THE R O M NEWSLETTER
USERS GROUP OF ORANGE COUNTY
173Ø1 SANTA ISABEL STREET
FOUNTAIN VALLEY, CA 927Ø8

5/89
Dallas TI Computer Group (DTIHCG)
PO Box 29863
Dallas,
TX 75229

JUNE 1989

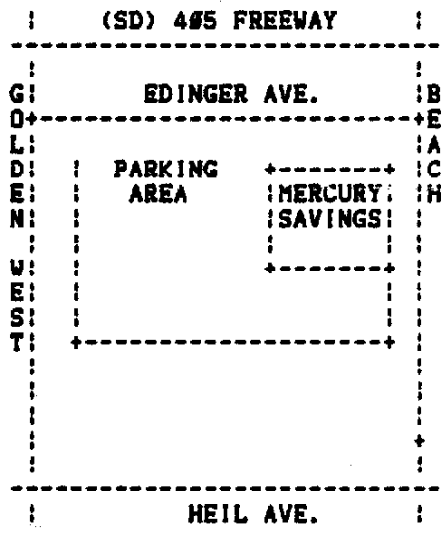
SERVING THE TI 99/4A HOME COMPUTER COMMUNITY

WE MEET AT MERCURY

TIME AND PLACE OF MEETING
The **FIRST** Thursday of each month at

MERCURY SAVINGS and LOAN
7:3Ø PM

West of Beach at 7813 Edinger Ave., Huntington Beach, Cal.
Use the WEST enterance. Park on the west side of the building. All are welcome.



H.U.G.O.C. OFFICERS

PRESIDENT	JIM SWEDLOW	897-92Ø9
VICE-PRES	JIM JOLLY	531-7782
SECRETARY	EARL RAGUSE	847-5875
TREASURER	JERRY RASH	631-Ø579
PAST-PRES	ROBERT HARPER	744-2517

COMMITTEE CHAIRMEN

JIM MORRIS	MEMBERSHIP	546-8354
SILES BAZERMAN	NEWSLETTER	897-2868
BEN HATHEWAY	BULLETIN BOARD	751-4332

SOFTWARE LIBRARY

KNUTE ERSLAND........................842-Ø859

NEWSLETTER LIBRARY

EARL RAGUSE....(Phone for time)...847-5875

PUBLICATION NOTICE

The ROM is published by the Home Computer Users Group of Orange County, a non-profit organization. Permission is granted to similar groups to reproduce articles published in the ROM provided author and source credits are included. Opinions expressed in this publication are those of the author and do not necessarily represent those of the ROM, it's Editor, or the HCUGOC, it's officers or members.

We solicit letters and articles of interest to the TI-99/4A user community. Material accepted may be edited for fit and format. No payment is offered nor intended (other than your byline).

NEWSLETTER CONTRIBUTORS

NEWT ARMSTRONG	EDITOR EMERITUS
EARL RAGUSE	PRODUCTION
EARL RAGUSE	TI FORTH
ADRIAN ROBINSON	ASSEMBLY
ROBERT SHAFFER	CIRCULATION
JIM SWEDLOW	AT LARGE
BILL NELSON	GRAPHICS

TI CLUB ACTIVITIES

CLUB	ACTION	DATE	INFO
BUG	GENERAL MEETING	JUL Ø3	532-1554
UGOG	GENERAL MEETING	JUL Ø6	897-92Ø9
ET99	GENERAL MEETING	JUL Ø8	837-8757
UGOC	LIBRARY, FTNVLY	JUL 1Ø	842-Ø859
UGOC	ASSEMBLY SIG	JUL 13	837-8757
UGOC	LIBRARY, FTNVLY	JUL 17	842-Ø859
UGOC	BOARD MEETING	JUL 2Ø	897-92Ø9
UGOC	ASSEMBLY SIG	JUL 27	537-1839
UGOC	NSLETTER LIBRARY	CALL	847-5875

UGOC HALL OF FAME
by Bill Nelson

This months Hall of fame selection is Barry A. Traver. He is the one that started us saving money on the phone line with Archiver 1.

Barry doesn't stop there, he is a SysOp (system operator) for GEnie, the editor of GENIAL TRAVelER, and has put out many a fine game programs. From my understanding he is also one heck of a nice guy to talk to.

Thank you Barry for all your time and devotion to the TI community. Without people like you our orphan would truly die.

MEMBERSHIP CORNER
By Jim Morris

Our membership is currently at sixty eight. Five are currently ninety days in arrears. Eleven members memberships are currently due. One member was dropped because of other interests. Three new prospective members are expected to attend the June meeting. We obviously have an attrition problem and somehow although we have a number of new prospective members very few actually join the club. Club ads placed in various computer publications are our main source of prospective new members. Some way has to be devised to increase our membership;any bright ideas would be appreciated. To date I haven't received even one input regarding how we can increase our membership!

A large number of updated library lists have been passed out and as you know a large number of excellent programs are available so please patronize it if at all possible.

