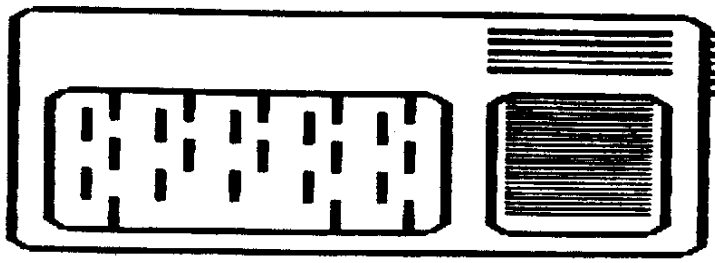
MARK THIS DATE: MON. OCT. 20, DEPTFORD - MYARC 'S LOU PHILLIPS AND GENEVE!



THE DATA BUS

Issue : VOL. 4 NO. 7 - - - - AUG. 1986

THE DELAWARE VALLEY USERS GROUP
P.O. BOX 6240 STANTON BRANCH, WILMINGTON DE 19804
DEDICATED TO THE TI AND COMPATIBLE HOME COMPUTER FAMILY
MEETS 4TH THURS. EACH MONTH/6:30-9:30 P.M./CHRISTIANA MALL COMMUNITY ROOM
KENT COUNTY, DE COURTHOUSE <CHAPTER> DEPTFORD, NJ MUNICIPAL BLDG.
2ND THURS. 7:00-9:00 P.M. <MEETING> 3RD MONDAYS 6:45-9:00 P.M.

WELCOME NEW MEMBERS - WE NOW TOTAL 134 DVUG HOUSEHOLDS:

DELAWARE: Ed Levin, Hans Borchardt MARYLAND: K.Gary Letts
NEW JERSEY: Brian Beckett, David Hauss

and our first members from PUERTO RICO:
Gloria Escobar, and husband Edwin Colon Skerrette

Welcome Back to Bill Anthos, rejoining us, as do nine other renewals this past month.

August Computer Events Calendar for the Area:

DELMARVA CHAPTER (DOVER), THURSDAY,AUGUST 14

SOUTH JERSEY CHAPTER (DEPTFORD), MONDAY, AUGUST 18

Tidewater 11th Annual Computer Fair & Hamfest, VIRGINIA BEACH,VA August 23-24, at the Pavillion. For Info: Manny Steiner, (804) 340-6185.

Delaware Tech. & Comm. College Annual Computer Fair, Aug.24 (Georgetown Campus) - For info, call the school.

CHRISTIANA MEETING, THURSDAY, AUGUST 28, 6:30 p.m.

Red Rose 4th Annual Computerfest, Guernsey Sales Pavillion, LANCASTER, PA August 31. Info: P.O. Box 5029, Lancaster,PA 17601.

New personnel will be assuming DVUG responsibility for several areas starting in September including THE DATA BUS Editor; see DVUG Doings inside. Brian Beckett will be the So. Jersey BBS SYSOP, as soon as we can get it on line.

CONTENTS OF THE AUGUST ISSUE OF THE DATA BUS:

NEWS BYTES . . .

IF YOU DON'T SUCCEED AT FIRST (OR, TI DOES IT AGAIN:

While maintaining a 1-year inventory of parts for its current TI Pro (PC) users, TI is ending its independent TI-DOS configuration to go strictly IBM-compatible, orphaning current TI Pro users. Completing the picture, Pro software will no longer carry TI's label, but will be identified as a 3rd-party product. TI needed to join the Big Blue clones to continue to have a competitive marketable product. ...

99/4A IBM keyboards (numeric key pad, separate Control and Function keys, etc.) have been produced by members of the Nutmeg TI 99ers, and are available from RAVE 99 Co., 23 Florence St., Bloomfield CT 06002. Call (203) 242-4012, or 872-9272 after 6 p.m. for descriptive brochure. 84-keys are $124.95; 101-key board, $149.95; custom keys $15 extra plus shipping. NO P-BOX NEEDED! ...

Richard Bailey of NH 99'ers will make no more Fairware since he finally got ONE donation for his TINYCAL, running on (DVUG and other) East Coast TIBBS and elsewhere for over 8 months. Meanwhile, TI yielded and will ship anyone a FREE copy of the TI-WRITER Manual for only $3 shipping after you call (800) TI-CARES. Funlwriter and TK-Writer owners, take note: No pay for Fairware, no products likely tomorrow. ...

If you WILL pay for excellent, inexpensive products, a $1 mailing for a catalog will obtain you a list of the many TAPE and DISK programs which are available for $3 each (if you send a blank tape/disk and mailer) from program author Cheryl Whitelaw, whom we came to know by her middle name of REGENA, while she wrote/edited for 99er Magazine, and later with COMPUTE! Drop a line - at least a thank-you ! - to: Regena, P.O. Box 1582, Cedar City UT 84720. Again, please remember, she's only one person, and you are many, when you await her response. ...

Great Lakes Software's JOY PAINT is compatible with an Epson, Gemini 10X and Axiom GP100TI printer, NOT Axiom's GP 100TI2, which is a 7-bit, not 8-bit graphics, printer. ...

On the Avatex modem, exchange pin locations 2 and 3 as well as locations 6 and 20, for 99/4A usage at 300/1200 Bd.

