between your workplace (school?) and home.
3) Transfer BASIC programs (in an ASCII format) between PCs and the TI99/4A, to facilitate conversions without the need for retyping.
4) Transfer of UCSD Pascal source code between PC and TI99/4A disk formats. A program is available for the TI99/4A to convert between Pascal screens and DIS/VAR 80 format.

```
100 REM Truckers Domain
110 REM Pgm by Sam Moore Jr.
115 REM TEXAS INSTRUMENTS CO
NTRIBUTED SOFTWARE 1983
120 DIM P$(51):: CALL CLEAR
:: FOR CC=1 TO 14 :: CALL CO
LOR(CC,2,2):: NEXT CC :: CAL
L SCREEN(2)
130 PRINT "      TRUCKERS DO
MAIN             ======== ==
===":"":"     You Will Be D
riving a  Truck In The Hill
Country OfTexas."
140 PRINT :"      The Job Wil
l Not Be    Easy, Since You
Must Keep   Your Truck On Th
e Road And  Avoid All Obstac
les."
150 PRINT :"      The Obstacl
es Are:":"":"     1) Giant A
rmadillos         2) Huge Po
tholes            3) Texas S
ized Skunks"
160 PRINT "     4) Giant Jac
krabbits          5) A Herd Of
 Turtles"
170 PRINT : : :"Press Any Ke
y To Continue..."
180 FOR CC=1 TO 14 :: CALL C
OLOR(CC,16,2):: NEXT CC
190 CALL KEY(0,K,S):: IF S=0
 THEN 190
200 FOR CC=1 TO 14 :: CALL C
OLOR(CC,2,2):: NEXT CC
210 PRINT :"      Each Obstac
le You Hit  Or Each Time You
 Run Off TheRoad Costs You A
 Spare Tire.": :" You Have
A Large Truck And"
220 PRINT "You Have Driven T
his Danger-ous Route Before
So You HaveStocked Up With 2
0 Spares."
230 PRINT :"You Gain A Spare
 With Each  Level Completed,
 But When   They're Gone,"
240 PRINT :"You Lose!":"":"B
ut The Fun Is Seeing How   M
any Miles You Can Get Under
Your Belt And How High A Lev
el You Can Obtain."
250 PRINT :"PLEASE USE JOYST
ICK NO. 1!": :"Press Any Key
 To Continue..."
260 FOR CC=1 TO 14 :: CALL C
OLOR(CC,16,1):: NEXT CC
270 CALL KEY(0,K,S):: IF S=0
 THEN 270
280 CALL CLEAR :: CALL SCREE
N(2):: FOR CC=1 TO 14 :: CAL
L COLOR(CC,4,1):: NEXT CC
290 FOR CC=3 TO 8 :: CALL CO
LOR(CC,1,4):: NEXT CC
300 CALL CHAR(47,"FFFFFFFFFF
FFFFFF")
310 RESTORE :: FOR I=98 TO 1
26 :: READ A$ :: CALL CHAR(I
,A$):: NEXT I
320 FOR I=1 TO 50 :: READ P$
(I):: NEXT I
330 SCORE=0 :: LVL=0 :: TR=2
0 :: A=7 :: B=6 :: GOSUB 670
340 FOR LEVEL=1 TO 99
350 READ L :: A=7 :: B=6 ::
CALL CLEAR :: CALL SCREEN(2)
:: FOR CC=1 TO 14 :: CALL CO
LOR(CC,L,1):: NEXT CC
360 FOR CC=3 TO 8 :: CALL CO
LOR(CC,1,L):: NEXT CC :: GOS
UB 670
370 TR=TR+1
380 NEXT LEVEL
390 PRINT "YOU WON IT ALL!"
:: END
400 SCORE=SCORE+1 :: CALL JO
YST(1,P,Y):: A=A+P/4
410 CALL GCHAR(7,A,GC):: IF
GC<>32 THEN TR=TR-1 :: CALL
SOUND(-100,110,30,110,30,899
,30,-4,0):: CALL SOUND(-100,
110,30,110,30,1599,30,-4,0)
420 IF TR<1 THEN 760
430 CALL SOUND(-2000,110,30,
110,30,260,30,-4,9):: CALL H
CHAR(7,A,101):: RETURN
440 DATA FFFFFFFFFFFFCF0C,FFF
CF0C
450 DATA 3C3C3C3C3C3C3C3C,3C
3C3C3C003C3C18,286C6C6C293B7
F02,000003FBFCFCFC94,0F3F7FF
F7F38,0000000CDEDE3F2D,08CCF
EFFE3E
460 DATA 18183C3C7E7EFFFF,00
040C3F0F0F0F0C,0103070F1F3F7
FFF,01060ACAEAF2EC3,80C0E0F0
F8FCFEFF,181E3F7F7F7F1C18
470 DATA FF7F3F1F0F070301,FF
FEFCF8F0E0C08,8080C0C0E0E0F0
F,F8F8FCFCFEFEFFFF,FFFF7F7F3
F3F1F1F,0F0F070703030101
480 DATA 0101030307070F0F,1F
1F3F3F7F7FFFFF,FFFFFEFEFCFCF
8F8,F0F0E0E0C0C0808,00000000
030F3FFF,030F3FFFFFFFFFFF
490 DATA C0F0FCFFFFFFFFFF,00
000000C0F0FCFF
500 DATA //     ////     /////
//////////,//     ////    q///
///////////,//s  u///o    q/
////////////,//t  v////o
q//////////
510 DATA ///s  u////o    q/
/////////,///t  pv/////op
 q////////,////s  u/////o
        DEAD,////t  v//////o
          /END
520 DATA /////s  u//////ofg
 ////////,/////t  v///////o
   ///////,//////    ///////
/o  ///////,DRIVE/ hj <DILLE
R//  SQUEEZE
530 DATA SAFELY     ///////bc
  ///////,/////y  w/////bc
   ///////,/////z  x///bc
   ///////
540 DATA ////y  w//bc
{|///////,////z  x//       {
|//////////,///y lnw/// i i i
TURTLE/HERD,///z  x///
 //////////
550 DATA ///     ////o    q/
/////////,///     /////o
fg     q//,//y p w//////o k
         q/,//z  xMERGE// q
o        /
560 DATA //     uLEFT///  /}‾
  ‾ p  /,//     v/////bc  //
/}‾      /,/r  k  ///bc   m/
////}‾    /,/   /  ///    m/
///////  m/
570 DATA /  p/  ///     /MEDI
AN//  m//,/p  /  ///  hj////
///bc  ///,/s  /hj///o  u//
//bc    ///,/t      ////   v/
/bc    m///
580 DATA //     w////op
p   m////,//s   x/////o   p
p  POTHOLE,//t  u//////o
p   COUNTRY,///s  v///////o
  p  q//////
590 DATA ///tp  u///////o
   q/////,////s  v////////o
      q////,////tp pu////////
o      q///,/////s  v///////
//op   fgq//
600 DATA /////t   u/////////
o      q/,//////s  v////////
//o        /,//////t  u//////
////op  p /,///////s  v/////
//////o     /
610 DATA ///////tp pu///////
///r     /,//MERGE/s  v//////
///r   hj/,//RIGHT/t   u////
///r       /,/////////s  v///
////        /
620 DATA /////////t   u/////
/     p  /,//////////   v////
//        /,//////////ln p///
///  ln    /
630 DATA 12,4,10,8,14,9,3,6,
15,12,4,10,8,14,9,3,6,15,12,
4,10,8,14,9,3,6,15
640 DATA 12,4,10,8,14,9,3,6,
15,12,4,10,8,14,9,3,6,15,12,
4,10,8,14,9,3,6,15
650 DATA 12,4,10,8,14,9,3,6,
15,12,4,10,8,14,9,3,6,15,12,
4,10,8,14,9,3,6,15
660 DATA 12,4,10,8,14,9,3,6,
15,12,4,10,8,14,9,3,6,15,12
670 FOR I=1 TO 20 :: PRINT P
$(I):: NEXT I :: GOSUB 400 :
: CALL SOUND(500,990,0):: LV
L=LVL+1
680 CALL HCHAR(1,1,47,96)::
DISPLAY AT(1,1):"PRESS/FIRE/
BUTTON/TO/START//////////////
//////////LEVEL";LVL:"///////
//////////////SPARES";TR
690 CALL HCHAR(3,28,61):: CA
LL HCHAR(2,28,61):: IF LVL<1
0 THEN CALL HCHAR(2,30,47)
700 IF TR<10 THEN CALL HCHAR
(3,30,47)
710 CALL KEY(1,K,S):: IF S=0
 THEN 710
720 FOR I=21 TO 50 :: CALL V
CHAR(7,A,32):: PRINT P$(I)::
 GOSUB 400 :: CALL VCHAR(7,A
,32):: GOSUB 400 :: NEXT I
730 FOR I=1 TO 50 :: CALL VC
HAR(7,A,32):: PRINT P$(I)::
GOSUB 400 :: CALL VCHAR(7,A,
32):: GOSUB 400 :: NEXT I
740 FOR I=1 TO 16 :: CALL VC
HAR(7,A,32):: PRINT P$(I)::
GOSUB 400 :: CALL VCHAR(7,A,
32):: GOSUB 400 :: NEXT I
750 CALL SOUND(500,990,0)::
RETURN
760 CALL CLEAR :: CALL SCREE
N(2):: FOR CC=1 TO 14 :: CAL
L COLOR(CC,4,1):: NEXT CC
770 FOR CC=3 TO 8 :: CALL CO
LOR(CC,1,4):: NEXT CC
780 CALL CLEAR :: DISPLAY AT
(10,1):"YOU/HAVE/RUN/OUT/OF/
SPARES//YOUR/SCORE/IS:";SCOR
E;"MILES.":"ON/LEVEL";LVL
790 DISPLAY AT(20,1):"PLAY A
GAIN? Y-N"
800 CALL KEY(0,K,S):: IF S=0
 THEN 800
810 CALL SOUND(444,880,0)
820 IF K=89 THEN 330 ELSE EN
D
```

```
100 REM *************
110 REM * BOA ALLEY *
120 REM *************
130 REM BY TARIK ISANI
140 REM 99'ER VERSION 2.6.1
150 REM
160 REM
170 CALL CLEAR
180 CALL SCREEN(2)
190 RANDOMIZE
200 PRINT "      *** BOA ALLE
Y ***": :"              BY":"
      TARIK ISANI"
210 PRINT "YOU MUST DIRECT A
 LONG": :"SNAKE-LIKE OBJECT
THROUGH"
220 PRINT "A MAZE HITTING RO
UND WHITE": :"TARGETS. USE T
HE JOYSTICK"
230 PRINT :"OR THE ARROW KEY
S TO MOVE.": :"IF YOU HIT YO
URSELF, THE"
240 PRINT :"BOUNDARIES OR TH
E DIVIDERS,": :"THE GAME WIL
L END."
250 PRINT :"[PRESS ANY KEY T
O CONTINUE]"
260 FOR I=1 TO 8
270 CALL COLOR(I,16,1)
280 NEXT I
290 CALL KEY(0,S1,S2)
300 IF S2=0 THEN 290
310 CALL CLEAR
320 FOR I=1 TO 8
330 CALL COLOR(I,1,1)
340 NEXT I
350 PRINT :"  METHOD OF INPU
T:": :"    1.ARROW KEYS": :"
    2. JOYSTICK": : : : :
360 FOR I=1 TO 8
370 CALL COLOR(I,16,1)
380 NEXT I
390 CALL KEY(0,01,02)
400 IF (01<49)+(01>50)THEN 3
90
410 CALL CLEAR
420 FOR I=2 TO 9
430 CALL COLOR(I,2,9)
440 NEXT I
450 CALL COLOR(9,10,1)
460 CALL COLOR(11,14,1)
470 CALL COLOR(12,16,1)
480 CALL COLOR(13,5,1)
490 CALL COLOR(14,9,1)
500 CALL CHAR(96,"3C7EFF9999
FF7E3C")
510 CALL CHAR(97,"3C66E7FFFF
E7663C")
520 CALL CHAR(112,"007E7E666
67E7E00")
530 CALL CHAR(120,"3C7EFFFFF
FFF7E3C")
540 CALL CHAR(129,"183C7EFFF
FFFC381")
550 CALL CHAR(132,"F87C3E3F3
F3E7CF8")
560 CALL CHAR(131,"B1C3FFFFF
F7E3C18")
570 CALL CHAR(128,"1F3E7CFCF
C7C3E1F")
580 CALL CHAR(136,"FFFFFFFFF
FFFFFFF")
590 OPTION BASE 1
600 DIM P(105,2)
610 CALL HCHAR(1,2,136,29)
620 CALL HCHAR(23,2,136,29)
630 CALL VCHAR(1,2,136,23)
640 CALL VCHAR(1,30,136,23)
650 FOR I=3 TO 21 STEP 2
660 FOR J=4 TO 28 STEP 2
670 CALL VCHAR(I,J,112)
680 NEXT J
690 NEXT I
700 Q=1
710 X1=10
720 Y1=21
730 M1=0
740 N1=-1
750 FL=0
760 L=0
770 SC=0
780 CALL HCHAR(1,2,136,29)
790 A$="SCORE:0"
800 J=10
810 GOSUB 1920
820 FOR I=6 TO 10 STEP 4
830 FOR J=7 TO 25
840 CALL SOUND(1,2000,0)
850 CALL VCHAR(I,J,132)
860 L=L+1
870 P(L,1)=I
880 P(L,2)=J
890 NEXT J
900 CALL SOUND(1,2000,0)
910 CALL VCHAR(I+1,J-1,131)
920 L=L+1
930 P(L,1)=I+1
940 P(L,2)=J-1
950 FOR J=25 TO 7 STEP -1
960 CALL SOUND(1,2000,0)
970 CALL VCHAR(I+2,J,128)
980 L=L+1
990 P(L,1)=I+2
1000 P(L,2)=J
1010 NEXT J
1020 CALL SOUND(1,2000,0)
1030 CALL VCHAR(I+3,J+1,131)
1040 L=L+1
1050 P(L,1)=I+3
1060 P(L,2)=J+1
1070 NEXT I
1080 FOR J=7 TO 25
1090 CALL SOUND(1,2000,0)
1100 CALL VCHAR(14,J,132)
1110 L=L+1
1120 P(L,1)=14
1130 P(L,2)=J
1140 NEXT J
1150 CALL SOUND(1,2000,0)
1160 CALL VCHAR(15,25,131)
1170 L=L+1
1180 P(L,1)=15
1190 P(L,2)=25
1200 FOR J=25 TO 21 STEP -1
1210 CALL SOUND(1,2000,0)
1220 CALL VCHAR(16,J,128)
1230 L=L+1
1240 P(L,1)=16
1250 P(L,2)=J
1260 NEXT J
1270 RX=INT(RND*22)+2
1280 RY=INT(RND*27)+3
1290 CALL GCHAR(RX,RY,C)
1300 IF C<>32 THEN 1330
1310 CALL VCHAR(RX,RY,120)
1320 FL=1
1330 IF 01=50 THEN 1510
1340 CALL KEY(1,S,T)
1350 IF S<>5 THEN 1390
1360 M1=-1
1370 N1=0
1380 GOTO 1550
1390 IF S<>3 THEN 1430
1400 M1=0
1410 N1=1
1420 GOTO 1550
1430 IF S+1<>1 THEN 1470
1440 M1=1
1450 N1=0
1460 GOTO 1550
1470 IF S<>2 THEN 1550
1480 M1=0
1490 N1=-1
1500 GOTO 1550
1510 CALL JOYST(1,A,B)
1520 IF ABS(A)+ABS(B)<>4 THE
N 1550
1530 M1=-B/4
1540 N1=A/4
1550 CALL GCHAR(M1+X1,N1+Y1,
C)
1560 IF C=32 THEN 1770
1570 IF C<>120 THEN 1650
1580 CALL SOUND(-100,110,0,1
000,0,500,0)
1590 SC=SC+1
1600 A$=STR$(SC)
1610 J=16
1620 GOSUB 1920
1630 FL=0
1640 GOTO 1770
1650 CALL SOUND(-500,-7,0)
1660 CALL SCREEN(12)
1670 CALL SCREEN(2)
1680 CALL KEY(0,S1,S2)
1690 IF S2<1 THEN 1680
1700 FOR I=2 TO 22 STEP 2
1710 CALL HCHAR(I,3,32,27)
1720 NEXT I
1730 FOR I=3 TO 29 STEP 2
1740 CALL VCHAR(2,I,32,21)
1750 NEXT I
1760 GOTO 700
1770 CALL VCHAR(X1,Y1,128+2*
(N1+1)+M1)
1780 X1=X1+M1
1790 Y1=Y1+N1
1800 CALL SOUND(-1,2000,0)
1810 IF M1=0 THEN 1840
1820 CALL VCHAR(X1,Y1,96)
1830 GOTO 1850
1840 CALL VCHAR(X1,Y1,97)
1850 CALL VCHAR(P(Q,1),P(Q,2
),32)
1860 P(Q,1)=X1
1870 P(Q,2)=Y1
1880 Q=Q+1
1890 IF Q<>106 THEN 1910
1900 Q=1
1910 IF FL=0 THEN 1270 ELSE
1330
1920 FOR I=1 TO LEN(A$)
1930 CALL VCHAR(1,I+J,ASC(SE
G$(A$,I,1)))
1940 NEXT I
1950 RETURN
```

