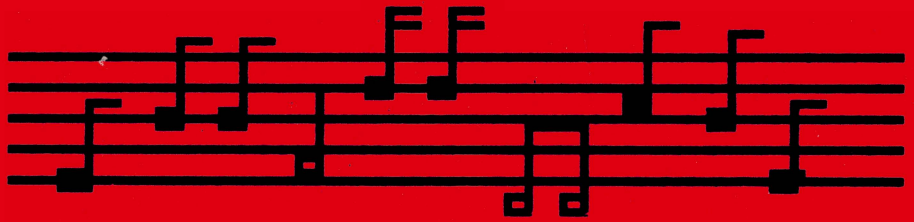

NOTES FROM MISOSYS

Issue IV - Decémber 1984

Table of Contents

NOTES FROM MISOSYS is a publication of MISOSYS, Inc., PO Box 239, Sterling VA 22170.

THE BLURB

Well, we made it to the fourth issue of NOTES. This is an important issue - lots of good stuff in it. Before I get too much into the BLURB, let me make the big announcement. I want to clean up the database and eliminate the names of individuals who no longer want to receive NOTES, or who have moved on without giving us a change of address, or those who have dropped out of the TRS-80 marketplace and are not interested in NOTES. Thus I am now asking for a positive response from each NOTES recepient in order to continue your name in the data base. THIS IS THE LAST ISSUE YOU WILL RECEIVE UNLESS YOU MAKE A POSITIVE AFFIRMATION TO US TO CONTINUE NOTES. There are a few ways achieve a positive response. 1) You can submit a registration card for a newly acquired product, or 2) you can send a short note containing the info which makes up your address label and ask us to keep you on. Notice that there is no need to make a purchase - I just want to ensure that the quantities of NOTES we are sending out are actually being used.

Here's an important announcement to all of our customers located in all of the countries which make up the Common Market. MISOSYS, Inc. and Molimerx, Ltd have arranged to cross-license our respective products for manufacture and sale beginning in 1985. Molimerx has been an important distributor of our product line for some time now. With the license to manufacture and sell MISOSYS products to the Common Market, I am sure that our European customers will get better service at more reasonable prices. MOLIMERX has an exclusive license to manufacture and sell into the Common Market. Therefore, if you reside in one of the Common Market countries, contact MOLIMERX or any of their dealers to obtain a MISOSYS product. You will not only receive your order faster, there should not be any import duty to pay. MOLIMERX can be reached at the following address:

MOLIMERX Ltd
1 Buckhurst Road, Town Hall Square
Bexhill-on-Sea, East Sussex
ENGLAND
Telex 86736 Sotex G Tel: (0424) 220391/223636

Charles C. Wright of San Ysidro, CA was dismayed to find that in Issue III of NOTES, I dropped the custom of using slashed zeroes to differentiate "zeroes" from "ohs". The reason for the omission of slashed zeroes in NOTES is that I am now printing NOTES with true proportional spacing (this is done for readability). NOTES is written using Model 4 SCRIPSIT patched with the MSCRIPT patch provided by us in DISK-NOTES-3. I also use the DW2PS/FLT proportional space and PSBF/FLT boldface filters (available at - $25 for the set). Using this approach, I have no facility yet for creating the slashed zero via a backspace and overstrike with slash. Maybe that's something which needs to be developed.

Some of you may know of the ball point pen solution to sticky diskettes. Jim Kyle has a solution to the problem of loose clamshells (they're the things which grip the hubs of the diskettes). When they loosen up a bit or if a diskette is a little tight in its sleeve, adding a second hub ring to the disk may add just the right amount of tension to cause the disk to turn freely. In my last stop at a Radio Shack computer center, I saw the hub ring kit still in stock.

Speaking of Radio Shack, I have been told by some of you that Radio Shack is no longer selling the cardboard disk mailers. MISOSYS to the rescue. If you have need for these mailers (and who doesn't), MISOSYS is now selling a 10-pak box of the cardboard mailers which are manufactured by Perma Products. These will be available only to customers within UPS shipping zones (i.e. continental US). They are the same style as Radio Shack's except that

there is no inside white sleeve.

George Geczy of JMG Software asked us why our software designed to run on the Model I under TRSDOS 2.3 requires the user to patch the DOS to be able to read our disks. That's a good question and a good topic. It's a topic that has been with me for quite some time. "Oldtimers" (you know, those hackers around the TRS-80 scene back in 1981) should remember the Data Address Mark discussions. George's question has raised the issue again. The root cause of Model I TRSDOS users needing to patch our Model I/III software disks is that they are duplicated on a Model III. The Model I uses a data address mark for the directory which cannot be generated on the Model III disk controller. Therefore, I worked up a one-byte patch to TRSDOS 2.3 which lets it read single density disks made on a Model III. Why do we use Model III's at MISOSYS for duplicating instead of a Model I? Let me respond.

"It has been over four YEARS since the Model I was manufactured. It is the only machine left that uses the old 1771 FDC data address mark [for the directory]. I cannot justify relying our duplication of media on the old Model I machines. We currently use Model III machines; however, in a pinch, we could easily use a 4 or a MAX-80. The IIIs, 4s, and MAX cannot generate the DAM needed by TRSDOS 2.3 on the Model I. Considering the proliferation of Model III and 4 machines, I sometimes have to even justify continuing to support the Model I with our software. For now, we are doing so. However, I do not think that it will continue for too much longer. No, I cannot use the Model I Data Address Mark and must continue to force the TRSDOS 2.3 user to "poke" in order to read my disks."

Now that you know why we don't use Model I's for duplication, that sort of leads us into the next topic - what's on the horizon from MISOSYS. Some of our customers have asked us our position on continuing product development for the TRS-80 machines and new developments on other machines - namely, the IBM-PC class. This is a very reasonable question. It's one that I have been asking myself for some time now. The microcomputer industry seems to be rapidly changing. Actually, the entire field of "techtronics" - a term I will apply to new technology in electronics - is itself changing rapidly. I am not even so sure that the 16-bit 8088/8086/80186/80286 series of machines will be around much longer. When it comes to this micro industry, the next few years will prove difficult ones for small companies like MISOSYS. It may be virtually impossible to find the specialized marketplace where we have operated succesfully up to now. However, we intend to continue supporting the TRS-80 low-end marketplace - as long as it's out there and as long as we can generate enough business to have some growth. That last point is an important one!

We have intentions of developing products for MS-DOS - initially on the Tandy Model 2000. MISOSYS does not have the resources to join the fray in the IBM PC world; thus, we expect to maintain the continuity to MS-DOS through the 2000. WE have had a 2000 sitting here since December 1983. I have said in the past that I would expect to see some product(s) developed in C by the end of 1984 - that's why the machine was purchased. Well we're at year end 1984 and this has not been accomplished. We will be getting some projects underway for it next year - that's a promise. In 1984, we have been too busy getting new software packages ready for the Models III and 4. Witness the delivery of TRS-80 Model I/III/4 software this year: ADE/PRO-ADE, IFC/PRO-IFC, PRO-LC floating point, LC/LIB enhancements, LCOPY, DD&T/PRO-DD&T, DW2PS/FLT, DESCRIBE, SAID, EDAS 4.3, PRO-ZSHELL, and more.

Right now, there are some heavy projects underway for the Models I, III, and 4. For 1985, I expect to release MRAS - a relocating macro assembler package compatible with Microsoft /REL files. I also expect to release LC/PRO-LC Version 2.0 - a full K&R C compiler. There will be fairly liberal

update policies on both of those products. We will also be developing a package, possibly named SYSLIB, designed to give you complete customization of the module makeup for the LDOS 5.1.4 and TRSDOS 6.2 system libraries. Karl is also working on a 128K Model 4/4p and compatible package tentatively called PRONTO. It will provide pop in windows and permit you to add your own modules to its application function handler. There is more on this in the new products section of NOTES. We will also be putting out some CP/M and CP/M+ software for the Model 4 and MAX-80. We do expect to be busy!

I occasionally get asked for machine recommendations. For my money, I would always recommend that a client obtain that equipment which satisfies the needs of the business. If the business need requires a half meg machine to run that big data base or 20,000 cell spreadsheet, then yes, get the big machine. If an 8-bit Z-80 based machine satisfies the bill, then continue with it. Too often, one jumps at the flashy new machine - because it is new. My personal opinion is that as long as the 8-bitters can economically fill a need, they are still good tools. I must admit that the biggest software houses are only supporting IBM and compatibles (and now the Macintosh); however, many of these are companies that were not even around prior to the release of the PC. You still have good solid companies supporting the Z-80. I feel that we are one of them and we expect to continue our support of the [8-bit] Tandy machines until they stop making them. Look for the price of the Model 4/4P to continue dropping making it even more of a bargain.

From time to time I have made mention of "The Hitchhiker's Guide to the Galaxy". For example, I told someone that Karl Hessinger obtained a Model 2000 and is calling it "deep thought". You must be a Hitchhiker's Guide buff to detect the underlying meaning of all this. As another example, I have mentioned "The Programmer's Guide" not to be confused with "The Hitchhiker's Guide". These references may puzzle a few of you. Well I know that I am not alone in my affinity to the Guide. It turns out that Infocom, with assistance from Douglas Adams - author of the Guide, has converted "The Hitchhiker's Guide to the Galaxy" onto disk as a computer game. No word yet as to which computer the game is to be supported under although a picture of Douglas Adams (author of the book) and Steve Meretzky (author of the program) showed them sitting in front of a Mac. If that's true, it's going to be one game which will cause me to have to get the apple. As a side note, the Infoworld column which noted this subject mentioned, "For some reason, documentation in the goofy packaging keeps telling us not to forget our towels, necessary equipment for this Cosmic Hitch. We're just a little nervous about booting the disk." I won't relate the significance of the towel - you'll have to read the book - or play the game!

I received an inquiry from John H. Deal of Naples FL concerning his need to change the use of the Model 4 function key's. John wanted F1, F2, and F3 to perform the functions of STOP/START scrolling, Clear screen, and Print Screen respectively. I wrote John a rather lengthy reply which explored some of the history of the Model 4 keyboard and what alternatives he had, if any. I believe my response to him may prove useful to some of my readers. "I can understand your suggestions concerning the generation of key codes and the use made of certain keys for certain functions. Let me share a little history with you.

I understand too well the importance of adequate keyboards. As the system analyst associated with LSI during the design of TRSDOS 6.x, I fought tooth and nail with Radio Shack for a full keyboard for the Model 4. LSI was first approached by Tandy for a DOS in mid-1982. Our first exposure to the Model 4 project was after they had a prototype (and even had thousands of keyboards on hand). I was aghast that they chose only to add the three function keys, control and caps. Their original keyboard design was firm.

The task at hand was how to squeeze in the maximum amount of functions into that primitive keyboard. First, we had to have a full ASCII character set. That's why you have the somewhat contorted depressions to generate the following characters: [\]^_{|, tilde, and DELETE. Next, the cursor movement keys (the four arrow keys) had to generate codes that were not printable ASCII characters. There was also a desire to remain as close as possible to the Model I and III codes used for the arrow keys. Only the unshifted up-arrow key was changed to eliminate the X'5B' code it generated on a Model I/III (that code is the left bracket). BASIC had to use the carat (^) since it's the proper character for indicating exponentiation and does not inhibit the '[' from use.

The Model 4 was designed with keyboard type-ahead. Thus the PAUSE and BREAK functions are interrupt task processes. SHIFT-@ was traditionally used for PAUSE. The function keys were expected to remain available for user applications and were not to be used internally for any functions. They generate 81H, 82H, and 83H (as well as 91H, 92H, and 93H for shifted keys). Since PAUSE is a task scanned operation, it hard codes the detection of the SHIFT and "@" keys in the keyboard matrix. It would not be a simple job to recode this and I would not care to spend time in such an endeavor.

As far as the UP-ARROW moving the cursor up one row, that is easily done in an application that is screen-oriented. If you are suggesting that this be made available in BASIC, that is something that needs to be taken care of in BASIC itself. The UP-ARROW is used to move the cursor up in all applications that I am using which permit cursor manipulation: PRO-CREATE, SCRIPSIT, PRO-ZGRAPH, and PRO-CESS.

Now then, what's the solution? First, we do have available a program that alters the code values generated by the function keys. This [has been] provided to our customers in the publication of NOTES FROM MISOSYS, Issue III. If you own and register a MISOSYS product, you will be able to obtain that issue. Next, it would be a simple matter to write a keyboard filter that detects the function key code for F1 and sets the PAUSE bit in the KFLAG$; detects the F2 key and issues the two bytes necessary to clear the screen (or converts to the SHIFT-CLEAR code). Since the screen-print function is internally driven, modification of the DOS would be necessary to implement that [see, if the keyboard included a PRINT key as on the 2000, you would have no problem]. It would be possible to construct a filter for a particular release of the DOS which could invoke the internal screen print routine after searching the driver area code for its exact location."

In the last issue of NOTES, I asked for comments relative to full-size 8-1/2 x 11 documentation vs the smaller 5 x 8 size. Surprisingly, the tally for each was close with full size getting the edge. You may be interested in some of the comments I received.

"On the subject of the small 'IBM-style' binders: NO! My shelves are set up for full-sized binders already. They fit in well with my collection of magazines and journals. Furthermore, it is easy for me to add my own supplementary documentation and listings to the full-sized binders; the small binders would require a new format for my printer as well as hard-to-get and expensive paper.

Another reader writes, "In response to your question about the 5-1/2 x 8-1/2 documentation, I think it's great. The smaller size makes for easier storage space and easier use when desk top space is limited.

Another, "Regarding your query in NOTES about document sizes -- For reference manuals, I prefer 8-1/2x11 as I like to use standard 3-ring binders compatible with the LDOS and LC manuals and with computer print-out. However,

please keep NOTES in its present size so that they can be kept together."

Still more response, "PLEASE stick with standard, 8.5" x 11", 3-hole, single or double sided printed material whenever possible. Although this size is somewhat unwieldy compared to the smaller, 'quick-ref' type of documentation that's showing up, I think that it's vastly more useful since maintenance is so much easier. Namely, additions, deletions, miscellaneous notes & hints, and expansions can all be placed into or removed from the documentation with relative ease. I keep NOTES in a pouch page with my EDAS doc."

Well, the verdict is not in; however, at MISOSYS, we'll continue with full-size documentation for now.

I have had requests from some NOTES readers to have a price list included in each issue. Well this issue has one. You will find it in the NEW PRODUCTS section. The prices contained in the list are formally effective starting January 1, 1985. They will appear in our 80 MICROCOMPUTING ad in the February issue which comes out in January. ALL NOTES READERS MAY TAKE ADVANTAGE OF THE 1985 PRICES IMMEDIATELY! Spend a little time looking over the new prices. You will find big DECREASES in price in almost every one of our packages. If you discern that the prices are slashed to the bone, you are about right. For us, 1985 will be the make or break year for the Model I/III/4 market. I'll be quite frank to all of the people who have supported MISOSYS since 1978. There is either too much theft of our software or too few customers out there. I am going to have to see an improvement in the TRS-80 market in order to be able to remain in this business in 1986. Our new price levels were designed to make our software comfortable to purchase. I don't think anyone has any qualms about the quality of our products. Nuff said! The most I can ask of my loyal customers is that if you like the quality and utility of our products, let your friends and associates know. Word of mouth is the best advertisement. If you are dissatisfied with our products, let us know. We'll try to accomodate you.

The last item to note about our products is NOTES itself. Don't forget that all of the program listings and fixes published in this issue of NOTES is available on DISK NOTES. The disk is available in one format only - 40D1 (which means 40 tracks double-density single-sided) LDOS-5/TRSDOS-6 data disk media format. DISK NOTES 4 is available for $10 within the United States and Canada. Outside of this territory, the price is $12. DISK NOTES is also available for issues II and III of NOTES. Original printed copies of NOTES FROM MISOSYS are still available for the following prices: Issue I - $2, Issues II or III - $3.

The last item I usually discuss is the status of the family. Are you ready for this? Here's a surprise for most of you. Stefanie Diane Soltoff was born on October 2, 1984. Stefanie weighed in at 8 pounds, 14 ounces - Brenda was really relieved after delivery. As I write this, that makes Stefanie a little over one month. Stacey just turned 17 months. She started walking pretty good at about 13 months and I can hardly remember her when she crawled. My how soon they forget about that! At the three-week weighin, Stefanie was just about ten pounds. While we had the opportunity of an "accurate" scale, Stacey was weighed and was up to around 25 pounds.

I get a chuckle now when Brenda talks about the "kids". The plural sound gets me. In the last NOTES, I mentioned that we were scheduling a move. Well it came off with a just a little hitch. It was delayed about a week forcing us to move our furniture prior to ourselves - I had to get all of the stuff out so that the old house could be repainted. That's not as bad as the problems Dennis Brent (PowerSOFT) had in getting to his new house. In any event, Stacey just loves running around the whole place. The neighborhood is full of little kids (of all ages). We now have lots of room for the family and the

business. However, I have another big job this winter in remodeling the office quarters from being an unfinished 1500 square foot of space to a multi room headquarters for MISOSYS. Don't worry, we'll still find the time for our programming projects.

MISOSYS Products Price List - Effective January 1, 1985

ADE $39.95	CMD-FILE $24.95	CON80Z $24.95
CONVCPM $39.95	DD&T $24.95	DESCRIBE $24.95
DW2PS/FLT $24.95	EDAS 4.3 ..... $74.95#	GRASP $24.95
HartFORTH ..... $74.95#	IFC $24.95	LC 1.2 ..... 124.95*
LCOPY $39.95	MACH2 ..... $24.95	MLIB $24.95
MSP-01 ..... $24.95	MSP-02 ..... $24.95	PaDS $24.95
PRO-ADE $39.95	PRO-CESS $24.95	PRO-CON80Z $24.95
PRO-CREATE .... $74.95#	PRO-CURE $39.95	PRO-DD&T $24.95
PRO-DESCRIBE .. $24.95	PRO-DUCE $24.95	PRO-ESP $24.95
PRO-GENY $24.95	PRO-HartFORTH . $74.95#	PRO-HELP $14.95
PRO-IFC $24.95	PRO-LC $124.95*	PRO-MACH2 ..... $24.95
PRO-MLIB $24.95	PRO-PaDS $24.95	PRO-SAID $24.95
PRO-XFTS $24.95	PRO-ZCAT $24.95	PRO-ZGRAPH $24.95
PRO-ZSHELL $24.95	SAID $24.95	SOLE $14.95
VRHARD $74.95	X-FTS $24.95	ZCAT $24.95
ZGRAPH $24.95	ZSHELL $24.95	

US/CANADA Shipping: * add $5, # add $4, All others add $2 each
Outside US/CANADA: We ship AO Air - multiply US/CANADA rates by 4.
Common Market Countries: Contact MOLIMERX Ltd.

C PROG LANG. by K&R . $21.95 PROGRAMMER GUIDE TRSDOS 6 by Soltoff .. $14.95
Shipping: $2 per book

MISOSYS 3/4" Binder .. $9.95 Perma Products DISK MAILERS 10-pak $9.95
Shipping: binders: $3 for 1, $1 each additional; mailers $2 UPS ground

CENTECH DISKS - Box of 10 - Clear plastic flip box - Specify COLOR .. $24.95
Available colors: red, orange, yellow, green, dark blue, burgundy, lavender, white; BUSINESS pak: 2 each brown, beige, burgundy, gray, dark blue; RAINBOW pak: 2 each red, orange, yellow, green, light blue.
Shipping: $2 for 1 box, $0.50 each additional box (UPS surface).

MISOSYS, Inc. PO Box 239 Sterling, VA 22170-0239 703-450-4181
We accept Check/MasterCard/VISA/CHOICE - Call or write for catalog

NEW PRODUCTS

The following section describes the products that are newly released from MISOSYS or are about to be released. Unless otherwise noted, software products named beginning "PRO" are for use under TRSDOS 6.x or MAXDOS 6. All others are for use under LDOS 5.1. Products designed to function under TRSDOS 1.3 or 2.3 and others, such as DOSPLUS, MULTIDOS, etc., will be specifically noted as such.

Notes from MISOSYS

DD&T/PRO-DD&T - by Richard N. Deglin

The DD&T package provides you with a major enhancement to the debugger supplied with your DOS. The Debugger Disassembler (DD) module is completely relocatable. Its interface to the DOS debugger is automatic. The TRACE utility can be used to help hone your programs to optimum efficiency. DD and TRACE together make DD&T a set of fine tools crafted to provide you with a rewarding assembler programming experience.

DD provides the capability of an on-line disassembler during your program debugging sessions with DEBUG. Once invoked, it resides in a high memory module interfaced to the system's DEBUG module - available at the touch of a button. DD may be SYSGENed if you find it convenient to have the Debugger Disassembler available at each boot. The high memory module takes less than 1600 bytes of memory. In the upper right hand corner of the debugger register display ("X" mode), DD will display a mnemonic disassembly of the Z80 instruction which the current program counter (PC) points to. DD will update this instruction disassembly automatically each time DEBUG refreshes the display screen.

DD also provides for Z80 disassembly directly from any memory location in your computer's 64 Kilobyte address space via the added "Z" DEBUG command. "Z<ENTER>" will start the disassembly from the current program counter while "Znnnn<ENTER>" will start the disassembly from the hexadecimal address given in the command as "nnnn". In each case, 16 or 24 lines of disassembled instructions will be displayed, depending on whether you are operating version 5 or 6 of the DOS. You may continue the display with one or more additional screenfulls of disassembly by typing any keystroke - type an "X" and you will be returned to the prior debugger display screen.

Programs may be coded so that they waste too much time in inefficiently written program routines. TRACE is a package of two utilities designed to help you optimize the design and coding of an assembler application program. It consists of two utilities; PTRACE which records program activity by execution address ranges and, STRACE which compiles and displays the statistics of a series of PTRACE runs.

PTRACE allows you to execute the target program in an environment which maintains a record of all program activity. Your selected address range is divided evenly into 256 "buckets". PTRACE keeps a counter for each bucket. Two other buckets are used to track activity in areas above and below the selected range. Every time a system heartbeat interrupt occurs, PTRACE determines the program counter at the time of interrupt and updates the counter corresponding to the correct bucket. While the trace is active, PTRACE will place a blinking asterisk (*) in the upper right corner of the video screen.

STRACE will report statistics computed from the data in one or more PTRACE result files. The report can be redirected to the printer by appending ">*PR" to the command line, or to a file by appending ">report file" to the command line. If a consecutive series of "buckets" within the trace range all have a count of zero occurrences, they are squeezed together to form one "bucket" in the output listing.

DD&T: For LDOS 5.1 - $24.95
PRO-DD&T: For TRSDOS 6.x - $24.95

Note: If you purchased DD&T or PRO-DD&T from our October sales flyer for $30, take a $5 credit on your next order placed with us.

DESCRIBE/PRO-DESCRIBE - by Roy Soltoff

DESCRIBE is a tool to extend your disk directory with a descriptor field for each file. The field is 63 characters in length and is used by you to add information describing each file stored in the directory. DESCRIBE provides commands to manage these descriptors as well as provide you the means to construct customized sorted directory displays to the display screen, your printer, and even a disk file. DESCRIBE also has a command to allow you to invoke a DOS command as if you were at DOS Ready. DESCRIBE is menu driven and very friendly. The extension to the directory should be totally transparent to the DOS and all applications and utility programs which access the directory - except for DESCRIBE itself. While the DESCRIBE main menu is displayed, commands are available to CHANGE to a new diskette, ALTER existing or add new descriptors, MODIFY the user-customized directory display, invoke any DOS command, SEARCH for all files in a directory which match a descriptor string of characters, obtain HELP information on each command, display a FILE directory, REMOVE the descriptor extension from the DOS directory, and create or load directory data from a Data Interchange Format (DIF) file.

More specifically, the ALTER command displays a file's name along with the MOD date and attributes. If any descriptor field is present, it will also be displayed. The UP/DOWN/ENTER keys permit you to scroll through each filespec. If you wish to edit a descriptor field, depress the letter "E". Editing keys permit you to DELETE the character at the cursor and shrink the line by one position; EXPAND the line by one character; MOVE the cursor left or right by one position; and either STORE or ABORT changes and proceed to next file.

If CHANGE is invoked while updated descriptor data remains to be written to the directory, DESCRIBE automatically updates the directory. The program attempts to ensure that the disk to update is the same as the current one by matching the disk pack identification. This guards against you switching disks prior to invoking the CHANGE command. Although this is not failsafe, it does give that extra measure of safety if you dilligently maintain unique disk names for all of your diskettes.

The directory listing display is formatted according to your defined format specification which defaults to our standard until you change it. This specification can be customized via the "M" command. The listing will be titled if the first character of the specification is a plus sign ("+"). The title is constructed with headings appropriate to each data item in the listing. A line of dashes completes the title. The listing format for each file is controlled by the sequence of keywords and other characters in the format specification. All keywords are displayed while the "M" command is operating. DESCRIBE supports the following keywords: ATT, DAT, DES, DEX, DRV, EOF, ERN, EXT, EXX, LRL, NAM, PRO, REC, VNM, VDT, VID, and SPC. They permit you to extract practically all of the information contained in a disk's directory of files.

The INTERCHANGE command allows you to load or save the directory data identified by the format specification from/to a data file. The file is structured in the Data Interchange Format (DIF). The load operation will extract the descriptor fields from the DIF file loaded and update the current set of descriptors when the file specification matches. The load operation requires that the following fields are present in the DIF file: $DES and $SPC or $DES and $NAM (or $NAX) and $EXT (or $EXX). An error will be generated if the required fields are not present. The save operation creates a DIF file in column format.

The SEARCH command allows you to invoke a file directory listing which includes all files with a descriptor character string that matches your search string. Your search string can be up to 32 characters in length. The

matching is performed without regard to UPPER/lower case (it is case insensitive).

DESCRIBE also allows you to obtain the directory display for a disk via the command: "DESCRIBE ambigspec", directly from DOS Ready. Ambigspec is considered to be an ambiguous file specification. This takes the form of a file name field, a file extension field, and a mandatory drive specification. Within the filename and extension fields, the character "*" will match all other characters remaining in the field while the character "?" will match all other characters in that position. If the filename field is blank, it will default to "*". If the extension field is blank, it will match only a file with no extension. If you omit both fields, the ambigspec defaults to */*:d.

DESCRIBE: For LDOS 5.1 on Model I/III.
PRO-DESCRIBE: For TRSDOS 6.x

X-FTS/PRO-X-FTS - by Rick C. Francis

X-FTS/CMD is a general purpose File Transmission System utility for use with Model I/III LDOS 5.1 while PRO-X-FTS is for use with either TRSDOS 6.X or DOS PLUS IV [We will use the term, FTS, to describe both products in the following information]. FTS allows you to send any file to another computer error-free via the RS-232. The program is compatible with the Ward Christensen or XMODEM protocol which is very popular with CP/M and MS-DOS users.

FTS may be used to transfer any file to another computer provided that the other computer is running FTS or a compatible program (i.e. XMODEM under CP/M, MODEM-80 under TRSDOS/LDOS). It can operate from direct user keystrokes or from a Job Control Language file. We will refer to your computer as the "local" computer and the other computer as the "remote" computer. The two computers may be side-by-side connected directly to each other with an RS-232 cable or they may be hundreds of miles apart connected by telephone lines. During the transfer, each computer's screen will show the number of each block as it is transferred (unless the "quiet" mode is invoked). It will also indicate the number of errors detected during the transfer. Each time an error is detected, the block is re-transmitted until the block is sent error-free (or until the number of retries is exhausted). When the file transfer is complete, each computer will be automatically returned to DOS Ready or to an application if invoked from one.