DELAWARE VALLEY USERS GROUP: AUG. 1986

DVUG EXECUTIVE COMMITTEE MEMBERS IN 1986

PRESIDENT ..... TOM AUGUST
VICE PRES ..... JIM DAVIS
SECRETARY ..... LYNN ACQUARD
TREASURER ..... TOM KLEIN
SGT.AT ARMS ..... JIM FOLZ
DELMARVA CHR ..... CHARLES BOWER
SO.JERSEY CHR ..... ERROL LANSBERRY

TIBBS (24 hrs. 300/1200): 302-322-3999
Our new So. Jersey TIBBS at 609-429-9348 is not online yet due to system changes.

SOFTWARE CHR: JACK SHATTUCK 302-764-8619

A Delaware Valley Users Group membership includes monthly newsletter, library and software privileges, plus other special benefits. Annual membership rates are: Family or Individual $15; Students $5; Newsletter only (beyond 75 miles) - $10

PLEASE TRANSMIT YOUR NEWSLETTER COPY TO: The Data Bus Editor -- JIM FOLZ. Call at Telephone: 302-995-6848, or use the DVUG mailing address as is shown on Page One. NEWSLETTER COPY WILL NOT BE ACCEPTED FOR AN ISSUE AFTER THE 2ND THURS. EACH MONTH

Advertising Rates in The Data Bus:
1/4 page = $ 5/issue, or $ 45/12 issues
1/2 page = $ 8/issue, or $ 75/12 issues
Full page = $15/issue, or $125/12 issues

DELAWARE VALLEY USERS GROUP LOCATIONS:
Plenary meetings: Delaware's Christiana Mall on Rte. 7, at I-95 Exit 4-S, in the Community Room. Enter between J.C.Penney and Liberty Travel inside the Mall. Call Tom Klein, 215-494-1372 or others above.

DELMARVA CHAPTER: Kent County Courthouse, Basement Conference Room #25, The Green & State Street, Dover, Delaware. Use entrance on The Green side. Contact: Jim England, 302-674-9256.

SO.JERSEY CHAPTER: Deptford Municipal Building, Cooper Ave. and Delsea Drive, (Rtes. 534 & 47), in Gloucester County. Enter and park in rear of the building. Contact: Carol Rosowski, 609-228-2445.

OFFICERS for the Delmarva Chapter of the Delaware Valley 99/4A Users Group (formerly Kent County 99/4A Users Group) are, left to right, Chuck Bower, president; Jim England, treasurer; Ken Ayers, vice president; and Kay Quillen, secretary. The group, which includes people interested in working with Texas Instrument 99/4A computers, meets 7 p.m. the second Thursday of each month in the Kent County Courthouse. Other enthusiasts are welcome.

 (Reproduction here was distorted due to size, but the Post's coverage was good!)

Recent DVUG DOINGS

DELMARVA MEETING (Aug. 14) - Kay Quillen

BBS & MODEMS:

Chairman Chuck Bower demonstrated a 1200 Baud Avatex modem, using TEII, 4/A-Talk and FAST-TERM.

At present, a TIBBS BBS program is being updated by DVUG and a cable change is being prepared to bring the DelMarVa BBS on line. However, Jim England can't host the BBS; we need another volunteer.

It is not clear how much demand now exists for a DelMarVa BBS: A VOTE WILL BE TAKEN AT THE SEPTEMBER MEETING. (Any Chapter members who can't make it should express themselves to an officer.)

FUNDRAISING:

The opportunity for a $5 rebate for DelMarVa members' DVUG memberships ($75, at present) was noted to be still under negotiation. [Ed. note - All DelMarVa members are encouraged to attend DVUG's Christiana meetings to discuss special circumstances.) Another potential member was welcomed to the August gathering.

DelMarVa or other DVUG members may still take chances on a DelMarVa raffle [see June's DATA BUS, p. 3), at $3 for members, $6 for non-members.

The yard sale was shifted to Rita Locey's house. [It was a slow sales day but members provided many donations.)

Fellow DVUG member Don Newsom has donated a full-height SS/SD disk drive to the Chapter, which is appreciated.

FUTURE ACTIVITY:

Storer Cable TV offers a $.50/month charge for announcements; details about frequency, etc., needed clarification; also whether NBOC (Channel 16) will make free Public Service Announcement (PSA's) if we advertise for charge (fifty cents) on Storer Cable.

At Sept. 11th meeting, Chuck Bower will demo the CorComp peripherals, plus Millers Graphics' Advanced Diagnostics.

CHRISTIANA MINUTES (July 26) - Jim Davis, Scribe

President Tom August opened with the Treasurer's report. Tom Klein reported a balance of $833.

With regret, we received the resignation of our newsletter editor. Much praise is due Jack Shattuck for his impressive accomplishments with the newsletter. Fortunately, he will be available for consultation. Jim Folz was appointed editor, beginning with the September issue.

The Bulletin Board committee reports purchase of a CorComp double density disk controller. The system is up and includes new software. The committee will publish a primer on "how to use". Meanwhile, the HELP FILE is more useful than previously. The Xmodem file transfers (FAST TERM) are faster than TEII and their use is encouraged. Please use proper logoff so that the Board is not locked up.

John Webb resigned as Software Librarian and also as Refreshment Chair. Jack Shattuck accepted the Chair as Librarian. We need a volunteer for the Refreshment Chair.

The Mancus Foundation, at 2308 Washington Street, Wilmington, sent their thanks for computer donations [- more on this in a future issue: Ed.]. Jim England won the TV raffle which benefitted the Bulletin Board, and Chuck Bower really struck it rich, winning both the 50/50 and the Gram Cracker.

