NEW HAMPSHIRE 99'er

JUNE 1988
NEWSLETTER VOL 6 NO. 6
POB 5991 MANCHESTER, NH 03108

CLUB NEWS

by Paul Bendeck, President

>OLD
Last meeting... a new beginning! Our first meeting at the SEE Center was a big success! This new location gives the club a much better, more flexible atmosphere for our technical discussions and demos. Maybe now we can get organized and have a few seminars. Any suggestions on what you would like to see first?

>NEW
Elections were held at the June meeting. The new club officers for the coming year are:

Paul Bendeck - President
Chris Agrafiotis - Vice President
Bob Bouchard - Treasurer
Tom Dupras - Secretary/Newsletter Editor

All of the officers except for Tom are incumbents. Tom will be taking over responsibility for the club newsletter from Curtis who has really done an outstanding job with the newsletter over the past year and more. Thank you Curtis, and good luck to Tom.

The raffle for the disk drive was held finally at the June meeting. This was delayed from last month due to the change in meeting locations. See the whole story inside this issue on how much money was raised and who won the prize.

Just a reminder about our program library. FREDDY is in stock again! Also, there are lots of copies left of all of the demo disks we put together for the fair. These disks and more will be available at the next club meeting.

Due to the holiday on Monday July 4, the July meeting has been moved to Tuesday July 12. The August meeting will return to the first Monday, August 1. Meetings start at 7:00 PM at the SEE Center 324 Commercial Street, Manchester, NH (near the Granite Street exit off I-293).

This newsletter is paid for by your annual dues. If you want to keep receiving this newsletter each month, please make sure that your dues are paid up. The club can not afford to keep sending newsletters to people who do not pay their dues on time. I will try to have a current membership list at each meeting so that you can check when your existing membership expires.

See you at the meeting.

QUADRUPLE YOUR DRIVE RESPONSE

by

Paragon Computing

With a very simple modfication, you can make a single drive respond to any drive command - DSK1, DSK2, DSK3, or DSK4. This will significantly speed up those programs that require disk name searches (such as MULTIplan, MYWORD, etc.)

While I prefer TI's method of drive access over IBM's, I was very unhappy with the search mechanism for disk names. The process of cycling through numerous drives - some of them non-existent - took precious time away from editing, recalculating, or whatever.

The advent of the Horizon RAM disk made TI's floppy search even worse. If the Horizon disk is not set for CRU address >1000, then your floppy controller will generally be searched first (there are programs that have worked around this - but not many).

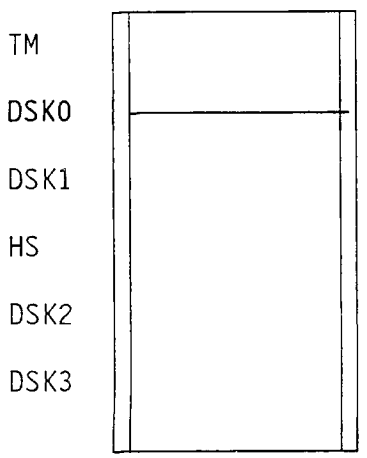
I named my Horizon RAM disk TIMP (the MULTIplan disk name) and filled it with MULTIplan overlays, expecting them to loaded with lightening speed when needed. They WERE loaded almost instantaneously - after the floppy controller took several seconds to ensure that the disks in drives 1 through 3 weren't named TIMP. The worst part was that I didn't HAVE three drives, so the check-out took even longer.

When I received my GENEVE, I also received Peter Hoddie's excellent MY-WORD processor. This program also loads files from the first disk it finds named MYWORD. I moved my Horizon RAM disk into the GENEVE box, named it MYWORD, and stuffed all the files into it. Unfortunately, the same thing happened - unbearably long delays while GENEVE looked through drives 1 through 4 (CORCOMP controller on this system).