FTS supports many parameters which customize its operation according to your tastes. The ABS parameter may be used to force FTS to replace an existing file without prompting the user when receiving a file.

The BLOCK parameter allows the user to set the size, in bytes, of the data blocks. The sending and receiving side must use the same value for this parameter. The default block size is 128 bytes which is the block size used by the XMODEM protocol.

FTS normally uses the device "*CL" for RS-232 communications. If you use a different device name for the RS-232, you can specify this with the DEVICE parameter. Under DOS PLUS IV, the RS-232 device is defined as "@RS" and the DEVICE parameter has no effect.

The CP/M operating system does not maintain its end-of-file marker down to the byte level. CP/M files are always some integer multiple of 128 bytes. Usually, a CONTROL-Z is used to mark the end of a text file but binary files have no byte-level EOF marker. The XMODEM protocol was first designed for CP/M systems so it provides no method of maintaining the EOF to the

byte-level. Under the TRSDOS family of operating systems, byte-level EOF is maintained in the directory. By using the EOF parameter on the sending side, FTS will maintain the EOF marker during the transfer. EOF mode detection is automatic on the receiving side.

The KEY parameter can be used to encode data being sent or decode data being received. If it is necessary to use a public access bulletin board to send a private file to another user, the sender can encode the file with KEY=N where N is a number from 0 to 255. At some later time, the receiver can download the private file from the bulletin board using the same KEY value used to send the file. The data encryption method is a very simple one and could be broken fairly easily. However, it should deter the casual snoop.

The XMODEM protocol provides no means of passing file attributes (i.e. protection level, logical record length, visible/invisible status, etc.) from the sender to the receiver. The LRL parameter is used in receive mode to set the logical record length of the received file if it does not exist on the receiver's system. This value can be obtained from the sender's directory. The default value for LRL is 0 (256 byte logical record length).

Under the "PRO" version operation, if the NOTIFY parameter is specified, FTS will provide an audible indication of the transfer status at the termination of the transfer. If the file transfer was successful, the beep(s) will be short and high pitched. If the file transfer was aborted due to some error, the beep(s) will be of longer duration and lower pitch. This parameter is ideal for long files or for multi-file JCL controlled transfers.

When FTS is running on a "HOST" computer, output sent to the display also goes to the RS-232 device. Since FTS normally displays the block number and error count on the screen, this information would be intermixed with the data being sent to the RS-232. Therefore, when FTS is being run on a computer in host mode, the QUIET parameter is specified to prevent this intermixing of data. Since FTS will be used quite often by a computer running in host mode, it is set-up to allow the user to change the program to force QUIET mode just by renaming the program to XFTS.

The RETRY parameter allows the user to set the maximum number of attempts that should be made in sending any one block before the transfer is aborted. This parameter also directly affects the amount of "silence" that can occur between blocks before FTS declares a "time-out error". The default value for this parameter is 9 which allows up to nine attempts to send a particular block and allows about 25 seconds of delay between blocks. Specifying RETRY=0 will allow an infinite number of block-send attempts and about 6400 seconds of delay between blocks.

Under TRSDOS 6.2, you can easily invoke FTS while communicating with another computer directly from the COMM program supplied with the DOS. The COMM program uses only the hardware interrupt function provided by the Radio Shack RS-232 interface to receive incoming characters. On entry to FTS, the COMM interrupt function is disabled so that no characters are "stolen" by the COMM program. Since FTS uses the Library Overlay Region of memory (x'2600' - x'3000'), it can be invoked from within COMM by requesting a DOS command. When the transfer is complete, control will be returned to COMM. Under all versions of TRSDOS 6, FTS may even be invoked during the running of a BASIC program with the SYSTEM command.

X-FTS: For LDOS 5.1 Model I or Model III
PRO-X-FTS: For TRSDOS 6.x or DOS PLUS IV

LCOPY - by Richard N. Deglin

LCOPY is a powerful tool which runs under certain CP/M systems and allows you to transfer files from various TRS-80 diskettes formatted by LDOS 5.x or TRSDOS 6.x onto selected CP/M formatted disks. You can obtain TRS-80 disk directories, transfer files, and invoke CP/M commands easily and rapidly from the program's menus; these functions are also accessible directly from the CP/M prompt.

The LCOPY disk includes a version for both the Lobo MAX-80 CP/M 2.2 and CP/M 3.0 version (LCOPYMAX) and the Radio Shack Model 4/4P CP/M 3.0 version (LCOPYM4). The disk is a 40-track single-sided double-density 256-byte sector Lobo MAX-80 CP/M 2.2 formatted data diskette, which is readable on both the MAX-80 under CP/M 2.2/3.0 and the Model 4/4P under CP/M 3.0. Certain TRS-80 formats may not be supported on the Model 4/4P under CP/M 3.0 [specifically double-sided, 80-cylinder, or 8-inch formats]. The following TRS-80 diskette formats are directly supported by LCOPY (where hardware permits the operation) 1) 8-inch, single or double-sided, double-density only; and 2) 5-inch, single or double-sided, single or double-density with 35, 40, 77, or 80 cylinders.

LCOPY provides three major functions under CP/M - the DISPLAY of a DOS disk directory, the TRANSFER of file(s) from the DOS diskette to a CP/M disk, and the invocation of a CP/M command from the LCOPY menu.

The <D>isplay directory command reads the target TRS-80 diskette and then displays a directory of all or some of the files on that diskette, sorted by FILE NAME and EXTENSION. The target disk is identified by either a DOS drivespec or partspec. You simply key in the letter of the drive in which the TRS-80 diskette is mounted, for example <B:>, to display all of the files. Alternatively, you may enter a TRS-80 partial filespec, such as <B:ABC*/ASM>, to display only those files which match your ambiguous spec. Optional parameters, such as SYS, INV, VIS, MOD, DATE, etc., may be entered in a fashion similar to DOS command line parameters to select particular TRS-80 file attributes or obtain a copy of the TRS-80 directory on your printer device.

The <T>ransfer files command allows you to move files from your TRS-80 diskette to a CP/M disk. This is the most powerful command in LCOPY. The DOS disk is identified by drivespec or partspec in the same way as the corresponding prompt in the <D>isplay directory command; you can enter a TRS-80 PARTSPEC, a TRS-80 DRIVESPEC, or <CTL-C> to cancel the command. The destination drive is identified with the letter of the drive onto which you want the TRS-80 files copied, such as <A:>. This must be an enabled CP/M drive (hard disk or floppy). Parameters are also available for the <T>ransfer files command to select particular TRS-80 file attributes, transfer all OLD or NEW files, ERASE any previously existing destination CP/M file, select a particular CP/M user area as the target of the file transfer, or have LCOPY QUERY you for the transfer of each TRS-80 file.

The <C>P/M command function provides access to operating system commands from the menu level. A second menu will be displayed which provide subordinate commands to <D>isplay a CP/M disk directory, <E>rase a CP/M file, <R>ename a CP/M file, <C>opy a CP/M file, <T>ype a CP/M file, <L>og a new default CP/M drive, change to another <U>ser area, or e<X>it back to the main menu.

Several functions of LCOPY may be invoked directly from the CP/M prompt without going through the program menu. When LCOPY is invoked in this way, the current default CP/M drive may be designated as the drive to contain the TRS-80 diskette. You can obtain the directory of a TRS-80 diskette with:

LCOPY <TRS-80 partspec> [DIR,<parm>,<parm>,<parm>,...]

You can transfer files from a TRS-80 diskette to a CP/M disk with:

LCOPY <TRS-80 partspec> <CP/M drive> [<parm>,<parm>,...]

LCOPY also includes CVTEXT - a tool to add the required LINEFEED to form the CP/M end-of-line sequence, CR-LF. This will be useful for post-processing TRS-80 format text files after a file transfer.

LCOPY: For Model 4/4P CP/M 3.0 and MAX-80 CP/M 2.2/3.0

HARTFORTH/PRO-HARTFORTH - by A. M. Graham, licensed from MOLIMERX, Ltd.

HARTFORTH is a full FORTH that conforms totally to the 79-STANDARD. It is not a modified FIG-FORTH but an entirely new implementation internally designed around the 79-STANDARD. The Model I/III version of HARTFORTH uses the conventional technique for FORTH implementation of indirect threading. It may be configured to run on either a TRS-80 Model I or Model III by a simple high-memory-pointer change. The TRSDOS 6.x version has been rewritten as a Direct Threaded implementation of FORTH which provides a greater execution speed in the range 10%-40% depending on the details of the actual program In addition to all of the 79-STANDARD required words, the HARTFORTH kernal contains some additional useful words and utiltites which turn HARTFORTH into a full-fledged FORTH development system.

Unlike many implementations of FORTH, HARTFORTH is designed to run under an operating system. The Virtual Memory that it accesses for storage and retrieval purposes is not the disk medium directly as would be the normal case with other FORTHs, but is a normal DOS file that is requested by the FORTH system when it is first entered. Doing this has several advantages in that it provides for FORTH files to be used in other language application programs and vice-versa. The fact that FORTH is running under an operating system is totally transparent to the programmer or user. Enhancements to the 79-STANDARD have been built into the HARTFORTH kernal in the form of functions to call the standard operating system file handling routines so that other files may be created and accessed, if required [this is not pure FORTH practice but blends in within the operating environment of HARTFORTH].

The virtual memory facility in HARTFORTH has two 1024 byte block buffers that are re-used on a least recently accessed basis. The terminal input buffer (BLOCK 0) is 80 bytes long. Filling it will cause an automatic ENTER to be generated by the 80th character. Both the KEY and EMIT words of HARTFORTH are defined as simple secondaries of:

": KEY KEY.PRIMITIVE ;" and ": EMIT EMIT.PRIMITIVE ;

so that input and outut can be vectored to another device as required by 'ticking' the Code Field Address of a word that drives the required peripheral device into the Parameter Field of either KEY or EMIT, as required.

The FORTH kernal includes extra functions over and above those required by the 79-STANDARD. The following lists note the words available to the HARTFORTH dictionary.

Stack manipulation: DUP, DROP, SWAP, OVER, ROT, PICK, ROLL, ?DUP, >R, R>, R@, DEPTH

Comparison: <, =, >, 0<, 0=, 0>, D<,U<, NOT

Arithmetic and Logical: +, D+, -, 1+, 1-, 2+, 2-, *, /, MOD, /MOD, */MOD, */, U*, U/MOD, MAX, MIN, ABS, NEGATE, DNEGATE, AND, OR, XOR

Memory: @, !, C@, C!, ?, +!, MOVE, CMOVE, FILL

Control structures: DO...LOOP, I, J, LEAVE, DO...+LOOP, IF...(true)...THEN, IF...(true)...ELSE...(false)...THEN, BEGIN...UNTIL, BEGIN...WHILE...REPEAT, EXIT, EXECUTE

Terminal input/output: CR, EMIT, SPACE, SPACES, TYPE, COUNT, -TRAILING, KEY, EXPECT, QUERY, WORD

Numeric conversion: BASE, DECIMAL, ., U., CONVERT, <#, #, #S, HOLD, SIGN, #>

Mass storage input/output: LIST, LOAD, SCR, BLOCK, UPDATE, BUFFER, SAVE-BUFFERS, EMPTY-BUFFERS

Defining words: : xxx, ;, VARIABLE xxx, CONSTANT xxx, VOCABULARY xxx, CREATE...DOES>

Vocabularies: CONTEXT, CURRENT, FORTH, DEFINITIONS, ' xxx, FIND, FORGET xxx

Compiler: ,, ALLOT, ", IMMEDIATE, LITERAL, STATE, <([is interpreted as a left bracket],)> [is interpreted as a right bracket], COMPILE, (COMPILE)

Miscellaneous: (, HERE, PAD, >IN, BLK, ABORT, QUIT, 79-STANDARD

Additional words in FORTH kernal: NFA, CFA, IN, OUT, BEGIN...AGAIN, INTERPRET, SEND, HEX, SYS, H, " xxx ", DU*, DU/MOD, D-, VM.DCB [6.X]

DOS file handling: ERROR, WRITE, READ, INIT, OPEN, CLOSE, KILL, POSN, DCB xxx, DATA, FILENAME, CHECK, NEW.FILE

General utility words: DOS, CURRENT?, CONTEXT?, BASE?, VLIST, FORGET-SYSTEM MODELI, MODELIII, SAVE-SYSTEM, RESTART, LOADS, BOOT, STAX, STAX?, DUMP

Primitive assembler: CODE xxx, ;CODE, C,, END-CODE, LABEL xxx, xxx

Additional Control functions: CASE: xxx, switch: XXX, ;switch, -->

Terminal and print functions: "IN, #IN, D#IN, ?KEY, U.R, .R, D., D.R

String handling: ", ", SEND, "VARIABLE xxx, "CONSTANT xxx, "@, "!, "LEFT, "RIGHT, "MID, "+, COMPARE, "COMPARE, "=, ">, "<

Double length words: 2!, 2@, 2CONSTANT xxx, 2DROP, 2DUP, 2OVER, 2ROT, 2SWAP, 2VARIABLE xxx, DO=, D=, DABS, DMAX, DMIN, DU<, DNEGATE, D+, D-, DU*, DU/MOD D*, D/, D/MOD, DMOD, D*/, D*/MOD, S>D, D>S, D>Q, Q>D

Arrays: ARRAY xxx, 2ARRAY xxx, "ARRAY xxx, CARRAY xxx, FARRAY

TRS-80 SCREEN WORDS: cursor, cls, line, tab, curs.off, curs.on

TRS-80 graphics functions: GSET, GCLR, G?, HLINE, VLINE, BOX

TRS-80 printer functions: PEMIT, PCR, PSPACE, PSPACES, PTYPE, PSEND, PLIST, PLISTS, P., PFF, P.", EMIT.TO.PRINTER, EMIT.TO.DISPLAY

Random numbers: SEED, RAND, RANDOM, RANDOMIZE

Floating point: FDROP, FDUP, FOVER, FSWAP, FROT, FABS, F@, F!, FCONSTANT xx

FVARIABLE xxx, FARRAY xxx, FNORM, FPACK, FUNPACK, F*, F/, F+, F-, F<, F>, F0<, F0>, D>F, F>D, SCI>FBIN, FBIN>SCI, F., F.LEN, FZERO, FCONS xxx

Debug facilities: DEBUG, RESUME, DSTACK?, RSTACK? DECOMPILE <name>, XREF <name>, REDEFINE <name1> <name2>

The HARTFORTH system also includes a memory editor and a screen editor.

HARTFORTH: TRS-80 Models I and III - TRSDOS, LDOS, DOSPLUS, MULTIDOS, NEWDOS
PRO-HARTFORTH: Model 4/4P under TRSDOS 6.x (or MAXDOS 6)

PRO-MLIB - by Richard N. Deglin

This is a total rewrite of our Model I/III REL librarian (Microsoft compatible relocatable files) designed to interface with TRSDOS 6.x and other cmpatible systems. Besides supporting the standard REL library usable by Microsoft's linkage editor, L-80, PRO-MLIB also supports the creation and managing of an indexed relocatable library (IRL) as defined by Digital Research, Inc., for use with their LINK-80 linker. For specific details of PRO-MLIB, see the appropriate documentation in our product catalog.

MSP-02 - by Richard N. Deglin and Karl A. Hessinger

This is a package of utilities drawn from the resources of PRO-ESP, designed to support your operation under LDOS 5.1 on the Model I or Model III computers. Included in this package from the capabilities of PRO-ESP is: CRLF/FLT, CTLG/FLT, IOMON, NAME, RD40, XONXOFF/FLT, UNKILL (version 5 of UNREMOVE), DOEDIT/FLT, and DED. In addition, NODAM has been added. This tool allows you to read a disk without regard to the particular Data Address Mark used to write the directory sectors. Thus, on a Model III, you can read a pure Model I single density disk.

MSP-02: For Model I/III LDOS 5.1

DW2PS/FLT - by Roy Soltoff

This is a filter for the Radio Shack Daisy Wheel II printer. It provides for proportional spacing of any text sent to the printer through the standard printer device handler. This filter package also has a boldface filter which achieves **BOLDFACE** via multiple overstrike. The PS filter supports control codes to switch it between its boldface mode and 12-pitch mode. With appropriate application software, you can thus switch printwheels between a proportional space wheel and a 12-pitch wheel in one printing operation. DW2PS comes equipped for both the Model I and Model III computers under LDOS 5.1 and Model 4/4P computers operating under TRSDOS 6.x.

DW2PS: For Model I/III LDOS 5.1 and TRSDOS 6.x

PRO-ZSHELL - by Karl A. Hessinger

This is an enhanced version of our popular Model I/III LDOS ZSHELL package redesigned for TRSDOS 6.x operation. ZSHELL, you will recall, provides the UNIX-type facilities of command line I/O redirection of standard input (STDIN) and standard output (STDOUT). ZSHELL interprets these device streams as *KI (the Keyboard Input) and *DO (the Display Output). ZSHELL adds the capability of referencing output to *PR as standard output in lieu of *DO during the execution of any command. More than one command may be stacked into the command line buffer by separating each command with the logical line end character which defaults to a semicolon.

Also in ZSHELL's repertoire of functions is the ability to pipe the standard output of one program into the standard input of a second program. The standard output of the second may also be piped into a third, and so on for as many commands as can be stacked into the command line. Under PRO-ZSHELL, you have the option of extending the SHELL's command line buffer to handle up to 255 characters. This gives you the extra room to handle more multiple commands and longer piping processing streams. Any one logical command is still limited to the DOS's max (79 characters under 6.2).

ZSHELL provides you with options to control the effect of an end of file reached on a redirected input device stream. Three options are available: revert standard input to the un-redirected handler, maintain the redirected stream but pass a BREAK to the application, or pass control to @ABORT. ZSHELL defaulted to the former under the Model I/III version and provided the latter two functions by appending either "#" or "@" to the redirection character. PRO-ZSHELL defaults to pass a BREAK but can be changed to the Model I/III convention when installed. In either case, the "#" sign forces the opposite behavior. The default operation under PRO-ZSHELL matches the behaviour experienced using redirection under our LC compiler - and the operation of redirection under UNIX, in general. Finally, the command line modification of the EOF detection result has also been incorporated into the piping operation via an appendage of the modification character to the piping specifier.

PRO-ZSHELL comes with a TRSDOS 6.x version of WC, the wildcard shell processor that allows you to invoke compatible commands on a number of file specifications that match a wildcardspec entered on the command line. You enter the command line once while the wildcard shell processor searches the designated disk drive(s) for files that match your wildcard specification. WC builds a Job Control Language file of your commands substituting each matching file specification for the wildcard specification on a separate command line. WC then automatically invokes the JCL file.

PRO-ZSHELL: For Model 4/4P under TRSDOS 6.x and MAX-80 under MAXDOS 6.2
ZSHELL: For Model I/III under LDOS 5.1

SAID - Full Screen Editor - by Karl A Hessinger

SAID is a full-screen text editor extremely powerful for creating and editing assembler and C source files, or plain ASCII text of any sort - even KSM files. SAID's hex input mode even permits the input of non-ASCII characters. SAID is very flexible. It comes with an installation program which allows you to specify every keystroke used to enter SAID commands. You can even set up extra function keys to duplicate a handful of SAID's commands to provide more than one key to do the same job. With this flexibility, it is easy to duplicate the key entries of other text editors; however, SAID does provide a pop-up menu which shows SAID's standard command keyboard so you can become comfortable with SAID's command keys. The Model I/III version of SAID comes supplied with a type-ahead keyboard driver which simulates the Model 4 keyboard entry for use with SAID. If you purchase our version 4.3 assembler, be advised that SAID is supplied with EDAS - PRO-SAID with PRO-CREATE.

A powerful feature of SAID is its ability to accept the TAB character entries, maintain the TAB character within the text buffer, but expand the TABs on the screen for your viewing pleasure. Even when the cursor is moved left over an "expanded" TAB, it correctly is moved to the proper column. Rather than perform word wrap (where a word is moved to the next video line if it can't fit on the current line), SAID bends, at the last video column, a long statement to the next line. For language source entry, this method guarantees that you know what is entered and what isn't.

Notes from MISOSYS

The Model 4/4P 128K user has a distinct advantage with PRO-SAID. This version automatically sets up as many extra buffers as you have available 32K RAM banks. Thus, if you are not using the extra 64K of RAM when you invoke SAID, it will provide you the capability of editing three files simultaneously with the limitation that the alternate bank files cannot exceed 32K each. SAID goes one better than most other text editors which use this extra RAM. With SAID, you can define up to 10 numbered blocks in each buffer and specify the copying of these blocks between buffers. Want to take a piece of code from one file into another? Load them both, define the needed block, then do an external block copy - it's as simple as that.

SAID cursor movement functions support: Left, Right, Up, Down, Word, Page up, Page down, Start of line, End of line, Start of file, End of file, and Insert a tab. The modes of text insertion are: Insert or overtype mode, Line insert, Hex insert, and Quote insert. Under insert mode, all entries push the text below the cursor down each time a character is entered. Under line insert, a line of spaces is opened up automatically each time a character insertion reaches the end of the opened line.

SAID provides many forms of character deletions. These include: Delete char, Delete word, Delete line, Delete block, Delete to top, Delete to end, Delete all, and Undelete. Note that last one! SAID can restore your text to its exact makeup as of the last delete operation. Do a DELETE ALL followed by an UNDELETE and you have lost only one character.

SAID provides for a MACRO key by which you can dynamically define a series of keystrokes (up to 64 characters to be exact) to be invoked each time you depress the MACRO button. This MACRO facility is a tremendous time saver when you need to repeat a series of keystrokes.

The I/O functions include: Print a block, Print the memory buffer, Load file at cursor position, Save file under current or new name, and Save block. Yes, you can save a marked block of text by itself out to a disk file. Speaking of blocks, the block functions include: Start block, End block, Copy block, Move block, and Unmark all blocks.

The Search and replace facility allows you to: Search, Reverse search, Replace, Again, and All. You can specify the search to be case insensitive or case sensitive. You can request the search replacement to be queried on each find. Both the search and replace strings may contain a wildcard character. The strings may also contain a '%' followed by two hexadecimal digits to provide for the search/replacement of non-ASCII or control characters.

You can invoke any DOS command from within SAID. You also have at your disposal a META command which provides these additional functions: A Calculator, External memory Swap and block Copy, Popup of help display, Set macro repeat count, Set SAID options for ASM mode, CCC mode, default extension, and set TABS, Strip bit 7 off all text in buffer, Uppercase next word and Lowercase next word.

The built-in calculator supports binary, decimal, and hexadecimal. You can convert any number from one base to another base. The following functions are supported: Multiplication, Division, Addition, Subtraction, Logical AND, Logical OR, and Logical XOR. Entering a period will cause the last result to be substituted. The built-in calculator can save you time and energy when trying to code values into source text for programs.

SAID: For TRS-80 Model I/III under LDOS 5.x, TRSDOS, and other compatibles.
PRO-SAID: For TRSDOS 6.x/MAXDOS 6.x

PRO-ESP - by Richard N. Deglin and Karl A. Hessinger

This package is a TRSDOS 6.x design of the popular ESP package by the above authors. As you know, ESP was designed to utilize the Model 4 hardware features while operating under LDOS 5. Well Rich and Karl have adapted their best to use while operating under TRSDOS 6.x. These functions extend the capabilities of your DOS and make it 6.x an even greater joy to operate with.

ALTDISK is a disk-drive simulator which creates a 32K or 64K DOS drive in the second (64K) bank of RAM. It uses less than 30 bytes of low memory and about 100 bytes of high memory, and can be operated with all DOS 6 products.

The ALTLD utility provides a simple way to rapidly save and restore the entire contents of the alternate RAM banks to and from a disk file. It will check the length of a file to load, and will only load it into alternate RAM if the file is exactly 64K bytes in length. This safeguard is there if you inadvertently attempt to load a file which is not a RAM image.

ALTRES is a replacement for the DOS command which places a specified system overlay into the upper alternate bank of memory. It functions in the same way as SYSTEM (SYSRES=nn) except that the system modules are stored in an alternate bank instead of upper memory. This saves your high memory area!

The CRLF filter allows the DOS 6.x video driver to properly handle carriage return/linefeed pairs sent to it. Every carriage return sent to the filtered device is changed into a beginning of line character.

The CTLG filter will produce an audible beep using your machine's internal speaker whenever an ASCII BEL character is sent to the filtered output device. This filter will find use in conjunction with applications which require the generation of an audible tone.

CVT324 will help you convert your old Model I/III BASIC programs to run on DOS Version 6 BASIC by performing most of the conversion work automatically. CVT324 will perform 1) add spaces around keywords, 2) strip any trailing information after a CLEAR statement, 3) convert PRINT @ statements for 80 x 24 screen, and flag with an appropriate message, any line that has one of the following keywords: INPUT, OUTPUT, POKE, PEEK, USR, SET, RESET, POINT, CLOAD, CSAVE, or SYSTEM.

The Disk EDitor (DED) is a full screen disk sector editor which allows you to easily modify the contents of any DOS compatible formatted disk. DED displays a sector of data at a time. The ASCII characters of each byte are displayed on the left hand side of the screen which correspond to the hexadecimal values to the right. Beneath the sector display is a line containing the current drive, cylinder, sector and the relative byte position within the sector; the value of the byte at that position in hexadecimal, binary, and decimal formats. A second line of information contains the name of the file to which the current sector is assigned. DED supports commands to: Enter the ASCII or hexadecimal modify mode, Search for an ASCII or hexadecimal string, Select a new drive, Go to next occurrence of search string, Print the current sector, Position to previous or next sector or cylinder, Print a TOF, Reposition to cylinder and sector, Save the sector buffer to the disk, Verify sectors, Exit to DOS, and Display a menu of commands.

The DOEDIT filter will allow you to edit any line of the video screen and will allow this edited information to be passed back through the keyboard device to any application which uses standard keyboard input. This very powerful feature will pay handsome dividends as you learn to take advantage of its potential for curing your typing mistakes. DOEDIT works great for recapturing and modifying the last DOS command entered, too! Once DOEDIT has

been activated by the keystroke you have specified, you can move about the screen, insert and delete characters, and pass a screen line back to the program which is requesting keyboard input.

FKEY allows you to redefine the codes returned by the function keys. You can also restore the function keys to their original DOS generated values.

IOMON/CMD is a disk drive "filter" which will monitor disk input/output for errors and allow you to take corrective action when such occur. When a trapped disk I/O error occurs, the following will be displayed:

```
Disk I/O error X'nn': <message>
Function X'nn', Drive n, Cylinder X'nn', Sector X'nn', Buffer X'nnnn'
<R>etry, <C>ontinue, <I>gnore, <A>bort?
```

Simply type the first letter for the desired action to be taken: R, C, I, A.

The MINIDOS filter allows you to access some DOS commands without the necessity of being at the DOS Ready level. The appropriate function key will produce one of the following actions: Toggle the TIME (CLOCK) display, Enter DEBUG, Display free space for a drive, Remove a file, Print a byte, Display a disk directory for a drive, Rename a file, Print a Top-of-Form.

The NAME utility allows you to change the name and/or date of a diskette. Any printable ASCII character (32-127) is acceptable for the disk name.

PRTOGGLE allows you to dynamically link the video device to the printer device by monitoring the keyboard device for depression of your activation keystroke. Upon its receipt, subsequent information directed to the video (*DO) device will also be copied to the printer (*PR) device. This "link" will remain active until the activation keystroke is again depressed.

