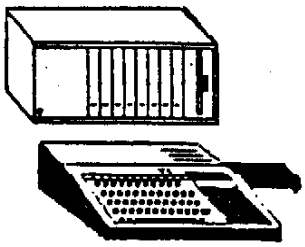
OFFICAL NEWSLETTER OF THE:
TIDEWATER 99/4 USERS GROUP INC.
51 GAINSBOROUGH PL.
Newport News, Va. 23602



NON PROFIT ORGANIZATION
U.S. POSTAGE PAID.
GRAFTON VA. 23692
Permit No. 61

DECEMBER 1987

A Non-Profit Virginia Corporation
deticated to educating and
enlightening TI-99/4 users
to the full potential
of home computing.

TT-99/4A
Dues FREE
Central Alabama 99/4A UG
551 Larkwood Drive
Montgomery AL 36109

PENINSULA OFFICERS:		
PRESIDENT	Vic Vogelsang	826-6958
VICE-PRESIDENT	Ken Silver	--------
SECRETARY	Nancy Miller	722-4953
TREASURE	Nancy Miller	722-4953
LIBRARIAN	MIKE MITCHELL	838-1942
		NEWSLETTER EDITOR

SOUTHSIDE OFFICERS:		
PRESIDENT	Ken Woodcock	85/-6151
VICE-PRESIDENT Administration	Billy Denny	420-9631
VICE-PRESIDENT Operations	Allen Leibrand	485-5889
SECRETARY TREASURE	Dirk Hanson	587-1949
LIBRARIAN Mac & Cathy	MacAllister	877-4699
Bill Miller 722-4953		

Please address any questions, requests, comments, articles, or reviews to:

Bill Miller
629 ChapeL Street
Hampton, Virginia 23669

Or call 722-4953 for voice and/or data communications.

>**********<

MEETING NOTICE

The Southside Chapter meets every first and third tuesday of each month at E.C.P.I. (Electronic Computer Programming Institute) located at 5555 Greenwich Road in Virginia Beach. Educational classes start at 6:30 P.M. followed by the regular meeting and discussion groups at 7:30 P.M.. Please note these dates and times on your calender.

The Peninsula Chapter meets every second Tuesday of each month in room 101 at Warwick High School located at 51 Copeland Lane in Newport News. Formal meetings begin at 7:30 P.M. with informal discussion before and after the meeting. The Library is open to members during informal sessions. Please note these dates and times on your calinder.

EDITORS NOTES:

The Peninsula Chapter proudly announces that the Newsletter printing costs have been reduced by the purchase of a copier. The exact cost savings has not been determined at this time but it is known for a fact that it has more the halfed the cost of printing the news letter. This also offers other advantages. For one thing we are no longer tied to an eight page minimum newsletter. Another advantage is the ability to make copies of club materials for our members at little or no cost. The newsletters we receive from other groups can be copied amd made available to both chapters making it possiable for the Peninsula Group to receive current news letters as opposed to those that are a year old.

Arangments have been made to allow the Peninsula members to have access to the program Library. The MacAlasters will be producing another catalog of the programs so that we may see what they have that we don't have as well as we being able to offer them some of the programs that we have that as yet are unavailable to them. It is this editors hope that both chapters work harmonisouly together because it is groups like these that are the life blood of the orphaned TI99/4 4A.

THE LIBRARIANS

Mac and Cathy MacAllister

There has been a lot of activity with the Newsletter Library lately. That is good. It makesthe work worth it. That's what Cathy says anyway. My back didn't. Seriously though It is being well used.

Everyone is doing pretty good at returning them quickly. We have lost very few issues to date.

One problem has developed with the Newsletter Library that needs to be stopped. Someone, or some people, are taking newletters apart to copy and are not puting them back together correctly. Sometimes they leave pages out altogether, or put the wrong pages in. I hope this is just an oversight. Remember, this library is for everyone. Copy pages if you want but leave the original in the library for other club members.

Cathy has turned over all the newsletters dated before 1 Jan 1987 to the Peninsula group.

We have only added a few programs lately to the disk library. They seem to be getting fewer. We have received Disk Cataloger (remarks can be added), Disk Repair, and Quickloader (a fairware program by Robert Amenta).

Let us all remember that fairware doesn't mean freeware. We need to support the people still producing TI-99 programs.

The only way our Library will continue to grow and be viable is if you take an active part in its improvement. If you get any new or updated versions of programs already in the Library, or modify one to improve its operation put a copy in the Library.