UGOC
BULLETIN
BOARD
3ØØ/12ØØ/24ØØ BAUD
(714) 751-4332

MINUTES OF 5/18/89 BOARD MEETING
By Earl Raguse, Secretary

The meeting of the newly elected board was called to order at 7:38 PM by VP Jim Jolly, at the home of Siles Bazerman. President Jim Swedlow was absent on business. Others present were Treas. Jerry Rash, Sec. Earl Raguse, MC Jim Morris, NLE Siles Bazerman, Stan Corbin, and Adrian Robison.

There were no minutes of last meeting, as the old Secretary was absent. It was agreed to table the question of money for the Grandfathered Hall of Famers until Jim returns.

Jim Morris reported that we currently have 68 members, 12 of which owed dues, 5 members were 5 months or more in arrears. E. Raguse was instructed to put "Last Copy" stickers on their ROM labels.

Siles Bazerman distributed a preview copy of the May ROM, he also delivered masters to Earl Raguse for printing. There was discussion of the offer by new member, Stu Haynie, to get the ROM printed free. Earl Raguse, current production chairman, is to contact Stu to see if arrangements can be made to get the printing done in the small time window available in order to get the ROM to members before the meeting. (FPO willing of course).

Siles is still looking for authors who are willing to write articles of interest.

The Boise club sent us a letter explaining that they had to discontinue newsletter exchange because they have dwindled to 5 members. They offered to pay some reasonable amount to continue the ROM even though they could not afford full membership. It was agreed to continue sending the ROM without charge provided they let us know periodically that they were still alive and reading it.

Siles reported that The Tidewater group plans to send their newsletter on disk. Mailing will be quarterly only.

The June meeting program will include a demonstration of TIWriter Control U codes by Earl Raguse, and Formatter dot codes by Siles Bazerman.

Irwin Metz has agreed to give us a demo on Multiplan usage at the July meeting.

The Hall of Fame names to be proposed next meeting are John Wilforth, Barry Traver and Charles Earl.

Meeting adjourned at 9:43.

Siles served coconut macaroons ect. We may meet at his house more often.

AND NO FORTH #Ø3
By Earl Raguse

Last time we could use FILO to create a file and display it, but we couldn't save it or recall it, or edit it. This time I will explain all. I hope you typed in last month's installment, up to the line 3Ø6Ø ! **PRINT A FILE**.

If you selected 6. STORE FILE TO DISK, from the menu, line 45Ø would send you to line 161Ø ! **STORE ON DISK**. There 164Ø would check if you had created (or loaded) a file, and if not, sent you off to 422Ø (kindergarden) to repent your sins. 165Ø-169Ø inquires as to the drive where you are hiding the data disk, and gives you the opportunity to specify the file name you are interested in. My original program was a one drive system, and this bit of protocol was omitted. Because of my desire to change file formats, explained last time, I added drive selection and the ability to change file names, not much additional complication. 171Ø lets you know what the program is doing, a nice touch I think.

Line 172Ø actually opens the file as DISPLAY, VARIABLE 8Ø, UPDATE format. The latter is redundant, since this is the default format. See Newt's Basically Basic in the March ROM. I kind of like that redundancy for my own info. If write it while I am still smart from reading the manual, it helps me later on when I forget these niceties.

Lines 173Ø-175Ø PRINT #1: records the file name, HEAD$, the number of records, N, and the number lines per record, NL, to the file. This could be done with one print statement with the items separated by commas, but then they would print on one line in the file, I like them better on separate lines, its easier to find them when you look at or create a file with TIW.

Lines 176Ø-18ØØ use a nested loop to print the records to disk. Then 181Ø closes the file 182Ø opens it again in R/I format for reading the file status info used for 184Ø-188Ø in printing sectors used and free, nice info to know. 189Ø-19ØØ lets you get back to the menu (26Ø).

RECALLing a file is essentially the same process except for the opening as INPUT DV 8Ø. The file is also opened in R/I format to read the sectors remaining.

Lets go back to Menu and pick 5. SORT ALPHABETICALLY, line 45Ø sends you to 139Ø ! **SORT FILE**. Line 141Ø insures that you have a file to sort. The actual sorting code is short. After 142Ø finds out which line (field) of the record you want to sort, and 143Ø warns you this might take a while, a loop 144Ø-157Ø sets FLAG=1 then compares consecutive file fields and if out of order, swaps their position and sets FLAG=Ø if a swap is made. This continues until FLAG remains >Ø indicating all is in order, at which time control returns to Menu (26Ø).

Suppose you select 2. EDIT FILE, from the menu, line 45Ø sends you to line 222Ø ! **EDIT THE FILE**, again 224Ø verifies that you have, in fact, a file to edit. Lines 225Ø-234Ø present a sub-menu for you to select the desired operation and routes you to the right step sequence.

APPENDING a record just continues with the sequence for creating, at line 55Ø.

DELETING a record, lines 237Ø-258Ø, consists of finding the specified record, verifying that that it is, in fact, the one you want to delete, then overwriting it A$(I,L) with A$(I+1,L) all the way to the bottom of the file and decrementing the number of records N.