RD40/CMD is a disk drive "filter" which, when installed, will allow the reading of a 40-cylinder diskette in an 80-cylinder drive. RD40 may be installed independently on more than one drive. Every time RD40 is invoked for a particular drive, the drive will be restored to cylinder 0 in preparation for further disk accesses to that drive.

The UNREMOVE utility allows you to recover a file which was inadvertantly removed from a disk. You will be able to recover the file, provided that you haven't already reused the disk space for some other file(s).

The XONXOFF filter implements the standard XON-XOFF handshaking protocol for those serial devices which require it. This filter is designed to allow the attachment of serial output devices which require your host computer to honor the XON-XOFF flow control protocol. The filter will operate correctly regardless of baud rate, parity, or word size.

PRO-ESP: For Model 4/4P under TRSDOS 6.x.

PRO-NTO - by Karl A. Hessinger

MISOSYS is going to do something it rarely does - preannounce a product. PRONTO is not expected to be released until March 1985; however, the internal development prototype of this package is so outstandingly significant, that I felt it most important to let you know what is on the immediate horizon. PRONTO sounds like TONTO - whom we all know was the sidekick of The Lone Ranger. Does that perk you up. PRONTO conjures up visions of being fast - i.e. pop up here, pronto! Pop up you say?

Notes from MISOSYS

PRONTO is for the Model 4/4P 128K machine. Pronto provides a WINDOW SuperVisor Call (SVC) to permit the use of overlayed windows from any and all applications. Pronto is a window application manager which supports function key keystroke invocation of up to FIVE small applications plus a LIBrary EXEC facility which gives direct access to all of the DOS's library commands. PRONTO requires one 32K RAM bank, about 2K of high memory, and a small piece of low RAM. We feel that PRONTO is going to be so useful and important to EVERY 128K Model 4/4P user and every systems house writing applications for the 4/4P that we are going to provide preliminary specifications for PRONTO. In addition, due to the expected popularity of PRONTO, you may wish to place a firm pre-paid order for PRONTO now. All orders for PRONTO will absolutely be filled in the sequence that they are received. Here's its capabilities.

Pronto allows an application to request a window on the current screen of whatever size you need - up to the maximum of 80 by 24. Windows requested of size 78 by 22 or smaller will have a box automatically drawn by the window manager. Under the window application environment, up to 4 windows may be nested, i.e. an application can invoke more than one window or nest to another application which requests a window. More windows may be available when operating solely under the window mode.

The window control SVC (@WNCTL) provides nine functions of which the first six are similar to @VDCTL: WNPEEK - character window peek, WNPOKE - character window poke, WNSETCUR - set cursor position in window, WNGETCUR - obtain cursor position in window, WN2WIN - mover buffer image to window, WN2BUF - move window image to buffer, WNCREATE - request a window of size and position, WNCLOSE - close current window and revert to previous window, WNDSP - display character at window's cursor, and WNDSPLY - display string at window's cursor. The window display driver supports CR, LF, Erase to EOWL, Erase to EOW, and destructive backspace.

The PRONTO application manager provides support of application programs which run totally within the DOS's library overlay region and do not exceed 2K in length. Five applications may be resident at one time and are stored in the 32K RAM segment. Applications run in the address range 2800H-2FFFH and have available two 512K data blocks from 2600H-27FFH and 2400H-25FFH. Where the program requires less than the 2K of code space, it has more space available for data. Pronto saves this 3K address space on each application invocation and the contents of the video screen are saved when a window is requested. Applications can be invoked recursively or consecutively. The application manager even displays a window menu describing each application available at a keystroke. The applications are loaded from core-image files during the installation of PRONTO.

PRONTO comes with its own set of applications. A perpetual calendar application can display a month at a glance - for any year since the adoption of the Julian calendar and for as many years as you care to project (there is some upper limit; however, its thousands of years away). PRONTO comes with a programmer's reverse polish notation calculator and a standard algebraic one for you normal folks out there. PRONTO comes with a notepad capable of keeping all of those scratchpad entries in order. Pronto comes with an Autodialer usable with a Hayes-compatible modem. This application doubles as an address file. The DOS library command executive is included as one of the six built-in applications. Also included are LC functions to interface with the window manager. Other applications are planned for release.

PRONTO is going to be the Model 4/4P event of the year. If you don't have your extra 64K yet, you better do that now in preparation. PRONTO will be priced at just $49.95. Again, it's targeted for March 85 delivery. We are accepting prepaid firm orders now (checks will not be cashed nor will charges be put through until PRONTO ships). You better hurry for PRONTO, pronto!

THE FOLLOWING IS A PRODUCT ADVERTISEMENT UNRELATED TO MISOSYS, INC

The wait is over, you can now get LDOS/TRSDOS 6 for the Lobo Max-80! This is a complete DOS 6 implementation, which includes:

Full access to all 128K
Full access to both serial ports
Full 8" and 5 1/4" drive support

Along with the pre-patched TRSDOS 6.x disk (called MAXDOS) you get another disk with a host of DOS Version 6 utilities:

ALTDISK	- Use the external banks as a RAM disk.
ALTLD	- Rapidly save/restore the contents of external RAM.
ALTRES	- Place system overlays in the external RAM.
CRLF	- Filter to suppress a LF after a CR.
CTLG	- Filter to beep on every ASCII BEL.
CVT324	- Convert Ver 5 BASIC programs to run under Ver 6 BASIC.
DED	- A complete disk "zapper".
DOEDIT	- Edit info on the video and pass it back through *KI.
FKEY	- Change the value returned by the function keys.
ICS	- Install a character set on a system disk.
INVERSE	- Filter to implement inverse video.
IOMON	- A complete disk I/O error trapper.
MINIDOS	- Filter to access some DOS commands.
NAME	- Change the name and/or the date of a disk.
NEWCLOCK	- Display current date & time on the status line.
PRTOGGLE	- Link the *DO and the *PR with a single keystroke.
RD40	- Read a 40 track disk in an 80 track drive.
STATLINE	- Setup the contents of the status line.
STATLINE	- Filter to allow an application to set the status line.
UNREMOVE	- Recover a file after it has been REMOVEd.
XONXOFF	- A filter to implement XON/XOFF handshaking protocol.

And there's more! Also included is the MISOSYS "Programmer's Guide to LDOS/TRSDOS Version 6" to allow you to begin programming right away!

The entire package is priced at $150.00*, but if ordered from now until January 31st, 1985, you can get MAXDOS for only $135.00* (10% off the list price). So hurry and order your copy today!

MAXDOS is available from:

MicroConsultants - East
7509 Wellesley Drive
College Park, MD 20740-3037
(301) 474-8486
After 7PM ET & Saturday

Riclin Computer Products
4901 Seminary Road, Suite 1003
Alexandria, VA 22311-1834
(703) 820-1103
After 7PM ET & Saturday

MAXDOS is solely supported by its distributors listed above. Please do not call either Logical Systems, Inc. or Tandy/Radio Shack.

LDOS and LS-DOS are trademarks of Logical Systems, Inc. TRSDOS and TRS-80 are registered trademarks of Tandy Corporation. MAX-80 is a trademark of Lobo Systems, Inc. MAXDOS is a trademark of Riclin Computer Products and .MicroConsultants - East.

** Special disk formats are available for an additional $10.00

ADE/PRO-ADE

There is an old joke that goes like this: I have some good news and some bad news. The bad news is that your water pipes burst while you were away on vacation. The good news is that you now have a swimming pool in your basement [you're supposed to laugh now]. Well, when it comes to ADE, I also have bad and good news. Any one who has purchased from us or registered with us a copy of ADE version 1.0 should have received a letter announcing a defect in that ADE version. That was the bad news. The good news, of course, was that the 1.0 version would be updated at no charge to the 1.1 version which corrected the defect. In case you may have missed the notice, read the following.

To: Owners of ADE version 1.0 or PRO-ADE version 1.0
Subject: Recall for update to version 1.1

Due to a design oversight, there is a bug in version 1.0 of ADE and PRO-ADE which keeps the first sector of a new granule from being correctly written when the file is not pre-allocated. Please return your copy of ADE and/or PRO-AE for a free update to version 1.1 which fixes the problem. In the interim, you can continue to use version 1.0; however, do one of the following for each file placed in an ADE "floppy": (1) Use COPY to write the file, or (2) Write the file twice, or (3) Use CREATE to assign file space then write the file. End of notice.

David B. Lamkins wrote in response to our recall letter, "Your recent recall notice for ADE has me a little concerned. Problem is: I have a couple hundred files moved into ADE disks and haven't had (or at least haven't noticed) any problems. Can you be more specific about the failure circumstances and/or symptoms so I'll know what to check for?

While on the subject of ADE, I've noticed that ADE overestimates the size required for the drive emulation file. I hand-calculated the space required and opened up an exact-fit chunk of contiguous free space, but ADE took a larger chunk of space at higher-numbered tracks instead (I'm using LDOS 5.1.4 with sequential space allocation)."

I responded, "You needn't be that concerned with the problem in ADE 1.0 if files were transferred to the /ADE "disk" via COPY, BACKUP, or by CREATE followed by writing the file via some program. The error was a problem whereby the first sector of a NEWLY allocated granule was written not to the first sector of the granule but to a relative sector within that granule based upon the relative sector of the file's directory record within the ADE directory. For instance, the first write to a newly allocated sector should be to the zeroth relative sector (assuming sequential writing is taking place). Under the problem, if the file's directory record was on sector 4 of the directory (that's relative 4), then the sector of data would be written to relative sector 4 of the newly allocated granule instead of to relative sector 0 as it should. Note that I emphasized NEWLY. If any of the three methods specified above were used to put a file onto the ADE "disk", the file space would be allocated first by the DOS then the file would be written. Thus, the procedure would guarantee that no NEW allocations would be done when the file itself was actually written. Version 1.1 overcame this difficulty with some cute code that interfaced with the DOS's routine which calculates the disk cylinder and sector of a file's record.

As far as the other "problem" which you noticed, it relates to the procedure which ADE uses to obtain disk space for the ADE file. ADE does not use the DOS's file allocation routines for obtaining disk space. Rather, it uses its own algorithm (an algorithm, I might add, that should have been part of the DOS's CREATE library command). ADE must guarantee that the needed file space is obtainable within four directory extents. This is to prevent the

recursion of the file allocation routines during an ADE file's disk I/O. If the open FCB contains all of the information pertinant to a file, there is no need to recurse through the file access routines to obtain the needed access info. In fact, in order for ADE to work under LDOS 5.1, this restriction is paramount. Thus, ADE uses an allocation routine developed by KARL in working up the MACH2 package. Essentially, the routine reads the GAT and develops a table of contiguous space by starting granule, then sorts the table in descending order (i.e. the largest contiguous space first). ADE then allocates space to the file from this table starting with the biggest chunk and working towards the smaller. If the needed space cannot be obtained within four extents, ADE aborts and issues an appropriate error message.

If you calculate the space needed for an ADE file, don't forget that it needs one additional sector to store the data describing the file's emulation of a floppy. That data is needed when re-installing an existing ADE file. In a number of cases, the need for an additional sector actually takes up another granule. For instance, a 40-cylinder DDEN floppy takes 40x18=720 sectors. If this is emulated on a drive which has granule sizes divisible by 6, then the 721st (ADE data) sector would force the allocation of another granule on the host drive.

Incidentally, it was Steve Mann who first brought to our attention, the problem which resulted in the recall. Although I considered the problem to be severe, it did not get observed in our testing. That was because in all of our tests, we only worked with files that were either backed up to an ADE floppy via BACKUP or copied to an ADE floppy via COPY. These two methods masked the problem. Thanks Steve, for bringing this to our attention.

Prior to Steve's notification, two others are credited with bringing other problems with ADE to our attention. Timothy D. Dondlinger, of Specialty Spheres in West Bend, WI came across a problem in our method of indexing the ADE file to arrive at the proper record correlating to the needed ADE "sector". Actually, it was a coding error mistyping an INC BC for an INC B; however, the result was still an algorithmic error.

Let me give a brief explanation as to how ADE works. You hackers out there may be interested in this. When you create an ADE "floppy", a file is created which contains storage space for the emulated floppy. For instance, if you select a 40-track single-sided double density disk to be an ADE file on a hard drive, the ADE floppy must be 720 records long (40x18=720). Actually, it needs 721 since the first sector of the ADE file is used by ADE to store information on the structure of the file. It is not used as a "sector" of the emulated floppy. Let's call this ADE floppy, FLOP1/ADE and assign it via the ADE/DCT driver to be drive 6.

A file, say TEST/DAT, stored in FLOP1/ADE looks like any other file to the DOS. When a program (or the system) wants to access a record of TEST/DAT once TEST/DAT has been opened, the DOS examines the File Control Block (FCB) associated with TEST/DAT and calculates the cylinder and sector of the floppy which stores the needed record. These figures are, of course, just logical figures since they are relative to the emulated floppy upon which TEST/DAT is stored. The DOS does not know that drive 6 is not "real". The system then passes a request for sector access via the disk driver noted in the Drive Control Table (DCT) associated with the disk. In the case of an ADE floppy, this passes control to the ADE/DCT driver.

The ADE driver locates the FCB corresponding to the FLOP1/ADE in its slot table (remember, you can request from 1 to 8 slots when you first bring up ADE). The driver now proceeds to calculate the record in FLOP1/ADE which corresponds to the logical cylinder and sector provided in the driver call. The code shown as follows:

```
LD      H,0
LD      L,D             ;Xfer cyl # to HL
LD      C,(IY+7)        ;P/u sectors per track
INC     C               ;  & adjust for zero offset
BIT     5,(IY+4)        ;If 2-sided,
JR      Z,$+4           ;  double the count
SLA     C
LD      B,E             ;Hang on to sector number
LD      A,@MUL16        ;Multiply HL by C (for LDOS 5.1,
RST     40              ;  LD A,U and use CALL 444EH)
ADD     A,B             ;Add sector to cyl x SPC
LD      B,L             ;Record number to BC
LD      C,A
JR      NC,$+3          ;Bump hi-order if carry
INC     B               ;Note: this was an INC BC in error
INC     BC              ;Bypass the ADE control sector
```

calculates a logical record of the ADE file by multiplying the cylinder number requested (in register D) by the number of sectors per track. This result winds up in the register triad, HLA. Next, the sector requested (in E transferred to B) is added to the triad. Registers L and A contain the mid order and low order values of the "cylinder times sectors per track" result. The high order in H will be zero. Register L is transferred to register B (which is the high order of the position). The sum of the low order and sector requested is transferred to register C. If this sum results in a carry, then as Timothy pointed out, the high order register, register B, should be incremented. The code had register pair BC incremented. The second INC BC is used to bypass the first sector of the file which contains ADE-specific data. At this point, register pair BC has the needed record position (in the example, it's the record position of FLOP1/ADE which corresponds to the sector corresponding to the TEST/DAT record needed. Patches were developed to fix the bug; however, the release of version 1.1 makes them moot.

Another bug uncoverd by Mike Gorman of Bowie MD was that ADE sort of goes bananas when you select a size 3 double sided double density diskette. After checking into the cause, I have had to restrict that combination. It can't be supported. When I originally proposed a smaller allocation unit to Karl Hessinger (Karl is the co-author of ADE) as an option to ADE, Karl suggested that we assign a granule size of 3 sectors. This could then be used to make an ADE floppy which emulates the allocation scheme used in TRSDOS 1.3 (which is 3 Sectors Per Granule and 6 Granules Per Track). So we did. However, both Karl and I overlooked the result of a user selecting 2-sides [of course, TRSDOS 1.3 does not support 2 sided media]. In a 2-sided disk, the number of GPT is doubled; however, you cannot double 6GPT and get 12GPT since the DOS can support only 8GPT. Therefore I developed a patch to restrict anyone from requesting 2-sides if they already specifed size 3 and DDEN. Again, there is no need to show the patch since version 1.1 has incorporated the patch code.

David Lamkins of Canton, MI presented an interesting ADE application. Dave wrote, "ADE arrived today. I think it will be very helpful in organizing my hard disk. Despite the clear statement of ADE's intent to implement a two-tier file system, I couldn't resist 'pushing' ADE to see how well it would handle a multi-tier system. To do this, I created a large-capacity ADE drive on my disk 3. I called this NESTTEST/ADE, and assigned a logical drive number of 4. I then created a smaller capacity ADE drive, called INNER/ADE, locating this on drive 4 (thus a file within NESTTEST/ADE) and called this drive 5. Much to my delight, this worked quite nicely for all operations which 'seemed safe' under such a configuration, e.g. copy, backup, create, kill, free, device, etc."

Dave went on to explain his shenanigans of disabling drive 4 without disabling drive 5. Drive 5 then showed up as write protected! ADE/CMD also yielded a 'Directory read error' message when it was invoked [note: ADE/CMD needs to access @FNAME for each ADE file in order to present the information in the ADE/CMD menu - thus, since Dave disabled drive 4, @FNAME could not get access to the drive 4 directory which resulted in the error message]. Dave also requested that the next iteration of ADE provide an automatic password similar to what I did with PaDS to minimize the inadvertant destruction of an ADE file.

My response to Dave went something like this: "Both Karl and I got a chuckle out of your nested ADE environment. True, there is no reason why it should not work and your results proved it did. The reason why Karl chose to ignore errors in ADE/CMD was that he didn't want to inhibit you from accessing control over the other AID slots if one could not be accessed. ADE needs to read the directory of the host drive in order to get the filespec of the ADE file via @FNAME. Since your host drive (the outer ADE file) was disabled, the directory could not be accessed resulting in the "directory read error". ADE can safely ignore this and go on to the next slot. Obviously, if all slots resulted in an error, there is no need to continue.

I think that you are right about ADE's needing a protection password. I'll suggest that to Karl in the next iteration, the file/ADE should get the same treatment as I gave to PaDS. Although in this case, it needs read/write access."

Now since I had reason to bring up version 1.1 of ADE, I implemented what Dave suggested. Henceforth, /ADE files generated with ADE/DCT version 1.1 will be password protected with the password, ".ADE". The password should be transparent to most of your operations since I grant read/write access without knowledge of the password. That means that you can move an /ADE file from one disk to another without specifying the password. You can also, of course, access an /ADE file via the ADE/CMD program or ADE/DCT driver without using the password. However, if you want to delete or rename an /ADE file, you will need the password.

Finally, I have received a report that you can't properly back up a MEMDISK simulated floppy to an ADE emulated floppy under TRSDOS 6.x. I checked into this arrangement and found the culprit. Both MEMDISK and ADE (as well as most hard disk drivers) simulate the data address mark convention for the directory. Because of this, an attempt to perform a mirror image BACKUP of a disk to an ADE drive that has the directory on a different cylinder fails because the BACKUP utility does not update the directory cylinder field of the destination DCT until after the BACKUP operation is complete. If BACKUP properly carried over that entry sooner, there would be no problem with copying a MEMDISK to an ADE floppy. One solution is to patch BACKUP. Another is to permit you to specify the number of the directory cylinder for the ADE floppy when it is created. We'll look into both those solutions and let the PRO-ADE users know in the next issue. In the interim, the following procedure should suffice. Make up an ADE floppy the same size as the MEMDISK - that should be 14 cylinders, DDEN. When ADE passes control to the FORMAT utility, the directory will automatically come out on cylinder 7. Make note of the FORMAT command invocation generated by ADE/DCT (i.e. write it down - this is a perfect illustration of the use of DOEDIT). Re-invoke the exact FORMAT command but add the TRSDOS 6.2 parameter, DIR=1. Issue the LOG command and target the ADE drive. Then apply a patch to the /ADE file used to contain the emulated floppy. The patch will change the header record ADE keeps in each /ADE file to be able to re-install the "floppy". The patch is, "PATCH filename/ADE (D00,0F=01:F00,0F=07)".

CONVCPM/PRO-CURE

Don't forget that you can still upgrade from CONVCPM Version 1.x to CONVCPM Version 2.x for $20 with the return of your master diskette. The price includes return shipping charges and a new manual (as well as the update to the disk). If you have an interest in obtaining the new LCOPY program which runs under selected CP/M systems (see THE BLURB), then you get a $20 discount from the purchase of LCOPY if you already own (or are buying) CONVCPM 2.x or PRO-CURE 2.x. What this means to you CONVCPM 1.x owners is that for $50, you can get your CONVCPM upgraded as well as obtain LCOPY.

And don't forget, Radio Shack's CP/M+ disk format for the Model 4/4P machines is the same as CP/M-86. Thus, to transfer files from that media, specify the "PC" disk format.

A late breaking bulletin: If you are using PRO-CURE under MAXDOS-6 on the MAX-80, PRO-CURE needs a patch. You see, we have to include code on the Model 4 version because the TRSDOS 6.x disk driver cannot handle any sector length in excess of 256 and we support CP/M formats for 512-byte sectors. The code needs to be modified for MAX-80 operation. The CONVCPM program running under LDOS 5 has the code to test for the MAX and make the appropriate change; however, PRO-CURE was released long before MAXDOS-6 was developed - thus, the patch. Here it is (it's on the DISK NOTES 4 as well).

```
. PCUREMAX/FIX - 11/16/84
D08,C8=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
F08,C8=FD 6E 01 FD 66 02 22 2B 3A 21 25 3A FD 75 01 FD 74 02
D09,D1=07
F09,D1=37
D19,FE=3E 2F 32 D9 39 32 E4 39 21 D8 07 22 11 3A 21 00 3A 00 00
F19,FE=21 D3 F4 22 10 3A 26 F0 22 DA 39 2E DB 22 E2 39 21 F6 39
D1A,DE=". 2.0MAX"
F1A,DE="sion 2.0"
```

DD&T/PRO-DD&T

Just a short note is needed on our new DD&T product. I goofed in preparing the PRO-DD&T master disk because the PTRACE/CMD file could not be loaded. We fixed this starting with PRO-DD&T registration number 620017. If you have an earlier PRO-DD&T copy then send it back in for repair. The problem did not get into the DD&T version.

By the way, if you use DEBUG, you need DD&T. Once you use it, you will find it absolutely essential to the DEBUG environment.

Now then, another item just popped in from Mills Landreth of Apex, NC. DD is absolutely great; however, if you want to use it under TRSDOS 6.1 or TRSDOS 6.0, you will have to apply a patch. No patch is needed for TRSDOS 6.2. The following DD61/FIX absolutely MUST be applied to DD if you are going to use it under TRSDOS 6.1 or earlier. Please apply the patch to a working copy only - NOT your master disk.

```
. DD61/FIX - 11/12/84 -
. Patch PRO-DD&T version of DD for use with TRSDOS 6.0 & 6.1
. DO NOT APPLY PATCH FOR USE WITH TRSDOS 6.2 [ slashes mean <ENTER>'
D00,F8=D6 / F00,F8=DC / D05,01=24 / F05,01=26
D05,04=25 / F05,04=27 / D05,0C=D5 / F05,0C=DB
D05,0F=D6 / F05,0F=DC / D05,41=DE / F05,41=E4
D06,7D=14 / F06,7D=1A / D08,43=0F / F08,43=15
D08,49=14 / F08,49=1A / D08,A6=14 / F08,A6=1A
```

DSMBLR/PRO-DUCE

David Miller of Okmulgee, OK had a problem with paper drift while generating printouts using DSMBLR on a Model I under TRSDOS 2.3. This has been a classic problem since the early days (the early days began in 1977 when Tandy released the TRS-80 Model I. In brief, I responded to David with, "Let me attempt to respond to the problems you are having with DSMBLR. First, I assume that you are running it under TRSDOS 2.3. The line drift problem is generally due to the lines per page value in the printer device control block. This address is 4028H. You most likely will find a 43H value which is 67 decimal. Thus, the way the printer driver in the Model I ROM works, it will put 67 lines on a page; however, a page has only 66 lines. Change this value to 42H. You can do this by running a small program that pokes the value (do it from a assembly language program ORG'd at 5200H). The second issue of NOTES FROM MISOSYS also addresses a patch to SYS0 of TRSDOS 2.3 to automatically fix that address."

Dave also could not generate a cassette tape usable by EDTASM+. His problem is due somewhat to the fact that the DSMBLR documentation does not specifically tell you what parameters are neccessary to customize the DSMBLR output for the various assemblers on the market. For anyone else with the same problem, my response to Dave continued with, "The second problem is also easy to fix. If you want to make a tape to be used in cassette EDTASM+, you must specify the NUMBER parameter and the HEADER parameter when invoking DSMBLR. The source output is defaulted to no line numbers and no header. EDTASM+ requires both."

In general, DSMBLR provides a plain text output for the SOURCE file (tabs are used to separate columns in lieu of multiple spaces). Some assemblers expect the source file to contain a 5-ASCII-decimal digit line number on each line. The line number convention established by Microsoft which has become "standard" is to set bit-7 of each digit of the number and follow the number with a space (Microsoft's EDIT-80 actually writes a TAB with bit-7 set as the byte following the line number. The original cassette EDTASM sold by Tandy which was written by Microsoft (I believe that Hal Chamberlin, an old-time buddy of Bill Gates, actually wrote EDTASM) incorporated a "header" record at the beginning of the cassette file. This header began with the byte X'D6' (which has always appeared to me to be the ASCII letter 'S' with bit-7 set) and is followed by a 6-character file name. The 'S' probably meant assembler "S"ource. Of course, my entire hyppothesis could be just coincidental. In any event, the header could be read by a program to differentiate an assembler source file from a compressed BASIC program which began with the byte, X'FF'.

The first disk-based assembler for the TRS-80 was done by Apparat. It was, in fact, a copy of the cassette EDTASM modified for disk I/O of source and object code. It still retained the JR bug which was in EDTASM 1.1. The second disk-based assembler was actually a huge modification patch to EDTASM 1.1/1.2 and it was called DISK*MOD. It was developed and released by MISOSYS. Since we were second, we kept the same source file structure as what was done by Apparat. Probably EDAS 3.4 was next (not counting the 4.0 version done by us for the Model II for Galactic Software Ltd). We eventually provided an option to write a source file without line numbers and without the header to provide for savings of disk space (those numbers actually take up a whole lot of space in a file). When EDAS 4.1 was released, the standard source file was unnumbered and unheadered - a standard which we promulgate.

The crux of the matter is that if you are using EDAS 4.1 (or later), then let DSMBLR default its output. If you are using Microsoft's M-80 macro assembler, it needs numbers and colons (no header, in fact, I believe it doesn't require the numbers). If you have any other assembler, take a look at

the source file (cassette EDTASM or EDTASM+ use what I mentioned above). If you see a header, then use the HEADER parameter in DSMBLR. If you see line numbers, use the NUMBER parameter. If you are using a macro assembler which requires that local labels be followed with a colon, use the COLON parameter. That's about it for the customizing parameters.

George Geczy of JMG Software International raised a couple of questions concerning the DB and DW output format we use to generate string, byte, and word source lines. I thought it would be useful to relate my comments concerning this issue to all of the DSMBLR users. I reponded, "First, the issue of DSMBLR and its generation of DB and DW instead of DEFB, DEFM, and DEFW. True, you hit the nail on the head. We tend to try to do things to promote our own products. That's why we chose to use the shortened form: DB, DW. Second, we also output string decomposition in the format that is usable by EDAS. I don't thnk any other assembler is as flexible in the mixture of bytes and string declarations in a single assembler pseudo-OP. It would be ridiculous for DSMBLR to force a DEFB on every single byte. The output would be enormous for EDAS users. We definitely are trying to promote EDAS."

