THE LDOS QUARTERLY January 1,1982 Volume 1, Number 3

LDOS

VERSION 5.1
THE TRS-80™ OPERATING SYSTEM
MODEL I AND III

LOGICAL SYSTEMS INC.

Now for the TRS-80™ Models I & III

LDOS

VERSION 5.1
THE TRS-80™ OPERATING SYSTEM
MODEL I AND III

∗ DOUBLE Sided & DOUBLE Density support
∗ AUTOMATIC Density recognition.
∗ 35, 40, 77, 80 and any other track counts are supported.
∗ All available drive stepping rates are supported.
∗ Hard Drive support, can be HANDLED AS A SINGLE DRIVE
∗ Hard drive partitioning, one drive can act as up to six.
∗ Intermix 5", 8" and Hard drives, up to a total of 8 drives.

- Compatible with the Model I Radio Shack Expansion Interface.
- Upward compatible with TRSDOS (2.3 & 1.2 as documented).
- Fully supports Microsoft language products for the TRS-80.
- Complete media compatibility Model I to Model III and back.
- Full support for LOBO's LX-80 interface.
- Full support for AEROCOMP LC double density controller (DDC).
- Full support for PERCOM's DOUBLER II.
- Complete documentation (well over 250 pages).
- Complete technical information.
- A TOLL-FREE 800 number for customer service.
- An LDOS users bulletin board on MicroNET.
- A Quarterly LDOS users magazine (The LDOS QUARTERLY).
- A liberal update policy.
- An enhanced BASIC (LBASIC) including:
  - ➢ Upward compatible with Microsoft Basic
  - ➢ High speed LOAD and SAVE.
  - ➢ Run multiple programs with common variables.
  - ➢ BLOCKED (variable length) files are supported.
  - ➢ DOS commands may be executed from LBASIC.
  - ➢ Built in string array SORT.
  - ➢ Single stepper for debugging.
  - ➢ Several new statements and file modes.
- A compiled JOB CONTROL LANGUAGE (JCL).
- CONVert utility to move files from Model III TRSDOS.
- An extended Debugging and Monitor program (with disk access).
- CMDFILE for movement of disk and/or tape system (/CMD) files.
- Device independent operation.
- Full LINKing, ROUTEing, FILTERing and SETting are supported.
- MiniDOS feature for constant access to certain DOS commands.
- RS-232 DRIVER for serial support.
- Sophisticated communications software included.
- Wildcard characters and partial Filespecs are supported.
- DATED FILES, show when a file was last written to.
- Backup: Mirror, by Class, if Modified, by Date, by Extension, etc..
- Selectable PURGE for fast disk "cleanup" of unwanted files.
- Print formatter, for control of printer output.
- Built in printer SPOOLER, to both disk and ram.
- Joblog to record all system operations with time stamps.
- UPPER and *lower* case support, throughout the system.
- Blinking cursor with selectable cursor character.
- 128 character TYPE AHEAD buffering for keyboard input.
- Assign strings to individual keys with Key Stroke Multiply (KSM).
- SUPER FAST operation with the SYSRES function.
- Extensive user control and system feedback.
- Advanced PATCH utility for easy maintenance.
- Complete transportability of software among all Z-80 LDOS systems through the use of the LDOS high ram supervisory call system (SVC).
- Dealers to serve users Nationwide and in the Common Market.
- The only DOS with a limited ONE YEAR WARRANTY!
- Enjoy a professional.operating system on your TRS-80.

*** Specific hardware is required to use these functions.**

The Ultimate In Operating Systems for Model I & III

Only $169⁰⁰

Version 5.1

∗ Model 1 LDOS provided on 35 track single density media.
∗ Model III LDOS provided on 40 track double density media.
∗ LDOS can be provided on special media configurations at an additional charge.
∗ Prices and specifications are subject to change without notice.
 Although not required, LSI recommends two or more drives when using LDOS.

For Further Information Contact The Distributor Or Dealer Nearest You:

(west)
LOBO DRIVES INT' L
354 S. Fairview Ave.
Goleta, CA 93117
(805) 683-1576

(Central)
Galactic Software Ltd.
11520 N. Port Washington
Mequon, WI 53092
(414) 241-8030

(East)
MISOSYS
5904 Edgehill Dr.
Alexandria, VA 22303
(703) 960-2998

(The Common Market)
MOLIMERX LTD
1 Buckhurst Rd, Bexhill
Sussex, England
(0424)-220391

LOGICAL SYSTEMS INC.