INSERTING a record, lines 282Ø-3Ø4Ø, consists of finding and verifying the right place, moving all subsequent records down to make room, accepting the new record and incrementing the record count.

CHANGING a record, lines 26ØØ-28ØØ, consists of finding the specified record, verifying that it is, in fact, the right one, displaying it, then editing it using ACCEPT AT.

I going to skip over File Printing and Searching till next time. The remaining menu selection item is Ø. CATALOGING THE DATA DISK, (or any disk for that matter). The program here is essentially identical to that given by TI in the manuals on disc controller and drives. Don't expect me to tell you why the catalog info must be accessed in the R/I format, that's the way it is. It may help explain why I first used that format for my files. I said that format must have some purpose, and just maybe this is it. Its such a short file that inefficiency probably doesn't matter.

Next time I will finish the explanations of file searching and printing. The latter is important to understand because to that section that is most likely to need changing to make FILO suitable to a specific application that I have not anticipated.

C U next time.

Stop Presses! Stop Presses!! The following corrections should be made on earlier stuff

```
49Ø CALL CLEAR :: N=Ø
55Ø Insert ERASE ALL after (16,1)
146Ø Insert  ;"LEFT TO SORT" after (2Ø,2):X
147Ø FLAG=1
15ØØ FLAG=Ø
```

```
3Ø7Ø !
3Ø8Ø IF FLAG=Ø THEN 422Ø
3Ø9Ø DISPLAY AT(1Ø,6)ERASE A
LL:"READY TO PRINT"
31ØØ DISPLAY AT(12,5):"SELEC
T FORMAT DESIRED"
311Ø DISPLAY AT(14,4):"1. RE
PORT"
312Ø DISPLAY AT(15,4):"2. LA
BEL"
313Ø CALL KEY(3,K,S):: IF K=
49 THEN 314Ø ELSE IF K=5Ø TH
EN 345Ø ELSE 313Ø
314Ø T1=4 :: T2=9 :: T3=4Ø
315Ø DISPLAY AT(18,4):"ENTER
 DISK NAME AND DATE  "
316Ø LINPUT "name mm/dd/yr >
 ":DD$
317Ø T=32-(LEN(HEAD$)/2)
318Ø OPEN #1:"PIO",DISPLAY ,
OUTPUT
319Ø LC=14
32ØØ PRINT #1:TAB(T);HEAD$:
:
321Ø PRINT #1:TAB(32-(LEN(DD
$)/2));DD$: :
322Ø PRINT #1:TAB(32-4);"FIL
ENAME": :
323Ø PRINT #1:TAB(32-(LEN(F$
)/2));F$: :
324Ø FOR I=1 TO N
325Ø PRINT #1:TAB(T1);I;
326Ø PRINT #1:TAB(T2);A$(I,1
);
327Ø FOR L=2 TO NL
328Ø LA=LEN(A$(I,L))
329Ø IF LA<35 THEN 335Ø
33ØØ P=POS(A$(I,L)," ",3Ø)
331Ø IF P=Ø THEN P=35
332Ø PRINT #1:TAB(T3);SEG$(A
$(I,L),1,P):: LC=LC+1
333Ø PRINT #1:TAB(T3);SEG$(A
$(I,L),P+1,LA):: LC=LC+1
334Ø GOTO 336Ø
335Ø PRINT #1:TAB(T3);A$(I,L
);:: LC=LC+1
336Ø NEXT L
337Ø IF LC<6Ø THEN 34ØØ
338Ø PRINT #1:CHR$(12)!Form
Feed
339Ø LC=4
34ØØ NEXT I
341Ø PRINT #1:CHR$(12)!Form
Feed
342Ø CLOSE #1
343Ø GOTO 26Ø
344Ø !
