THE LDOS QUARTERLY April 1, 1982 Volume 1, Number 4

LOGICAL SYSTEMS INC.

Table of Contents

V I E W   F R O M   T H E   B O T T O M   F L O O R

This has been a very busy quarter for our staff, as LDOS has become one of the leading operating systems for the TRS-80 and demand for our products has risen dramatically. To our many thousands of users, THANK YOU. We will make every effort to make the LDOS product line the best designed, written, documented and supported line of software for the TRS-80.

In the last newsletter I stated that the Extended Support Agreement (ESA) would be $50 per year. After that newsletter went to press it was decided not to include 800 line support in the ESA and to drop the price of this support to just $25. The ESA allows the subscriber to receive updates within the same version for just $5, as often as he wishes, receive four issues of this publication and to be a member of the LDOS bulletin board on MicroNET (if you are a COMPUSERVE subscriber).

Due to large increases in sales volume and some changes in warranty and support services, we have been able to reduce the suggested retail price of LDOS to $129.

The current version of LDOS is 5.1.2 for both the MODEL I & III. The features are now the same for both machines, other than some slight differences due to hardware. This new version is available now. You must send in your MASTER disks with a check for the correct amount (see the customer service section of your manual) to receive this update. Model I users should note that we require BOTH of their master disks to perform this update.

In the past we have issued patches to LDOS to correct minor problems or to add minor features. In the future we will be offering very few corrections or enhancements in printed form. We feel that it is better to maintain our system through the use of our update policy. We feel our update policy is a fair and equitable way of dealing with the ongoing cost of support. In this way we are able to deal with support in a straight forward manner, because we are sure of the state of the version in use by the user. This does NOT mean that we will not be publishing patches from time to time. We will publish patches for such things as optional enhancements or very minor corrections that are not considered critical in nature.

We now have two new LDOS packages. They are both designed around a new product we call "smal-LDOS" (pronounced 'small-dos'). This product is a minimal runtime DOS based on LDOS. It will be marketed to OEMs and software publishers and is available for the Model I and III. The OEM version comes with a manual in a 5-1/2 by 8-1/2 format, while the software distribution version contains only a master smal-LDOS disk and NO documentation. Both of these new products will be available only when they are a part of a software or hardware package. However, any registered LDOS owner can order (direct from LSI) one copy of the smal-LDOS (disk only) product for $25. This may be of value for those who wish to have more storage on a "system" disk, as the libraries (SYS 6 & 7) are much smaller on this product, or to those who just want a optimised minimal LDOS to use when running applications. This product is available in Model I 35 track single or double density and in Model III 40 track double density (specify when ordering).

LSI will provide LDOS on several types of media to meet certain needs. The charge is $10 if we provide the diskette and $5 if the user provides the disk. These will of course be provided to REGISTERED LDOS owners only. The system files on these disks are located in such a fashion that the overall performance of the system is optimised. The following disk types are available:

FOR THE MODEL I TRS-80:

- 35 Track, Single sided, Single density (two disk set)
- 35 Track, Single sided, Double density
- 80 Track, Single sided, Single density
- 80 Track, Single sided, Double density
- 77 Track, Single sided, Double density, 8-inch

FOR THE MODEL III TRS-80:

- 40 Track, Single sided, Double density
- 80 Track, Single sided, Double density

NOTE: These disks are available in addition to the standard media supplied and not in lieu of the standard media. You must specify exactly what type of disk you want (from the above list) when ordering, and include your LDOS serial number. We do not offer double sided system disks at this time. We are evaluating the reliability of double sided drives and when they prove to work reliably we will offer system disks in the two sided format.

The MicroNET bulletin board is a very good place to receive timely information and assistance when using LDOS. Users of this board are hereby warned that we will NOT tolerate the misuse of this service. The use of profanity or the leaving of slanderous, libelous or unsubstantiated critical remarks on the board will be cause for canceling the membership of the user. So please be honest, fair and serious when using this service. Also if you have messages to you that are not of general interest to LDOS users, please delete them after you have read them. Also do NOT ask questions about clerical matters, such as "has my update been shipped ?", "I didn't receive my last quarterly" or change of address notices and the like. The bulletin board is handled by our TECHNICAL staff not our clerical staff. Any clerical services request must be in writing to our customer service department.

Our telephone support group no longer has any portion of the LDOS source code available to them in their area of our building. So requests for any information that will require checking the source code MUST be in writing to customer service, ATTENTION: SYSTEMS GROUP.

Please be careful when ordering products that are supposed to be LDOS compatible. If we don't definitely state that we support a product (software or hardware) then do not look to us for support. When a vendor states that his product is LDOS compatible, we take that at face value and assume he is telling the truth, but we will not be responsible for false information given to us by vendors. Check with us before you buy. We will be happy to tell you our position on the product that you are considering. Don't complain to us after you have purchased a product and find you have problems.

With regard to the Model III, we are aware of "Model III type" computers from several sources. LSI officially supports ONE type of Model III, and that of course is the official standard machine produced exclusively by Radio Shack.

LDOS seems to work with most of the other machines..... BUT ..... We do NOT have these machines in house and therefore do not support them. Many thousands of users (with standard Radio Shack hardware) use LDOS with little or no problems. To call and say LDOS won't format disks in your Brand "X" Model III will probably meet with a ....."I'm sorry we can't help you........", response. We are willing to support most quality products that wish to be supported by LDOS, but only with the full cooperation of the manufacturer and with hardware provided by that manufacturer.

Now for some bad news.... Our "C" compiler, known as "LC", has been put on HOLD for the moment. It was very close to release when unexpected "personal demands" on the author made it impossible to proceed. We hope to be able to revive the "LC" project in the near future.... but no promises. So, for the time being the LDOS "C" compiler has been withdrawn as an offering of the LDOS product line. Our apologies to those who have been awaiting the release of this product.

Any registered LDOS user may order an additional copy of the LDOS manual for $59 or may order a replacement for a damaged or older version of our manual for $29. If ordering a replacement, the old manual must be sent in with the order! Any manual order includes a new binder and tab set. Please add $5 for shipping and handling. Phone orders will not be accepted for "replacement" manuals. All manual orders must be placed directly with LSI. Manuals are NOT available from dealers or distributors.

The LX80 interface for the Model I is still available direct from LOBO DRIVES for a $449.00 This price includes 32K of RAM, and support for both 5" and 8" drives in single and double density. The dual channel RS232 option is an additional $50.00. We have found the LX80 to be a very reliable piece of hardware, and use them daily at LSI.

If you are not on the ESA or covered by the LDOS warranty, you may still update your Master disk to the latest release within the version you purchased for $10. This is a lifetime right for all registered LDOS owners, subject only to eventual cost increases to cover our costs of providing this service.

This newsletter is no longer included as part of the standard support for LDOS but is available to those on the Extended Service Agreement, or to those covered on our "OLD" one year warranty. For those who will not have access to the newsletter, we are installing a LDOS HOTLINE. This will be a three minute tape recording stating the latest LDOS news, patches, new products, etc. It will be available 24 hours a day, so that it may be called at low rate times, at a very low cost to the user. This service will be available in early July of this year. The number will be in the next newsletter and will be available through directory assistance for area code 414.

Radio Shack has announced a double density controller for the Model I. As this product becomes available, we have every intention of providing official support for it. Please give us a little time after it comes out so we may check it out completely. We will also support the newVisicalc and the new Super Scriptsit, when time permits.

LED - THE LDOS EDITOR

LED is a screen oriented text editor that is designed to work with the LDOS operating system. Although very versatile, the LED commands are easy to learn. Those familiar with the LDOS LSCRIPT version of Scripsit will notice a similarity in the command key layout. This is the LED command menu, and can be displayed at the bottom of the video screen while using LED.

```
INDNT  FIND CHANGE HEX  UNMRK DNP  UPP  ALL  AGN       NAME EXIT
 =1=   =2=   =3=   =4=   =5=  =6=  =7=  =8=  =9=       =:=  =-=
INSRT  LIN   DEL   WRD   BLK  END  TOP  SPA  TAB       MENU SAVE
                              {TEST/TXT:0-R}    ( 0):X'00'|35751
```

The display contains the name of the file currently being edited, the current cursor column, the hex value of the character under the cursor, and the available memory in the text buffer.

Since LED uses the LDOS keyboard driver, type-ahead and all keyboard filters are available for use with LED. Also, all 128 ASCII characters are available directly from the keyboard.

Cursor positioning is done in the normal manner, with the 4 arrow keys controlling the cursor motion. The <CLEAR><ARROW> keys will move to the top or bottom of the text, or to the left or right end of a line. The <SHIFT><LEFT> and <SHIFT><RIGHT> arrows also perform movement to the ends of a line unless tabs are set. Then, they position either to the next tab location or back to the previous one. There are 4 different cursor characters, depending on the mode you are in (typeover, insert, insert line, or delete).The UPP and DNP commands are used to move the display buffer up or down a full page at a time. If the file to be edited has a /KSM extension, LED will automatically display the alphabetic letter before each line assigned to that letter.

LED can be used on many different types of files. The FIND and CHANGE commands make it handy for doing global changes in BASIC programs. The AGN and ALL commands let you find or change things one at a time or all at once.

A very useful feature is the HEX mode. This mode is available either when overtyping or when inserting. It allows you to input characters as two hexadecimal digits over the entire X'00' to X'FF' range, making possible direct editing or inputting of graphics characters.

Certain parameters may be specified when first entering LED. TABS will cause any X'09' tab character to be expanded. Tabs normally appear as a small graphics block. SAVE="filespec" will save a file under a different name than was used to load the file. XLATE=X'fftt' will perform a character translation when loading and saving a file. Two other parameters, END=X'00' and WP deal with word processor files that use an X'00' to mark the end of a file.

One very nice feature is an automatic SAVE prompt. If you request an exit back to LDOS Ready, and have modified the text buffer, LED will automatically ask you if you want to save the file. If no modifications have been made, an immediate exit to LDOS is done without the prompt.

LED is available for $40.00 from any of the LDOS distributors.

M O D E L   I   5 . 1 . 2   U P D A T E

The following new files will be on your LDOSXTRA disk:

MOD1/EQU has been renamed to EQUATE1/EQU.

MOD1/DCT - A file used to set default values for floppy drives, normally used along with hard drives.

UTILITY PROGRAM CHANGES

BACKUP now allows you to cancel the QUERY parameter during a backup. Pressing <C> in response to the prompt will cause the backup to continue non-stop from the current file.

FORMAT has a new parameter, WAIT=. This parameter was added to make up for deficiencies in certain 80 track drives, and is NOT normally needed. Its purpose is to provide a delay between eachstepin to another track. It should only be used when ALL tracks above a certain point get locked out. The value for WAIT will probably be a number between 5000 and 50000, and may vary greatly from drive to drive. It is suggested that a value of 25000 be used initially, and then adjusted up or down as needed.

DRIVER/FILTER PROGRAM CHANGES

KI/DVR has been significantly changed. Two new parameters, DELAY= and RATE=, have been added. DELAY is the initial delay between pressing a key and the first repeat, and can be any value greater than 10. The default is 30, and provides about 3/4 of a second delay. RATE is the key repeat rate, and can be any value larger than 1. The default is 3, and provides a repeat rate of about 10 per second. All of the KI parameters (TYPE, JKL, DELAY, and RATE) can now be abbreviated to their first character. Also, two new bits have been assigned in the KFLAG$ memory storage location. Bit 6, if set to "1", sets the Extended Cursor Mode. Bit 5, if set to "1", sets the CAPS LOCK mode.

KSM/FLT has a new parameter, ENTER=. This will allow you to set any character to be used as an embedded <ENTER>. The default remains the semicolon. ENTER= may be followed by the ASCII value of the character, or may be entered as a character between quotes. For example, to set a colon as the new character, either ENTER=58 or ENTER=":" will work. This parameter may be abbreviated "E".

LBASIC PROGRAM CHANGES

LBASIC has a new parameter, EXT=ON/OFF. The default is ON. EXT=ON means that LBASIC will use a default extension of /BAS during LOAD, RUN, MERGE, and SAVE operations. File OPEN and KILL operations will never use the default extension.

The RUN"filespec",V command will now save any fielded variables used with random files.

M O D E L   I I I   5 . 1 . 2   U P D A T E   N E W S

LSI is proud to announce the release of LDOS-512 for the Model III. There have been many enhancements made in 5.1.2 over 5.1.0B. This section of the newsletter will detail some of these enhancements.

NEW FILES

The following new files will be on your LDOS master disk:

EQUATES3/EQU - For assembly language programmers. This is an equate file for use with an editor/assembler program, in the format used by the EDAS editor/assembler. It contains the labels from the memory map in the technical section.

LSCRIPT/FIX - An enhanced Scripsit patch which makes use of LDOS's KI/DVR program and other keyboard features.

VC/FIX - A patch file for Visicalc

MOD3/DCT - A file used to set default values for floppy drives, normally used along with hard drives.

LIBRARY COMMAND CHANGES

The DEVICE command will now show if VERIFY is on.

The DIR command has two new parameters, SORT and MOD. The normal directory display will be sorted in alphabetical order. SORT=NO will disable the sort. The MOD parameter will now show just those files with MOD flags.

The SYSTEM library command has 4 new parameters. They are:

SYSTEM (DATE=ON/OFF) Enables or disables the DATE prompt when booting.
SYSTEM (TIME=ON/OFF) Enables or disables the TIME prompt when booting.

SYSTEM (BSTEP=n) Sets the default bootstrap step rate used by the FORMAT utility.

SYSTEM (DRIVE=,CYL=n) Sets the default cylinder count used by the FORMAT utility for the specified drive.

UTILITY PROGRAM CHANGES

BACKUP now allows you to cancel the QUERY parameter during a backup. Pressing <C> in response to the prompt will cause the backup to continue non-stop from the current file.

FORMAT uses different defaults if the QUERY=N parameter is specified, or if the <ENTER> key is pressed in response to a prompt. All prompts, including NAME and PASSWORD, may be defaulted. There is also a new parameter, WAIT=. This parameter was added to make up for deficiencies in certain 80 track drives, and is NOT normally needed.

The LCOMM utility has had several changes and additions. The XLATE parameter has been changed to allow a send character to be translated to some other character. Likewise, a receive character may be translated to any other character. Also, the <7> key has now been assigned the following function:

DTD ... <CLR><7>

The (DTD) Dump To Disk is used to write the memory buffer used with FR to the disk. DTD may be turned on before or after a file has been received. If turned on before, the file will be written to disk as it is being received.

The menu display will be different, and will show which devices and functions are active, as well as the amount of available memory, and the functions currently active.

The DCC (Display Control Characters) function has been added, and will force a display of any character received that has a value less than an X'20' as a two digit hexadecimal number surrounded by braces. Also, a CLS (Clear Local Screen) function has been added, and will erase the contents of the screen without transmitting any character to the communications line.

DRIVER/FILTER PROGRAM CHANGES

KI/DVR has been significantly changed. All 128 ASCII characters are now available from the keyboard, and an "Extended Cursor Mode" has been added. Two new parameters, DELAY= and RATE=, have been added. DELAY is used to set the initial delay between pressing a key and the first repeat. RATE is used to set the key repeat rate. All of the KI parameters (TYPE, JKL, DELAY, and RATE) can now be abbreviated to their first character.

KSM/FLT has a new parameter, ENTER=. This will allow you to set any character to be used as an embedded <ENTER>.

MiniDOS/FLT has a new key available. The <P> key will allow you to send a hex character directly to the lineprinter.

PR/FLT has the new parameter SLINE. This may be used to establish either the Model I or Model III default for the initial line count.

LBASIC PROGRAM CHANGES

LBASIC has a new parameter, EXT=ON/OFF. If this parameter is specified as on, LBASIC will use a default extension of /BAS during LOAD, RUN, MERGE, and SAVE operations.

The RUN"filespec",V command will now save any fielded variables used with random files.

The CMD"N" will now renumber ERL= lines.

WHAT'S NEW ?

New from MIDWEST DATA SYSTEMS is a package called AUTO-WRITER. It is a data management package that allows you to use your word processor to create and maintain your data base. AUTO-WRITER consists of 5 programs. STATS gives the status of a data file, telling things such as the location and length of the fields, the format of the data base, and will even show errors or inconsistencLes. SELECT lets you construct new files from your data base, based on "Plain-English" truth statements. SORT sorts in ascending or descending order, and by user specified key. LETTERS and REPORT generate form letters and reports based on user design. AUTO-WRITER is available for $72.83.

As this newsletter goes to press, the finishing touches are being put on AUTO-WRITER +. Enhancements to the basic package will include a two level sort, a math pack, a new input editor, and more. AUTO WRITER + is available for $120.00, with a special update price available for current AUTO-WRITER owners. For more information, contact MIDWEST DATA SYSTEMS, 5624Girard Ave. South, Minneapolis, MN 55419 (612) 866-9022

From LYNN COMPUTER SERVICE is a product called LYNN'S A/R, an accounts receivable package. This package consists of 12 integrated programs. It is available for $49.95 + $2.00 S&H, from LYNN COMPUTER SERVICE, 6831 W. 157th, Tinley Park, IL 60477 (312) 429-1915. Documentation and sample printouts are available separately for $10.00.

SPEAK! is a program which "adds lips" to your TRS-80 Model III. It requires a cassette recorder and mic, and a small amplifier. Words or phrases are spoken and learned by the program for later access by other programs. For more information contact Bill Neville, PO Box 2581, Houston, TX 77001, or Lee Perryman, P0 Box 2972, Tampa, FL 33601.

From MISOSYS is a new product called SOLE. This utility will allow double density booting on the Model I, Radio Shack E.I. It is available for $25.00 from MISOSYS, PO Box 4848, Alexandria, VA 22303-0848 (703) 960-2998.

Also from MISOSYS is a program called CON80Z. This program converts assembler source files in Intel 8080 mnemonic code toZilog Z80 mnemonic code. The output is EDAS/EDTASM compatible. The price is $50.00 + $3.00 S&H.

There is a HELP utility and a quick reference card available from MISOSYS. The package sells for $25.00, and the reference card is available separately for $3.00.

See Tim Daneliuk's review of the JOHNSON ASSOCIATES software package in his column.

From SOFT SECTOR MARKETING comes THE COMPLETE IDIOT'S BOOKKEEPER. This is a bookkeeping package for personal or small business use. The package comes with programs to provide indexes, edit the data files, build separate monthly files, remove deleted entries, etc. This program was tested with both 5.0 and 5.1 LDOS, and is available for Model I and III. For more information, contact Soft Sector Marketing, 6250 Middlebelt, Garden City, MI 48135 (313) 425-4020.

The SNAPP EXTENDED BASIC packages should be available for the Model III at this time. These packages provide enhanced cross reference and renumbering, keyword and string cross reference, variable dump, program compress and uncompress, "college educated" garbage collector, and more. For information, contact SNAPP Inc., 3719 Mantell, Cincinatti, Ohio 45236 (800) 543-4628.

Available from GALACTIC SOFTWARE is a manual on LBASIC. This is a 50 page manual, and explains all of the disk Basic statements that are available with LBASIC. Although not a tutorial on Basic programming, examples of most statements are included. Also, all LBASIC extensions to normal disk Basic are explained. The manual sells for $9.95 plus $2.05 S&H, and is available from Galactic Software, Ltd, 11520 N. Port Washington Rd, Mequon, WI 53092 (414) 241-8030.

Also available from GALACTIC SOFTWARE is the MemDISK program. This program creates a simulated disk drive in memory. All disk I/O functions work, including backup and copy. The memory disk can even be switched to drive 0 with the SYSTEM command. It is available for $39.00.

Available from ALCOR SYSTEMS is a version of the PASCAL language. The manual includes a reference and a tutorial section. For further information, contact ALCOR SYSTEMS, 13534 Preston Rd., Suite 365, Dallas, TX 75240 (214) 226-4476.

As we went to press, we had many more software companies who had received copies of LDOS and were checking on full compatibility with LDOS. The next Quarterly should list all of these packages.