continued from page 17

```
1050 G(L)=G(L)-A(L)*G(L-1)
1060 GOTO 1080
1070 G(L)=G(L)-A(L)*G(L-1)-B
(L)*G(L-2)
1080 S1=S1+S(L)*G(L)
1090 NEXT L
1100 U(J)=S1
1110 L=N
1120 FOR I2=2 TO N
1130 G(L)=G(L-1)
1140 L=L-1
1150 NEXT I2
1160 G(1)=0
1170 NEXT J
1180 T=0
1190 FOR L=1 TO M
1200 C(L)=0
1210 J=N
1220 FOR I2=1 TO N
1230 C(L)=C(L)*X(L)+U(J)
1240 J=J-1
1250 NEXT I2
1260 T3=Y(L)-C(L)
1270 T=T+T3^2
1280 NEXT L
1290 IF (M-N)<>0 THEN 1320
1300 T5=0
1310`GOTO 1330
1320 T5=T/(M-N)
1330 Q7=1-T/(T9*(M-1))
1340 NM1=N-1
1350 PRINT "POLYFIT OF DEGRE
 ";NM1
1360 PRINT "INDEX OF DETERM=
 ";Q7
1370 PRINT
```

continued on page 16

```
100 REM ***************
110 REM *             *
120 REM * BATTLE STAR *
130 REM *             *
140 REM ***************
150 REM
160 REM 99'ER VERSION 1.6.1.
XB49
170 REM RANDOMIZE
180 DIR=1 :: CALL CLEAR
190 CALL COLOR(9,7,1):: CALL
 COLOR(10,6,1):: CALL SCREEN
(2)
200 CALL CHAR(96,"0000000000
070707"):: CALL CHAR(97,"181
8183C7EFFDB99")
210 CALL CHAR(98,"0000000000
E0E0E0"):: CALL CHAR(99,"070
E1CFFFF1C0E07")
220 CALL CHAR(104,"18423C999
93C4218"):: CALL CHAR(101,"E
07038FFFF3870E0"):: CALL CHA
R(102,"070707")
230 CALL CHAR(107,"104628240
A923044")
240 CALL CHAR(103,"99DBFF7E3
C181818"):: CALL CHAR(100,"E
0E0E0")
250 CALL CHAR(112,"30787C477
C7830"):: CALL CHAR(113,"101
0386CEEEE7C")
260 CALL CHAR(114,"0C1E3EE23
E1E0C"):: CALL CHAR(115,"007
CEEEE6C381010")
270 CALL CHAR(116,"101038FE3
81010"):: CALL CHAR(117,"000
0183CFF7E2442")
280 CALL CHAR(105,"181818181
8181818"):: CALL CHAR(106,"0
00000FFFF")
290 FOR COL=1 TO 12 :: CALL
COLOR(COL,16,1):: NEXT COL
300 L=100 :: S=5 :: SC=0 ::
SA1,SB1,SA2,SB2,SA3,SB3,SA4,
SB4=0 :: T=0
310 GOSUB 350
320 GOSUB 390 :: GOSUB 650
330 L=L-.5 :: IF L<1 THEN L=
1
340 DISPLAY AT(24,3):SC :: G
OTO 320
350 CALL SPRITE(#10,96,16,81
,113,0,0,#11,97,16,81,121,0,
0,#12,98,16,81,129,0,0)
360 CALL SPRITE(#13,99,16,89
,113,0,0,#14,104,7,89,121,0,
0,#15,101,16,89,129,0,0)
370 CALL SPRITE(#16,102,16,9
7,113,0,0,#17,103,16,97,121,
0,0,#18,100,16,97,129,0,0)
380 RETURN
390 CALL KEY(0,K,S):: IF S=0
 THEN RETURN
400 IF K=69 THEN 450
410 IF K=83 THEN 500
420 IF K=88 THEN 550
430 IF K=68 THEN 600
440 RETURN
450 IF SA1=0 AND SB1=0 THEN
CALL VCHAR(1,16,105,10):: CA
LL SOUND(10,800,0):: CALL VC
HAR(1,16,32,10):: SC=SC-10 :
: RETURN
460 IF SB1=0 THEN CALL VCHAR
(2,16,105,9):: CALL SOUND(50
0,110,2,-5,2):: CALL VCHAR(2
,16,32,9):: SC=SC+50 :: SA1=
0 :: RETURN
470 CALL POSITION(#1,P1,P2):
: IF P1>76 THEN 840
480 P1=INT(P1/8)+1 :: CALL V
CHAR(P1,16,105,10-P1):: CALL
 SOUND(200,110,10,-5,8):: CA
LL VCHAR(P1,16,32,10-P1)
490 CALL DELSPRITE(#1):: SC=
SC+20 :: SB1=0 :: RETURN
500 IF SA2=0 AND SB2=0 THEN
CALL HCHAR(12,1,106,14):: CA
LL SOUND(10,800,0):: CALL HC
HAR(12,1,32,14):: SC=SC-10 :
: RETURN
510 IF SB2=0 THEN CALL HCHAR
(12,3,106,12):: CALL SOUND(5
00,110,2,-5,2):: CALL HCHAR(
12,3,32,12):: SC=SC+50 :: SA
2=0 :: RETURN
520 CALL POSITION(#2,P1,P2):
: IF P2>86 THEN 840
530 P2=INT(P2/8)+1 :: CALL H
CHAR(12,P2,106,15-P2):: CALL
 SOUND(200,110,10,-5,8):: CA
LL HCHAR(12,P2,32,15-P2)
540 CALL DELSPRITE(#2):: SC=
SC+20 :: SB2=0 :: RETURN
550 IF SA3=0 AND SB3=0 THEN
CALL VCHAR(14,16,105,10):: C
ALL SOUND(10,800,0):: CALL V
CHAR(14,16,32,10):: SC=SC-10
 :: RETURN
560 IF SB3=0 THEN CALL VCHAR
(14,16,105,10):: CALL SOUND(
500,110,2,-5,2):: CALL VCHAR
(14,16,32,10):: SC=SC+50 ::
SA3=0 :: RETURN
570 CALL POSITION(#3,P1,P2):
: IF P1<110 AND P1>0 THEN 84
0
580 P1=INT(P1/8)+1 :: CALL V
CHAR(14,16,105,P1-14):: CALL
 SOUND(200,110,10,-5,8):: CA
LL VCHAR(14,16,32,P1-14)
590 CALL DELSPRITE(#3):: SC=
SC+20 :: SB3=0 :: RETURN
600 IF SA4=0 AND SB4=0 THEN
CALL HCHAR(12,18,106,14):: C
ALL SOUND(10,800,0):: CALL H
CHAR(12,18,32,14):: SC=SC-10
 :: RETURN
610 IF SB4=0 THEN CALL HCHAR
(12,18,106,13):: CALL SOUND(
500,110,2,-5,2):: CALL HCHAR
(12,18,32,13):: SC=SC+50 ::
SA4=0 :: RETURN
620 CALL POSITION(#4,P1,P2):
: IF P8<142 AND P8>0 THEN 84
0
630 P2=INT(P2/8):: CALL HCHA
R(12,18,106,P2-15):: CALL SO
UND(200,110,10,-5,8):: CALL
HCHAR(12,18,32,P2-15)
640 CALL DELSPRITE(#4):: SC=
SC+20 :: SB4=0 :: RETURN
650 IF SB1=0 THEN P1,P2=0 ::
 GOTO 660 ELSE CALL POSITION
(#1,P1,P2)
660 IF SB2=0 THEN P3,P4=0 ::
 GOTO 670 ELSE CALL POSITION
(#2,P3,P4)
670 IF SB3=0 THEN P5,P6=0 ::
 GOTO 680 ELSE CALL POSITION
(#3,P5,P6)
680 IF SB4=0 THEN P7,P8=0 ::
 GOTO 690 ELSE CALL POSITION
(#4,P7,P8)
690 IF P1>76 OR P4>B6 OR(P5<
110 AND P5>0)OR(P8<142 AND P
8>0)THEN 840
700 NS=INT(RND*L):: IF NS>10
 THEN RETURN
710 NS=INT(RND*4)+1 :: ON NS
 GOTO 730,760,790,820
720 IF SA1=1 AND SB1=1 THEN
RETURN
730 CALL HCHAR(2,16,115):: S
A1=1 :: IF L<80 AND SB1=0 TH
EN CALL SPRITE(#1,116,7,17,1
20,11-(L/10),0):: SB1=1
740 RETURN
750 IF SA2=1 AND SB2=1 THEN
RETURN
760 CALL HCHAR(12,3,112):: S
A2=1 :: IF L<80 AND SB2=0 TH
EN CALL SPRITE(#2,116,7,88,1
7,0,11-(L/10)):: SB2=1
770 RETURN
780 IF SA3=1 AND SB3=1 THEN
RETURN
790 CALL HCHAR(23,16,113)::
SA3=1 :: IF L<80 AND SB3=0 T
HEN CALL SPRITE(#3,116,7,175
,120,-11+(L/10),0):: SB3=1
800 RETURN
810 IF SA4=1 AND SB4=1 THEN
RETURN
820 CALL HCHAR(12,30,114)::
SA4=1 :: IF L<80 AND SB4=0 T
HEN CALL SPRITE(#4,116,7,88,
216,0,-11+(L/10)):: SB4=1
830 RETURN
840 CALL DELSPRITE(#1,#2,#3,
#4):: CALL SOUND(2000,110,2,
220,2,1000,30,-4,2)
850 FOR BUB=10 TO 18 :: CALL
 MOTION(#BUB,INT(RND*40)-20,
INT(RND*40)-20):: CALL PATTE
RN(#BUB,107):: NEXT BUB
860 CALL SOUND(1000,110,2,22
0,2,110,2,-5,2):: CALL SOUND
(1,40000,30)
870 CALL DELSPRITE(ALL):: CA
LL CLEAR
880 DISPLAY AT(12,7):"YOUR S
CORE IS":TAB(10);SC
890 CALL DELSPRITE(ALL)
900 DISPLAY AT(22,1):"DO YOU
 WISH TO PLAY AGAIN?(Y/N)."
910 ACCEPT AT(23,8)VALIDATE(
"YN"):ANS$ :: IF ANS$="N" TH
EN 950
920 CALL CLEAR :: GOSUB 350
:: SC=0 :: L=100
930 SB1,SB2,SB3,SB4,P1,P2,P3
,P4,P5,P6,P7,P8=0
940 RETURN
950 RUN "DSK1.LOAD"
960 END
```

continued from page 15

```
1380 PRINT "TERM","COEFFICIE
NT"
1390 FOR J=1 TO K
1400 I2=J-1
1410 PRINT I2,U(J)
1420 NEXT J
1430 PRINT
1440 PRINT "* PRESS 1 TO CON
TINUE *"
1450 PRINT
1460 CALL KEY(0,KEY,STATUS)
1470 IF STATUS=0 THEN 1460
1480 GT=KEY-48
1490 IF GT=1 THEN 1570
1500 GOTO 1570
1510 PRINT "ELEVENTH DEGREE
IS THE LIMIT"
1520 GOTO 1650
1530 PRINT "TOO FEW POINTS
FOR FITTING DEGREE ":N
1540 GOTO 1650
1550 PRINT "TOO MANY POINTS-
MAXIMUM IS 100"
1560 GOTO 1650
1570 PRINT "Y-ACTUAL      Y-C
ALCULATED"
1580 FOR L=1 TO M
1590 PRINT Y(L);TAB(12);C(L)
1600 PRINT "DIFF.=";Y(L)-C(L
)
1610 FOR TIME=1 TO 500
1620 NEXT TIME
1630 NEXT L
1640 PRINT
1650 END
```

```
1 REM W.D. GRIFFIN SAN RAFAE
L CA.  94903
100 CALL CLEAR
110 DIM B(100),X(100),Y(100)
120 DIM P(100),Q(100),A(100)
125 DIM S(100),C(100)
130 PRINT "POLYNOMIAL CURVE"
140 PRINT "BY METHOD OF LEAS
T SQUARES"
150 PRINT
160 PRINT
170 PRINT "THE NUMBER OF OBS
ERVATIONS"
180 PRINT "ARE THE NUMBER OF
 X-Y"
190 PRINT "COORDINATES TO BE
 ENTERED"
200 PRINT
210 PRINT
220 PRINT
230 PRINT
240 INPUT "NUMBER OF OBSERVA
TIONS ":M
250 INPUT "LOWEST DEGREE OF
POLYNOMIAL ":N
260 IF M<0 THEN 1260
270 Z=0
280 0=1.0
290 K=12
300 N=N+1
310 IF (N-12)>0 THEN 1510
320 IF (M-N)<0 THEN 1530
330 IF (M-100)>0 THEN 1550
340 T7=Z
350 T8=Z
360 W7=Z
370 FOR I=1 TO M
380 INPUT "X(I) ":X(I)
390 INPUT "Y(I) ":Y(I)
400 NEXT I
410 FOR I=1 TO M
420 W7=W7+X(I)
430 T7=T7+Y(I)
440 T8=T8+Y(I)^2
450 NEXT I
460 T9=(M*T8-T7^2)/(M*M-M)
470 XM=W7/M
480 YM=T7/M
490 STD=SQR(T9)
500 PRINT "LEAST-SQUARES POL
YNOMIALS"
510 PRINT
520 PRINT "NUMBER OF POINT="
;M
530 PRINT "MEAN VALUE OF X="
;XM
540 PRINT "MEAN VALUE OF Y="
;YM
550 PRINT "STD ERROR OF Y=";
STD
560 PRINT
570 PRINT TAB(7);"*  RUNNING
 *"
580 PRINT
590 FOR I=1 TO M
600 P(I)=Z
610 Q(I)=0
620 NEXT I
630 E1=Z
640 F1=Z
650 W1=M
660 N4=K
670 I=1
680 K1=2
690 IF N=0 THEN 710
700 K1=N4
710 W=Z
720 FOR L=1 TO M
730 W=W+Y(L)*Q(L)
740 NEXT L
750 S(I)=W/W1
760 IF (I-N4)>=0 THEN 990
770 IF (I-M)>=0 THEN 990
780 E1=Z
790 FOR L=1 TO M
800 E1=E1+X(L)*Q(L)^2
810 NEXT L
820 E1=E1/W1
830 A(I+1)=E1
840 W=Z
850 FOR L=1 TO M
860 V=(X(L)-E1)*Q(L)-F1*P(L)
870 P(L)=Q(L)
880 Q(L)=V
890 W=W+V^2
900 NEXT L
910 F1=W/W1
920 B(I+2)=F1
930 W1=W
940 I=I+1
950 GOTO 710
960 FOR L=1 TO 13
970 G(L)=Z
980 NEXT L
990 G(1)=0
1000 FOR J=1 TO N
1010 S1=Z
1020 FOR L=1 TO N
1030 IF (L-2)<0 THEN 1080
1040 IF (L-2)>0 THEN 1070
```

