May 1985

TSX—PLUS USER'S GROUP NEWSLETTER

TSX-PLUS USERS' GROUP (TSXPUG) NEWSLETTER

Newsletter

TSXPUG
c/o Jack Peterson
Horizon Data Systems
1901 Wildflower Terr.
Richmond, VA 23233

Administration

TSXPUG
c/o Glenn Burnett
Zia Corporation
P.O. Box 351
Morris Plains, NJ 07950

IMPORTANT NOTICE

The TSX-Plus Users' Group Newsletter is a quarterly publication of the TSX-Plus Users' Group (TSXPUG), a volunteer organization dedicated to the exchange of information among users of TSX-Plus. TSXPUG is sanctioned by, but is otherwise independent of and receives no financial support from S & H Computer Systems, Inc., developers of TSX-Plus. All expenses are paid from membership fees.

Jack and Glenn, whose addresses are above, run TSXPUG. Jack writes and edits the newsletter; Glenn handles administration, collecting money, maintaining the membership list, and mailing the newsletter. If you have a contribution for or a question about the newsletter, write to Jack; direct all other correspondence to Glenn.

Membership fees are $15.00 (US) per year in the USA, Canada, and Mexico, $20.00 (US) per year elsewhere. Memberships must be paid in US dollars with an instrument payable at a bank in the United States. All memberships terminate on December 31 each calendar year. Partial year memberships are NOT pro-rated. Current members will receive a renewal notice in January for the coming year; payment must be returned before February 25th to assure receipt of the first newsletter of the year. New members may join at any time by completing the application on the inside back cover, and including the appropriate membership fee with the application.

Reader contributions to the newsletter are solicited. Brief articles may be submitted in hardcopy form, but we prefer lengthy articles on magnetic media. We can read RX01, RX02, and 1600 bpi magnetic tape; in a pinch, RX50 and RL02; in a real pinch, RL01. Our preferred format is .RNO or .DOC. We will return media to you if you ask us to do so. All submissions become the property of the TSX-Plus Users' Group. Thanks for your cooperation and your contributions.

All material contained herein is presumed to be correct, but TSXPUG, its staff, and newsletter contributors assume no responsibility for any errors that may appear. Readers use any and all published material at their own risk.

COPYRIGHT NOTICE

CONSULTANT HOTLINE

Since starting this column in November, we have published the names, addresses, and telephone numbers of firms providing TSX-Plus consulting services as they have made themselves known to us. This information is now spread out over several newsletter issues, and a few more names have arrived. What follows is a complete list, in zip code order, of all TSX-Plus consultants known to us. We neither endorse nor recommend any of these firms (except ourselves, of course), but publish this information for the benefit of readers who may need special help with their TSX-Plus systems.

Please save this list for future reference. We now plan to publish a complete list once each year, probably in the May issue. If you provide TSX-Plus consulting services and are not on this list, send us the necessary information. We will see that you are included in the next list.

Ned Rhodes
Software Systems Group
1684 E. Gude Drive
Rockville MD 20850
(301) 340-2773

Jack Peterson
Horizon Data Systems
1901 Wildflower Terrace
Richmond VA 23233
(804) 740-9244

Lorin S. Jordan
ABCCI
7400 Metro Boulevard
Suite 109
Edina MN 55435
(612) 893-1354

Peter Heinicke
Precision Computer Methods, Inc.
No. 3 2S779 Winchester Circle
Warrenville IL 60555
(312) 393-6478

Tom Smith
Data Processing Services, Inc.
P.O. Box 40128
San Antonio TX 78240
(512) 692-0953

David C. Ketchum
STW Software, Inc.
7860 W. 16th Avenue
Suite 115
Denver CO 80215
(303) 232-9164

Nick A. Bourgeois, Jr.
NAB Software Services, Inc.
3236 Candlelight N.E.
P.O. Box 20009
Albuquerque NM 87154
(505) 298-2346

Milton D. Campbell
Talisman Systems
Drawer CP-255
Manhattan Beach CA 90266
(213) 318-2206

Shal Farley
Cheshire Engineering Corporation
650 Sierra Madre Villa
Suite 201
Pasadena CA 91107
(818) 351-5493

Jim Crapuchettes
Omnex Corporation
2483 Old Middlefield Way
Mountain View CA 94043
(415) 966-8400

Gregory Merrick
IRIS Registration, Inc.
75 The Donway West
Suite 1014
Don Mills, Ontario M3C 2E9
Canada
(416) 446-1066

David Clements
AccountAbility Computer Services
P.O. Box 6043, Stn 'A'
Calgary, Alberta T2H 2L3
Canada
(403) 236-0740

A GOOD USE FOR A DETACHED JOB

Detached jobs are one of the useful features of TSX-Plus, but many people do not know how to use them. At Horizon, we have developed the concept of a system configuration detached job we call SETSYS (Set System).

With the advent of Version 5.1, TSX-Plus has provided users a high level of flexibility in altering the characteristics of time sharing and communication (CL) lines. Parameters such as line speed and phone operation previously required modifications to TSGEN.MAC, reassembly, relinking, and rebooting. But now, these parameters and others can be changed by commands.

In our installation (and those of our customers), we define a detached job file called SY:SETSYS.TSX and specify, via the DETACH macro in TSGEN.MAC, that it be executed automatically when the system starts. TSGEN.MAC is built to reflect our normal configuration, and SETSYS is used to alter that configuration on an as needed basis. For example, if we later wish to use a spare port (always genned as a time-sharing line) as a CL, we place the appropriate SET CLn LINE=xx command in SETSYS. Or, if we later wish to use that port as a dial-in line, we SET the line as PHONE and AUTOBAUD in SETSYS. In this way, we eliminate many SYSGENs and/or versions of TSX-Plus, and are confident that the system will always come up with our latest configuration because SETSYS is executed whenever TSX-Plus is restarted.

SETSYS may contain any commands that affect the system on a global basis because it runs with operator privilege. For example, handler SET commands can be issued to ensure that handlers have a known initial configuration. SETSYS cannot be used to make global logical name assignments, however, because such assignments are local to each line.

To use SETSYS, you must specify at least one detached line in the TBLDEF macro of TSGEN.MAC, but, because SETSYS runs for only a short period, you need not otherwise increase the number of detached lines you already use. Simply place the SETSYS commands at the beginning of one of your other automatically executed detached files.

SENDING MESSAGES TO INACTIVE TERMINALS

One of the items mentioned in our Christmas wish list was the ability to send messages to inactive terminals. We have occasionally found this useful, particularly after we have brought the system back up after a brief outage. Inspired by Milton Campbell's creative use of CL in the last newsletter, and ideas from others, Horizon has developed a program, ONLINE, reproduced at the end of this newsletter, which sends a user-defined message to all inactive terminals. The comments on the program are self-explanatory and should allow you to customize the program as needed for your own installation. You must have at least one free CL unit on your system in order to use ONLINE.