REMEMBER IT IS YOUR LIBRARY

DISK PROBLEMS?

Doing a 'catalog disk' fresh from our users library produced only a partial listing on screen. Then the disk started clicking like crasy. Also at the bottom of the screen I received an I/O error. After several tries with the same results I loaded up DISK FIXER to see where the problem might be. As I stepped through the sectors I got an error code 22 at the same place I did when I cataloged to screen. Hitting enter like the screen said suddenly gave me that sector. At first everything appeared O.K. After the header there were all zeros to the bottom EXECPT about 8 bytes from the end there were several different codes other than zeros. I wrote these codes down on a peice of paper for later refference. I then continued on going through the sectors looking for oddities. Then I decided to step back (FCTN 4). when I once again encountered the bad sector (03 in my case) error code 22 once again came up. Pressing enter got me into the bad sector. Looking over the sector for the second time I noticed that the last bytes were now DIFFERENT from what I had only a few moments previous recorded. Maybe I was on to something! I then moved the cursor to the bottom of the screen and changed all the odd ball bytes to zero. I then did a (FCTN 8) write to sector Y/N, answered yes, crossed my fingers, and pressed ENTER. The deed was done . Now to test the disk. I pressed (FCTN =) ,quit and went into DM 1000 and PRESTO!! The system cataloged up properly and no I/O errors were observed. The programs were then loaded and tested with no further problems.

The problem has since come up again on a different disk and following my own instructions I was able to once again correct the problem. There appears to be a possible problem with the library computer whem making copies. Some further investigations will hopefully turn up some answers. I'll keep you posted.

By Mike Couture

WEATHER-FAX NEWS

Vacations, business trips and job changes have a tendency to interrupt certain vital schedules and the WEFAX project is no exception. The following should bring everyone up to date thus far:

As of Aug, the software has been finalized for this portion of the project. The Analog-to-digital converter and digitizer have been completed and tested on pulse type signals with all OK, and a very unique inout arrangment through the joyport has been designed and tested. The success of the joyport, as the inout channel, has eliminated the requirement for the cassette port. This requirement was thought necessary and discussed in an earlier issue. Because of this unique single input design, both software and hardware have been reduced. We'll go into greated detail on this in a future issue. The circuitry mentioned above is used to process the data after digitizing into the required format to be compatable with "TI"

architecture.
Hardware used to process the analog signals from either of two satellite sources, has been built and tested satisfactory. A microwave converter for the "GEOS" satellite is still required, but pretaped analog data is being used from other non-geostationary satellites and operating on frequencies more readily accessible. Still other software work is being developed to accomplish this same feat, but using a different input port, and signal formatting scheme.