continued on page 15

```
********************************
*                              *
*    SIMPLE READ-DATA DEMO     *
*    ==================        *
*                              *
*     M. Amundsen 3-30-85      *
*                              *
*                              *
*   Example program using the  *
* "PULSAR" assembly routines.  *
*                              *
* NOTE:                        *
*   The PULSAR master file     *
* MUST be loaded into memory!  *
*                              *
********************************

* LOAD DEF/REF TABLE

       COPY "DSK2.START-EA/S"

* PUT UP TITLE
{* "BASIC" VERSION *}

BEGIN  BL    @SCRCLR
{100 CALL CLEAR }
       BL    @PRNTAT
{110 DISPLAY AT(1,8):L1$ }
       DATA D1,D8,L1
       BL    @PRNTAT
{120 DISPLAY AT(2,8):L2$ }
       DATA D2,D8,L2
       BL    @PRNTAT
{130 DISPLAY AT(3,8):L3$ }
       DATA D8,D2,L3

* READ and DISPLAY INTEGERS

NLOOP  BL    @SETDAT
{140 RESTORE "DLIST1" (INTEGER DATA) }
       DATA INTDAT,DLIST1
       BL    @FOR
{150 FOR LOOP=10 TO 19 STEP 1 }
       DATA LOOP,D10,D19,D1
       BL    @GETDAT
{160 READ DTEMP }
       DATA DTEMP
       BL    @INTFLT
{170 FLTEMP=DTEMP (TURN INT TO FLT) }
       DATA DTEMP,FLTEMP
       BL    @NUMSTR
{180 STRING$=STR$(FLTEMP) }
       DATA FLTEMP,STRING
       BL    @PRNTAT
{190 DISPLAY AT(LOOP,8):STRING$ }
       DATA LOOP,D8,STRING
       BL    @NEXT
{200 NEXT LOOP }
       DATA LOOP


* READ and DISPLAY STRINGS

SLOOP  BL    @PRNTAT
{210 DISPLAY AT(8,15):L4$ }
       DATA D8,D15,L4
       BL    @SETDAT
{220 RESTORE "DLIST3" (STRING DATA) }
       DATA STRDAT,DLIST3
       BL    @FOR
{230 FOR LOOP=10 TO 19 STEP 1 }
       DATA LOOP,D10,D19,D1
       BL    @GETDAT
{240 READ STRING$ }
       DATA STRING
       BL    @PRNTAT
{250 DISPLAY AT(LOOP,15):STRING$ }
       DATA LOOP,D15,STRING
       BL    @NEXT
{260 NEXT LOOP }
       DATA LOOP

* WAIT FOR KEYHIT

       BL    @PRNTAT
{270 DISPLAY AT(23,5):L5$ }
       DATA D23,D5,L5
REGET  BL    @KEYCON
{280 CALL KEY(0,K) *NO "S" }
       BL    @IFTHEN
{290 IF K=QUIT THEN 310 }
       DATA KEYHIT,EQ,QUITKY,EXIT
       BL    @IFELSE
{300 IF K=FCTNS THEN 100 ELSE 280 }
       DATA KEYHIT,EQ,FCTN8,BEGIN,REGET

* END PROGRAM

EXIT   LIMI 2
{310 *TURN ON INTERRUPTS* }
       JMP  $
{320 GOTO 320 }

* VARIABLES

L1     DATA 14
       TEXT 'READ-DATA DEMO'
       EVEN
L2     DATA 14
       TEXT '=============='
       EVEN
L3     DATA 8
       TEXT 'NUMBERS:'
L4     DATA 8
       TEXT 'STRINGS:'
L5     DATA 18
       TEXT 'PRESS REDO OR QUIT'
LOOP   DATA 0
FLTEMP BSS  8
STRING BSS  20
DTEMP  BSS  20
DLIST1 DATA 1,2,3,4,5,6,7,8,9,0
DLIST3 DATA 1
       TEXT 'A'
       DATA 2
       TEXT 'BB'
       DATA 3
       TEXT 'CCC'
       DATA 4
       TEXT 'DDDD'
       DATA 5
       TEXT 'EEEEE'
       DATA 6
       TEXT 'FFFFFF'
       DATA 7
       TEXT 'GGGGGGG'
       DATA 8
       TEXT 'HHHHHHHH'
       DATA 9
       TEXT 'IIIIIIIII'
       DATA 10
       TEXT 'JJJJJJJJJJ'
       EVEN
       END
```

Review of Desk Top Publisher

by Ron Prewitt, USA

Desk Top Publisher is a cartridge program produced by DataBioTics that allows you to create a graphic picture and then include the picture in your text. The text can be printed in one to three columns with an Epson compatible printer.

Performance:

The documentation recommends that the console be turned off when inserting the cartridge module. The title of the module will appear on the master selection list as "2" on the TI or Myarc and "3" on CorComp controller card. The documentation does not mention that you must use the space bar to get to the secondary selection screen with the CorComp otherwise the module will not function.

The program consists of three major sections that are selected from the main menu. These are "1" PICTURE MAKER, "2" WORD MAKER, and "3" PRINT PAGE.

The PICTURE MAKER is a graphics or drawing program that has many of the drawing functions of other graphic programs like TI-Artist, Graphx etc.. The drawing modes are represented by a single key input. The drawing modes are Draw, Point, Frame, Box, Circle, Disc, Fill, Line, Connect Line, Rays Horizontal. The crosshair shaped cursor can be moved about with either the joy stick or the FCTN "arrow" keys. The mode is activated by either the ENTER key or joystick fire button. There is a text mode that lets you type in the drawing area. You can select different size fonts with the FCTN[1] through to 0 keys. The other functions are CLEAR to clear the work area, Save Picture to disk or casssette and Load Picture from disk or cassette. There is no mention of being able to use pictures created by any of the other drawing programs.

WORD MAKER is the text input program. You will first be asked to choose 1, 2 or 3 columns for inputting your text. Choosing 1, 2 or 3 columns will allow input of 78, 39 or 26 characters per line respectively. Making this selection will then take you to the text editor screen. The first task is to position the picture that was created or loaded from the PICTURE MAKER. Using the FCTN "arrow" keys or the joystick will position the picture anywhere on the page. To set the picture position, use ENTER or the joystick fire button. This will make the text editor ready to accept your input. The editor will only display 5 straight forward. It allows input of your printer device (the default is "PIO.CR") and whether to include the picture in the printed output.

Ease Of Use:

The program is fairly easy to use. Most everything is menu driven with The screen can be scrolled horizontally to view the entire line. The very top line of the screen shows the location of the cursor by column. row and the position within the line.

The bottom of the screen displays a graphic representation of the entire page showing the position of the cursor and the picture. The screen also has framed areas that show several status conditions.

The editor functions are Delete Character, Insert Character, Delete Line and Insert Line. There are no Move, Copy, Replace String or Reformat functions.

Other utility commands are Roll-up, Roll-Down, Page-Right to scroll to the right, Word-Wrap toggle. Previous Menu, Save Text, Load Text, Place Picture and Select Text Style. The last four functions can be selected from either assigned function keys or the Editor Menu.

The saved text should be re-loaded in the same 1, 2, or 3 column mode it was originally created and saved as Loading text that was saved as 1 column when you are in 3 column mode will truncate the text beyond position 26.

The Text-Style function allows the selection of several type styles. The type style chosen will affect the entire line.

There is no capability to limit the type style to a word or several words. The type styles available are Normal, Italics, Bold, Emphasized and Underline. More than one type style can be selected for a line in combination; an exmple is Bold and Emphasized.

The text buffer will only hold one page regardless of column format. If you need additional pages for your text input, they must be created and saved as separate files.

The PRINT PAGE selection is pretty yourself. This RPG was the FIRST, in AUSTRALIA to incorporate users playing against (and killing if you do not like your opponent), other USERS as opposed to just playing the computer. You have the choice of PROVING yourself against easy to follow prompts.

One thing that would make the program a lot easier to use is being able to reformat the text although lines can be inserted, you end up having to retype a lot of text to eliminate having a real short line.

Another inconvenience is losing the special type styles you have set when the text is saved and then loadedd back in from disk. They are not lost when saved and loaded back in from cassette.

Another feature that would have made it easier is Right-Justify to eliminate the ragged edge on the Right edge of the text. This can be done manually by turning the Word-Wrap mode off and inserting additional blank spaces between words. It also would have helped, if the program would have automatically caused the text to bypass the Picture area. Typing text in the Picture area will overlay the text on the Picture when printed. There is an on-screen status box that indicates when your text is in the Picture area but it is still easy to end up with you busily typing in your text. You also have to remember that if you insert lines the type styles you have set will be off by the number of lines inserted. The PICTURE MAKER would have been more functional if it had the capability to work with pixels in a zoom or magnify mode. Being able to use pictures from other graphic programs would have been helpful also.

Documentation:

The documentation consists of a 7 page booklet including the Contents and In Case of Difficulty pages.

There was also an addendum insert of corrections to the booklet. This still only provided "bare bones" information. There was no explanation of the Status Boxes or that some of the type styles would be used together on the same line. These are just a couple of examples of information that could have been provided.

Value:

The value is greater for those with an expanded system. It is a minimal text processor that allows you to prepare your text in 1, 2, or 3 columns. Although the advertisments show a page in a printer of almost a full page of graphics, the Picture area is actually only about 7 rows by 27 columns of text. There is only the capability to use one picture per page. Note: This article was prepared using Desk Top Publisher. o

For Sale

MAIL TO : ALL
MAIL FROM : SECRETARY

Two ex members have advised that they have the following items for sale:- Peter, telephone (02)389 7025 has Multiplan for $30, TI-Writer for $25 and TE2 for $35.

Jeff, telephone (02)631-7839 has a number of modules at $8 each plus Editor/Assembler for $50.

MAIL TO : ALL
MAIL FROM : DOBELL

I have a limited number of A4 tractor feed plain white printer paper for sale. It is in boxes of 1500/2500 sheets and is for sale at $10.00 a box

Ring me at home, (02)699 5137 day time or evenings.

Geneve Update

by Jack Adams, Canada

Myarc has just recently mailed out to those of us who have sent in our Geneve WARRANTY REGISTRATION forms, three diskettes of update software.
Diskette 1 - MYARC-DOS Version 1.01
2 - MYARC GPL Interpreter Version 0.99
3 - Cartridge saver; Multiplan Upgrade; MYWORD Processor Version 1.10

Apparently, no further upgrades are to be expected for Multiplan or Myword Processor. On testing MYARC-DOS, I have noticed that DSK5 cannot be formatted and therefore cannot be used unless you are working from GPL. In the GPL mode of operation, DSK5 is automatically formatted as double sided 720 sectors. Version 0.98 did not do this.

As promised, here is a brief review of MYARC'S MYART.(note the T)

This software is loaded directly from MYARC-DOS by simply entering MYART at the A> prompt for DSK1. The version in review is 1.0. Something that will help the first user immensely is a template that fits over the function keys. This template contains all the commands required from the key board and the mouse is available from Canaria Data Incorporated, 264 Weber St. W, Kitchener, Ontario, N2H 4A6.

I am writing this as a first user myself. I have tried TI-Artist previously and find MYART superior in ease of handling and in artwork in general. Much finer work can be expected of MYART. There are two resolution modes, 256 and 512. You should note here that if you start a picture in either mode, you cannot transfer one to the other without losing all your work. A picture saved in one mode will not load into the other mode from the disk. Also note that if the diskette becomes full, there is no warning that this is occuring.You may end up with only part of your work saved. The saving routine stops when the diskette is full and no warning is given. It is best to check your diskette directory before using it.

512 Mode

512 mode offers finer detail and ability to mix your own colours from the basic Red, Green and Blue combinations. You may also choose from a palette of 16 colours that will be displayed at the bottom of the screen. The art work achievable with 512 mode is truly amazing. With zoom and control over the speed of movement of the icon, fine detail can be displayed easily. The pixel shape in this mode is a vertical rectangle; consequently any circles drawn will be elliptical. Unfortunatly there is no way to correct this.

256 Mode

When MYART is first loaded, it is automatically in this mode of operation. This mode is identified by the fact that a multicoloured bar appears at the bottom of the screen. This bar contains 256 colours which are not all displayed at one time.

The colour bar can, however, be moved left or right along the bottom of the screen by use of either the mouse or the arrow keys. The behaviour of the colour bar gives the appearance that it is on an endless drum. The pixel shape in this mode is square and the circles drawn are nearly circular (actually a little like horizontal ellipses).

In General.

There are 15 commands in all. A command is enacted whenever the first letter (usually) of the command is typed from the keyboard. These commands will produce Boxes, Rectangles, Circles and Straight lines and allow free-hand sketches in any colour. They will also allow you to gain access to a Help File, to Load and Save, Format a disk, Cut and Paste, Type Text, Fill a space with a chosen colour and provide several levels of Zoom.

By depressing CTRL plus a letter, it is a simple matter to toggle between 512 and 256 mode, rotate text in 90-degree steps, clear the screen, display a disk directory, change the icon colour and print what is on the screen. When printing the screen you have two sizes available and you can print a picture in shades of grey or in outline only. (Sorry, I do not think colour printing is possible.)

The mouse has three buttons. One button toggles the colour palette, another controls the levels of zoom and erases the most recently produced line or product (depressing it a second time brings it back) and the third enables colour change and engages the activity desired on the screen. The speed of movement of the icon can be controlled in five stages by simply pressing the number keys 1 to 5. This is important, particularly when working on details while in zoom format mode.

All in all I feel that this is a super product that is available locally for about $200 from any of the Geneve dealers listed in this bulletin. o

TI-Writer Format Commands

Author unknown

Text Dimension commands, as the name implies, move or shape the words in the document (margins, line spacing, right justify, etc.)

.FI	: Fill	: Puts as many words on a line as will fit in the margins.
.NF	: No Fill	: Cancels fill.
.AD	: ADjust	: Aligns the text to the left and right margins (right justify).
.NA	: No Adjust	: Cancels adjust.
.LM n	: Lf Margin	: Sets the left margin to "n".
.RM n	: Rt Margin	: Sets the right margin to "n".
.SP n	: SPace	: Leaves one or "n" blank lines at this point.
.LS n	: Line Sp	: Sets the line spacing to "n" lines for each new line.
.PL n	: Pg Length	: Defines number of lines to a page.
.BP	: Begin Pg	: Forces the start of a new page.

Internal Format commands control the spacing of characters on a line.

.IN n	: INdent	: Creates an indent from the left margin.
.CE n	: CEntre	: Centres the next "n" lines between the margins.

Highlighting commands control functions such as underline or bold and allow you to redefine characters to use them to send control codes to the printer.

^	: required : space	: Joins words together when required : to prevent splitting in reformatting, underline, bold, etc.
&	: underline	: (underscore) Underline all text following until next space.
@	: bold	: (overstrike) Retypes following text (until next space) four times.
.TL xx	: Trans- : Literate	: Allows reassignment of one : character to represent a number of character codes, to send codes to the printer. The full format is .TL n1:n2,n3,...
.CO t	: COmment	: Similar to REM in BASIC; allows notes that do not print.