To use ONLINE, first enter the program as file ONLINE.MAC. Then assemble and link it using MAC ONLINE and LINK ONLINE commands. Finally, copy the resultant ONLINE.SAV to your system disk. ONLINE is run by the following commands:

```
R ONLINE
<Put Message To Be Sent Here>
```

In our installation, we put this command (the message reads *** SYSTEM IS OPERATIONAL...YOU MAY NOW LOG ON ***) in SY:SETSYS.TSX (see previous column). Whenever TSX-Plus is restarted, the message is displayed on all inactive terminals.

CUSTOMIZING STARTUP COMMAND FILES

We are indebted to S & H for this idea.

Many TSX-Plus installations, especially those with dial-in lines, use account codes and passwords to regulate access to system features and files, and to perform automatic mounts and logical device

assignments for users. In TSGEN.MAC, each line definition may include a CMDFIL macro naming a startup command file executed when the line becomes active. Often, all (or most) lines specify a common command file that ends with R/LOCK LOGON to start the logon process. After the user is identified, a command file unique to that account, containing ACCESS, MOUNT, ASSIGN, and other commands, is executed.

Many users would like to put ACCESS commands in one command file which the user cannot change (.TSX extension on the system disk), then chain to a user-modifiable command file so each user can control default assignments, mounts, and UCL definitions. Unfortunately, once a startup command file has been started, TSX-Plus interprets all input (terminal or command file) as part of the startup process, so a user could defeat file access permissions by putting ACCESS commands in his command file.

Startup privilege (during which ACCESS commands are allowed) ends whenever a .GTLIN request (usually issued by KMON) finds no input available. If we could arrange to have this event occur, we could then execute a user-modifiable command file during startup without startup privilege. The following program, LOGCHK, does exactly that, and a little more. We present the program, explain how it works, and how we use it. Feel free to copy and/or modify this program for your own installation.

```
        .TITLE   LOGCHK
;
;       HORIZON DATA SYSTEMS, RICHMOND VA
;
        .ENABL   LC
        .MCALL   .EXIT,.GTLIN
        .MCALL   .LOOKUP,.TTINR
        .ASECT
        .=44
        .WORD    10100
        .=510
        .WORD    2$-1$
1$:     .ASCIZ   "@LOGON"
2$:
        .PSECT
START::
        .TTINR
        BCC      START
        .GTLIN   #BUF,#BUF,TTINP
        .LOOKUP  #BUF,#0,#DBLK
        BCS      1$
        BIS      #4000,@#44
1$:     CLR      R0
        .EXIT
DBLK:   .RAD50   "DK LOGON COM"
BUF:    .BLKW    41.
        .=BUF
        .NLIST   BEX
        .ASCII   <15><12>"Prompt:"<200>
        .END     START
```

LOGCHK first drains the terminal input ring buffer, discarding all typed-ahead input. This has two benefits. First, typed-ahead input is not interpreted as part of the startup process, enhancing security. Second, it forces the following terminal-mode .GTLIN request to hang for lack of input data, ending startup privilege. The .GTLIN request displays the message "Press Return to Continue:" (abbreviated "Prompt:" in the program above due to column width limitations).

After the user responds, type-ahead is permitted, and LOGCHK determines whether file DK:LOGON.COM exists. If so, it is executed, terminating the original startup command file; otherwise, control is returned to the startup file permitting default assignments, etc., to be made. Neither commands in LOGON.COM nor those remaining in the startup command file are executed with startup privilege.

To use LOGCHK, enter the program as LOGCHK.MAC, assemble and link it using MAC LOGCHK and LINK LOGCHK commands, and copy the resultant LOGCHK.SAV file to your system disk.

In our installation, all lines are directed (via the CMDFIL macro) to command file SY:TSXGO.TSX containing assignments (ASS CL2 LP) common to all users and terminated by R/LOCK LOGON. All lines have the $QTSET flag set so startup files are not displayed. Each logon account names a unique command file that defines access permissions as appropriate, mounts a subdevice unique to that account, and

assigns it logical name DK:. A R/SINGLE LOGCHK is the last command in the file (/SINGLE is needed to prevent LOGCHK from hanging while purging the input stream). Users are invited to place MOUNT, ASSIGN, and other commands, as well as UCL definitions in their own LOGON.COM files, and to make changes as desired, subject only to privilege and access permissions.

Several modifications to LOGCHK could be useful. For example, .GTLIN could request entry of a password to be compared against a key stored in LOGCHK or in a file accessible only to the system manager. This places password protection in the hands of the system manager because users, who can change their logon passwords to strings easily guessed by intruders, cannot change this value. Another modification could interpret the .GTLIN response as the name of a command file to execute.

TSX-PLUS .FORK PROBLEM

Jim Crapuchettes and John Rose of Omnex Corporation have forwarded an analysis of and a solution to a .FORK problem in TSX-Plus that affects users of DSD 880 Winchester/floppy drives, and those who write handlers that perform internal queueing. Their article follows.

A. DSD 880 PROBLEM

A DSD 880 contains a single controller which services both a Winchester drive (emulating RL02s) and a floppy disk drive (emulating an RX02). Under TSX-Plus Version 5.1, simultaneous Winchester/floppy access on a DSD 880 can cause a system crash: a kernel mode trap (KTP) from the DY handler may occur upon attempted accesses to the floppy Control and Status Register (DYCSR). The problem is most apparent when using an LSI-11/73 processor, but it can also occur on other LSI-11 and PDP-11 systems.

B. ANALYSIS

Under TSX-Plus, both the DL and DY handlers execute a .FORK in their interrupt handling routines. However, the default priority of I/O initiation (12.) is lower than the default priority of an interrupt .FORK (50.). Apparently, the following sequence of events occurs:

1. The DY handler initiates I/O on the emulated RX02, with the initiation routine completing normally.

2. The system sets the .FORK priority to FP$IOS (12.) and calls the DL handler to initiate a new I/O request.

3. The DL handler starts to initiate I/O on the emulated RL02.

4. Upon completion of the previously initiated DY I/O, the RX02 signals an interrupt. The DL handler is interrupted and program control is passed to the DY handler interrupt entry point, which does a .FORK call with an implied priority of FP$IOF (50.).

5. The CPU executes the rest of the DY interrupt routine, since its priority is FP$IOF (50.), which is higher than the DL initiation priority of FP$IOS (12.).

6. Apparently the DSD 880 interface hangs because of a protocol problem (some, but not all, of the steps involved in starting a DL I/O have been done, so the DSD 880 thinks it is a DL now), and does not recognize the DYCSR address.

7. After a few microseconds a bus timeout occurs, causing a kernel mode trap.

C. SOLUTION

The solution is to change the TSX-Plus .FORK priority for I/O initiation from 12 (decimal) to 50 (decimal). Find the following line (in the non-user-modifiable section of TSGEN.MAC:

```
FP$IOS = FP$FLG+12. ;I/O initiation
```

Change "12." to "50.", as follows:

```
FP$IOS = FP$FLG+50. ;I/O initiation
```

After making this change, regenerate your TSX-Plus system.