George's other DSMBLR question concerned using DSMBLR with Tandy's Model I double density DOS, TRSDOS 2.7DD. Dave also brought up 2.7DD. I told Dave that I do not know whether DSMBLR will run under TRSDOS 2.7DD or 2.8. Tandy's Model I double density operating systems are so alien as far as entry points that most software (even their own) will not work. The way to find out is to try it. Well George did try it. He learned that TRSDOS 2.7DD can not load a file that has a X'1F' record type. For those interested in load module record types, see the Appendix (149-153) of "The Programmer's Guide To LDOS/TRSDOS Version 6.x" or look at Roy's Technical Corner appearing in THE LDOS QUARTERLY, Vol 1, April, 1982 pages 42-46. I was flabbergasted when I read George's letter concerning the X'1F' record type. I told him, "The last point, I guess, is about the X'1F' record used in my products. I have NEVER looked in depth at TRSDOS 2.7DD. I would NEVER support any of my products under that system as it is so incompatible with either TRSDOS 2.3 or TRSDOS 1.3 that I classify it further from the TRSDOS-like systems than I classify NEWDOS-80. How could Tandy not support the X'1F' record type? By the way, you are the first person to tell me this. You really ought to advise your customers to use something other than 2.7DD. I mean, gosh!"

Charles East of Heath, Surrey wanted to use DSMBLR to disassemble programs from protected media and from the LDOS library files. Since this topic may interest others, my answer to Charles was, "The DSMBLR can disassemble target programs from either in memory or directly from a disk file. Considering this, there is no way that you can directly disassemble self-booting programs such as SU+ because it is not a "file" on the self-boot disk nor can you load it into memory then regain control.

As far as the library files, SYS6 and SYS7, go, they too cannot be directly disassembled because the files are not programs - they are libraries. However, in this case I have a solution for you. Use the CMDFILE program to extract the member you want to disassemble and write it out as a normal /CMD file. Then you can disassemble it." For you TRSDOS 6.x users, currently, the only way I know to extract a SYS6/SYS7/SYS8 library member short of writing your own program to do it is to use either the LDOS 5.x CMDFILE program or the TRSDOS 6.x PRO-CESS program - both available from MISOSYS.

One last little bit of information is needed to get the record straight on DSMBLR. NOTES FROM MISOSYS, Issue III presented a DSMBLR patch on page 3-12 that was described as DSMBLR36/FIX. James A. Sladek of Norfolk, VA reported that it should have been correctly identified as DSMBLR37/FIX as a DSMBLR36/FIX appeared in NOTES Issue II (page 2-12). I stand corrected - James is absolutely right.

As I was working on NOTES, I received a disk from John Carroll of Sheridan WY. He writes, "Thank you for writing your disassembler. I have learned more about assembling since I received it than ever before. Included [on the disk] is EQUSUB/BAS. It uses a file like EQUATE3/EQU [such a file is generated by XREF via the EQU option] to substitute recognizable names for the 'Mxxxx' symbols created by DSMBLR/CMD. It works great. Distribute it if you wish. You have my rights. It begs to be written in C, perhaps I will."

The following is the BASIC program supplied by John and the corresponding documentation. The program was written for LDOS 5.1 and LBASIC. It should be easily adaptable to TRSDOS 6. DISK NOTES 4 contains the material supplied by John (with the exception of LOG/CMD). I have saved the BASIC program in ASCII so that it can be loaded and played with under TRSDOS 6 BASIC. Note: a line feed is noted as {LF}.

```
10 'This is saved as "EQUSUB/BAS"
20 CLS
30 DEFINT A-Z
32 CLEAR10000 ' GOT LOTS, USE A BUNCH
35 E$="EQU":DIR$="DIR":MM$="M":CM$="CMD"
45 ON ERROR GOTO5000
47 LINEINPUT"What is the name of the EQUATEs file?  Enter 'DIR'{LF} with any
extensions for a look at the directory {LF}  ";F1$
49 IF INSTR(F1$,DIR$):CMDF1$:GOTO47
53 OPEN"I",1,F1$
54 CLS:PRINT"Stand by, memorizing ";F1$
80 LINEINPUT#1,T$                'GET A LINE THROUGH &HOD
85 IF EOF(1):PRINT "Finished with ";F1$:CLOSE:GOTO205
90 IF LEFT$ (T$,1)=";" :GOTO 80 'DON'T FOOL WITH THE COMMENTS
100 Z=INSTR(T$,E$)              'FIND THE POSITION OF "EQU"
110 AD$=MID$(T$,Z+4,4)          'THIS IS THE ADDRESS STRING
120 NA$=LEFT$(T$,Z-2)           'THE NAME STRING
125 NL=LEN(NA$)
130 CO$="":IF MID$(T$,Z+10,1)=";":CO$=RIGHT$(T$,LEN(T$)-Z-8)
140                             'IF THERE IS A COMMENT WE WANT
150                             '  TO INCLUDE IT
155 CL=LEN(CO$)                 'LENGTH OF THE COMMENT STRING
160 TP$=AD$+MKI$(NL)+NA$+MKI$(CL)+CO$
170 IF LEN(O$(X))+LEN(TP$)>250:X=X+1  'DONT GET TOO CLOSE TO255
180 O$(X)=O$(X)+TP$:TP$=""
200 GOTO 80
205 'THIS IS THE END OF INPUTTING EQUATES
240 '
242 'DEFINE AND OPEN THE PROCESS FILES
244 '
245 ON ERROR GOTO5005
250 LINEINPUT"What is the name of the source file (DIR OK, CMD OK) {LF}";F2$
252 IF INSTR(F2$,DIR$):CMDF2$:GOTO250
254 IF INSTR(F2$,CM$),LINEINPUT"Input your SYSTEM command  ";C$:CMDC$:GOTO250
260 OPEN"I",1,F2$
265 ON ERROR GOTO 270
270 LINEINPUT"What is the name of the output file? /CMT will be{LF}appended,
DIR is available{LF}";F3$
280 IF INSTR(F3$,DIR$) :CMD F3$:GOTO270
285 INPUT "Which drive";D$
292 F3$=F3$+"/CMT:"+D$
295 OPEN"O",2,F3$
298 ON ERROR GOTO 5050
1900 '
2000 'THIS ROUTINE FINDS THE 'M'SYMBOLS AND DECIDES IF THEY NEED TO BE
```

```
EVALUATED FOR REPLACEMENT
2002 '
2100 TA$="; This file originated as "+F2$+".  It was processed {LF}on
"+TIME$+" using "+F1$+" to become "+F3$
2110 PRINT#2, TA$:PRINT TA$        'TA$=TARGET$
2130 Z=0:LC=LC+1
2140 LINEINPUT#1,TA$:PRINT TA$,,
2150 IF EOF(1),CLOSE:PRINTLC;" lines through put":GOTO242
2160 IF LEFT$(TA$,1)=";",GOTO2110    'OUTPUT THE COMMENT LINES
2170 PO=Z+1              'DON'T WANT TO FIND THE SAME  M AGAIN
2180 Z=INSTR(PO,TA$,MM$)          'LOOK FOR ANOTHER M
2190 IF Z=0 GOTO2110   'OF NO INTEREST,OUTPUT
2200 IF Z>1 GOTO 2230   'PROCESS IT
2210 IF INSTR(PO+6,TA$,E$)<>7,GOTO2170  'WE DONT PROCESS{LF}
SYMBOLS IN THE FIRST POSITION UNLESS IN THE {LF}                EQUATES TABLES
2220 GOTO 2250          'EQUATES SKIP THE NEXT TEST
2230 Z$=MID$(TA$,Z-1,1)
2240 IF NOT(Z$=CHR$(&H09) ORZ$=","OR Z$="("),GOTO 2170
2250 M$=MID$(TA$,Z+1,4) 'THE STRING FOR WHICH TO FIND A SUBSTITUTE
2270 GOSUB 3000 'RETURNS NA$ FOR A SUBSTITUTE, CO$ FOR A COMMENT, L=0 FOR A
NOFIND
2300 'CUT AND PASTE
2319 IF L=0 GOTO 2170
2320 TA$=LEFT$(TA$,Z-1)+NA$+RIGHT$(TA$,(LEN(TA$)-Z-4))+CO$
2340 GOTO2170  'TRY FOR ANOTHER M
2990 '
3000 'SUBROUTINE TO FIND A STRING IN O$()
3010 FOR Y=0 TO X
3020 L=INSTR(O$(Y),M$)
3030 IF L : GOTO3050
3040 NEXT Y                      'NOT IN THIS STRING
3045 GOTO3100 'NOFIND, GET READY TO RETURN
3050 NL=CVI(MID$(O$(Y),L+4,2)) 'LENGTH OF THE NAME
3060 NA$=MID$(O$(Y),L+6,NL)      'GOT THE NAME
3070 CL=CVI(MID$(O$(Y),L+6+NL,2))  'LENGTH OF THE COMMENT
3080 CO$=MID$(O$(Y),L+NL+8,CL) 'GOT A COMMENT, EVEN IF IT
3085 IF CO$<>"",CO$=CHR$(09)+CO$'          IS A ""
3100 RETURN
5000 PRINT "Some sort of a file error with ";F1$
5001 RESUME47
5005 PRINT"Some sort of error with ";F2$
5006 RESUME 250
5010 PRINT"Some sort of error with ";F3$
5011 RESUME 270
5050 PRINT"Some sort of processing error" :STOP
```

EQUSUB/BAS is a basic language utility to post-process the output files created by DSMBLR/CMD. It uses information from an EQUATEs file to substitute a recognizable symbol name for a 'Mxxxx' symbol name generated by DSMBLR/CMD.

THE EQUATE FILE can be any file of the same form as EQUATE3/EQU furnished with LDOS 5.1.4. It is necessary however to get rid of the line numbers. The best way is to load EQUATE3/EQU into EDAS and then write it back out again without specifying the inclusion of line numbers. Comment lines which comprise a full line make no difference, they are ignored. Comments that follow the address in an EQUATES line, either immediately after the address or in the next tab position are included and are appended to what ever comments exist in the source line being processed. Comments can accumulate from multiple passes of EQUSUB. A proper response to the request for a file name is: 'EQ3/ASM' without the quotes. You may also make a system DIR request like 'DIR' for a look at everything, or 'DIR /ASM:2' for a look at less. Use

capital letters.

THE SOURCE FILE is an assembly source file. I know the file can be as produced by EDAS, DSMBLR or EQUSUB, you may have to work EQUSUB over to function with other sources. Comments are preserved and appended to. The file should not have line numbers included. When prompted for an input file a typical response could be: 'LOG/CMD:1' no quotes wanted or allowed. In addition to 'DIR' an optional response of 'CMD' is available. To use it answer 'CMD' and then give a system command such as 'EDAS' at the next query. You will be returned to the query point when the command has executed. If the files are not so big as to run out of memory it is nifty to go out to EDAS from this point and check out what is happening or to edit your equate file.

THE OUTPUT FILE will look about like the input file except for the substitutions. All 'Mxxxx' which had a symbol name associated with the same address in the EQU file will have a substitution made. As you become more familiar with the program you are dissasembling, you can make a new EQU file with the knowledge you have acquired and continue the symbol substitution. Multiple passes work fine. One particular substitution is never made. A symbol in the first position, except within the source files equates tables, is not touched. The logic is this. On the first pass, using EQ3/ASM, the only substitutions made are from the system equates table. These certainly cannot be a symbol defined in the source file. For subsequent passes using equates you name, you should darn well know where the symbol is defined within the program and you can change it at its single defination location with the editor.

TO USE EQUSUB with symbols you name, put the symbol in an equates file. Process the source file with EQUSUB using your new equates file and all references to the symbol will be searched out and changed to the new name you have declared. You do not actually have to make a brand new file for each newly found equate, just add it to a file you have going, along with a comment if you wish, and rerun. Try to resist adding it to EQ3 because you will forget it and possibly contaminate the next file you disasemble. The file could take the form of: 'LOG/ASM'. Again, no quotes needed or allowed. /CMT will be appended to the the spec you provide. You will then be asked for an output drive number. Respond with the drive number, no colon. The following are the files associated with EQUSUB:

EQUATE FILE is the file of symbol names,addresses and comments
SOURCE FILE is the assembly file you are rewrting. It is not modifed
OUTPUT FILE is SOURCE FILE with substitutions made. It will be appended /CMT
EQUSUB/BAS is the program that does the work.
EQ3/ASM is really EQUATE3/EQU from LDOS
EQ1/ASM is a worked over version of EQUATE3/EQU
LOG/TXT is a screening file for LDOS 5.1.4 LOG/CMD
LOG/CMD is LOG/CMD
EQUSUB/TXT is the file you are reading

THE FIRST EFFORT should be some thing simple and predictable. Disassemble LOG/CMD. You will end up with a file named LOG/ASM. Remember that the password can give you fits. Enter BASIC and run EQUSUB/BAS. Respond in sequence: EQ3/ASM, LOG/ASM, LOG, 1 or 0. When the smoke clears you should have a new file LOG/CMT The input and output will roll past as EQUSUB works. You will end up back at the point where the source file is input for another run without reinputing the equate file.

REMEMBER that EDAS has wonderful find and replace abilities. EQUSUB is at it's best on the first pass, making the system call substitutions. GOOD LUCK! John Carroll, Home Ranch #1, Sheridan Wy. 82801

EDAS/PRO-CREATE

In this issue, I'll address a number of issues concerning our macro assembler. The first item concerns a collection of enhancements that have been added via a patch - albeit a BIIIG patch. Let me give some background. LC users have for some time been frustrated by the lack of feedback during a C-language JCL compilation and assembly when the assembler aborts. The most frequent reason for the abort is a mistype in the C source which results in an undefined symbol during the assembly. The LC manual advises that the way in which to ascertain the exact error is to reinvoke the JCL using the SHOW parameter which provides an assembler listing pass by not specifying -NL. I have been troubled by this for some time and wanted to add an undefined symbol table listing. This then became one of the enhancements. Thus, if there are any undefined symbols at the conclusion of the assembler's second or third pass, they will be forcibly displayed even if you had specified -NL.

Don Brandt had requested DATE and TIME pseudo-OPs which generate the system date and time as if they were DB strings. That's been implemented. I needed to add the support of a strange pseudo-OP I called, DSYM. It is being used in a DBUG option under development for LC. The following describes the changes to EDAS that were provided with this enhancement. Things don't come for nothing. The cost was that EDAS 4.1 and 4.2 no longer support a few pseudo-OPs as explained below. The following info is included as a file on the DISKNOTES4 disk as well as a DOC file on the assembler disk.

10/16/84 - Enhancements to EDAS.

The enhancement FIX has been applied to this version of EDAS. It adds the following pseudo-OPs: DATE, TIME, DSYM, and DX. In addition, if any symbols remain undefined after the second assembler pass, a table of undefined symbols will be output after the completion of the assembly. The table is directed to the video screen and cannot be surpressed. LC users will find this table invaluable in detecting mistypes in your C-source which result in assembly aborts.

In order to add functions for the four pseudo-OPs, EDAS no longer supports the following duplicative pseudo-OPs: DEFM, DEFS, DEFW, DM. Use DB, DS, DW, and DB in their place.

One known bug has also been corrected. It relates to macros defined in the memory source file prior to use of *GET.

If you use the pseudo-OP LORG, EDAS will now specifically test for the case, LORG $, so that it forces a new load block where one is required.

The following describes the use of the added pseudo-OPs. In all cases, the use of "LABEL" is optional.

LABEL DATE
This assembles to an 8-byte date string, MM/DD/YY, as if it were a DB pseudo-OP. The actual date is retrieved from the DOS and is the sign-on date. DATE would find use to document, within the object file, the assembly date of a file.

LABEL TIME
This assembles to an 8-byte time string, HH:MM:SS, as if it were a DB pseudo-OP. The actual time is retrieved from the DOS and is the sign-on time. TIME would find use to document, within the object file, the assembly time of a file.

LABEL DSYM name

This assembles "name" as a string of characters. It would find use primarily in a macro call to be able to assemble a symbol name as if it were a DB string. EDAS does not replace a #name within a quoted string. Thus, DSYM could be used.

LABEL DX expression

This evaluates expression and assembles the 16-bit result as if it were a DB string of hexadecimal characters (4-hex digits).

The following patches implement the features:

```
. EDAS422/FIX - Enhancements to the assembler
. Applied EDAS 821163, LC 920822 - 10/23/84
X'8A99'=21 7A 8A E5 36 27 23 3A A1 56 FE 02 28 28 FE 03 28 2F EB 2A D4 7B FE
01 20 03 2A DA 7B 22 BB 8A EB CD 00 00 21 83 8A 06 0A 36 27 78 32 9D 56 E1 22
9B 56 C3 80 79 47 CD C6 81 28 ED 77 23 04 18 F6 CD F9 6A CD E4 8A 06 06 18 DE
4A CD E9 8A 4B 79 0F 0F 0F 0F CD F2 8A 79 E6 0F C6 90 27 CE 40 27 77 23 C9
X'8973'=04 44 53 59 4D 16 02 04 44 41 54 45 16 00 04 54 49 4D 45 16 01 02 44
58 16 03
X'89C1'=99 8A
X'8AFD'=32 B1 56 32 32 8B C9 CD 9C 69 3A B1 56 3D C8 32 32 8B C9 7E E6 0F 28
0A 47 CB 7E CA D2 70 04 C3 E2 70 21 E6 0F 22 C9 70 3E 7E 32 C8 70 F1 C3 36 5B
32 AD 56 3E 00 B7 C8 3E C3 21 10 8B 32 C8 70 22 C9 70 21 F9 6D CD 36 5B CD 2C
5B 2A B4 56 C3 06 71 22 00 83 C0 3A 7F 54 B7 C4 35 83 C9 2A CE 57 09 22 CE 57
2A 8C 5E C9
X'7073'=CD 2E 8B
X'5A41'=66 8B
X'7B29'=CD 4F 8B
X'7343'=04 8B
X'6EE3'=CD FD 8A
X'83F0'=CD 5B 8B

. PCREAT09/FIX - Enhancements to the assembler
. Applied PRO-CREATE 820228, PRO-LC 920183 - 10/23/84
X'69DC'=21 BD 69 E5 36 27 23 3A A1 34 FE 02 28 19 FE 03 28 20 C6 12 EF 21 C6
69 06 0A 36 27 78 32 9D 34 E1 22 9B 34 C3 9A 58 47 CD 03 61 28 ED 77 23 04 18
F6 CD D5 49 3E 63 EF 06 06 18 DE
X'68B6'=04 44 53 59 4D 16 02 04 44 41 54 45 16 00 04 54 49 4D 45 16 01 02 44
58 16 03
X'6904'=DC 69
X'6A18'=32 B1 34 32 4D 6A C9 CD 78 48 3A B1 34 3D C8 32 4D 6A C9 7E E6 0F 28
0A 47 CB 7E CA DE 4F 04 C3 EE 4F 21 E6 0F 22 D5 4F 3E 7E 32 D4 4F F1 C3 B3 39
32 AD 34 3E 00 B7 C8 3E C3 21 2B 6A 32 D4 4F 22 D5 4F 21 DB 4C CD B3 39 CD A9
39 2A B4 34 C3 12 50 22 3D 62 C0 3A 7F 32 B7 C4 72 62 C9 2A D4 35 09 22 D4 35
2A 71 3D C9
X'4F58'=CD 49 6A
X'38BD'=81 6A
X'5A46'=CD 6A 6A
X'5257'=1F 6A
X'4DC8'=CD 18 6A
X'6313'=CD 76 6A
```

This next patch is for PRO-CREATE

```
. PCREAT10/FIX - 09/20/84 - Applied PRO-CREATE 820221 PRO-LC 920174
. This fix corrects the behavior of the assembler when a MACRO
. name has been previously defined as a label.
. Requires that PCREAT05/FIX must have been installed!!!
D02,58=3A A9 52 4F C3 78 48
F02,58=00 00 00 00 00 00 00
.
```

```
D1D,3D=C3 21 38
F1D,3D=C3 78 48
. end of patch
```

The following patches are for XREF. XREF5C/FIX is for the Model I/III version while XREF6C/FIX is for the PRO-CREATE version.

```
. XREF5C/FIX - 09/20/84 - Applied EDAS 821161, LC 920817
. This patch inhibits XREF from generating an EQUATE line
. for a MACRO name if the EQU option is invoked
D03,23=CB 70 CB B0 28 01 3E AF 32 FC 55 23 5E 23 56 23 C3 56 59
. was =CB B8 CB 70 CB B0 28 01 3E AF 32 FC 55 23 5E 23 56 23 E5
D06,D9=CB 78 CB B8 E5 CA C3 55 C3 2A 56
. was =00 00 00 00 00 00 00 00 00 00 00
. End of patch

. XREF6C/FIX - 09/21/84 - Applied PRO-CREATE 820223, PRO-LC 920172
. This patch inhibits XREF from generating an EQUATE line
. for a MACRO name if the EQU option is invoked
D02,E9=CB 70 CB B0 28 01 3E AF 32 FE 33 23 5E 23 56 23 C3 78 37
F02,E9=CB B8 CB 70 CB B0 28 01 3E AF 32 FE 33 23 5E 23 56 23 E5
D06,BF=CB 78 CB B8 E5 CA C5 33 C3 2C 34
F06,BF=00 00 00 00 00 00 00 00 00 00 00
. End of patch
```

We are currently engaged in writing EDAS version 4.3. Work has almost been completed prior to my shifting over to writing NOTES. As I am writing this issue of NOTES in late October through early November, my expectations are for version 4.3 to be ready by mid-December - which is shortly after I expect that you will be reading NOTES. Therefore, it is prudent for me to officially announce the policy for updating to version 4.3 from earlier EDAS versions. Before I go into the policy, you may want to know what is going to be included in EDAS 4.3. Here's the list of features:

1. DATE, TIME, DSYM, and DX pseudo-OPs supported. EDAS 4.3 retains support of the older DM, DEFM, and DEFW.

2. Macros correctly handle string defaults.

3. Macro calls within FALSE conditionals are neither listed nor are labels added to the symbol table.

4. Intel in-line macros REPT, IRP, and IRPC are supported and can be nested within an outer REPT, IRP, or IRPC or macro as well as macros nested within.

5. Comments that are appended to macro model statements are now carried through to expansion listings; however, they can be suppressed by using a double semi-colon.

6. Macro definitions can be nested. The inner macro will not be defined until the outer macro is expanded. Macros still cannot be redefined.

7. The -WS symbol table listing no longer displays macro entries.

8. All logical pseudo-OPs are suppressed from a listing if -NC is optioned (except in the display of a macro model)

9. Leading underline now acceptable as the first character of a symbol.

10. Logical pseudo-OPs IF1, IF2, and IF3 have been added to test which assembler pass is active, pass 1, pass 2, or pass 3.

11. Binary operators: .GE., .GT., .LE., .LT., .SHL., and .SHR. have been added. The binary shift operatives are equivalent to "<" and "<-".

12. Unary operators: .HIGH. and .LOW. have been added.

13. Assembler option -MF has been added to force a search of the Macro Table before the OP code table. This permits the redefinition of Z-80 OP codes via macros via use of -MF.

14. The pseudo-OP, "REF syml,sym2,sym3,...,symn" has been added which forces a reference to the symbols placed in the argument list.

15. Print format pseudo-OPs SUBTTL, TITLE, and PAGE no longer will be printed during the listing pass. SUBTTL no longer requires opening and closing angle brackets. SUBTTL also forces a PAGE.

16. EXITM pseudo-OP added to force a premature exit from a macro expansion. It's usually used within a conditional clause.

17. The pseudo-OP, "OPTION switch,switch,switch,..." has been added to permit assembler invocation switches within the source file. They are prefixed with either a '+' or '-' to turn on or off the switch so that switches entered on the command line can be altered.

18. Macro models now support handling of #name within a quoted string via the '&' concatenation operator (e.g. DB '&#name').

19. If you omit the END statement from the main module, the "total errors" quantity will now be correct!

20. The MEM parameter has been removed. This has been long overdue.

21. The Q command now supports the execution of any command capable of being invoked from DOS Ready with the exception of any command that alters HIGH$. Q functions this way for both the Model I/III and TRSDOS 6.x version.

22. The 'l' command has been added to PRO-CREATE to alter the number of printed lines prior to a form feed.

23. The ECM parameter has been added to invoke use of the LDOS 5.1's keyboard driver's ECM mode. This function was mandatory under EDAS 4.1. It is now optional.

24. The EDAS editor is now included as a separate program, MED.

25. The macro assembler is now available as a separate program, MAS. This version will assemble directly from a disk file only; however, it will permit a larger buffer area for symbol table space.

26. EDAS is still supplied as a combination editor/assembler.

27. Model I/III EDAS 4.3 will be supported under TRSDOS 2.3 (Model I), TRSDOS 1.3 (Model III), LDOS 5.0, LDOS 5.1, and other systems compatible to either LDOS or TRSDOS (not NEWDOS-80 as I use the @PARAM DOS call).

28. Finally, Version 4.3 includes SAID, the full-screen text editor written by Karl A Hessinger. SAID is described in NEW PRODUCTS.

Okay, that's the list of features. The cost of the update is as follows:

Return of EDAS-3.4 or 3.5 master disk: $45 (includes new manual)
Return of EDAS-4.1 or PRO-CREATE 4.1/4.2 master disk: $15 (adds pages)
Outside of United States, Canada, and Mexico add $2

Requests for updates will be filled in the order they are received. As noted above, I hope to have version 4.3 ready for shipment by mid-December. If delayed, 4.3 updates will be held in queue.

This next subject responds to a request from Eugene David of Lucky Lake, SASK for ways to learn assembly language. "I commend your quest for learning assembly language; alas, I cannot recommend a text to aid you in your endeavor. The PRO-CREATE manual is not a tutorial - it is a manual which documents the use of the assembler package. The method I can recommend is to explore all of the assembler articles in magazines such as 80 MICROCOMPUTING. Second, obtain a disassembler (ours is a super one) and look at a disassembled listing of some existing program whose operation you are familiar with. Try to understand the purpose of code that is already written. Make small modifications to existing code to alter the function. It is so much easier to make changes to existing code (once it is understood) than it is to start writing a big program from scratch. It is frustrating to start with just a program that outputs a message to the screen.

An important consideration is to understand the assembly language interface to your DOS so that you don't "learn" inappropriate methods. There are some elementary books on coding around. Most are just details of the Z-80 instruction set. A few spend too much time with boolean math and number systems. The "meat" that I know you are looking for is absent or too brief. Barden's book sold by Tandy may be useful. The important thing is to research the articles in the magazines. Purchase of a usable and state of the art assembler is of secondary importance. If you're using TRSDOS 6.x, I would also recommend our book, THE PROGRAMMER'S GUIDE TO LDOS and TRSDOS Version 6.

Gary Lee Phillips of Chicago, IL still had problems with paging in PRO-CREATE when using FORMS/FLT in the printer device chain. The following may prove useful to others. Pages 19/20 of NOTES Issue II addressed the problem of the EDAS paging vs external *PR filters controling paging so I won't go into detail here. Suffice it to say that if you want pagination from EDAS (PRO-CREATE), then omit the FORMS/FLT. If you need FORMS/FLT to implement support for a soft form feed, then you will probably need to use the procedure outlined by Knute Johnson.

What I have done is to revise the patch listed in NOTES II that disables the paging and added one to change the test from 56 lines per page to a value you select. These are as follows:

```
. Patch to PRO-CREAT04 4.2 after applying PCREAT04/FIX.
. Patch disables paging check on 56 lines.
D04,90=C9
F04,90=C0
. end of patch
. Patch to PRO-CREATE 4.2 after applying PCREAT04/FIX.
. Patch changes the 56 lines per page (48H) to your value.
D04,8F=hh
F04,8F=38
. The hh is your desired value (entered as two-digit hexadecimal)
```

Nate Salsbury of New Bern NC writes, "While carefully studying your sample filter programs, I came across several mysterious entries. In TRAP/ASM (p. A-189) I find code to handle error. At those locations you point HL to the message and then you'DB 0DDH'. I can't figure THAT out." I have had other queries from people puzzling over this code (I believe that I first intro-

duced this type of code compaction in LDOS). Here's the answer to the puzzle.