345Ø !  **PRINT A LABEL**
346Ø !
347Ø DISPLAY AT(1Ø,Ø)ERASE A
LL:"TURN ON THAT **PRINTER**
!!!"
348Ø DISPLAY AT(14,Ø):"THE L
ABELS LINED UP?   Y/N
                         I
F NO THEN WE'LL TEST ONE,
      ELSE WE GO!"
349Ø CALL KEY(3,K,S):: IF K=
89 THEN 351Ø ELSE IF K=78 TH
EN 35ØØ ELSE 349Ø
35ØØ M=N :: N=1
351Ø OPEN #1:"PIO",DISPLAY ,
OUTPUT
352Ø FOR I=1 TO N
353Ø FOR L=1 TO NL
354Ø PRINT #1:A$(I,L)
355Ø NEXT L
356Ø FOR CR=1 TO 6-NL
357Ø PRINT #1:CHR$(13)!BLANK
 LINE
358Ø NEXT CR
359Ø NEXT I
36ØØ CLOSE #1
361Ø IF K=78 THEN N=M :: CAL
L CLEAR :: GOTO 348Ø
362Ø GOTO 26Ø
363Ø !
364Ø ! **SEARCH THE LIST**
365Ø !
366Ø CALL CLEAR
367Ø IF FLAG=Ø THEN 422Ø
368Ø INPUT "ENTER LINE NUMBE
R TO SEARCH >  ":L
369Ø LINPUT "ENTER THE CUE W
ORD , MENU ENTER (M) >   ":Q$
37ØØ PRINT
371Ø IF Q$="M" THEN 26Ø
372Ø LQ=LEN(Q$)
373Ø FOR I=1 TO N
374Ø DISPLAY AT(2Ø,2)SIZE(3)
:N-I
375Ø LA=LEN(A$(I,L))
376Ø FOR J=1 TO LA+1-LQ
377Ø IF SEG$(A$(I,L),J,LQ)=Q
$ THEN 384Ø
378Ø NEXT J
379Ø NEXT I
38ØØ CALL CLEAR :: PRINT "NO
 MATCH FOUND > TRY AGAIN?(Y/
N) >"
381Ø CALL KEY(3,K,S)
382Ø IF S=Ø THEN 381Ø
383Ø IF K=89 THEN 366Ø ELSE
26Ø
384Ø CALL CLEAR
385Ø FOR L=1 TO NL
386Ø PRINT :: PRINT A$(I,L)
387Ø NEXT L
388Ø PRINT
389Ø PRINT "FIND ANOTHER";"
";Q$;" ?(Y/N) >"
39ØØ CALL KEY(3,K,S):: IF S=
Ø THEN 39ØØ ELSE IF K=89 THE
N 369Ø
391Ø CALL CLEAR :: PRINT "CO
NTINUE IN SEARCH MODE TYPE (
S),RETURN TO MENU TYPE (
M)"
392Ø CALL KEY(3,K,S):: IF S=
Ø THEN 392Ø
393Ø IF K=83 THEN 366Ø ELSE
IF K=77 THEN 26Ø ELSE 391Ø
394Ø END
395Ø !
396Ø !**CALATOLG DISK**
397Ø !
398Ø CALL CLEAR
399Ø PRINT "   WHAT DRIVE NU
MBER?       (HOLD SPACE TO S
TOP SCROLL)"
4ØØØ INPUT DN$ :: CALL CLEAR
4Ø1Ø TYPE$(1)="DIS/FIX"
4Ø2Ø TYPE$(2)="DIS/VAR"
4Ø3Ø TYPE$(3)="INT/FIX"
4Ø4Ø TYPE$(4)="INT/VAR"
4Ø5Ø TYPE$(5)="PROGRAM"
4Ø6Ø OPEN #1:"DSK"&DN$&".",I
NPUT ,RELATIVE,INTERNAL
4Ø7Ø INPUT #1:C$,J,J,K
4Ø8Ø PRINT "DSK";DN$;"- disk
name = ";C$;" available =";K
;" used =";J-K
4Ø9Ø PRINT " filename  size
   type    p"
41ØØ FOR LOOP=1 TO 127
411Ø INPUT #1:C$,A,J,K
412Ø IF LEN(C$)=Ø THEN 418Ø
413Ø PRINT C$;TAB(12);J;TAB(
17);TYPE$(ABS(A));
414Ø B$="  "&STR$(K)
415Ø PRINT SEG$(B$,LEN(B$)-2
,3);
416Ø CALL KEY(3,K,S):: IF K=
32 THEN 416Ø
417Ø NEXT LOOP
418Ø CLOSE #1
419Ø PRINT : :"PRESS (M) FOR
 MENU"
42ØØ CALL KEY(3,K,S):: IF S=
Ø THEN 42ØØ ELSE IF K=77 THE
N 26Ø
421Ø !
422Ø !  **FILE LOADED?**
423Ø !
424Ø DISPLAY AT(1Ø,Ø)ERASE A
LL:"   YOU MUST HAVE FIRST
     CREATED OR LOADED A
 FILE."
425Ø DISPLAY AT(16,Ø):" PRES
S (M)TO RETURN TO MENU"
426Ø CALL KEY(3,K,S):: IF S=
Ø THEN 4Ø5Ø ELSE IF K=77 THE
N 25Ø
```