). INTERNAL QUEUEING HANDLERS

This same problem might affect internally queued handlers, such as the KERMIT KM handler. Internally queued handlers often assume that a .FORK in the interrupt code will eliminate potential interlock problems during internal queue manipulation. This is not necessarily the case under Version 5.1 of TSX-Plus, because of the difference between initiation and interrupt .FORK priorities.

Without proper caution, internally queued handlers could experience interlock problems similar to the DSD 880 problem described above. Programmers writing handlers for Version 5.1 should acquaint themselves with TSX-Plus .FORK priorities (Chapter 4, TSX-Plus System Manager's Guide), and take them into account when using .FORK requests.

If you have further questions or difficulties concerning .FORK priority problems under TSX-Plus V5.1, contact:

Jim Crapuchettes
Omnex Corporation
2483 Old Middlefield Way
Mountain View CA 94043
(415) 966-8400

AN IND HINT

If you use timeouts in IND .ASK, .ASKN, or .ASKS directives, be sure to SET TT NOWAIT before running IND or the timeout will be ineffective (the system will hang waiting for terminal input).

HINTS AND KINKS

Ned W. Rhodes
Software Systems Group
1684 East Gude Drive
Rockville, MD 20850
(301) 340-2773

The response to my first column was so underwhelming that I decided to contribute another. As always, if you have an unusual problem or a workaround to one that I identify, send it to me and I will pass it on in the next newsletter.

DY Revisited

In our last adventure, a system manager was having problems with floppy drives, experiencing "Trap to 4" errors. I suggested you lock PIP, DUP and DIR into low memory to solve that problem. You will notice in a column above that Jim Crapuchettes gave additional information about the DSD 880 disk. I intend to unlock PIP, DUP, and DIR and try his suggestion. I will report the results in the next issue.

A FORTRAN Question

The following was contributed by Bob Walraven. Care to have a crack at it?

One of our students has a FORTRAN program that he ran under TSX-Plus V5.0. The program opens a couple of logical units and computes for a while before trying to open a third logical unit. At this point the program aborted indicating insufficient memory for the new I/O buffer. That made sense, but here is the interesting part. After thinking about the problem a bit, he had a flash of brilliance and ADDED another 2Kword array to his program. When he ran it again, it completed successfully. Do you know why?

The answer might appear a little later on.

MT/MS Revisited

In the last issue, we mentioned some MT/MS problems. Tom Shinal has reported the following:

PROBLEM: If a tape operation is aborted using the MS Handler while under TSX-Plus, the system hangs and a reboot is required. An abort can be a Control-C or any other anomoly (no, an anomoly is not a sea urchin).

HARDWARE: Cipher Cachetape with Emulex TC-02 controller.

SOLUTION: Check the Emulex controller for

the revision level of the PROMS. We had trouble with REV D. REV H is the current revision and solves the problem. Emulex replaced them at no charge. I won't speak for them, but if you have the same problem, give them a call.

As Jack Peterson has pointed out before, there are cases in which the mag tape handler can get corrupted, usually when aborts or other unexpected things happen. Under RT-11, the solution is simple -- unload and reload the handler. Under TSX-Plus we can't reload handlers...yet.

FORTRAN Solution

His original program required about 55 Kbytes, so TSX-Plus gave him 56K bytes (even though MEM 64 had been issued). When the program started, it did a .SETTOP to get FORTRAN buffer space, receiving about 1Kbyte, enough for two buffers, but not three. By adding the 2Kword array, the job size passed the 56Kbyte boundary, so TSX-Plus allocated a full 64Kbytes for the job, ensuring adequate room for the I/O buffers. The problem does not occur with small jobs because TSX-Plus allocates the 56Kbyte default.

Moral of story: If your FORTRAN job is about 54 or 55Kbytes, you may want to make it a little over 56 Kbytes in size. Running SETSIZ to add additional space to the original program will do it.

A Small Problem

John Davies at the Naval Ship Research and Development Center shared this "feature" with me. If you turn your terminals off at night then on again in the morning, TSX-Plus drops the first character of your logon. I didn't believe him until I tried it out. The problem occurs only during the first logon; for the rest of the day, no problem. Have you seen this?

[Editor's note: Yes, we have. Anytime we power up a terminal AFTER TSX-Plus has started running, the carriage return typed to activate the line gets through okay, but the first character of the account name is invariably discarded. Very, very strange.]

The Dreaded TO State

To finish up, I would like to share a problem concerning the terminal output (TO) state. The WHO system command produces a snapshot of the current status of each line, including its execution state: HI means high priority, TI means waiting for terminal input, and TO means terminal output is pending, but the terminal has sent an XOFF character suspending further output operations to it.

A problem occurs when a job in TO state is killed either by the KILL keyboard command or the user hanging up the phone without logging off. Because the system is waiting for an XON from the terminal, it cannot output the logoff message so the line is left active and in the TO state.

This problem occurs frequently on our somewhat noisy dial-in lines. A user may be getting a directory display when a burst of noise causes the modems to drop carrier and disconnect. If the line was in the TO state at that time, the line hangs in this state and is not logged off. TSX-Plus drops Data Terminal Ready (DTR) (as it should), but because the line is not inactive, subsequent calls to the port modem go unanswered, and a dial-in line has been lost.

One workaround is to force DTR high on the modem at all times permitting the modem to answer incoming calls without intervention by TSX-Plus. Then a user calling in can enter a Control-Q (XON) to clear the TO state, and use the line. But this causes security problems because the new caller inherits the old user's output buffer. Do you really want everyone looking at your resume?

Version 5.1C was supposed to fix the problem by waiting a while before automatically flushing the output buffer. My experience says this is not happening. Has anyone else experienced the same problem? I have also had the problem with hardwired (non-modem) lines. Drop me a note if you have had similar problems.

THE DIBOL/DBL WIZARD RETURNS

Due to circumstances beyond our control (as they say), John Stewart's column was noticeably missing from the last newsletter. He is now back with a vengeance. Even if you don't know a DIBOL from a SNOBOL or a COBOL or a BASE-BOL (go Atlanta!), you might find his column quite interesting. Read on...

SYSTEM-WIDE SEARCHING

A system manager often needs to search many files efficiently for a given keyword or phrase. A change affecting the entire system may require it, or a troublesome glitch may be causing problems and it is not where you thought it was. Your need for this will be especially acute if you do not have a nice documentation package.

Searching efficiently is the big problem for minis doing global searches. You don't have all day to get the answer. Your system has 500 source files and you want to find a keyword in less than 5 or 10 minutes.

The DBL/DIBOL programmer has seen all kinds of line- and character-oriented search routines that were fine if you didn't mind taking all day running a program that slowed everybody down for the whole time. Several hours is typical for this type of global search routine.

The routines provided in this issue do global searches of several hundred sources in 5-10 minutes on an idle system. The DBL version shown below searched TSGEN.MAC for the word "SPOOL" and output all lines on which it was found on the terminal and in a disk file in less than six seconds. We can live with that speed.