ITEMS OF GENERAL INTEREST

On the Model III, our USTOR$ pointer is at X'4DFE'. This points to an 8 byte storage area available to the user. TRSDOS has its user pointer at X'4CFE'. If you are using Radio Shack programs that use this area, you should change the address accordingly.

When using the 5.1.1 or later LCOMM Utility to download files, the DTD (Dump To Disk) function is turned OFF when the receive file is reset. If you are downloading more than one file and want the DTD to remain on all the time, be sure to do a DTD ON <CLR><7><CLR><:> before turning on each successive receive file.

New Compuserve users have requested instructions on getting to the LDOS board on MicroNET. Simply type in:

```
   R LDOS
```

from the OK prompt of Compuserve.

There has been a new RAM STORAGE ASSIGNMENT designated for both the Model I and III. It is DAY$, and is used in conjunction with DATE$ to store information about the current system date. The new assignments are as follows. The Model III descriptions will match those of the Model I.

Model I

```
DATE$=X'4044'-X'4046'
+0...Contains the two digit year
+1...Contains the day of the month
+2...Contains the month

DAY$=X'4047'-X'4048'
+0...Contains bits 0-7 of the day of the year
+1...
     bit 0........Contains bit 8 of the day of the year
     bits 1-3.....Contains the day of the week (Sunday=0)
     bits 4-6.....Reserved
     bit 7........Set to "1" if leap year
```

Model III

```
DATE$=X'421A'-X'421C'
DAY$=X'4417'-X'4418'
```

The SPOOL command for the 5.1.2 version has had two visible changes made. First, if KI/DVR is not established, the SPOOL command will abort with an appropriate error message. Secondly, the system will no longer allow a SYSGEN if the spooler is active.

"How can you switch between upper or lower case without making the operator press the <SHIFT><0>?" "How can you force keyboard input to be in either upper or lower case?" These questions from our users resulted in a change to the KI/DVR program for the 5.1.2 update. The location known as KFLAG$ (X'4423' or 17443 on the Mod I, X'429F' or 17055 on the Mod III) now has bit 5 designated as the CAPS LOCK bit. If the bit is set, you will be in the CAPS LOCK mode. Therefore, a BASIC program on the Model I can switch into upper case only by POKEing the KFLAG$ location with PEEK(&H4423) OR 32, and switch to upper/lower by POKEing the location with PEEK(&H4423) AND 223. Model III would use the same method with PEEK(&H429F). Using the PEEK command and the logical OR or AND assures that any other bits in the KFLAG$ location will remain untouched.

When your disk drive takes a nap . . . . and then resumes 15 to 30 seconds later, chances are it was bitten by the 300 RPM bug. During normal disk I/O, the only time the interrupts are disabled is when the system is actually going to transfer a 256 byte sector to or from the disk. This makes possible interrupt driven features such as type ahead, the spooler, and LCOMM. However, since the interrupt clock rate is evenly divisible into a disk rotation speed of 300 RPM, it also can provide those mysterious naps from time to time when an interrupt occurs just as the physical I/O is going to happen. The solution is simple; change the disk speed to 302 RPM. This will assure that the interrupt and disk rotation will be out of sync without degrading disk I/O performance.

When using the LSCRIPT patch to Scripsit, you can still duplicate most of the original Scripsit key controls by using the <CTRL><key> sequence (<CTRL> being <SHIFT><DOWN ARROW>). For instance, <CTRL><S> puts you in the insert mode, <CTRL><V> forces a page marker, etc.

The following procedure can be used to simulate turning on and off a link between the video and the printer from within an LBASIC program. At the LDOS Ready prompt, type in the commands ROUTE *DU (NIL) and then LINK *DO *DU. In the LBASIC program, use a CMD"ROUTE *DU *PR" to link the video to the printer, and a CMD"RESET *DU" to turn off the link. This procedure may be repeated as often as desired.

Once again, it's time to mention the data address mark. For those of you who need them, use the following patches to force LDOS to write the old DAM. These are for the Model I, Radio Shack E.I.,VERSION 5.1.2 ONLY!

```
.SYS0 patch
X'467C'=A9
.EOP

.PDUBL patch
X'5472'=A9
.EOP
```

TCHRON - Time and Date board Patch

```
. TCHRON/FIX 03/09/82, ROBERT J. NEWTON
. PATCH SYS0/SYS.SYSTEM USING TCHRON
. This patch is for the Model I TCHRON time & date board
. and is for LDOS Version 5.1.x ONLY! It has been supplied
. by an LDOS user and is unsupported by LSI.
. Note: Parts of this patch have been derived from the
. TTIMER fix listed in the last issue of the QUARTERLY
. and copyrighted by Roy Soltoff
.
. It is recommended that you convert this to a DIRECT
. patch using FED before applying it to SYS0. This is
. to ensure that no additional space is taken up by the fix.
X'45C1'=D1 45 ED 78 0D A4 CD A5 47 ED 78 0D 85 12 1B C9
X'45D1'=11 43 40 01 75 03 60 CD C3 45 10 FB
X'4E97'=21 44 40 E5 01 7C 0F CD B9 4E 2323 CD B9 4E 2B
X'4EA7'=06 03 CD B9 4E 21 B9 50 DB 78 CB 57 28 01 34 D1
X'4EB7'=18 21 ED 78 0D A0 07 57 07 07 82 57 ED 78 0D 82
X'4EC7'=77 C9 00 00 00 00 00 00 00 00 00 00 00 00 00 00
X'4ED7'=00 00 00
. End of patch
```

Updated TTIMER Patches

There were a couple of bugs that inadvertently slipped into two of the TTIMER patches in the last issue. The underlined bytes in the following patches indicate the areas of change. The "x" in the 5.1.1 patch indicates that the original "13" byte should be removed from the patch.

```
. TTIMER Model I Version 5.0.2 & 5.0.3 patch
. Copyright (C) 1981 by Roy Soltoff, All rights reserved
. PATCH SYS0/SYS.WOLVES
```

```
D04,65=D8 45 ED 78 0D CD A1 47 ED 78 0D E6 0F 85 12 1B
D04,75=C9 11 43 40 01 C5 03 CD C9 45 10 FB
D0D,C0=21 46 40 01 CA 01 CD AA 4E 06 03 CD AA 4E 01 CC
D0D,D0=0F CD AA 4E EB 13 DB CB E6 03 21 92 50 20 01 34
D0D,E0=18 34 ED 78 0D A0 07 57 07 07 82 57 ED 78 0D E6
D0D,F0=0F 82 77 2B C9
. end of patch
```

```
. TTIMER Model I Version 5.1.1 patch
. Copyright (C) 1981 by Roy Soltoff, All rights reserved
. PATCH SYS0/SYS.SYSTEM
D04,05=D2 45 ED 78 0D CD A6 47 ED 78 0D E6 0F 85 12 1B
D04,15=C9 11 43 40 01 C5 03 CD C3 45 10 FB
D0D,4E=21 46 40 01 CA 01 CD B8 4E 06 03 CD B8 4E 01 CC
D0D,5E=0F CD B8 4E EB xx DB CB E6 03 21 B9 50 20 01 34
D0D,6D=18 21 ED 78 0D A0 07 57 07 07 82 57 ED 78 0D E6
D0D,7D=0F 82 77 2B C9
. end of patch
```

FILTER PACKAGE NEWS

The LDOS Filter Package was released in January. Since then, some changes have been made to the disk. The source code for all of the programs now comes on the disk, and is also available for those of you who purchased the original package. This will be a free update. Also, for those of you who have the filter package with dates earlier than 2/20/82, apply the following patch to CALC/FLT.

```
.CALCA/FIX
. fixes problem with bad entry message display
D00,ED=0B 54 00 00 1F 0A 43 41 4C 43 2D
.EOP
```

This should stop problems when an incorrect entry is made.

HIGH MEMORY AND LDOS

The question of high memory usage comes up time and time again when talking about LDOS and application program compatibility. There also seems to be some confusion regarding when LDOS uses high memory. To make things perfectly clear, remember the following two points, and then read the explanation.

1) LDOS high memory usage is done in a totally relocatable manner. There should always be a way to avoid conflict with applications programs.

2) LDOS never uses high memory when booted without special configuration.

Avoiding memory conflicts

A memory conflict occurs when two programs or modules want to occupy the same memory locations. Since LDOS provides many advanced features, it needs to store the code for these features somewhere in memory.

It does this in the following manner:

1) Find the first available high memory address by looking at the value stored in the HIGH$ location.

2) Install the necessary code in memory below the current HIGH$ value.

3) Lower the HIGH$ value to protect the new code.

Any code that LDOS stores in high memory is written to be relocatable. This means that it can load anywhere in memory, and is not restricted to a specific area. Since LDOS always respects the HIGH$ value, it will never attempt to overlay any programs loaded and protected by changing the HIGH$ value in this manner. Now, if all other applications programs did the same thing when installing code in high memory, there would obviously never be any memory conflict.

Unfortunately, TRSDOS and some of the other operating systems do not respect the HIGH$ value. As a result, programs or BASIC USR routines that load in high memory are not generally written in a relocatable manner. They have a fixed load address, and MUST be loaded there to execute properly. This means that they will overlay any existing LDOS code that happens to be in those memory locations. When the LDOS code is something like the KI/DVR program, this usually results in an immediate system crash.

To resolve a memory conflict, you need only to know the load address and length of the unrelocatable code. We will consider two cases - when the code loads at the very top of memory, and when it loads at some other point.

When the conflicting code loads at the very top of memory, it is very easy to resolve the problem. Since you know the load address of the code, use the MEMORY library command to change the HIGH$ value to one byte below that address. For example, if a piece of code loads from address X'F900' and goes to the top of memory, you would issue a MEMORY (HIGH=X'F8FF') command. LDOS will now put any of its own high memory code below X'F900', protecting the module that will load there. You may want to use the SYSTEM (SYSGEN) command to keep this area permanently protected.

When the conflicting code does not load at the top of memory, you can use the same method just described to protect it. However, this will waste any memory between the end of the program and the top of memory. Let's consider the case where a module loads at X'F200' and extends to X'F3FF'. There is 3K of space between the end of the module and the top of memory. To avoid wasting this space, use the following procedure.

1) Load an LDOS module into high memory (i.e., SET KI/DVR, install a filter, etc).

2) Type in the command MEMORY with no parameters to see the current HIGH$ value.

3) If the HIGH$ value is above X'F3FF', repeat steps 1 and 2. If the value has gone below X'F3FF', you will need to start over, stopping before you load the module that caused the HIGH$ value to go below X'F3FF'.

4) Now, issue a MEMORY (HIGH=X'F1FF') command. This will protect the block of memory that will be needed by theunrelocatable module.

5) Continue to load any other LDOS modules as desired.

You may now use the SYSTEM (SYSGEN) command to permanently save this configuration with its protected block of memory.

CMD"O" - Implementation and Possible uses

In many Basic application programs, there exists a need to sort the data used by the program. There have been many different methods used and explained in the past to reach such a goal. This article will explain another possible sorting process inherent in LBASIC, namely the CMD"O" command.

The CMD"O" sorting routine allows you to sort information which is contained in a single dimension array, and can only be used with strings. The only constraint on the amount of information that can be sorted is determined by the amount of memory available. Realize that sufficient string space will need to be allocated for the data via the CLEAR statement. The syntax used for the CMD"O" command is:

```
CMD"O",number of elements to sort, first element of array to sort
```

You will note there are 2 parameters that need to be specified when issuing the CMD"O" command. The first parameter is the number of elements in the array that you wish to be sorted. This parameter may be specified as a numeric constant or a numeric expression.

The second parameter is the array that you wish the sort to be performed on, and the element number of the array where the sort is to begin. It is specified as the array name and a subscript. The subscript number may be specified as a numeric constant or a numeric expression.

Before we say anything more about the CMD"O" command, an example of implementing it is in order.

Example

Suppose you have the string array A$ dimensioned to have 6 positions (0-5), and the following assignments have been made to the array:

```
A$(0)="ZEKE"     A$(1)="HANK"     A$(2)="BOB"
A$(3)="GABE"     A$(4)="CLIFF"    A$(5)="DON"
```

The following sort commands will have these affects on the A$ array, assuming that the above assignments have been made to the array prior to invoking each sort command.

```
CMD"O",6,A$(0)
```

```
A$(0)="BOB"      A$(1)="CLIFF"    A$(2)="DON"
A$(3)="GABE"     A$(4)="HANK"     A$(5)="ZEKE"
----------------------------------------------
```

```
A%=6:B%=0:CMD"O",A%,A$(B%)

Same results as previous example
----------------------------------------------
```

```
CMD"O",3,A$(2)
A$(0)="ZEKE"     A$(1)="HANK"     A$(2)="BOB"
A$(3)="CLIFF"    A$(4)="GABE"     A$(5)="DON"
----------------------------------------------
```

Please note that whenever using the CMD"O" function, the sort cannot be performed beyond the last element of the array. For instance, in the last example, if the sort were still to begin with array position 2, the most elements that could be specified to be included in the sort would be four (specifying a number greater than four would force the sort beyond the highest subscript of the array).

It can be seen that the CMD"O" function provides a convenient way of performing a sort on a string array in ram. But what happens if you wish to create an alphabetic index file using the sorted array? In many applications, if a sort is performed in ram on an array, the subscript number of each element usually represents some type of index information (e.g. a record number in a random file), and is generally kept with each item throughout the sorting process. As it stands, the CMD"O" function will not allow you to keep track of an item's subscript number throughout the duration of the sorting process.

However, there are several alternatives available which can be used in conjunction with CMD"O" to maintain an alphabetic index file. One such alternative is to add a two byte string onto the end of each string in the array. This two byte string would be a compressed integer representing the index number associated with the data element (perhaps the arraypostion of the element prior to the sort).

The following example will illustrate how to create an index file using the CMD"O" command.

Example

Suppose you have a random file (MYFILE/DAT) which contains an unknown amount of 25 byte records, and you wish to sort these records alphabetically, creating an index file of sorted position according to record number. The following routine will illustrate one way of doing this using CMD"O".

```
OPEN"R",1,"MYFILE/DAT",25:FIELD1,25 AS D$
A%=LOF(1):DIM S$(A%)
FOR L=1 TO A%:GET 1,L:S$(L)=D$+MKI$(L):NEXTL
CMD"O",A%,S$(1)
OPEN"R",2,"MYINDEX",2: FIELD2,2 AS E$
FOR L=1 TO A%:LSET E$=RIGHT$(S$(L),2):PUT 2,L:NEXT L:CLOSE
```

After the above lines have been executed, the random file MYINDEX will have been created, and will contain 2 byte records representing an alphabetic index (by record number) of the file MYFILE/DAT stored as compressed integers..

PARITY = ODD

Since this is the first installation of what will hopefully be a regular feature, I thought it would be appropriate to introduce both myself and this column. I am by profession an electrical engineer, and by avocation a writer. You may have chanced upon some of my product reviews and articles in BYTE and INFOWORLD (I love to drop names!). My purpose here is twofold:

1) To review products of particular merit with an emphasis on their compatibility with LDOS.

2) To serve as a forum for the burning issues of the day which relate to LDOS.

I am interested in any and all reader feedback. If there is a particular piece of software you would like to see covered or a topic which needs to be discussed please drop me a line or leave a message on MNET.

Tentatively scheduled (very tentatively!) is a review of several spelling checkers, and a review of some recent statistical software for the TRS-80.

By no means is this column limited to reviews, however. From time to time I'll delve into the mysterious world of Assembly Language programming (to the great delight of all you experts out there) and perhaps explore the hidden secrets of the LDOS JCL. The day may even come when a reader contribution will be seen in these hallowed lines: provided you readers start contributing!

One final word is in order regarding the product reviews. My responsibility is first to you the readers. I have been told by the moguls who publish this journal, that advertising in no way influences what I can or can't say about a product. If its good I'll say so, if its not, I'll say that too, hopefully with a coherent reason why.

For those of you who wish to make yourselves heard, I herewith provide the necessary information:

Tim Daneliuk
4927 N. Rockwell St.
Chicago, IL 60625
MNET# 70745,1520

Now, onward and upward to today's exciting feature

PRODUCT OVERVIEW

NAME: DATAENTR 200 and ISAM 200
MANUFACTURER: Johnson Associates Software
Box 1402
Redding, CA 96099
916-221-0740

PRICE: $80.00 for DATAENTR 200, $90.00 for ISAM 200 (in BASIC), and $140.00 for ISAM 200 (in machine language).

DOCUMENTATION: Approximately 20 pages for DATAENTR 200 and 15 pages for ISAM 200

HARDWARE: TRS-80 Model I, II, or III, and CP/M systems.

INTRODUCTION

DATAENTR 200 and ISAM 200 are two packages which represent a growing trend in software development. These programs are not applications programs in themselves, but rather are a set of subroutines designed to be integrated into a larger package. Using this software, it might be possible to create a mailing list program, or a generalized data base manager for example. They are both available written in BASIC, and ISAM 200 is optionally available in machine language where optimum speed performance is essential.

DATAENTR200 - FEATURES

This set of sub-routines provides a convenient way for the applications programmer to develop keyboard entry programs. This not only simplifies the programming process, but also forces a bit of program structure by separating the input routines from the computational routines.

The heart of this input process is a data entry screen. This is the screen which the final user of the program sees as he or she enters data. In creating this screen the programmer defines the data field names and lengths. When the end user actually enters the data for these fields, it is returned as elements in a string array via the DATAENTR 200 subroutines. According to the manual, this program gives the applications programmer eight principal features to integrate into his or her code:

1) Load the screen from disk and hold it in memory.
2) Display the memory stored data entry form.
3) Provide full cursor and data entry control for 1 to 30 fields.
4) Provide verification and correction services for the data entry forms.
5) Provide keyboard controlled Menu selection.
6) Display data within the data entry form.
7) Clear screen data fields.
8) Provide single field correction.

These features are implemented in three distinct sets of programs.

The first of these programs is a set of run-time data entry subroutines. These routines read a screen from disk, display the screen, set up menu selections, input data (with optional checking of the fields as they are entered), display data, prompt the user to verify and correct data, clear data fields, and correct individual fields. The documentation describes each sub-routine individually, giving the variables that must be set before the routine is called, a description of what the subroutine does, and a short example.

The second program consists of run-time utility sub-routines. These provide time delay, blinking message, date/time strings, Y or N response checking, packing and unpacking of fields, a moving sign board message display, and a dated program listing with both time and date stamps. As with the previous routines, each sub-routine is individually documented. These two sets of programs are actually both contained in the program DATAENTR, but may be separated to save space.

The third program is a screen creation utility called UTSCREEN. This program allows screens to be created, examined, modified, stored, and printed. In creating the screen both the normal keyboard set and graphics characters (except on some CP/M systems) are permitted The utility also has features like line drawing and enclosing screens in video "boxes". The major feature of this screen utility is the ability to define field checking at the time the screen is created. This provides for run-time checking of data entered into each field. For example, it is possible to check if all the characters entered are alphabetic.

DATAENTR 200 - EVALUATION

By it's very nature, this program cannot be rated on overall performance. Those routines which I experimented with performed exactly as indicated in the documentation. The UTSCREEN screen utility has one major failing in that it is insufficiently error-trapped. Several times I inadvertently keyed in an incorrect key sequence and was rewarded with an LBASIC error message which bombed the program and destroyed that session's screen editing. UTSCREEN also has a problem in naming the files which store the screen configurations. The programmer is prompted for the file name (the extension /SCR is automatically appended), but one cannot specify which drive to save the file on! To use the program I had to write-protect my system disk (which is always full), to force the system to write the screen file to drive 1. Since I keep a JCL procedures library on the system drive, it became very cumbersome to have to remove the write protection every time I wanted to compile and execute a JCL procedure.

The documentation is generally adequate but a bit terse. One excellent feature of the documentation is a set of six step-by-step lessons which demonstrate the use and application of DATAENTR 200.

ISAM 200 - FEATURES

As with the program above, this program is a set of subroutines designed to be implemented in a larger package by the applications programmer.

These sub-routines are available written in either BASIC or machine language. As a whole, this set of routines implement ISAM (Indexed Sequential Access Method) file handling within a BASIC program. Specifically, the following seven file handling procedures are implemented:

1) Open the ISAM file.
2) Get a specific record.
3) Update an existing record.
4) Put a new record into the file.
5) Delete an existing record.
6) Get the next record in sequence.
7) Get the first record in the file.