Page Identification commands print notes in the top or bottom margins of each page, either headers or footers.

.HE t	: HEader	: Prints text (t) (and page number if % symbol is present) at the top of each page.
.FO t	: FOoter	: Prints text (t) (and page number if % symbol is present) at the bottom of each page.
.PA n	: PAge #	: Resets page number in .HE and .FO

File Management commands.

.IF f	: Include : File	: Merges a file to print a document : too large for one file.

Mail Merge Option commands are used to supply values to the variables in a letter that has been set up for the mail merge option

.ML f	:Mail List	: Identifies value file (f) for mail list.
n	:Variable	: Inserted in text as variable for assignment from value file.
.DP n:t	:Display : Prompt	: Prompts you using text "t" to : assign a value to variable (*n*).

o

PULSAR Assembly Language Utilities

by Michael Amundsen, USA

THE SOBERING ASPECT OF ASSEMBLY

If you are like me, you are probably one of those people who bought the Editor/Assembler package because it was "a good deal" and not because you knew everything there was to know about writing in assembly language. You probably were as foolish as I was and assumed that the big fat book that came with the Editor/Assembler package would "explain it all." Since the documentation is less than enlightening, this probably also means that, like me, you are not churning out skads of super-fast bit-byte-ing assembly programs either. You have discovered, like I have that writing assembly programs is a complex, error-prone and frustrating way to get things done. It may be true that a program written in assembly runs fast, but it takes forever to write!

THE HARD PART

This past winter I resigned myself to learn 9900 Assembler. It took many sleepless nights, but I finally got the hang of it. That is when I realized how much longer each assembly program can be. When I need to get a string from the keyboard in BASIC I just write INPUT A$. In assembler it takes almost 100 lines of code to do the same thing! Not only that, I also realized that each time I write an assembly program I have to "re-write" the same simple routines! What a waste!

What I needed was a set of "portable" utilities that would make writing assembly easier. Ones that would handle the boring stuff like, getting a line of input, printing text on the screen, performing the math fnctions, etc. Why re-write them every time? That is where the Programmer's Utility Library of Specialised Assembly Routines or PULSAR comes in.

PULSAR MAKES IT EASY

PULSAR is a set of 'black box' routines designed to take the drudgery out of assembly progrmming by performing the tasks usually covered by statements and functions in high-level languages like BASIC, Forth, Pascal, etc. By using PULSAR, the programmer can write relatively "bug-free" assembly code in less than half the time it takes to write assembly code from scratch. Of course, this speed in the development stage will be paid for in a slightly slower running time, but in most cases, the delay will be minimal.

PULSAR is an attempt to bring the speed of assembly language, the extensibility of Forth, the modularity of Pascal and the english-like syntax of BASIC all into one. At present, PULSAR is just a series of routines that are accessed via the BL instruction in TI9900 assembly code. The programs are written using Editor/Asssembler syntax coventions and must follow all the same rules as writing "pure" assembly code. Eventually, PULSAR will have its own editor and interpreter, thus making a true "programming language." Of course, the interpreter will be written using PULSAR!

WHAT CAN PULSAR DO?

The current version of PULSAR (V1.2) contains 50 routines covering all the floating point mathematical operations, simple keyboard/screen I/O (print, input, keyboard scan, etc.), graphics commands (CALL COLOR, CALL CHAR, etc.), flow control routines (IF-THEN-ELSE and FOR-NEXT-STEP), and memory manipulation routines (PEEK and POKE, READ-DATA and arrays). In addition, several frequently used values are "pre-defined" to make generating TI9900 code easier and faster.

In most cases, the "syntax" for PULSAR follows the conventions of TI BASIC. This means even those with little assembly experience can learn to write programs very quickly. Advanced programmers may not ever use the more simple routines (the integer mathematics set, for example), but the more complex ones (like the floating point set) will be appreciated by even the skilled TI9900 programmer.

HOW DO I USE PULSAR?

All PULSAR routines are compiled into a library file called PULSAR/O. This file contains all fifty routines and the "pre-defined variables". To access this material, the programmer simply uses the TI9900 instruction BL and passes the necessary data to the routine. Below is an example of the PULSAR version of the TI X-BASIC statement DISPLAY AT:

```
     BL   @PRNTAT
     DATA ROW,COL,MSG
```

In PULSAR, all data is stored in pre-declared variables. This means that all variables are pointers to the data, not the data itself. In the above example the following actual numbers are involved:

```
     ROW = >A000
     COL = >A002
     MSG = >A004
     >A000 = >000C
     >A002 = >0010
     >A004 = (* ASCII chars *)
```

The above information tells the PULSAR routine to print the data stored at MSG on the screen row stored at ROW and the screen column stored at COL. This may seem a bit cumbersome at first, but it is easy once you get the hang of it.

It is also important to note that since all PULSAR routines are accessed via the BL instruction (not the BLWP), the routines use the host program registers. This means you cannot use your workspace registers to store values - they will most likely be corrupted by the PULSAR routines. It is best to store any important data at a pre-declared address.

Since the PULSAR routines allow for "nesting," there is a return stack and a set of loop stacks. These are indexed using R9. Any alteration of R9 could cause the system to crash. Also, R11 is used as the return stack and variable-passing register and must be left alone.

There are other consideratios when mixing PULSAR and "pure" assembly but, instead of boring some of us with that info here, I will talk more about that at the upcoming New Horizons Demonstration of PULSAR. For now, let us look at an example program.

EXAMPLE PROGRAM

The program we will look at, will demonstrate many of the routines in PULSAR including converting floating point data to integer, using READ-DATA commands, scanning the keyboard and printing data on the screen. As an aid, I have included a BASIC "translation" of the PULSAR code for those of us who are more familiar with TI BASIC.

Housekeeping

TI9900 code requires that any external references be declared ahead of time. This directs the assembler to search a list in memory of existing routines that may have been written in another program. This is handled with the include file INIT-EA/S. This line *must* be the first actual line in any PULSAR program. This file also contains the workspace and program entry information. All PULSAR programs start with "RUN."

Title Screen

The next lines of code simply clear the screen and print the program title lines. Note the use of D1, D8, etc., to indicate row and column numbers. The decimal values 0 to 40 are "pre-defined" as D0-D40 and are contained in the Housekeeping file called INIT-EA/S.

Read and Print Integers

In PULSAR, the READ-DATA routines are a bit different than those in BASIC. You must first position the READ-pointer to the start of the data list. You must also tell the routine the data type it will be accessing. In this case we will be reading integer data starting at the address DLIST1.

PULSAR loops require not only the usual holding variable, start and stop value, but always require a step value (in this case the step value is one).

The PULSAR equivalent of BASIC's READ is GETDAT. Since the information is stored as integer type, we will first convert the data into a floating-point type, then convert the floating-point type into a string type, then display it on the screen. This is not the fastest way to do it, but it does show off more PULSAR commands!

Read and Display Strings

The next section of code does the same thing with string data. Note that each string starts with a word that contains the length of the string. This is vital to the operation of the PRNTAT routine.

Scan the Keyboard

After printing the data, the keyboard is scanned for any keystroke (BL @KEYCON) and then the program

continued on page 26

How Programs are Stored on Disk

by Terry Atkinson, Canada

In the Feb issue, Tim had an excellent write-up on how BASIC programs are stored in memory. This is required reading for it will help you to understand what I am about to put forth here.

To re-iterate, programs are stored in memory in two sections. A line number table and (program) statement table (tokenized). Each segment of the line number table consists of 4 bytes of information. The first two bytes are the line number, and the next two bytes point to the address in memory where the actual statement is held. Once a program is executed ("RUN"), the line numbers are not used unless a GOTO/GOSUB is encountered. Then the computer will look up the new line number in its directory and begin to execute the program sequentially at the new location. Now, back to the disk.

When a program is "saved" to disk, certain pointer information is put on the disk, followed by the line number table and followed by the actual program statements. Remember, though, the lines of the program are stored just as they sit in memory, so you are going to be looking at your program backwards; highest line number first, then lowest line number. I will not deal with what happens when you "insert" a line in the middle of a program at this time.

The pointer information is stored in the first 8 bytes of the first sector of the disk used by the program. It contains information such as start and end addresses of the line number table, address of highest memory location used by the program, etc. These 8 bytes are followed by the complete line number table. If the program is a long one, the line number table will take up several sectors. Finally the program statements themselves are placed on the disk (again, in reverse order). OK. Enough for generalities. Let us get down to specifics.

First, initialize a disk and key-in the program below and save it to disk. I have done it in Extended BASIC, but you can do it in BASIC if you wish, however, some of the bytes will be different. Alternatively, you can follow the example in this article. I think "hands-on" is much better, though. Key in this program; do not put any extra spaces or different line numbers!

```
100 REM TEST PROGRAM FOR TINS NEWSLETTER
110 CALL CLEAR
120 PRINT "WE ARE IN NEED OF ARTICLES FOR OUR
    NEWSLETTER!"
130 END
```

Now, boot up your disk-fixer and have a look at sector >22. It should look like the example shown below:

```
ADDR =  0 1  2 3  4 5  6 7  8 9  A B  C D  E F
-----------------------------------------------
0000 = 0017 3773 3764 37D7 0082 3775 0078 3778
0010 = 006E 37AB 0064 37B5 028B 0032 9CC7 2E57
0020 = 4520 4152 4520 494E 204E 4545 4420 4F46
0030 = 2041 5254 4943 4C45 5320 464F 5220 4F55
0040 = 5220 4E45 5753 4C45 5454 4552 2100 099D
0050 = C805 434C 4541 5200 239A 2054 4553 5420
0060 = 5052 4F47 5241 4D20 464F 5220 5449 4E53
0070 = 204E 4557 534C 4554 5445 5200 AA3F FF11
0080 = 0030 0000 0200 01D4 4553 5458 4220 2020
0090 = 2000 0000 0000 0100 0000 0000 0000 0000
00A0 = 0000 0022 0000 0000 0000 0000 0000 0000
00B0 -> END all zero's.
```

Let us first examine the pointers at addresses >0000->0007 inclusive.

ADDR	Contents(word) and remarks:
0000	>0017 - This is the exclusive or (XOR) of the words at >0002 and >0004. See elsewhere in this issue for further instructions. This word tells TI DOS that the file it is accessing is a program file, and whether or not it is a "protected" file. (eg. "SAVE DSK1.TEST,PROTECTED"). If it is protected, this word will be in its 2's (16's) compliment form.
0002	>3773 - End address of the line number table (14195).
0004	>3764 - Start address of line number table (14180).
0006	>37D7 - Address of the highest memory location used by the program itself (14295).

That is it for the pointers. Next comes the line number table. Each line number takes up two words (4 bytes). Our program (above) contains 4 lines, therefore the next 8 words are for the line number table. The first word is the actual line number, and the second word is the address of the beginning of that line. Remember the actual lines are stored in reverse order, as are the statements. Let us look at the line number table closely.

0008	>0082 - Line # of the last line (130)
000A	>3775 - Address of where the last line starts (14197).
000C	>0078 - Second last line number (120).
000E	>3778 - Address of the above line (14200).
0010	>006E - Third last line number (110).
0012	>0064 - Fourth last line number (our first line) (100).
0016	>37B5 - Address of the above line (14261).

Since our example contains only 4 lines, the line number table ends at >0016/>0017. Of course, if our program were longer, so would the line number table. Next comes the statements themselves, each preceded by a byte of data which indicates the length of the statement. Looking at >0018, the byte is >02. Look at the program. Our statement is "END", which is tokenized at >8B. All statements are flagged with a "00" at the end, therefore, the byte counter (>02) counts this flag as well.

0018	>02 - Byte count of the last statement (2).
0019	>8B - Token for "END".
001A	>00 - End of statement flag.
001B	>32 - byte count of second to last statement (50).
001C	>9C - Token for "PRINT".
001D	>C7 - "string follows" flag.
001E	>2E - Length of string (46 chars).
001F-004C	- Ascii for->WE ARE IN...ETC...LETTER!
004D	>00 - End of statement flag.
004E	>09 - Byte cound of 3rd last statement (9).
004F	>9D - Token for "CALL".
0050	>C8 - "command string follows" flag.
0051	>05 - byte count of non-tokenized command.
0052	>43 - C
0053	>4C - L
0054	>45 - E
0055	>41 - A
0056	>52 - R
0057	>00 - End of statement flag.
0058	>23 - Length of final (first) statement (35).
0059	>9A - Token for "REM".
005A	>20 - Space
005B-007A	- Ascii for the REM statement.
007B	>00 - End of statement flag.

So that takes care of the nitty gritty stuff. However, as you may notice, there is still more to come. Some of the next lot of junk escapes me, others I can figure out, but cannot understand the reason for it being there. I will give it my best shot. Remember, I am not a professional!

007C-007E - Characters >AA >3F >FF seem to be found on all program types. Perhaps an "end of program" marker. Notice that, in the FDBs, the byte at >0010 points to this marker.

007F >11 - Unknown, but present on all programs I have examined.

0080 >03 - Default for CALL FILES. If I had done a CALL FILES(1) prior to entering/saving this program, >0080 would contain >01.

0081-0083 - Unknown. Set to zero on programs I have examined.

0084 >02 - Location of the FDB fo this program.

0085 >00 - Unknown.

0086 >01 - Drive number used when program was saved.

0087 >D4 - This has got to be the stangest of all. For some reason, TI has seen fit to add >80 to the first letter of the filename. (in this case, a 'T'). Note that >54+>80=>D4. Coincidentally, >80(128) is one of the mystery tokens.

0088-0091 - ASCII for the remainder of the filename.

0092-0095 - Unknown. Set to zero in those I have examined.

0096 >01 - Number of sectors in the program (less FDB).

0097-00A2 - Unknown. Set to zero.

REMAINDER - Block links again. Do not ask me why they are there. Another of TIs strokes of genius?

Well, that is about it for programs. Do not bother informing me if I missed something, but I sure would like to know (curious) what all those "unknowns" are all about. Remember that the sector addresses will be different for all programs. There is no substitute for going through it word-by-byte-by-nybble just like I did above. After all, it took me only 3 hours and 20 minutes to produce the first draft of this article! o

Dealing with Fractured Files

by Terry Atkinson, Canada

In the February issue of the TINS newsletter, I introduced, among other things, details on how TI DOS handles fractured files. This month, I wish to further expand on that segment. Before I do, however, I wish to bring to your attention, bytes >0012 and >0013 of the File Descriptor Block (FDB). In the article, I described their use for "FIXED" type files. Contrary to some people's belief, they ARE used for variable type files as well, but since they more-or-less duplicate other sector addresses, I left that information out. In fact, for "variable" type files, byte >12 contains the low order bye for the number of sectors used in the file (same as byte >F) and byte >13 contains the high order byte for the number of sectors in the file (same as byte >E). These bytes are not used by program type files.

Having cleared the air, let us get on with the main theme: Multiple Fractures. Recall that a file may be fractured up to 76 times (that is all the room there is in the FDB to handle 6 byte fracture pointers). For my example, I will use the following. I am looking at FDB >3 and find a program called "TEST". It contains several fractures, the pointers starting at sector >3 address >001C are:

```
ADDR: 001C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A
CONT:   74 50 00 88 30 01 38 31 02 05 40 02 07 50 02
               2B 2C 2D 2E 2F 30
               0C 60 02 14 20 03
```

The remainder of the sector contains all zeros.