There are two solutions, each with its own advantages:

1. A TECO routine that you can key in quickly and get running.

2. A DBL system that is more than twice as fast as the TECO routine. It uses DBL's assembly-language search subroutine for forward searches and a backward version using DBL's excellent assembly language interface.

Both routines find the key and display the text line or record where found, both on the terminal and in a file, along with the name of the file. You can modify these routines to search files containing non-printing characters; only the file name is displayed. This modification is quite easily done.

This column is dealing with global searches where the key is matched regardless of where it appears in a record. There are many methods to limit a search to certain positions within a record or line. One example is using RTSORT for positional searching. It is fast. The sort key must be stated, but it is really a dummy if only one match is found. Use the /INCLUDE selection key for your search. It is also possible to modify the methods here to test the distance from the keyword to the beginning of the containing line and to accept matches only if this distance is within a defined range. Other than these brief comments, this section will focus exclusively on global searches.

The TECO Global Search

At the system prompt, enter:

```
SRCH KEY FILENAM
```

* and % may be used as wildcards in the FILENAM parameter.

Enter the following program into file SRCH.COM. ^ is a real caret; $ is a real dollar sign. Structuring and documentation are omitted to conserve space:

```
^>R/N TECO
6ET^$^$0,128ET^$-1^^X^$
EWSRCH.TMP^$I
Search for ^1 in ^2.
Results stored in SRCH.TMP.
^$HTHPWHKEN^2^$
<HK:EN^$;IER^$G*27I^$HXAHKG*
14+(0^Q)<I ^$>HT13^THKMA
<:E_^1^$;0^Q+.,^Q-2+.FB^N^ES^$
.-1,.:FB;^$"LLF<'.-1,^Q+.XAL
```

```
.UAZJ.UZGAQZJG*QZ+15-.<I ^$>I>  ^$
VQZ,ZPWQZ,ZKQAJ>>
HKISearch for ^1 in ^2.
Results stored in SRCH.TMP.
^$HTEX^$^$
```

As each file is scanned its name appears on the screen. If a match is found, the containing line is displayed after the file name. The same information is output to SRCH.TMP for printing, etc. For its size, this is a very powerful routine, proving there will always be a need for TECO in the commercial software area.

If you are using TSX-Plus 5.1B, you may find your TECO command files no longer properly process TSX-Plus command file control characters. Here is a fix to the 5.1B TSGEN.MAC file from S & H that allows TECO programs to operate without echoing:

Locate these lines in the non-user-modifiable part of TSGEN.MAC:

```
.RAD50  /TECO  /
.WORD   AF$SCA!AF$HIE!AF$NOW
```

Replace the last line with:

```
.WORD   AF$SCA!AF$NOW
```

This change disables high efficiency terminal mode, and allows TECO to operate without echoing while maintaining compatibility with RT-11. According to S & H, TSX-Plus Version 5.1C will include this change [Editor's note: it does]. It still does not allow all command file control characters (Chapter 3, TSX-Plus Reference Manual) to work the way they did before. ^! does turn echoing off and ^) turns it on, but ^(seems to be ignored, so you get the TECO commands along with the desired terminal output. Perhaps S & H can work this out in the future.

S & H had some nice ways TECO could be used in a command file. Many of us have dozens of these running on our systems. Contrary to the opinions of some, TECO is not dead. It is hard to use, granted, but some of us went over the hump years ago and haven't found anything since with quite so much power on TSX-Plus.

The Faster DBL Method

First, this method can be done only in DBL, not DIBOL, because DIBOL requires files to have pretty "straight" DDF format, while DBL can be used to process any file, including those with non-printing characters. DIBOL assembly language routines could be used, but with much more risk and difficulty. More importantly, DIBOL won't let you read binary files. Assembly language routines are absolutely necessary for speed and DBL's assembler interface is well documented.

This program searches considerably faster than TECO, although TECO runs rings around most other high-level search routines. VFSRCH is a DBL program that does forward and backward searches, mostly in assembly language, at 15K to 20K bytes per second on our 11/34 TSX-Plus system with or without the generalized data cache. Because of its speed, it is used very often and is probably my most valuable productivity tool.

Included here is the core of the program only, vastly simplified from what I use, but you can expand on what is here. A great deal of what I do works under TSX-Plus as layers, so the whole system listing would be larger than we could print in this newsletter.

Don't be afraid of BSTR.MAC. Very easy instructions are given, so if you have never programmed in anything but DIBOL or DBL, you'll do just fine with BSTR.MAC. It is included because an efficient backward search is needed to isolate the text line where a match is found so the full line may be displayed.

The fast DBL forward search routine INSTR comes with DBL and is one of my favorites. I used to wonder why DISC didn't just change the direction of the search, but, once I got into it, I found that the fuctionality was quite different. It did set up similarly, though. Incidentally, include BSTR.OBJ in your subroutine library if you find it useful. BSTR documentation would be similar to INSTR except that it goes backward.

ome of the things you can do to make VFSRCH more productive (very much so) is front-ending it with a directory listing obtained from a wildcarded filespec, as you can do with the TECO routine described earlier. Again, I am not including our complete implementation because of space, but it is available if you don't mind developing a complete system. The basic key is a subroutine that does keyboard commands from within a DBL program -- all done in DBL.

In our version, we ask DIRECTORY to create a file of names that satisfy our wildcarded and parenthetical (not supported by the TECO routine) file specifications, then our program reads the file DIRECTORY produced. [Editor's note: Horizon has developed an IND program that does the same thing; we will try to publish some of it in the next newsletter]. Our implementation can thus look through multiple files at a rate of 500 typical source files in less than 10 minutes on an 11/34. That's nice. We don't usually ask for a full scan, so most of what we run takes only a few seconds, not minutes.

Another variation outputs only the file name whenever a match is found, so we can scan binary files, too. Just drop the parts that output the text line and look for nulls and EOF characters, and use a general error trap to detect when processing is done. BSTR is not needed since this variation does not output text lines (although you could have the routine find and display a limited number of nearby printable characters if you wished, but this should be done in high-level DBL).

DBL also allows a device name as a filespec. Our 65,535-block system disk was searched in 28 minutes, about 20K bytes per second! Major changes would not be required to report the block and byte where matches occur, or to test for printable characters in the neighborhood of the match. Such a program could be handy the next time you have to rebuild a directory!