The various file attributes like field lengths, field names, and record blocking (LRL) are stored within the file itself. When an ISAM file is opened, the ISAM 200 routines read these attributes and adjust themselves accordingly.

Several other programs are also included which assist in creation and maintenance of ISAM files. The INIT program is used to create a new file and write the file attributes to be used later. ISAMPRNT lists a file to either the printer or the screen. REORG is used to reorganize a file which has had many insertions and deletions. The program removes deleted records, and re-sequences the file to optimize run-time file handling.

As with DATAENTR 200, each sub-routine in ISAM 200 is individually documented with complete information regarding variable setting, general description of routine, error return codes, and short examples. Information is also supplied which discusses how ISAM 200 uses memory, and how to use multiple sort keys with the routines.

ISAM 200 - EVALUATION

This program can only be considered to be minimally compatible with the LDOS operating system. To begin with, the machine language routines which I evaluated were not fullyrelocatable, though 32K and 48K versions were provided on the master diskette. Worse yet was the fact that these programs did not set the HIGH$ memory pointer to protect themselves. This makes running any LDOS high memory options impossible unless one manually sets the memory pointer each time ISAM 200 is loaded.

Operationally, the routines seem to work well. A sample mailing list program is provided to demonstrate the power of the ISAM routines, and it does so convincingly. In general, the experienced BASIC programmer should have minimal difficulty implementing ISAM 200 in very sophisticated applications if the problems mentioned above are circumvented.

SUMMARY

Neither of these programs are for the novice. Both have good potential applications, but some expertise on the part of the applications programmer is mandatory.

PRODUCT REVIEW :
S B T (Structured Basic Translator)
written by Gene Bellinger
published and produced by Acorn Software Products, Inc.
$49.00

Reviewed by S. L. Ratkowski, Milwaukee, Wisconsin

==

Structured Programming is the magic buzzword in programming circles. Everyone 'knows' that BASIC is a poor language to program in because it is not 'structured'. In actual practice, it is possible to write structured programs in BASIC, but it takes a little more thought on the programmer's part than is required in a truly structured language such as Pascal.

Acorn Software has a new utility that is intended to make structured programming a little easier for the BASIC programmer -- they call it SBT, for Structured Basic Translator. SBT is a line oriented translator that understands only three type of lines:

1) STRUCTURE ELEMENTS.

All of the SBT structure elements begin with the '%' character. The allowable elements are:

%PROC - this is the PROCedure label. Think of a Procedure as a subroutine. All programs are made up of one or more procedures.

%CALL -- this is used to CALL aPROCedure.

%WHILE -- One of the three allowed loop structures. A DO-WHILE loop is used in a situation where you may not want the body of the loop to execute even once, since the test for the exit condition is done before the body of the loop is executed.

%UNTIL -- This loop structure is used whenever you want the body of the loop to execute at least once, no matter what. The exit test is done after the body of the loop is executed.
(The third loop structure is simply the BASIC FOR ... NEXT loop.)

%IF -- Combined with the next two structures, allows the construction of several forms of the IFThEN-ELSE structure.

%ELSE and

%ELSEIF -- Used with %IF to construct statements of the IF-THEN ELSE variety. They allow very versatile uses.

%ON -- This is used to create the CASE-CALL, which performs a conditional call of one of several procedures, depending on the value of it's argument. It translates to the BASIC sequence ON ... GOSUB ...

%ON ERROR -- Used to code error handling routines of the ON ERROR GOTO .. type. The programmer must be careful using this, as the last %ON ERROR block encountered is the one that controls when the error occurs.

'RESUME linenumber' cannot be used because SBT source code has no numbered lines.

%END -- This statement is simply a marker to delineate the end of a %PROC, %UNTIL, %WHILE, %IF, and %ON ERROR.

2) COMMENTS -- A comment in an SBT source program is any line that begins with a non-alphabetic character other than '%'. This type of line is ignored during processing. If you want to include REMarks to be included in the final BASIC program, include them in the source by beginning the line with 'REM'. You may not use the abbreviation " ' ", since SBT takes this line as a comment line not to be processed into the object program.

3) BASIC STATEMENTS -- Any line which is not a structure element, and is not a comment is ASSUMED to be a VALID BASIC statement, and is passed directly to the object program. SBT does NO checking on the validity of such statements, and this can cause some rather interesting crashes when you include comment lines not preceded by non-alpha characters, or incorrect BASIC statements.

To use SBT, you use the primitive text editor supplied on the disk, or any word processor capable of producing an ASCII file, to write the 'source' code of the program. This 'source' file must have the extension 'SBS'. I used SCRIPSIT to produce the following:

(Any lines that have '<--- ' in them are not in the source, but are used to explain what is happening in that program line.)
(The first 5 lines are examples of COMMENT lines.)

```
(*-------------------------------------------------------
(*    EXAMPLE OF PROGRAM WRITTEN USING  S B T
(*    --->> Structured Basic Translator <<---
(*        by Acorn Software Products, Inc.
(*-------------------------------------------------------
CLEAR 1000  <--- This is a plain BASIC statement
%CALL INIT   <---  I am calling the Procedure named INIT
%UNTIL CHOICE = 0
   %CALL MENU  <--- Calling the Procedure named MENU
   %CALL INPUT <--- Calling the Procedure named INPUT
   %ON  (CHOICE)  CALL ENTER,EDIT,PRINT,FILE  <--- Depending  on your  choice,
this will call one of the 4 Procedures.
%END UNTIL
PRINT SML$   <---  This is a pure BASIC statement.
END   <---  So is this.
(*-----------------------------------------------
(*    PROCEDURE  INIT  -- SHOWS USE OF
(*    CONDITIONALS TO SET UP CONSTANTS FOR
(*    EITHER MODEL I/III OR MODEL II USAGE.
(*-----------------------------------------------
%PROC INIT <--- This is the Procedure Label.
```

<--- The next 11 lines are all BASIC statements. I can use this as a general purpose menu routine just by changing these lines for each application I have.

```
   DIM DT$(100)
   N=5
   DIM MN$(N)
   MN$(1)= "1) Enter Data"
   MN$(2)= "2) Edit Data"
   MN$(3)= "3) Print Data"
   MN$(4)= "4) File Data to Disk"
   MN$(5)= "5) End Program"
   M$= "Enter your choice..."
   M2$="MENU OF OPTIONS"
   E$= "Invalid Input.  Please re-enter..."
```

<--- The following lines are examples of CONDITIONAL TRANSLATION. When I execute the Translator, I can set SWITCHES that will tell the translator to include some lines but not others. I work on a Model III at home, and use a Model II at the office. I would love to be able to write programs that would run on both. By using this feature of SBT, I can.

All of the lines below that begin with '$M2$' will only be translated if I set the proper switch at translation time. If I specify NO switches at translation time, none of these statements will be taken to be comments by SBT.

<--- Model II constants

```
   $M2$ SML$= CHR$(30)   (*  CLS AND SET TO 80 CHAR MODE *)
   $M2$ BIG$= CHR$(31)   (*  CLS AND SET TO 40 CHAR MODE *)
   $M2$ ROW = 24          (*  24 LINES ON CRT *)
   $M2$ COL = 80          (*  80 CHAR LINES *)
```

<--- Model I/III constants

```
   $M13$SML$= CHR$(28) + CHR$(31)  (*CLS & SET TO 64 CHAR MODE *)
   $M13$BIG$= CHR$(28) + CHR$(31) + CHR$(23) (* CLS & SET TO 32 CHR MODE *)
   $M13$ROW = 16          (*16 LINES ON CRT *)
   $M13$COL = 64          (*64 CHAR LINES *)
%END PROC (INIT)
```

Once you have created your source file and saved it as an ASCII file with the extension '/SBS', you go into BASIC and run SBT. Under LDOS 5.1.2 on the Model III, the procedure goes as follows:

```
LDOS Ready

LBASIC RUN"SBT/"


LBASlC - Version 5.1.2 - 03/01/82
(C)  1981 by Logical Systems Incorporated


                        SBT - Structured Basic Translator
                              Ver # 3.1 18-Jan-81
```

F1x> EXAMPL/$M13$ <-- The program prompts you for the file name of the source code. The extension '/SBS' is assumed.

```
$F$EXAMPL/$M13$
   00100  -  INIT  <--- As  SBT  processes  the  source,  it  shows  you  the
```
subroutines it is setting up from thePROCs in your program.

```
   00200 - MENU
   00300 - INPUT
   00400 - ENTER
   00500 - EDIT
   00600 - PRINT
   00700 - FILE
READY
```

At this point, you can LOAD and RUN the object program, which is saved on the disk as an ASCII program with the extension '/BAS'. Notice that I used the switches to tell SBT to only translate the Model I/III constants, not the Model II constants. I include both the source code of the program (FIGURE 1) and the resulting BASIC program (FIGURE 2) as an example of what SBT can do.

The translation of this program took 2 minutes and 16 seconds. The writing of the source took me about an hour and a half, starting from scratch with the manual in hand. It took three tries before I cleaned out the goofs caused by reading the manual too quickly.

The program is easy to use, but is not for the beginning programmer. If you make a mistake in the syntax the translator is looking for, you can wind up with a BASIC program that does not make sense. You must have a feel for what should come out of the translator to figure out what you did wrong in the source.

I like structured programing, but it sometimes requires too much planning when I program in BASIC. The SBT is definitely a tool that would help me write more structured programs. The documentation is good, and the examples will help you over the rough spots.

This product comes on a 35 track single density TRSDOS diskette that can be booted on a Model I, or CONVERTed by a Model III. The instructions for use are included in a small format, 25 page booklet that includes all of the allowed syntax of the translator and examples of their usage. In addition, the disk includes three source programs, and the translated object code that SBT produces. Interestingly enough, one of the examples is SBT itself.

One other note is that the source program for SBT includes the switches $TRS$, $CPM$, and $I$. To run on the TRS-80, it must be translated as SBT/$TRS$/$I$. If you wish to translate it to run with Microsoft BASIC 4.51 under CP/M vers. 2.2, use the switches SBT/$CPM$/$I$. If you with to run it under Microsoft BASIC vers. 5.2, use SBT/$CPM$.

```
============================
     FIGURE1
============================

(*--------------------------------------------------------
(*    EXAMPLE OF PROGRAM WRITTEN USING  S B T
(*    --->> Structured Basic Translator <<---
(*        by Acorn Software Products, Inc.
(*--------------------------------------------------------
CLEAR 1000
%CALL INIT
%UNTIL CHOICE = 0
   %CALL MENU
   %CALL INPUT
   %ON (CHOICE) CALL ENTER,EDIT,PRINT,FILE
%END UNTIL
PRINT SML$
END
(*-------------------------------------------------
(*    PROCEDURE  INIT  -- SHOWS USE OF
(*    CONDITIONALS TO SET UP CONSTANTS FOR
(*    EITHER MODEL I/III OR MODEL II USAGE.
(*-------------------------------------------------
%PROC INIT
   DIM DT$(100)
   N=5
   DIM MN$(N)
   MN$(1) = "1)   Enter Data"
   MN$(2) = "2)   Edit Data"
   MN$(3) = "3)   Print Data"
   MN$(4) = "4)   File Data to Disk"
   MN$(5) = "5)   End Program"
   M$= "Enter your choice..."
   M2$="MENU OF OPTIONS"
   E$= "Invalid Input.  Please re-enter..."
   $M2$ SML$= CHR$(30)   (*  CLS AND SET TO 80 CHAR MODE *)
   $M2$ BIG$= CHR$(31)   (*  CLS AND SET TO 40 CHAR MODE *)
   $M2$ ROW = 24          (*  24 LINES ON CRT *)
   $M2$ COL = 80          (* 80 CHAR LINES *)
   $M13$SML$= CHR$(28) + CHR$(31)  (*CLS & SET TO 64 CHAR MODE *)
   $M13$BIG$= CHR$(28) + CHR$(31) + CHR$(23) (* CLS & SET TO 32 CHAR MODE *)
   $M13$ROW = 16          (* 16 LINES ON CRT *)
   $M13$COL = 64          (*64 CHAR LINES *)
%END PROC (INIT)
(*-------------------------------------------------
(*    PROCEDURE MENU  --  DISPLAY MENU
(*-------------------------------------------------
%PROC MENU
   PRINT BIG$
   L = ((COL/2)- LEN(M2$))/2
   PRINT TAB(L) M2$
   PRINT
   X=1
   %UNTIL X > N
      PRINT MN$(X)
      X=X+1
   %END UNTIL
%END PROC
(*-------------------------------------------------
(*   PROCEDURE INPUT  --  GET CHOICE
(*-------------------------------------------------
%PROC INPUT
   SCR = (ROW - 1) * COL
   PRINT
   %UNTIL VAL(A$)>0 AND VAL(A$)<6
      L = ((COL/2)- LEN(M$))/2
      PRINT TAB(L) M$
      %UNTIL A$<> ""
         A$= INKEY$
      %END UNTIL
      %IF VAL(A$)<1 OR VAL(A$)>5
         PRINT @ SCR, E$
      %END IF
   %END UNTIL
   CHOICE = VAL(A$)
   %IF CHOICE = N
      CHOICE = 0
   %END IF
%END PROC
(*-------------------------------------------------
(*   PROCEDURE ENTER  -- GET DATA FROM KB
(*-------------------------------------------------
%PROC ENTER
   PRINT SML$
   PRINT "DUMMY ENTER ROUTINE"
   INPUT"HIT 'ENTER' TO CONTINUE";A$
%END PROC
(*-------------------------------------------------
(*   PROCEDURE EDIT  --  EDIT DATA
(*-------------------------------------------------
%PROC EDIT
   PRINT SML$
   PRINT "DUMMY EDIT ROUTINE"
   INPUT"HIT 'ENTER' TO CONTINUE";A$
%END PROC
(*-------------------------------------------------
(*    PROCEDURE  PRINT  -- PRINT DATA
(*-------------------------------------------------
%PROC PRINT
   PRINT SML$
   PRINT "DUMMY PRINT ROUTINE"
   INPUT"HIT 'ENTER' TO CONTINUE";A$
%END PROC
(*-------------------------------------------------
(*    PROCEDURE   FILE  --  FILE DATA
(*-------------------------------------------------
%PROC FILE
   PRINT SML$
   PRINT "DUMMY FILE ROUTINE"
   INPUT"HIT 'ENTER' TO CONTINUE";A$
%END PROC


============================
     FIGURE  2
============================

1 CLEAR 1000
2 GOSUB 100
3 GOSUB 200
4 GOSUB 300
5 ON((CHOICE))GOSUB 400,500,600,700
6 IF NOT(CHOICE = 0) THEN 3
7 PRINT SML$
8 END
100 DIM DT$(100)
101 N=5
102 DIM MN$(N)
103 MN$(1) = "1)   Enter Data"
104 MN$(2) = "2)   Edit Data"
105 MN$(3) = "3)   Print Data"
106 MN$(4) = "4)   File Data to Disk"
107 MN$(5) = "5)   End Program"
108 M$= "Enter your choice...
109 M2$="MENU OF OPTIONS"
110 E$= "Invalid Input. Re-enter..."
111 SML$= CHR$(28) + CHR$(31)
112 BIG$= CHR$(28) + CHR$(31) + CHR$(23)
113 ROW = 16
114 COL = 64
115 RETURN
200 PRINT BIG$
201 L = ((COL/2)- LEN(M2$))/2
202 PRINT TAB(L) M2$
203 PRINT
204 X = 1
205 PRINT MN$(x)
206 X = X + 1
207 IF NOT(X > N) THEN 205
208 RETURN
300 SCR = (ROW - 1) * COL
301 PRINT
302 L = ((COL/2)- LEN(M$))/2
303 PRINT TAB(L) M$
304 A$= INKEY$
305 IF NOT(A$<> "") THEN 304
306 IF NOT(VAL(A$)<1 OR VAL(A$)>5) THEN 308
307 PRINT @ SCR, E$
308 IF NOT(VAL(A$)>0 AND VAL(A$)<6) THEN 302
309 CHOICE = VAL(A$)
310 IF NOT(CHOICE = N) THEN RETURN
311 CHOICE = 0
312 RETURN
400 PRINT SML$
40l PRINT "DUMMY ENTER ROUTINE"
402 INPUT"HIT 'ENTER' TO CONTINUE";A$
```

```
403 RETURN
500 PRINT SML$
501 PRINT "DUMMY EDIT ROUTINE"
502 INPUT"HIT 'ENTER' TO CONTINUE";A$
503 RETURN
600 PRINT SML$
601 PRINT "DUMMY PRINT ROUTINE"
602 INPUT"HIT 'ENTER' TO CONTINUE";.A$
603 RETURN
700 PRINT SML$
70l PRINT "DUMMY FILE ROUTINE"
702 INPUT"HIT 'ENTER' TO CONTINUE";A$
703 RETURN
```

THE LIBRARY by Earle Robinson

The library found on your LDOS disk is a continuation of the system begun under the first versions of TRSDOS. You may have frequently wondered what it is and what is its use. The library was created in order to provide the user with a number of utility programs of relatively short length so as to occupy a minimum amount of disk space. Note that I said these are programs because that is what they are. It would be possible for each and every one of the library programs to be on your disk as /CMD files. You could easily extract these routines and create /CMD files on your disk. You would, however, have a much more encumbered disk, using over double the space and occupying some 33 file spaces.

The library was created in order to save space on the disk. Remember that the Model I disk drive system relied on 35 track diskettes in single density. Model I TRSDOS has far fewer programs than does LDOS, but the library, even in that rather bare bones operating system, saved enormous space on the disk. Under LDOS, if the 33 different library programs were individually stored as programs on the disk, they would occupy at least 165 sectors or 17 tracks (if a double density disk, 198 sectors, or 11 tracks). You can appreciate that such a disk would not leave much spare space. Also, not to be forgotten is the maximum number of permitted files on a single density disk, 48 (*1). If the library were split into files, the available number of files would be reduced to only 15! The library in fact concentrates the 33 library programs into a space of 88 sectors. How is this done?

Each program file on a disk uses a minimum of onegran of a disk (5 sectors in single density or 6 sectors under double density). Many of the library routines are quite short, occupying 1 or 2 sectors. If they were stored as files, each would be allocated at least 1gran of space though using only a sector or two. The library packs these routines into 2 separate system files, SYS6 and SYS7. Two files are used due to the huge size of the library which would make a single one impractical. In addition, the use of a second file permits allocating space between each of them based on their importance in day to day use. (Single drive users of LDOS can easily run most of the time with a system disk without SYS7 being present). Not only does the packing of the library programs economize disk space, but it permits double or even triple use of the same code for certain routines such as COPY and APPEND. This is accomplished by using different entry points to the same routine where much of the same code is used by different routines. CONT. Page 37

MISOSYS MISOSYS MISOSYS MISOSYS MISOSYS MISOSYS MISOSYS MISOSYS MISOSYS MISOSYS MIS

Now you can obtain the Assembler used to maintain **LDOS** and save **$1Ø** to boot. **EDAS** is a Mod I/III disk only assembler which produces executable object code from one or more source disk files and/or memory buffer. Supported features are conditional assembly, paged and titled listings, global search and replace, 14-char labels, lower case text entry, block move, **XREF** utility, and more. **EDAS** is priced at $69 + $4 S&H until May 1.

User Partitioned Data Sets are now possible with the **PDS** utility. Now you can build your own library - just like **SYS6** and **SYS7**. Archive files without cluttering up directories. Saves disk space! The **LDOS** user will find **PDS** files useful for customizing **SYSTEM** disks and other **CMD** libraries. Includes commands to: **APPEND**, **BUILD**, **COPY**, **DIR**, **KILL**, **LIST**, **PURGE**, and **RESTORE**. Requires Mod I/III version 5.Ø.2 or later. **PDS** is $4Ø + $2 S&H.