I decided to make one drive respond to any and all drive requests. In other words, even if you have only one floppy drive, you can use DSK2, DSK3, or DSK4 (if you have MYARC or CORCOMP cards). The biggest benefit comes from disk name searches, although there is another benefit I'll mention in a moment. With a single disk in the drive, the floppy controller will search it three or four times (depending on the card) very quickly then go on to your RAM disk.

Even if you don't have a RAM disk, the speed obtained by circumventing non-existent drives is worth this simple modification.

Every make of drive I have ever seen has either a socket or set of jumper pins in one corner that looks like this:



The actual acronyms will vary, as well as the number. Just look for a socket with printing along side. If the printing includes #'s 0 to 3 or 1 to 4 then you have found it!

For each drive number, connect the open pins from one side to the other. In this example, the drive is factory set, i.e it will respond to DSK1.

Modify drives at your own risk!

The actual order of the markings may vary, but you will invariably see DSK0, DSK1, DSk2, and DSK3 (or equivalents). If you complete the connection from one side of the socket (or pins) to the other, then you will enable that drive to respond to that sequence. Noting that the numbers start with DSK0, if we complete that circuit as well as DSK2, then the drive will respond to DSK1. and DSK3. We can enable all four the same way.

********** NOTE **********

The original TI cables provided a 'key' which was a small printed circuit board designed to shift drive lines in the cable. The card would essentially disconnect the original DSK0 line and shift the remaining lines over one so that DSK1 would now activate the DSK0, DSk2 would activate DSK1, etc. The reason behind this was that new drives come with only DSK0 enabled. By connecting each additional drive through one of TI's little circuit cards, the user never had to modify the drive. The point of all this is to warn you to also shift your settings. For example, assume that you have a drive in the box and a stand-alone drive connected through TI's little circuit card. You want the box drive to respond to DSK1 and DSK3 and the stand-alone to respond to DSK2 and DSK4. On the box drive, complete circuits DSK0 and DSK2 - on the stand-alone, also complete DSK0 and DSK2.

Completing the circuit can be as simple as jammimg a big staple into the socket (that's how I originally tested my theory). The pin connector may require more finesse - I suggest either using commercially available jumper connections or wire-wrapping. If you can't do this yourself, bring you drive to the club meeting and I'll modify it for you (just call be a few days before the meeting t warn me).

I mentioned earlier that there was another good reason for this modification besides disk name searches. That reason involves large programs which require 'data disks.' Some programs practically fill an entire SSSD disk, and then require that you provide a disk for temporary and permanent storage. However, even the TI disk controller can handle a double sided drive, so my first upgrade was to purchase a double sided drive. I could then copy the program onto a disk and still have 360 sectors left over for data. BUT! The program was smart enough to realize that the drive number of the data disk was the same as the program disk. Every time I wanted to switch access from data to program or back, I was prompted to swap disks and press a key.

The multiple access changed all that. I tell the program that the program disk is in drive 1 and the data disk is in drive 2 - but there really is only one drive and one disk. The program doesn't miss a beat.

********** WARNING **********

If you fool a program into thinking that it is talking to two different disks, you may lose or corrupt some files. Some programs modify the data disk to store special information. This modification may destroy program sectors. An example of this would be using PRBASE. If you forget that you have modified drive 1 to respond to drive 2, you might destroy your program disk (assuming you committed the cardinal sin of not using a write protect tab).

Have fun - but be carefull!

AND THE WINNER IS ....

For at least one of our Canadian brothers-in-TI, the trip to the April FAYUH was worth the expense. At the June meeting we raffled off the quad density, double sided drive donated by Paul Johnson. Mr. Vallieles of Bromptonville, Quebec was picked by our very own Helene LaBonville (those Frenchies always stick together....).

The purpose of the raffle was to raise money to buy TWO half-height, double density, double sided drives and ship them to the McGovern's in Australia. Toward this goal we have raised $75. This doesn't seem like much, when you consider that the drive we raffled off was worth more than that - BUT - we have our scouts out looking for a good deal on new drives and might be able to pull it off.