Let's take a look at TRAP/ASM on page A-190 of The Guide. Indeed, the LD HL's of PRMERR and VIASET are each followed by a "DB 0DDH" instruction. A little thinking should reveal that the 0DDH value is the first byte of a Z-80 index instruction. In fact, since it precedes a LD HL,nnnn instruction, it turns the next three bytes into a "LD IX,nnnn" instruction. This has the effect of maintaining the value in HL and falling through to the @LOGOT invocation. The older way of coding a series of "LD HL,message" error exits was to follow each "LD HL,nnnn" with a "JR ERROUT" where "ERROUT" labeled the @LOGOT invocation. The "JR" instruction takes two bytes whereas the use of the "DB 0DDH" uses only one. You can get away with this if you can alter the value of IX with impunity."

There are many other examples of code compaction. For instance, a two way entry into a routine with subsequent detection of the specific entry can usually be accomplished with the following:

```
ENTRY1  DB      3EH     ;This entry sets A=0AFH
ENTRY2  XOR     A       ;This entry sets A=0
        PUSH    AF      ;Save for later test
```

So long as you can affect the value of the accumulator and flag states, this is a good way of saving some code. Many other examples can be illustrated. Maybe I'll save them for my next book on assembly language programming.

Remi Habak of Montreal, Quebec had a problem with our documentation on interfacing to PRO-CREATE with the Z-cmd and how to determine the patch space set aside for the user. Maybe the next info will clarify this. To begin with, the PRO-CREATE manual states on page 7-39 that "A vector pointing to this space is located at ... X'3609'". Perhaps I should have emphasized the word, "vector". This says that the 50-byte space is not at X'3609' but that X'3609' (and obviously X'360A') contains a pointer which points to where the space is. Thus, to find out where the space is in memory, you can load the assembler, then examine the two byte contents with DEBUG to ascertain the address. By using a vector, I can change the location of the space with iterations of EDAS assembly and still maintain correct documentation.

In PRO-CREATE, if you invoke DEBUG from within the assembler, there is no easy way that I can constantly turn it off. You have the control to do this by using the "Q" command. If you have entered DEBUG from EDAS and wish to turn DEBUG off, simply type a "Q DEBUG (N)" command.

In the patches listed in NOTES, the "applied xxxxxx" number means that MISOSYS first applied the patch starting with the number provided. If your registration number is higher, you should not apply the patch (in fact, under TRSDOS 6.x, you cannot apply the patch since PATCH would discover a mismatch error). You are correct on the page II-15 misprint. The three patches pertain to PRO-CREATE version 4.1, not PRO-DUCE. NOTES Issue I does NOT contain any patches for your version of PRO-CREATE. You may still want to get Issue I.

Ron Higgs of Anderson, SC discovered a bug in EDAS that has existed since the EDAS414/FIX was applied to version 4.1 on 4/18/83. He's the first to find it. I don't know if that means you all are not using macros or that if you are, you always load them via a *GET macrofile. Here's the answer to that puzzle.

I found the solution to your GED2 problem. Guess what, it was a bug in the assembler. The problem stemmed from your having macros defined in the main program prior to any *GET (and use of *GET later). Let me explain the scanario. Macro names are stored in the symbol table along with all other

symbols. A bit is maintained to indicate the macro. EDAS keeps a pointer to the last macro loaded into the synbol table so that macro searches can bypass symbols added after the macros. In the case of a macro, the value field contains a pointer to the macro model storage. The symbol table is expanded from top-of-memory downward while the macro model storage expands from the bottom up (the bottom meaning the unused buffer area immediately following the last program line in memory.

When you invoke an *GET, EDAS creates a 136-byte region at the top of memory. It does this by moving the symbol table down 136 bytes. It then reduces the pointer to the start of the symbol table by 136. Unfortunately, EDAS neglected to reduce the macro table pointer. Thus, since you invoked an *GET after macros were defined, any macros invoked after the *GET could not be found since the pointer to their start in the symbol table was wrong. If you would have loaded the macros via an *GET or defined one within the *GET prior to invoking any, you would not have seen the problem.

The end result is that I have provided you with a patch to fix the problem. I have been working on an enhancement patch to the assembler. Your bug is fixed along with the enhancements. [as an aside, Ron has contributed some material to the CONTRIBUTIONS section].

George Geczy of JMG Software had some questions concerning our EDAS restrictions to LDOS only. My response to George was, "EDAS version 4.1 currently runs only under LDOS (it does with DOSPLUS after application of a DOSPLUS patch and no, that patch has not sold a fair number of copies of EDAS - probably caused a fair number of pirated copies, though). EDAS version 3.4 and 3.5 ran under all the DOSses except NEWDOS-80 which really went their own way in not supporting the standard DOS call, @PARAM. No way will I support anything under NEWDOS-80. I decided to drop support for the others when I brought out version 4. This decision was based on a number of factors. First, most of my time was spent in developing LDOS and TRSDOS 6.0 (I would assume that you would be aware of my involvement with LDOS and my role as one of the founders of Logical Systems). Second, as EDAS 4.x was being developed, MISOSYS and LSI were discussing the possibility of a merger. I then decided to migrate our products to be specific to LDOS. That is why MISOSYS products are predominately supported only under LDOS (it still was a good decision as LDOS became Tandy's alternate DOS and TRSDOS 6.x is a licensed version of LDOS). In spite of the above, I am preparing EDAS 4.3 which should be available by the end of the year. It will run under LDOS, TRSDOS 2.3, TRSDOS 1.3, and DOSPLUS. Our PRO-CREATE 4.3 product will run under TRSDOS 6.x.

HELP/PRO-HELP

Jim Seitz of Lima OH wanted to change the number of lines displayed by PRO-HELP before it waited for a key entry to continue. If you have a similar need, read on. "I have analyzed the function in PRO-HELP which decides whether or not to wait for a keyboard input before returning to what invoked the help file. According to the code, it picks up and positions the cursor to the row number following the last line of your HELP screen then moves the screen data into the video screen via a SVC request of the DOS. It then compares the row value to 21. As long as the row number is 0-20, an immediate return is made, otherwise a keyboard request (@KEY) is invoked.

I did find a bug in HELPTXT/BAS in the one case where your text file contains more than 23 lines. A fragment of the code is:

```
320 PRINT "Processing ";FS$:I=1
330 LINE INPUT#2,L$(I):IF EOF(2) THEN 340 ELSE I=I+1:IF I < 24 THEN 330
340 CLOSE 2:LS=0:MS=&H28:OPEN "O",2,FS$+"/HLP:"+OD$:C=1:R=1
```

```
350 N=I:I=1:FOR I=1 TO N:C=1
440 PRINT #2,CHR$(1);CHR$(4);CHR$(&H3);CHR$(&H26);CHR$(0);CHR$(N);
```

The program does not read in any more than 23 lines inhibited by the restriction in line 330. Note, though, that when the twenty third line is read, "I" increments to 24 before the test causes the reading to cease. The variable "N" is set to the value of "I" which has counted the number of lines input from the text file. Thus "N" is one too many when the text has more than the twenty three lines permitted. Since the value of "N" is passed to the processed text output via line 440 and is used to set the cursor address, the value under text > 23 would be 24. The DOS would ignore this request. Thus line 330 needs to be corrected by adding the code: ELSE I = I - 1. This would reduce I to the correct value. Alternatively, the code could be changed to read as follows:

```
320 PRINT "Processing ";FS$:I=0
330 I=I+1:LINE INPUT#2,L$(I):IF EOF(2) THEN 340 ELSE IF I < 23 THEN 330
```

Now for your other request, you can apply a one-byte patch to the help file created by HELPGEN/CMD so that the comparison is on whatever line you choose. The patch, "PATCH helpfile (D01,B4=xx:F04,B4=15)" will change the test row value from 21 (15H) to your choice. For example, to output 22 lines before the key request, the "xx" above should be 17 (23 decimal). If you change the value to 18, there should never be a key request.

Richard A Belz of Gainesville, FL also raised some questions conserning our PRO-HELP screen on the SYSTEM command in which we make reference to the parameter, "RESTOR". Check this out. "This is in response to your questions concerning PRO-HELP. First, the keyword "RESTOR" is correct. The parameter table used in the TRSDOS SYSTEM command is the Version 5 format. Thus, parameter keywords are limited to no more than six characters. Thus, "RESTOR" is an acceptable entry. A "feature" of the @PARAM supervisor call is that for the version 5 table, a command line parameter string can contain a keyword longer than six characters. Any alphanumeric character beyond the sixth character of a keyword is ignored provided the parameter table contains a full six-character keyword. "RESTORE" and "RESTOR" are thus equivalent.

Next, the @PARAM SVC supports the following operands as equivalent: "YES", "YE", "Y", and "ON" to indicate the "true" response. Also, entering the keyword without an operand also indicates "true". Likewise, a "false" response is indicated by any of the following: "OFF", "OF", "NO", "N", or any string starting with "N" - or any string starting with "OF". A keyword followed only by an equal sign is also taken to imply a "false" response. Thus, "RESTOR=" is the same as "RESTORE=NO". See what you have just learned?

The BASICB help file included with PRO-HELP was intended to provide information on the reserved words (i.e. BASIC verbs) and not on operators. That's why we didn't include screens on AND, OR, etc. There currently is no plan on adding a screen on the operators.

Lastly, I have puzzled over your statement concerning INPUT and LINE-INPUT. I do not show a semicolon after the statement. The semicolon is provided between the reserved word and its operand. This is as shown in the L and I HELPB screens.

IFC/PRO-IFC

Here's a couple of small patches to IFC/CMD and IFCLIST/CMD which are included with the IFC and PRO-IFC packages. The first two, IFCL1/FIX and IFC2/FIX) are for IFC. The next two, PIFCL1/FIX and PIFC2/FIX) are for the PRO-IFC version.

```
. IFCL1/FIX - 09/21/84 - Applied 470035
. Apply this patch to IFCLIST to be able to CLIST files with no extension.
D00,D3=21;  WAS =CD

. IFC2/FIX - 09/21/84
. Corrects COPY of file with NULL length
D0E,46=35; WAS 19
. Corrects display header on BREAK after DOS COMMAND
D16,19=CD 66 6A; WAS 32 1F 73
D18,FD=32 1F 73 3E 02 32 20 73 C9; WAS 00 00 00 00 00 00 00 00 00
. Corrects sort order of files without extension
D07,6D=C3 6F 6A 00; WAS 1A BE 28 04
D19,06=7E FE 3A 20 02 3E 2E 4F 1A FE 3A 20 02 3E 2E B9
. WAS  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D19,16=CA 22 59 C3 1E 59
. WAS  00 00 00 00 00 00
. Change version to "b"
D1A,89="b"; WAS "a"

. PIFCL1/FIX - 09/21/84 - Applied 470023
. Apply this patch to IFCLIST to be able to LIST files with no extension.
D00,DF=00
F00,DF=EF

. PIFC2/FIX - 09/21/84
. Corrects COPY of file with NULL length
D0E,3E=35
F0E,3E=19
. Corrects display header on BREAK after DOS COMMAND
D15,66=CD 74 3C
F15,66=32 84 45
D17,03=32 84 45 3E 02 32 85 45 C9
F17,03=00 00 00 00 00 00 00 00 00
. Corrects sort order of files without extension
D07,71=C3 7D 3C 00
F07,71=1A BE 28 04
D17,0C=7E FE 3A 20 02 3E 2E 4F 1A FE 3A 20 02 3E 2E B9
F17,0C=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D17,1C=CA 26 2D C3 22 2D
F17,1C=00 00 00 00 00 00
. Change version to "b"
D18,92="b"
F18,92="a"
```

By the way, IFC was reviewed in the October 15th issue of THE LAWYER'S MICROCOMPUTER. Here's some excerpts: "IFC is a treat. It presents you with a scrollable, sorted directory on your video display screen and a collection of operations to perform on them. This is a helpful addition to one's arsenal against data loss by file glut. It would be unfair to say IFC is what the operating systems' DIR library function ought to be: IFC goes beyond what you need everytime you look at a directory."

LC/PRO-LC

Let me start out this LC column with a copy of the LCDOC/TXT file which appears on the lastest LC 1.2 disks:

>>>>> LC Documentation Update - Version 1.2a <<<<<

10/23/84: This version fixes a few small bugs in LC/LIB. Note also the new address of LCIG (Earl got married). There is also a file named EDASDOC/TXT which contains info on the enhancement fix to EDAS.

07/25/84: Known compiler bug - you are cautioned! The modulo operator, "%", and the modulo assignment operator, "%=", always use @div to perform the operation regardless of whether the rvalue and lvalue are signed or unsigned ints. This cannot be easily corrected via a patch. Therefore, the use of the modulo operator with an unsigned int may not always produce the correct results.

07/24/84: Known compiler bug - you are cautioned! The division assignment operator, "/=", always uses @div to perform the operation regardless of whether the rvalue and lvalue are signed or unsigned ints. This cannot be easily corrected via a patch. Therefore, if you are going to be dividing unsigned ints, use the expanded form of the division operator: "a = a / b".

06/01/84: Notification of passwords. Although we thought that the passwords used on the LC files would have been obvious, we sometimes get reports of inability to COPY certain files (i.e. the EXEC only LC/CMD). There is no attempt to keep you from moving the LC files for whatever needs you may have. Passwords are applied for the protection against inadvertant destruction of a valuable file. All LC password-protected files have a password of ".LC".

04/06/84: Addition of #option BREAK. The LC getc() and checkc() functions [and those functions which invoke getc(), i.e. getchar()] normally interpret a BREAK character received from a character device [such as *KI or *CL] as an end-of-file. If BREAK is optioned OFF via the statement, #option BREAK OFF, the character input functions will not treat a BREAK character as the end-of-file. This may be useful for binary input.

10/23/84: Notification of LC Interest Group. A national LCIG is chaired by Earl C. Terwilliger, Jr. Contact Earl at 943A Snowfall Spur, Akron OH, 44313 for details concerning this group. [216-928-4714].
=====> end of line.

If you are an astute observer, you will note that the most recent version of LC mentioned in the LCDOC/TXT file shown above is 1.2a dated 10/23/84. If you have a 1.2 release with a DISK DATE earlier than 10/23/84, you may want to get your disk refreshed to our current version. Now as I write this notification, I am also noting that I expect to release the version 4.3 EDAS in mid-December which includes the SAID full-screen editor. Therefore, update charges for EDAS will apply. The following table can be used to note what the charge is to get the most recent LC and EDAS (I really don't want to get too complicated but I must strive to be fair).

LC/PRO-LC 1.2 ..... $15
PRO-LC 1.1 ..... $15
LC 1.0 or LC 1.1 ..... $20

The next big item pertains to the availability of a preprocessor for LC. If you are an active hacker, you should have been aware of "p-A Small C Preprocessor" which appeared in DR. DOBB'S JOURNAL, #93 July 1984 (pp 46-82). Well both Richard N. Deglin and myself noticed it and thought it desirable to

adapt it to LC (it was designed for CP/M). I did most of the keyin work in order to test out Karl's SAID text editor while Rich did all of the work to adapt it to LC (Rich added a bunch of his filespec parsing routines). The result is a relatively complete C preprocessor that supports the preprocessor statements: #define, #define with macro substitution, #include with nesting of #includes, #undef, #if, #ifdef, #ifndef, #else, and #endif - all as described in K&R [one restriction limits a macro call to be contained entirely on one line].

The preprocessor program is on DISK NOTES 4 and is entitled PP/CCC. It is compilable under LC 1.2 - either LDOS 5.1 Model I/III or TRSDOS 6.x. A brief documentation file is entitled PPDOC/TXT. For complete info, I recommend getting a copy of the abovementioned Dr. Dobb's. If you don't subscribe, I heartedly recommend it. Dr. Dobb's is big on C articles. Contact M&T Publishing, Inc., 2464 Embarcadero Way, Palo Alto CA 94303.

Turning to my correspondance file, A. J. Williams of USEC Brussels asked about trapping I/O errors in a C program. Part of A. J.'s needs were satisfied by the introduction of ferror() in release 1.2. Here's a little of my response to A. J.

"You seem to ask the right questions at the right time. The "trapping" of the error code in LC has been added in release 1.2. What has been done is to provide a function, ferror(), that can be invoked to recover the last error code encountered on a file stream. For example, errnum=ferror(stdout); returns the error associated with standard output. With this function, you can then use the "#option ERRORMSG OFF" to suppress the display of error messages within the LC-compiled program. The display of error messages within LC is done by using the DOS @ERROR routine which automatically uses @DSPLY (via @LOGOT). TRSDOS 6.x provides a method of recovering the error message string from @ERROR. I suppose that it would be an easy task to write a new @erret function for PRO-LC to recover the message string, then output it using fputs(string,stderr). Thus, if standard error output was redirected to a bit bucket, the error messages would be suppressed but the control would be at run-time rather than at compile time.

One could also write an @erret replacement function for Model I/III that trapped @DSPLY for the duration of the function call. It could then recover the message string via the trap. A little messy, but nevertheless operational. I may consider such a function for the next issue of NOTES!"

Well this is that next issue of NOTES so I sat down and worked up a function that I am calling, errmsg(). I wrote one in C for use with PRO-LC since it was an elementary task to write and anyone could really have implemented it. The Model I/III function is a little more difficult since it has to trap @LOGOT, recover the message string, then untrap @LOGOT. If you assembler users out there are reading this column, you will find that the little routine presented for the Model I/III can be adapted to any program. In fact, I am going to adapt this to the release of DESCRIBE. The PRO-LC function is as follows:

```
/* errmsg/ccc - 10/26/84 */
/* function to provide DOS error string given error number */
#include stdio/csh
#option INLIB
#define BC 1
#define DE 2
#define IY 5
errmsg(errnum,errstr) int errnum; char *errstr;
{   int regs[6]; char *flags;
    call(101,regs);             /* get @FLAGS */
```

```
    flags = regs[IY];
    *(flags+2) |= 128;          /* @ERROR => string */
    regs[DE] = errstr;          /* point DE to string buffer */
    regs[BC] = errnum | 192;    /* return to me */
    call(26,regs);              /* @ERROR */
    while (*errstr++ != '\n')
        ;
    *errstr = '\0';             /* terminate string with NULL */
}
```

The corresponding function for LC is in assembler as follows:

```
;errmsg5/asm - 10/28/84
; function to provide DOS error string given error number
; syntax: errmsg(errnum,errstr) int errnum; char *errstr;
*M
ERRMSG  $GA     DE,HL           ;Get DE=errnum, HL=errstr
        LD      ($?1+1),HL
        LD      A,E             ;Mask for abbrev
        OR      192
        PUSH    AF
        LD      HL,447BH+1      ;@LOGOT Model I
        CALL    @MOD13          ;Check on machine
        JR      NZ,$+5
        LD      HL,428AH+1      ;@LOGOT Model III
        LD      ($?3+1),HL
        LD      D,H
        LD      E,L
        CALL    @GINT           ;Get the vector
        LD      ($?2+1),HL      ;Save for later
        LD      HL,$?1          ;Hook for @LOGOT
        CALL    @PINT           ;Change vector of @LOGOT
        POP     AF              ;Get the error code
        JP      4409H           ;Go to error routine
$?1     LD      DE,0            ;P/u the string address
        LD      A,(HL)
        LD      (DE),A
        INC     HL
        INC     DE
        SUB     13
        JR      NZ,$-6
        LD      (DE),A          ;Terminate with \n
$?2     LD      HL,0            ;P/u old vector
$?3     LD      DE,0            ;P/u where to stuff
        JP      @PINT
```

I sometimes am asked how difficult is it to learn the C language. My philosophy on this is as follows. It is based on the premise that you already know another programming language, say BASIC. C should not be totally greek. I am sure that if one re-examines their original work with BASIC, they will probably find themselves sitting there again but with C. They have an advantage over someone else that is learning C for their first language. Once you learn any programming language, you have developed skills that are fundamental in all programming languages. Thus, learning the second, third, and fourth programming language is easier than the first.

For a good basic book on C, pick up "INTRODUCTION TO C", by Paul Chirlian (Matrix Publishers, Inc., Beaverton, OR). Two C compilers were used in the writing of that book - LC was one of them. I think that you will find Paul's writing style a little easier than some of the other authors.

Dr. Enrico Savazzi of Arlington, VA pointed out that I had a problem with the chkx(), getx(), and putx() functions in PRO-LC. Seems like that if you do not reference chkx(), neither the putx() nor getx() routines are included. I told Enrico, "Yes, you are absolutely right. The reason why your program did not work properly when you referenced getx() without referencing chkx() is that the conditional statement, IFREF CHKX!GETX!PUTX, can only be evaluated based on the first label in the operand, CHKX. The assembler IFREF conditional does not currently handle the ".OR." operator. Therefore, I have composed an updated LC/ASM file which breaks apart the bank interfacing code into four modules - each surrounded with an appropriate conditional." If anyone else needs the correction, a fragment of LC/ASM which incorporates the character bank I/O routines are included here.

```
;*=*=*
;       Routines for bank character I/O
;*=*=*
*M
        IFREF   CHKX
CHKX    $GA     BC
        LD      HL,0            ;Init FALSE
        LD      A,C             ;Check for range <1-7>
        DEC     A
        CP      7
        RET     NC
        LD      B,2             ;Init for check available
        LD      A,102           ;SVC-@BANK
        RST     40
        RET     NZ
        INC     HL              ;Set TRUE
        RET
        ENDIF
        IFREF   GETX
GETX    $GA     BC,HL           ;Get BANK, POINTER
        CALL    $?3
        LD      L,(HL)
        LD      H,B
        JR      $?0
        ENDIF
        IFREF   PUTX
PUTX    $GA     DE,BC,HL        ;Get char, bank, pointer
        CALL    $?3
        LD      (HL),E          ;Put the character
        EX      DE,HL           ;Return char in HL
        ENDIF
        IFREF   $?3
$?0     LD      A,102
        RST     40              ;Invoke old bank
$?1     LD      SP,0            ;P/u old stack pointer
        EI
        RET     Z
$?2     LD      HL,-1
        RET
        DC      16,0            ;Small stack LOW
$?3     POP     IX              ;Pop the return address
        LD      A,C             ;Check range <1-7>
        DEC     A
        CP      7
        JR      NC,$?2
        DI
        LD      ($?1+1),SP      ;Save stack pointer
        LD      SP,$?3          ;Switch to low stack always
```

```
LD      B,0             ;Init for switch bank
LD      A,102
RST     40
JR      NZ,$?1          ;Return EOF on error
SET     7,H             ;Use upper 32K address
JP      (IX)
IFGT    $,7FFFH
ERR     'Bank routines must be below 8000H!!!
ENDIF
ENDIF
```

Speaking of character bank I/O, I have had some requests to include such routines for LC operating under LDOS 5.1 on a Model 4 or MAX-80 - each with the extra 64K RAM. Well Scott Loomer provided me with the routines he used in a plotting program he calls DGPLOT. This program uses an Epson bit image graphics mode to emulate a plotter with standard plotting commands. The image is actually written to the banked RAM configured as if it were a plotting surface, then to the printer. The routines are on DISK NOTES 4 in LC5/ASM.

Glen Rathke of Arlington Hts, IL wanted a patch for the EDAS Z command to invoke a "DO LC (FILE=...". A patch could always be done; however, there is another way you can do this. Look at the following JCL:

```
. LCEDAS/JCL
edas (lc
do lc (file=#f#
//exit
```

Notice that the EDAS invocation does not specify the JCL parameter. Thus, when EDAS is invoked, it operates from normal keyboard input. When you exit EDAS via the "B" command, the JCL continues and winds up invoking the LC JCL file. If you invoke this LCEDAS/JCL with something like, "DO LCEDAS (F=name", then the LC JCL will be invoked with that name also. Of course, the name should be the name of the /CCC file you will be editing.

Masaki Adachi of Livonia, MI has noticed that many texts on C illustrate the procedure of explicitly defining the type of an external function such as fopen(). Since others may have been puzzling why some of our examples do not do this, I thought it would be a good idea to share my explanation. The C statement:

```
FILE *input_file, *fopen();
```

defines "input_file" as a pointer to type FILE. It also defines "fopen()" as a function that returns a pointer to type FILE. We cheat a little in LC. We tell you that you only have to define the file pointer (*input_file) in your case). LC is implemented on a 16 bit machine. Books discuss the similarity between a pointer and an int. In most cases, LC can treat an int and a pointer alike. Since a function's return value defaults to int if not explicitly defined, LC treats the value returned from fopen() as an int. However, its utilization is the same as if it were defined as a function which returns a pointer to type FILE.

The end result of the above explanation is that it would be useful for you to continue to define the return value of the fopen() function since it would be portable across all compilers on all systems (or should be). LC will react the same way with fopen() in either case.

Notes from MISOSYS

R. W. Odlin of Sedro-Wooley, WA gave me a chuckle with his input on the results of a "eating up the stack" test done on a few compilers. I must relate his results. "The enclosed listings are versions for LC, BDS C 1.50a, C/80 verr. 3.0 and Small-C ver. TK/7.0-3 (public domain), of the same small program I write for the sole purpose of eating up the stack. No optimizations have been done for any of them, and the results are (in order of execution speed):

C/80 = 23893 recursions, crash with no error msg;
LC = 20757 recursions, crash with "stdfileopenerror";
BDS = 12171 recursions;
Small-C = 21464 recursions, reboot on conclusion;

and (CoCo) OS9 C = 183 recursions, crash with "Stack overflow" error message. (This is with an unaltered boot file eating vast amounts of memory.) I haven't tried with Dyna-C under Flex, but would expect it to be somewhere mid-range of these results.

Nothing here is conclusive, but I found it interesting to note that LC placed respectably among the sampling." R. W. went on to inquire about the "string not found" error message appearing during a LC compilation and assembly via JCL. His programs are as follows for LC, BDS, C80, Small-C:

```
#include stdio/csh
#option ZVAR
unsigned int i;
main()
{ printf("%d ",i++); main(); }
```

```
#include <bdscio.h>
int i;
main()
{ printf("%d ",i++); main(); }
```

```
#include "printf.c"
int i = 0;
main()
{ printf("%d ",i++); main(); }
```

```
#include ccio.h
int i;
main()
{ printf("%d ",i++); main(); }
```

Naturally, I had to comment to R.D. It went as follows: "I got a chuckle from the C-compiler "performance" program you sent me. Although how well a C compiled program crashes is not as important as how well it runs, it was nevertheless refreshing to know LC brought in a respectable performance. I may just write this up in the next issue of NOTES.

Turning to another issue, there is a very good explanation to the "string not found" error message you noticed during the LC/JCL execution. If you list the file, you will see the two lines, "llc" and "c/cprogram/#file#". The first, of course, loads the LC/ASM file into EDAS. The second issues a global change command which will change the statement "*GET CPROGRAM" to the statement "*GET #file#" with the "#file#" representing the name passed in the LC/JCL invocation. The global change command will always issue the "string not found" when it can't find any more occurrences of the find string.