The **LDOS** Editor, **LED**, is the latest professional product from **LSI**. **LED** is a general purpose screen oriented text editor. It supports **INSERT** block, line, character; **DELETE** block, line, character, word, end; global **FIND** & **REPLACE**; page up & down, top, end, left, & right; **TAB** forward & backward; **INDENT**; **HEX** entry, **MENU** mode; and more. Use **LED** to edit KSM, FIX, JCL, & TXT files. Use **LED** to edit any **ASCII** file. **LED** is here!: $4Ø + $2 S&H.

⟨[<(Model I **DDEN** Users)>]⟩ **MISOSYS** releases **SOLE** - the utility to implement double density **BOOT** on LDOS. **SOLE** **DDEN** diskettes can be backed up with the **BACKUP** utility. Put a **SOLE** on your **BOOT**: $25 + $1 S&H. Paper **SOURCE** $25.

Now, from **MicroConsultants**, is the **HELP** command series. **HELP** provides extremely fast screen prompting on the **LDOS** Library, Utilties, Filters, Drivers, **LBASIC**, & **PDS**. The Quick Reference Card and a **HELP** generator are included. **HELP** with **QRC** is $25, $2 S&H **QRC** sold separately for $3.

You've tried the others, now get the best! If you ever do any modification of disk files, then **FED**, the screen oriented file editor from **Galactic**, is the solution. **FED** will locate any byte by its load address or position in the file. **FED** permits modification in ASCII or HEX as well as global search. Record position forward, backward, end, or beginning. Displays split screen in ASCII & HEX with triple cursor. **FED**: $4Ø + $2 S&H

The **LDOS FILTER** disk adds additional power to your system. The 14 filters include: XLATE, LISTBAS , TITLE, TRAP, STRIP7, STRIPCNT, LOWER, UPPER, CALC, PAGEPAWS, LINEFEED, SLASHØ. The **FILTER** has complete SOURCE code: $6Ø + $3 S&H

Have you played around with **CP/M** and now want to get at those files using **LDOS**? Here is the solution: **CONVCPM** will transfer **CP/M** files to **LDOS**. Works with **SDEN**; 8" or Lifeboat's and Omikron's 5". **CONVCPM**: $3Ø + 2$ S&H.

MasterCard

MISOSYS
P. O. Box 4848
Alexandria, VA 223Ø3-Ø848
7Ø3-96Ø-2998

VISA

MISOSYS MISOSYS MISOSYS MISOSYS MISOSYS MISOSYS MISOSYS MISOSYS MISOSYS MISOSYS MIS

SUPER UTILITY +

SUPER UTILITY +
(C)(P) 1981 BY
BREEZE/QSD, INC.

By: KIM WATT

Read Paul Wiener's review on page 366
in the Jan. '82 80-MICROCOMPUTING.(tm)
SU+ has even MORE features now!!

New Packaging Policy!! Now get Super Utility Plus I/III
on the same disk for only $74.95 (80 trk available on req.)

Compatible with LDOS(tm) Mod I/III as well as other DOS's.

POWERDRAW

A SCREEN GRAPHICS UTILITY FOR MOD I or III
SAVE SCREENS IN 6 DIFFERENT FORMATS!

By: KIM WATT   only 3995

PowerDraw allows you to design graphic screens on
your video, and save them to as many as 33 memory
buffers. The buffers may be saved in 6 formats:

Compressed	BASIC Strings
EDAS	BASIC data
EDTASM	Load File

These files can be MERGED with your BASIC or assembly
program very easily. You can easily design professional
screen displays. You can also print the screens out on
a "graphic" printer. This ad was created using an Epson
MX80 (no GRAFTRAX) and PowerDraw. PowerDraw contains a
built-in MENU that really makes drawing a BREEZE! If
you own an EPSON printer, you WANT and NEED PowerDraw!

YOU CAN NOW EASILY CREATE, SAVE,
OR PRINT GRAPHIC SCREEN DISPLAYS!!

POWERSOFT

GREAT for EPSON's! With or without GRAFTRAX-80(tm)

11500 Stemmons Expressway, Suite 125 - Dallas, Tx 75229

M/C-VISA
CARDS
Accepted

(214)484-2976 -- MicroNet# 70130,203
Join OUR Xtra-80 Sig ON MicroNet!
From the "OK" prompt, type: R QSD

Write for
our FREE
Catalog!

DEALER INQUIRIES INVITED

Mention that you saw this ad in the LDOS
News and WE PAY SHIPPING! Save $5! (US/Can)
PowerSoft is a division of Breeze/QSD, Inc.
LDOS is a TM of LSI, Inc.

FOREIGN add $10
shipping (AIR)
NO CHECKS!
U.S. FUNDS ONLY

Contact us for OUR price on the LAREDO 5 Meg Hard Drive!

More for your Dollars!

LDOS PACKAGE SPECIALS.

THE ULTIMATE DISK-OPERATING SYSTEM LDOS™

The new generation of operating system for the TRS-80™ Far superior to any on the market. It is a totally independent device system, capable of device linking, routing, setting, and filtering. LDOS will support 5" and 8" floppies, single/double density, single/double sided, and up to 80 tracks. Excellent documentation in a tab indexed manual (over 280 pages). Call today. Available for Model I or III. . . only $129.00. LDOS™ is a product of Logical Systems Inc. Also available: EDAS 3.5 Model I and III. . . $79.00

LDOS PACKAGE SPECIALS

		YOU SAVE
LDOS + MAIL/FILE	$199.00	$89.00
LDOS + INVENTORY MOD I	$199.00	$89.00
LDOS + INVENTORY MOD III	$299.00	$89.00
LDOS + STOCK MARKET MON.	$179.00	$49.00
LDOS + EDAS	$179.00	$29.00
LDOS + FED (FILE EDITOR)	$149.00	$20.00
LDOS + LED (LDOS EDITOR)	$139.00	$20.00
LDOS + FED & LED	$169.00	$30.00
LDOS + FILTER PACKAGE	$169.00	$20.00
LDOS + "LC" LANGUAGE	$229.00	$50.00
LDOS + MONITOR	$139.00	$15.00
LDOS + ULTRA TREK	$129.00	$19.95
SUPER LDOS DEVELOPMENT PACKAGE		
LDOS + LC + FED + LED + EDAS + FILTER PKG. + MONITOR	$369.00	$144.00

**Offer good during April & May, 1982

FED - THE LDOS FILE EDITOR

This is the ultimate in file "ZAP" type tools for use with your LDOS system. Full access to files, ASCII and HEX searches, find a load address or where a byte loads. Full HEX or ASCII modify, even to the directory. Two display modes and a built in menu plus unique features that have never been available in this type of system tool. Available now for just. . . $40.00

LC - LDOS "C"

This is a very powerful implementation of the "C" programming language. It comes complete with an extensive function library and generates executable code or assembler source that is compatible with either the MAC-80 or EDAS. For efficient creation of both systems and applications code, this is the language. The language of the future now for just. . . . $150.00

LED - LDOS TEXT EDITOR

This is the official LDOS text editor designed for writing and maintaining almost any type of text. Handles both line number and unnumbered text files with word processor like functions. Great for all types of pure ASCII source files, even BASIC, and for PATCH and JCL files. Many special features make this editor a "must have" program for the serious user. Finally, a functional text editor for just. . . $50.00

FILTER PACKAGE

This package is a collection of powerful filter for the LDOS "devices" to bring out the true power of the LDOS concept. Included are a HEX-DECIMAL-BINARY-CALCULATE-CONVERT filter for your keyboard, a filter to expand basic code to a formated structure during listing and a complete translations filter to convert any or all 256 possible characters to other characters even in a by directional mode. Use this package to give your system a DVORACK keyboard or to have your system talk in EBCDIC. Many, many handy additions to your LDOS system for just. . . $60.00

MONITOR - THE LDOS DISK I/O MONITOR

This program brings to LDOS the power of a mini or mainframe error monitor system. When a disk error of any sort occurs, the monitor takes control and allows the operator to select: ABORT, CONTINUE, RETRY or IGNORE. This allows recovery from many problems that before would have caused the loss of data or the interruption of program. Don't let a simple parity error ruin your day again! Get the program that pays for itself the first time it does its job. Be in control of your system for just. . . $25.00

MAILING SYSTEMS - MAIL/FILE-SERIES II

NEW Series II Mail/File has all of the outstanding features of the original Series I Mail/File. PLUS many additional qualities which again will set standards for this type of system. "Word processor" type input editor, fast sorting, and fabulous editing capabilities are a few of its features. Name and ZIP code are under constant sort. A really great Mailing List-Data Base manager. Model I system will handle up to 600 names. Model III system will handle up to 1200 names. . . $159.00

BUSINESS SYSTEMS - INVENTORY MASTER SYSTEM

The program to fill your needs. Control up to 2700 items. With today's market, keeping on hand only what your demand calls for is reason enough to let your TRS-80 accurately and effectively take care of your inventory. Always know what to stock and when to stock it. This system has many features which were modeled after a main frame system of large capacity. Model I version just. . . $159.00. Model III version just. . . $259.00. Also available for the Model I or III Stock Market Monitor: Cassette version. . . $89.00. Disk version. . . $99.00

Money orders, credit cards & cod's shipped within 24 hours.

TRS-80™ is a trademark of Tandy Corp.

VISA

Can't wait? Call

414/241-8030

galactic software ltd.

11520 North Port Washington Road
Mequon, Wisconsin 53092

LDOS (tm)LSI UTILITY DISKS

from

By: KIM WATT

POWERSOFT

a division of Breeze/QSD, Inc.

UTILITY DISK #2 (c)1982

$29.95

Disk #2 Includes:

PMOD/CMD - A Disk-File-Memory Modification Utility.

```
    PMOD,aaaa (display memory address)
    PMOD,filespec,aaaa (file to display,starting rel. sector)
    PMOD :b,c,d (drive,cylinder,sector to display)
```

UTILITY DISK #3 (c)1982

$29.95

Disk #3 Includes:

PCHECK/CMD - LDOS Directory Check Utility. Checks a drive or a file, (GAT, HIT, BOOT, etc.) for integrity.

PFIX/CMD - FIXES problem discovered by PCHECK/CMD. You may repair GAT or HIT table, repair BOOT sector, or repair FILE records. This program might fix that crashed disk that has you going nuts!

UTILITY DISK #4 (c)1982

$29.95

Disk #4 Includes:

PFIND/CMD - FIND a string, replace a string, find a byte, find a WORD, replace a byte or word! FIND ANYTHING on the disk FAST! Replace it FAST!

PCOMPARE/CMD - Compare a FILE or a sector to any other file or sector. Detects the minutest differences! Send the outputs from above to video or printer.

UTILITY DISK #5 (c)1982

$29.95

Disk #5 Includes:

PREFORM/CMD - Disk Re-Formatter Utility. Fixes CRC errors and "stengthens" the format on older disks. A FORMAT without ERASE utility(5" floppy only)

PVU/CMD - a Disk Verification Utility. Detects bad sectors. Fix them with PREFORM/CMD.

PERASE/CMD - Disk Bulk Eraser (5" floppy only)

UTILITY DISK #6 (c)1982

$29.95

Disk #6 Includes:

PCLEAR/CMD- Erases unassigned sectors and Directory slots. Removes ALL traces of "killed" files!

PSS/CMD - Sector Status Utility. Finds WHAT file is assigned to WHAT sector.

PMAP/CMD - Disk-File Mapping Utility. Map a FILE or a disk! Shows exactly WHERE all files are with extents.

UTILITY DISK #7 (c)1982

$29.95

Disk #7 Includes:

PMOVE/CMD - A SUPER FAST Multiple Copy Utility!

PDIRT/CMD - Read a Mod III TRSDOS directory from LDOS!

PASSGO/CMD - Removes passwords from A file or a WHOLE disk!

PUN/CMD - UN-REPAIR a disk. (Mod I only)

PEX/CMD - A disk exerciser for head cleaning kits.

UTILITY DISK #8 (c)1982

$29.95

Disk #8 Includes:

PMX/FLT-MX80 graphics

PHELP/CMD -Instant HELP! Very complete. Saves time!

PBOOT/FIX - Customize YOUR LDOS Boot graphics and message!

PFILT/FLT - A User Definable Filter. Very flexible!

DVORAK/FLT - DVORAK/QWERTY Keyboard Filter.

DVORAK/JCL - DVORAK Keyboard Table (used with PFILT/FLT)

CODE/JCL - Keyboard EN-Coding TABLE (used with PFILT/FLT)

DECODE/JCL - Keyboard DE-Coding Table (used with PFILT/FLT)

UTILITY DISKS for LDOS(tm) are now ready for delivery! These fine utility packages are written entirely by Kim Watt (auth. of SUPER UTILITY) and requires LDOS 5.1x Mod I or III. They fully support all drive configurations, including single and double density, double sided drives, and the LAREDO 5 Meg Hard Drive! Expand your use of LDOS to the MAXIMUM!

ALL of the UTILITIES are supplied on disk with full doc on heavy paper punched for your LDOS binder. Each package only..... $29.95

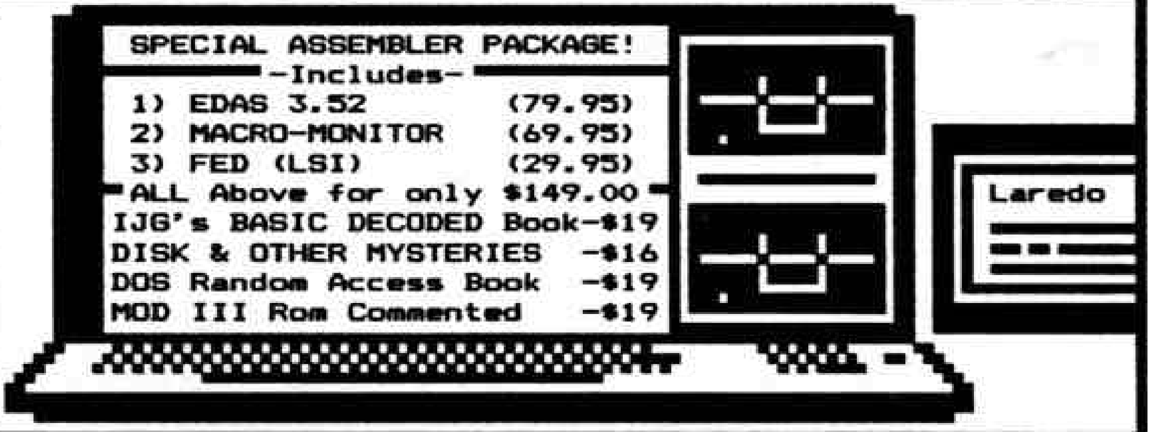
MACRO-MON...THE SHADOW!

From Advanced Opsys By: Jake Commander

The SHADOW allows you disassemble and examine program instructions from ANY part of memory. You can single step thru ROM. You can load a machine lang. proram from tape or disk and begin execution at a user-specified breakpoint, one instruction at a time, with user defined time delay between instructions. It will disassemble each routine as it is being executed and route the current register values to your screen or printer. You may also search thru memory for a specific string up to 16 bytes in length. The SHADOW allows machine language prorams to be easily relocated within memory, with ALL internal calls and jumps changed AUTOMATICALLY to execute in the new location. Completely user-relocatable and compatible with LDOS as well as other major DOS's. Complete with attractive binder and full doc.

---> Mod I disk - $59.95 / Mod III disk - $69.95 <---

SPECIAL VALUE


SPECIAL ASSEMBLER PACKAGE!

-Includes-

1) EDAS 3.52 (79.95)
2) MACRO-MONITOR (69.95)
3) FED (LSI) (29.95)

ALL Above for only $149.00

IJG's BASIC DECODED Book	-$19
DISK & OTHER MYSTERIES	-$16
DOS Random Access Book	-$19
MOD III Rom Commented	-$19

(These ads ENTIRELY produced with PowerDraw on an Epson-80)

PowerSoft 11500 Stemmons Expway, #125 Dallas, Tx 75229

M/C-VISA CARDS Accepted

(214)484-2976 -- MicroNet# 70130,203
Join OUR Xtra-80 Sig ON MicroNet!
From the "OK" prompt, type: R QSD

PROGRAM SUBMISSIONS WELCOME!

DEALER INQUIRIES INVITED

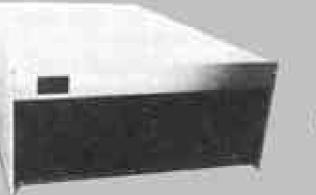
Mention that you saw this ad in the LDOS News and WE PAY SHIPPING! Save $5! (US/Can)
PowerSoft is a division of Breeze/QSD, Inc.
LDOS is a TM of LSI, Inc.

FOREIGN add $10 shipping (AIR)
NO CHECKS!
U.S. FUNDS ONLY

Contact us for OUR price on the LAREDO 5 Meg Hard Drive!

Quality Computer Services presents..

Hard Disk Systems

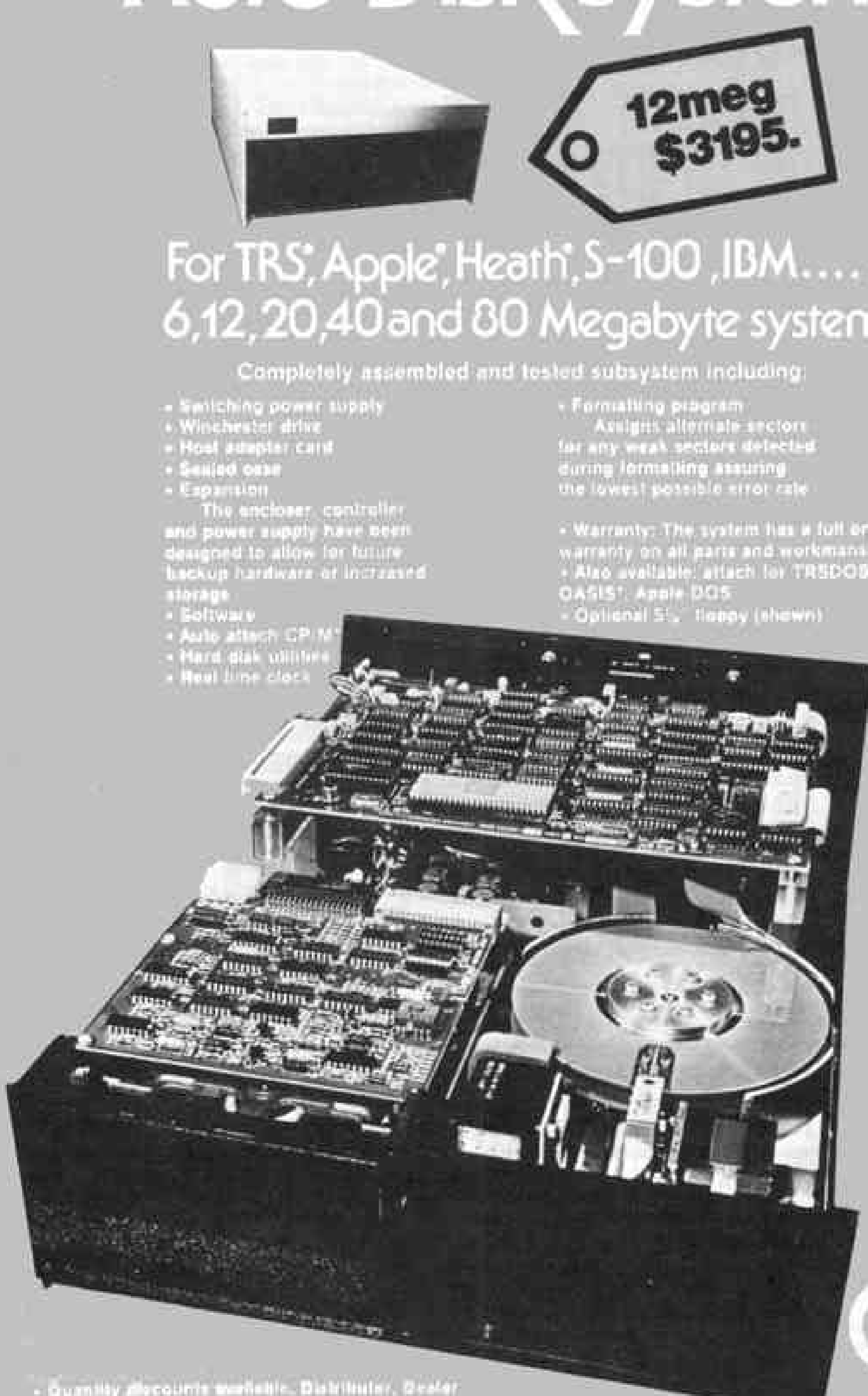


12meg
$3195.