When run, the program prompts the user for both the key and the filespec expression, which may contain any valid DIRECTORY constructs, including wildcards and factoring, e.g., PRG:*.(DBL,SUB). If the first valid character (not necessarily the first character) in a file line is a semicolon (a comment), any match on that line is ignored unless a /C switch is appended to the key value.

```
 .TITLE VFSRCH   ;Very fast search
;Very fast search.
;Scans a device or file up to EOF or NUL.
;
;Stores results in VFSRCH.TMP or optional
;  file name.
;
;Entire line where match found is
;  displayed and output except:
;     (optionally) leading space and tabs
;     (optionally) if a semicolon (;) is
;        the first non-tab/space on a line,
;        any match on the line is ignored.

RECORD
        FN,     A14
        OFN,    A10
        ALL,    D1
        KEYSZ,  D2
        COMM,   D1
        ENTRY,  A14

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Adjust these sizes upward or downward so
; it still fits.  Use 512-byte increments.
; Adjust blocking factor (see DBL notes
; for TSX implementations) like this:
; PAGE/512 = Blocking factor.
        PAGE,   A14848
         HALF,  2A7424  @PAGE
        HFPGSZ, D4,     7424
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        CRLF,   A2
         CR,    A1  @CRLF
         LF,    A1  @CRLF+1
        CTRLZ,  A1
        NULL,   A1
        NUPAG,  D1
        TAB,    A1
        KEY,    A50
        SPECSZ, D2
        EOFERR, D1
        SIZ,    D5
        BEG,    D5
        EOL,    D5
```

```
        KEYFND, D5
        BOL,    D5
        EOF,    D5
        NULFND, D5

PROC
  OPEN (15,I,'TT:')
  XCALL DFLAG (111,1)
  XCALL FLAGS (01001010)
  XCALL ASCII (0,NULL)
  XCALL ASCII (26,CTRLZ)
  XCALL ASCII (9,TAB)
  XCALL ASCII (13,CRLF)
  XCALL ASCII (10,LF)
  DISPLAY (15,
& 'Ignores comments unless /C appended to
& key.',CRLF,'Wildcards, etc. accepted.',
& CRLF,'Key=')

KEYENT,
  READS (15,KEY)
  IF .NOT.$RDLEN THEN
      BEGIN
      DISPLAY (15,'A valid key must be
        & entered:',CRLF)
      GO TO KEYENT
      END
  KEYSZ = $RDLEN
OUTDEV,
  DISPLAY (15,10,27,'M',
      &  'Output file or device name
      & [default=VFSRCH.TMP]:  ',
      &  27,'7')
  READS (15,OFN)
  IF .NOT.$RDLEN THEN
      BEGIN
      OFN = 'VFSRCH.TMP'
      DISPLAY (15,27,'8',OFN,CRLF)
      END
  ON ERROR OUTDEV
  OPEN (3,O,OFN)
  OFF ERROR
  IF OFN.NE.'TT:' THEN
    DISPLAY (3,CRLF,CRLF,CRLF,'<<<Search
      & for ',KEY(1,KEYSZ),
      & ' in filespec ')
  IF KEY(KEYSZ-1,KEYSZ).EQ.'/C' THEN
      BEGIN
      COMM = 1
      KEYSZ = KEYSZ-2
      KEY(KEYSZ+1,KEYSZ+2) =
      END
  IF .NOT.KEYSZ THEN
      BEGIN
      DISPLAY (15,'<<<VFSRCH-F-No valid
        & characters for key.',CRLF)
      XCALL DAESC (13)
      STOP
      END
  XCALL DAESC (13)
  DISPLAY (15,'Strip leading spaces and
      & tabs [default=Y]?  ')
  READS (15,ENTRY)
  IF ENTRY.EQ.'N' THEN ALL=0 ELSE ALL=1
FILE,
  DISPLAY (15,'File to search:',CRLF)
  XCALL ACESC (13)
  READS (15,FN)
  DISPLAY (15,CR)
  IF .NOT.$RDLEN THEN GO TO FILE
  SPECSZ = $RDLEN
  DISPLAY (3,FN(1,$RDLEN),CRLF)
  OPEN (2,I,FN,30)
  DISPLAY (15,CR,FN,' >  ',CR)
  GETS (2,PAGE,PX)
  IF $FALSE THEN
      BEGIN
PX,
      EOFERR = 1
      END
  CALL EOFCK
NXSRCH,
  XCALL INSTR (BEG,PAGE,KEY(1,KEYSZ),
        & KEYFND)
  IF .NOT.KEYFND.OR.EOF.AND.
        & EOF.LT.KEYFND THEN
      BEGIN
      IF EOF THEN GO TO EOFILE
      CALL HALF
      IF .NOT.EOF.AND.EOFERR
        & THEN GO TO EOFILE
      BEG = HFPGSZ+1
      GO TO NXSRCH
      END

;Key is found

  NUPAG = 0
EOLCK,
  XCALL INSTR (KEYFND,PAGE,CRLF,EOL)
  IF .NOT.EOL THEN
      IF .NOT.NUPAG THEN
          BEGIN
          NUPAG = 1
          IF KEYFND.GT.HFPGSZ THEN
              BEGIN
              CALL HALF
              KEYFND = KEYFND -HFPGSZ
              GO TO EOLCK
              END
```

```
          ELSE EOL = EOF
          END
  IF EOL THEN EOL = EOL-1
  XCALL BSTR (KEYFND,PAGE,CRLF,BOL)
  IF BOL THEN BOL = BOL+2  ELSE BOL = 1
  IF BOL.LT.EOL THEN
      BEGIN
      WHILE ALL.AND.(PAGE(BOL,EOL).EQ.' '
        &  .OR.PAGE(BOL,EOL).EQ.TAB)
        &  DO PAGE(BOL,EOL) =
        &  PAGE(BOL+1,EOL)
      IF PAGE(BOL,EOL).NE.';'.OR.
        &  COMM.OR.ALL.EQ.0 THEN
          BEGIN
          IF .NOT.ALL THEN SIZ=EOL+1-BOL
              &  ELSE XCALL TRIM
              &   (PAGE(BOL,EOL),SIZ)
          IF SIZ THEN
              BEGIN
              DISPLAY (3,FN,' >  ',
                & PAGE(BOL,BOL-1+SIZ),
                & CRLF)
              DISPLAY (15,FN,' >  ',
                & PAGE(BOL,BOL-1+SIZ),
                & CRLF)
              END
          END
      END
  FN =
  BEG = EOL+1
  GO TO NXSRCH

EOFILE,
  CLOSE 2
  CLOSE 3
  DISPLAY (15,'Output goes to ',OFN,CRLF)
  XCALL DAESC (13)
  STOP

HALF,
  HALF = HALF(2)
  GETS (2,HALF(2),CTRLZ)
  CALL EOFCK
  RETURN

CTRLZ,
  EOFERR = 1

EOFCK,
  IF $RDTRM.EQ.0.OR.$RDTRM.EQ.26
        &  .OR.EOFERR THEN
      BEGIN
      XCALL INSTR (0,PAGE,CTRLZ,EOF)
      XCALL INSTR (0,PAGE,NULL,NULFND)
      IF .NOT.EOF.OR.NULFND.LT.EOF
          & THEN EOF = NULFND
      IF EOF THEN EOF = EOF -1
      END
    ELSE EOF = 0
    RETURN
END
```

```
Next, BSTR.MAC:

        .TITLE  BSTR
        .ENABL  LC
        .MCALL  .TTYOUT

PTR:    0       ;Where string is.
LNG:    0       ;How long string is.

        .GLOBL  BSTR
        .GLOBL  $CKARG,$CLRLT,$POPNM
        .GLOBL  $FMTNM,$CHKLT

BSTR:   JSR     R0,$CKARG
        .WORD   4
        CALL SETUP
        CALL MATCH
        CALL DONE
        RETURN

SETUP:  MOV     -(R5),-(SP)
        MOV     -(R5),-(SP)
        CALL    $CLRLT
        MOV     -(R5),LNG
        MOV     -(R5),PTR
        CALL    $CLRLT
        MOV     -(R5),R1
        MOV     -(R5),R0
        CALL    $POPNM
        CALL    $CHKLT
        MOV     (SP)+,(R5)+
        MOV     (SP)+,(R5)+
        RETURN

MATCH:  DEC     R0
        MOV     R0,-(SP)
        TST     LNG
        BEQ     40$
        CMP     R1,R2
        BLT     40$
        CMP     R2,LNG
        BLT     40$
        ADD     R2,R0
        MOV     R2,R1
        BR      10$
9$:     DEC     R0
10$:    MOV     PTR,R1
        BEQ     20$
```

```
15$:    DEC     R0
        CMP     R0,(SP)
        BLT     40$
        BR      10$
20$:    MOV     LNG,R4
        MOV     R0,R3
        INC     R3
21$:    DEC     R4
        BEQ     45$
        CMPB    (R1)+,(R3)+
        BNE     9$
        BR      21$
40$:    MOV     (SP),R0
45$:    SUB     (SP)+,R0
        RETURN
DONE:   MOV     R0,R4
        JSR     R4,$FMTNM
        RETURN
        .END
```