THE LDOS QUARTERLY January 1,1982 Volume 1, Number 3

Table of Contents

VIEW FROM THE BOTTOM FLOOR by Bill Schroeder

We are very pleased to announce that LDOS 5.1.1 MOD I was released to hundreds of users on November 30th and hundreds more are now awaiting homes on dealer shelves around the world. You will find an expanded list of new features that 5.1.1 offers elsewhere in this newsletter. We feel that 5.1.1 is without doubt the best thing to ever happen to the TRS-80 world and we are sure you will agree. Any 5.0.x user can still "trade-in" a 5.0.x for a 5.1.x as stated in the last LDOS newsletter.

5.0.1 Service Renewal

The fee to extend your LDOS support for an additional year has been tentatively set at $50.00. This fee will be subject to change without notice!! We have made every effort to set this fee based on the cost of providing support, but we had to guess at what percentage of our users will renew their support. If we are close with our estimate, this amount will stand. If not, it will have to be adjusted. All renewal notices will contain a "Extended Support Agreement" which will clearly state the services to be received and the total cost for the user to subscribe to those services. This will be an extension of the current support being provided to new LDOS purchasers. If a user does NOT extend his support agreement, he will not be allowed use of the $5.00 update service, the toll-free 800 line, the MicroNET bulletin board and will stop receiving the LDOS quarterly Newsletter. When your warranty expires you will be unsupported, as with any other warranty. This means that LSI will provide NO services to you. This is the purpose of the Extended Agreement. We will provide services to those who wish them but not for free. Each user will be offered the extended support agreement as his warranty expires. There will be a 30 day grace period after warranty expiration, during which time the user may execute his Extended Support Agreement. After that grace period, the user's registration will be totally removed from our support and customer service system. If at some later date the user wishes to re-establish his support, a fee somewhat higher than the extension agreement fee will be charged.

Updates, upgrades, and ordering

Please note that all upgrades or updates to LDOS products are handled through LSI directly at 11520 N. Port Washington Rd., Mequon, Wisc. 53092. A very long delay could occur if you send to a dealer or a distributor. In the Common Market you should send your LDOS to MOLIMERX at 1 Buckhurst Rd., Bexhill, Sussex, England. LSI does NOT accept credit cards or purchase orders. All upgrades of any sort must be accompanied by a check or money order in U.S. FUNDS. Molimerx, of course, should be sent funds in English pounds.

There has been some confusion regarding what is meant by LSI, LSI Distributor and LDOS Dealer. All upgrades and updates come from LSI directly and must be accompanied with a check for the exact amount (LSI does not honor credit cards). LSI DOES NOT SELL ANY OF ITS PRODUCTS RETAIL!! There are four official LSI Distributors; LOBO Drives in California, Galactic Software in Wisconsin, MISOSYS in Virginia and MOLIMERX in England.

These distributors wholesale to dealers in their areas and handle retail sales. All distributors honor COD, CASH, CHECK and CREDIT CARD retail orders for all LSI products. LDOS Dealers are retail sellers of LDOS and certain other LSI products, but SOME LSI PRODUCTS ARE AVAILABLE FROM DISTRIBUTORS ONLY. If you have any questions about where and how to obtain any LSI product, call LSl at (414) 241-3066.

A $5.00 update to 5.1.1 will be available for owners of 5.1.0 MOD III LDOS in the near future, at which time the MOD I & MOD III Systems will contain the identical set of features throughout. Watch for an announcement of MOD III 5.1.1 availability this coming spring.

There are several small patches for each of the LDOS versions in this newsletter. You may apply these to the proper version of LDOS or send in your Master disk with 0 and we will update it. Should you elect to apply the patches yourself, it will be your responsibility to keep track of the status of your system. We feel our update service is very important and any prudent user should send us his Master disk several times a year to make sure of having an official "current" master. The current versions of LDOS are:

MODEL #	VERSION #	FILE MOD DATE
MOD I	5.0.3A	11/15/81
MOD I	5.1.1	12/15/81
MOD III	5.1.0A	12/15/81

If you use our updating service to keep your system current, you can determine from the above table if you should send in your disk for updating. To check your Master, look at the label and then do a "DIR (S,I,A)" on that disk. The MOD Date will be shown in the display. The /SYS and LDOS utility files should show MOD dates later than or equal to those shown in the above chart. If not, send in your MASTER with $5.00 and we will take care of it. If you are applying the maintenance patches yourself, you must keep accurate records as to the status of your system.