For TRS*, Apple*, Heath*, S-100, IBM......
6,12,20,40 and 80 Megabyte systems

Completely assembled and tested subsystem including:

- Switching power supply
- Winchester drive
- Host adapter card
- Sealed case
- Expansion
  The encloser, controller and power supply have been designed to allow for future backup hardware or increased storage
- Software
- Auto attach CP/M*
- Hard disk utilities
- Real time clock
- Formatting program
  Assigns alternate sectors for any weak sectors detected during formatting assuring the lowest possible error rate
- Warranty: The system has a full one year warranty on all parts and workmanship
- Also available: attach for TRSDOS, OASIS*, Apple DOS
- Optional 5¼" floppy (shown)



See local dealer
or call
Quality Computer
Services
178 Main Street
Metuchen, New Jersey
201-548-2135

QCS inc

- Quantity discounts available. Distributor, Dealer and OEM inquries invited.

*OASIS is a trademark of Phase One Systems, CP/M is a trademark of Digital Research

Toll free number: 800 631 5944

FASTER THAN ANY FLOPPY DISK DRIVE!

FOR LESS THAN $40.0

Yes . . . through the power and flexibility of the **LDOS** Operating System, Galactic Software brings the TRS-80 user . . . memDISK.

By using a portion of the existing RAM in your TRS-80, memDISK will create a fully functional disk type device. This RAM based disk may be accessed as a normal disk drive with copy, backup, free, dir, save, load, dump, etc. You can even have your memDISK appear automatically everytime you boot your system by using the **LDOS** System (SYSGEN) feature. The amount of storage available on your memDISK is user selected from 1.5k to 28.5k. The access time of your memDISK device is guaranteed to be **FASTER THAN ANY FLOPPY DRIVE** available, and it's even faster than most hard drive systems. To assure reliability, memDISK even tests the RAM area it will use, before installing itself. Now you can do things with your system that were too slow or impossible before the memDISK system became available. Best of all memDISK involves no additional hardware of any sort, nothing to align, adjust, clean or break. It's all software! All this for as little as $30 (when purchased with **LDOS**).

The memDISK system requires a minimum of a 32k, single drive, TRS-80 (Mod I or III) and the **LDOS** Operating System.

memDISK is just $39.00 for **LDOS** owners or you can get Mod I or III **LDOS** 5.1.1 with memDISK for a package price of just $159.00 complete. Source code is available to memDISK owner for $50.00.

Also available from Galactic:	PRICE		SHIPPING & HANDLING
1. FED - The **LDOS** File Editor	$ 40	+	$2
2. LED - The **LDOS** Text Editor	$ 40	+	$2
3. LC - The **LDOS** "C" Compiler	$150	+	$4
4. I/O MONITOR - The **LDOS** Disk Error Monitor	$ 25	+	$2
5. PDS - Partitioned Data Sets For **LDOS**	$ 50	+	$2
6. MAIL/FILE - The **LDOS** Compatible Mailing List System	$159	+	$4
7. EDAS - The **LDOS** Editor Assembler	$ 79	+	$4
8. FILTER PACKAGE - 13 useful **LDOS** Filters Complete with Source	$ 60	+	$3
9. **LDOS** & memDISK - An Unbeatable Combo	$159	+	$5
10. And of course Mod I or Mod III **LDOS**	$129	+	$5

If you order any two items listed, deduct 10% as a package discount.

11520 North Port Washington Road
Mequon, Wisconsin 53092

For TRS-80 Models I and III

Introducing the Lowest Cost Hard Disk System Ever . . Laredo's LS525.

$1995*

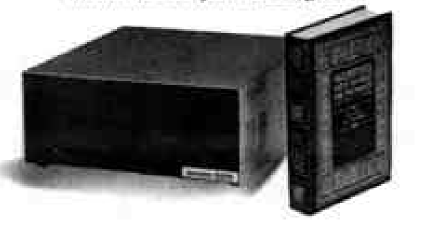
5 million low cost bytes of on-line storage.

At $1995.00 affordable hard disk storage is now within your reach.

The LS525 hard disk system is easier to use than your printer . . . just plug it in switch it on and your disk storage problems are over forever.

The LS525 gives you unheard-of speed and reliability. For starters we built a "mainframe like" controller that can detect disk surface defects and relocate the data to alternate tracks. Talking about speed the LS525 can access files 10 times faster than your floppy. This coupled with proven "winchester" technology is what we mean by reliability and speed.

We also provide a user-friendly TRSDOS compatable operating system in which all of your existing software can run on. It provides features that can make the LS525 look like one big 5M byte drive or four small 1.25M drives or whatever you like.



The LS525 is a compact (less than one half cubic foot) "color-keyed" package that will fit perfectly next to your TRS-80. So if your tired of seeing "Disk Space Full". . . . .

. . . there's no excuse.

Call toll free and order yours now.

(800) 538-5137

Inside California call (408) 980-1888.

* Model I or III host adapter $250 & LDOS $160 extra, specify when ordering.

laredo systems inc.

. . .mass storage for personal computers.

Laredo Systems Inc. 2264 Calle De Luna, Santa Clara, California 95050

TRS-80 and TRSDOS are trademarks of Radio Shack a Tandy Co.

THE
ALTERNATE
SOURCE

Programmer's Journal

We could tell you that it's the best available,

BUT WE'D RATHER PROVE IT.

Supporting TRS-80 Models I and III only...

If you're the type of person whose satisfaction is NOT from loading a computer program, BUT from knowing what makes it tick, what makes it work and how to improve it, then you are invited to accept a guaranteed, no risk, money back offer. The Alternate Source Programmer's Journal would like YOU to critique it's contents. A thinking, creative mind like yours is not content to simply accept things the way they are. Just now we are learning to effectively utilize the power inherent in that little gray box of yours. You're looking for newer, faster, and better ways to utilize this power. Thousands of computerists rely on TAS to provide this information; we GUARANTEE their satisfaction. Consider a NO RISK trial subscription today -- if you're unhappy with the first issue, drop us a note and we'll refund your money. We think you'll like it, though, with our easy-to-understand features and dozens of enlightening articles on programming the computer you love. In over two years of publication, only TWO persons have requested their money back. Discover why -- try an issue of The Alternate Source Programmer's Journal today! Twelve issues are $24. Published eight times annually.

(Subscription rates for outside USA differ; please inquire. Thanks.)

Special for LDOS users!

LBOOT allows you to boot double density Mod I diskettes when using LDOS. The one utility needed to make your LDOS & double density TRS-80 a winning team! Our regular price for LBOOT is $19.95, but you can take advantage of this special offer: subscribe to TAS, and get LBOOT for just $10! Limited offer.

( ) I'll try 12 issues of TAS; enclosed is $24.
( ) I'll subscribe and take LBOOT; $34 total.
( ) I'd like LBOOT; $19.95 plus $3 P&H.

Name: ____________________
Address: ____________________
City: ____________________
State, Zip: ____________________

The Alternate Source
704 North Pennsylvania Avenue
Lansing, Michigan 48906

Ph. 517-482-TAS0

Micronet: 70150,255
Source: TCH565

The method used is described in RoySoltoff's description of his new program, PDS, which stands for Partitioned Data Sets, published in the last issue of the LDOS Quarterly.

Due to the construction of the library, it is impossible to SYSRES it as can be done with the other overlays. In any case, the very long length of SYS6 in particular swallows far too much of your RAM. It is possible, however, to put individual routines into a /CMD file. The choice would depend on your activity. For example, if you are doing a great deal of copying of files, then the creation of a command file of the COPY routine might be envisaged.

For those who are interested I shall briefly describe how the library is accessed. Let us assume that you wish to display the directory. You will type in DIR (plus any parameters required) and hit the <ENTER> key. That action will start the long (in number of bytes, but short in time) interpretation of this command at the address 4405 hex. The accumulator (the A register) will be loaded with the value of B3 hex and a RST 28 (*2) will follow.

That will put you down at the address of 28 hex in the ROM where there is a jump instruction to another address in RAM at 400C hex (*3) From there the B3 value in the accumulator will lead to SYS1 being loaded (*4), and DIR being parsed, matching this library command with the list located in SYS1. Once the name DIR is found, the index values following it will be extracted. These are 21 hex and 80 hex. The latter tells the system that the DIR routine is in SYS6 (if it had been SYS7 the value would have been C0 hex). The former value is the index for DIR, located in SYS6.

Another RST28, this time with the value of 88 hex in the accumulator, will load the first sector of SYS6 where the indices for the programs therein are examined until 21 hex is located. SYS6 (and SYS7 too) contain each of the indices for the library routines at the beginning of the file. Each index is followed by the relative load address for the routine, its entry point, and the relative address for loading of the code used to run the routine. Note that this method avoids loading the complete library file in order to reach the routine desired. This speeds things up a great deal over the method used by TRSDOS which loads the complete library, and also permits limiting use of memory to the area between 5200H and 6FFFH. In fact, most routines use much less space.

The LDOS library files have also been optimized in so far as is practical. Based on feedback from users, and onLSI's own experience, the routines most frequently used are placed earlier in the library in order to reduce access time. In the same way, the split between SYS6 and SYS7 was made in such a way so that the most frequently accessed routines are in SYS6. It is frequently asked why TRACE is a library routine since it is rarely used and is not very practical. Since TRACE is a TRSDOS library routine, LSI included it in SYS7 in order to maintain the upward compatibility which is one of the most important features of LDOS.

Footnotes

(*1) In fact, a single density disk has space for 64 files, 8 sectors of 8 files each. However, 16 of these are reserved for system files. Only 14 are used at present, including BOOT and DIR). On a double density disk, there are 112 files (after deduction of the 16 system slots allocated).

(*2) RST is an assembler instruction called a restart of which there are six for the TRS80. The number is the hex value where the processor is sent. One advantage of a restart instruction is that it uses only byte of instructions versus 3 bytes for a normal jump command The RST28 is used to tell the system which overlay is to be loaded into memory.

(*3) It is frequently asked why the instruction at 400C is merely a jump to another vector. Why not jump immediately to the second vector? There are two reasons for this. First of all, it is impossible to modify instructions in ROM. Secondly, LDOS strives to maintain upward compatibility with TRSDOS and maintains all 'standard' vector values. This means that there are frequently jumps to a second (or even a third) vector where execution begins.

(*4) The system subtracts 2 from the value to determine the overlay number to be loaded; the upper 4 bits of the byte tell the system which part of the overlay to go to. In this case, B3 tells the system that SYS1 is required, and that the command is to be parsed. The value of 93, also calling SYS1, would have led to the "LDOS Ready" message.

INSIDE THE EXPANSION INTERFACE by Earle Robinson

Model I LDOS users occasionally encounter problems with the Expansion Interface. As a very non-hardware oriented person myself, I have encountered my full share of these and will discuss them in the following remarks. Some of my own experiences and problems may help others in recognizing what is happening (or often, what is not happening), and more easily overcome or resolve things like silent death, re-boots, and data write errors.

The first annoyance encountered by Mod I users with the Expansion Interface is that of what is called 'Silent Death'. The screen display freezes, and nothing can be entered from the keyboard. Although one may be tempted to throw the whole computer out the window, a more practical solution is to push the Reset button. Silent death happens most frequently during disk I/O, and can be very frustrating if a file or a program is lost. The source of the problem is within the Expansion Interface, and is called time-out. There is insufficient delay accorded to the Seek command due to a too small capacitor, (thanks to Tandy, which in its infinite wisdom, did not wish to be accused of over-engineering their products.) This capacitor will be replaced upon demand at your local RS computer center. However, if you prefer, and are hardware oriented, you may do it yourself.

Unwanted reboots usually happen while in the midst of completing a most arduous task, running, writing or debugging a program. They also typically occur very late in the evening, when you don't have a backup of the file, so that it is lost. The one positive aspect of such a calamity occurring after Midnight is that your loved ones will already be asleep and therefore less likely to be in immediate danger of the overwhelming wrath which will seize you. The usual reason for these re-boots is defective memory above HIGH$. LDOS uses that area for drivers and devices constantly. Even one bad bit somewhere in that memory can be disasterous. Fortunately, memory chips have now become quite cheap, usually around $16 per set of 8. Replace that memory immediately! Nine times out of ten this will solve the problem.

The final disaster that we'll cover is that of data read or write errors. Localizing this problem successfully is much more difficult. The usual causes are: 1) a defective disk controller chip (WD-1771); 2) oxidization of the connector busses 3) bad diskettes 4) disk drives out of alignment. Tandy, with its scrupulous concern to avoid being accused of over-engineering its products bought the series -01 of the controller chip from the manufacturer, Western Digital. This series might be called less charitably, "seconds". Since Tandy also avoided installing a so-called data separator in the system, the problem with the controller chip frequently only becomes evident when a user installs a separator or a double-density board (which also has a data-separator). When I installed my double-density board, at first I could only successfully write in double-density and not in single density. Clearly, my double-density controller was O.K. but not the old single one. I replaced it with a series -02, and that was that.

Another cause of data errors can be the connections between the keyboard unit, the interface and the disk drives. The connector bus on your keyboard and those on the interface were coated with tin, which oxidizes, rather than gold, which doesn't.

Though cleaning with a pencil eraser (the so-called pink pearl treatment) will provide temporary relief, it will also eventually, if repeated too often, remove the tin. The best solution is to get the connectors replated with silver or gold or solder another connector, a gold one to the original. Note that Tandy didn't wish to encourage gold bugs when designing the Model III either, and also repugned gold connectors on that model.

Use of poor quality diskettes is a frequent cause of data errors, clobbered directories, and other dire problems. Use first rate diskettes always. LSI (and this writer) prefer Scotch. Many of my friends condemn Memorex. I know those who swear by Verbatims, and others who swear at them. Dysans have a Rolls Royce reputation... .and a similar price, not really all that justified. For some time I also used Verbatims until I found that the coating seemed to wear off (especially those used with 80 double density tracks). If you only operate under single density, then the choice of diskettes is, I might add, somewhat less critical.

The final cause of data errors is frequently misaligned disk drives. This source of problems is usually left as the last resort since it usually costs a minimum of $35 to get a drive re-aligned. It is usually recommended that a drive be serviced at least once each six months. However, it really depends on how much use the drive gets, the manufacturer.....and luck. I have two drives from the same manufacturer. The first one has been in service for over 2 1/2 years and has never been re-aligned. The second one required service after only 5 months of service. I do recommend, however, that a small fan be placed so as to aerate the power supplies of disk drives for two reasons. First of all, heat build up will cause more wear. More important still, the speed of a drive will vary widely between the time it is turned on and when it gets warm. Naturally, cleaning the head of the drive occasionally is also highly recommended to remove oxidized matter. This can be done with a special diskette kit available from most dealers including Radio Shack stores. Be careful, however. The head is actually NOT on the side from which the door closes, but the opposite side. This is not explained in the documentation of the drive cleaning kits and many people have destroyed the pressure pad by applying the wet cleaning side of the diskette against the pressure pad rather than the head itself!

Finally, a number of people complain that they have more disk I/O problems with LDOS than with brand X operating systems. They forget that LDOS offers type-ahead, the most accurate system clock and a reasonably good software based spooler. None of the other systems can equal these features which disable interrupts much earlier during disk I/O, which eliminates use of type-ahead and the spooler at that time.

AN ARTICLE by Charlie Butler of The Alternate Source

Per Bill's request, I am going to editorialize; carte blanche, one page, he says. Thanks for nothing pal. Your turn is coming. Let's talk about

Changing Operating Systems Mid(bit?)stream -- Should you do it?

Remember the day you first got disk? After getting the cables figured out and determining which way the write-protect notch went you were on the way. Your first goal was to learn the operating system. You probably started with DIR, LIB and FREE, but it would be months (and seem like years) before you were up to the heavies like APPEND and ATTRIB. And now, it's beginning to make sense. Most of it, anyway. Wasn't it all fun?

Quite frankly, it was a pain in the butt. Learning a new DOS slows down productivity. You don't have enough time to do everything now, much less fight with a new set of commands and the compatibility problems that invariably pop up. Why spend weeks of frustration learning a new DOS?

Try this: Get your most important data file. It could be a program you just spent many hours writing or your customer master list. From BASIC, type

```
OPEN,"O",1,"filename"<ENTER>
```

Now say to yourself, "Oh, that's my very important file." Then type

```
CLOSE<ENTER>
```

Depending on which DOS you're using, your file will either be OK, or it will be a null file (a file that contains no information). A close inspection of the directory will reveal that all directory extension pointers have been reset and the file contains no sectors. Recovery will take hours.

Combine this problem with a library of wierd acronyms, disrespect for active drivers when chaining and general incompatibilities in BASIC.

For example, the BASIC "MERGE" and "LOAD" commands. TRSDOS allows you to write BASlC statements in ASCII to a file in any order and corrects that order when LOADing or MERGEing the program. (Many of the new "program generators" or programs that write other programs use this principle.) For illustration, study the following program:

```
10 OPEN"O",1,"TEST:0"
20 PRINT#1,"20 PRINT I"
30 PRINT#1,'10 FOR I = 1 TO 10"
40 PRINT#1,"30 NEXT I"
50 CLOSE
```

Run this program, then RUN"TEST" with your operating system. If only one number prints on the screen, the program lines from the file TEST were not merged properly.

True, the best laid plans go astray, but I (the programmer) have a choice of which operating systems I will support. In order to stay in business, I must support TRSDOS, but any other system must prove its worth.

The above examples are only a couple of the reasons why I have started using LDOS for my programming projects. I don't have to worry about the above problems. But that's not all. What about other "bugs" I discover? I frequently hear customer complaints about the frequency of updates coming from Wisconsin ("Do you have the 2pm or the 4 pm version?"). You should be applauding.

Updates mean someone is listening to complaints and at least trying to correct problems. Most of the updates you probably don't need, anyhow. It is nice to know they are there, if you do.

Lest you think I am trying to hard-sell LDOS, I am not. It is not a cure-all for DOS problems (and they ALL have them). Just looking at their manual shows you they have made a remarkable effort) but it still falls short of "ideal." It is not indexed to suit me and it doesn't include enough user examples. Still, for my applications) LDOS is the best product on the market. It is also the best documented DOS on the market.

Should you hassle with another operating system? Maybe. If you are having problems like mine (above), definitely. I don't envy you; just make sure the DOS you're learning is worthwhile.

Roy's Technical Corner

This is the fourth in my regular series of articles on LDOS technical subjects. The first explained the Data Address Mark convention used in LDOS. The second detailed the functions of the @PARAM vector and how you can ease the development of your assembly language programming efforts by incorporating @PARAM to parse command line parameters. The third dealt with device independence and its implementation on the Models I and III. Commencing with this issue, my series has been named, "Roy's Technical Corner" - and you thought RTC meant Real Time Clock, ha!

One of the many interesting things unique to LDOS amongst the many TRS-80 DOSes is our extensive use of distinct record types in load modules. Load modules, you say? What's a load module? Let's set the record straight on this one and clear up some of the semantics concerning load modules (according to Pete Barbutti, semantics is the study of salmon and ticks}.

A load module is simply a file that contains information on where it is to load into memory. It is usually loaded by the SYSTEM loader. If it can be directly executed as a program, it then becomes known as an executable load module (ELM). The usual term that has been applied to such a file is "CMD". That's because a directly executable load module is thought of as a command. We further use the default file extension of /CMD for these command files (ever wonder where Command File Utility got its name?).

Another problem of semantics arises when we consider the output of the DUMP library command. The default file extension used is /CIM, short for core image. This has been an unfortunate specification because the core image dump is constructed exactly like a load module it, in fact, IS a load module file. The term core-image, assumed originally coined under TRSDOS by Randy Cook, does in fact mean executable load modules on main frame computers. It is generally NOT the case with the TRS-80, although it can be. I have problems with the use of the CIM extension, in light of the extensive use of CMD as a directly executable module. Perhaps we shall change the default in DUMP to something more meaningful.