Now compile:

```
R DBL
VFSRCH=VFRCH
^C
R MACRO
BSTR=BSTR
^C
R LINK
SY:VFSRCH=VFSRCH,BSTR,DLIB
^C
```

Now run it:

```
R VFSRCH
```

Several people have indicated a willingness to share what they have on global searches, but as of this writing nothing has arrived. If I get something later, I'll publish it in a future newsletter. Send me yours.

MEMORY SAVER

You need lots of main memory to run DIRECTORY if you sort the output (/ALP, /SORT:category). Otherwise, only about 14 Kbytes is required. Enter the following as file DIRSRT.COM:

```
^!^>R SETSIZ
SY:DIR/D:64.
^C
^)^(DIR ^1
^>R SETSIZ
SY:DIR/D:0
^C
```

For non-sorted directory operations, use the DIR command as usual. For sorted directories, use DIRSRT, which temporarily allocates maximum memory for DIR.SAV before running it, then restores its original, smaller size. For example:

```
DIRSRT SY:*.SAV/SORT:DATE/REVERSE
```

If you are running DIR from a program or command file and do not know whether sorts will be required, use DIRSRT.

COMMENTS ON 5.1B

Everyone else must have gotten this version long before I did, so you may have all the glitches worked out. [Editor's note: Actually, John, we've about worn out 5.1C]. Probably the most devastating change is the lack of terminal support for TECO in command files. We have 33 quick TECO routines (literally programs) in TSX-Plus command files, many of which run from startup command files.

No matter how big our silos, input characters get dropped and the programs have to be rerun. Maybe help will come with the next version??? There could be some better information on how to set up silos.

We're still trying to find a way to end a CL program's input when no more comes in. The program is halted in IO and presently requires a reboot to flush. Some of our problems may be due to the lack of DBL support for some of the CL handler special functions. If it can be rigidly controlled, though, there are some very nice features with CL. One struggling point: Ever notice how many places you need to find to assemble the full set of CL documentation?

TSX-Plus has done some very nice things in 5.1. We'll have to give them time while we both work out some of the glitches. It should be well worth the effort.

DBL VERSION 4

DBL v4 will begin shipping around July. I had been assured by some that it would be out in January 1985, but haven't heard much of that nonsense since about March.

It is almost embarrassing to get excited over a new compiler. We used to become depressed about it -- those of us who had DIBOL v3b. I imagine there are still some who are running 3b rather than a new version because of the workarounds each version required in those days. But there were some of us that knew things had to get better. DIBOL has since become a solid product and is a pleasure to use. It is concise, programs simply, compiles quickly, is not as wordy as COBOL, and runs at a respectable speed if properly implemented.

I understand DIBOL is so solid it may become an ANSI standard. DEC used to say it outsold FORTRAN on their systems, but that may not be the whole story because few DIBOL systems are developmental.

Many of us went on to DBL, seeing it as a superset of DIBOL. Programming started to be more fun. And DBL has a lot of things that are worthless until you have to maintain a program (which will probably be 60% of the time spent on that program). It carries the principles of structured programming beyond the basics. DBL v1 was just DIBOL compatible. Version 2 added structured programming constructs. Skip v3 -- it went to VAX. That brings us to v4.

So, those of us using DBL under TSX-Plus are waiting for July and will be talking about many things that will be new. Here are some of them:

"Floating" environment: DBL will now run essentially the same wherever it goes. The environment "floats" as you move a DBL program around. We moved our VT-52 DIBOL programs to VT-100s under DBL without changing the code. Now we can move to RSX and our programs delete the old version of a file when we CLOSE, just as they do under TSX, but not as RSX usually does it. Also, TT: is TT: even when it should be KB:, if you know what I mean. The differences between systems when using DBL are minor.

Sort: This is a fast internal sort. Some of DBL's implementations do 1,000 80-character record sorts in 15 seconds, while the slowest computer took 25 seconds; this is RTSORT-level performance. DISC says there will not be a lot of difference in speed between RTSORT and their internal sort. They account for the difference by optimizing for the kinds of sorts that DBL is likely to be doing. There may be some functional differences, however, such as mapping.

Virtual environment: Like VAX, write your programs in a single dimension -- long. If you move your program to a VAX, the hardware provides virtual code and data management; while running under TSX-Plus, DBL does the same thing with software. DBL includes a compiler AND a linker that manages the amount of code and data that is really needed in memory at any time. It sets flags to control use, re-use, or freeing of memory-resident code and data. Its paging system uses a least recently used algorithm for maximum efficiency and minimum I/O overhead.

Assembly language interface: This excellent feature of DBL will be as well documented as before, enabling us to use high level DBL for I/O and assembly language for crunching. How it is done will change, though. For example, it now requires PIC, not object code.

Multi-key ISAM: The really nice thing about DBL's ISAM is that it is self-reorganizing. There was a time when I spent 14 hours a week reorganizing DIBOL ISAM files, believe it or not. Under DBL's v2 ISAM you could occasionally run out of stalls for data even though there was enough file space. This should no longer happen in v4 due to roomier index blocks and better insertion and removal algorithms.

SET: Initializes a whole batch of variables so your listing doesn't need two inches of =0s. Isn't that nice?

NOP: It will be nice to have a no-op in a string of sequential decimals in CASE and USING blocks.