5.0.3A updates are now being returned with a MINIDOS filter on the the disk. The documentation has been appended in a special way to the end of the filter file. If you have this file on your 5.0.3A disk, you can get instructions for it by using the command "LIST MINIDOS/FLT". Let the list run through to completion. At that point the documentation will be on your screen.

In the case of our 5.1.1 MOD I LDOS, there are TWO disks in the system. One is the system disk called LDOS 5.1.1 and the other is LDOSXTRA. When sending in a 5.1.1 system for updating DO NOT SEND THE LDOSXTRA DISK. If we should need to update the LDOSXTRA disk we will let you know.

Some of our users are under the impression that they MUST trade up to 5.1.1, or that we are dropping support for 5.0.3A. This is not at all correct. We will continue to support 5.0.x as long as a practical number of users are still using it and are covered by their original warranty or their Extended Support Agreement. Future products from LSI, Galactic or MISOSYS that are designed for LDOS will probably be designed to work under 5.0.x or 5.1.x. This will not be 100% true; some of our future products will require the advanced features of our 5.1.x product line and will not be available to 5.0.x users.

Customer Service

We have been listening to you, our users, and have completely redone and expanded our manual, The MOD I 5.1.1 manual is now nearly 400 pages long. It is sequentially page numbered within sections and contains a complete INDEX, TABLE OF CONTENTS and TAB INDEXES. I believe it is absolutely the best manual in the Micro industry, bar none.

Many users have been taking us up on our offer to create LDOS system disks of other than standard configurations. This is fine, but please tell us EXACTLY what you want. We need to know the number of tracks, one or two sided, 5 or 8 inch, and the density. We can make special LDOS system disks in almost any configuration. Each official LDOS master is created in an "OPTIMISED" fashion with controlled placement of files on the disk to increase overall proformance. The charge is still $5.00 if you provide the disk and $10.00 if we provide the disk. We will NOT modify your MASTER LDOS disk under any circumstances.

We will also provide the proper version of SCRIPSIT for use with LDOS if an LDOS owner cannot find an original 1.0 SCRIPSIT to patch. The charge again is $5.00 if you provide the disk and $10.00 if you do not. You MUST also send proof that you have purchased some "legal" copy of SCRIPSIT, or you can send an original SCRIPSIT master disk with the Radio Shack label. Without proof of ownership of SCRIPSIT we will not be able to help you.

Some Model III users have complained that the only way they can get files off of a Model I TRSDOS type disk with their MOD III and LDOS is to "REPAIR" the disk and then copy the files off. THIS IS NOT SO!!! Once the REPAIR command is used on the MOD III you cannotun-REPAIR (put back the old data address mark) on the MOD III. This is not a flaw in the software as the MOD III is not capable of writing the "OLD" data address mark. If you have a MOD I type disk (35 track, Single Density) and you don't have a MOD I computer that you can use to make a backup before doing the REPAIR, you can proceed as follows:

1> Boot under TRSDOS 1.2 or 1.3
2> Place the MOD I type disk in drive 1
3> Use the TRSDOS "CONVERT" command to move the files to MOD III TRSDOS
   in drive 0
4> After CONVERT has finished remove both disks
5> Place LDOS in drive 0
6> Place the MOD III TRSDOS that contains the moved files in drive 1
7> Reboot the system
8> Use the LDOS "CONV" command to move the files from the drive 1 TRSDOS
   disk to the drive 0 LDOS disk

This procedure will leave the original MOD I type disk untouched and directly usable by MOD I TRSDOS. We do not understand why some users feel they must have the original MOD I type disk with the "OLD" data address marks, especially if they own only a MODEL III. But this is the procedure to use if you must.