Let's get back to the issue at hand. A load module has been said to include certain "loading" information pertinent to a loader routine. This is in contrast to a data file, a BASIC program, an ASSEMBLER source file, etc. which contain no such information. Think of the load module as a sequence of

RECORDS. Note that I did not say an ordered sequence. Thus, the implication is that the records do not have to be in an ascending order (contiguous load addresses). Each record stands by itself and can be dealt with by a loader. The records must have some indicator as to what TYPE of record they are. This TYPE code is used to denote a record as a HEADER, a TRANSFER ADDRESS, an ISAM DIRECTORY, a LOAD RECORD, or other meaningful structure. A record must also have a LENGTH which is the length of the data area field. Under TRS-80 operating systems, the length is constrained to a one-byte value and can be from 1-256 in value. The remaining part of the record is its DATA AREA and is used to store program code, directory information, messages, etc. I have identified three different fields of information for the record; TYPE CODE, LENGTH BYTE, DATA AREA. If you are familiar with BASIC random access files, you will see the similarity in the fielding of records - except in this case, we have variable length sequentially accessed records (with partitioned data sets, we also have variable length indexed sequential accessed records).

Let me now build a table of record types used in LDOS. After that is accomplished, I will then fill in the details where necessary.

TYPE	DATA AREA
01	object code (load block)
02	transfer address
04	end of partitioned data set member
05	load module header
06	partitioned data set header
07	patch name header
08	ISAM directory entry
0A	end of ISAM directory
0C	PDS directory entry
0E	end of PDS directory
10	yanked load block
1F	copyright block

Any code above X'1F' is invalid as a record type. In addition, any code not listed in the above table is reserved for future use. For example, codes 11-18 are reserved for relocatable code module structures as defined in the LC implementation. As an aside, Tandy uses a type code of X'03' in lieu of the transfer address type code of X'02' to indicate a load module that is not executable.

Let's look at a sample file. Start by listing the first sector of LBASIC via LIST LBASIC/CMD.BASIC (H). Notice it starts out with:

```
05 06 4C 42 41 53 49 43 1F 32 43 ...
 .  .  L  B  A  S  I  C  .  .  C ...
```

What you have here is a load module header (TYPE=05). The length byte (LENGTH=06) follows the TYPE code. The 6-byte DATA AREA field is the header name. ALL RECORDS FOLLOW THIS "FIELDING" ORDER. A record is organized with a TYPE, LENGTH, DATA sequence. The X'1F' begins the second record. Quick now, what kind is it? It happens to be a copyright record with a LENGTH of X'32' or 50 decimal bytes. Incidentally, the TYPE=1F record is generated automatically by the "COM" pseudo-op in EDAS, the assembler used to maintain LDOS.

Note that each record begins with the TYPE code and the first byte following the end of a record is always the TYPE code of the next record. The only exception is when a TYPE code indicates the end of a file. If you look further in the record displayed at relative position X'3C', or if you count 50 bytes down from the "C" of "Copyright", you will see:

```
01 A1 00 4E C3 8E 5B ...
```

The record TYPE is a load block (TYPE=01), and the length of the data area is X'A1', or 161 data bytes. The two-byte field following the LENGTH is the starting load address for the rest of the field. This is a special case. Since the LENGTH value includes the 2-byte load address, a length of x'03' would indicate only one load byte. A length of x'04' would indicate two load bytes. A length of X'FF' would indicate 253 load bytes. A length of X'00' would indicate 254 load bytes. To be able to have a data area with up to 256 bytes of loadable data, the LENGTH values of x'01' and X'02' are indicative of 255 and 256 load bytes respectfully. This is accomplished by having the system loader decrement the length value by two when reading a load address. The resultant value becomes the true length of theloadable data.

If you let the BASIC listing proceed to the end of the file, the last four bytes should appear as:

```
02 02 C9 52
```

This will represent the TRANSFER ADDRESS record (TYPE=02). Again, we have a LENGTH byte which shows a 2-byte data field. The data field contains the transfer address or entry point to the program in standard low-order, high-order sequence. Of course, the transfer address value you will see is dependent on which LBASIC you are listing.

These are the four types you will find in other operating systems. Now it's time to get down to thenitty gritty. List the first record of SYS6/SYS via LIST SYS6/SYS.SYSTEM (H) {note: if you are not using 5.1.1, then the password to use is WOLVES}. Look at the area just past the copyright. You will see something like this:

```
08 06 1E 00 52 0000 DB 08 06 21 ...
```

The TYPE code of X'08' indicates an ISAM DIRECTORY RECORD. The LENGTH byte denotes a DATA area of six bytes. After the sixth byte, you will see another TYPE=08 starting another ISAM directory record. SYS6 is a partioned data set. The TYPE=08 records are its directory.

In LDOS, the directory data area is used by the SYSTEM loader to locate where a particular member can be found in the file. The data area has sub fields as follows. The first byte (1E) is the ISAM entry number. This value is provided to the SYSTEM loader by the SYS1 command line parsing routine upon the discovery of a library command request. The entry number is the PDS member that will execute your request. The SYSTEM loader will search the PDS directory for a match. The next two-byte sub-field is the transfer address of the member. The transfer address is contained in the directory so that more than one transfer address can be applied to a member (i.e. a member can have multiple entry points).

The three-byte field remaining is the triad pointer which points to the first byte of the member. The triad pointer is composed of the Next Record Number (NRN) and Relative Byte Offset for the member's first byte. Consult the LDOS Technical Reference Guide File Control Block section for more information on these values. Thus you have six bytes of data as specified by the LENGTH byte. In the PDS utility offered by MISOSYS, the ISAM directory record has a length of nine because it includes a three-byte subfield which contains the TRUE length of the member. That piece of information is needed in many PDS commands.

While you are looking at the first sector of 5Y56, proceed to the first byte following the last ISAM directory record. You will observe the sequence:

```
0A 01 00 04 01 00 01 02 00 52 ...
```

The TYPE=0A indicates that it is the end of a PDS directory. The SYSTEM loader uses this to discover that the requested ISAM entry happens to not be in the file being examined. If it gets to the TYPE=0A byte without a match on the ISAM number, the member is not in the directory. The LENGTH=01 is needed because ALL load module records MUST have a length byte. The DATA area contains only a single byte) X'00' We cannot indicate a null record because a length byte of X'00' indicates 256 data area bytes. Thus) the X'0A' record type must have a minimum of one byte in its data area.

The record following is a TYPE=04 to indicate the end of a PDS member. This record serves but one purpose when used immediately following the directory - it will result in a load file format error if you attempt to execute SYS6 or SYS7 as if they were CMD files. When not expecting a partitioned data set file) the SYSTEM loader will ignore record types other than X'01' and X'02' except for the X'04'. The file reading will terminate at the X'04' with the above-mentioned error message.

The record type X'04' is usually used at the end of a partitioned data set member. If you list through SYS6, you will discover that each member ends with "04 01 00". LDOS uses this code in lieu of the transfer address code because the SYSTEM loader needs to take action different from that when a standard load file has been completely loaded. Also, the transfer address for the member is stored in the ISAM directory itself.

The next record type to discuss is that used in a PDS MEMBER DIRECTORY. If you have purchased the PDS utility from MISOSYS, list it in hex. Notice that it starts with X'06' in lieu of an X'05' which is the normal header type for a load module. Well, PDS uses the X'06' in certain PDS commands to note whether the target file is a partitioned data set compatible with PDS utilities. There is a bit set in the LDOS system directory to indicate a file is a PDS; however that is to be used in a future release of PDS.

If you list past the front end loader, you will see the start of the PDS MEMBER DIRECTORY at relative 0001:14. It reads as follows:

```
0C 0B 64 69 72 20 20 20 20 20 01 01 7A 0C ...
 .  .  d  i  r                 .  .  z  . ...
```

The TYPE=0C indicates a PDS member directory record. The LENGTH byte specifies that the data area is an 11-byte field.

The DATA AREA is subfielded as an 8-byte member name (in lower case), a one-byte ISAM number that is used to match up with a corresponding ISAM directory record) and a 2-byte field of member data. The first byte uses bit position 7 to indicate a data member in contrast to an executable CMD program. Bit positions 4-6 are reserved for future use. Bits 0-3 and the next byte contain the date that the member was added to the PDS and is in a format identical to that explained as DIR+1 and DIR+2 in the DIRECTORY RECORD section of the LDOS Technical Reference manual. As you look through the PDS member directory, you will get to the "0E 01 00" record which indicates the end of the MEMBER directory. The front end loader uses this to note whether the requested member is in the PDS. The ISAM directory follows.

One last little one to wrap up is the record types associated with the PATCH utility. When you apply an X-patch to a file, the name of the patch file is used as a header name with a record type of X'07'. Thus, if you want to YANK the patch, the PATCH program can read through the file and search for a like-named header. If a matching header is found, PATCH will proceed as follows. Since it may be impossible to remove the patch without bubbling up any code blocks following the patch (another patch maybe?), PATCH will change the TYPE=01 records to TYPE=10 records. The TYPE=l0 records will not be loaded by the SYSTEM loader but will be considered as non loadable comment records. It is thus possible to Un-yank" a yanked patch; however, this feature is not implemented in the PATCH utility.

There we have it, the relatively complete explanation for load module format records. Send in your requests for the nextissues's RTC column. Here is where you can obtain detailed information on the technical aspects of LDOS. Regards until next time. By the way, this entire article was composed using LED, the LDOS Text Editor, and the latest in professional products from Logical Systems, Inc. After all text editing was done and a print copy was needed, LSCRIPT was used to print the document.

THE JCL CORNER by Chuck

This month's column will deal with a variety of topics. The first is the announcement of two new JCL features available with the 5.1.2 update release. JCL will now flush the typeahead buffer before exiting. This means that all <ENTERs> pressed in response to a //PAUSE or other macro will not be passed back to the system when the JCL terminates.

A new compilation symbol, the percent sign (%), has been designated and allows passing values directly to the system as though they came from the keyboard. The values should be represented as hexadecimal digits. Any JCL file using the % symbol must be compiled, or JCL will abort with a "Bad Format" error upon encountering the line. Also, the character values must be those normally accepted by the @KEYIN ROM routine. The following example will clear the screen and print the comment line:

```
   %1F. This is a cleared screen comment
```

As you can see, the characters are inserted directly in the line - symbol and value, symbol and value, and then the period followed by the comment string. DO NOT put them on a separate line ending with a carriage return. Doing so will clear the screen and then output the carriage return, generally aborting the JCL. This same format holds true when % is used with execution macros.

```
%1F//pause press <ENTER> to continue
%1F//flash 10 Starting up job stream
```

There are some limitations to the use of this new symbol. It is not possible to pass values to the MiniDOS or KSM filters using this new symbol. However, programs that do utilize the @KEYIN call for keyboard input should be able to respond to these characters.

There were three compilation macros that were not discussed in the last Quarterly; //. COMMENT, //QUIT, and //INCLUDE. The first two are very easy to understand. The //. COMMENT will allow a message to be displayed during the compile phase. Unlike an execution comment, the compilation comment will not be written to the SYSTEM/JCL file. One use for the compilation comment is to check logic statements as a JCL is compiling. For example:

```
.TEST/JCL
//if A
//. A was true
LBASIC RUN"TEST/BAS"
//stop
//else
//. A was not true
//end
```

Consider this JCL file as a piece of a large procedure. As the file is being compiled, the //. COMMENT lines will be displayed. Thus you will know the status of this section of JCL logic before execution begins.

The //QUIT macro is used, as its name would imply, to abort the JCL compiling procedure. It is primarily used when it is absolutely necessary that certain tokens be specified upon entering a JCL procedure. For example:

```
.TEST1/JCL
//if -s+-d
//.    You MUST enter both S and D
//quit
//end
backup :#s# :#d#
//exit
```

The //if statement (if not S or not d) tests to see that both tokens s and d have a logical true value. If not, the comment is displayed and the compilation aborts. The end result of aborting the JCL procedure could also be accomplished by using the //ABORT execution macro instead of the //QUIT. In that case, however, the entire JCL file would be compiled first, and the abort would actually come during the execution phase.

Before we discuss the //INCLUDE macro in detail, one important point needs to be stressed. An //INCLUDE macro CANNOT END A JCL FILE A JCL file ending with an //INCLUDE will produce a "Record number out of range" error.

The //INCLUDE macro is used to select other JCL files from disk to be included in the compilation of a JCL procedure. For example:

```
.TEST2/JCL
//if X1
//INCLUDE test3:0
//else
//INCLUDE test4:1
//end
```

This example tests the token X1. If X1 is true, the file TEST3/JCL will be read in from drive 0, and its contents compiled and written to the SYSTEM/JCL file. As the example shows, the default extension for included files is /JCL. If X1 is not true, the file TEST4/JCL:1 will be read in and compiled. It is not necessary to use the //INCLUDE inside an //IF. An //INCLUDE can be used anywhere in a JCL file.

//INCLUDE's can be nested up to a level of 10 - that is, an //INCLUDEd file can contain //INCLUDE statements. Refer to the following example, taken from the 5.1.1 (or later) manual, page 5 - 8.

```
File #1 =>   //. NEST0/JCL
             . nested procedure example
             //INCLUDE nest1
             . this is the end of the primary JCL
             //EXIT

File #2 =>   //. NEST1/JCL
             . this is the first nest
             //INCLUDE nest2
             . this is the end of the first nest

File #3=>    //. NEST2/JCL
             . this is the second nest
```

The above will result in a nest level of two (two pending //INCLUDEs). If these three JCL files are saved as NEST0/JCL, NEST1/JCL, and NEST2/JCL, and the NEST0/JCL is compiled and executed, it will result in the following dialogue:

```
//. NEST0/JCL
//. NEST1/JCL
//. NEST2/JCL
. nested procedure example
. this is the first nest
. this is the second nest
. this is the end of the first nest
. this is the end of the primary JCL
```

The three compilation comments will be shown immediately as the JCL file is compiled. When the compilation phase is complete, the compiled SYSTEM/JCL file will be executed. In this example, the execution phase will merely display a series of execution comments. As you can see from the order of the displayed comments, the files are executed similarly to nested FOR-NEXT loops in BASIC. After all //INCLUDEs are detected, the innermost (last encountered) //INCLUDE file completes execution first, with execution proceeding back towards the original //INCLUDE.

The //INCLUDE macro can very easily be used to compile a large JCL procedure from a series of smaller JCL routines. If the finished SYSTEM/JCL file is a procedure that will be executed many times, it may easily be saved by copying SYSTEM/JCL to a file with another name.

Even after last issue's column, there have been questions about the proper structuring of the //IF and //END statements. The proper use of //IF requires one //END for each //IF. What might not be readily apparent when nesting //IF's is that the last //END corresponds to the 1st //IF. For example:

```
   First IF      //IF conditional #1             Line #1
                 first block of lines            Line #2
   Second IF     //IF conditional #2             Line #3
                 second block of lines           Line #4
                 //ELSE                          Line #5
   Third IF      //IF conditional #3             Line #6
                 third block of lines            Line #7
                 //END (ends third IF)           Line #8
                 //END (ends second IF)          Line #9
                 fourth block of lines           Line#10
                 //END (ends first IF)           Line#11
```

Evaluating this example produces the following results, When the first //IF is false, the entire block between the first //IF and its corresponding //END are not compiled. Thus, none of the lines in this example would get written to the SYSTEM/JCL file.

Assuming from this point on that the first //IF is true, evaluating the results of the second and third //IF's would produce the following.

When the second IF is true, Line #4 would be written to SYSTEM/JCL file. Lines 5 through 8, the //ELSE conditional, would be ignored. Line #10 would also be written, as it comes after the //END statement corresponding to the second //IF.

If the second //IF was false, the third //IF in the //ELSE block would be evaluated. If it was true, Line #7 would be written to the SYSTEM/JCL file. Whether or not this third //IF is true, Line #10 will be written out.

As long as the first //IF is true, Line #10 will always be written to the SYSTEM/JCL file regardless of the logical evaluation of the second and third //IF macros. This is because it comes after the //END macros for both of those //IFs. AlS,Q, you can see that the first //END in Line #8 corresponds to the third //IF in Line #6, the second //END to the second //IF, and the third //END to the first //IF.

Next issue's column will probably deal with the keyboard macros, and interfacing JCL with applications programs. If you readers have any subject you want to see discussed in The JCL Corner, drop a letter in the nearest mail box, addressed to Customer Support, Attn: Chuck.

P.S. The CMDFILE utility can be totally controlled by a JCL file. Try creating a JCL, the first line being CMDFILE, and the next lines being the proper responses for the CMDFILE prompts.