The idea for the raffle was from club member Paul Johnson. Paul has been in regular contact with the McGovern's - authors of FUNNELWEB and other utility programs for the TI. Paul remarked that equipment was outrageously priced down-under - when it was even available!

Paul thought that it would be a nice gesture if at least one user's group in the US (aka Land-o'-plenty) showed their appreciation for the development and continued support of FUNNELWEB. I can assure you that developing large programs on a TI is no fun unless you have the hardware to support the effort. Working with a single drive - particularly one that is single sided - is a lesson in frustration. Let's hope that the McGovern's are able to make excellent use of this gift. They deserve it.

Let's also thank Paul Johnson for his generous donation and for coming up with the raffle idea. If we all give of ourselves just a little, our club will grow stronger and benefit us all.

July 1988

SUN	MON	TUE	WED	THU	FRI	SAT
					1 CAN-ADA DAY	2
3	4 INDEP DAY	5	6	7	8	9
10	11	12 NEXT MEETING 7:00 PM	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

SELF-MODIFYING PROGRAMS

by

PARAGON COMPUTING

During a recent phone conversation with Barry Traver, he described an adventure program he had translated from another computer. Adventure programs typically deal with a large amount of data which must be 'reconfigured' depending on the adventurer's actions. In particular, Barry's version has numerous lines of data - e.g. one for each room. Using the RESTORE command, Barry was able to select which lines of data would be read next.

This problem would typically require a large number of RESTORE's accessed from either almost as many ON GOTO's or ON GOSUB's. However, Barry came up with an easier way - self-modifying code.

If you have studied software development at all, you have heard that self-modifying code is BAD, NO-NO, NAUGHTY! If you are responsible for millions of lines of code, or someone's life might be endangered by a software fault, then I agree. However, programming games on the TI hardly falls into either category.

And let's face facts - there is less than 24K of space available for XBASIC programs (some room is taken up by numeric variables). If we are to squeeze every ounce of performance out of this thing, then I say GO FOR IT!

If Barry had already solved the problem, then why this article? Barry's version actually used XBASIC code to PEEK system data, compute the location of the line number to be changed, then LOAD a new value. Although effective, this method is slow and consumes valuable XBASIC program space. This is just the type of challenge that Paragon Computing wants.

The code listed on the next page was written to modify mbers within any XBASIC statement. The context of the number is not important, as long as it isn't in a comment. For example, line numbers can be changed inside (ON) GOTO, (ON) GOSUB, RESTORE, IF-THEN-ELSE, DISPLAY (and PRINT) (AT) USING, ON ERROR, RETURN, BREAK, and UNBREAK. Let's look at some examples:

1) You are attempting to debug a program. Every time you get one thing fixed, another problem pops up. You have decided to BREAK the program immediately before an error, so that you can examine variables. Are you going to rewrite the BREAK line every time you run the program? Why not have a variable BREAK? -)

```
100 CALL INIT :: CALL LOAD("DSK1.CHANGE")
110 INPUT "BREAK at which line? ":A :: CALL
LINK("CHANGE",120,1,A)
120 BREAK 100
*** (rest of program)
```

2) You have a generic error handling routine for closing bad files, but you want to RETURN to different parts of the program. Simply set the RETURN ### in each specific block:

```
100 CALL INIT :: CALL LOAD("DSK1.CHANGE")
***
200 CALL LINK("CHANGE",1010,2,300)
*** (some file access section here)
300 !CONTINUE WITH PROGRAM
***
1000 ON ERROR 1010 :: CLOSE #1 :: 0=""
1010 ON ERROR 1000 :: RETURN 0
```

(The CLOSE might create another error. If it doesn't, the 0="" will. Either way, the file will be closed and control passed to line 1010)

3) You have a program which displays numbers in different formats (e.g. spreadsheet). Instead of using different DISPLAY (AT) USING's, use different IMAGE's:

```
100 CALL INIT :: CALL LOAD("DSK1.CHANGE")
110 DISPLAY AT(23,1):"Enter desired format #:     1-INTEGER
2-MONEY 3-SCIENTIFIC"
120 ACCEPT AT(24,28) SIZE(1) VALIDATE("123"):C
130 CALL LINK("CHANGE",300,1,300+C)
*** (program to calculate values, etc.)
300 DISPLAY AT(ROW,COLUMN)USING 0:A
301 IMAGE ##########
302 IMAGE $##########.##
303 IMAGE #####^^^^
***
```