SCAN: Use it in WHILE and UNTIL lines to look for either the beginning or the end of the condition. Do you have any idea of the number of lines of code this will replace? Remember how you did it in DIBOL v3b?

IF-THEN-ELSE and other structured extensions will still be present in DBL. By the way, there are many undocumented variations of IF-THEN-ELSE that caused some to avoid v2.2. Check them out before you say it can't be done!

Multi-dimensional arrays: This is what we had hoped it would be. It replaces the old computational method of addressing positions within a set of data. How many times has your code for addressing parts of a tax table looked like an engineering algorithm? It is nice to see this feature in a DIBOL-like language.

Fixed-point decimal: Now we can put a decimal point in our numbers.

Why is DBL being released so late? Those of us in the front line of this business know some of the problems that interfere with getting a product out. Ken Lidster of DISC gave the flow of how it went since I first knew about their version 4 plans back in 1983:

Before DISC was to start work on the TSX-Plus version of the new DBL, it wanted to have its micro program in full swing. The project took five programmers seven man-years and the time block slippage was 3-4 months; not all that bad if you consider the size of the project. Well, the TSX-Plus project was originally targeted for mid-1984, then refined to October 1984. Due to the micro slippage, it became January 1985.

Porting to TSX-Plus from MS-DOS ran into a major problem with the PDP-11 addressing limitation of 64Kbytes, affecting the memory resident overlay structure. A new front end to the TSX-Plus system had to be built to make DBL reasonable in size and performance; it would limit systems to 2-3 users under one alternative. Since the product could not be introduced that way, developing the new front end became more important than the delay. Ken Lidster's spacious office was crammed with four tables and three more lines were brought in. The front end was redesigned.

DBL will go to field test in June. Ken is quite confident it will ship in July. I'd sure hate to have to make a statement like that!

With all the effort put into the TSX-Plus version of DBL, what does DISC see for the future of RT-11? Milton Campbell's "RT-11 Perspective" column in the April Hardcopy brought up the question; he correctly made the distinction between the "end" of RT-11 and the end of enhancements to it. Lidster sees "no end in sight" because of TSX-Plus. There is no other small system with the features for the price of TSX-Plus. He sees a limit of ten users running "average" programs because of the RT-11 file structure, especially the directory system, which TSX-Plus has improved with its caching algorithms.

With products like TSX-Plus and DBL, the picture keeps getting better all the time!

DIBOL MESSAGE INTERFACE

DIBOL users have had to modify their programs somewhat to handle messages under TSX-Plus. A user has contributed (1) a routine that will handle DIBOL messages to RTSORT via SEND/RECV, and (2) a more general DIBOL message interface that allows sending and receiving messages with no chance of message misdirection. It is for use with DIBOL earlier than v8.00.

Contact me for directions on how to get a copy. It is well written and documented and would be an essential part of someone's DIBOL/TSX kit.

CONTRIBUTIONS

Send your routines (and money, if you'd like to) to:

John J. Stewart
Minn. Assn. of Comm. & Industry
480 Cedar St.
St. Paul, MN 55101
(612) 292-4684

CURRENT RELEASES

The latest release of TSX-Plus is Version 5.1C. For PDP-11/LSI-11 users, this release principally fixes several bugs left over in 5.1B, but it also makes TSX-Plus runnable on the PRO-380. If you are under support, you might want to pick up this version. To the best of our knowledge, there is no release currently under field test, although S & H is planning another new release in the near future (hopefully, we will see the end of 5.1x before x gets to Z).

V5.1C adds several new features. Chief among them is the ability to set character length and parity for DZ11, DZV11, DH11, and DHV11 ports via commands; running SYSMON on Hazeltine and ADM3A terminals in addition to VT52, VT100, and VT200 terminals; two new CL handler special functions; increased numbers of ACCESS and ASSIGN commands; a change to the DIBOL DTSUB routine to handle record numbers larger than 65,535; and a change to TSAUTH which inhibits display of account passwords.