As many of our old-timers know, I've been active with DVUG since 1982, when we were still meeting in DuPont's shop on Greenhill Avenue, before DVUG was formally organized.

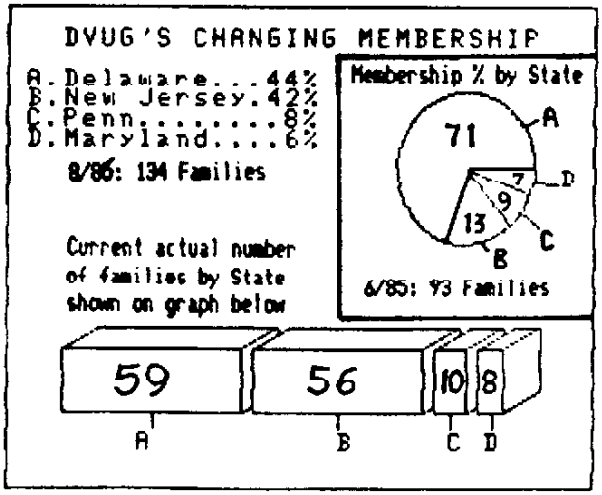
Since Jack Thorpe turned THE DATA BUS over to me, after the last issue of 1984, it has been 20 months of deadlines. My family wants some fun too. Thus do I pass on THE DATA BUS to Jim (& Pamela) Folz AND TO THE REST OF US ! to take over. I'll be in Buffalo,NY, during August - see you in September!

Jack Thorpe's revised Radio Shack program permitted our two-column setup in Pica, which we then revised in Elite, to use until George Steffan's MULTIPRINT allowed convenient use of parallel condensed columns, as well. Tom August nudged us into a little experimentation this past Spring - using both photo reduction and tightened line-spacing, to increase our printed output capacity. (Over 10 pages increases postage.)

The compressed line space method was new to me. Using that technique (available on most printers for TI), we can increase from 60 to 80 lines per page (e.g., see p. 6-9). That's about as fine as you can get and still be legible. I have tortured, or pleased, you that way for several issues.

Reliability of a 10-page newsletter, month after month, season after season, has been a matter of pride I'm pleased to have shared with you. DVUG has grown incredibly, despite three years' passage since the last TI-99/4A was produced; a result of much hard work recruiting users, especially in the South Jersey area. See the DVUG graph; glad I'm part of it.

Now I'll be picking up sofware library chores - work to be done, but with fewer time constraints, to continue to see DVUG succeed in its mission to our members. (I will provide columns occasionally, too.) Special thanks go to Pat Fretz, among my many DVUG (and nationwide) TI friends in almost 200 pages of THE DATA BUS I've shared, as Editor JACK SHATTUCK.



PAPERBACK OLDIES-BUT-GOODIES AND A NEW PRINTER REFERENCE TEXT

Reviewed by The Data Bus Editor:

COMPUTE!'s moved its main public contact point to the 6th Floor, at 825 7th Avenue, New York, NY 10019.

Their TI paperbacks still exist. ISBN numbers allow the user to place orders at any bookstore, or call them using (800)346-6767 or (212) 887-8525; Dealer bulk orders from (800)638-3822.

Only their last published volume, TI Collection Vol. Two, comes on disk, but you'll learn from studying text comments as you type in programs. Many programs need the text documentation.

Programmer's Reference Guide to the TI -99/4A, by Regena, ISBN 0-942386-12-4 $14.95

Here's the comprehensive TI BASIC introductory programmer reference that TI should have put out. Doesn't get to more involved subjects, like creation of files, but shows excellently how to use many TI BASIC commands. 312 Pages.

33 Programs for the TI-99/4A $12.95
By Brian Flynn, ISBN 0-942386-42-6

A variety of very basic, standard computer utility math routines, plus a few games. Probably the least sought of the TI volumes. TI BASIC,199 Pages.

Creating Arcade Games on the TI-99/4A Seth McEvoy, ISBN 0-942386-27-2 $12.95

13 of 14 chapters are for BASIC, with 8 type-in games; one chapter uses sprites in a game. CHAR designs, key and joystick control, scrolling, sound and other basic techniques. 200 Pages.

TI Games for Kids $12.95
Robert P. Ingalls ISBN 0-942386-39-6

32 games, from pre-school to high school level, that teach and entertain at the same time. TI BASIC. 188 Pages.

COMPUTE!'s First Book of TI Games, Ed. by C.Regena, ISBN 0-942386-17-5 $12.95

Maze, chase, creative, favorite, you-name-it variety in 38 games, with 7 in Extended BASIC including a lively Hangman, "Mystery Spell". 211 Pages.

COMPUTE!'s Guide to TI/99/4A Sound and Graphics, Raymond J. Herold, $12.95 ISBN 0-942386-46-9

Sprites in detail, sound, graphs, 3-D, CHARs and even speech. Examples and complete games. 210 Pages.

COMPUTE!'s Guide to Extended Basic Home Applications on the TI-99/4A, by Christopher Flynn ISBN 0-942386-41-8 $12.95

File management, spreadsheet, an appointment calendar, card file, plus several bar/chart graphs and a system LOAD program. 199 Pages.

COMPUTE!'s TI Collection, Volume One, ISBN 0-942386-71-X $12.95

Utilities, program tips, music, games, spreadsheets, a mini-data base, word processing, print fonts, etc.etc. Both BASIC and XBasic. 389 Pages.