Recall that we grouped the bytes in groups of 3, then numbered each nybble from 1 to 6. Then we rearranged the nybble order to read 412563. The 412 nybble became the start sector for the fracture, and the 563 became the number of sectors in the offset. Since we are dealing with multiple fractures, I will mention at this point that the sector offset is cumulative; it always keeps track of the TOTAL number of sectors from the beginning of the file. Also, for the first fracture, you should add "1" to the value, since a file which takes only 1 sector has an offset of "0". I am sure you can appreciate, in simple mathematics, that 9-1=8, but counting on your fingers from 1 to 9 produces 9 different combinations. "00", therefore, is a combination. So here are the bytes grouped and numbered.

```
Bytes: 745000 883001 383102 054002 075002 0C6002
#'d  : 123456 123456 123456 123456 123456 123456
              142003
              123456
```

Now, rearrange in 412-563 groups and separate them. The groups should now look like this:

074 005, 088 013, 138 023, 005 024, 007 025, 00C 026, 014 032

OK, the hard part is done. We can safely say our program contains 7 fractures. The first fracture starts at sector >074 and continues for >005 sectors. But lets analyze all the fractures with all numbers in hexadecimal with their decimal values in brackets after them.

First fracture: Starts at sector >074(116) and continues for >005 sectors. The end sector for this fracture is >74+>5=>79(121); a total of 6 sectors 5+1.

Second fracture: Starts at >088(136) and continues for >013-(>5+1)=>D(13) sectors. The end sector for this fracture is >88+>D=>95(149).

Third fracture: Starts at >138(312) and continues for >023-(>13+1)=>F(15) sectors. The end sector will be >138+>F=>147(327).

Fourth fracture: Starts at >005(5) and continues for >024-(>23+1)=0 sectors. The end sector is >5+>0=>5(5).

Fifth fracture: Starts at >007(7) and continues for >025-(>24+1)=0 sectors. The end sector for this fracture is >7+>0=>7(7).

Sixth fracture starts at >00C(12) and continues for >26-(>25+1)=0 sectors. The end sector for this fracture is >C+>0=>C(12).

Seventh and final fracture is at >014(20) and continues for >032-(>26+1)=>B(11) sectors. The end sector for this fracture (and the program) is >14+>B=>1F(31).

The total number of sectors used for this file is the final offset plus 1 (>32+1=>33(51) sectors). If you were to catalog this disk, it would show 52 sectors used for this program. Do not forget the FDB!!

OK, so, I think that explains the subject a bit better. Have a look at a few files of your own to see if you can find the start and end sectors and fracture points. With very little practice, you will be able to read the block links without problems. Guaranteed! o

Electronic Bible

If anyone wants the KJ Bible as text on disk, contact: Raymond K Hamilton, Rt #2, WILDER, Idaho, USA, 83676. It needs around 70 SSSD disks! Do not forget to send $$$ or IMOs when writing.

=======

continued from page 5

The '.VA' command can also be used to supply arguments for other specification commands. Consider an example using the time delay command, '.TD'. At the start of the program, you may wish a message to be displayed for a longer period of time than later, when the user is familiar with the messages given. The following code demonstrates this.

```
100 CALL LINK("XDP")
110 DELAY=50
120 CALL LINK("DISPLY",1,1,"THE USER READS THIS!.TD(.VA
   )",DELAY)
130 ! REST OF PROGRAM.
```

This will print the message for about five seconds and then continue with the rest of the program. If however, later in the program you wish this delay to be two seconds, the value of DELAY should be changed to 20, and provided line 110 is not executed, this shorter delay will occur next time the message is printed.

Further references can be found in the XDP utility guide that can be printed from the demo program. Read pages 2-10, as well as the following commands: CHAR (page 15), CHRPAT (page 16), COLOR (pages 18-19), DISPLY (pages 20-21), SCREEN (page 30) and XDP (page 37)

Remember: XDP is available from the club shop as either double sided or flippy disks.

Questions about the package may be forwarded to my home address:
Mr. Craig Sheehan
21 Suzanne Rd.,
Mona Vale 2103

Next Month: We will examine XDP's enhanced ACCEPT command, forty column mode and scrolling. c

Enhanced BASIC

by Stephen Shaw, UK

The original console design concept was that there would be a peripheral device into which you could insert several modules. These would ALL appear on your opening menu screen for selection; that is, you could select modules by software instead of having to use a mechanical switch. Further, subroutines in each module would be available to you either in machine code or BASIC.

The peripheral was never built, but you can still sometimes see its shadow with the error message "Review Module Library". And two modules have routines which you can use in TI BASIC.

If you insert either the Personal Record Keeping or Statistics modules into the port, and select TI BASIC, you have access to additional CALLS which we shall describe here and in forthcoming articles.

If you have a disk system, and run the PRK/Stats modules from disk (using any suitable loading module or peripheral), after loading the module code, you will find yourself looking at a menu offering say:

1. TI BASIC
2. EXTENDED BASIC
3. PERSONAL RECORD KEEPING.

If you select 1. TI BASIC, the extra calls are still available, just as though the actual module was inserted.

The details of the CALLs seem to have appeared in Holland, via Paul Karis, who wrote an article on CALL A and CALL D for 99er Magazine. We have a printed booklet which seems to be a dump of the TI hard disk archive file "ARCHIVE.PRK.DOC.SUBRLST" and this has been used in putting together this (and preceding) articles. The archive document indicates also an intention to allow CALL FILES(0), but that was also not implemented.

The extra calls are CALL A (Accept); CALL D (Display); CALL G (Getput); CALL H (Header); CALL L (Load); CALL P (Prep); and CALL S (Save).

These allow you to set aside an area of VDP ram for data storage, move data to and from this area, and to and from the screen, and save and load data stored in the area to an external device in MEMORY IMAGE format. This is a very fast way of storing data, especially for cassette users.

We shall look first at the ACCEPT and DISPLAY subroutines as they relate to the screen input and output. Then we shall get into the juicy bit of reserving VDP ram and doing something with it.

CALL A - ACCEPT SUBPROGRAM.

There are four formats. The fourth we shall look at later, verifies input based on a predefined header.

i and ii :
CALL A(ROW,COLUMN,WIDTH,NUMRETVAR,NUMERIC,[LOW,HIGH]).

ROW AND COLUMN are screen positions; 1 to 24 for row, and 1 to 28 for column.

WIDTH is how many screen characters are to be entered. If this takes you past the screen edge, WIDTH is effectively reduced to the room available. You can of course actually enter a shorter input, as trailing spaces will be dropped.

NUMRETVAR must be a numeric variable. Its value is changed according to which key you use to trigger the input:
NUMRETVAR will be
1 if you use ENTER and the input is valid and non-null.
2 if you use ENTER but the field is blank.
3 if you press AID (FCTN[7])
4 if you press REDO (FCTN[8])
5 if you press PROC'D (FCTN[6])
6 if you press BEGIN (FCTN[5])
7 if you press BACK (FCTN[9])
- think about how you can use this function!!

NUMERIC is a numeric variable, into which your input is placed.

LOW and HIGH are optional, and can be used to specify a range of acceptable inputs; perhaps you want the user to enter a number between 56 and 121? Not quite the same as the VALIDATE command we are used to in Extended BASIC eh!

iii :
CALL A(ROW,COLUMN,WIDTH,NUMRETVAR,STRINGVAR)
is as above except that it is used to input a string variable. There is no range checking available.

Notes: Additional checking is possible using a fourth format which refers to a predefined header, which we shall cover soon.

Row and Column numbers are MOD(24) and MOD(28) respectively. It will not crash if you use a number over 24 or 28. If you try to use a negative or zero number, the default of 1 is used. Play around with this a while. It is not too difficult!

Even easier is:

CALL D - DISPLAY SUBPROGRAM.
CALL D(ROW, COLUMN, WIDTH, VALUE,,,,,,,,,,)

Where ROW and COLUMN are screen positions, MOD(24) and MOD(28) respectively, with zero or negative values read as 1.

WIDTH is interesting. If WIDTH is positive, then that number of characters will be replaced with spaces before the display is inserted. If the display is wider than the specified width, the display is truncated. If WIDTH is negative, the display will be inserted up to the specified length, but any excess length is not cleared. WIDTH is reduced to the distance to the edge of the screen if you use too high a number.

VALUE may be a number, string, or numeric or string variable.

You may use CALL D to display as many items as you can fit into your BASIC program line, specifying the parameters for each display. Try it.

SAMPLES:
```
100 CALL CLEAR
110 CALL D(1,1,28,"Using CALL D and CALL A")
120 CALL D(3,1,28,"PRK/STATS sub programs")
130 FOR T=1 TO 30
140 CALL D(5,1,3,T,5,5,2,-T,5,10,1,T,5,15,4,T/7)
150 NEXT T
160 CALL HCHAR(5,1,32,32)
170 CALL D(6,1,28,"Input a number from 6 to 9:")
180 CALL A(7,20,4,RTN,NBR,6,9)
190 CALL HCHAR(12,1,42,160)
200 CALL D(13,1,28,"DIVIDED BY 12 IS:")
210 CALL D(14,12,-5,NBR/12)
220 CALL D(9,1,28,"PROC'D for next section")
230 IF RTN=5 THEN 240 ELSE 180
240 CALL HCHAR(6,1,1,32,320)
250 CALL D(12,1,28,"PRESS:",13,2,28,"REDO or PROC'D")
260 CALL A(14,12,2,RTN,NUL$)
270 IF RTN=4 THEN 100
280 IF RTN=5 THEN 290 ELSE 260
290 CALL CLEAR
300 CALL D(12,1,28,"End of Demo")
310 END.
```

Now we shall dive into the mysteries of CALL L (load), CALL P (partition), CALL H (header), and CALL G (getput).

The best way to introduce these perhaps is with a working program, then in the next issue we can get down to a little more detail!

Program first, then discussion. This program will use an already created PRK data file, and in the TI BASIC environment will read that data and display it on the screen. It will demonstrate several of the calls in a fairly simple manner, so you can get used to the idea of using PRK data in a BASIC program! If you have a disk system, free up memory by typing:
```
CALL FILES(1)
NEW
```
before you load and run this program!

As this program occupies some VDP memory, you may find that a few PRK data files will be too long to load. Try deleting some records to make the data file fit. Remarks will be in lower case between the program lines! Before loading the program you must allocate an area of VDP memory for the data, by using the partition command, in command mode, thus:

```
CALL P(10900)
NEW
```

Now load the following program and run it:

```
Firstly, load the prk file from disk or cassette
100 CALL L("CS1",Y)
or call l("dsk1.filename") as you wish
now to check to see if the file is loaded
110 IF Y=0 THEN 500
no there was an error
else carry on as follows
how many fields in each record? let us find out
120 CALL H(1,5,0,F)
where f will return number of fields.
130 CALL CLEAR
140 CALL D(1,1,28,"# OF FIELDS:"STR$(F))
then we need to know how many records there are in the
data file, think of a record as a page:
150 CALL H(1,6,0,R)
160 CALL D(2,1,28,"# of RECORDS:"STR$(R))
now let us loop through each record and display it on
the screen:
170 FOR RCD=1 TO R
taking each field in turn
180 FOR FLD=1 TO F
but does the field contain a number or a string?  let
us find out:
190 CALL H(1,10,FLD,TYPE)
200 IF TYPE=1 THEN 240
as type is 0 data is a number:
get the data from field FLD of record RCD:
210 CALL G(1,RCD,FLD,Z,RD)
and print the result on screen:
220 CALL D(2+FLD,1,28,RD)
and jump over string section
230 GOTO 260
this is string section:
240 CALL G(1,RCD,FLD,Z,RD$)
250 CALL D(2+FLD,1,28,RD$)
that is one field dealt with, on to the next:
260 NEXT FLD
and now all fields in the record are on screen let us
give the record number and a pause
270 CALL D(23,18,10,"RECORD:"STR$(RCD))
280 CALL D(24,1,28,"PRESS ENTER     NEXT")
290 CALL A(24,28,1,RC,RV$)
and on to the next record
300 NEXT RCD
310 PRINT "NO MORE"
320 GOTO 320
500 CALL CLEAR
510 PRINT "LOAD ERROR"
520 GOTO 520
530 END
```

CALL P sets aside an area of memory for the data to be stored in, and must of course be sufficient to hold the data, but for BASIC use, we must also have enough room for our BASIC program! After using CALL FILES(1), disk users have a maximum of 12768 bytes of VDP RAM free to split between BASIC program and data area, while cassette users have 13820 bytes available (see how much memory a disk system eats up!). By using 10900 for our data, we have left even disk users with at least 1868 bytes for our little program!

CALL L just loads the data file into the partitioned area, the return variable following the device/file name is a 0 (zero) if an error occurs, for instance, the files is not on the disk, the data is too large for the partition, or other I/O errors. The device/filename may be a string variable if you wish, e.g. CALL L(A$,A)

CALL H deals with the "header" information, the database specification you create when using PRK/Stats: what sort of data is held in each field and what name have we given the field?

As used in the above example (there are many other uses!!!):

```
CALL H(1,5,0,F)
CALL H(1,6,0,R)
CALL H(1,10,FLD,TYPE)
```

The first digit (1) is a read instruction. We would use a 0 to write.

The fourth digit is a variable placed in or obtained from the header data.

The second digit (5, 6 or 10 above) is the header item number, from 1 to 14, where 5 is the number of fields per record, and 6 is the number of records.

10 is the type of field, returning to the fourth item, a numeric variable, a 1 for characters (string), a 2 for Integer, a 3 for decimal, and a 4 for scientific notation.

We shall deal with the other 11 header items, and with writing to the header, in a later article.

CALL G deals with the actual data we have stored, and as used above (again there are other uses.):

```
CALL G(1,RCD,FLD,Z,RD)
CALL G(1,RCD,FLD,Z,RD$)
```

Again the first item (1) is a read instruction, with 0 used to write.

The second and third items identify the record and field number that we are going to read/write, while the final item is the variable that the contents of that particular record/field will be placed in for us to use. Note that we must use a string variable for characters! That fourth item (Z) is a numeric variable, which can be used to indicate a blank field. It takes a value 0 if data is found, and 1 if data is missing, that is, not entered!

When using CALL G to WRITE, the fourth parameter as above is not used, you just go on to the value you are writing, for instance:

```
CALL G(0,RCD,FLD,"STRING INPUT")
```

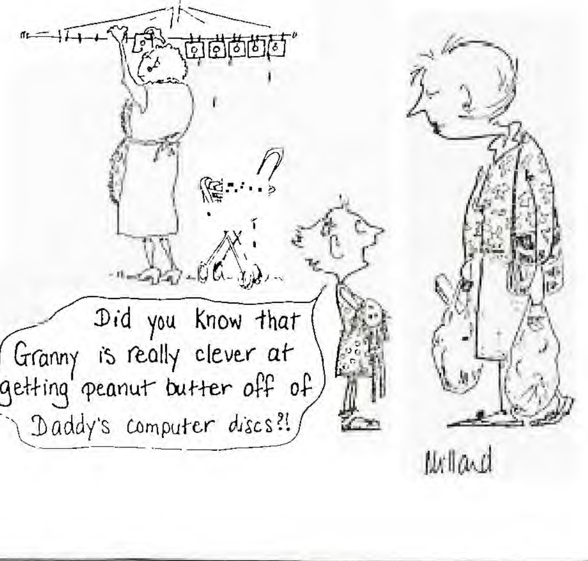
That fully covers CALL G, apart from a sample of how to write!

CALL S will save the data area to cassette/disk, very easily indeed:

CALL S(DEVICE$,VARIABLE) with variable again indicating success or failure!

That leaves us with a great deal to look at with CALL H, and also we need to look at writing data, and (perhaps!) maybe even creating a data file without using the PRK create file routine. Until later...

((Is anyone reading this please!!! Is this the right level/approach for you?)) o


Did you know that Granny is really clever at getting peanut butter off of Daddy's computer discs?!
Mrs. Mullard

Tips from the Tigercub #50

by Jim Peterson

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

Over 120 original programs in BASIC and Extended BASIC, available on cassette or disk, NOW REDUCED TO JUST $1 EACH!, plus $1.50 per order for cassette or disk and postage and handling. Minimum order of $10. Cassette programs will not be available after my present stock of blanks is exhausted. The Handy Dandy series, and Color Programming Tutor, are no longer available on cassette. Descriptive catalogs, while they last, $1, which is deductable from your first order.