A good example of the utility of the C language is the useful pair of programs I have entitled "enspace" and "despace". I have a need to archive a

series of ASCII files that are prepared with a text editor. Since these files have a lot of consecutive spaces within the body of the text, I felt that I could compress the spaces out prior to archival. If I needed to restore any particular file, I could then un-compress the spaces. The following enspace program removes two or more consecutive spaces from a text file by converting them to a TAB followed by a byte count that indicates the quantity of spaces compressed. The algorithm handles more than 255 consecutive spaces by outputting the compressed code sequence once that limit has been reached. Since the code used to indicate a compression is a TAB, if a tab appears in the text, it is converted to a TAB followed by a count of one. The programs are classic filter programs in that they take their input from standard input and their output is sent to standard output. Thus, the input and output files are denoted via I/O redirection (e.g. enspace <filein >filout).

```
/* enspace/ccc - 11/01/84 - Roy Soltoff */
/* program to pack spaces in a text file */
/* spaces > 1 are packed to '\t','count' */
#include stdio/csh
main()
{   int count, c; count = 0;
    while (( c = getchar()) != EOF)
    {   if ( c == ' ')
        {   if (++count < 255) continue;
            else
            {   putchar('\t'); putchar(count); count = 0; }
        }
        else
        {   if (count == 1) putchar(' ');
            else if (count)
            {   putchar('\t'); putchar(count); }
            count = 0; putchar(c);
            if ( c == '\t' ) putchar(1);
        }
    }
}
```

The following despace program un-compresses the file resulting from enspace:

```
/* despace/ccc - 11/01/84 - Roy Soltoff */
/* program to unpack "enspaces" in a text file */
#include stdio/csh
main()
{   int count, c; count = 0;
    while (( c = getchar()) != EOF)
    {   if ( c == '\t')
        {   if ((count = getchar()) == EOF)
            {   fputs("Unexpected EOF on input!\n",stderr);
                exit(-1);
            }
            else if ( count == 1) putchar('\t');
            else while (count--)
                putchar(' ');
        }
        else putchar(c);
    }
}
```

If you have the MSP-01 or PRO-GENY package, then you can use PARMDIR to generate a JCL file to invoke enspace on a collection of similarly named files using a command similar to the following:

PARMDIR /scr:3 enspace/jcl:1 (a,b="<",x=">$nam/txt:",y)

Here is a gem of a program that I threw together in about a half hour. One day while trying to debug a problem with a hard disk formatter over the phone with a user, I entered a DEBUG breakpoint at the wrong address and totally wiped out a hard disk drive's partition by generating a fresh directory [i.e via FORMAT :d (SYSTEM)]. Since I had many files on the disk, I wanted to be able to recover them. Now it is virtually impossible to reconstruct a directory if the entire directory is missing. However, since the files that I needed to recover were ASCII files (program source code, etc.), it was an elementary task to look through the disk sector by sector with a utility such as DED and note the sectors that contained pieces of the files that used to be in the directory. Now that I had the cylinder and sector numbers belonging to a file, I wrote a utility to read the disk's sectors according to the sequence of cylinder and sector numbers I had ascertained and write them out to a file on another drive. The following program, entitled generate/ccc, does this:

```
/* generate/ccc - 08/17/84 */
#include stdio/csh
#option INLIB
#define AF 0
#define BC 1
#define DE 2
#define HL 3
#define IY 5
#define CARRY 0
/*  Use.......Model I    Model III   LDOS/TRSDOS 6.x */
#define CKDRV /* 0x44B8 - 0x4209  */    33
#define RDSEC 49
#define GTDCT 81
int rc,d;
char *dct, *regs[6], buffer[256];
main(argc,argv) int argc, *argv;
{
   int spc, spt, fix_file, n_items, bgn_record, end_record;
   if (argc != 3)
   {   puts("** Bad parameters\n");
       puts("   Proper format: GENERATE <fix_file> <source_drive>\n");
       exit(1);
   }
   if ((fix_file = creat(*++argv,7)) == EOF) exit(1);
   if (( d = atoi(*++argv) ) > 7 )
       {puts("Invalid source drive number\n");exit(1);}
    regs[BC]=d;
   if ((rc=call(CKDRV,regs)) != 0)
       {printf("Drive %d not ready\n",d); exit(1);}
   if (regs[AF]&(1<<CARRY))
       {printf("Drive %d is write protected\n",d); exit(1);}
   rc = call(GTDCT,regs); dct = regs[IY];
   spt = *(dct+7) & 0x1f; spc = (*(dct+7) >> 5) + 1; spc *= ++spt;
   if (*(dct+4) & 0x20) spc += spc;
   regs[HL] = buffer;
   puts("Enter cylinder/record data... BREAK line to finish.\n");
   while (( n_items = scanf("%x %x",&bgn_record,&end_record)) != EOF)
       { while (bgn_record <= end_record)
          {regs[DE] = bgn_record;
           rc = call(RDSEC,regs); write(fix_file,buffer,256);
           if ( ((++bgn_record & 255) == spc) && spc )
               bgn_record = (bgn_record & 0xFF00) + 256;
          }
```

```
      }
   close(fix_file);
}
```

Here's another quickie that turns an object code file into an LDOS 5 or TRSDOS 6 X-patch. I call it "makex".

```
/* makex/ccc
   COMMAND FORMAT: MAKEX <binary_file> <fix_file>
   Written by: Roy Soltoff 09/11/84
 */
#include stdio/csh
#option REDIRECT OFF
FILE *bin_file;
main(argc,argv) int argc; char *argv[];
{  int c, len; unsigned int address; FILE *fix_file, *fopen();
   if (argc != 3)
   {   puts("** Bad parameters\n");
       puts("   Proper format: MAKEX <binary_file> <fix_file>\n");
       exit(1);
   }
   if ((bin_file=fopen(*++argv,"r"))==NULL)
   {   fputs("Cannot open input file\n",stderr);
       exit(1);
   }
   if ((fix_file=fopen(*++argv,"w"))==NULL)
   {   fputs("Cannot open output file\n",stderr);
       exit(1);
   }
   puts("Converting from binary input file to X-patch file\n\n");
   while (1)
   {   switch (c=getbyt()) {
           case 31:
               len = getbyt();
               fputs(". ",fix_file);       /* make a fix comment */
               for ( ; len; --len)
                   putc( getbyt(), fix_file);
               putc('\n', fix_file);
               break;
           case 2:
               fclose(fix_file); exit(0);
           case 1:
               len = getbyt();
               len-=3;     /* adjust to read one less */
                           /* byte in loop below */
                           /* and also for load address */
               address = getbyt() + 256 * getbyt();
               printf("load: length = %d, address = %04x\n", len+1, address);
               fprintf(fix_file,"X\'%04x\'=",address);
               for ( ; len; --len)
                   fprintf(fix_file,"%02x ",getbyt());
                fprintf(fix_file,"%02x\n",getbyt());
                break;
           default:
               len = getbyt();
               printf("Type %d comment: length = %d\n",c,len);
               for ( ; len; --len)
                   getbyt();
               break;
       }
       continue;
```

```
    }
}
getbyt()
{   int c;
    if ((c=getc(bin_file)) != EOF) return c;
    else
        fputs("Unexpected end-of-file on input\n",stderr);
        exit(1);
}
```

Knute Johnson of Burbank, CA writes, "I have really enjoyed the new additions to the standard library. The clearerr() and clearerof() have been extremely useful. I had planned to write a simple terminal program in assembler to use with Compuserve and some bulletin boards just to avoid using Radio Shack's cumbersome COMM program [note from me, Ugh!!!]. The new functions made it a lot easier to write it in LC. The listing is attached if you would like to use it in 'NOTES' or for whatever. To run the programs, install the comm line driver, use SETCOM to set up the RS232 parameters [note, Model I/III LDOS users just set the parms when invoking the RS232T driver], and then execute the program. To end the program press the BREAK key.

```
/*
   CTERM -  A simple Terminal Program Written in C
   Version 1.3 - 9 September 1984 - Written By: Knute Johnson
*/
#include stdio/csh
#option ARGS OFF
#option REDIRECT OFF
#option KBECHO OFF
#option MAXFILES 2
#define COMM_DEVSPEC "*cl"
#define LF 0x0a
main()
{   int c; FILE *com_in,*com_out;
    puts("\nCTERM - Version 1.3\n\n");
    if ((com_in = fopen(COMM_DEVSPEC,"r")) == NULL)
        exit();
    if ((com_out = fopen(COMM_DEVSPEC,"w")) == NULL)
        exit();
    while (TRUE)
    {   if (checkc(com_in))
            switch (c = getc(com_in))
            {   case EOF:   cleareof(com_in);
                            break;
                case LF:    break;
                default:    putchar(c);
                            break;
            }
        if (checkc(stdin))
        {   if ((c = getchar()) != EOF)
                putc(c,com_out);
            else
                break;
        }
    }
    fclose(com_out);
    fclose(com_in);
}
```

The UNIX Programmer's Manual describes the functions setjmp() and longjmp() as useful for dealing with errors and interrupts encountered in a

low-level subroutine [function] of a program. Setjmp() saves its stack environment in "env" for later use by longjmp(). It returns a value of zero as its return code. Longjmp() restores the environment saved by the last call to setjmp(). It then returns in such a way that execution continues as if the call of setjmp() had just returned the value "val" to the function that invoked setjmp(). The limitations are that all accessible data have values as of the time longjmp() was called.

These functions, although dangerous, have some merit for use. Rich Deglin worked up a model for their implementation and I tweeked them a bit. The following assembler routines implement setjmp() and longjmp() for LC use.

```
;val = setjmp(env)
;       struct
;       {       unsigned pc;    /* program counter */
;               unsigned sp4    /* (SP-4) */
;               unsigned sp;    /* stack pointer */
;       } *env;
SETJMP  POP     BC              ;get PC
        POP     DE              ;get &env
        $HS     2               ;get SP, adj for next POP
        POP     AF              ;Get (SP-4)
        PUSH    AF              ;return stack to previous condition
        PUSH    DE
        PUSH    BC
        EX      DE,HL           ;HL-->env, DE=SP
        LD      (HL),C          ;save PC
        INC     HL
        LD      (HL),B
        PUSH    AF
        POP     BC
        INC     HL
        LD      (HL),C          ;save SP-4
        INC     HL
        LD      (HL),B
        INC     HL
        LD      (HL),E          ;save SP
        INC     HL
        LD      (HL),D
        LD      HL,0            ;return FALSE
        RET
;longjmp(env,val)
;       struct
;       {       unsigned pc;    /* program counter */
;               unsigned sp4    /* (SP-4) */
;               unsigned sp;    /* stack pointer */
;       } *env; int val;
LONGJMP         POP     AF              ;discard return address
        POP     HL              ;get &env
        LD      C,(HL)          ;get old PC
        INC     HL
        LD      B,(HL)
        INC     HL
        LD      E,(HL)          ;get old SP-4
        INC     HL
        LD      D,(HL)
        INC     HL
        LD      A,(HL)          ;get old SP
        INC     HL
        LD      H,(HL)
        LD      L,A
```

```
        POP     AF              ;Get VAL to be returned
        LD      SP,HL           ;SP restored
        PUSH    DE              ;Restore (SP-4)
        PUSH    BC              ;setjmp() "args" back to stack
        PUSH    BC              ;setjmp() "args" back to stack
        PUSH    AF
        POP     HL              ;Return VAL
        RET
```

If you examine the following program which illustrates the use of setjmp() and longjmp(), the use of these functions may become evident. Pay close attention to the manner in which setjmp() is invoked. Remember that "val" will be zero on the invocation of setjmp(); however, val will be a one when longjmp() is called at the function, func3(). In order to use these complementary functions, your program must establish storage space for env as illustrated in jmptest/ccc

```
/* jmptest/ccc - 09/25/84 - Richard N. Deglin */
char env[6];
main()
{  int i, val;
   if (val=setjmp(env))
       printf("Error %d on i=%d\n",val,i);
   else
       for (i=0;i<3;++i)
       {   func1(i); printf("No error %d\n",i); }
}
func1(i) int i;
{  func2(i); }
func2(i) int i;
{  func3(i); }
func3(i) int i;
{  if (i<2) return;
   else longjmp(env,1);
}
#asm
*GET   JUMPS
#endasm
```

David B. Lamkins of Canton MA has supplied a routine for a vectored BREAK interrupt within an LC program. David's work is pertinant to the LDOS 5.1 Model I/III implementation of LC. Something similar could be done using the @BREAK SVC in TRSDOS 6.x. David first shows a C program which illustrates an example of using the vectored BREAK he calls LCBREAK.

```
/* Test Program for LCBREAK */
#include   stdio/csh
main()
{
int i, t;
   break_on( 0, myhandler );
   for ( i = 0 ; i < 256 ; i++ )
       { puts( "." ); for ( t = 0 ; t < 256 ; t++ ); };
   break_off();
}
myhandler()
{
   while ( EOF != getchar() ) {};
   puts( "\nBREAK!" );
   exit( 1 );
}
```

```
dummy()
{
#asm
*GET LCBREAK
#endasm
}
```

Next is the "#asm# code which implements the vectored break functions.

```
; Asynchronous BREAK key handler for LC programs
; Title: LCBREAK/ASM
; Author: D. B. Lamkins - Date: 28 Feb 84
; Last Edit:
;      08 Mar 84, Update documentation.
;      01 Mar 84, Update documentation.
;      29 Feb 84, Save slot from break_on for break_off.
; Language: MISOSYS EDAS IV Assembler
; Target: TRS-80 Model I/III under LDOS 5.1
; Use: From within LC program, associate BREAK key
;      with a task vector and handler routine using
;              break_on( slot, handler );
;      (slot = 0..7) and disable BREAK key using
;              break_off(); .
;      You may use break_on and break_off as desired to
;      enable and and disable the break handler.  You
;      MUST do a break_off before exiting your program,
;      otherwise the task associated with the BREAK key
;      will remain active but contain undefined code --
;      the result is invariably bad.
;
;      Note that invoking the handler automatically does
;      a break_off(); .
;
;      A typical BREAK handler could look like
;              handler()
;              {   /* first flush stdin */
;                  while ( getchar() != EOF ) {};
;                  /* then acknowledge break */
;                  puts( "\nBREAK!" );
;                  /* finally quit to system */
;                  exit( 1 );
;              } .
;
;      Note that rather than exit( 1 ), you could set a
;      global flag which would signal the main program
;      to initiate job-interrupted processing, or even
;      do that processing within handler().  Be aware,
;      however, that stdin will return EOF forever after
;      you hit the BREAK key, so further keyboard input
;      will be disabled.
;
;      If you use ZSHELL, don't use LCBREAK with any
;      program which will be take its input from a pipe.
;      The reason for this is that ZSHELL (version 2.1)
;      transfers control back to the keyboard when the
;      input pipe runs empty, requiring the user to hit
;      the BREAK key to pass an EOF to the program.  If
;      LCBREAK was installed, it would prevent use of
;      the BREAK key as EOF input.
;
; Other:
```

```
;       Install in user library using
;               pds(a) lcbreak/map user/lib (m,d)
;       where lcbreak/map contains
;               lcbreak/asm,break_on,0,break_of,0
;       and lcbreak/asm is this file.  Alternatively,
;       include this file directly using
;               dummy_routine()
;               {
;               /* define break_on and break_off */
;               #asm
;               *GET LCBREAK
;               #endasm
;               }
;
*MOD
BREAK_ON
        $HS     BC,HL           ; slot number, handler address
        LD      A,C             ; ensure valid slot num
        AND     7
        LD      C,A
        LD      (TCB@?+4),A     ; save for BREAK_OFF()
        LD      (TCB@?+2),HL    ; handler addr to TCB
        CALL    @MOD13          ; which @ADTSK, KFLAG$ ?
        LD      A,C             ; slot number
        LD      DE,TCB@?        ; TCB
        JR      NZ,LO5@?
        LD      HL,429FH        ; Mod III
        RES     0,(HL)
        JP      403DH
LO5@?   LD      HL,4423H        ; Mod I
        RES     0,(HL)
        JP      4410H
BREAK_OF        ; 8-char name limit
        LD      A,(TCB@?+4)     ; ensure valid slot num
        LD      C,A
        INC     A
        RET     Z
        CALL    @MOD13          ; which @RMTSK ?
        LD      A,C             ; slot number
        JP      Z,4040H         ; Mod III
        JP      4413H           ; Mod I
TCB@?   DW      BKCK@?          ; pre-handler addr
        DS      2               ; user handler addr
        DB      OFFH            ; no handler initially
BKCK@?  $HS     18              ; get addr of foreground
        LD      E,(HL)
        INC     HL
        LD      D,(HL)
        LD      HL,5200H        ; in O.S. ?
        AND     A
        SBC     HL,DE
        RET     NC
        CALL    @MOD13          ; which HIGH$, KFLAG$ ?
        LD      HL,(4411H)      ; Mod III
        LD      A,(429FH)
        JR      Z,L10@?
        LD      HL,(4049H)      ; Mod I
        LD      A,(4423H)
L10@?   SBC     HL,DE           ; in protected RAM ?
        RET     C
        RRA                     ; BREAK key ?
```

```
RET     NC              ; no
CALL    BREAK_OF        ; disable further BREAK
LD      HL,(TCB@?+2)    ; address of user handler
JP      (HL)            ; go to user handler
```

One of my users came upon a problem when trying to utilize a 3-character named C function in a PaDS library. In the Advanced Topics section on LC Identifier Output, The LC manual advises that LC appends a dollar sign to identifiers which are three characters or less. This is done to eliminate the possibility of using a C identifier name which happens to be a reserved word in the assembler. The problem is that the PDS(APPEND) module does not allow you to specify a member name which uses aspecial character. There are a few solutions to the problem. First, you can append the C member. Then use a file editor, such as FED, to look into the PaDS directory for the member name and change the space following the last character to a dollar sign.

An alternative procedure is to patch the PDS(APPEND) module to inhibit it from restricting the member name to A-Z, 0-9. You still won't be able to access the member using the PaDS utilities; however, EDAS will still be able to find the member via its *SEARCH facility. If you first PDS(COPY) the APPEND module out of PDS and name it LCLIB/CMD, You can then append it back into PaDS for use with LC Libraries. The LDOS 5.1 version of PDS(APPEND) needs to be patched as follows:

```
PATCH LCLIB (X'5535'=C3 48 55)
```

whereas the PRO-PaDS version of PDS(APPEND) needs to be patched with:

```
PATCH LCLIB (X'2931'=C3 44 29)
```

You then will need to use a MAP record to "lclib" the C function.

In Issue II of NOTES, Jim Frimmel (the creator of LC) presented a technique of preassembling the LC library thereby developing an object deck library file. This saves a lot of time during the development of a C program. The technique was pertinent to LC 1.1 and is almost usable under LC 1.2. The problem stems from the addition of the "%f" and "%e" translations available in the scanf() and printf() functions of LC 1.2. Within these functions is a conditional piece of code that interfaces to the ftoa(), dtoa(), atof(), and atod() floating point library routines. However, for the code to be included, the "#option FPLIB" must have been optioned. This is accomplished via the assembler statement, "@_FPLIB DEFL -1". Thus, if the program you are developing is to incorporate floating point, the preassembly must include both the standard library and the floating point library and must reference that option. If you are not going to need floating point, the preassembly technique noted in Issue II will function properly with LC 1.2.

Finally, although there is no room here in printed NOTES to document the next item, I have added it to DISK NOTES 4. Alan Cox, of Orwell OH, has been working on a post optimizer to LC for at least a half year. He has produced some useful results. At press time, his optimizer which is called "LCCO" has shown improvements of 10%-30% in execution speed of various benchmark programs. LCCO includes a macro file, LCCOMACS which is also on the disk. LCCO will be in either C-source form - if space permits - or compiled to LC and PRO-LC versions. A thanks to Alan for his valuable input.

MACH2/PRO-MACH2

The MACH2 column begins with a patch to MAPPER. The bug which necessitated this patch was confirmed by R. P. Weaver, of Lubbock TX. Seems that if you were using a hard drive with multiple heads assigned to a partition without double cylindering (i.e. no need for the DBLBIT), then MAPPER would only calculate the directory as having 32 sectors instead of the 34 actually assigned. Thus, any files that used the last 16 directory record entries would not be captured by MAPPER with the result that MAPPER would show Granule Allocation Table errors. I can recollect that about two other users reported a similar problem to me before R.P. However, I also recollect that I had requested documentation of the problem - a printout of the MAPPER listing as well as a hex listing of the DIR/SYS file. R.P. was the one to provide me with the information which enabled me to run down the bug. This points out the necessity to fully document repeatable problems which will provide us with the info to investigate and fix bugs.

```
. MAP01/FIX - 10/24/84 - Roy Soltoff - Applied 540040
. This fix corrects LDOS 5.x release of MACH2-MAPPER
. for hard drives configured to multiple heads
. without using the DOUBLE bit.
. Apply: PATCH MAPPER USING MAP01/FIX
D01,12=18;  WAS  24
D01,20=32 B9 59 FD CB 04 6E 28 13 18 0A FD CB 04 6E 28
. WAS  00 00 00 00 00 00 32 B9 59 FD CB 04 6E C8 C3 30
D01,30=0B 21 B9 59 34 3A B5 59 87 32 B5 59 3A B5
. WAS  53 CB 27 32 B3 59 C9 FD CB 04 6E C8 21 B9
D01,3F=4F;  WAS  34
D01,41=B6;  WAS  B5
D01,43=47 AF 80 0D 20 FC B7 20 01 3D
. WAS  CB 27 32 B5 59 3A B4 59 CB 27

. PMAP01/FIX - 10/24/84 - Roy Soltoff - Applied 450025
. This fix corrects TRSDOS 6.X release of PRO-MACH2-MAPPER
. for hard drives configured to multiple heads
. without using the DOUBLE bit.
. Apply: PATCH MAPPER USING PMAP01/FIX
D01,17=18
F01,17=24
D01,25=32 91 36 FD CB 04 6E 28 13 18 0A FD CB 04 6E 28
F01,25=00 00 00 00 00 00 32 91 36 FD CB 04 6E C8 C3 35
D01,35=0B 21 91 36 34 3A 8D 36 87 32 8D 36 3A 8D
F01,35=31 CB 27 32 8B 36 C9 FD CB 04 6E C8 21 91
D01,44=4F
F01,44=34
D01,46=8E
F01,46=8D
D01,48=47 AF 80 0D 20 FC B7 20 01 3D
F01,48=CB 27 32 8D 36 3A 8C 36 CB 27
```

MLIB/PRO-MLIB

By the time that you read this, the TRSDOS 6.x version of MLIB, called PRO-MLIB and written by Richard N. Deglin, should be available. Rich decided to redesign the methods he used to implement MLIB. While he was about it, he also implemented support for the Indexed Relocatable (IRL) library structure used by Digital Research in their LINK-80 linker. The IRL support was chosen because the relocatable version of EDAS (to be called MRAS) currently under development will support indexed relocatable libraries as well as Microsoft compatible REL files.

Notes from MISOSYS

Now as of this date, I am not aware of a Microsoft compatible assembler for TRSDOS 6.x such as the old M-80 for the Model I (with LDOS patches supplied by Logical Systems for using M-80 on the Model III). I have been told that Tandy was to release a Model 4 version of M-80 with their Model 4 FORTRAN package (which is supplied by Microsoft). Can anyone out there confirm this? In any event, while waiting for MRAS to appear, Rich spent a little time going over the M-80 package and came up with patches to the most recent version of M-80, L-80, and CREF-80 (these are the versions that were patched by Logical Systems) to permit their operation under TRSDOS 6.x. A pat on the back to Rich for his efforts. Note again that these patches are for the version already patched with LSI's series of Model I/III patches found on their FIX disk (contact LSI for any info needed on those patches). Also note that these patches are included on the DISKNOTES4 disk (see blurb).

```
. M806/FIX -  R. N. Deglin - 09/16/84
. patch for Microsoft Macro-80 assembler for operation under DOS 6.x
X'5203'=C3 13 52 21 00 00 45 3E 64 EF C9 00 00 00 00 00
. was   CD 5D 96 20 0B 21 11 44 22 DF 92 2E 4B 22 64 96
X'3000'=B7 C8 C5 4F 3E 02 EF C2 0B 30 79 C1 C9
X'3010'=C5 4F 3E 04 EF C2 19 30 79 C1 C9
X'3020'=B7 C8 C5 4F 3E 06 EF C2 2B 30 79 C1 C9
X'965D'=4F 3E 1A EF C9 00 00 00 00 00 00 00
. was   3A 25 01 FE 49 C9 C3 73 44 13 43 C9
X'92D5'=CD 5D 96
X'92DE'=CD 06 52
X'9363'=3E 4E EF; was   CD 1C 44
X'9378'=3E 4F EF; was   CD 63 96
X'9386'=3E 4E EF; was   CD 1C 44
X'939B'=3E 4F EF; was   CD 63 96
X'93A7'=3E 4E EF; was   CD 1C 44
X'93B5'=3E 4F EF; was   CD 63 96
X'94A3'=CD 00 30; was   CD 33 00
X'94D3'=3E 09 EF; was   CD 40 00
X'94FF'=3E 3A EF; was   CD 20 44
X'9557'=3E 3B EF; was   CD 24 44
X'958D'=3E 3C EF; was   CD 28 44
X'95F5'=CD 10 30; was   CD 1B 00
X'9602'=CD 00 30; was   CD 33 00
X'960C'=CD 20 30; was   CD 3B 00
X'9632'=CD 10 30; was   CD 1B 00
X'9646'=3E 03 EF; was   CD 13 00
X'966C'=3E 16 EF; was   C3 2D 40
X'9795'=3E 4E EF; was   CD 1C 44
X'979F'=3E 4F EF; was   CD 63 96
X'97A8'=3E 3B EF; was   CD 24 44
X'97C3'=00 00 00 00 00 00 00 00 00 00 00
. was   CD 5D 96 20 06 21 33 30 22 D2 97
X'97D1'=3E 12 EF; was   CD 70 44
X'9810'=3E 0A EF; was   CD 67 44
X'6B0D'="DOS6"; was   "LDOS"
X'92BF'=00 00; was   D6 20
X'965A'=00 00; was   D6 00

. L806/FIX - R. N. Deglin - 09/15/84
. patch for Microsoft Link-80 link editor for operation under DOS 6.x
X'5203'=C3 13 52 21 00 00 45 3E 64 EF C9 00 00 00 00 00
. was   CD F9 58 20 0B 21 11 44 22 28 52 2E 4B 22 56 60
X'3000'=B7 C8 C5 4F 3E 02 EF C2 0B 30 79 C1 C9
X'3010'=C5 4F 3E 04 EF C2 19 30 79 C1 C9
X'58F9'=4F 3E 1A EF C9 00; was   3A 25 01 FE 49 C9
X'52EC'=CD F9 58; was   CD 09 44
```

```
X'5227'=CD 06 52; was   2A 49 40
X'604C'=3E 4E EF; was   CD 1C 44
X'6055'=3E 4F EF; was   CD 73 44
X'5898'=CD 00 30; was   CD 33 00
X'555B'=3E 09 EF
X'72D3'=3E 3A EF; was   CD 20 44
X'58D1'=3E 3B EF; was   CD 24 44
X'58B5'=3E 3C EF; was   CD 28 44
X'7340'=3E 3C EF; was   CD 28 44
X'73A1'=CD 10 30; was   CD 1B 00
X'58E5'=3E 03 EF; was   CD 13 00
X'590B'=3E 16 EF; was   C3 2D 40
X'53C0'=01 0B 59; was   01 2D 40
X'54CC'="DOS6"; was   "LDOS"

. CREF806/FIX - R. N. Deglin - 09/16/84
. patch for Microsoft Cref-80 cross reference utility
. for operation under DOS 6.x
X'5203'=C3 80 5C 21 00 00 45 3E 64 EF C9 00 00 00 00 00
. was   CD 39 5C 20 0B 21 11 44 22 1A 59 2E 4B 22 40 5C
X'3000'=B7 C8 C5 4F 3E 02 EF C2 0B 30 79 C1 C9
X'3010'=C5 4F 3E 04 EF C2 19 30 AF C1 C9
X'5C80'=21 89 5C 3E 0A EF C3 13 52;  was garbage
X'5C89'="Cref-80 DOS6 Ver. 3.43a Copyright (c) 1981 Microsoft"
X'5CB3'=0D
X'5C39'=4F 3E 1A EF C9 00; was   3A 25 01 FE 49 C9
X'5910'=CD 39 5C; was   CD 09 44
X'5919'=CD 06 52; was   2A 49 40
X'599B'=3E 4E EF; was   CD 1C 44
X'59BA'=3E 4E EF; was   CD 1C 44
X'5C3F'=3E 4F EF C9 00 00; was   C3 73 44 ?? ?? ??
X'5AA8'=CD 00 30; was   CD 33 00
X'5AD8'=3E 09 EF; was   CD 40 00
X'5B04'=3E 3A EF; was   CD 20 44
X'5B4E'=3E 3B EF; was   CD 24 44
X'5B7C'=3E 3C EF; was   CD 28 44
X'5BD7'=CD 10 30; was   CD 1B 00
X'5C09'=CD 10 30; was   CD 1B 00
X'5C1D'=3E 03 EF; was   CD 13 00
X'5C48'=3E 16 EF; was   C3 2D 40
X'58FA'=00 00; was   D6 20
X'5C36'=00 00; was   D6 20
```

MSP-01/PRO-GENY

The MSP-01/PRO-GENY column starts out with two patches to the PRO-GENY release. The DOCON1/FIX concerns itself with correcting DOCONFIG for use with Job Control Language and for operating it with TRSODS 6.2. It seems that LSI made a change to the SYSGEN module of SYS8/SYS which negatively impacted the linkage that I coded to utilize that module (boy that sounded good!). All right, you say that with SVC's I should not be doing strange interfacing. Well it turns out that some of the more esoteric stuff that we do such as DOCONFIG and ZSHELL require some cutesy stuff. Pete Carr of Port Orange, FL first exposed the JCL bug in DOCONFIG and the problem in MEMDIR. It took a while for him to get across his point concerning MEMDIR. Persistance paid off. I told him, "After I sent you my last letter, I decided to look into the MEMDIR problem you raised (i.e. low memory directory still continues to show modules after removal). I believe that I discovered the reason for the bug. After receiving your July 30th letter which detailed the command sequence, I became positive that I understood what you were getting at.