COMPUTE!'s Beginner's Guide to Assembly Language on the TI-99/4A, by Peter M.L. Lottrup ISBN 0-942386-74-4 $14.95

Useful for Editor/Assembler after learning basic Assembly principles but THIS requires MiniMem cartridge, Line-by-line Assembler, plus tape recorder. Step-by-step with line-by-line editing in a highly praised tutorial. 262 Pgs.

COMPUTE!'s TI Collection, Volume Two, ISBN 0-87455-036-X $14.95 247 Pages.

Picks up with additional tutorial articles, games, programs. For $12.95 instead, get a program disk (36XBDSK) - but you'll want the documentation.

For the novice who just acquired a Dot Matrix printer, and also for a more experienced owner who wants to convert software programs among the TI printers most popularly in use:

MINUTE MANUAL FOR THE DOT MATRIX PRINTER, Minute Mare, P. O. Box 2392, Columbia,MD 21045, ISBN 0-913131-04-0, 166 pp., by Jim Pirisino, has 8 quite readable and interesting chapters plus a hexadecimal listing, and index.

This cottage production explains how dot matrix features work, in clear detail. The different fonts, widths, alternate character sets, speed, form handling, buffers, interfaces, ribbons and dip switches are all discussed for six popular printers.

These are the Epson FX and RX, Gemini 10X, C.Itoh Prowriter and the NEC 8023, the Okidata 92 and the Apple DMP and Imagewriter.

The author, in clear prose and by invaluable chart form, distinguishes one version or brand from another with advantages, disadvantages and unique features of each.

Example of their print styles are shown, plus a thorough listing of all the Escape codes for these models. If you can OPEN #1:"PIO" or "RS232", and can type PRINT #1:CHR$(27); etc., off you go with the TI. You'll even find a reference or two for the Axiom GP100 printer (but in passing only).

An intriguing hint that Prowriter users might be able to custom design a character set came to naught without a memory addition. It was not available for the NEC, which the author says is also C.Itoh-manufactured (although the companies insist on having a separate identity). One more distinction.

Also, for persons not interested in remembering Escape codes, author Pirisino had originally offered a No-Programming Solution, which was to be a PRINTER COMMAND DISK, that he would sell with the relevant codes for your printer/computer setup. In the face of light market demand, that project was discarded. (Most users rapidly develop such a program themselves - see DVUG's library for PRINTMODE for the Epson, Gemini and NEC/Prowriter codes, run in TI BASIC.)

This valuable guide has a $12.95 cover price but will be sold for $9.75 if you buy from the author, mentioning THE DELAWARE VALLEY USER GROUP. There may be a postage charge; I bought mine at McMahon's bookstore (last copy)!

```
      TITL 'your heading for listing '
* DSK2.sname
*****************************************
*                                       *
*                                       *
*            program                    *
*              name                     *
*                                       *
*                                       *
*            PROGRAM                    *
*           WRITTEN BY                  *
*            your name                  *
*                                       *
*                                       *
*         DATE ASSEMBLED:               *
*      TEXT  'mm/dd/yy'                 *
*                                       *
*                                       *
*            VERSION 1.0                *
*                                       *
*                                       *
*    USES    DSK2.sname     ALL COPIES  *
*   SKELA                   MAIN VARS   *
* DISK LABEL                INPT RTNS   *
* STRUCTURE  DSK2.obj       OBJECT      *
*            DSK2.listing   LISTING     *
*****************************************
      DEF  entry
      REF  VMBW,KSCAN
      COPY "DSK2.SKELA1"
BLCNTS DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
BLCEND EQU  $
rout1# EQU  0           * rout1
rout2# EQU  2           * rout2
rout3# EQU  4           * rout3
*   OTHER DATA DEFINITIONS FOLLOW
*
      PAGE
entry EQU  $
      COPY "DSK2.SKELA2"
*   MAIN ROUTINE CODE FOLLOWS
*
      LI   RTN,rout1    * CALL
      BL   @BLROUT      *   rout1
*
*
      LI   RTN,rout2     CALL
      BL   @BLROUT          rout2
*
*
*
      B    @RETSYS      * RETURN TO SYSTEM
*
      COPY "DSK2.SKELA3"
*   EXAMPLE OF USERS LEVEL 2 ROUTINE
*****************
*   DESCR rout1 *
*****************
rout1  DATA BLCNTS+rout1#    ADDR OF COUNT EXECUTES
*
      LI   RTN,rout2    * CALL (NESTED) rout2
      BL   @BLROUT      *   rout2
*
*****************
*   DESCR rout2 *
*****************
rout2  DATA BLCNTS+rout2#    ADDR OF COUNT EXECUTES
*
      LI   RTN,rout3    * CALL (NESTED) rout3
      BL   @BLROUT      *   rout3
*
      END
```

Figure 1. SKELETON main program code for DEBUG MODE

Source code for DSK2.SKELA1, DSK2.SKELA2 and for DSK2.SKELA3 are published on the following pages. DVUG members can obtain all four programs from the Software Librarian. User groups may get them from DVUG with the article by sending a disk and mailer.

TI-99/4A ASSEMBLY DEBUG MODE: Part I in a Series

by Norm Sellers - Delaware Valley Users Group (July, 1986)

One of the fundamental practices of good programming techniques is to write programs, to not only run, but to be debugged. Many programs are written to run only, with little or no consideration for debugging. When they produce wrong results or hang the system, often the information needed to debug the program has been destroyed--not by accident but by design. This is particularly detrimental when writing in assembly language since we are now on our own. If we do not program to look for a bad condition, the system just goes to sleep when it occurs. However, when writing in BASIC, we find that it always tries (and usually succeeds, I might add) to tell you the line number in the program and the reason for an error.