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

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

NUTS & BOLTS DISKS

These are full disks of 100 or more utility subprograms in MERGE format, which you can merge into your own programs and use, almost like having another hundred CALLs available in Extended BASIC. Each is accompanied by printed documentation giving an example of the use of each. NUTS & BOLTS (No. 1) has 100 subprograms, a tutorial on using them, and 5 pages of documentation. NUTS & BOLTS No. 2 has 108 subprograms, 10 pages of documentation. NUTS & BOLTS #3 has 140 subprograms and 11 pages of documentation. NOW JUST $15 EACH, POSTPAID.

TIPS FROM THE TIGERCUB

These are full disks which contain the programs and routines from the Tips from the Tigercub newsletters, in ready-to-run program format, plus text files of tips and instructions. TIPS (Vol. 1) contains 50 original programs and files from Tips newsletters No. 1 through No. 14. TIPS VOL. 2 contains over 60 programs and files from Nos. 15 through 24. TIPS VOL. 3 has another 62 from Nos. 25 through 32. TIPS VOL. 4 has 48 more from issues No. 33 through 41. NOW JUST $10 EACH, POSTPAID.

```
******************************
*         NOW READY          *
* TIPS FROM TIGERCUB VOL. 5 *
*   Another 49 programs and *
* files from issues No. 42 *
* through 50. Also $10 ppd *
******************************
```

TIGERCUB CARE DISKS #1,#2,#3 and #4.

Full disks of text files (printer required). No. 1 contains the Tips news letters #42 through #45, etc. Nos. 2 and 3 have articles mostly on Extended BASIC programming. No. 4 contains Tips newsletters Nos. 46-52. These were prepared for user group newsletter editors but are available to anyone else for $5 each postpaid.

This educational program is a much expanded version of a routine I published before.

```
100 DIM M$(100)
110 GOTO 150
120 S,K,A$(),J,M$(),Y$,Z$,Z,X,ING$,A,AN$
130 CALL CLEAR :: CALL COLOR :: CALL SCREEN ::
    CALL CHAR :: CALL KEY :: CALL ING :: CALL HCHAR
140 !@P-
150 CALL CLEAR :: FOR S=0 TO 12 :: CALL COLOR(S,2,8)::
    NEXT S :: CALL SCREEN(5)::
    DISPLAY AT(3,1):"LEARNING TO ""ING"" IT V.1.1"
160 CALL CHAR(64,"3C4299A1A199423C")::
    DISPLAY AT(5,1):"@ Tigercub Software 1987 forfree
    distribution - no priceor copying fee to be charged
    "
170 CALL KEY(3,K,S)
180 A$(1)="No, if the word does not end in B, D, G, M,
    N, P, R or T you always just add ING"
190 A$(2)="No,if the last letter is not E and the
    next-to-last    letter is not a vowel, just add
    ING"
200 A$(3)="No, if the word has two    vowels just
    before the last letter, just add ING"
210 A$(4)="No, if a word ends in B, D, G, M, N, P, R or
    T with one vowel (but not two vowels!) just before
    it, you must    double the last letter and add
    ING"
220 A$(5)="No, if the word ends in IE, change the IE to
    Y and add  ING"
230 A$(6)="No, BE is an exception to  the rules,"
240 A$(7)="Some dictionaries give EYING but EYEING is
    better"
250 A$(8)="No, if a word ends in E (ex-cept BE and
    words  ending in IE,OE,UE AND YE) you must drop the
    E and add ING"
260 A$(9)="No, if the word ends in EE, or OE or UE,
    just add ING"
270 A$(10)="No, QUIP, QUIT and QUIZ are exceptions to
    the rule.    Double the last letter and  add ING."
280 FOR J=1 TO 100 :: READ M$(J):: NEXT J
290 FOR J=1 TO 100 :: Y$=Y$&CHR$(J):: NEXT J :: Z$=Y$
300 DISPLAY AT(3,1):"":"":"":" Type the word with the
       correct ING suffix"
310 RANDOMIZE :: Z=INT(RND*LEN(Z$)+1)::
    X=ASC(SEG$(Z$,Z,1))::
    Z$=SEG$(Z$,1,Z-1)&SEG$(Z$,Z+1,255) :: IF LEN(Z$)=0
    THEN Z$=Y$
320 CALL ING(M$(X),ING$,A)
330 DISPLAY AT(12,1):M$(X):: ACCEPT AT(12,15):AN$
340 CALL HCHAR(15,1,32,280):: DISPLAY AT(10,1):"" :: IF
    AN$=ING$ THEN DISPLAY AT(10,10):"CORRECT!" :: GOTO
    310
350 DISPLAY AT(15,1):A$(A):" ":"The word is ";ING$ ::
    GOTO 310
360 !@P+
370 DATA LODGE,BUY,HOPE,QUIP,TITHE, WISH,CUT,DRIVE,
    SEE,EYE,GO,CRY,TRY,AGREE,QUIT
380 !@P-
390 DATA BOIL,COOL,HURT,BUTT,CAGE,BE, ROVE,PITY,SAVE,
    COOL,RULE,MEASURE,TUNE,RAVE
400 DATA RUN,BEG,STOP,THINK,ERR,BORE, TEAR,BAR,CARE,
    BARE,BEAR,LET,QUIZ,HOOT,HEAT,COME
410 DATA DREAM,TAKE,FRY,CADDY,FLEE,HOE, SEW,TRIP,HOPE,
    RIG,DRAG,SUE,KNEE,BOO,BABY,NURSE,CRUISE
420 DATA LIE,TIE,DIE,BELIE,VIE,DODGE, LIVE,DRIVE,LOVE,
    LEAVE,HUM,HOP,BEG,BEGIN,BOMB,BOB
430 DATA ADD,AID,BAT,BOAT,PRAY,LAY, QUOTE,SNORE,STARE,
    HIRE,FIRE,LINE,CRY,SAY
440 DATA BOOGIE,RAGE,RATTLE,GRATE, LEAVE,STRIVE,DRAW,
    WRITE
450 !@P+
460 SUB ING(M$,ING$,A):: E$= SEG$(M$,LEN(M$),1)::
    F$=SEG$(M$,LEN(M$)-1,1):: A$="ING" ::
    C$="BDEGMNPRT" :: V$="AEIOU"
470 GOTO 500
480 C$,E$,ING$,M$,A$,A,V$,F$
490 !@P-
500 IF LEN(M$)=4 AND SEG$(M$,1,3)="QUI" THEN
    ING$=M$&E$&A$ :: A=10 :: SUBEXIT
510 IF POS(C$,E$,1)=0 THEN ING$=M$&A$ :: A=1 :: SUBEXIT
520 IF E$="E" THEN 550
530 IF POS(V$,F$,1)=0 THEN ING$=M$&A$ :: A=2 :: SUBEXIT
540 IF POS(V$,SEG$(M$,LEN(M$)-2,1),1)<>0 THEN
    ING$=M$&A$ :: A=3 :: SUBEXIT ELSE ING$ =M$&E$&A$ ::
    A=4 :: SUBEXIT
550 IF F$="I" THEN ING$=SEG$(M$,1,LEN(M$)-2)&"YING" ::
    A=5 :: SUBEXIT ELSE IF F$="E" OR F$="O" OR F$="U"
    THEN ING$=M$&A$ :: A=9 :: SUBEXIT
560 IF M$="BE" THEN ING$="BEING" :: A=6 :: SUBEXIT
570 IF M$="EYE" THEN ING$="EYEING" :: A=7 :: SUBEXIT
580 ING$=SEG$(M$,1,LEN(M$)-1)&A$ :: A=8
590 !@P+
600 SUBEND
```

I still have a sort of an old-fashioned idea that the computer can be a useful educational tool -

```
100 CALL CLEAR :: FOR SET=0 TO 12 ::
    CALL COLOR(SET,2,8) :: NEXT SET :: CALL SCREEN(5)::
    DISPLAY AT(3,6):"NOUN TOADJECTIVE" ::
    CALL KEY(3,K,S)
110 CALL CHAR(64,"3C4299A1A199423C")::
    DISPLAY AT(5,5):"@ Tigercub Software":"":" For free
    distribution - no  price or copying fee to be
    charged."
120 DISPLAY AT(12,1):" One moment...loading memory"
130 DATA ROGUE,ROGUISH,HOG,HOGGISH,PIG, PIGGISH,SWINE,
    SWINISH,THIEF,THIEVISH, KNAVE,KNAVISH,BRUTE,BRUTISH
    or BRUTAL
140 !@P-
150 DATA FAME,FAMOUS,TUMULT,TUMULTUOUS, RIOT,RIOTOUS,
    SCANDAL,SCANDALOUS, MOUNTAIN,MOUNTAINOUS,ODOR,
    ODOROUS or ODORIFEROUS
160 DATA CAVERN,CAVERNOUS,VILLAIN, VILLAINOUS,DANGER,
    DANGEROUS,PERIL, PERILOUS,ADVANTAGE,ADVANTAGEOUS
170 DATA BARB,BARBED,FORK,FORKED, BORDER,BORDERED,
    WHEEL,WHEELED,HUNGER, HUNGRY,ANGER,ANGRY
180 DATA PARLIAMENT,PARLIAMENTARY, PLANET,PLANETARY,
    LEGISLATURE, LEGISLATIVE,PARISH,PAROCHIAL
190 DATA CONGRESS,CONGRESSIONAL, ELEPHANT,ELEPHANTINE,
    FANTASY, FANTASTIC,BULL,BULLISH
200 DATA GIRL,GIRLISH,BOY,BOYISH,BABY, BABYISH,AMATEUR,
    AMATEURISH,FEVER, FEVERISH,DEVIL,DEVILISH,FOOL,
    FOOLISH
210 DATA OAF,OAFISH,SHEEP,SHEEPISH, CHILD,CHILDISH or
    CHILDLIKE,VIRTUE, VIRTUOUS,PRIDE,PROUD or PRIDEFUL
220 DATA HATE,HATEFUL,DOUBT,DOUBTFUL, THOUGHT,
    THOUGHTFUL,SHAME,SHAMEFUL, FEAR,FEARFUL,SORROW,
    SORROWFUL
230 DATA WISH,WISHFUL,PEACE,PEACEFUL, EVENT,EVENTFUL,
    TRUTH,TRUTHFUL,SKILL, SKILLFUL,MAN,MANLY
240 DATA WOMAN,WOMANLY,FATHER,FATHERLY, MOTHER,
    MOTHERLY,BROTHER,BROTHERLY, SISTER,SISTERLY
250 DATA NIGHT,NIGHTLY,HOUR,HOURLY, MONTH,MONTHLY,
    ORDER,ORDERLY,SERIES, SERIAL
260 DATA TIME,TIMELY,GRAVEL,GRAVELLY, FRIEND,FRIENDLY,
    WOOL,WOOLLY,YEAR, YEARLY,SOUTH,SOUTHERN or
    SOUTHERLY
270 DATA NORTH,NORTHERN or NORTHERLY, WEST,WESTERN or
    WESTERLY,EAST,EASTERN or EASTERLY
280 DATA CHARITY,CHARITABLE,TERROR, TERRIFIED or
    TERRIBLE,HORROR,HORRIFIED or HORRIBLE or HORRIFIC
290 DATA RAG,RAGGED,MILITARY, MILITARISTIC,ART,
    ARTISTIC,CAT,CATTY, DOG,DOGGY,FOG,FOGGY,SUN,SUNNY
300 DATA BAG,BAGGY,LEG,LEGGY,BOG,BOGGY, STUB,STUBBY,
    FUN,FUNNY,FUR,FURRY,GUM, GUMMY,AVARICE,AVARICIOUS
310 DATA CLOUD,CLOUDY,RAIN,RAINY, FLOWER,FLOWERY or
    FLORAL,GREED,GREEDY, THIRST,THIRSTY,AIR,AIRY,BUSH,
    BUSHY, FISH,FISHY
320 DATA SOUP,SOUPY,BLOOD,BLOODY,FOAM, FOAMY,BEAD,
    BEADY,SWAMP,SWAMPY,SILVER, SILVERY,COPPER,COPPERY,
    DUST,DUSTY
330 DATA DIRT,DIRTY,GUILT,GUILTY,SALT, SALTY,GRAIN,
    GRAINY,OIL,OILY,TRICK, TRICKY,HILL,HILLY,
    ROCK,ROCKY
340 DATA SAND,SANDY,SOAP,SOAPY,SUDS, SUDSY,SILK,SILKY,
    WOOD,WOODY,MODESTY, MODEST,PIETY,PIOUS,DAY, DAILY
350 DATA TREE,TREELIKE,TOY,TOYLIKE, FINGER,FINGERLIKE,
    SWAN,SWANLIKE,WAR, WARLIKE,DISH,DISHLIKE,PLATE,
    PLATELIKE
360 DATA SPOON,SPOONLIKE,BIRD,BIRDLIKE, SNAKE,SNAKY,
    WIRE,WIRY,BONE,BONY,SMOKE, SMOKY,FLAKE,FLAKY
370 DATA NOISE,NOISY,BRINE,BRINY,TASTE, TASTY,STONE,
    STONY,WAVE,WAVY,GORE,GORY, PASTE,PASTY,BUBBLE,
    BUBBLY
380 DATA LABOR,LABORIOUS,ORNAMENT, ORNAMENTAL,
    GOVERNMENT, GOVERNMENTAL, CONTINENT,CONTINENTAL,
    MUSIC,MUSICAL
390 DATA MAGIC,MAGICAL,TOPIC,TOPICAL, SENSATION,
    SENSATIONAL,LOGIC,LOGICAL, ALARM,ALARMING,
    ARTERY,ARTERIAL
400 DATA GOLD,GOLDEN,EARTH,EARTHEN, GLAMOUR,
    GLAMOURIZED,DEPUTY,DEPUTIZED,
    ENERGY,ENERGIZED,PART, PARTIAL,FIRE, FIERY
410 DATA ANGEL,ANGELIC,CHERUB,CHERUBIC, BURDEN,
    BURDENSOME,TROUBLE,TROUBLESOME, BEAST,BESTIAL
420 DATA HISTORY,HISTORICAL,GEOGRAPHY, GEOGRAPHICAL,
    BOTANY,BOTANICAL,BIOLOGY, BIOLOGICAL,LITURGY,
    LITURGICAL
430 !@P+
440 DIM A$(175),B$(175):: FOR J=1 TO 174 :: READ
    A$(J),B$(J):: Z$=Z$&CHR$(J):: NEXT J :: Y$=Z$ ::
    RANDOMIZE
450 DISPLAY AT(7,1):"":"Type the adjective form of
    -":""
460 X=INT(RND*LEN(Y$)+1):: Y=ASC(SEG$(Y$,X,1))::
    Y$=SEG$(Y$,1,X-1)&SEG$(Y$,X+1,255):: IF LEN(Y$)=0
    THEN Y$=Z$
470 DISPLAY AT(12,1):A$(Y):: ACCEPT AT(12,14):Q$ :: IF
    POS(B$(Y),Q$,1)=0 THEN 490
480 DISPLAY AT(18,1):"":"" :: FOR D=1 TO 100 :: NEXT D
    :: DISPLAY AT(18,1):" That is the word in my
    memory banks.":"" :: GOTO 460
490 DISPLAY AT(18,1):" The adjective in my memory banks
    is ";B$(Y):: GOTO 460
```

When one program is run from from another by RUN DSK.., the screen is not cleared, sprites are not deleted, and screen color, character definitions and sprite magnification are not returned to the default values. This can cause some strange results, which can be prevented by CALLing CLEARALL just before the RUN.

```
1000 SUB CLEARALL :: CALL CLEAR ::
    CALL DELSPRITE(ALL):: CALL SCREEN(8):: CALL CHARSET
    :: CALL MAGNIFY(1)
1001 FOR CH=65 TO 90 :: CALL CHARPAT(CH,CH$) ::
    CALL CHAR(CH+32,
    "OO"&SEG$(CH$,1,12)&SEG$(CH$,15,2)) :: NEXT CH
1002 CALL CHAR(96,"000201008",123, "0018202040202018",
    124, "001010100010101000300808040808300000205408")
1003 FOR CH=127 TO 143 :: CALL CHAR(CH,"0"):: NEXT CH
    :: SUBEND
```

The routine in line 1001 can be used, by deleting the +32 if necessary, to modify some of the character sets on my Nuts & Bolts disks.