PROGRAMS FROM USERS

This is a directory mapping program from Bill Fields. Besides printing a directory map, it lists the files in alphabetical order, along with the grans they occupy.

```
20 REM GIVE BACK ANY STRING SPACE SO THAT MEM IS ACCURATE
40 CLEAR 0
60 REM I% IS A FACTOR REPRESENTING THE SPACE REQUIRED FOR ALL
80 REM NON-STRING VARIABLES AND STRING ARRAY OVERHEAD
100 REM IF OUT OF MEMORY THEN INCREASE I%. IF TOO MUCH
120 REM GARBAGE COLLECTION, THEN DECREASE I%.  THE SCREEN
140 REM WILL ALWAYS HAVE A CHANGING DISPLAY THAT INDICATES
160 REM (BY STOPPING) THAT GARBAGE COLLECTION IS IN PROGRESS.
180 I%=5300
200 CLEAR MEM-I%
220 REM ***************************************************
240 REM *                                                 *
260 REM * -----> DISK MAPPER FOR LDOS FORMATTED DISKS <----- *
280 REM *                 03/05/82 VERSION                *
300 REM *          WRITTEN BY BILL FIELDS                 *
320 REM *                      POST OFFICE BOX 1120       *
340 REM *                      GLENDALE HEIGHTS,ILL 61037 *
360 REM *                                                 *
380 REM * NON-COPYWRITED CODE.  PUBLIC DOMAIN AS LONG AS  *
400 REM * AUTHOR'S NAME AND ADDRESS ARE NOT REMOVED OR    *
420 REM * ALTERED. NOT FOR SALE! THE AUTHOR WOULD WELCOME *
440 REM * ANY IMPROVEMENTS OR ENHANCEMENTS TO EITHER      *
460 REM * PERFORMANCE OR FUNCTION. ALSO REPORT BUGS,PLEASE.  *
480 REM *                                                 *
500 REM * DIRECTORY STRUCTURE ALLOWS FOR EIGHT GRANULES PER  *
520 REM * CYLINDER (1-8) & 96 CYLINDERS PER DISKETTE (0-95). *
540 REM * IF THE DISKETTE IS DOUBLE DENSITY THEN EACH     *
560 REM * GRANULE IS SIX SECTORS AND IF IT IS SINGLE      *
580 REM * DENSITY EACH GRANULE IS FIVE SECTORS.  A SECTOR IS *
600 REM * ALWAYS 256 BYTES.                               *
620 REM ***************************************************
640 REM FOLLOWING DIMENSION STATEMENT PUTS VARIABLES INTO
660 REM BASIC1S SYMBOL IN THE ORDER FROM MOST USED TO LEAST.
680 REM COMPUTING THE ORDER OF VARIABLES USE IS DONE BY FASTER
700 REM WHICH IS AVAILABLE FROM PROSOFT POST OFFICE BOX 839
720 REM NORTH HOLLYWOOD, CALIF 916$3
740     DIM    I%,T%,D%,J%,P%,SF%,W%,IK$,E%,DS$,PN$,G%,GC%,S%,Y8%,TC%,EE$,F1$,
Y9$,B4%,TX%,B7%,P1$,P2$,B0%,B1%,B2%,B3%,U%,Y1%,SP%,B5%,B6%,DN$,K1!,ZZ%,F2$
760 CLS
780 REM BIT MASKS FOR BIT0 THRU BIT7
800 B0%=&H01
820 B1%=&H02
840 B2%=&H04
860 B3%=&H08
880 B4%=&H10
900 BS%=&H20
920 B6%=&H40
940 B7%=&H80
960 PRINT @ 260,"LDOS FORMATTED DISK MAPPER"
```

```
980 PRINT @835,"USE WHICH DRIVE?  ";
1000 FOR T%=1TO5
1020   DN$=INKEY$:IFDN$=""OR(DN$<>""AND(DN$<"0"ORDN$>"7"))THEN T%=1ELSET%=6
l040 NEXTT%
1060 PRINT @835,"                    ";
1080 IF DN$="0"THEN 1140
1100 PRINT @ 320,"PLACE  LDOS  FORMATTED DISK  IN  DRIVE "DN$"  <PRESS  ENTER>
  ";
1120 GOSUB 4620
1140 PRINT  @320,"Written by Bill  Fields P  0 Box  1120 Glendale Heights, Ill
60137"
1160 REM EACH DIRECTORY ENTRY IS 32 BYTES IN LENGTH
1180 REM SO OPEN DIRECTORY WITH LOGICAL RECORD LENGTH OF 32
1200 OPEN "R",1"DIR/SYS:"+DN$,32
1220 FIELD 1,32 AS F1$
1240 PRINT @835,"Reading GAT                                ";
1260 REM GET 7TH 32 BYTE RECORD
1280 REM THIS IS X'C0'-X'DF' OF THE GAT RECORD
1300 GET 1,7
1320 REM GAT+X'CB' = VERSION OF SYSTEM UNDER WHICH THE DISKETTE
1340 REM WAS FORMATTED. IF OTHER THAN LDOS5.0 OR LDOS5.1 THEN
1360 REM WARN THAT UNPREDICTABLE RESULTS MAY OCCUR
1380 IF (ASC(MID$(F1$,12,1))  <>  (&H50)) AND (ASC(MID$(F1$,12,1))  <> (&H51))
THEN PRINT  @ 643,"** WARNING ** NON-LDOS DISKETTE - UNPREDICTABLE RESULTS MAY
OCCUR"
1400 REM GAT+X'CC' HAS DISKETTE CYLINDER COUNT EXPRESSED AS
1420 REM EXCESS OF 35 CYLINDERS. TC WILL HAVE TOTAL CYLINDERS.
1440 TC% = ASC(MID$(F1$,13,1)) + 35
1460 REM IF CYLINDER COUNT OF DISKETTE IS LESS THAN 35 THEN
1480 REM THE CYLINDER COUNT AT GAT + X'CC' WILL BE NEGATIVE
1500 REM LIKE, FOR INSTANCE, X'E2' FOR A 5 TRACK DISKETTE
1520 REM THEREFORE,IF CYLINDER COUNT IS > 255 AND < 291
1540 REM THEN ADJUST IT DOWNWARD.
1560 IF (TC%>255 AND TC%<291) THEN TC%=TC%-256
1580 REM IF GAT+X'CD' HAS BIT 5 = 1 THEN DOUBLE SIDED MEDIA,
1600 REM ELSE SINGLE SIDED MEDIA
1620 IF ASC(MID$(F1$,14,1)) AND B5% THEN S% = 2 ELSE S% = 1
1640 REM IF GAT+X'CD' HAS BIT 6 = 1 THEN DOUBLE DENSITY,
1660 REM ELSE SINGLE DENSITY.
1680 REM GC=GRANULES PER SIDE OF CYLINDER (TRACK) OF A
1700 REM DISKETTE. IF DOUBLE DENSITY THEN GC=3, IF SINGLE
1720 REM DENSITY THEN GC=2.
1740 IF ASC(MID$(F1$,14,1)) AND B6% THEN GC% = 3 ELSE GC% = 2
1760 REM IN CONSIDERATION OF OUR SMALL STORAGE FRIENDS,
1780 REM DIMENSION ONLY WHATS NEEDED.
1800 DIM X$(TC%-1,GC%*S%)
1820 PRINT @ 835,"Initializing for "TC%" cylinder diskette with"S%"side(s)";
1840 FOR I%--0TOTC%-1
1860   FOR J%=1TOGC%*S%
1880     X$(I%,J%)="****empty****"
1900   NEXT J%
1920 NEXT I%
1940 REM DISKETTE NAME AT GAT+X'D0'
1960 DN$ = MID$(F1$,17,8)
1980 REM GET NUMBER OF ENTRIES IN DIRECTORY
```

```
2000 ZZ% = LOF(1)
2020 PRINT @771,"total 32 byte records in directory =";ZZ%
2040 REM SET CYLINDERS PROCESSED (FOR LOCKOUT) TO ZERO
2060 TX% = 0
2080 REM NOW SCAN TRACK LOCKED OUT INFO
2100 FOR I% = 4 TO 6
2120   REM WHEN I=4 GET GAT+X'60'-GAT+X'7F'
2140   REM WHEN I=5 GET GAT+X'80'-GAT+X'9F'
2160   REM WHEN I=6 GET GAT+X'A0'-GAT+X'BF'
2180   PRINT @835,"Reading GAT                                  ";
2200   GOSUB 4560
2220   GET 1,I%
2240   REM EACH BYTE IS A CYLINDER. IF = X'FF' THEN LOCKOUT
2260   PRINT @835,"Processing cylinder lockout bytes in GAT     ";
2280   FOR J% = 1 TO 32
2300     REM INCREMENT CYLINDER COUNT
2320     TX% - TX% + 1
2340     REM IF ALL BITS ARE ONE THEN SETUP AS LOCKED OUT
2360     IF ASC(MID$(F1$,J%,1))  = B0%+B1%+B2%+B3%+B4%+B5%+B6%+B7% THEN FOR D%
= 1 TO 8 :  X$((I%-4)*32+J%-1,D%) = "*LOCKED OUT*" : NEXT D%
2380     IF TX% = TC% THEN J%=33:I%=7
2400   NEXT J%
2420 NEXT I%
2440 REM Y1 = AVAILABLE GRANULES ON DISKETTE
2460 Yl% = 0
2480 REM SET CYLINDERS PROCESSED (FOR ALLOCATION) TO ZERO
2500 TX% = 0
2520 FOR I% = 1 TO 3
2540   REM IF I=1 THEN PROCESSING GAT+X'00'-GAT+X'1F'
2560   REM IF I=2 THEN PROCESSING GAT+X'20"-GAT+X'3F'
2580   REM IF I=3 THEN PROCESSING GAT+X'40'-GAT+X'5F'
2600   PRINT @835,"Reading GAT                                  ";
2620   GOSUB 4560
2640   GET 1,I%
2660   PRINT @835,"Processing granule allocation bytes in GAT    "
2680   FOR J% = 1 TO 32
2700     TX%=TX%+1
2720     FOR D%=1TOGC%*S%
2740      E%=l
2760       IF X$((((I%-1)*32)+J%-1),D%)="*LOCKED OUT*"THEN E%=0
2780 REM   THE BRACKET IN THE NEXT STATEMENT IS REALLY AN ARROW
2800 REM   THANK YOU MX-80!!!!!
2820       IF NOT ASC(MID$(F1$,J%,1)) AND (2[(D%-1)) THEN Y1%=Y1%+1:E%=0
2840       IF E%=1 THEN X$((((I%-1)*32)+J%-1),D%)="==>ERROR<==="
2860     NEXT D%
2880     IF TC%=TX% THEN I%=4:J%=33
2900   NEXT J%
2920 NEXT I%
2940 FIELD1,1 AS Y9$,4 AS F1$,8 AS P1$,3 AS P2$,6 AS F2$,10 AS EE$
2960 REM K1=FREE SPACE ON DISK IN K
2980 IF GC%=3 THEN D%=6 ELSE D%=5
3000 K1! = Y1%*(D%*.25)
3020 REM RECORDS 01-08 ARE GAT RECORD, AND
3040 REM RECORDS 09-16 ARE HIT RECORD, SO
3060 REM THEREFORE START AT THE 17TH 32 BYTE RECORD
```

```
3080 FOR Y8% = 17 TO ZZ%
3100   PRINT @835,"reading         directory entry at record          ";Y8%;
3120   GET1,Y8%
3140   IF (ASC(Y9$) AND B4%) = 0 OR (ASC(Y9$) AND B7%) = 128 THEN 3580
3160   PRINT @835,"Processing FPDE directory entry at record          ";Y8%;
3180   IF P2$ <> "   " THEN PN$ = LEFT$(P1$,INSTR(P1$+" "," ")-1) + "/" + P2$
ELSE PN$ = P1$
3200 IF LEN(PN$)<>12 THEN PN$=PN$+STRING$(12-LEN(PN$)," ")
3220   FOR J% = 1 TO 10 STEP 2
3240     T%=ASC(MID$(EE$,J%,1)) : W%=ASC(MID$(EE$,J%+1,1))
3260     IF T% = 255 AND W% = 255 THEN 3580
3280     IF T% <> 254 GOSUB 4360 : GOTO 3560
3300     E% = W% AND (B7% + B6% + B5%)
3320     D% = W% AND (B0% + B1% + B2% + B3% + B4%)
3340     E%=E%/32
3360     D% = (D%+2)*8+E%+1
3380     PRINT @835,"Processing FXDE directory entry at record          ";D%
3400     GET1,D%
3420     FOR D%=1TO10STEP2
3440       E%=0
3460       T%=ASC(MID$(EE$,D%,1)):W%=ASC(MID$(EE$,D%+1,1))
3480       IF T%=255 AND W%=255 THEN E%=1
3500       IF T% <> 254 AND E% = 0 THEN GOSUB 4360 : E%=1
3520       IF E%<>1 THEN GOTO3300
3540     NEXT D%
3560   NEXT J%
3580 NEXT Y8%
3600 CLS
3620 CLOSE
3660 IF (PEEK(&H37E8) AND (B7%+B6%+B5%+B4%))=48 THEN P%=1 ELSE P%=0
3680 IK$=""
3700 SP%=0: TC%=TC%-1
3720 IF GC%*S%>3 THEN SF%=4 ELSE SF%=10
3740 IF P%=1 THEN LPRINT "Map of diskette by track/gran " ELSE PRINT  "Map of
diskette by track/gran"
3760 IF P%=0 AND IK$<>"" THENCLS
3780 IF GC%=2 THEN DS$="Sdns" ELSE DS$="Ddns"
3800 IF  P%=1 THEN  LPRINT "Diskette "DN$" Tracks" TC%+1"  free  "Y1%"g "K1! "k
Sides="S%" Dens=";DS$ ELSE PRINT  "Diskette  "DN$" Tracks "TC%+1" free  "Y1%"g
"K1! "k "S%" sides "DS$
3820 IF P%=1 THEN LPRINT "Trk "; ELSE PRINT "Trk ";
3840 FOR D%=1 TO GC%*S%
3860 IF P%=1 THEN LPRINT "Granule "D%"  ";ELSE PRINT "Granule "D%"  ";
3880 NEXT D%
3900 IF P%=1 THEN LPRINT ELSE PRINT
3920 FOR I%=SP%TOSP%+SF%
3940   IF I%>TC% AND P%=0 THEN 4120 ELSE IF I%>TC% AND P%=1 THEN4720
3960   IF P%=1 THEN LPRINT USING "##";I%;ELSE PRINT USING "##";I%;
3980   FOR J% = 1 TO GC%*S%
4000     IF  P%=1   THEN  LPRINT  USING "  %          %";X$(I%,J%);ELSE PRINT
USING " %          %";X$(I%,J%);
4020   NEXT J%
4040   IF P%=1 THEN LPRINT ELSE PRINT
4060 NEXT I%
4080 IF P%=1 THEN SP%=SP%+SF%+1:GOTO3920
```

```
4100 IF SF%=4 THEN SF%=5
4120 U%=0:D%=0
4140 IF SP%<TC%-SF% THEN D%=-1:PRINT @960,"D = Scroll down";
4160 IF SP%>0 THEN U%=-1:PRINT@976,"U = Scroll up";
4180 PRINT @989,"<enter>=end";
4200 PRINT @1000," M = Map by file";
4220 IK$=INKEY$:IF IK$=""THEN 4220
4240 IF (IK$="U" OR IK$="u") AND U% THEN SP%=SP%-SF%:GOTO3760
4260 IF (IK$="D" OR IK$="d") AND D% THEN SP%=SP%+SF%:GOTO3760
4280 IF IK$=CHR$(13) THEN END
4300 IF IK$="M" OR IK$="m" THEN 4720
4320 GOTO4220
4340 REM record location of file's extent
4360 G% = W% AND (B0% + B1% + B2% + B3% + B4%)
4380 G%=G%+1
4400 W%=INT(W%/32)+1
4420 IF  LEFT$(X$(T%,W%),1)<>"="THEN PRINT @899,"Directory error in ";PN$;
"TRACK="T%;"SECTOR="W%" press <enter> to continue"; :GOSUB4620
4430 PRINT @899,STRING$(76," ")
4440 X$(T%,W%)=PN$
4460 G%=G%-1
4480 IF G%=0 RETURN
4500 W%=W%+1
4520 IF W%>GC%*S% THEN W%=1:T%=T%+1
4540 GOTO4420
4560 FOR T%=1TO100
4580 NEXT T%
4600 RETURN
4620 FORA%=1TO5
4640    IK$=INKEY$
4660    IF IK$ = CHR$(13) THEN A%=6 ELSE A%=1
4680 NEXT A%
4700 RETURN
4720 I%=-1
4740 IF P%=1 THEN LPRINT CHR$(12)
4760 CLS
4780 DIM S$(GC%*S%*(TC%+1)-1)
4800 FOR D%=0TOTC%
4820   FOR T%=1TOGC%*S%
4840     I%=I%+1
4860     IF LEN(STR$(D%))=2 THEN DS$=STR$(D%) ELSE DS$=RIGHT$(STR$(D%),2)
4880     IF LEN(STR$(T%))=2 THEN PN$=STR$(T%) ELSE PN$=RIGHT$(STR$(T%),2)
4900     S$(I%)=X$(D%,T%)+DS$+PN$
4920     PRINT @835,"PREPARING TRACK "D%;"  SECTOR"T%;"INTO RECORD "I%" FOR
SORT"
4940     X$(D%,T%)=""
4960   NEXT T%
4980 NEXT D%
5000 PRINT @835,"Sorting ......
5020 CMD "O",GC%*S%*(TC%+1),S$(0)
5040 IF P%=1 THEN PRINT@835,"OUTPUTTING";
5060 DS$=""
5080  IF P%=1 THEN LPRINT:LPRINT:LPRINT "MAP OF DISKETTE "DN$" BY FILENAME"
:ELSE CLS:PRINT "MAP OF DISKETTE "DN$" BY FILENAME"
5100 SF%=12
```

```
5120 FOR I%=0TOGC%*S%*(TC%+1)-1
5140    IF  ((I%-(INT(I%/SF%)*SF%)=0)  AND  (P%=1)  AND  (I%=0))  THEN  LPRINT
"FILENAME   TRACK GRANULE"  ELSE IF ((I%-(INT(I%/SF%)*SF%)=0) AND (P%=0)) THEN
PRINT "FILENAME    TRACK GRANULE"
5160   IF MID$(S$(I%),1,12)<>DS$ THEN E%=2 ELSE E%=1
5180   ON E% GOTO 5200,5240
5200   IF P%=1 THEN LPRINT "             ";ELSE PRINT "             ";
5220   GOTO5260
5240   IF  P%=1 THEN LPRINT MID$(S$(I%),1,12)" ";ELSE PRINT MID$(S$(I%),1,12)"
";
5260   DS$=MID$(S$(I%),1,12)
5280    IF  P%=1  THEN LPRINT  MID$(S$(I%),13,2);"    ";MID$(S$(I%),15,2) ELSE
PRINT MID$(S$(I%),13,2);"    ";MID$(S$(I%),15,2)
5300 IF P%=1 THEN 5540
5320 IF INT((I%+1)-(INT((I%+1)/SF%)*SF%))<>0 THEN5520
5340 D%=$:U%=0
5360  IF  I%<INT((GC%*S%*(TC%+1)-1)/SF%)*SF%  THEN   D%=-1 :PRINT@960,"D=SCROLL
DOWN";
5380 IF I%=>SF% THEN U%=-1:PRINT@976,"U=SCROLL UP";
5400 PRINT @989,"<ENTER>=END";
5420 IK$=INKEY$: IFIK$=""THEN 5420
5440 IF  (IK$="U" OR IK$="u")  AND U% THEN I%=I%-SF%*2:DS$="":CLS:IFI%<0  THEN
I%=-1: GOTO5520ELSE GOTO5520
5460 IF (IK$="D" OR IK$="d") AND D% THEN :DS$="":CLS:GOTO5520
5480 IF IK$=CHR$(13) THEN END
5500 GOTO5420
5520 IF I%=GC%*S%*(TC%+1)-1 THEN I%-INT(I%/SF%)*SF%:GOTO5340
5540 NEXT I%
```

This section of the newsletter contains some programs that require the use of the BINHEX/BAS program to convert them to executable binary files.

Note that the hex program listings must be put into the proper load module format before they can be used. To use the program, follow these steps:

1) Use Scripsit, LED, or the BUILD command to create an ASCII file containing the hex code. DO NOT leave spaces between the hex characters - the spaces were put in for readability only!. The ASCII files must NOT contain more than 254 characters (127 byte pairs) per line.

2) Enter LBasic, and Run the program BINHEX/BAS, choosing the "Hex to Binary" mode.

BINHEX/BAS

Following is the listing for the LBasic program, contributed by Tim Mann.

```
10 REM -- Hex to binary/Binary to hex file converter
20 REM -- Tim Mann
30 CLS:PRINT:PRINT"Hex to binary/Binary to hex"
35 PRINT"     file converter":PRINT
40 CLEAR 5000
50 GOSUB 58000
100 PRINT "Type 1 to convert a binary file to hex"
110 PRINT "     2 to convert a hex file to binary"
120 PRINT:INPUT D
130 PRINT
140 ON D GOTO 400,200
150 GOTO 100
200 LINE INPUT "Hex file name: ";HF$
210 LINE INPUT "Binary file name: ";BF$
220 OPEN"I",1,HF$
230 OPEN"O",2,BF$
240 IF EOF(1) THEN 320
250 LINE INPUT#1,D$
255 IF D$="" OR D$="OK" THEN 240
260 FOR I=1 TO LEN(D$) STEP 2
270    PRINT#2,CHR$(FND2(MID$(D$,I,2)));
300 NEXT I
310 GOTO 240
320 CLOSE
330 PRINT:PRINT"Done":PRINT
340 GOTO 100
400 LINE INPUT "Binary file name: ";BF$
410 LINE INPUT "Hex file name: ";HF$
420 OPEN"RO",1,BF$,1
430 OPEN"O",2,HF$
440 FIELD 1,1 AS F$
450 FOR I=1 TO 30
455    IF EOF(1) THEN 505
460    GET 1
470    PRINT#2,FNH2$(ASC(F$));
480 NEXT I
490 PRINT#2,
500 GOTO 450
505 PRINT#2,
510 CLOSE
520 PRINT:PRINT"Done":PRINT
530 GOTO 100
58000 DEF FNH1$(X)=MID$("0l23456789ABCDEF",(X AND 15)+1,1)
580l0 DEF FNH2$(X)=FNH1$(X/16)+FNH1$(X)
58040 DEF FND1(X$)=INSTR("123456789ABCDEF",LEFT$(X$,1))
58050 DEF FND2(X$)=FND1(RIGHT$(X$,1))+16*FND1(RIGHT$(X$,2))
58070 RETURN
60000 END
```

DELETE/CMD: A MultiDle-KILL Utility by Renato B. Reyes Ph.D.

LDOS has two commands for deleting files from disk, ie., KILL and PURGE.

KILL deletes single files, while PURGE deletes several files from the same disk, either automatically (using partspecs and/or wildcard specifications) or under user control. Between these two library commands, however, there exists a wide gap.

I have often found it necessary to kill several files at once, without being able to use the PURGE (Q=N) command. Sometimes the filespecs bear no resemblance to each other, which prevents the use of wildcards or partspecs, or, worse, they bear too much of a resemblance to other files on the disk. And going through a manual PURGE when you have seventy or eighty files on a disk can be time consuming. The only alternative was to KILL the files one by one.