The bug was traced to a routine that determined when MEMDIR was through tracing low memory. I had no code to see if the current pointer was equal to the value maintained by the system that indicates the first available byte of low memory. My test just checked for an invalid module header as was done in the high memory directory. Once you install a module, even if you DOCONFIG to another configuration, if the newer one had no extra low memory used (beyond the system drivers), MEMDIR's trace would not detect a bad header." Here's the DOCONFIG patch.

```
. DOCON1/FIX - 08/20/84 - Applied starting with 230060
. This fix is for PRO-GENY version of DOCONFIG
. It corrects operation for JCL and 6.2
.
. Fix operation under JCL
D01,EE=3A C5 2B B7 00
F01,EE=11 00 00 7B B2
. Remove extraneous byte
D02,B2=00
F02,B2=DD
. Adapt to 6.2 under version test
D00,68=CD B7 2D
F00,68=CD EF 2B
D03,86=CD EF 2B 3E 65 EF FD 7E 1B FE 62 C0 3E 8C 32 3E 2A 3D 32 49 2A C9
F03,86=00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
. End of patch
```

The following patch is also for the PRO-GENY package. It is used to cure a minor blemish in the low-memory display which would have listed a previously loaded module no longer accessible because of a re-boot.

```
. MEMDIR1/FIX - 08/20/84 - Applied starting 230060
. Apply patch to PRO-GENY's MEMDIR/CMD program
. Corrects low-memory directory on modules removed
X'2699'=CD 93 29
X'26D7'=C3 8C 29
X'298C'=22 99 29 D1 C3 09 27
X'2993'=23 7C B5 C8 EB 21 00 00 ED 52 EB C0 E1 C3 31 26
. End of patch
```

The next piece discusses an item concerning MSP-01. On page 12 of the documentation, it provides an example of involving a DO command with the statement, **DO DOFILE/JCL:1 (A="LIST",X="(HEX)").** It's a bad example because it just won't work under Model I/III LDOS. The reason is two fold. First, DO does not accept double quotes in the parameter. Second, the first closing parenthesis would terminate the parameter parsing. I'll not go into how this particular example got into the manual. As an aside, in the case of TRSDOS 6.x, the above DO command would work because I added support in the DO compiler to handle quoted strings as the argument of a parameter. All characters within a quoted string are accepted. The string terminates on the closing double quote, carriage return, or ETX.

Apparently, my users are very devoted. Some believe that if I say it can be done, then it can be done. Folks, we do make mistakes from time to time. However, I must admit that this user in England was quite inventive in an attempt at making it work. He wrote, "Last month I bought this disk and DO-AUTO, DOCONFIG, and MEMDIR work beautifully, I wonder how I ever managed without DOCONFIG. However, PARMDIR does not work as documented. All is as in the manual until page 12, but the JCL command at the top of page 12 [see above] does not work as the literal string in [double quotes] is not translated by LDOS. If these are missed out then the A=LIST works but the brackets

in X=(HEX) causes a parameter error. A way of overcoming the difficulty should be to use the "%hh" feature of JCL as described in PARMDIR page 13, **PARMDIR /CMD DOFILE/JCL:1 (A=LIST,X="%28",Y,Z="%29").** This does not work as for some perverse reason the JCL file looks like

#A# AJEDIT/CMD:0 %28#Y# %29

and the space following #Y# produces a parameter error. To overcome this use,

PARMDIR /CMD DOFILE/JCL:1 (A=LIST,X="%28",Y,Z="%08%29")

which incorporates a backspace (08) into the Z to fool the command line interpreter into thinking that the space following (HEX does not exist.

This all seems very perverse and I wonder if PARMDIR needs some attention? I am reticent to suggest that there might be an error in the program because it comes from a stable with an excellent track record."

To further clarify the problem, here's my response. "The problem in question was the example being illustrated on pages 11 and 12 of the manual. In this example, the "A" and "X" parameters were being used to generate a JCL file of the form, "#A# filespec #X#". The example went on to suggest an invocation using a command line of,

DO DOFILE/JCL:1 (A="LIST",X="(HEX)")

which, of course cannot work under LDOS 5.1 (it does work under 6.x). The reason it cannot work is that LDOS 5.x JCL does not expect nor can it accept quoted strings as the operand of a parameter token. In addition, the only characters acceptable for the operand string are "A-Z", "a-z", "0-9", "/", ".", and ":". Thus, even if the quotes are omitted, the parentheses are rejected by JCL and the operand of X in the above case would be interpreted as a NULL string. In LDOS 6.x (alias TRSDOS), I added the support of a quoted string which would then accept any character within the quotes.

JCL does not support the percent operator's use anywhere except the JCL file. Thus, you cannot fool the JCL command line interpreter to accept the "%28" for a token parameter operand. Your attempt to provide a left and right parenthesis via PARMDIR parameters of X="%28",Y,Z="%29" would have almost worked. A space would be provided after the #Y# which is standard for the way in which the PARMDIR fields are generated. This then leaves a space between the parameter field and the closing parenthesis which is prohibited. Since the closing parenthesis is not needed, that can be dropped. Thus there would not be a need to use the kludgy backspace. I commend the user for his/her thinking. I note that with the %28 placed into the generated JCL file, it would require an entry for the #Y# token.

To sum up, the PARMDIR example on pages 11/12 is wrong. The manual will be corrected at the next printing. Perhaps I should explain the matter in a future NOTES."

Turning to the PARMDIR supplied with PRO-GENY, Pete Carr of Port Orange, FL came upon a bug when trying to input the parameter input from the *KI device [via a parameter of PARMS=*KI]. My response: "The PARMDIR supplied with PRO-GENY does indeed have a bug with parameter input from *KI. You are the first one to report it. After investigating the bug, I have come up with the following patch which will appear in NOTES IV:

. PARMDIR2/FIX - PRO-GENY version - 07/26/84 - Applied 230058
. This patch corrects character device input for PARMS.
. Apply to TRSDOS 6.x PRO-GENY release only!
.

```
X'3239'=CD A7 3F
X'3B92'=C3 A1 3F
X'3FA1'=3E 03 EF C8 B7 C9 4F 3E 02 EF C9
. eop
```

As an aside, there is also a misprint in the manual where you referenced the item on page 15, paragraph 2. The correct syntax is, PARMS="*KI". I omitted the "S" from the doc. PARMS can also be abbreviated as "P"."

PaDS/PRO-PaDS

Sidney L. Bloom of Frederick, MD had a problem with my patches listed in NOTES Issue 3 on page 3-37 used to alter the characters surrounding a member spec when invoking a PaDS member. You should recollect that it was Sidney's requst that I was satisfying with those patches. Recollect also that NOTES 3 provided PPADSD/FIX which reorganized the PRO-PaDS module.

I wrote back to Sidney, "The PRO-PaDS patch mystery has been solved (I think). My letter to you which provided the patch was dated February 8th. The procedure to implement the PPDSD/FIX was dated April 4th. This fix reconstructed the PaDS file and obviously changed the location that should have been patched. Unfortunately, I listed, in NOTES, the patch that I gave to you which is wrong for anyone implementing PPADSD/FIX. Okay, why could you not locate the correct values to patch in your PRO-PaDS when I sent you the patch? The clue is that you probably never applied the PROPADSA/FIX which was listed in NOTES ISSUE II undated but reflecting that I had applied the fix effective with #220044. Your disk is number 220039. This fix added code to the APPEND module which occurs before the BUILD module. Thus, when PRO-PADSA/FIX was applied, it pushed the BUILD module farther down in the file. We were out of sync! The bottom line is that I worked up your two patches on a disk that had PROPADSA/FIX installed. This patch wasn't going to work on your disk. When you finally sent me your disk to get the "latest" version, I had already changed my master disk. Again, the patch would not be correct. Guess what? I worked up the patch for the latest version of PRO-PaDS. Here it is:

```
. PROPADS/FIX
. The following removes the restriction for ")"
D00,4D=21 38
F00,4D=0D 28
D23,F0=21 38
F23,F0=0D 28
. The following changes the "(" to the character of your choice
D00,32="c"
F00,32="("
D23,D5="c"
F23,D5="("
```

Hope this helps." I guess that it pays to keep current with the patches listed in NOTES - or send in your disk for a $5 refresh.

William Altaffer of Enid, OK suggested that there is a bug in PaDS that keeps it from accessing certain members in an LC assembler source library. Since this "problem" pertains to anyone using PaDS to deal with LC libraries, my response needs to be aired. William also suspected a problem in the inability to invoke similarly named members such as T, TE, TES.

"There is no bug in PaDS that inhibits the listing of FP/LIB members. When PaDS was first implemented, it was designed to hold executable object code files as members. I even permitted data files as members. One restric-

tion was that the member name be a valid file name. On one hand, a member name was a member's original file name. All modules that make up PaDS use a check routine to screen a member name field.

When LC was being put together, I did not want to invent another library structure; thus, I chose to use the PaDS tool, which was already available. However, filenames and member specs as well are restricted to the alphanumeric character subset A-Z, 0-9. Since the LC libraries are composed of assembler source code, and EDAS uses a technique of the member name being a defining label in the member, it is sometimes necessary that a library member be named with a special character (i.e., "@" or "$" or "_"). Members named thusly are appended to the library with a patched PDS(APPEND) module. The patch removes the routine that checks the member name for "validity"; thus, member names can be anything. It turns out that we use the special characters so that a user cannot inadvertantly construct a function name, statement label, or variable that is identical to those library functions.

The second "suspected" bug is not a bug. You cannot have two similar names in a PaDS file without the shorter one being first. This follows from the "feature" that was implemented to allow you to abbreviate the member name when invoking an executable member. The abbreviation feature is noted on page 4 and again on page 5 of the PaDS manual. Of course, the shorter vs longer limitation may not be evident at first glance since I do not make specific mention of it."

To set the record straight, PaDS has a feature that permits you to abbreviate a member name when invoking it from the command line. This feature was coded specifically to provide you a means to minimize the keystrokes necessary to invoke a member. However, there arises a problem if you have a member whose name is longer than another but begins with the same character string as the shorter. For example, if you had added a member named TESTIT to a PaDS and subsequently appended a member named TEST, you cannot invoke the member named TEST since the PaDS Front End Loader will "correctly" find TESTIT first and it will match your request. There will be no way to invoke TEST via a command line invocation. Both modules will, however, be accessible to PaDS commands since the internal commands require the full member name. If you are going to have two or more modules similarly named, then you should append the shorter-named member to the PaDS prior to the longer-named member.

David F. Roberts of Cirencester, ENGLAND writes, "In response to your invitation to send patches for MISOSYS products I enclose the following patch for the append member of PaDS. One of the main uses I find for the programme is for archiving letters, old versions of programmes, etc.. As such, I feel it is rather useful to have the original creation date of the file listed in the directory rather than the date of the addition to the PaDS. I have developed this patch to enable this. I can't forsee any problems with using it but if you do please let me know!!" Well I did not see any problems and thought it a godd idea. I made one small change to the patch. I changed the code stream: LD HL,FILEFCB+6 / LD C,(HL) / INC HL / LD B,(HL) to the single statement, LD BC,(FILEFCB+6). I also worked up a similar patch for PRO-PaDS. In order to apply this patch, you will have to copy the APPEND module out of the PaDS to patch it. The following statements will do this:

```
PDS(C) PDS(APPEND) APPEND
PATCH APPEND PDSC
PDS(K) PDS.PDS(APPEND)
PDS(PURGE) PDS.PDS
!APPEND APPEND PDS.PDS
```

Note the exclamation point in the fourth line. If you are not aware of its utility, it forces the DOS to invoke the /CMD program that has a name ident-

ical to a DOS library command. Here's the Model I/III patch which I'll call PDSC/FIX.

```
. PDSC/FIX by David F. Roberts
. Patch to write file modification date to PaDS directory
. instead of date of addition.
X'52E6'=18 16
X'5966'=C2 F2 55 ED 4B 70 58 CD 10 4B C2 F2 55 23 D5 11
X'5976'=F3 58 7E E6 0F 12 23 13 7E 12 D1 C9
X'53E9'=CD 66 59
. end of patch
```

The corresponding PRO-PaDS patch is:

```
. PPADSD/FIX Suggested by David F. Roberts
. Patch to write file modification date to PaDS directory
. instead of date of addition.
X'26CE'=18 19
X'2D75'=C2 EE 29 ED 4B 7F 2C 3E 57 EF C2 EE 29 23 D5 11
X'2D86'=02 2D 7E E6 0F 12 23 13 7E 12 D1 C9
X'27DC'=CD 75 2D
. end of patch
```

PaDS can be useful for other purposes. My new DESCRIBE program is a strange animal. On one hand, it incorporates some interesting features such as the ability to read or write a DIF file of directory data. It also has a small screen editor as well as a user customizable directory display module. It also permits the invocation of the directory display from DOS Ready. All of these functions take up code space. I wanted to minimize the amount of time it takes to load the program for the directory display function. I therefore came up with the idea of using a PaDS file structure where one of the members is a fully functioning user directory command. I did not want to have separate overlays because I did not want to clog up a user's disk with extra files. The PaDS approach is great for combining a main-line program with its overlays all in one file.

I went a little further and wanted to make it easier for a user to invoke DESCRIBE without having to use the complete PaDS syntax which specifies the distinct member module via the entry of the memberspec. I decided to modify the Front End Loader (FEL) to automatically invoke a particular member if the member name was omitted from the command invocation. I chose to use the default name of "main". Thus, by using this new FEL, I could effect the invocation of the member named "MAIN" just be entering the PaDS name. The FEL is included on DISK NOTES 4 in both LDOS 5.1 and TRSDOS 6.x modules. They are named FELMAIN5/CMD and FELMAIN6/CMD respectively.

Finally, Richard J. Edgar of Madison WI writes, "I thought you'd be interested in knowing that I'm very impressed by the quality of the NOTES; they make the use of the MISOSYS products I've purchased much more enjoyable.

Inspired by your comments about using filters in PaDS's, I've done a bit of hacking that might be of interest to your readers [Note: Richard's comments which follow pertain specifically to the LDOS 5.1 Model I/III version of PaDS. The content, of course, can be modified to be applicable to the TRSDOS 6.x version, PRO-PaDS. Such a modification may appear in the next issue of NOTES.] First, I changed your FEL a bit, so that it saves the DCB pointer on the stack before loading the member, and so it returns to a point after the call to @LOAD in the (still resident) FILTER (or SET) command. I found it was also necessary to change the PDS(append) command so that it allows putting filters into a PDS as program members (without the bother of renaming all my filters xxxx/cmd first). So here's what I did.:

In the standard PDS FEL, I inserted after the label GOTBGN these two instructions:

```
PUSH  DE        ;Points at DCB
LD    DE,5B58H  ;Points at FILTER's FCB
```

Then after LINPTR does its LD HL,$-$, I inserted "EX (SP),HL" and replaced the "LD HL,(@RUN+1)" through "ADD HL,DE" with

```
LD    HL,5A27H  ;After CALL @LOAD in FILTER
```

so that the return after loading the member comes to the FILTER command (or SET), allowing the registers to be properly set up prior to executing the filter/driver. These changes should work under LDOS 5.1.4 on either the Model I or the Model III version; perhaps something similar can be done for the 6.x, but I don't have it, so I wouldn't know.

To append filters to a PDS as program members, I implemented a new parameter, EXT, which allows disabling the extension checking piece of PDS(append). Thus typing,

pds(a) pr/flt fltpds/flt.pds (ext=off)

will allow PR/FLT to be appended as a program member, despite the fact that the extension is /FLT. To accomplish this, I disassembled PDS(append) using the attached screening file. I then inseredd after a SET 7,(HL) line that loads at 543AH the following:

```
        LD      DE,-1           ;EXT param defaults ON
EPARM   EQU     $-2
        LD      A,E
        OR      D
        JR      Z,FRSTBYT
```

and affixed the label FRSTBYT to the LD BC,(M550E) line that loaded at 5456H originally. The parameter table was extended also (at 5867H) with:

```
        DB      'EXT  '
        DW      EPARM
        DB      'E    '
        DW      EPARM
```

Also, the data section at the end (all those EQU's) need to be changed to DS's before you can re-assemble it. I also integrated the call to the patch into the code.

I hope this is decipherable. Keep up the good work!" My thanks to Richard for his expert detective work. Richard also supplied the following screening data file for use with the disassembly of PDS(append):

$5634-569A,$569B-56BA,$56BB-56C3,$56C4-56D1,$56D2-56EA,$56EB-570B,$570C-5723
$5724-5740,$5741-575E,$575F-578A,$578B-57A9,$57AA-57CD,$57CE-57E4,$57E5-57F5
$57F6-580C,$580D-5825,$5826-5845,$5846-5848,$5849-584E,#584F,$5851-5856
#5857,$5859-585E,#585F,$5861-5866,#5867,5869,$594E-5965.

PROGRAMMER'S GUIDE

David A. Smith of Berkeley Hts., NJ wanted a filter to add a linefeed after a carriage return towards an output device. I provided him with the following. "I can't go into total detail for the ADDLF filter but it is easily worked up using the TRAP filter as a base. The following code will perform the filtering part:

```
ADDLF   JR      START
OLDHI   DEFW    $-$
        DB      MODDCB-ADDLF-5
        DB      'ADDLF'
MODDCB  DEFW    $-$
        DEFW    0
;
START   JR      Z,TEST
        JR      NC,OUTP1        ;Don't check @CTL calls
TEST    LD      A,C             ;Check on ENTER
        CP      13
        JR      NZ,OUTP1        ;Just put it out if not ENTER
        CALL    OUTP1
RX01    EQU     $-2             ;Relocate this address
        LD      C,10            ;Now put the
```

The second edition of THE PROGRAMMER'S GUIDE TO LDOS/TRSDOS VERSION 6 incorporated material applicable to the 6.2 revision of TRSDOS 6. Some of the people who purchased the first edition have asked for the 6.2 material. Since our practice is to not provide updates to books [I am not aware of any publisher providing updates to books], I thought it practical to at least convey this information in NOTES. Of course, if you purchased a copy of TRSDOS 6.2 from Radio Shack (the only source for TRSDOS 6.2), you should have received documentation on the changes to the Tandy Technical Reference Manual. In any event, the following excerpts the changes in content of the pertinant TRSDOS 6.2 material.

The COM driver data area changed the contents of DATA+3. It now includes the following:

> +3 - Flag to indicate KFLAG$ support [1 = ON; 0 = OFF]. Effective with LDOS 6.2.0, this byte contains the BREAK character code, LOGBRK. If non-zero, then reception of that byte value from the communications line will cause the BREAK bit of the KFLAG$ to be set. If zero, no input character will be interpreted as a BREAK.

The following references the changes and/or additions to SVCs.

@CKBRKC SVC-105

This SVC was installed effective release 6.2.0. It checks to see if the BREAK key has been pressed. It also clears the BREAK bit of the KFLAG$ if a break condition is detected.

> Registers Affected: AF.
> Z <= BREAK was not detected.
> NZ <= BREAK was detected. SVC returns only when BREAK is released.

@CLS SVC-106

This SVC was installed in release 6.2.0. It will clear the video screen via an @DSP of HOME and CLEAR-TO-END-OF-FRAME.

Registers Affected: AF
Z <= Set if no error was encountered, otherwise reset (i.e. NZ).
A <= Contains the error code under an NZ condition.

@FLAGS$ SVC-101
AFLAG$ - added 6.2.0

This "allocation" flag contains the starting cylinder number that is used by the system's file space allocation routine when searching for free space on disk media. The system defaults this value to cylinder 1.

DFLAG$

Bit 3 - If set, it indicates that SYSTEM (SMOOTH) is active.
Bit 5 - If set, it indicates that FORMS is active.
Bit 6 - If set, it indicates that KSM is active.

EFLAG$

This flag byte is used to indicate the presence of an Extended Command Interpreter (ECI) program in the SYS13/SYS slot. A non-zero value indicates that the user's ECI be used to interpret the command line in lieu of the system's command interpreter. On entry to your ECI, bits 4-6 of this flag are imaged in the accumulator and are available for immediate test.

IFLAG$

This flag is used in international systems. Bit assignments are:

Bit 0 - Set to indicate French.
Bit 1 - Set to indicate German.
Bit 2 - Set to indicate Swiss.
Bits 3-5 - reserved
Bit 6 - Special DMP mode on/off.
Bit 7 - Set 7-bit ASCII mode on/off.

LFLAG$

Bit 6 - Reserved for Interrupt Mode 2 hardware.
Bit 7 - Reserved for Interrupt Mode 2 hardware.

NFLAG$

This "network" flag is used for control in network situations. The bits are assigned as follows:

Bit 0 - If set, the "file-open" bit will be written to the directory when a file is opened with update or higher access.
Bits 1-5 - reserved
Bit 6 - Set if the system's task processor is in control. NOTE: do not execute an EI instruction within any driver or filter routine if this bit is set.
Bit 7 - reserved

PFLAG$

This flag is assigned to printer operations. Bits are as follows:

Bits 0-6 - reserved
Bit 7 - Set to 1 if the SPOOLer is in a paused state.

TFLAG$

This is the machine type flag. It's value indicates the computer model running the DOS. Some of the typical TRS-80 values are: 2 = model 2; 4 = model 4; 5 = model 4P; 12 = model 12; 16 = model 16, 80D = MAX-80.

UFLAG$

This is a user flag. It is available for whatever purpose you wish to make of it. It will remain unused by the system; however, the flag contents will be part of any SYSGEN configuration file.

@VDCTL SVC-15

A CAUTIONARY NOTE. In release 6.2, the video block transfer functions 5 and 6 that move a "2K" block to/from the video screen no longer move a full 2K. Instead, the move is only of 1920 bytes, not the 2048 that was available under TRSDOS 6.0 and TRSDOS 6.1. Thus, if you had attempted to utilize the additional video RAM for storage of data, etc., it is no longer accessible via the @VDCTL SVC. Another change to VDCTL was the addition of the video line transfer function.

VIDEO LINE TRANSFER

Registers Affected: AF, BC, DE, HL.
B => 9; Invoke line transfer
C => transfer direction; 0 = buffer to video, 1 = video to buffer.
H => video row to transfer (0-23).
DE => A pointer to the user's 80-character buffer.
A <= Will contain the error code if an error was encountered.
Z <= Set if the operation was successful.

ZGRAPH/PRO-ZGRAPH

Paul Rehberg of Houston TX writes, "PRO-ZGRAPH is the best software I have purchased in the past year! I hope you are paying K. Hessinger handsomly. With ZGRAPH from MISOSYS and BSORT from LSI, I finally feel I'm getting my $$ worth out of the Model 4. Thanks for a great product at a good price."

Pasquale A. Mancuso of Trenton, NJ brought to my attention a problem he was having with the TRSDOS 6.x version of BINPLAY, one of the applications included with the ZGRAPH package. Seems that under TRSDOS 6.2, screen images starting from the second would be displayed incorrectly even though the /MBF file would look proper while in ZGRAPH itself. After researching the problem, I responded to Pasquale, "I have investigated the problem that you were experiencing with BINPLAY while operating under TRSDOS 6.2. It appears that LSI changed the operation of one of the supervisor calls in TSDOS 6.2, Specifically, VDCTL (SVC-15) has two functions for moving a block of RAM to or from the video screen (functions 5 and 6). Under TRSDOS 6.0 and 6.2, exactly 2048 bytes were moved (this is more than a screen size; however, the video RAM is 2048 bytes in size). Under TRSDOS 6.2, LSI decided to use the undisplayable video RAM for the storage of system data and thereby reduced the length of the block moved under VDCTL to 1920 bytes. BINPLAY relied on a move of 2048 bytes.

To correct the incompatability of BINPLAY with TRSDOS 6.2, I worked up a patch which to permit correct operation under 6.0, 6.1, and 6.2. In fact, it should behave properly under future releases of the DOS. The patch has been applied to the BINPLAY on the disk you forwarded to me which I am returning to you. Sorry for the inconvenience." The following is the patch that needs

to be applied to the PRO-ZGRAPH release of BINPLAY/CMD.

```
. PBPLAY1/FIX - 11/08/84 - Applied 410067
. Patch to correct BINPLAY's operation under TRSDOS 6.2
. due to change in VDCTL function 5 (buffer to video)
D01,21=CD 16 32
F01,21=3E OF EF
D02,28=21 OD E5 3E OF EF E1 01 00 08 09 C9
F02,28=20 66 6F 72 20 61 76 61 69 6C 61 62
. End of patch
```

ZSHELL/PRO-ZSHELL

Gorden Gibson of San Jose CA had a problem in using ZSHELL. He had to depress the BREAK key to get a program to terminate. Gorden also has LC and uses the built in command line I/O capabilities inherent in LC. Therein lies the rub. Our current release of ZSHELL does not behave exactly like LC does upon reaching an end of file. If you use LC's redirection, an end of file is passed to the program accepting input. C programs normally are written to take action on this end of file condition. ZSHELL was designed primarily for use with programs that wouldn't understand an end of file from the keyboard.