I have designed a coding technique that I call "DEBUG MODE" in assembly language. "DEBUG MODE", at all times maintains:

1) A trace table of the latest Branch and Link, 'BL' routines that have been called,
2) A count of how many 'BL' routines have been executed.
3) A nesting trace table of return addresses from 'BL' routines when one 'BL' routine calls another 'BL' routine upto 6 levels of calls (or however many you wish), and
4) A counter showing at anytime how many times each 'BL' routine has been executed.

If the program is written to use 'BL' routines, these tables and counters are valuable to detect loops, or determine where in the program execution you were when the system hung up.

Incidentally, it is a very good programming practice to design programs into routines with each routine having a particular purpose. This approach often shortens programs by eliminating repeated code. Programs written in routines are, in general, easier to create and easier to maintain later.

It is very simple to use the "DEBUG MODE". Make a source file of the SKELETON program shown in Figure 1. Whenever you are writing a new assembly program, just start with this source file, adding your particular main routine which calls 'BL' routines to accomplish your task. Notice the lower case labels etc. These must be replaced by your code or deleted if not needed. The assembler will not allow lower case labels. These 'BL' routines, then in turn may (and often should) call other 'BL' routines. There are a few restrictions you must observe when using this new system:

1) Registers 13, 14 and 15 are dedicated to maintaining these tables and should not be used for application programming, unless they are carefully backed up and used only within one 'BL' routine, and restored before returning or 'BL'ing to another routine. Also, routine 'BLROUT' uses R0, R1, R2, so data cannot be stored in these when 'BL'ing to a routine.

2) Usually to Branch and Link, 'BL' to a routine, you would code:

```
      .
      .
      .
      BL @ROUTNE
      .
      .
      .
```

This statement of course loads R11 with the address of the statement following the 'BL' statement, and puts the address of the '@ROUTNE' in the program counter. Therefore, the next statement to be executed is the one at '@ROUTNE'. For example:

(Continued on next page)

DEBUG MODE (Continued from previous page)

```
ROUTNE NOP
        .
        .
        .
       RT    RETURN
```

The NOP, not neccessary for routines, is a 'no-operation' code which is equivalent to 'Jump to the next statement', (JMP $+2). Your code is depicted by the dots. The routine does what it was intended to do, then executes the 'RT' statement which is equivalent to ' B XR11 ' (remember R11 was loaded with the address of the statement following the 'BL @ROUTNE' statement. This effects a return to the original routine. Notice one requirement with this regular approach is that if we again 'BL' to a second routine while in a routine called by a 'BL', we better have saved R11 someplace before 'BL'ing to a second routine. Similarly, we had better have restored R11 with its original value before executing the 'RT' statement to return from the first 'BL' routine.

With the "DEBUG MODE", to call a 'BL' routine, two statements are always required, as follows:

```
        .
        .
        .
       LI  RTN,ROUTNE    CALL
       BL  @BLROUT           ROUTNE
        .
        .
        .
```

In this case, RTN is equated to R14. Therefore, the 'LI' statement loads the address of the user's routine 'ROUTNE' that we are about to call into R14. We then ALWAYS Branch and Link to my systems routine '@BLROUT', no matter what 'BL' routine we need or what the nesting level of the call is. The routine 'BLROUT' is permanently in all programs using this new approach by being in the SKELETON source program.

3) Every routine must have the form that is given in the following:

```
        .
        .
        .
ROUTNE DATA BLCNTS+ROUTNH#   ADDR OF COUNT EXECUTES
        .
        .
        .
       B   @BLRETN          RETURN
```

The DATA BLCNTS+ROUTNH# at the label of the routine is necessary to give the address of a two byte counter where a count of the times the routine has been executed is kept. The ROUTNH# is a naming convention I have adopted by putting a '#' after the routine names when the names are 5 or fewer characters long, or replacing the 6th character of the name with a '#'. The only requirement is it must be unique. This count is automatically maintained by the 'DEBUG MODE' system.

Also the B @BLRETN is required to return from any BL routine called at any nesting level.

There are some obvious benefits in using the new approach:

1) Any time you look at the memory of the program, whether it is run for the first time or after a minor change, the "DEBUG MODE" gives much valuable information needed to either debug or ascertain that the program is working exactly as expected.

2) Ordinarily the extra memory used and the extra time needed to run the new approach are negligible.

3) This new approach encourages improved design and memory use by encouraging the modularized or routine approach using 'BL' routines.

4) The new approach releaves us as programmers, of saving and restoring return addresses, as long as we do not exceed the BLRET table size (I have set it at 6 but this is easy to change).

EXPLANATION OF THE "DEBUG MODE".

The dedicated registers are:

NEST (R13) contains a binary value to indicate the level of the 'BL' call in progress. The value 0 indicates the main program level; the value 2 indicates the first level call; the value 4 the second level etc. with a limit of 14 if the BLRET table contains 8 words. If we need more levels, simply add zeros to the BLRET DATA statement. There is automatic checking in the program to see if this table has been exceeded during a run. We may monitor BLMAX which indicates the highest level call. When this gets close to the BLRETN table size, the table size should be increased.

RTN (R14) contains the address of the routine that is about to be called. Note: this address begins at the word used as a counter address to show how many times the routine has been executed. It is therefore necessary to actually start executing the routine two bytes after the routine's label. This is automatically handled in the system routine BLROUT.