Thats all folks, thanks for your patience. I recommend the following changes. Its nice and you may find use for it elsewhere.

```
198 FOR I=1 TO 8Ø
2ØØ CALL KEY(3,K,S)::IF S<>Ø
THEN 21Ø !Controllable wait
2Ø2 NEXT I
```

UGOC
BULLETIN
BOARD
3ØØ/12ØØ/24ØØ BAUD
(714) 751-4332

BASICALLY BASIC
by N.Armstrong

During the past year, I've acquired a few monitors and TVs, some ailing and some not. These become objects of the Garage SIG's scrutiny and ministrations. I wanted to check the linearity and convergence on the most recent arrivals, so I looked for but could not find a cross hatch program in the stack of disketts at hand. Hence, the following:

```
9Ø REM SAVE DSK1.TVCON
1ØØ DATA "for linearity/conv
ergence checks",3,"  to cha
nge background color",21,"
      press keys b through p"
,22
11Ø CALL CLEAR
12Ø CALL SCREEN(5)
13Ø CALL COLOR(5,16,2)
14Ø CALL CHAR(65,"181818FFFF
181818")
15Ø CALL CHAR(32,"ØØØØØØ1818
")
16Ø FOR I=1 TO 2Ø STEP 6
17Ø FOR J=1 TO 3Ø STEP 8
18Ø CALL HCHAR(I,J,65)
19Ø CALL HCHAR(I,J+7,65)
2ØØ CALL HCHAR(I+5,J,65)
21Ø CALL HCHAR(I+5,J+7,65)
22Ø NEXT J
23Ø NEXT I
24Ø GOSUB 32Ø
25Ø GOSUB 32Ø
26Ø GOSUB 32Ø
27Ø CALL KEY(3,K,K)
28Ø IF K<>13 THEN 29Ø
285 STOP
29Ø IF (K<66)+(K>8Ø)THEN 27Ø
3ØØ CALL SCREEN(K-64)
31Ø GOTO 27Ø
32Ø READ A$,B
33Ø FOR I=1 TO LEN(A$)
34Ø A=ASC(SEG$(A$,I,1))
35Ø CALL HCHAR(B,I,A)
36Ø NEXT I
37Ø RETURN
```

Statement 9Ø is a reminder of the program name and where to store it. If I don't put in a remark like this, I end up with unfinished programs filed under different names on at least two disks. Also, because I write in the Extended Basic environment, this command does not have to be retyped each time I want to save the program.

The DATA statement (1ØØ) contains two types of information: a string to be displayed and the number of the line on which to display it. Simple routines can be made very versatile by including operating parameters in DATA statements.

These five CALL statements (11Ø-15Ø) initialize the display parameters: the screen is cleared and colored dark blue; a white foreground, black background is ordered for character set 5 which contains ASCII 65. ASCII 65 and 32 are redefined as targets for the linearity/convergence check.

These are the statements (16Ø-26Ø, 32Ø-37Ø) that put the data on the screen. They can be divided into three groups: the nested loop, the CALL Horizontal, and the subroutine.

A nested-loop is easy to understand; an inside loop is counted completely for each count in the next outside loop. Because step amounts are specified, the counts will not be consecutive; for I the sequence will be 1, 7, 13, and 19. For J it will be 1, 9, 17, and 25.

The Call Hchar command allows placement of any character at any position on the screen. Three variables are required, row, column, character code. Using the nested-loop counts with selected constants for the row/column values places the 65 target in the screen locations strategic to dynamic convergence. A fourth variable, when supplied, designates number of repititions (along the horizontal axis; a Call Vchar command would repeat along the vertical axis).

The subroutine is used to put the Data Statement information onto the screen. There are three sets of data so GoSub is used three times. Identical results could have been achieved with other programming techniques, i.e., nested loops.

The Read statement gets display (A$) and location (B) data. Display data is transfered to the screen, letter by letter in a For/Next loop. The number of iterations through the loop is determined right in the loop, LEN(A$). In each iteration through the loop, the ASCII number for the Ith character in the string is determined, A=ASC(SEG$(A$,I,1)). Then the Call Hchar statement is used to place character (A) at row (B), column (I).

On returning from the third trip through the subroutine, the program goes into a holding pattern (27Ø-31Ø). If <ENTER> is pressed, the program stops; if B through P (representing 2 through 16) is pressed, the screen will change to the color represented by that code. The screen will not blank out. and thats about it. >>>BYE NEWT<<<

If it's low-cost computing you want,
For spreadsheet wordprocess or font,
Play games on the screen,
Bring "Scotty" down on a beam,
Get a TI, your idiot savant. --N.A.

ASSEMBLY LANGUAGE
by Adrian Robinson

MORE BOOT TRACKING:

After submitting last month's article to the ROM, I had an afterthought. It was too late to include last month so I will continue it here. The afterthought was to allow XBasic types to do boot drive tracking without even getting close to the Editor/Assembler. This can be done by using the following XBasic CALL LOAD program:

```
      10 ! XBASIC DRIVE TRACKING
           by Adrian Robinson
      15 CALL INIT :: CALL CLEAR
      20 DATA 194,139,2,0,62,235,4
      ,32,32,40,2,33,48,0,216,1,38
      ,0,4,224,131,124,4,90
      25 DATA 68,82,73,86,69,32,36
      ,244
      30 FOR A=9460 TO 9483 :: REA
      D N :: CALL LOAD(A,N):: NEXT
      A
      35 FOR A=16376 TO 16383 :: R
      EAD N :: CALL LOAD(A,N):: NE
      XT A
      40 CALL LOAD(8196,63,248)
      45 !************************
      50 CALL LINK("DRIVE"):: CALL
      PEEK(9728,A):: D$=CHR$(A)
      55 PRINT "DRIVE NUMBER ";D$
```

I cannot, however, simply present a CALL LOAD program without discussing both its application and its roots. The program can be divided into two parts. Lines 15 thru 40 load a simple machine language program into LOWMEM. Line 50 executes the program and obtains its result, the number of the last drive accessed, in A. This can be used in a variety of ways. The first part could be included at the beginning of your XB programs or merged in after being saved in merge format. The CALL LINK and CALL PEEK could then be executed any time thereafter. As an alternative, lines 15 thru 40 can be run separately before running other XBasic programs. The machine code will remain in memory as long as another CALL INIT is not executed. The CALL LINK and CALL PEEK can then be done anywhere in your following XBasic programs.