Although these examples have been rather trivial, it should be obvious that extremely powerful program control is now possible in XBASIC. However, with that power comes responsibility and the need for extra safeguards. I hate to think of what would happen if you fully debugged a program containing CHANGE - and then resequenced it!

****

If you would rather not enter the source code and assemble it yourself, you may type in the object code directly:

Save to disk in DIS/FIX mode using TI-writer (or clone) as follows:
Press PF (for PrintFile), then type in this line:
F DSK1.CHANGE
The "F" will 'print' the file to disk in DIS/FIX mode which is the format required by object code loaders.

```
0007E          A0000A000CBC900BC80BC007CB02E0C0000B0205C0004B04C07F2F9F    0001
A001CB0201B0001B0420B200CB0420B2018B12B6B05B1BCD60B834AB02B57F2E7F         0002
A0032C000AB11F5BC020B8330BC060B8332B90B0B1604B9830C0005B13097F2E5F         0003
A004BB0600B0220B0003B8040B11F6B0200B1C00B0420B2034BD0B0BD8107F307F         0004
A005EC0005BD032B13F7B9180B16FCB0603B1302B05C2B10F8BDC84BDCA07F263F         0005
A0074C0009B02E0B83E0B0460B00007F968F                                       0006
5000ECHANGE7FD19F                                                          0007
:PARAGON COMPUTING - 17 CONSTANCE STREET - MERRIMACK, NEW HAMPSHIRE - 03054 0008
```

```
*****************************************************************************
* CHANGE: Written by Curtis Alan Provance on June 3, 1988 5:25 PM           *
* This routine changes line numbers inside XBASIC statements.  Format is:   *
* CALL LINK("CHANGE",statement_number,position_in_statement,new_line_number)*
* For example, if line 200 reads: 200 ON A GOTO 210,220,230                 *
* Then CALL LINK("CHANGE",200,2,240) makes: 200 ON A GOTO 210,240,230       *
*****************************************************************************

R2LB    EQU  $+5                Point to R2's lower byte
R4LB    EQU  $+9                Point to R4's lower byte
MYWSP   BSS  12                 Reserve registers R0 through R5 for use by routine
        DATA >C900              R6 - This is the flag that indicates line number
        DEF  CHANGE
CHANGE
        MOV  R11,@SAVRTN        Save the GPL return address
        LWPI MYWSP              Don't use GPL workspace
        LI   R5,MYWSP+4         Point to R2 in the current workspace
        CLR  R0                 This routine assumes no array passing, i.e. B()
        LI   R1,1               Get the parameters in order of CALL LINK
LOOP1
        BLWP @>200C             Get the number
        BLWP @>2018
        DATA >12B8              Convert the floating point into an integer
        INC  R1                 Prepare to get the next parameter
        MOV  @>834A,*R5+        Save each value as it is converted
        CI   R5,MYWSP+10        Have we loaded all three values?
        JLT  LOOP1              Do this loop three times
        MOV  @>8330,R0          This points to the last line number in table
        MOV  @>8332,R1          This points to start of actual lines
LOOP2
        CB   *R0+,R2            Try to match the first byte of line number
        JNE  NEXT               No match, try next one
        CB   *R0+,@R2LB         Try to match the second byte of the line number
        JEQ  FIND               There is a match!  Find the line to be changed
        DEC  R0                 Match R8 condition from first byte mismatch
NEXT
        AI   R0,3               Point to next line number in table
        C    R0,R1              Have we reached the end of the table?
        JLT  LOOP2              No, go back and keep looking
ERROR
        LI   R0,>1C00           This error reports "Bad Argument"
        BLWP @>2034             Report the error and return to program
FIND
        MOVB *R0+,R2            Get high byte of line address in R2
        MOVB *R0,@R2LB          Get low byte of line address in R2 lower byte
LOOP3
        MOVB *R2+,R0            Get the next byte to be checked
        JEQ  ERROR              Reached the end of the line - didn't find it
        CB   R0,R6              Is this a line number flag?
        JNE  LOOP3              No, go back and get another byte
        DEC  R3                 Is this the line number we want?
        JEQ  REPLAC             Yes, replace it and return
        INCT R2                 No, push past this one and try again
        JMP  LOOP3
REPLAC
        MOVB R4,*R2+            Load the high byte of the address
        MOVB @R4LB,*R2+         Load the low byte of the address
        LWPI >83E0              Restore GPL workspace
SAVRTN  EQU  $+2
        B    @0                 Ready to return!
        END
```