We are finding some very disturbing statistics regarding our 800 customer service lines. We get over 90% of our calls from LESS THAN 10% of our customers. We find that a vast majority of our calls are handled simply by telling the users the same thing that is in the manual (which they neglected to read). Toll free 800 service is not clearly understood by most people. I have talked with many people who think we pay some FLAT RATE and can then have all the incoming calls we want. This is incorrect. WE PAY FOR EVERY SINGLE CALL THAT COMES IN ON THE 800 LINES BY THE MINUTE. At the present rates we pay over $1500 dollars a month just for our 800 service. So please, call us when you need us, but check your manual first before you call. I can't see throwing away a technician's valuable time plus 35 cents a minute just to read the LDOS manual to a user over the phone.

Before calling customer service with a "bug report" please make a simple check yourself. Make a mirror backup of your MASTER LDOS disk and make sure that you can duplicate the problem with that disk. If not, there is something wrong with the files that are on the offending disk. Remember that when we make major changes in the system, you must move ALL related files. This is very important. For example, a 5.0.x version of BACKUP or FORMAT may not work with a 5.1.x system. Also, you should understand that we use our product over 200 hours per week. If a major area of the system did not work we would know about it in short order. So if your keyboard goes dead, or your video goes blank, or LDOS won't boot or your basic program gets scrambled when you save it, the chances are that YOU have a problem with your hardware or the use of the software.

We also get calls that have NOTHING TO DO WITH LDOS, such as calls about hardware modifications that we know nothing about. We are not a hardware company! So if you hook up a right handedwigit to your expansion port while holding your RS232 cable at a right angle to the interface and connect your "OH-MY-GOSH" drive to your "YOUGOTTOBEKIDDING" interface and reverse your MASTER disk in your "FLOPPED" drive..........and your system won't boot????.......PLEASE DON'T CALL. We make it very clear what hardware we support. If we don't come out with a support statement then it is safe to assume that we know nothing about it.

We also get many calls asking us how to make this or that program run with LDOS. We do not have the time to do this. It is not our responsibility to make other companies software work with LDOS. You should contact the authors of the product and ask them to correct or modify their product. It is also prudent that an LDOS user specify to a vendor that he intends to run a program on LDOS when ordering that program. We now have æsysLem by which any prominent software author can get complimentary copies of LDOS for development purposes. If you know of a software package you want supported, have the author contact us for details.

From time to time we will offer patches or procedures to make selected programs run with LDOS, as we have with PENCIL, SCRIPSIT,VISICALC, MAC-80, and several others. We are now looking at MOD I & III PROFILE with the intent of correcting it to work correctly on LDOS (no promises, but we will try). As our product line expands and we have a larger user base, it will become harder and harder for us to find the time to deal with other company's products. We are fortunate that more and more companies are providing LDOS compatible products every day. This is of great benefit to both the LDOS users and LSI.

Some users have been saying that they have trouble getting through on the 800 lines. I have been monitoring these lines very closely and it is very rare that all the lines are busy. It does happen, but not often. If you get a busy signal, check the time. Remember that the 800 lines are open from 10am to 12 noon and from 4pm to 6pm Central time only. At any other time you will get a busy signal. If you must have assistance outside of these hours you can call (414) 241-3066. This is the main number at LSI.

About our mail.... we get many letters every day from satisfied users. Most of these ask questions soliciting our comments or recommendations on products, programming procedures or LSI policies. We just don't have the personnel to answer all of these letters. If the answers to your questions are important to you, then give us a call at (414) 241-3066 and we will try to answer them. It is much less expensive to talk on the phone to you for 5 or 10 minutes then to generate lengthy correspondence. We do try to answer certain technical letters with accurate information and will continue to do so. Please DON'T stop writing. We do like to hear from our users and we do consider all written input when we set policies or design new products.

Now I would like to mention a very touchy subject........ SOFTWARE PIRACY. Most anything I could say about thismega-buck theft situation has been said. I would like to make an important point that has NOT been expressed. Those of you who are honest, legitimate owners of software should realize that the condoning of this crime, even without participation, is just as bad as participating. Software of the magnitude of LDOS MUST BE SUPPORTED, and the more sales that are made, the better support and development for that product will be. If you see someone giving away a copy of LDOS, just think of it as ONE LESS NEW FEATURE that will be in the next version of YOUR LDOS. Our development money does come from sales. We could go to elaborate disk protection schemes, but that would be very unfair to the vast majority of LDOS users, the honest ones, who would not give away or sell a copy of our system. But it appears that the 99% may have to suffer at the hands of the 1% of criminals that are out there. We hope this practice will decrease so we will not find it necessary to take protective actions in the future.