From an idea in a program by Ed Machonis, here is an improvement to my 28-Column Converter published in Tips #18. After line 160, insert

```
165 DISPLAY AT(20,1):"Tab setting? 1" ::
    ACCEPT AT(20,14)SIZE(-2)BEEP:T
```

And change line 290 to -

```
290 PRINT #2:TAB(T);L$ :: S=S+28 :: GOTO 410
```

MEMORY FULL! - Jim Peterson

continued from page 20

checks to see which key is hit using the IFTHEN or IFELSE routines. If the QUITKY is hit, the program turns on the interrupts and jumps to the title screen, ending the program. If the REDO key (FCTN[8]) is hit, the routine is repeated. If some other key is hit, the program simply re-scans the keyboard until a valid key is hit.

FOR THE ADVANCED PROGRAMMER

As you can see, it is relatively easy to write assembly programs using the PULSAR routines. PULSAR is quick and efficient, but does need some improvements. This is where the advanced assembly programmer comes in.

Routines still needed include disk I/O, string manipulation (SEG$, POS, ASC, CHR$, LEN, etc.), plus access to TI's Bit Map, Multi-color graphics, Sprites, speech and sound capabilities.

The next step in PULSAR development should be the incorporation of the BLWP command in place of the BL. This would allow much easier mix of PULSAR and "pure" assembly language commands.

If you do get interested and begin to write your own routines, be sure to let me know.

NOTE: This article originally appeared in the April Edition of the Ohio New Horizons User Group Newsletter. It is reprinted by permission. Anyone who would like a copy of the PULSAR source and object code should send an initalized SSSD disk along with a self-addressed/stamped return evelope and $5 to:

PULSAR UTILITIES
c/o SUBFILE99
POB 533
[illegible] Green, Ohio 43402

The Forth Column

Forth Forum <6>

by George L Smyth

This month I will give a quick go-over of Forth's conditional IF..THEN construct. The similarity in usage of this word set to BASIC's namesake is close, with the exception that we use this word "Forth style". A comparison of the uses of these words, as defined by the two languages, will give the user an idea of how Forth's implementation differs from BASIC.

Also presented this month is a word which will allow user specified precision of integer division. Although its useability is questionable, it is fun to divide two numbers and take the division out 1000 places beyond the decimal point. So, to begin.

Getting iffy

The Forth equivalent of BASIC's IF..THEN statement is also IF..THEN, although the construct is somewhat different. Before we examine this form, let us look at a few words which will often help us set up this word pair.

= (n1 n2 --- f)

A while back I stated that the equal sign in BASIC did not have the same function in Forth. The Forth language uses '=' as a comparison operator. A comparison operator is a word which removes the two numbers from the top of the stack, compares them in some way, and reports the status of the requested comparison by placing a flag on the stack. A flag is represented by the system as either a '1' or a '0', meaning true or false, respectively. The equals sign returns a flag to indicate whether or not the two numbers on the top of the stack are equal or not, a '1' if they are and a '0' if they are not.

> (n1 n2 --- f)
< (n1 n2 --- f)

The "greater than" and "less than" signs are also comparison operators which determine the relationship of the next to the top of the stack value to the value at the top of the stack value. When using the '>' operator, if the second to the top of the stack value is larger than the top of the stack value, a true flag is left on the stack. Conversely, when the '<' word is used, this situation will result in a false flag being returned to the stack.

The reason we want to place a flag (sometimes called a Boolean flag) on the stack is to set up a situation whereby we may be able to utilize the IF..THEN structure. The word IF examines the stack to determine if a false flag (0) or a true flag (<>0) has been left. If it finds a true flag, execution continues until the word THEN ends the routine. A false flag will force the system to ignore the routine between the IF and THEN words.

The word ENDIF is a synonym for THEN and the two may be interchanged either way without affecting program execution. I have seen both THEN and ENDIF used in the same word when nesting conditionals to allow for additional clarity. I personally prefer THEN because it relates to the functional purpose of BASIC's namesake, but you should decide for yourself which one you like.

Just as BASIC employs an IF..THEN..ELSE structure, Forth also has a similar operation, IF..ELSE..THEN. After IF looks at the stack, it will execute either the words between IF and ELSE if it finds a true flag, or the words between ELSE and THEN if it finds a false flag. Let us write a word to demonstrate this.

```
: 5COMP 5 = IF ." It is equal" ELSE ." It is not
equal" THEN ;
```

This word named 5COMP places the number 5 on the stack and compares it with the number that had previously been placed on the stack. If the number '5' was on the stack before 5COMP was executed, the display will show "It is equal". If anything other than a 5 was on the stack, "It is not equal" will be printed. Try it:

```
5 5COMP It is equal
13 5COMP It is not equal
```

Because I figured everyone was sick and tired of the "Guess the Number" game in BASIC, I decided to write it in Forth. Actually, this will give us a bit of experience in writing programs correctly. We need a couple of words we have not gone over to complete it, so I made it as simple as possible.

```
SCR #45
 0 ( Guess the Number )
 1 BASE->R DECIMAL
 2 0 VARIABLE NUMBR RANDOMIZE
 3 : THE_NUMBER 100 RND NUMBR ! ;
 4 : WRONG ." Too " NUMBR  >
 5     IF ." Large" ELSE
 6        ." Small" THEN ;
 7 : GUESS DUP NUMBR  ROT =
 8     IF ." Correct" DROP
 9        ELSE WRONG THEN ;
10
11
12 R->BASE
13
14
15
```

To "play" the game, enter "THE_NUMBER" to have the program pick number between 0 and 99 and place it in the variable 'NUMBR' (the word 'NUMBER' is a word already used in Forth). Now you can make your guess by entering the number followed by the word "GUESS", for example, '50 GUESS'. If you are correct, "Correct" will be returned, if not, you will be told if your guess was too large or too small.

Several things should be noted concerning this program that I consider good programming practice, although I could get some arguments . First off, note the words BASE->R and R->BASE. Before we enter this program it is quite possible that we could be in the HEX number system, or any other number system for that matter. If we want to retain that numbering system after we finish loading that screen, we would like to save the base and return it to the system after entering this program. These words do just that. BASE->R takes the base and saves it on the return stack (something else we have yet to talk about (yes there are two stacks)). R->BASE returns this number back to the system after loading in this screen. This way we can be sure that we are in base 10 (decimal) while loading the application, and we will return to whatever base we were working with previously.

Also note the indentations inherent in the words. This is a point of contention. I have argued against those who feel that the programmer should fill the screen with as much information as possible. It is true that the application may load a bit faster, but the tradeoff is decreased clarity. The function of the word "WRONG" is easy to see partly because of the indentations. Unlike BASIC, program execution is not changed at all by this. The ability to revise the program is, of course, also enhanced.

Another thing I am sure to get flak about is the fact that I never nest IF..THEN statements. Note that the word "WRONG" could easily fit into the word "GUESS" without any problems. There is nothing wrong with ending word with "THEN THEN", except for the fact that I feel that clarity and easy revision suffer somewhat. If you are like me in that you are never satisfied with a program and are constantly revising it, you may wish to consider this.

Last but not least, please note that each word is short. Yes, it is simple program and I would have trouble being verbose here, but the fact remains that short words constitute good programming practice. All professional Forth programmers I have talked with have told me that their words typically are 15-20 words in length maximum! This may sound a bit restrictive, but it forces the programmer to use the concepts Forth was built around. If you were to look at the programs submitted by J Volk, you would see words containing over 100 words within them. How does one go about modifying them? I did not bother trying. I was glad to see them at time when few Forth programs were available for our computer, but was frustrated when I tried to dissect them. We have plenty of room on our screens, . . [illegible] them?

I will go over some of these elements in later articles when I present a catalog program I wrote in Forth which has several advantages of being able to "see" things hidden from BASIC catalog programs.

The Forum

Thought for the month:
"There is always one more bug!"

A while back I lamented the fact that using floating point routines were an extension which required learning new words, which I still have not done for the most part. Of course, speed is enhanced when the computer does not have to worry about where the decimal point is, but I still wondered if there was some way I could divide two numbers and get an answer with value extending beyond the decimal point. Well, needless to say, there is. To all mathematicians, I do realize the rules which relate to the significance of precision, but we are just having fun here.

The program listed will divide two integers (whole numbers) to the precision you wish to indicate. If you want to use a number which is not an integer, merely multiply the numbers by any factor of 10 until they both are, i.e. 123/4.56 = 12300/456. When I was considering how to write this word, I figured that the best way to go would be to do it the way I normally divide two numbers using longhand division. The main falling point with this routine is that the only thing that you can do with this answer is to display it. As far as the system is concerned, this is merely a string of numbers.

```
SCR #46
 0 ( Multi-precision Pseudo-non-modulus )
 1 : X/MOD ( Dividend Divisor Precision ---  )
 2         ROT >R SWAP R> OVER
 3         /MOD . ." . " ROT
 4             0 DO 10 * OVER
 5               /MOD . LOOP
 6         DROP  DROP  ;
 7
 8
 9
10
11
12
13
14
15
```

Example:

```
100 13 5 X/MOD 7 . 6 9 2 3 0
100 51 23 X/MOD 1 . 9 6 0 7 8 4 3 1 3 7 2 5 4 9 0 1 9
    6 0 7 8 4 3
```

Have fun with this and hopefully I will have some more stuff next month.

George L. Smyth
3017 Sylvan Dr.
Falls Church, VA, 22042
(703)533-8710

Using TI-Keys

by Wes Johnson, USA

TI-Keys is a program which allows the user to define 36 keys so that when they are typed as control keys, they will display up to 31 characters of text or code. The program is menu driven, disables the quit key, and changes the cursor shape when it is active. Other features include saving the user defined keys to disk, editing the keys, and the ablilty to turn 'off' the program so that true control characters can be typed.

LOADING INSTRUCTIONS

TI-Keys is loadable in two different ways. It is saved in Extended BASIC program format and is named LOAD. RUN this program, and TI-Keys will load quickly. The disadvantage to this method is it will erase any BASIC program already in memory.

The other format is the standard CALL LOAD format. Type CALL INIT :: CALL LOAD("DSKx.MAC"):: CALL LINK("MACRO"). The disadvantage to this method is the time involved loading the program.

USING TI-Keys

Once the program is loaded, BASIC's cursor will be a hollow box. This indicates TI-Keys is loaded and functional. Now press CTRL, and you will notice that the cursor is not blinking. The program is waiting for a key to be pressed. Now press the 'A' key while holding down CTRL. ACCEPT will be printed on the screen. Now release CTRL, and BASIC's cursor will begin blinking again. Make-Keys works with all letter keys A - Z, and the number keys 0 - 9. Any key can be redefined by the user, and saved to disk at any time.

By pressing CTRL[=], the menu will appear. The program options are listed on the screen as follows: 1 to EDIT, 2 to SAVE, 3 to LOAD, 4 to TURN OFF KEYS, 5 to RETURN TO BASIC.

1 EDIT - The program will ask 'KEY TO CHANGE?'. Press the key you wish to change, and the 'PRESENT VALUE' of the key will be displayed. Now TI-Keys asks 'CHANGE TO?'. Simply type in the string as it will apear in BASIC, and press enter.

2 SAVE - The program will ask 'SAVE FILENAME'. Type in any valid filename except CS1. If an error occurs, FILE ERROR will be displayed. Press a key to get back to the menu. If no error occurs, the menu will be displayed as soon as the file is saved.

3 LOAD - The instructions for load are the same as option.

4 TURN OFF KEYS - Will turn off TI-Keys so that true control characters can be typed, or so that other assembly programs can be loaded with out lock up.

WARNING If CALL INIT is performed, and another assembly program is loaded, the computer may lock up. This is prevented by turning TI-Keys OFF, and then typing CALL INIT, and loading the other program.

5 RETURN TO BASIC - Does just what it says. When you are finished with the menu, press 5 to get back into BASIC.

PREDEFINED KEYS

When the program is loaded, the keys have the following text strings stored in them.

A - ACCEPT	S - SAVE "DSK
B - BEEP	T - TAB(
C - CALL	U - U
D - DELETE "DSK	V - VCHAR(
E - END	W - CALL INIT
F - FOR	X - CALL LOAD("DSK
G - GOSUB	Y - CALL LOAD(-
H - HCHAR(	Z - CALL LINK("
I - IF	1 - RUN
J - JOYST(	2 - \
K - KEY(	3 - |
L - LINPUT	4 - |
M - MERGE "DSK	5 - \NO PREDEFINED
N - NEXT	6 - / VALUE
O - OPEN	7 - |
P - PRINT	8 - |
Q - Q	9 - |
R - RUN "DSK	0 - /

This is a "Fairware Program". Try it! If you like it please send the author your $10.00 thanks. I am a high school student trying to upgrade to DSDD and other nice things! Thanks!

Wes Johnston
404 Furman Lane
Ladson, SC 29406
United States of America

I'm the last person to question a chap's credibility but let's face it — he's not being sued by Atari.

Braille'n Speak

by Irwin Hott, USA

Braille'n Speak: a new computer for the blind.

I am actually beginning to write this article while riding a COTA bus downtown. I am using the Braille'n Speak. It is about 8 inches long 4 inches wide and 1.5 inches high (20cm by 10cm by 4cm in metric, ED). It weighs less than 1 lb (.5 kgm). The small size is made possible by the use of a braille keyboard and a speech synthesizer for the "display". The Braille'n Speak has rechargable batteries as well as an RS232 port.

The braille cell is made up of 6 dots 1,2,3 from the top to bottom on the left, 4,5,6 on the right. The unit uses 7 keys (1 for each dot plus the spacebar). You simply press dots simultaneously to get the desired character. Dot 1 is A and 2 is B, dots 1 and 4 are C etc..

I do not propose to go into the intricate patterns of Braille here except to say that there are different "grades" involving the number of contractions (abbreviations) used e.g. nec for necessary, AL for also and words such as WITH written as dots 2,3,4,5,6.

I am writing in grade 1 braille now. There are no contractions and I am writing in lower case most of the time. I can switch to upper case by hitting a U chord. That means pressing U and the space bar at the same time. That will give me the next character in upper case. If I hit U chord twice, upper case lock will be on. Most of the commands such as file. Cursor and parameter are made by pressing a key combination with the spacebar.

The Braille'n Speak has about 200K of RAM. About 180 of that may be used for file storage. Files may be as short as 1 page (4096 characters) or up to 45 pages. The maximum number of files allowed is 30. Right now I have 6 files open in Braille'n Speak with 40 pages of memory remaining. I have a 2 page file for this article; notes from Lima; a phone list; a BBS list; a help file which is always resident and clipboard. Clipboard is a 1 page file that is used to house deleted material as well as data copied from one file to another. I just exited "art" and looked at clipboard. It contained a sentence I had deleted from this article. I could have deleted a line from this file and put it into another file. It is very easy to move from one file to another. When I do that I return to the exact place where I left the file. Many features such as these make the unit a joy to use.