Now let us talk about the source of the data for the CALL LOAD program. First we start with a simple assembly program to find the drive number:

```
         DEF   DRIVE
DRIVE    MOV   R11,R10
         LI    R0,>3EEB
         BLWP  @>2028
         AI    R1,>3000
         MOVB  R1,@>2600
         CLR   @>837C
         B     *R10
         END
```

Then we assemble with the LIST option which results in the following LIST file:

```
 0001                          DEF  DRIVE
 0002 0000 C28B  DRIVE  MOV  R11,R10
 0003 0002 0200         LI   R0,>3EEB
      0004 3EEB
 0004 0006 0420         BLWP @>2028
      0008 2028
 0005 000A 0221         AI   R1,>3000
      000C 3000
 0006 000E D801         MOVB R1,@>2600
      0010 2600
 0007 0012 04E0         CLR  @>837C
      0014 837C
 0008 0016 045A         B    *R10
 0009                   END
 0000 ERRORS
```

The third column here lists the machine code(hex) corresponding to the assembly instructions. Converting the hex values to decimal byte values provides the data for line 20 above(eg, C28B becomes 194, 139). Line 30 loads the machine code to low memory at address >24F4 (9460). Line 35 loads the DEF table entry from line 25 for CALL LINK and line 40 sets the LFAL(start address of DEF table).

The assembly routine stores the drive number, in ASCII, at the arbitrarily chosen address >2600 (9728) where it can be accessed with CALL PEEK.

AND STILL MORE:

I have discovered that I have at least one reader out there. Following last month's article, I had a phone call from Gary Sweers of Florida with a question about boot drive tracking. Since it may be of interest to others, I will repeat here what I told him.

Gary has an E/A program saved in XBasic with Todd Kaplan's ALSAVE program, an XB assembly loader. On return from his E/A program, he runs an XB program. His program would look something like this:

```
10 CALL INIT :: CALL LOAD(8196,63,248):
: CALL LOAD(16376 ,65,32,32,32,32,32,25
5,48):: CALL LINK("A")
100 CALL LINK("EAPROGRAM"):: GOTO 120
110 ! RUN must be in LAST program line
120 RUN "DSKx.XBPROGRAM"
```

Now he wants to do boot drive tracking in his assembly program and wanted to know how to pass the drive number to his XB RUN statement. My letter to him described the following

In your assembly program, include the following routine with the high byte of R5 containing the drive number in ASCII code.(eg, >3300)

BASICALLY BASIC
by N.Armstrong

During the past year, I've acquired a few monitors and TVs, some ailing and some not. These become objects of the Garage SIG's scrutiny and ministrations. I wanted to check the linearity and convergence on the most recent arrivals, so I looked for but could not find a cross hatch program in the stack of disketts at hand. Hence, the following:

```
9Ø REM SAVE DSK1.TVCON
1ØØ DATA "for linearity/conv
ergence checks",3,"  to cha
nge background color",21,"
     press keys b through p"
,22
11Ø CALL CLEAR
12Ø CALL SCREEN(5)
13Ø CALL COLOR(5,16,2)
14Ø CALL CHAR(65,"181818FFFF
181818")
15Ø CALL CHAR(32,"ØØØØØØ1818
")
16Ø FOR I=1 TO 2Ø STEP 6
17Ø FOR J=1 TO 3Ø STEP 8
18Ø CALL HCHAR(I,J,65)
19Ø CALL HCHAR(I,J+7,65)
2ØØ CALL HCHAR(I+5,J,65)
21Ø CALL HCHAR(I+5,J+7,65)
22Ø NEXT J
23Ø NEXT I
24Ø GOSUB 32Ø
25Ø GOSUB 32Ø
26Ø GOSUB 32Ø
27Ø CALL KEY(3,K,K)
28Ø IF K<>13 THEN 29Ø
285 STOP
29Ø IF (K<66)+(K>8Ø)THEN 27Ø
3ØØ CALL SCREEN(K-64)
31Ø GOTO 27Ø
32Ø READ A$,B
33Ø FOR I=1 TO LEN(A$)
34Ø A=ASC(SEG$(A$,I,1))
35Ø CALL HCHAR(B,I,A)
36Ø NEXT I
37Ø RETURN
```

Statement 9Ø is a reminder of the program name and where to store it. If I don't put in a remark like this, I end up with unfinished programs filed under different names on at least two disks. Also, because I write in the Extended Basic environment, this command does not have to be retyped each time I want to save the program.

The DATA statement (1ØØ) contains two types of information: a string to be displayed and the number of the line on which to display it. Simple routines can be made very versatile by including operating parameters in DATA statements.