TRCP (R15) contains the trace table pointer. The value is the displacement into the TRCE table to find the address of the latest 'BL' routine to be called. We may also monitor TRCC which counts all BL routines called during the execution of the program.

The reserved memory DATA statements are:

BLRET DATA 0,0, ... ,0

This is the table of nested calling return addresses. Every time we 'BL' to a routine, R11 contains the return address. This return address is stored in this table. The position in the table that each address is stored at is determined by the nesting level. Each time a 'BL' statement is executed before returning to the calling routine, the level indicator, NEST, is increased by two (since it is used as a displacement of 2 byte addresses). Similarly, every time a B @BLRETN statement is executed the level (NEST value) is decreased by 2.

TRCE DATA 0,0, ... ,0

This is the trace table of routines called for execution. If memory were no problem, this table could be made to be enormous and every routine called, and the order it was called could be recorded in this table. Since we usually have better use of memory, I have limited this table in a cyclic way. After the last entry of the table is loaded, the pointer is reset to zero to again go through the table entering routine addresses as they are called. When we are looking at a dump (memory contents) after a run of our program, whether it hung the system or came to normal termination, TRCP, the trace pointer which is equated to R15, should contain the displacement into the TRCE table of the latest entry. The word before this entry is the next to the last 'BL' routine called etc. Keep in mind the entry immediately before the first table entry is the last table word.

(Article continued on next page.)

DEBUG MODE (Continued from previous page)

```
BLCNTS DATA 0,0,0,0,0,0,0,0
ROUT1# EQU 0
ROUT2# EQU 2
ROUT3# EQU 4
ROUT4# EQU 6
ROUT5# EQU 8
ROUT6# EQU 10
ROUT7# EQU 12
ROUT8# EQU 14
```

This is a table of routine execution counters. There must be a 0 in the DATA statement and a corresponding EQU with a unique even number which gives the table displacement of the counter for every user 'BL' routine in the program.

Other Variables defined are:

BLMAX: Contains highest 'BL' level executed.

MYREG: Label for initialized user's registers.

LOWOP: Contains >01FF used to see if a DATA statement follows 'BL' containing arguments to the routine.

ZERO: Word contains value >0000.

ONE: Word contains value >0001.

TWO: Word contains value >0002.

BLCEND: Used to see if BLRET table is exceeded.

BLERR: Label of error routine.

BLTER1: Error message 'BLRET table exceeded.'

BLTER2: Error message '# of routines exceeds BLCNTS.'

BLRSAV: Save area for R11 for routine BLROUT.

RETSYS: Label to Branch to to return to the system.

The "DEBUG MODE" System Routines are:

BLROUT Routine:

This routine is used to control the initiation of every routine that we wish to Branch and Link to. It performs the following functions:

1) The NEST Register is incremented by two.

2) The TRCP Register is set to the displacement of the next trace table entry.

3) Save the return address in the BLRET table at a location that depends on the nesting level of the call. This is done to relieve the programmer of the need to worry about return addresses, especially when the next level 'BL' to a routine is desired.

4) Add one to the execution counter in the routine about to be executed.

5) Add two to the RTN Register to point to the first executable command in the routine (remember the routine begins with a two byte counter address).

6) Branch to the first executable command of the called routine.

BLRETN Routine:

This routine is used to control the termination of every routine that we Branch and Link to. It performs the following functions.

1) Decrease the NEST Register by two.

2) Restore R11 with the return address that was originally placed in R11 by the 'BL' statement.

3) Return to the calling routine.

This completes the execution of a called routine.

Some final notes:

1) If you wish to change the size of the TRCE trace table, put on the TRCE DATA statement, the desired number of zero entries. Next, look at the 'CI TRCP 16' statement in the BLROUT Routine. This 16 is how many bytes in the TRCE DATA statement.

2) Care must be taken not to accidentally use one of the names used in the SKELETON, other than ONE which may serve as a full word binary one for you. Just do not change it. If you wish to use a name (for example NEST) as one of your variables, it would be necessary to change every occurance of the name you wish to use in the SKELETON to something else before you start entering the code for your routines into the SKELETON.

3) The DEBUG MODE could be adapted to be used with MINI-MEMORY; however, available memory limitations must be critically watched. This 'DEBUG MODE' is best suited in big, multi-faceted programs, as can be written with the Editor/Assembler.

Next month I will publish a basic program that prints a hex/character formatted dump of memory. Basic was used since it almost never overlays the memory from the assembly program to be tested.

The following month I will publish a sample program using the DEBUG MODE. This program will print a Cross Reference listing from the source of an assembly program to be run under E/A BASIC.

Figure 2. SKELA1 source code to accompany SKELETON

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
*   SOURCE DSK2.SKELA1   07/23/86
*
BLMAX  DATA 0               HIGHEST LVL IN RUN
BLMAXE EQU  16              SIZE OF BLRET TABLE
BLRET  DATA 0,0,0,0,0,0,0,0 NESTED 8 RETURNS TABLE
NEST   EQU  13              NEST LEVEL
RTN    EQU  14              ADDR OF ROUTINE TO EXEC
TRCP   EQU  15              POINTS TO LAST TRCE ENTRY
TRCC   DATA 0               COUNT EVERY BL ROUTINE EXEC
TRCE   DATA 0,0,0,0,0,0,0,0 LAST 8 ROUTINES CALLED
*                           (1ST FOLLOWS LAST)
*
MYREG  DATA 0,0,0,0,0,0,0,0 INITIALIZED
       DATA 0,0,0,0,0,0,0,0    TO ZERO.
LOWOP  DATA >01FF           LOWEST OPCODE - 1
ZERO   DATA 0               USED BY RETSYS
ONE    DATA 1               USED BY BLROUT ROUTINE
TWO    DATA 2               USED BY BLROUT ROUTINE
*   END OF COPY SKELA1
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

(This article concludes with source codes for both SKELA2 and SKELA3, found on the following page.)