There are several word-processing functions built in. Right now I have key echo turned on. However my brailing is faster and I am ahead of speech most of the time. There is a backspace command which is destructive. I can move through text, a paragraph, line, word or character at a time in either direction. If I do not understand a character such as p or t there is a phonetic alphabet built-in. I can insert up to 255 characters from the keyboard. If I insert from another file I can add up to 4069 characters in one move. I can also delete anywhere from one character to the entire contents of the file. I can set a "mark" in text and delete to that "mark" in either direction. One of the minor drawbacks is in Replace String. I can overwrite a character or find a string but I cannot replace all occurrences of a string. It is very easy to transfer material to and from Braille'n Speak. The RS232 port is controlled by software commands. I can set the Baud rate from 75 to 19,200; set parity, duplex handshaking and stop bits. I can transmit complete text, text to "mark" a character, line or paragraph. If I want to receive text, all I have to do is open a file, set the parameters in the RS232 port and turn on the RS232 port. All incoming material will be stored in the open file.

I can listen to the material as it comes in or just let it build up in RAM. I frequently dump files from the TI99/4A so I can listen to them in the Braille'n Speak. It is much easier to read text here because I can skip around in the file and carry the machine around with me. As an example I dumped a series of messages from HUG TI BBS in Houston about using 3.5 inch disk drives. I thought it would make an interesting file on Spirit of '99, so I edited it in the Braille'n Speak This took about 10 minutes to edit out all of the extraneous information. On the TI99/4A without being able to use TI-WRITER or the equivalent it would have taken at least 45 minutes. I was able to quickly search for key words to delete, such as message numbers. I entered a note at the beginning of the file, merged the new file description into the old description file dumped both from the Braille'n Speak to the TI99/4A and I was ready to go. I can format the text I am sending from the Braille'n Speak. It can be formatted as to page length, line length, left margin and top margin. There is no way in writing text (such as this article) to specify a line length. I will format the file when I send it to the TI99/4A. It would be nice if I could set a line length and have a warning if I was approaching the end of the line. However that is a relatively minor drawback.

The Braille'n Speak has a clock built-in. The current time is 12:16. There is a calendar, a timer and a four function calculator. I can "paste" answers to calculations into a file such as I did with the time above.

Now that I have the Braille'n Speak I wonder how I ever got along without it. The program (using a 512k EPROM) has been carefully written to make it as easy to use the device as possible. Much careful thought has gone into it. Not that it was easy to learn. I have not counted but I suppose there are at least 50 new commands I had to learn. For the first couple of days I wondered if I was ever going to master it. After that it started to get much easier. There were also some bugs in earlier versions of the EPROM. I was one of about half a dozen people who helped test some of the updates. That was for the most part a lot of fun. However it was not without dangerous moments. A couple of times I erased memory! Once through my own carelessness, the other time an error in the program. Fortunately, I had backup copies of most files on the TI99/4A. I have not found any errors in this new version of the EPROM.

I am not really sure I can explain how nice it is to have a device such as this. The possibilities of use are just about endless. It can be used for phone messages, receipts, editing programs and so much more. The cost is reasonable at $US895.00, which for a high technology low production device is fairly unusual. As an example the first talking calculator cost $495. Now for the first time blind people have a computer, at least, on a par with those used by their sighted counterparts. Previously, portable lap-top computers with speech cost at least $2000. This put it out of the price range for many individuals. It is not absolutely necessary to interface Braille'n Speak with another computer. There is a tape interface device available as an option. It works through the RS232 port.

I hope this gives you a little idea of just how I use the Braille'n Speak.

Contact-: BLAZIE ENGINEERING
2818 COLLEGE VIEW DRIVE
CHURCHVILLE, MD 21028
UNITED STATES OF AMERICA.
0011 1 301 879-5504

I would also be glad to correspond with anyone who would be interested.

MY ADDRESS-:
Irwin Hott
1540 Northbridge Road
Columbus, OH 43224
0011 1 614 263-5319

o

****
WANT TO SELL
****

I have a 34cm (14") Colour TV with NTSC (American Colour System) implant built in. This is in good full working condition, and with that NTSC implant you can use it as a Monitor for any American models of the TI99/4A or other brand computers. I wish to sell it for only $300

If you wish to purchase this PAL/NTSC TV/MONITOR, for only $300, Please call me on (02)550 0014 or at work on (02)570 6388

Bi 4 now, Shane Anderson

continued from page 1

I finally made it to a Sydney meeting last month and had a talk to a few people. Ross Mudie told me about how busy he was at work at the moment trying to solve the rather strange problems of the FAX world. This has meant that he has had little time for his Club activities other than ensuring the smooth running of the BBS, which he does so well. This has meant that his contributions to the magazine have not been as frequent nor as long as the standard he set earlier in the year. However I notice that he had time for an article this month for which we should all be grateful. I will say thanks for everyone Ross, and hope that work settles down to a dull roar soon.

****

Chris Buttner also had a chat which I gained the impression was to put me in the picture as to the current state of the group (what are we now? a club, a group, a company, a rabble?) as he saw it from the top. I did not take notes and as I am writing this some 2 weeks after the event my memory is a bit sketchy about all that he said. He said that he would be on a course in Canberra for the next 2 or 3 months and would miss some meetings. He said that the finances of the group were a bit uncertain until a stocktake of the shop was complete, but the board was expecting that they would have to use some of the money currently in investments (assets) for the general running of the group. I said I felt that was a better use for that money, than leaving it in the bank. He said that the board had not been able to get together with the Hunter Valley committee in time to organise the meeting set down for October 1, and that there was a feeling that a date in the middle of the school holidays was not very suitable anyway. That is all I can remember of our conversation.

****

If there are any TI99ers with an interest in amateur radio, then Graeme Virtue of 9 Minyon Street, Brunswick Heads, NSW 2483, would like to hear from you.

****

I had a few words with Les Andrews at the TIsHUG meeting. He was very disappointed that the proposed get together with the Hunter Valley club was not to be held. It seems that the committees on both sides did not pursue the event with enough enthusiasm. Both in the TND and in the Hunter Valley 99ers user group newsletter the event had been announced many months before hand so that even Brisbane picked it up and suggested that some of their members might make the trip to Newcastle. That would have made the long weekend a good choice, but the opportunity has been missed. Les was disappointed as he had suggested the idea on the way back from the TI-Faire. He also wrote a few letters to try and keep the idea alive but all too late. Perhaps the amusement day will turn out to be a good way of achieving the original objective, so I would urge all those who can manage it to contact Albert and go along.

****

The review of the newsletters this month comes from Lou Amadio.

TI-SIG newsletter, San Diego, May 1988. Tandy have announced a write/read CD disk which can be used in the same way as a floppy disk. Storage is expected to be >100Mbyte and will be a boon for animated graphics applications. John Johnson announced a direct connect disk drive. SpadXIII runs faster if 32K is installed internally. Waldo Hamilton writes on the proper way to terminate cables, diagrams included. June 1988 has a plea for information on using Extended BASIC II with Horizon RAMdisks, another DV80 word counter (includes hyphens), more on (video) cable termination. July 1988 mentions a PACMAN type game for 2 joysticks, mention of a new graphics programming language from Adelaide (SA), a beginner's BASIC tutorial, TI-Writer tip: R tab should be 1 higher than Formatter .RM, an article on how to build an Atari joystick adapter. August 1988 has details of the TI-FEST West in February 1989, an article on ribbon cable connectors, CALL LOAD to disable disk drives, procedure to load very large Extended BASIC programs.

ROM Newsletter, Fountain Valley CA, July 1988. Adrian Robinson on Assembler language windowing, Earl Raguse writes about an improved Forth editor and an Extended BASIC 28 column program lister, N. Armstrong on uploading DV80 files, Jim Swedlow gives tips on using Funnelweb V4.1, Telco V2.1 and acceptable charararacters for file names.

LA 99ers TopIcs, Los Angeles, June 1988. Tom Freeman writes on fast loading a module into the 9640, Earl Raguse on beginner's Forth #2, C. DeMarti on how to make your printer print in different languages and an EZ BASIC tutorial, Bill Gaskill on PRbase Tips (V2.1 works on the 9640), Danny Nelson writes on how to use a modem. May 1988 has an in depth article on disk controllers by Jerry Coffey, an Extended BASIC program to create character highlighting, Curt Borders on load, hold and reset switches for the TI99/4A, C. DeMarti on spicing up CALL KEY routines, use of 3.5" drives more on EZ BASIC and another column lister program. Page 13 has a table on TI-Artist and Enhancement Functions, Dr Fudge reviews Funnelweb, Earl Raguse starts a Beginning Forth series, Steve Mehr on a new TI99/4A database called MICROdex V1.1, review of Video Chess Module, a Marketplace page full of interesting software and lastly a diagram on a PEB speech interface.

Northern NJ 99ers, New Jersey, June 1988. Page 2 has a BASIC address labeller for mailing labels, Jim Swedlow writes on how to send for fairware, Tips from the Tigercub #52, page 10 has an Extended BASIC program to list in 28 columns, a 3D title program, an unusual screen clearing routine as well as a number of other useful hints and a stress diet all from C. DeMarti.

Two newsletters from CIM 99, Montreal - still looking for a translator.

The Tacoma Informer, August 1988: Announcement of "ASGARD NEWS" magazine for the TI community.

The PUG Peripheral, Pittsburgh, August 1988: A review of TI-Base by Gene Kelly, TI-Writer "MOVE" command by Stan Katzman, RAMdisks Part VII by John Willforth is on the Rave 9 "MX01" PEB card, Lutz Winkler writes a new series of introductory Forth, Frank Zic has a "Tips For Beginners" column on The Printers Apprentice, Mickey Schmitt writes on getting the most from your cassette system, page 15 has a CALL LOAD to disable FCTN[4] in Extended BASIC, Tips From The Tigercub #45, page 19 has a tip on adding extra graphics to Certificate 99, page 20 Charles Good writes on the features of Funnelweb V4.1 and describes it as the most significant software ever for the TI99/4A.

Sacramento 99er Users Group, August 1988. Page 2 reviews KARATE CHALLENGE from Boundless Systems Software rating it 4 out of 5, Mike Morrow writes on "BASIC to FORTAN", and lastly, page 5 has an article on "Death of a Computer.

Australian Newsletters

CHUG-A-LUG, Canberra, July 1988. Page 5 has an article on "LEGENDS" which is a role playing adventure game, Jack Sugrue writes about PLUS! word processing utility package, page 7 has an article on graphics compatability in TI99/4A software, page 12 has a review of the Brisbane TI-Faire and declared it a great success, Asgard News available from Garry Christensen (Brisbane TI99/4A Users) at $12 for 4 newsletters, page 15 shows how to connect double joysticks to the TI99/4A, page 17 shows a modification to "release" alpha lock for games, page 18 on a double console reset switch which allows modules to be inserted without causing a reset.

TI-UP TITBITS, Perth, July 1988. Page 5 shows a joystick port connection and an Extended BASIC program which allows the TI99/4A to be used as a burglar alarm and page 7 shows an Extended BASIC program to print message tags.

TI BUG BYTES, Brisbane, June 1988. Report on the now well known Brisbane TI-Faire. There were representatives from every major TI99/4A user group in the country with demonstrations of software and hardware from here and around the world, good and bad news on Myarc hardware imports, a minimemory version of HELICOPTER and RTTY (radio teletype) on the TI99/4A. In August 1988, page 3 has a proposal that TIsHUG and HV99ers co-host a Faire in 1989, page 4 has a useful TI-Writer Reference Guide, page 6 has the "Geneve Corner" column, Col Christensen writes on embedding assembly into Extended BASIC, Garry Christensen writes on Bugs in computer programs including the now infamous Virus.

continued on page 31

Regional Group Reports

Meeting summary.

Banana Coast	9/10/88	Sawtell
Carlingford	19/10/88	Carlingford
Central Coast	15/10/88	Toukley
Glebe	6/10/88	Glebe
Illawarra	17/10/88	Keiraville
Liverpool	14/10/88	???
Northern Suburbs	??/10/88	Davidson
Sutherland	21/10/88	Jannali

BANANA COAST Regional Group
(Coffs Harbour area)

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

CARLINGFORD Regional Group.

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

CENTRAL COAST Regional Group.

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

GLEBE Regional Group.

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

ILLAWARRA Regional Group.

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

LIVERPOOL Regional Group

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

Meetings coming up.

14th October, at Stan Puckle's house, 15 Richmond Crescent, Campbelltown. We should have a stack of new programs from the US to demonstrate and some new hardware.

Last meeting was at Marcel Zaia's house. There were demonstrations of some new programs. Two of the programs were TIsHUG Wheel of Fortune and the US version. The US version leaves the TIsHUG version for dead.

NORTHERN SUBURBS Regional Group.

If you want any information please ring Dennis Norman on (02)452 3920, or Dick Warburton on (02)918 8132.

SUTHERLAND Regional Group.

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

Uploaded 09:17:47 04/09/88 by SUTHERLND Although the Group is small in number, there are always plenty of new and interesting subjects for discussion. TI Base and PR Base programmes are two pieces of software currently arousing interest.

Future meeting dates are October 21st and November 18th.

TIsHUG in Sydney

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

Meetings planned this year. October 1 - Joint meeting with Hunter Valley 99'ers. This meeting will not now go ahead as there has been insufficient time to organise and notify members appropriately in advance. Coupled with this is the fact that this is school holiday period and a number of members have expressed the view that it is an unsuitable time. Hopefully the idea of a joint meeting will be looked at again early for 1989.

October 8 - Software copython. Be at Woodstock at 2pm to be able to get some of the latest software. Other minor events, yet to be finalised, are also planned for this afternoon. A note on the copython - on this occasion only full disks will be copied with a maximum of 6 disks at a time. This will enable the maximum amount of copying for each member in the time available during the afternoon. For those members requiring single files copied, then this facility will be available at the November full day tutorial meeting.

November 5 - Full day tutorial workshop. Here the themes and other activities are yet to be finalised, however, like all full day events in the past, this is guaranteed to be a fun day. As usual there will be a luncheon BBQ at a very reasonable price. (Hamburger and soft drink for $2)

December 3 - Christmas party. Given a fine day this will be one of the major events of the year, with plenty of food and drink available and a great chance to chat with fellow members in a social and relaxed environment. There will be plenty of software released for this meeting, so you will have plenty to keep you occupied over the Christmas/New year holiday break. o

continued from page 30

Hunter Valley 99ers, August 1988. Joe Wright reports on a stand alone 32K with optional Supercart available from The Captains Wheel USA, Neil Quigg is awaiting orders to manufacture RAMdisk boards, another warning from Ottawa UG on using any version of DM1000 greater than V3.5, Tony McGovern writes on Assembly Squeezing (part 2), and Kevin Cox on a different approach to speech with CALL LOADS, Jack Sughrue writes about "Good Old Days Are Coming Back" and some useful hints for TI-Writer CR, SF, LF, Bob Carmany writes on the Evolution of the TI99/4A and Ron Kleinschafer on The QED Utilities Loader V3.1, Richard Terry on Struggling Forth where he codes for "Any Size Editor" and lastly Bob August writes on arrays in BASIC.

****

Rolf has just noticed an error in the June issue on page 12. There has been an insertion of words from another article into this one in the second column about two thirds of the way down the page. This is caused by a bug in either TI-Writer or DM1000. It happens to me about once each issue as I am editing articles. It is most annoying and because it occurs so infrequently is hard to decide where the problem lies. I feel that it is in the editor or the disk controller software. Has anyone else had this problem, where, after saving a file to disk, when it is next examined, lines from another file are imbedded in the middle? I would be very pleased to hear about any problems others may have, as that would help to point the finger in the correct direction. I changed to using Funnelweb to see if that would make a difference, but it has not. I have other strange problems in transferring files to RAMdisk and in using the SD command to a RAMdisk with Funnelweb which may be from the same problem and which I shall write to the McGoverns about. Any evidence from others would be greatly appreciated.

o