These five CALL statements (11Ø-15Ø) initialize the display parameters: the screen is cleared and colored dark blue; a white foreground, black background is ordered for character set 5 which contains ASCII 65. ASCII 65 and 32 are redefined as targets for the linearity/ convergence check.

These are the statements (16Ø-26Ø, 32Ø-37Ø) that put the data on the screen. They can be divided into three groups: the nested loop, the CALL Horizontal, and the subroutine.

A nested-loop is easy to understand; an inside loop is counted completely for each count in the next outside loop. Because step amounts are specified, the counts will not be consecutive; for I the sequence will be 1, 7, 13, and 19. For J it will be 1, 9, 17, and 25.

The Call Hchar command allows placement of any character at any position on the screen. Three variables are required, row, column, character code. Using the nested-loop counts with selected constants for the row/column values places the 65 target in the screen locations strategic to dynamic convergence. A fourth variable, when supplied, designates number of repititions (along the horizontal axis; a Call Vchar command would repeat along the vertical axis).

The subroutine is used to put the Data Statement information onto the screen. There are three sets of data so GoSub is used three times. Identical results could have been achieved with other programming techniques, i.e., nested loops.

The Read statement gets display (A$) and location (B) data. Display data is transfered to the screen, letter by letter in a For/Next loop. The number of iterations through the loop is determined right in the loop, LEN(A$). In each iteration through the loop, the ASCII number for the Ith character in the string is determined, A=ASC(SEG$(A$,I,1)). Then the Call Hchar statement is used to place character (A) at row (B), column (I).

On returning from the third trip through the subroutine, the program goes into a holding pattern (27Ø-31Ø). If <ENTER> is pressed, the program stops; if B through P (representing 2 through 16) is pressed, the screen will change to the color represented by that code. The screen will not blank out. and thats about it. >>>BYE NEWT<<<

If it's low-cost computing you want,
For spreadsheet wordprocess or font,
Play games on the screen,
Bring "Scotty" down on a beam,
Get a TI, your idiot savant. --N.A.

```
        MOV  @>8330,R2   Line Table ADR
        INCT R2     Point to Pgm Line ADR
        MOV  *R2,R2      Get Pgm Line ADR
        AI   R2,6     Point to "x" in RUN
        MOVB R5,*R2  Replace "x" w/Drv #
```

That's all there is to it. You have found the RUN statement in HIMEM and "poked" the drive number into it.

Now that we have come this far, let us suppose that you have an assembly program that can select an XB program from a list or menu. You can also "poke" the program name into the RUN statement. Suppose the name is stored in:

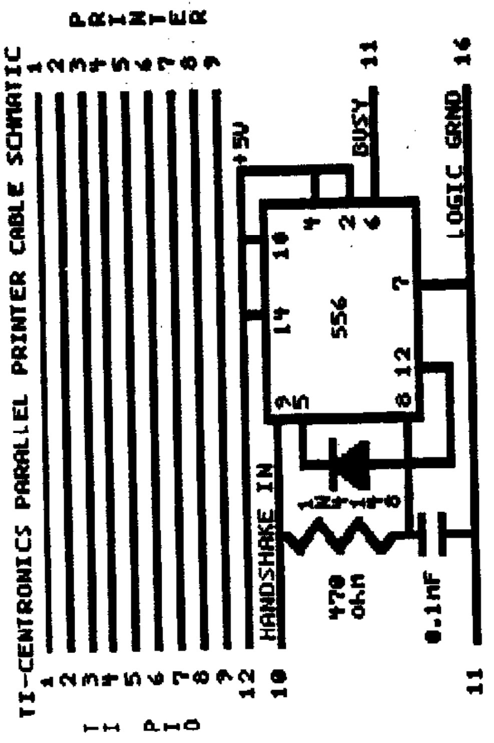
```
PNAM   TEXT 'CHOSENPGM '
```

and its length is in R3. At the end of the above routine, R2 points to the drive number, so just append the following routine to it:

```
        INCT R2             Skip over "."
        LI   R1,PNAM
        MOVB *R1+,*R2+  Poke PNAM to RUN
        DEC  R3
        JNE  $-4
        MOVB R3,*R2      (R3=0) Ends line
```

and you have reconstructed the RUN statement.

KEEP ON TRACKIN'.



DIPS and CHIPS
by Siles Bazerman

A new 80 column program has arrived from Germany. It is XHi, and is a high resolution graphics program. It is a memory resident vehicle using "CALL LOAD"s to do a number of interesting graphics routines. All documentation is in English as are the included demonstration programs. It runs on the Geneive and the Dijit AVPC. It should also run on the Megatronics card. Any comment, Jerry?

We are still awaiting the latest release from the McGoverns. This was released last month at the Lima, Ohio TI fair. It is reported to be a full 80 column version of Funnelweb. I will review it when I have had a chance to try it.

The smoke you see rising over Orange County is from three new EPROM burners working overtime. Some interesting results have been obtained including modulizing several EA5 programs. Anyone with spare Navaronne modules, please contact me, Jerry or Ben. Sometime I will have to show my Corcomp Disk Controller modified with modified Miller EPROMs. Does this make it FORTH party cards, or did I leave you in the DARK?