Careful reading of the ZSHELL documentation reflects three forms of treatment upon reaching an end of file. These are "<filespec", "<@filespec", and "<#filespec". All three forms cause standard input to be redirected according to your device or file specification but cause different things to occur if an end-of-file is reached. If the '<' symbol is entered immediately followed by the devicespec or filespec, ZSHELL will automatically disengage the redirected input and restore the original *KI handling. Thus the standard input is once again retrieved from the original *KI device."

That's why he had to depress <BREAK> to terminate the CLONE program he had written using ZSHELL's redirection capability. If you read further in the ZSHELL manual, you will see that if an input redirection specification of "<#" is followed by the spec, then a BREAK character will be passed on to the program accepting input upon reaching end-of-file. You see, redirecting standard input via ZSHELL provides no way for a C program to detect the EOF. You must pass the BREAK via "<#" since LC supports a BREAK as end-of-file on the keyboard. The redirection capability of ZSHELL is a little more flexible since it has to deal with programs with unknown handling of inout and output.

You all may be pleased to know that I have taken this situation to heart. The release of PRO-ZSHELL has been rewritten to reverse the sense of the pound sign input redirection appendage. PRO-ZSHELL will default to passing the BREAK upon reaching end-of-file while the "<#" syntax will indicate that control should be returned to the keyboard device upon EOF.

David B. Lamkins of Marlboro, MA writes, "I've noticed that piping always transfers control to *KI when the input file is exhausted. This works out well if the pipe is feeding a program (say LED) where you really need keyboard control. Consider, however, something like,

```
DIR (N) | FILTER1 | FILTER2 >MYFILE
```

To get my processed directory all the way to myfile, I have to hit BREAK twice; once to pass an EOF to filter1 and once for filter2. I'd like to suggest that a future revision of ZSHELL adopt a convention similar to that employed for input redirection, allowing use of "|#" and "|#+" to signify that EOF is to be passed through the pipe."

David made his point. The PRO-ZSHELL release supports the use of the pound sign to alter the effect of end-of-file and I attribute it to his suggestions. Another thing which PRO-ZSHELL has is an invocation option which allows you to establish a ZSHELL command line buffer in excess of the DOS limit of 79 characters. Actually, you can set up space for up to a 255-byte buffer. This can be quite useful when entering long piping commands or when stacking up multiple commands using the semicolon separator.

Now we have not forgotten the Model I/III ZSHELL users. Karl worked up a new release incorporating the features introduced into PRO-ZSHELL. It's available for the cost of an update - $5.

Programs which are written for a general DOS environment sometimes use coding techniques which are unworkable with a DOS that provides good device independence. For instance, LDOS 5.1 and TRSDOS 6.x support I/O redirection using ROUTE. ZSHELL, of course, goes many times beyond the simple ROUTE function by giving you the redirection control during the exexecution of a command. With ZSHELL, you can redirect the keyboard input so that it is fetched from a disk file instead of the physical keyboard. Now what happens with programs which constantly invoke the @KBD DOS call so that a BREAK or a PAUSE request can be ascertained. If the *KI device is routed to a disk file for those kinds of programs, the disk file would rapidly reach its end of file and be worthless for satisfying the "real" keyboard input. Such was the case exposed by Frederick Miller of Parsippany NJ. He writes, "I had ordered ZSHELL specifically for use with PROFILE III+. Unfortunately, that software is apparently riddled with what your specifications call 'certain programming techniques .... which constantly call @KBD and look for X'01' for a break. Are you aware of any patches which have been developed for PRO-FILE III+ to eliminate this unfortunate approach? Anyway, ZSHELL looks like a very nice product."

Now I am not aware of any patches to PROFILE. If any of my readers are aware of PROFILE patches to defeat it's scanning of @KBD for BREAK detection, please let me know.

CONTRIBUTIONS

Pete Carr or Port Orange FL has supplied us with three utilities. They are described as:

PCE - An LDOS 5.1 command line editor. This one is more powerful than the one that appeared in the L.S.I. Journal a couple of issues back. It takes a minimum of high memory (about 100 bytes) by using an overlay method. PCE has true insert and delete command line editing. For the Model 3 and 4 (3 mode).

INVBEL3 - An LDOS 5.1 inverse video and ASCII BEL filter. Gives LDOS 5.1 users inverse video capability on a Model 4 in Model 3 mode. Also gives a short sound prompt when a PRINT CHR$(7) is issued.

CLICK3 - An LDOS 5.1 key click filter. For the Model 4 (in 3 mode).

Pete supplied me with BASIC programs which will create the above mentioned utilities. He also supplied documentation files for each. This is all good stuff; however, if I would print all of the information, it would create a rather lengthy issue of NOTES. I also do not think it prudent for you to have to key in all of the material. Each of the programs and their respective documentation can be found on DISK NOTES 4. The BASIC programs are saved in ASCII so that you can read them into any text editor or BASIC to "play" with the code. In order to give you some flavor of Pete's work, I am printing the INVBEL3 filter and documentation here.

```
5 '******* INVBEL3/FLT  CREATE  PROGRAM  *********
6 '*******        BY  PETE CARR           *********
10 CLS
20 DEFINT A-Z : C=31376
30 PRINT @10,"CREATING INVBEL3/FLT ON LDOS 5.1 - DRIVE :0"
40 OPEN"R",1,"INVBEL3/FLT:0",1
50 FIELD 1, 1 AS D$
60 I=I+1
70 READ A
80 IF A=999 THEN GOTO 130
90 LSET D$=CHR$(A)
100 PUT 1,I
110 S=S+A
120 GOTO 60
130 CLOSE
140 IF S<>C THEN PRINT @138,"ERROR - CHECK DATA STATEMENTS":END
150 PRINT @138,"INVBEL3/FLT NOW READY."
160 PRINT @202,"NOW TYPE  FILTER *DO INVBEL3 <PRESS ENTER>."
170 CMD"S"
500 DATA 5,6,73,78,86,66,69,76,1,2,0,82,213,26,245,58,16,66,203
510 DATA 119,32,5,62,68,50,228,82,33,17,68,17,138,66,34,44,82
520 DATA 34,56,82,237,83,81,82,241,221,225,221,110,1,221,102,2
530 DATA 34,98,83,42,0,0,34,207,82,1,151,0,175,237,66,34,0,0,35
540 DATA 221,117,1,221,116,2,235,33,205,82,237,176,33,86,82,205
550 DATA 103,68,195,45,64,205,0,0,195,48,64,10,73,110,118,101
560 DATA 114,115,101,32,86,105,100,101,111,32,38,32,66,101,108
570 DATA 108,32,70,105,108,116,101,114,32,45,32,86,101,114,115
580 DATA 105,111,110,32,51,46,48,10,70,111,114,32,77,111,100
590 DATA 101,108,32,52,47,52,112,32,40,105,110,32,51,32,109,111
600 DATA 100,101,41,32,97,110,100,32,76,68,79,83,32,53,46,49
610 DATA 120,10,67,111,112,121,114,105,103,104,116,32,40,99,41
620 DATA 32,49,57,56,52,32,98,121,32,80,101,116,101,32,67,97
630 DATA 114,114,7,0,10,13,24,10,0,0,7,73,78,86,66,69,76,51,56
640 DATA 29,245,121,254,7,32,25,243,17,255,138,66,62,1,211,144
650 DATA 16,254,66,60,211,144,16,254,29,32,240,251,24,104,24
660 DATA 103,121,254,18,32,28,62,1,102,0,83,1,50,36,64,58,173
670 DATA 64,246,8,50,173,64,211,132,58,35,64,50,255,77,62,95,50
680 DATA 35,64,24,69,121,254,19,32,23,62,0,50,36,64,58,173,64
690 DATA 238,8,50,173,64,211,132,58,255,77,50,35,64,24,41,58,36
700 DATA 64,254,1,32,34,121,254,16,40,15,254,17,40,18,58,254,77
710 DATA 254,1,32,18,203,249,24,14,62,1,50,254,77,24,7,62,0,50
720 DATA 254,77,24,0,241,195,0,0,2,2,0,82,999
```

INVBEL3/FLT 3.0 - INVERSE VIDEO & ASCII BEL FILTER

For LDOS 5.13 / 5.14 TRS-80 Model 4/4P (in Model 3 mode) By Pete Carr

INVBEL3/FLT is a display (*DO) filter written for the TRS-80 Model 4/4P user who would like to use that machine's inverse video (black on white characters) and sound feature while using LDOS 5.13 / 5.14 in the Model 3 mode. It is very easy to activate INVBEL3. From LDOS Ready type:

FILTER *DO INVBEL3 <press enter>

INVBEL3 will locate itself into high memory and adjust the system's high memory pointer, accordingly. Once INVBEL3 is activated, it may be sysgened with your other filters such as PCE, CLICK3, KSM, KI, etc.

How To Use INVBEL3/FLT

There are two features of INVBEL3. One is inverse video, the other is the ASCII BEL (beep prompt). The inverse video feature works much the same way as it does in the TRSDOS 6.x model 4 mode. Like the model 4 mode, you must use character codes to control the inverse video. The following four codes control inverse video and ASCII BEL:

CHR$(18) = switch in inverse video mode.
CHR$(19) = switch out inverse video mode.
CHR$(16) = start display of inverse video.
CHR$(17) = stop display of inverse video.
CHR$(7) = sound ASCII BEL (beep prompt).

Codes 18 and 19 are the master switches and are used to tell the system that you will be using the inverse video mode. They are NOT used to display the actual inverse video text. Codes 16 and 17 are used to actually control what text will be displayed in inverse video or regular video. It is very easy to use inverse video with INVBEL3 as the following BASIC example demonstrates:

```
10 PRINT CHR$(18) ' Switch in inverse video mode.
20 PRINT CHR$(16);" THIS IS INVERSE VIDEO ";CHR$(17)
30 PRINT " THIS IS REGULAR VIDEO"
40 FOR X=1 TO 1500 : NEXT X
50 PRINT CHR$(19) ' Switch out inverse video mode.
60 CLS : END
```

It is advisable to always issue a PRINT CHR$(19) when finished with the program that you use inverse video with. If this is not done, certain graphic and control characters will not be displayed normally. If by chance the system gets put in the CHR$(18) mode mistakenly (this might happen when listing certain CMD files to the video screen and a CHR$(18) passes through), just issue a PRINT CHR$(19) and it will go back to normal. This is no real problem, and it hurts nothing, but, it has happened to me a couple of times and you should know what is happening.

Model 4/4P Inverse Video Idiosyncrasies

As was mentioned earlier, using inverse video with INVBEL3 (in Model 3 mode) works almost identically to using inverse video with TRSDOS 6.x in the Model 4 mode. It also has its same idiosyncrasies which are described next:

1) Cursor character in inverse video mode.

When the INVBEL3 filter detects a CHR$(18) it automatically changes the normal block cursor character to a thin underline CHR$(95). This is done because the normal block cursor character is displayed as an undesirable inverse 0 while in the inverse video mode. This is NOT something that is attributable to INVBEL3/FLT, but to the the Model 4/4P hardware. The same phenomenon occurs with TRSDOS 6.x in the Model 4 mode. So, instead of having the strange looking inverse 0 as the cursor, I decided to switch-in the underline cursor while the system was in the inverse video mode. The underline cursor will look normal, either in the inverse video mode or regular video mode. When the system is switched back out of the inverse video mode, your previous cursor character will be restored.

2) Graphic characters in inverse video mode.

While in the inverse video mode certain graphic characters will not be usable in the normal fashion. This also is attributed to the Model 4/4P hardware and NOT the INVBEL3 filter. TRSDOS 6.x works the same way in the Model 4 mode.

ASCII BEL (beep prompt)

Another feature of INVBEL3 is the ASCII BEL. This is useful anytime you want to sound a short beep to get the user's attention . To activate the ASCII BEL feature just PRINT CHR$(7) and a short beep will sound. For example:

```
10 PRINT"THERE IS NO MENU SELECTION 8."
20 PRINT CHR$(7); ' this will sound a beep.
30 PRINT"PLEASE TRY AGAIN."
```

Notes: INVBEL3 filter may not be compatable with some other filters or utilities that try to use the Model 4/4P ports while in the Model 3 mode. This is attributable to the fact that there is no standard memory address to save the systems port image byte. For those that need to know, INVBEL3 saves its port image byte at address X'40AD'. INVBEL3 works fine with 99% of the filters I have tried such as PCE, CLICK3, L.S.I. filter disks #1-#2, PowerSoft "TOOLS", etc.

Important: Before using INVBEL3/FLT you must first install 3 small patches to LDOS 5.13 / 5.14. After the patches are installed your LDOS 5.13 / 5.14 will run as before, except INVBEL3/FLT will be compatable with it.

```
PATCH SYS0/SYS.RSOLTOFF (D05,23=38) Use with LDOS 5.14 ONLY
PATCH SYS0/SYS.RSOLTOFF (D05,22=38) Use with LDOS 5.13 ONLY
PATCH SYS0/SYS.RSOLTOFF (D08,DD=D2) Use with 5.14 or 5.13
PATCH SYS0/SYS.RSOLTOFF (X'40AD'=00) Use with 5.14 or 5.13
```

Ron Higgs and his two boys Julian and Austin of Anderson SC have supplied my NOTES readers with a number of Interesting items. Ron writes, "As promised I attach (as if you can do anything else on a disk!) a number of little programs you may wish to look at and consider for your next issue of NOTES FROM MISOSYS. Any you do not wish to use in this issue or the next will become part of a package my boys (Julian and Austin) and I are preparing for possible sale, so let me know those of no interest to you.

First let me say that we have learned so much from you from your books, publications and progammming that I welcome the chance to thank you for myself and my boys, both of whom are professional computer types and greatly respect your knowledge and skills.

Now, on to the disk. You will see that all /ASM programs are backed-up by /BAK files. I have also assembled them as /CMD or /FLT so you can run them without taking the time to assemble!. [The] disk is a TRSDOS 6.2 Format.

SHOW/CMD - A utility (might be similar to MISOSYS MEMDIR, so you better then not publish!) to display names and start addresses of modules loaded above HIGH$. Also displays LOW$ and HIGH$ and size of user and protected memory. Invoke SHOW from TRSDOS.

QKEY/CMD and FKY/FLT (Go togther). QKEY is a shell for FKY/FLT. Invoking QKEY from TRSDOS will look for FKY/FLT and if not loaded will prompt for <I>nstall?. Invoking at any time will allow changing Shifted F1,F2,F3 keys as commands and will allow removal of filter from high memory and reseting of devices. QuickKEY is documented in the /ASM header quite fully so I will say no more (simply invoke it and follow the prompts!). We talked of this on the phone last week and I tossed it together after the discussion. I like KSM/FLT but if you are frequently changing DOS commands in backup and format etc. work, editing the KSM file, reseting the filter etc. can be a chore. QKEY carries the key commands in FKY/FLT which it loads and edits in High Memory.

COMMUNICATIONS - The Model 4 makes a nice host to Remote Terminals such as the DEC VT series and to most all micros using comm programs but you must add a line feed for many of them and most that dont need the extra line feed will automatically discard one if it finds it. I use the DEC from the office via modem to access the MODEL 4 at home. LFEED/FLT invoked as a filter (See /ASM header) will do the job. We always set it. Setting up a MODEL 4 as a host requires:

```
SET *CL COM/DVR
LINK *KI *CL
LINK *DO *CL.
```

You cannot do these LINK's from a /JCL file. A /JCL file invoking SET *CL COM/DVR, then JLINK (sw=on) followed by SET *LF LFEED/FLT and FILTER *CL *LF will set you up as a host! JLINK (sw=on) and JLINK (sw=off) allows quick setting and reseting of *KI, *DO, *CL. JLINK/CMD is one is one of Julian's programs. When the calling Remote (This assumes an auto-receive modem at the Model 4) types BYE the BYE/CMD then sends a modem break to the modem which breaks the connection and resets it to wait for the next call! Just like the Biggies!!!

Enjoy looking through these even if you decide not to publish any of them. You should recognize much of your own style and technique in most of them (and perhaps some of my own long cuts - these are the opposite of efficient short cuts!!!).

P.S. Julian has a nice program called DIAL/CMD. It is invoked from within COMM/CMD using the DOS COMMAND. It holds 10 preset auto-dial numbers which then dial up the modem at a keytroke from the menu. Very handy for dialing the long numbers used with TEL-CALL etc. He says he will be happy to send it to you if you are interested."

Now Ron and his boys did a pretty good job. The source files are well documented via the statement comments. For now, I'd like to publish the LFEED filter here in NOTES since we get lots of queries on how to write filters [the queries always come from someone who has not purchased a copy of THE PROGRAMMER'S GUIDE TO LDOS/TRSDOS VERSION 6]. We will probably get around to putting another of the Higgs' programs into the next issue of NOTES; however, if you'd just like to get your hands onto one of them a lot sooner, why don't you get in touch with Ron. He'll probably have a couple collections of his work available for sale. You can contact them at White Oak Software, 1103 White Oak Drive, Anderson S.C. 803-226-9519.

Ron and company based the line feed filter on one of the filters printed in The GUIDE. Here is just one more example then of a TRSDOS 6.x filter. If you program in assembler, they're quite easy to work out (p.s. the blank lines that Ron had inserted into the code for extra readability have been removed in an effort to conserve space when printed in NOTES. Also, LFEED/ASM and /FLT are both on DISK NOTES 4).

```
;LFEED/FLT - Add a Line Feed to A Carriage Return by Ron Higgs - 04/28/84
; Structured from SLASH0/FLT, page A-192, LDOS/TRSDOS VERSION 6
; Manual by Roy Soltoff.....
      COM '<White Oak Software (c) Ver.04/28/84>'
;     Released to Public Domain by Ron Higgs 09/26/84
;     All Rights Reserved.....
; This filter will provide a line feed on a carriage return to the
; RS232 port and is needed when a DEC VT131 is used as a Remote to
; the Model 4 in a Host Mode - Link *DO *CL, Link *KI *CL.
;     To Filter Issue: SET *LF TO LFEED/FLT
;                      FILTER *CL USING *LF
```

```
; Can test by: SET *LF TO LFEED/FLT.... FILTER *DO USING *LF
; ************ NOTE: ASSEMBLE TO LFEED/FLT **************
LF      EQU     10
CR      EQU     13
@CHNIO  EQU     20
@HIGH$  EQU     100
@DSPLY  EQU     10
@FLAGS  EQU     101
@LOGOT  EQU     12
        ORG     3000H           ;Put above system
BEGIN   PUSH    DE              ;
        POP     IX              ;Get DCB
        LD      (MODDCB),DE     ;Stuff DCB pointer
        LD      HL,HEADER$      ;Point to Header
        LD      A,@DSPLY        ;Display Header
        RST     28H             ;Do it to it !!
;
; Check if entry was a SET command
;
        LD      A,@FLAGS        ;Get flags pointer
        RST     28H             ;
        BIT     3,(IY+'C'-'A')  ;System request?
        JP      Z,VIASET        ;If not display error
;
; Instal new HIGH$ and relocate filter
;
        LD      HL,0            ;Get info request
        LD      B,L             ;Deal with HIGH$ only
        LD      A,@HIGH$
        RST     28H             ;Do it !
        JR      NZ,NOMEM        ;Error if no mem available
        LD      (OLDHI),HL      ;OK, put in filter header
;
; Relocate the internal references in the driver
;
        LD      IY,RELTAB       ;Point to relocation table
        LD      DE,MODEND       ;Last byte of Filter
        OR      A               ;Clear carry flag
        SBC     HL,DE           ;Subtract MODEND from old HIGH$
        LD      B,H             ;Put HL in BC
        LD      C,L             ;Ditto
RLOOP   LD      L,(IY)          ;Get address to change
        LD      H,(IY+1)        ;Ditto
        LD      A,H             ;
        OR      L               ;
        JR      Z,RXEND         ;
        LD      E,(HL)          ;Pick-up address
        INC     HL
        LD      D,(HL)          ;Get next byte
        EX      DE,HL           ;Put in HL
        ADD     HL,BC           ;Offset it
        EX      DE,HL           ;
        LD      (HL),D          ;And put it back
        DEC     HL
        LD      (HL),E          ;Second byte
        INC     IY
        INC     IY              ;'Cos two byte address
        JR      RLOOP           ;Loop till done
;
; Move driver into high memory
;
```

```
RXEND   LD      DE,(OLDHI)      ;Destination
        LD      HL,MODEND       ;Last byte of module
        LD      BC,LENGTH       ;Length of filter
        LDDR                    ;Move it (reverse fashion)
        EX      DE,HL           ;Move new HIGH$ into HL
        LD      A,@HIGH$        ;Set new HIGH$ into system
        RST     28H             ;Do it to it !!
        INC     HL              ;Bump to filter entry
        LD      (IX+0),40H.OR.7 ;Stuff TYPE byte
        LD      (IX+1),L
        LD      (IX+2),H        ;Install address into DCB
        LD      HL,DONE$        ;Point to Msg.
        LD      A,@DSPLY
        RST     28H
        LD      HL,0            ;Successful....
        RET
VIASET  LD      HL,VIASET$      ;Point to SET error msg.
        DB      0DDH            ;Dont know why this works, but it does
NOMEM   LD      HL,NOMEM$
        LD      A,@LOGOT
        RST     28H
        LD      HL,-1
        RET
HEADER$ DB      LF,'LFEED/FLT - White Oak Software (c) Ver.04.28.84',CR
DONE$   DB      LF,'Filter Installed......',CR
NOMEM$  DB      LF,'High memory not available!',CR
VIASET$         DB      LF,'Must install via SET command!',CR
;
; The Filter itself.......
;
LFEED   JR      START
OLDHI   DW      $-$             ;HIGH$ before filter instalation
        DB      MODDCB-LFEED-5  ;
        DB      'LFEED'
MODDCB  DW      $-$             ;Gets loaded with DCB pointer
        DW      0               ;Reserved for TRSDOS
START   JR      NZ,OUTP1        ;Go if not @PUT
        LD      A,C             ;Else put char. in A
        CP      CR              ;Is it a carriage return?
        JR      Z,OUTCF         ;If so, go to add a LF
OUTP1   PUSH    IX              ;Save current pointer
        PUSH    BC              ;Save in case affected downstream
        LD      IX,(MODDCB)     ;Pick-up this modules DCB
RX01    EQU     $-2
        LD      A,@CHNIO        ;Chain to next
        RST     28H             ;Do it !
        POP     BC
        POP     IX
        RET
;
; Do the adding
;
OUTCF   CALL    OUTP1           ;Put the CR
RX02    EQU     $-2
        LD      C,LF            ;The Line Feed!
        JR      Z,OUTP1         ;Unless error
MODEND  RET
LENGTH  EQU     $-LFEED         ;Calculate it!
RELTAB  DW      RX01,RX02,0
        END     BEGIN
```

Notes from MISOSYS

Karl Hessinger of College Park MD uses both LDOS 5.1 and MAXDOS 6.2 (a version of TRSDOS 6.2 which runs on the Lobo Systems MAX-80) on his MAX-80. Occasionally he would goof and invoke an LDOS 5 CMD file while running under MAXDOS 6.2. Karl therefore came up with the following program which attaches itself to the system loader and flags an alert if a CMD program has a transfer address higher than 51FFH. Thus, if you invoke an LDOS 5.1 program while under MAXDOS 6.2 (or TRSDOS 6.2 - NOFIVE runs under it also), an alert message will appear and you will be provided an opportunity to cancel the invocation. NOFIVE/CMD is on DISK NOTES 4. I am printing it here as a direct patch. If you key in the patch and save it as NOFIVE/FIX, the following procedure will create the CMD file.

```
DUMP NOFIVE/CMD (E=X'3216')
PATCH NOFIVE NOFIVE (O=N)
```

The "(O=N)" is important as it tells TRSDOS 6.2's PATCH utility not to worry about checking the Find lines. Here's the direct patch for NOFIVE:

```
. NOFIVE/FIX
D00,00=05 06 4E 4F 46 49 56 45 1F 37 28 43 29 20 43 6F
D00,10=70 79 72 69 67 68 74 20 31 39 38 34 20 4B 61 72
D00,20=6C 20 41 2E 20 48 65 73 73 69 6E 67 65 72 20 2D
D00,30=20 4D 69 63 72 6F 43 6F 6E 73 75 6C 74 61 6E 74
D00,40=73 01 02 00 26 21 DA 26 3E 0A EF 11 55 27 3E 53
D00,50=EF CA 9C 26 3E 65 EF FD 66 1A 2E 9A 5E 23 56 21
D00,60=0D 00 19 22 AB 27 2B 22 A8 27 11 9C 27 01 03 00
D00,70=ED B0 21 00 00 45 3E 64 EF 20 62 22 52 27 FD 21
D00,80=3E 27 11 E1 27 B7 ED 52 44 4D FD 6E 00 FD 66 01
D00,90=7C B5 28 0F 5E 23 56 EB 09 EB 72 2B 73 FD 23 FD
D00,A0=23 18 E7 ED 5B 52 27 21 E1 27 01 92 00 ED B8 EB
D00,B0=3E 64 EF 23 22 A5 27 CD A0 27 3E 65 EF FD 7E 1C
D00,C0=FD 6E 1D FD 66 1E 32 AF 27 22 B0 27 21 A0 27 FD
D00,D0=36 1C C3 FD 75 1D FD 74 1E 21 00 00 C9 21 BD 26
D00,E0=DD 21 A6 26 3E 0C EF 21 FF FF C9 4E 4F 46 49 56
D00,F0=45 20 61 6C 72 65 61 64 79 20 61 63 74 69 76 65
D01,00=21 0D 48 69 67 68 20 6D 65 6D 6F 72 79 20 69 73
D01,10=20 4E 4F 54 20 61 76 61 69 6C 61 62 6C 65 0D 0A
D01,20=4E 4F 46 49 56 45 2F 43 4D 44 20 2D 20 57 72 6F
D01,30=6E 67 20 44 4F 53 20 41 6C 65 72 74 20 2D 20 56
D01,40=65 72 73 69 6F 01 E4 00 27 6E 20 31 2E 30 61 0A
D01,50=43 6F 70 79 72 69 67 68 74 20 31 39 38 34 20 2D
D01,60=20 4B 61 72 6C 20 41 2E 20 48 65 73 73 69 6E 67
D01,70=65 72 20 2D 20 4D 69 63 72 6F 43 6F 6E 73 75 6C
D01,80=74 61 6E 74 73 0A 0D 82 27 77 27 66 27 82 26 85
D01,90=26 88 26 70 26 73 26 00 00 18 0A 00 00 03 4E 4F
D01,A0=35 00 00 00 00 E5 21 04 00 39 7E 23 66 6F 22 74
D01,B0=27 01 00 52 B7 ED 42 E1 38 2B D5 E5 11 00 00 21
D01,C0=CC 27 3E 63 EF 06 29 3E 68 EF 21 B2 27 3E 0A EF
D01,D0=3E 01 EF F6 20 FE 79 28 0A FE 6E 20 E8 21 FF FF
D01,E0=3E 16 EF E1 D1 00 00 00 C9 F5 E5 3E CD 21 00 00
D01,F0=32 00 00 22 00 00 E1 F1 C9 C9 C9 57 61 72 6E 69
D02,00=6E 67 20 3A 20 45 6E 74 72 79 20 70 6F 69 6E 74
D02,10=20 3D 20 78 27 30 30 30 30 27 0A 45 78 65 63 75
D02,20=74 65 20 28 59 2F 4E 29 20 3F 03 02 02 00 26
```

Incidentally, if you are looking for a similar program but one which would be called NOSIX (you figure it out), contact Karl at MicroConsultants, 7509 Wellesley, College Park, MD 20740-3037 (301-474-8486).

MISOSYS, Inc
P.O. Box 239
Sterling, VA 22170-0239
703-450-4181

Contents: Printed Matter