On timesharing systems such asMicroNET, the DELETE command can take several filenames at once. These filenames are separated on the command line by some valid terminator such as a comma, and the system goes through the list and kills each file from your directory. I realized that a similar facility would be very easy to set up for LDOS.

The accompanying program, called DELETE, is such a "multiple-KILL" facility. To use it, you simply type in

```
          DELETE filespec,filespec,filespec ....
```

for as many filespecs as you can fit on the command line. Filename, extension and password (if any) are required; howeverdrivespecs may be omitted. The files must be separated from each other by a comma or a space. DELETE will go through the list and search the currently mounted disks for the specified files, deleting them one by one. Any errors which occur cause the program to display the offending file name and the accompanying error message. At the end of the process, DELETE will display the number of files successfully killed.

If in the process of killing the listed files DELETE comes upon an invalid filespec or unrecognizable terminator, it will display the message, "Invalid filespec/terminator encountered." and attempt to continue, skipping past the error if possible.

DELETE has one advantage over PURGE; it can search every mounted disk for a particular file to kill, whereas PURGE is restricted to searching a particular drive only. However, DELETE is limited by the size of the command buffer. It may be a good idea to rename this utility and give it a single letter for a filename, in order to maximize the available space in the command buffer. In spite of this limitation, however, DELETE is much easier to use than either KILL or PURGE where only two or three (or even four) files need to be killed.

Use the BINHEX/BAS program to convert this to a /CMD file. For those of you who have the compiled BINHEX/CMD program, the checksum is *F0.

```
05 06 44 45 4C 45 54 45 01 02 00 52 E5 21 9B 52 CD 67 44 E1 7E FE 0D 28 1E DD
21 7F 53 11 80 53 CD 1C 44 32 7E 53 20 18 CD 24 44 20 24 CD 2C 44 20 1F DD 34
00 18 E6 21 F5 52 CD 67 44 C3 30 40 3A 7E 53 FE 0D 28 3A E5 21 3D 53 CD 67 44
E1 18 CC E5 F5 3A 80 53 CB 7F 20 03 CD 28 44 21 80 53 3E 0D 01 20 00 ED B1 3E
03 77 21 80 53 CD 67 44 21 66 53 CD 67 44 F1 CB F7 CB FF CD 09 44 E1 18 9C 21
```

```
7F 53 11 6A 53 CD 87 52 21 6A 53 CD 67 44 C3 2D 40 7E 0F0F 0F 0F CD 90 52 7E
E6 0F C6 90 27 CE 40 27 12 13 C9 44 45 4C 45 54 45 20 2D 20 4D 75 6C 74 69 70
6C 65 20 66 69 6C 65 20 64 65 6C 65 74 65 20 75 74 69 6C 69 74 79 20 66 6F 72
20 4C 44 4F 53 20 2D 20 56 65 72 2E 20 32 28 33 29 0A 28 63 29 20 31 39 38 32
20 62 79 20 52 2E 20 42 2E 20 52 65 79 65 73 2C 20 50 68 2E 44 2E 0D 0A 46 69
6C 65 20 73 70 65 63 28 01 A3 00 53 73 29 20 72 65 71 75 69 72 65 64 0A 53 79
6E 74 61 78 20 69 73 3A 20 44 45 4C 45 54 45 20 66 69 6C 65 73 70 65 63 2C 66
69 6C 65 73 70 65 63 2C 66 69 6C 65 73 70 65 63 2C 2E2E 2E 0D 49 6E 76 61 6C
69 64 20 66 69 6C 65 73 70 65 63 2F 74 65 72 6D 69 6E 61 74 6F 72 20 65 6E 63
6F 75 6E 74 65 72 65 64 2E 0D 20 2D 20 03 3030 20 66 69 6C 65 28 73 29 20 64
65 6C 65 74 65 64 2E 0D 00 00 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 0D 02 02 00 52
```

NODAM - Read those OLDAM disks directly - by Chuck

This program came about as a result of needing to read someone's TRSDOS data disk on my LX-80 at home. This disk also had to be returned to the owner in original condition. The following program will allow Model III and Model I LX-80 owners to directly read a diskette that uses the old Data Address Mark. It is very simple to use, but like some simple programs, it creates a very simple way to destroy diskettes. Here is a description of the program, and how to avoid losing data when using it.

The program bypasses the checking for the directory track DAM by indicating that the proper DAM was found WHENEVER THE SYSTEM ACCESSES THE DIRECTORY TRACK AS INDICATED IN THE DRIVE CODE TABLE. This means that you MUST either do a DEVICE or LOG command after installing the program and before attempting to read the diskette. To install the program, type in the following command, with the ":d" being the drive number to be used:

```
   NODAM :d
```

Then, either do a DEVICE command, or a LOG :d command. This will properly update the DCT with the correct directory track. The programs may now be read off of the disk onto a normal LDOS disk. The software write protect bit will also be set, and should show in the DEVICE command if you are using LDOS 5.1 or later. Now for the pitfalls .

Since the system is relying upon the DCT information, if you switch diskettes in a NODAMed drive and forget to log it in, and the diskettes have directories on different tracks, scratch one diskette after the first attempted write operation to that drive. Also, using the BACKUP command to move information off of the NODAMed drive will write the new DAM onto that disk when removing the MOD flags unless it is write-protected! Since the software write protect will not work with LDOS 5.0, or on LX-80's without the 5.1 ROM, the best procedure to follow is to always physically write-protect any diskette to be read in the NODAMed drive.

Once the disk is read and copied, NODAM may be removed as follows. For LX-80 owners with any LDOS version, use the SYSTEM (DRIVE ,DRIVER) command to restore the proper driver information. For those with the 5.1.2 update, use the SYSTEM (DRIVE=,DRIVER) command and the MOD1/DCT or MOD3/DCT program. Model 1, 5.1.1 owners can use the SYSTEM (DRIVE=,DRIVER) command, pressing <BREAK> to restore the dafault driver address. For Model I, 5.0 and Model III 5.1.$, do a global RESET or BOOT to restore the normal drive configuration.

Following is the code for the NODAM/CMD program. Use the BINHEX/BAS program to convert it, following instructions found with the program. For those with BINHEX/CMD, the checksum is *59.

```
05 06 4E 4F 44 41 4D 31 01 FE 00 52 E5 21 6B 52 CD 67 44 3A 25 01 FE 49 21 49
40 20 03 21 11 44 22 35 52 22 3E 52 E1 7E 23 FE 20 28 FA FE 3A 20 3D 7E FE 30
38 38 FE 38 30 34 D6 30 4F CD 8F 47 2A 00 00 01 24 00 B7 ED 42 22 00 00 23 E5
21 E3 52 FD 7E 01 77 23 FD 7E 02 77 E1 FD 75 01 FD 74 02 FD CB 03 FE EB 21 D8
52 ED B0 C3 2D 40 21 C6 52 CD 67 44 C3 30 40 1F 4E 4F 44 41 4D 20 2D 20 4F 6C
64 20 44 61 74 61 20 41 64 64 72 65 73 73 20 4D 61 72 6B 20 72 65 61 64 65 72
20 2D 20 56 65 72 20 31 2E 30 0A 43 6F 70 79 72 69 67 68 74 20 31 39 38 32 20
62 79 20 47 61 6C 61 63 74 69 63 20 53 6F 66 74 77 61 72 65 2C 20 4C 74 64 2E
0A 0D 42 61 64 20 64 72 69 76 65 20 6E 75 6D 62 65 72 21 0D 18 08 00 00 05 4E
4F 44 41 4D CD 00 00 F5 7A FD BE 09 20 0E 78 FE 09 28 04 FE 0A 20 05 F1 3E 06
B7 C9 F1 C9 02 02 00 52
```

MX80/FLT

This filter will allow the MX-80 to properly print the TRS-80 graphics set. It will work with both 5.0 and 5.1 versions, and on both the Model I and III. The BINHEX/CMD checksum is *D4.

```
05 06 4D 58 38 30 20 20 01 02 00 52 D5 1A F5 21 72 52 CD 67 44 3A 25 01 FE 49
28 08 21 49 40 11 7B 44 18 06 21 11 44 11 8A 42 22 41 52 22 4D 52 ED 53 6D 52
F1 CB 5F 20 32 CB 67 20 33 CB 4F 28 34 DD E1 DD 6E 01 DD 66 02 22 20 53 2A 00
00 22 08 53 01 1C 00 AF ED 42 22 00 00 23 DD 75 01 DD 74 02 EB 21 06 53 ED B0
C3 2D 40 21 CE 52 18 08 21 E0 52 18 03 21 F1 52 CD 00 00 C3 30 40 1F 4D 58 38
30 20 2D 20 4C 44 4F 53 20 4C 69 6E 65 20 50 72 69 6E 74 65 72 20 46 69 6C 74
65 72 20 2D 20 56 65 72 73 69 6F 6E 20 31 2E 31 0A 43 6F 70 79 72 69 67 68 74
20 28 63 29 20 31 39 38 31 20 62 79 20 4C 6F 67 69 63 61 6C 20 53 79 73 74 65
6D 73 2C 20 49 6E 63 2E 0A 0D 44 65 76 69 63 65 20 6E 6F 74 20 61 63 74 69 76
65 0D 44 65 76 69 63 65 20 69 73 20 72 6F 75 74 65 64 0D 4E 6F 74 20 61 6E 20
6F 75 74 70 75 74 20 64 01 24 00 53 65 76 69 63 65 0D 18 07 00 00 04 4D 58 38
30 38 0E F5 79 FE 81 38 07 FE C0 30 03 C6 20 4F F1 C3 00 00 02 02 00 52
```

The following patches for Radio Shack's PROFILE program were developed by Dick Yevich. They are for both the Model I and III. Following are the text and patches from Dick.

PROFILE ON LDOS

The following patches will allow the program PROFILE to run under LDOS. They are presented for both the 3.1 version for the MOD-I and the 3.2 version for the MOD-III. After these patches is an alternate patch which offers some changes to the product to allow it to be more easily used. Following this brief description are the patches which affect this result:

1. The PROFILE/CMD and the overlays SORT, INIT, ACCESS, and PRINT can be on any drive.

2. As standard PROFILE, the PRODAT file must start on drive 0, and the other files of INFOFILE, FORMFILE, and LPFORM will also be built on :0.

3. PRODAT will use:

```
MOD-I:     drive 0    - 144 sectors
           drives 1-3 - 302 sectors
MOD-III:   drive 0    - 448 sectors
           drives 1-3 - 672 sectors
```

4. For the MOD-III 3.2 version only, KI/DVR is utilized, but TYPE AHEAD is turned off on entry and turned-on on exit (if it was on at the start). The <SHIFT><0> is supported for upper and lower case for field names, video display, and print headings.

```
. 3.1.1 patch for PROFILE/CMD, vers 3.1, MOD-I
X'527F'=20 20 4C
X'52AC'=2E 31 20
X'534F'=4C 44 4F 53 20
X'5AFD'=03
X'5B04'=03
X'5B0D'=03
X'5B15'=03
.EOP

. 3.1.1 patch for INIT, vers 3.1, MOD-I
X'7388'=90 00 2E 01 2E 01 2E 01
.EOP

. 3.1.1 patch for SORT, vers 3.1, MOD-I
X'76F1'=20 4C
X'7701'=20
.EOP

. 3.1.1 patch for PRINT, vers 3.1, MOD-I
X'73BD'=20 4C
X'73D1'=20
.EOP

. 3.2.1 patch for PROFILE/CMD, verB 3.2, MOD-III
X'527E'=20 20 4C
X'52AB'=2E 31 20
X'534E'=4C 44 4F 53 20
X'5AFC'=03
X'5B03'=03
X'5B0C'=03
X'5B14'=03
X'5FB3'=00 00 00 00 00 00
X'5FD0'=C3 00 67
X'65C3'=C5 CD 4D 52 FE 61 38 06 FE 7A 30 02 E6 DF C1 C9
X'6700'=3E 00 21 1C 40 CB BE CB 4F 21 15 67 C2 99 42 C3
X'6710'=2D 40 53 59 53 54 45 4D 20 28 54 59 50 45 29 0D
.EOP
```

```
. 3.2.1 patch for INIT, vers 3.2, MOD-III
X'7031'=00 00 00 CD 80 76
X'7062'=1A
X'738A'=C0 01 A0 02 A0 02 A0 02
X'7680'=21 1C 40 CB FE 3A 89 42 32 01 67 AF 32 05 44
X'768F'=3E 0A C3 40 40
.EOP

. 3.2.1 patch for PRINT, vers 3.2, MOD-III
X'73BD'=20 4C
X'73D1'=20
.EOP

. 3.2.1 patch for SORT, vers 3.2, MOD-III
X'76F1'=20 4C
x'7701'=20
.EOP
```

ADDITIONAL PATCH FOR THE MOD-III

This changes 2 and 3 above for the MOD-III as follows:

1 - The PRODAT file will start being formatted on drive 1, and the other files will also be built on drive 1, allowing a normal SYSRES to be used.

```
2 - PRODAT uses:  drive 1    - 592 sectors
                  drives 2-4 - 672 sectors
```

On drives 2-4, that leaves 1.5K for BOOT, 4.5K for SYS0 and 6.0K free for misc. On drive 1, it leaves enough room to hold the INFOFILE, FORMFILE, LPFORM, and the command and overlays.

------- these are additions to the above patches.

```
. 3.2.2 patch for PROFILE/CMD, vers 3.2, MOD-III
X'52AB'=2E 32 20
X'5AD9'=31
X'5AE4'=31
X'5AED'=31
X'5AF6'=31
.EOP

. 3.2.2 patch for INIT, vers 3.2, MOD-Ill
X'7101'=00
X'7198'=01
X'7292'=37
X'7355'=00
X'738A'=50 02 A0 02 A0 02 A0 02
.EOP

. 3.2.2 patch for SORT, vers 3.2, MOD-III
X'77B4'=CD 7C 7B
X'7B7C'=3C DD 77 07 C9
.EOP
```

Any problems encountered can be conveyed to us and we will attempt a resolution. We will assist anyone in adding the KI/DVR to the MOD-I version. Call Dick Yevich at Yevich Assoc., Inc. (609) 428-0021.

Old Subject Revisited

By popular demand, the following section is a reprint of the SALVAGE/BAS program and Radio Shack patches from the October newsletter.

Since Radio Shack made some major changes to TRSDOS for the Model I and III, we have developed a program to move files off of the new TRSDOS 2.3B Model I disks. This is an LBASIC program called SALVAGE/BAS. It will copy files from a 2.3B or later TRSDOS disk to an LDOS disk.

```
10 REM--SALVAGE/BAS
20 REM--Program to get files from a TRSDOS 2.3B disk
30 REM-- onto LDOS disks, Model I or III.
40 REM--Use only on the Model I, 2.3B or later. Model III owners should
50 REM-- use CONV to get the files from theThSDOS 1.3 disk.
100 LINE INPUT "Source filespec? ";SF$
110 LINE INPUT "Destination filespec? ";DF$
200 OPEN"RO",1,SF$
210 OPEN"R",2,DF$
220 FIELD 1,1 AS D1$   'Find source FCB
230 FIELD 2,1 AS D2$   'Find destination buffer
240 Z1=VARPTR(D1$):Z1=PEEK(Z1+1)+256*PEEK(Z1+2)
250 Z2=VARPTR(D2$):Z2=PEEK(Z2+1)+256*PEEK(Z2+2)
260 POKE Z1-32+3,Z2AND&HFF:POKE Z1-32+4,Z2/&H100 'Use 1bfr
270 N=PEEK(Z1-32+12)+256*PEEK(Z1-32+13) 'Get ERN
280 IF PEEK(Z1-32+8)=0 THEN 300 'Adjust if needed
290 N=N+l:POKE Z1-32+12,NAND&HFF:POKE Z1-32+13,N/&H100
300 FOR I=1 TO N 'Copy file
3l0 GET 1:PUT 2
320 NEXT I
330 POKE Z2-32+8,PEEK(Z1-32+8) 'Transfer offset
340 POKE Z1-32-2,0 'Don't close source file!
350 END
```

It was also necessary to develop some small patches for Radio Shack's Cobol and RSBasic compilers. The Cobol package will refuse to run under anything but TRSDOS 2.3B on the Model I or TRSDOS 1.3 on the Model III unless patched, and both packages do some peculiar things in attempting to handle the <Break> key, which may not work under LDOS. Although this may not be clear from the documentation, these programs were designed to run on either the Model I or Model III, so the patches can be applied to a copy taken from either the Model I or Model III disks in the Radio Shack package. The resulting patched programs will run on either Model I or Model III LDOS. Due to the incompatibility between TRSDOS 2.3B and earlier versions of TRSDOS, it was also necessary to write a special program to recover files from 2.3B disks. The REPAIR utility is not capable of repairing 2.3B diskettes (see SALVAGE/BAS, page 15). If you have LDOS 5.1, it is recommended that you use the CONV utility to transfer the Model III package to an LDOS formatted disk.

```
. RSCOBOL/FIX
. Patch for Radio Shack COBOL compiler version 1.3B
.   to run under LDOS Model I and III
X'A196'=4F
X'A1D8'=50
X'9A4F'=D2
X'9A5A'=3E C9 32 0C A0 00
. End of patch

. RUNCOBOL/FIX
. Patch for Radio Shack COBOL Version l.3B runtime package
.   to run under LDOS Model I and III
X'AE6B'=4F
X'AE7E'=50
X'9B38'=38
X'9B4A'=3E C9 32 B0 A9 00
. End of patch


. CEDIT/FIX
. Patch to RS Cobol editor version 1.3B to run under LDOS
.   model I or III.
X'5832'=3E C0 32 13 5C 00 00 00 00 00 00 00
X'5C05'=C3
. End of patch


. RSBASIC/FIX
. This patch corrects the Radio Shack Basic Compiler 2.4
.   for proper operation under LDOS Model I and III.
.   Either the Model I or Model III version may be patched,
.   and the resulting program will run on either model0
X'9971'=C9
X'997B'=00 00
. End of patch


. BEDIT/FIX
. This patch corrects the Radio Shack Compiler Ba8ic editor
.   program for proper operation on Model I/III LDOS.
.   Either the Model I or Model III version may be patched,
.   and the patched version runs on both models.
X'58DB'=00 3E C9 32 7C
X'58E4'=00 00 00
X'5C6E'=C3
. End of patch
```

Patches to MOD I - LDOS 5.1.2 - 03/23/82

These patches are to be installed on all MOD I - 5.1.2 disks with files dated before 03/20/82.

Fixes a problem with the <ENTER> key when using a single drive or the (X) parameter in BACKUP.

```
Patch BACKUP/CMD.RRW3
X'5408'=E8 55
X'55E8'=CD 2B 00 B7 C9
.EOP
```

Fixes a problem with RS232R.

```
Patch RS232R/DVR.GSLTD
D02,49=00 00
D02,4F=98
.EOP
```

The KI driver setup the wrong type byte in the DCB.

```
Patch KI/DVR.GSLTD
D02,82=05
.EOP
```

FORMAT initialized the wrong passwords for BOOT and DIR.

```
Patch FORMAT/CMD.RRW3
D04,F2=AE 01 F5 9C
D05,12=AE 01
.EOP
```

The PDUBL driver did not setup all 8DCTs

```
Patch PDUBL/CMD.RRW3
D00,40=51
D00,86=08
D00,8F=0A
.EOP
```

Patches to MOD III - LDOS 5.1.2 - 03/23/82

These patches are to be installed on all MOD III - 5.1.2 disks with files dated before 03/20/82.

The KI driver setup the wrong type byte in the DCB.

```
Patch KI/DVR.GSLTD
D02,85=05
.EOP
```

FORMAT initialized the wrong passwords for BOOT and DIR.

```
Patch FORMAT/CMD.RRW3
D04,F2=AE 01 F5 9C
D05,12=AE 01
.EOP
```

RS232T could not be "configed".

```
Patch RS232T/DVR.GSLTD
D02,5D=C7
.EOP
```