One more point for the pirates out there; WE ARE ABLE TO IDENTIFY EVERY LDOS and have terminated support for several users, and in one case are contemplating legal action against a user. So beware.... we are not blind.

About Hardware and Software and ???

Some LDOS users are actively considering getting into double density. When deciding on a double density modification, you should also consider moving up to the LX-80 interface. The price has been dropped to just $449.00, and at that price it becomes a very attractive alternative. Consider the following. Your interface with ram must be worth $200 to $250 on the used market, and if you spend $150 (for a doubler) or $250 (for a 5" and 8"doubler), you have a $350 to $500 value involved. At just $449.00 for an interface that handles 5", 8", Double and Single density, Double and Single sided, up to 8 total drives, and is capable of BOOTING A DOUBLE DENSITY DISK, the LX-80 begins to sound like a pretty good deal. We at LSI use many of these fine interfaces and would not be without them. So for double density, do yourself a favor and consider selling your interface and getting into the LX-80 from LOBO.

Whenever you order any product for use with your computer, whether it be hardware or software, you should always include your LDOS registration number. Some products require LDOS, and some companies will give discounts to LDOS users. Your LDOS registration number can save you money and get you the right version of a product. If the vendor doesn't know you have LDOS he will not be certain to send you the LDOS version of his product. A simple statement like LDOS SERIAL NUMBER #########, could gave you a lot of aggravation and may save you some money.

If you or your company has a significant product that is LDOS compatible or LDOS specific and it does not compete directly with an LSI product, I will be glad to mention it in the "What's New" section. Just send me a copy of the completed product and a brief letter describing its functions, where it can be purchased, and at what price.

I would like to see short reviews of LDOS compatible hardware and software for publication in this newsletter. If you have had experience with a product that would be of interest to other LDOS users, write an article about it and send it to the QUARTERLY EDITOR at LSI. If it is worthwhile we will publish it.

Hard Disk Systems are now available featuring LDOS. All LDOS users should understand a few important facts about these systems. First off THEY ARE NOT ALL DONE BY LSI. Many of the Systems being advertised as featuring the LDOS operating system have the needed drivers and support software written and supported by companies other than LSI. We trust that these are competent implementations, but we have no way of knowing. LSI is directly supporting several hard drive systems to date. Two that are available as of now are from LOBO Drives International at 354 S. Fairview Ave.Goleta, Ca. 93117 and from Laredo Systems Inc. at 2264 Calle De Luan, Santa Clara, Ca. 95050. Both offer hard drive sub-systems for the MOD I & III.

The above mentioned companies have contracted with LSI to do the software creation and support for their systems. There is a difference in acquiring an OFFICIALLY SUPPORTED system and a partially supported system. If a hard drive vendor elects to feature LDOS as his operating system and does his own software development, he must support the system!! If you should buy a system that is not officially supported by LSI and you have problems or questions about the use of LDOS with the hard drive, don't call us!!! We will simply refer you to the place you bought the drive. When we officially support a hard drive system we must have those drives in our customer service department and in our development department. We will NOT be of any help with systems we did not create. We will of course still handle any problems that have nothing to do with the hard drive.

For LSI to author a hard drive implementation costs a vendor from $6,000 to $25,000 (this includes drivers, formatters, utilities, documentation and support). Then the vendor buys special finished LDOS packages directly from LSI and delivers them with each hard drive installation. LSI then handles all system software problems and customer assistance. If a vendor creates his own LDOS hard drive package, he is totally responsible for his portion of the package. This is not to say that other hard drive implementations of LDOS won t work or will have problems. They probably will work just fine, but we don't know that and will not be responsible for work we did not do!!

Roy Soltoff and myself attended COMDEX in Las Vegas in November. We were in search of the next machine for LDOS to be implemented on. We saw many new, well designed Z-80 systems and were impressed by several. It seems likely that LDOS will appear in 1982 on several all RAM MACHINES, the Z-90 from Zenith and the MODEL II from Tandy are just two of the possibilities. We will keep you posted as to what machines will be supported and when. In the next issue of this newsletter Roy will probably have an article on the basic design structure that we will be using on our ram based systems.