V5.1C also corrects over 35 bugs in V5.1B ranging from stack overflows when using a parallel printer (LP handler) to incorrect SYSMON time displays when the system clock ran at 50 Hertz. As usual, S & H Technical Support has been responsive in acknowledging and correcting problems encountered with new releases.

```
        .TITLE  ONLINE - TELL EVERYBODY THE SYSTEM IS UP
;
;       COPYRIGHT (C) 1985 HORIZON DATA SYSTEMS
;                          1901 WILDFLOWER TERRACE
;                          RICHMOND VIRGINA
;
;       ONLINE SENDS A USER-DEFINED MESSAGE TO ALL TSX-PLUS TIME SHARING LINES
;       THAT ARE NOT PRESENTLY LOGGED ON.  ONLINE SENDS THE MESSAGE BY
;       ASSIGNING A FREE (EXTRA) CL UNIT TO EACH LINE IN TURN, AND WRITING
;       THE MESSAGE TO THE CL UNIT IF THE LINE IS NOT IN USE.  TO USE ONLINE,
;       THEREFORE, THE "CLXTRA" PARAMETER IN TSGEN MUST BE AT LEAST 1.
;
;       TO RUN ONLINE, USE THE FOLLOWING COMMAND SEQUENCE:
;
;               R ONLINE
;               <INSERT MESSAGE TEXT HERE>
;
;       THE MESSAGE IS PRECEDED BY <CR><LF> AND TWO BELLS, AND IS FOLLOWED
;       BY <CR><LF><LF> WHEN SENT.
;
;       YOU MUST DEFINE TWO PARAMETERS IN THIS PROGRAM BEFORE ASSEMBLING IT.
;       THE FIRST, "CLUNIT", IS SET TO THE UNIT NUMBER OF THE FREE (EXTRA)
;       CL USED TO SEND THE MESSAGE.  THE SECOND, "V51C" IS SET TO 1 IF YOUR
;       TSX-PLUS IS VERSION 5.1C OR LATER, 0 OTHERWISE.  WHILE RUNNING UNDER
;       5.1C, ONLINE USES .SPFUN 262 TO ENSURE THE MESSAGE HAS BEEN DELIVERED
;       TO EACH TERMINAL BEFORE REASSIGNING THE CL UNIT.  OTHERWISE, A TIMED
;       WAIT OF ONE SECOND IS USED BETWEEN SENDS TO PREVENT CONFUSION CAUSED
;       BY RAPID CHANGES TO CL ASSIGNMENTS.
;
;       THIS PROGRAM MAY BE FREELY COPIED OR MODIFIED BUT NOT FOR COMMERCIAL
;       PURPOSES.
;
;
;       USER DEFINED PARAMETERS
;
        CLUNIT=0                                ;USE CL0 TO SEND MESSAGES
        V51C=1                                  ;RUNNING UNDER V5.1C
;
        .MCALL  .EXIT,.GTLIN,.LOOKUP,.PURGE,.WRITW
        .IF     EQ V51C
        .MCALL  .TWAIT
        .IFF
        .MCALL  .SPFUN
        .ENDC
;
;       CONSTANTS
;
        BELL=7                                  ;BELL CHARACTER
        CHAN=0                                  ;USE CHANNEL 0 FOR I/O
        CR=15                                   ;RETURN CODE
        ERRBYT=52                               ;EMT ERROR BYTE
        LF=12                                   ;LINE FEED CODE
        NOLINE=3                                ;LINE DOES NOT EXIST
        OUTCHK=262                              ;OUTPUT CHECK .SPFUN CODE
;
```

```
START::
        MOV     #MSGBUF,R3                      ;POINT TO USER MESSAGE TEXT
        .GTLIN  R3,#PROMPT                      ;GET MESSAGE TEXT
1$:     TSTB    (R3)+                           ;LOOK FOR TERMINATING NUL
        BNE     1$                              ;KEEP LOOKING TILL FOUND
        DEC     R3                              ;POINT TO NUL
        MOV     #TRAIL,R0                       ;POINT TO TRAILER STRING
2$:     MOVB    (R0)+,(R3)+                     ;COPY TRAILER OVER
        BNE     2$                              ;  UNTIL TERMINATOR FOUND
        SUB     #MSG,R3                         ;GET MESSAGE LENGTH
        ASR     R3                              ;CONVERT TO WORD COUNT
        CLR     R2                              ;START WITH LINE 0
;
;       CHECK NEXT TIME SHARING LINE
;
3$:     INC     R2                              ;BUMP THE LINE NUMBER
        MOV     R2,LINE                         ;PUT IN CONTROL BLOCK
        MOV     #SEIZE,R0                       ;POINT TO CONTROL BLOCK
        EMT     375                             ;SEIZE LINE AS CL
        BCC     4$                              ;BRANCH IF WE GOT IT
        CMPB    @#ERRBYT,#NOLINE                ;SEE IF ALL LINES DONE
        BNE     3$                              ;TRY NEXT LINE IF NOT
        .EXIT                                   ;ELSE WE ARE DONE
;
;       OPEN A CHANNEL TO THE TERMINAL AND WRITE THE MESSAGE
;
4$:     .LOOKUP #AREA,#CHAN,#DEVBLK             ;OPEN THE CHANNEL
        BCS     3$                              ;IGNORE IF FAILS
        .WRITW  #AREA,#CHAN,#MSG,R3,#0          ;OUTPUT THE MESSAGE
        .IF     EQ V51C
        .TWAIT  #AREA,#TIME                     ;WAIT A SECOND FOR MESSAGE
        .IFF
5$:     .SPFUN  #AREA,#CHAN,#OUTCHK,#AREA,#1,#0,#0 ;SEE IF OUTPUT IS DONE
        TST     AREA                            ;CHECK CHARACTERS PENDING
        BNE     5$                              ;CHECK AGAIN IF NOT DONE YET
        .ENDC
        .PURGE  #CHAN                           ;CLOSE UP THE CHANNEL
        CLR     LINE                            ;LINE = 0 MEANS DROP CL
        MOV     #SEIZE,R0                       ;POINT TO CONTROL BLOCK
        EMT     375                             ;RELEASE THE LINE
        BR      3$                              ;TRY THE NEXT LINE
;
;       CL LINE SEIZE/RELEASE CONTROL BLOCK
;
SEIZE:  .BYTE   0,155                           ;EMT FUNCTION CODES
        .WORD   CLUNIT                          ;CL UNIT NUMBER
LINE:   .WORD   0                               ;LINE NUMBER TO SEIZE
;
;       DEVICE I/O CELLS
;
AREA:   .BLKW   6.                              ;EMT AREA
DEVBLK: .WORD   ^RCL0+CLUNIT                    ;DEVICE NAME FOR LOOKUP
        .RAD50  "TEMP  TMP"                     ;DUMMY FILE NAME
        .IF     EQ V51C
TIME:   .WORD   0,60.                           ;ONE SECOND TIMER BLOCK
```

```
        .ENDC
;
;       CHARACTER BUFFERS
;
MSG:    .BYTE   CR,LF,BELL,BELL                 ;MESSAGE HEADER
MSGBUF: .BLKB   81.                             ;FOR USER MESSAGE TEXT
        .BLKB   4.                              ;ALLOW FOR TRAILER
        .EVEN
TRAIL:  .BYTE   CR,LF,LF,0                      ;NULL TERMINATED TRAILER
PROMPT: .ASCII  "MSG>"<200>                     ;.GTLIN PROMPT
        .EVEN
        .END    START
```

ANSWERS PRICE LIST

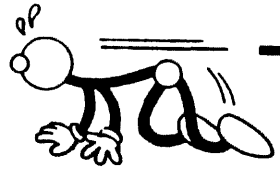
Answers ..... 75¢

Answers (requiring thought)..... 1.25

Answers (correct) 2.50

Dumb Looks Are Still Free

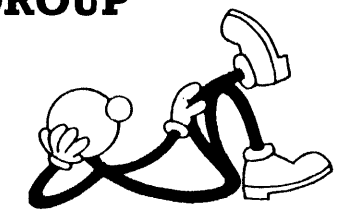
TSX—PLUS USER'S GROUP



Six Phases of a Project

- **Enthusiasm**
- **Disillusionment**
- **Panic**
- **Search for the guilty**
- **Punishment of the innocent**
- **Praise and honors for the non-participants**

TSX—PLUS USER'S GROUP



TSX-PLUS USERS' GROUP
C/O ZIA CORPORATION
P.O BOX 351
MORRIS PLAINS NJ 07950

ON-SITE MEMBERSHIP APPLICATION

Please type or print neatly. Limit data to 36 characters per line so it all fits on our mailing labels.

Name [NA]: _____

Title [TI]: _____

Department [DP]: _____

Company Name [CO]: _____

Address, Box, M/S [AD]: _____

_____

City, State, Zip [CT]: _____

Country [CN]: _____

Computer Types [CP]: _____

Storage Devices [MS]: _____

Printer Types [PT]: _____

Terminal Types [TM]: _____

S & H Software Used (check all that apply):

[ ] TSX [ ] TSX-Plus [ ] DPS [ ] RTSORT [ ] COBOL-Plus

Programming Languages Used (check all that apply):

[ ] MACRO [ ] FORTRAN [ ] PASCAL [ ] C [ ] BASIC

[ ] DBL [ ] DIBOL [ ] APL [ ] Other _____

PLEASE briefly describe what you would like the Users' Group to do:

Submit only one application per person. You may copy this form as required. Include payment with your application and send to the address at the top of this form. Memberships are $15.00 per year in USA, Canada, and Mexico, $20.00 per year elsewhere. All prices are in US $; foreign subscribers must remit by check or money order payable at a US bank. Submitted applications become the property of the TSX-Plus Users' Group.

TSXPUG
c/o Glenn Burnett
Zia Corporation
P.O. Box 351
Morris Plains, NJ 07950

FIRST CLASS
U.S. POSTAGE
PAID
PERMIT No. 13
Morris Plains, NJ 07950