THE WEFAX GROUP

==========

DEBUGGING

by Jim Peterson

When you have finished writing a program, the next thing you should do is to run it. And, very probably, it will crash! Don't be discouraged. It happens to the very best of programmers, very often. So, the next thing to do is to debug it. And you are lucky that you are using a computer that helps you to debug better than some that cost ten times as much. There are really three types of bugs. The first type will prevent the program from running at all - it will crash with an error message. The second type will allow the program to run, but will give the wrong results. And the third type, which is not really a bug but might be mistaken for one, results from trying to run a perfectly good program with the wrong hardware, or with faulty hardware. As for instance, trying to run a Basic program, which uses character sets 15 and 16, in Extended Basic.
First, let's consider the first type. The smart little TI computer makes three separate checks to be sure your program is correct. First, when you key in a program line and hit the Enter key, it looks to see if there is anything it can't understand - such as a misspelled command or an unmatched quotation mark. If so, it will tell you so, most likely by SYNTAX ERROR, and refuse to accept the line. Next, when you tell it to RUN the program, it first takes a quick look through the entire program, to find any combination of commands that it will not be able to perform. This is when it may crash with an error message telling you, for instance, that you have a NEXT without a matching FOR, or vice versa. And finally, while it is actually running and comes to something that it just can't do, it will crash and give you an error message - probably because a variable has been given a value that cannot be used, such as a CALL HCHAR(R,C,32) when R happens to equal 0. The TI has a wide variety of error messages to tell you when you did something wrong, what you did wrong, and where you did it wrong. But, it can be fooled! For instance, try to enter this program line (note the missing quotation mark). 100 PRINT "Program must be s aved in:"merge format." And, sometimes you may be told that you have a STRING-NUMBER MISMATCH when there is no string involved, because the computer has tried to read a garbled statement as a string. Also, the line number given in the error message is the line where the com- puter found it impossible to run the program; that line may actually be correct but the variables at that point may contain bad values due to an error in some previous line. If the error occurs in a program line which consists of several statements, and you cannot spot the error, you may have to break the line into individual single statement lines. This is the easiest way to do that. Be sure the line numbers are sequenced far enough apart. Bring the problem line to the screen, put a ! just before the first ::, and enter it. Bring it back to the screen with FCTN 8, retype the line number 1 higher, use FCTN 1 to delete the first statement and the ! and ::, put a ! before the first ::, and continue. Then, when you have solved the bug, just delete the ! from the original line and delete all the temporary lines. Pages 212-215 of your Extended Basic manual list almost all the error codes, and almost all the causes of each one - it will pay you to consult these pages rather than guessing what is wrong. You may create some really bad bugs when you try to modify a program that was written by someone else - especially if you add any new variable names or CALLs to the program. Your new variable might be one that is already being used in the program for something else, perhaps in a subscripted array. I have noticed that programmers rarely use @ in a variable name, so I always tack it onto the end of any variable that I add to a program. Also, the program that you are modifying may have ON ERROR routines, or a prescan, already built in. The ON ERROR routine was intended to take care of a different problem than the one you create, so it could lead you far astray - you had better delete that ON ERROR statement until you are through modifying. The prescan had better be the subject of another lesson, but if the program has an odd looking command !@P- up near the front somewhere, it has a prescan built in. And if so, if you add a new variable name or use a CALL that isn't in the program, you will get a SYNTAX ERROR even though there is no error. One way to solve this is to insert a line with !@P+ just before the problem line, and another with !@P- right after it. When a program runs, even though it crashes or is stopped by FCTN 4 or a BREAK, the values assigned by the program to variables up to that point will remain in memory until you RUN again, or make a change to the program, or

clear the memory with NEW. This can be very useful. For instance, if the program crashes with BAD VALUE IN 680, and you bring line 680 to the screen and find it reads CALL HCHAR(R,C,CH) just type PRINT R;C;CH and you will get the values of R, C and CH at the time of the crash. You will find that R is less than 1 or more than 24, or C is less than 1 or more than 32, or CH is out of range. In Extended Basic, you can even enter and run a multi- statement line in immediate mode (that is, without a line number), if no refer ence is made to a line number. So, you can dump the current contents of an array to the screen by FOR J=1 TO 100::PRINT A(J);: : NEXT J - or you can even open a disk file or a printer to dump it to. You can also test a program by assigning a value to a variable from the immediate mode. If you BREAK a program, enter A=100 and then enter CON, the program will continue from where it stopped but A will have a value of 100. You can temporarily stop a program at any time with FCTN 4, of course (the manual says SHIFT C, but it was written for the old 99/4), and restart it from that point with CON. Or you can insert a temporary line at any point, such as 971 BREAK if you want a break after line 970. Or, you can put a line at the beginning of the program listing the line numbers before which you want breaks to occur, such as 1 BREAK 960,970,980 Note that in this case the program breaks just BEFORE those listed line numbers. You can also use BREAK fol- lowed by one or more line numbers as a command in the immediate mode. The problem with using BREAK and CON is that BREAK upsets your screen display format, resets redefined characters and colors to the default, and deletes sprites. So, it is sometimes better to trace the assignment of values to your variables by adding a temporary line to DISPLAY AT their values on some unused part of the screen. If you want to trace them through several statements, it will be better to GOSUB to a DISPLAY AT. And if you need to slow up the resulting display, just add a CALL KEY routine to the subroutine. Sometimes, your program will appear to be not flowing through the sequence of lines you intended (perhaps because it dropped out of an IF statement to the next line!) and you will want to trace the line number flow. This can be done with TRACE, either as a command from the immediate mode or as a pro- gram statement, which will cause each line number to print to the screen as it is executed. If used as a com mand, it will trace everything from the beginning of the program, so it is usually better to insert a temporary line with TRACE at the point where you really want to start. Once you have implemented TRACE, the only way to get rid of it is with UNTRACE. TRACE has its limitations because it can't tell you what is going on within a multi statement line, and it will certainly mess up any screen display. Sometimes it is better to insert tempo rary program lines to display line numbers. I use CALL TRACE( ) with the line number between the parentheses, and a subprogram after everything else 30000 SUB TRACE(X) ::DISPLAY AT(24,1):X :: SUBEND Some programmers use ON ERROR combined with CALL ERR as a debugging tool, but I can't tell you much about that because I have never used it. ON ERROR can give more trouble than help if not used very carefully, and I cannot see that CALL ERR gives any information not available by other means. Sometimes you can debug a line by simply retyping it. It is only very rarely that the computer is actually interpreting a line differently than it appears on the screen, but retyping may result in correcting a typo error that you just could not see. In fact, most bugs turn out to be very simple errors. When you are debugging a string-handling routine, don't take it for granted that a string is really as it appears on the screen - it may have invisible characters at one or both ends. Try PRINT LEN(M$) to see if it contains more characters than are showing; or PRINT "*"&M$&"*" to see if any blanks appear between the asterisks and the string. There is no standard way to debug a program. Each problem presents a challenge to figure out what is going wrong, to devise a test to find out what is really happening. Don't debug by experimen- ting, by changing variable values just to see what will happen, etc. Even if you succeed, you will not have learned what was wrong so you will not have learned anything - and if your program contains lines that you didn't understand when you wrote them, you will have real problems if you ever try to modify the program. (Believe me, I speak from experience!)