DEBUG MODE (Continued from previous page)

Figure 3. SKELA2 source code to accompany SKELETON

```
*****************************************************************
*  DSK2.SKELA2          07/23/86
*  ENTRY EQUATE PRECEEDS SKELA2.
       STWP R0             STORE ADDR OF SYS REGS
       MOV  R0,@RETSYS+2   TO RETURN TO SYSTEM
       LWPI MYREG          SET UP USERS REGISTERS
       CLR  NEST           NEST=0
       LI   TRCP,-2        TRCP=-2
*  END OF COPY SKELA2
*****************************************************************
```

Figure 4. SKELA3 source code to accompany SKELETON

```
*****************************************************************
*  DSK2.SKELA3          07/23/86
BLROUT MOV  R11,@BLRET(NEST) SAVE NESTED RETRN
       C    NEST,@BLMAX      SAVE HIGHEST LEVEL RTN
       JLE  $+6                LEVEL ROUTINE
       MOV  NEST,@BLMAX          CALLED.
       MOV  R11,@BLRSAV      SAVE TRUE RETURN FOR RTN
       C    *R11,@LOWOP    IF NEXT WORD IS GREATER
       JH   $+16           THAN THE SMALLEST OPCODE
*                             THEN THERE ARE NO ARGS
       A    *R11,@BLRET(NEST) CALCULATE THE RETURN
       A    *R11,@BLRET(NEST)  RETURN
       A    @TWO,@BLRET(NEST)   ADDRESS.
       AI   TRCP,2          TRCP+2
       CI   TRCP,16         LIMIT TRCP
       JLT  $+4                TO 16 FOR 4 ADDRS
       CLR  TRCP            TRCP=0
       MOV  RTN,@TRCE(TRCP) SAVE RTN IN TRCE
       INC  @TRCC           COUNT ALL BL ROUTINES
       MOV  *RTN+,R11       TO INCREASE THE COUNT
       LI   R1,BLTER2        EXECUTES TABLE
       LI   R2,20            IF THE TABLE HAS
       CI   R1,BLCEND        AN ENTRY, ELSE
       JGT  BLERR            BOMB WITH ERROR MSG.
       A    @ONE,*R11        COUNT+1
*                             RTN+2
       MOV  @BLRSAV,R11     RELOAD R11 FOR ROUTINE RTN
       AI   NEST,2          NEST+2
       LI   R1,BLTER1
       LI   R2,20           LIMIT NEST
       CI   NEST,BLMAXE     TO 16 FOR 8 ADDRS
       JGT  BLERR
       B    *RTN            BRANCH TO RTN ROUTINE
BLRETN AI   NEST,-2         NEST-2
       MOV  @BLRET(NEST),R11 R11=RETURN ADDR
       RT                   RETURN
BLERR  LI   R0,736
       BLWP @>2024
       CLR  @>8374
       BLWP @>201C
       MOVB @>837C,@>837C
       JEQ  $-10
       CLR  NEST            CLEAR NEST FOR NXT RUN
RETSYS LWPI 0               RESTORE SYS REGS
       MOVB @ZERO,@>837C    CLEAR STATUS BYTE
       RT
BLTER1 TEXT 'BLRET TABLE EXCEEDED'  20
BLTER2 TEXT '# OF ROUTINES EXCEEDS BLCNTS' 28
BLRSAV DATA 0
       PAGE
*
*  ROUTINES FOLLOW:
*              END OF COPY SKELA3
*****************************************************************
```

[THE DATA BUS is honored to present this contribution from Norm, author of a MUSIC PREPROCESSOR Fairware program - Ed.]

CURSOR CONTROL IN TI BASIC - By Jim Peterson

(Printed in the Syracuse 99'ers U.G. Newsletter, May 1986)

Many programs require the movement of a cursor or a figure around the screen by the use of the arrow keys, and it is usually also desirable to be able to move diagonally using the W, R, Z and C keys, and to avoid crashing the program by preventing any attempted movement beyond the 24 x 32 area of the screen, or to permit "wrap-around".

The programming routines often used for this purpose are quite lengthy, requiring 35 lines or more in BASIC for 8-directional movement. However, they do move the cursor quite rapidly, which may be essential in game programs. Much more compact routines are available, but they may be slower. The following very compact little routine is attributed to Kurt Garcia of the Houston User's Group.

```
100 R=1
110 C=3
120 CALL KEY(3,K,ST)
130 IF (K<>68)*(K<>69)*(K<>8
3)*(K<>88)+(ST=0)THEN 120
140 C=C+((K=68)*(C<30))-((K=
83)*(C>3))
150 R=R+((K=88)*(R<24))-((K=
69)*(R>1))
160 CALL HCHAR(R,C,42)
170 GOTO 120
```