****Announcing****
"A New Magazine for the TI-99/4A"

The 99/4A is a remarkable story that isn't over yet. Even 5 years after being discontinued there are still hundreds of thousands of users of this remarkable device. The range of familiarity with the 4A is almost as wide as the range of users - everyone from the bare beginner with a garage-sale console up to the "power-user" with hard-drives, and everything in between. The number of applications is equally as broad - tens of thousands of programs run on the 4A from 9 year old musty databases to the latest in word processing.

No one or dozen magazines can adaquately describe this incredible diversity. Asgard Publishing felt that there were a lot of things that weren't being covered - that is why we created ASGARD NEWS.

What is ASGARD NEWS?

Well, for the owner of one or more Asgard Software products it is an indispensible source of tips, update notices, and new or related product information. It will tell you what's wrong and what's right about what Asgard Software publishes, and what we have up our sleave next for the TI community.

Fpr the average user, it is a source of the most important news from the grapevine - what the "movers and shakers" of the TI community are doing now. It will keep you informed without you having to spend hundreds of dollars on telecommunications networks and phone calls.

For the person in the market for 99/4A hardware - it is an unbiased, nuts and bolts reviewer of the latest in TI technology. It will help you by dissecting a new device for you so YOU won't have to spend hundreds of dollars before finding out it's a lemon or a prize.

For every TI-99/4A owner, ASGARD NEWS is a forum where all the issues facing the 99/4A will be discussed - from every angle. It is a magazine of opinion that is an intellectual free-for-all. We'll deal with the hard issues - what is the significance of the Geneve? Is piracy killing software support? Where will the 4A be in 2 years? 10 years?? ASGARD NEWS invites anyone with a clear opinion on anything to speak up - and anyone who doesn't like what is said to write in response.

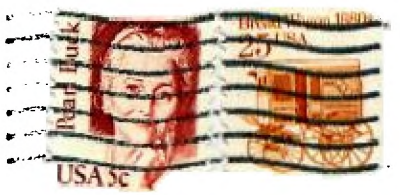
Finally, ASGARD NEWS is devoted to the non-programmer. It is a users magazine with an emphasis on "users". We recognize that 90% of 99/4A owners don't care about programming, and aren't interested in programming. All most users are interested in is getting the most from what they have, and finding out about things they don't have. They want to know what something is USEFUL for - what BENEFITS it offers - not feature lists.

In short, ASGARD NEWS is a news magazine for the 4A. We'll keep you informed, we'll outrage, and we'll prove you right or wrong. If you are interested in the 99/4A - this magazine will interest you.

This 16-32 page quarterly can be obtained until July 31st for only $6.00/4 issues - 50% off the cover price! Canadian, European and Australian owners please add $3.00 for air mail delivery.

An offer like this will never come again - take advantage of our special introductory price because soon it - like the Commodore Vic 20, the IBM PCjr and the Coleco Adam - will disappear.

NEW HAMPSHIRE 99'ers
PO BOX 5991
MANCHESTER, NH 03108



EDMONTON USERS GROUP

PO BOX 11983

EDMONTON, ALBERTA

CANADA T5J-3L1