NOTES

Special thanks go out to Allen Leibrand for an outstanding job providing instructional material for the 6:30 class.

We welcome Howard Ingram, Rodney Llewellyn, Bill Durham and Arthur Stroup to the Peninsula Chapter and we welcome Jim Simpson back to the Southside Chapter. Reluctantly we say goodby to Davis Simms , Dan Risinger and Bob Jones and wish them an enjoyable move as well as a Marry Christmas and a wonderful New Year to come.

Early in the month of December, the Southside group decided to do away with the bulk mailing and return to the first class mailings. To reduce the cost of this change, we will be cutting back on the number of newsleters by half. This should allow us to produce larger and better newsletters at a lower cost. Members that are way behind in their dues will be droped from the mailing lists so if you are interested in keeping the newsletter comming regularly, you should send in your dues now as this is the last newsletter you will receive.

END OF LINE!

MORE FLEXABILITY IN USING PRBASE

by Ken Woodcock

Ever since I got version 2.1 of PRBASE from Mike Dodd (the version which works on both the TI and GENEVE) I have been urging everyone to convert their previous version data disks to this format. Now there is an additional reason to. If you have ever wished to be able to access your PRBASE data files from a BASIC/XB program, now you can!

As I was perusing a stack of newsletters from user groups in our exchange program, I came across an article in the NOV 87 WEST PENN 99'ers which caught my eye (their newsletter always has something of interest). This article was entitled "TRANSFERING PRBASE FILES" by Doug Gootee - CIN-DAY User Group. His article described a method he had devised to create a D/F 128 "SHELL" file on another disk and then transfer the PRBASE records to it using the XBASIC PRBUTIL program by John Johnson. I tried his method and it worked fine. . . . .but it seemed like a pain to have to keep a separate disk. Then last night while watching the 49'ers beat the Browns I got this inspiration! Why not build the "SHELL" file around the data on the PRBASE data disk?? Yeah, that's it! After all, Mike Couture had just given us a fine discussion on disk sector formats at the last meeting, plus, I had just received version 4.0 of "DISK UTILITIES" by John Birdwell (a great program-ya'll send him some $$). This would not be possible with v2.0 or earlier data disks because sector 0 and 1 are used as part of the "HEADER" section in those versions but Mike Dodd moved everything down two sectors to allow for a standard disk header, so we can put the file pointer onto sector 1: the problem is, where to put the FILE DESCRIPTOR RECORD (hereafter called the FDR). The FDR consumes one complete sector and contains all the information about the file such as name, type, record length, location, size etc. It can exist anywhere on the disk - the pointer in sector 1 tells the disk controller where to find it. If you are running single density (TI controller), the best location is probably the last sector on the disk - 359 (>167) for single sided or 719 (>2CF) for double sided. This will mean you'll have to give up one more record in addition to the 2 that Mike Dodd took leaving 347 maximum for SSSD and 707 max for DSSD. Well, as the wise man said. . .You must give something to get something. If you are fortunate enough to own a double density disk controller, you don't have to give up anything. In fact you can even take back the 2 records that Mike Dodd appropriated! In this case, format the disk DSDD. Then put the FDR at sector 722 (>2D2). This is the next sector following the space reserved for the 710 (max) records. The rest of the disk space will be available for normal program storage after we "doctor-up" sector 0 to prevent overwriting the PRBASE data. Ok, lets get started. We will have to use a sector editor to setup sector 0,1 and the FDR (I highly recommend DISK UTILITIES). Lets do sector 0. Go to byte 56 (>38). On a newly initialized disk this should be 03. Starting here, put an F in each position - all the way to the end if the disk is not DSDD. For a DSDD disk, only go through sector 146 (>92). Write this edited sector, then go to sector 1. In the first 4 positions put the sector number of the FDR (0167 for SSSD 02CF for DSSD/SSDD 02D2 for DSDD). Write this sector to disk. Now all that remains is to create the FDR. The easiest way to get most of the information for this sector is to run this program using another freshly initialized disk of the same format.

```
OPEN #1:"DSK1.XXX",RELATIVE 346,fixed 128::PRINT #1,REC 346:"X"::CLOSE #1
```