That routine is a bit slow, taking about 20 seconds to move the cursor around the perimeter of the screen, and it does not permit diagonal moves. This next routine allows diagonal moves but is even slower, requiring 26 seconds to traverse the perimeter.

```
90 CALL CLEAR
100 R=1
110 C=3
120 CALL KEY(0,K,ST)
130 IF ST=0 THEN 120
140 C=C+(ABS((K=82)+(K=68)+(
K=67))*ABS(C<32))+((K=87)+(K
=83)+(K=90))*ABS(C>2)
150 R=R+(ABS((K=90)+(K=88)+(
K=67))*ABS(R<24))+((K=87)+(K
=69)+(K=82)*ABS(R>1)
160 CALL HCHAR(R,C,42)
170 GOTO 120
```

The following is perhaps the best compromise between compactness and speed. It permits diagonal movement, goes around the perimeter in about 20 seconds, and is extremely adaptable.

```
100 R=1
110 C=3
120 CALL KEY(3,K,ST)
130 IF ST=0 THEN 120
140 ON POS("WERDCXZS",CHR$(K
),1)+1 GOTO 120,210,190,180,
160,150,250,240,220
150 R=R-(R<24)
160 C=C-(C<31)
170 GOTO 260
180 C=C-(C<31)
190 R=R+(R>1)
200 GOTO 260
210 R=R+(R>1)
220 C=C+(C>2)
230 GOTO 260
240 C=C+(C>2)
250 R=R-(R<24)
260 CALL HCHAR(R,C,42)
270 GOTO 120
```

(Continued on back page)

CURSOR CONTROL IN TI BASIC (Continued from Page 9)

You will usually want the cursor to erase itself as it moves along. The movement will be smoother if you "save" the values of the old position while the new position is being computed, then use them to erase the cursor just before it is reprinted.

```
102 R2=1
112 C2=3
260 CALL HCHAR(R2,C2,32)
```

```
265 CALL HCHAR(R,C,42)
266 R2=R
267 C2=C
```

In this version, the cursor will stop at the screen border or will diagonally run along it if struck diagonally.

If you prefer it to "wrap around", change lines 150 and 250 to R=R+1(R=24). Change lines 160 and 180 to C+C+1+(C+32). Change 190 and 210 to R=R-1-(R=1), and change 220 and 240 to C=C-1-(C=1).

If you want the program to also respond to joystick input, change line 130 to read IF ST=0 THEN 142 and add the Lewis routine:

```
142 CALL JOYST(1,X,Y)
143 Z=((X+3)*Y)/4)+5
```

```
146 ON Z GOTO 240,250,150,22
0,120,160,210,190,180
```

That will activate joystick #1. If you want response from either joystick, you can add:

```
143 IF (X<>0)+(Y<>0)THEN 145
```

```
144 CALL JOYST(2,X,Y)
```

However, all these options will slow up the response time. Usually, it is better to require a choice of keyboard or joystick at the beginning of the program. This routine utilizes Doug German's neat little routine which activates whichever joystick is being held when the fire button is pressed.

```
50 PRINT "Will you use":"(1)
 arrow keys, or":"(2) joysti
cks?"
52 CALL KEY(3,Q,ST)
54 IF (ST=0)+(Q<49)+(Q>50)TH
56 IF Q=49 THEN 100
58 PRINT "Press fire button"
```

```
60 CALL KEY(1,K1,S)
62 CALL KEY(2,K2,S)
64 IF K1+K2<>17 THEN 60
66 J=INT(K1/18+K2/9+1)
110 IF Q=50 THEN 142
130 IF ST=0 THEN 120
142 CALL JOYST(J,X,Y)
```

Delete lines 143 and 144.

```
146 ON Z GOTO 240,250,150,22
0,142,160,210,190,180
```

```
270 IF Q=49 THEN 120 ELSE 14
2
```

Perhaps you will want to move the cursor around the screen without erasing any text or graphics already on the screen. This is easily done:

```
260 IF (R=R2)*(C=C2)THEN 270
262 CALL GCHAR(R,C,G)
```

```
264 CALL HCHAR(R2,C2,G2)
268 G2=G
```

And if you want to also use other keys for program input, just add more letters to the string in line 140, and more line numbers in the corresponding positions. For instance, if you want to use the F key to fire at the alien invaders,

```
140 ON POS("WERDCXSF",CHR$
(K),1)+1 GOTO 120,210,190,18
0,160,150,250,240,220,600
```

```
600 CALL SCREEN(16)
610 CALL SCREEN(4)
620 GOTO 120
```

Jim Peterson now has available TIPS FROM THE TIGERCUB #3 (62 programs, routines, etc. from Nos. 25-32), $15 or $27 for two TIPS disks or $35 for all 3 (Tips 1-3) postpaid.

His catalog of over 130 original programs ON TAPE OR ON DISK is available for $1 (deducted from first order).

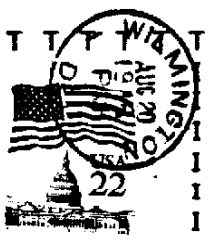
Nuts & Bolts No. 2, 185 new XBasic merge routines fully compatible with disk #1, is $19.95; #1 and #2 together= $37.

Mention the Delaware Valley Users Group, ordering from TIGERCUB SOFTWARE, 156 Collingwood Ave., Columbus, OH 43213.

DELAWARE VALLEY USERS' GROUP
P.O.BOX 6240
STANTON BRANCH
WILMINGTON, DE 19804

Subscription
expires with
date on label

FIRST
CLASS
MAIL



TO:

0000 FOREST LANE USERS GROUP
JOE GILLO - 4413 CORNELL DR
GARLAND TX 75042

GOING AWAY TO SCHOOL? Give DVUG your address!