If you are looking for an other word processing program check out RAGTIW. It is an enhanced TIW (disk based) with several new commands in the Editor mode, and six new Formatter commands. I wish it supported 80 columns. It looks that good, even though it does not support the file marking or mailbox features of FWeb.

COMPRODINE has several new releases that are excelent. There is a new update of JIFFY CARD, with improved documentation well worth getting.See Roger Merritt for this. Also is a new COLOR version for the Star NX1000 Rainbow printer. Now you can have cards in multi -color. Excelent job by Roger and Robby (Adrian Robinson).

If you like to create posters and BIG banners, then you need Giant Artist Posters by Paul Coleman. This is also distributed by COMPRODINE.

Speaking of the Rainbow printer; Jerry Rash just bought one. He found that for the Corcomp RS232 card, the stock EPROM works fine. With the TI RS232 card you will need a new EPROM available on an exchange basis from STAR MICRONICS. I wonder what the difference between the two RS232's is. Also some differences between the PIO printer cables have been noted for these cards.

DARK THEORY CHALLENGED

By Earl Raguse with J Newton Armstrong

I was completely amazed at the chaotic furor caused by my dark thesis in the April ROM. It seemed to arouse more skepticism than the room temperature (con)fusion experiment by the University of Utah professors Pons and Fleischmann. DARK has been challenged on all fronts. But, with the able assistance of my scientific advisor, J. Newton (Isaac?) Armstrong, I shall attempt to allay the concerns of the doubters and unbelievers, in question and answer format.

Q) By sagacious high school student.
"You state dark cannot penetrate solid objects. How then do you explain dark leaking out through window glass?"

A) Glass is not a solid in the normal sense. Glass is a non-crystaline fusion of silica. It may be viewed as a highly viscous fluid or jell. Glass continues to flow under the influence of gravity forever. Panes of glass in old cathedrals tend to be twice as thick at bottom as at the top.

This is a good point however, and proves that young people still think. I should have said opaque solids, there are in fact transparent solids and liquids to be discussed later.

Q) By mature doubter and old skeptic.
"How do you account for dark printing in the newsletter etc. How come that isn't sucked up?"

A) You bring up a good point, but that is the problem when one oversimplies for a audience of limited inteligence. We had not yet gotten around to the more complex subject of adhesives and stuff with strong affinities for dark. One of these substances is printers ink. Others are black acrylic paint, graphite and carbon black etc.

These aren't perfect however, leave them exposed to the dark sucking action of the sun for a few decades and they will fade to light black.

Q) By my barber, who claims to be an MIT physics professor on sabatical.
"You state that dark travels 186,ØØØ miles/second, that may be true in a vacuum, but in your example there was an atmosphere and hence it probably would take 24 picoseconds for dark to travel from the corner to the light sucker."

A) I was forced to agree with him about the atmosphere slowing down dark, but I carefully pointed out to him that the distance was only 22.67 feet so that the 23 picoseconds was rounded up. Precise calculation allowing for atmospheric slowing showed the time was still slighly less than 23 picoseconds. BOY was he embarrased!

Q) By interested non-bright citizen.
"What happens to all the dark that the sun sucks up?"

A) Dark mediates the hydrogen nuclear reactions in the sun and is consummed in the process. That is where the sun gets the energy to radiate so much heat and ultra violet rays etc.

Q) By innocent bystander.
"How do you explain eclipses of the sun?"

A) When an opaque (note that I said it) body like the moon gets between the sun and the earth, it blocks the path of dark so the sun can't suck it up, so we are all in the dark just like you are. The sun is not really affected, it just looks that way. Actually, I suppose the temporary small loss of dark from the earth may have a barely measurable effect on energy production.

Q) By a rather smart friend of mine.
"Light energy is transmitted by photons, how is dark transmitted?"

A) This brings up the real controvercy of dark theory. It almost exactly analogous to the problem with currently used electrical current theory suggested by Ben Franklin before we knew about electrons. He said electrical currents (or was that raisins) flowed from positive to negative.

We now know, course, that "current" is carried by electrons and flows in the opposite direction. Engineers and scientists still use the outmoded current theory because it works even if it is backward. What the heck, we are still using feet and inches yet!

Light theory is the same way, it is backwards but the theory works to solve problems. The truth is that dark is transmitted by Darkons. I know at least one physicist that suggests they be called Photoffs. However, he admits to writing assembly language programs.

Q) By me, in a pensive moment.
"Now that we know dark is transmitted by darkons, and we mentioned transparent solids, how are these connected?"

A) Firstly, dark acts as though it is both a wave and a particle phenomena. When considering transmission through transparent solids and liquids, wave theory fits best.

Dark travels at different speeds in different media. It travels faster in air than it does in water, cold beer or quartz crystals. According to Fermat's principle, wave propagation through a collection of media, takes the path that requires least time. By Snell's law the dark rays are bent (we scientific types say "refracted") toward the slower media That's why they call them media, because the dark is slowed by a medium amount.