Substitute 706 for 346 (2 places) for SSSD or SSDD disks. For DSDD use 709. (Relative files start with record 0) Now you can copy the FDR from this disk (it should be at sector 2) to the disk you are setting up. Remember to copy it to the appropriate sector (359 (>167) for SSSD, 719 (>2CF) for DSSD/SSDD or 722(>2D2) for DSDD). Now all that remains is to modify 3 bytes on this FDR. These are 28, 29 and 30 (>1C, >1D and >1E). DON'T FORGET THAT THE FIRST BYTE IS ZERO!

SSSD - 0C A0 15 DSSD/SSDD - 0C 20 2C DSDD - 0C 60 2C

If this seems like alot of work, it only has to be done once. Thereafter you can copy the three sectors to your other data disks. Besides, think of all you'll learn from the experience! If you're really not in the mood for learning, send me a disk initialized in the desired format with a reusable mailer and return postage and I'll do it for you. (If double-density, 18 sector/track only).

Ken Woodcock
4701 Atterbury Street
Norfolk, VA 23513

How to use your 256K Horizon Ramdisk with GENEVE
by Ken Wooccock

1. Initialize the ramdisk.
2. Boot up your sector editor. Edit sector 0 starting with byte 10 (>A) (REMEMBER TO START COUNTING WITH ZERO). Byte 10 and 11 indicate the total sectors on the disk. Change it to 03E0 (992 decimal). Change the next byte from 09 to 10. Change bytes 18 and 19 to 0202 (indicates double side/double density). Now goto byte 56 (>38). You should find 03 there. If not, put it there. This is the start of the sector bit-map. The 03 indicates that sector 0 and 1 are used. Put zeros in all the following positions until you get to byte 180 (>64). Put F in all the remaining positions.
This assumes that you are using the 8K ROS that was used with JJ's MENU program ver 6.3 or later, otherwise you will have to modify the bitmap and bytes 10 and 11 for 976 sectors.

The easy way to accomplish this is to copy sector zero of a HRD newly initialized on a [illegible] ystem onto a disk, then copy that sector zero into the HRD in the GENEVE system.

==========

A bug exists in the Show Directory (SD) command of MY-WORD. If you catalog a disk which has exactly 21 files on it, pressing the PgDn key will fill the screen with "garbage" requiring reboot.

==========

Selecting Your Colors in MY-WORD
by Ken Woodcock

You say you're not happy with any of the five color combinations which are available by using Control 3 in the Editor of MY-WORD? Well, you don't have to be stuck with what "they" gave you. To install your own choices, break out your handy SEDCTOR EDITOR program (DISK UTILITIES by John Birdwell is my favorite) and get to work. The sector that you want is the 3rd sector of the file EDITOR. Starting with byte >xx you will see 87F4 87F3 8717 87F6 871A (in hexidecimal). Leave the 87's alone. The byte following each 87 contains the FOREGROUND and BACKGROUND color for one choice. Example: the 1st choice is 87F4. The F sets the foreground color to WHITE, the 4 sets the background color to DARK BLUE. If you don't like that, change it to suite you. When you start up after making your changes, you will still have the last colors that you saved with SaveOptions (SO), but only the new combinations can be selected with CONTROL 3. Below is the table of colors.

0	TRANSPARENT
1	BLACK
2	MEDIUM GREEN
3	LIGHT GREEN
4	DARK BLUE
5	LIGHT BLUE
6	DARK RED
7	CYAN
8	MEDIUM RED
9	LIGHT RED
A	DARK YELLOW
B	LIGHT YELLOW
C	DARK GREEN
D	MAGENTA
E	GRAY
F	WHITE