Many users have asked what publications we would recommend for LDOS users. We would consider one publication to be of major importance to the TRS-80 user. This is 80-US MAGAZINE. 80-US has just gone monthly and has held its subscription rate at just $16.00 per year. This is a real bargain for the original publication dedicated to the TRS-80 line of computers. The people at 80-US have also come out with a book called "THE CAPTAIN-80 BOOK OF BASIC ADVENTURES". For those interested in the adventure concept this is a must have book. See the ads for this book and 80-US magazine elsewhere in this newsletter or call 80-US at (206) 475-2219.

When Roy and I began the LDOS project we decided that our system would have NO SECRETS. We are willing to talk about any part of LDOS. But, this requires costly technician's time so details about the system code will not be given out for free. The charge is $50.00/hour, plus phone expenses, one hour minimum, any part of an hour treated as an hour. Customer service will help with any operational or functional questions, but they do not have the resources to answer questions about the system code. In keeping with our open door policy we have included in your manual a complete technical section and we will be expanding on certain areas of the systems operation in each issue of this newsletter. In the last issue Roy detailed the use of the LDOS parameter scanner. In this issue you will find a detailed article on the BYTE I/O concept and how LDOS deals with it.

The LDOS newsletter now contains a few ads for products relating to the TRS-80 and/or LDOS. We hope to have more in the future. The LDOS QUARTERLY is still being delivered third class, as there is no way that the present budget could handle the many thousands of dollars it would cost to send it first class (even though we would like to.)

One interesting comment that I have heard from several of our users is that OUR LAST NEWSLETTER WAS AS LARGE AS THE DOSPLUS 3.3 MANUAL. I checked and that is a correct statement, as both are about 48 pages.

WHAT'S NEW ?

New and exciting things are happening in the LDOS world. Many products are now becoming available in LDOS compatible or LDOS specific versions. There is not room to talk about all of them here, but at the risk of sounding like a commercial I would like to touch on a few.

New from GALACTIC software is a FILE EDITOR called "FED". This product is for the novice user, the serious user and the "real pro". It is simple and easy to use and is also very powerful. We use it constantly to maintain the LDOS system, and our secretaries use it to correct data bases. LDOS patches are now developed and implemented with FED. FED is a screen oriented file editor capable of doing most any type of file modification in both HEX and FULL ASCII. It will locate a HEX or ASCII string in a file or find the byte that loads at a specified address, or even tell you where the byte you have the cursor on (one of three cursors) will load. Advanced printer support is provided as well as two types of sector display modes. Elsewhere in this newsletter is additional information on FED. FED is available now for 5.0.x and 5.1.x from Galactic Software, 11520 N. Port Washington Rd Mequon, Wi. 53092, for just $40.00 (MOD I & III). Also from Galactic is a special LDOS version of their popular MAIL/FILE system at $159.00.

The "FILTER" package is available now from LSI distributors (MISOSYS, GALACTIC & LOBO) for $60.00. This package provides many very useful filters for almost all LDOS users. During development of this package, new ideas were continuously being implemented, and the resultant package contains many more filters than originally announced. The XLATE FILTER, BASIC LISTING FILTER or the CALC FILTER are worth the price of this package by themselves. Do yourself a favor and check this out. Details and a list of the filters in this package can be found elsewhere in this newsletter.

Many of our users have asked about Editor/Assemblers. We are proud to recommend one as the most powerful and versatile for the novice as well as the experienced programmer. That, of course, is EDAS. This is the Editor Assembler that LDOS is written and maintained with. There is that old saying about "WHY RE-INVENT THE WHEEL?". Well, this Editor Assembler wouldn't have been created had there been an assembler that was as powerful, flexible and easy to use. There wasn't, and so EDAS was written. EDAS comes complete with a very powerful LABEL CROSS REFERENCING SYSTEM, Editor Assembler and very good documentation. If you need or want an Assembler contact MISOSYS at 5904 Edgehill Dr. Alexandria VA. 20303 for more information. EDAS sells for just $79.00 (MOD I & III).

There is now a BASIC compiler written for LDOS by Bill Stockwell, and published by Breeze/Quality Software Distributing. This is (for lack of a better term) a "SEMI-COMPILER". That is, it generates optimized code, replacing most time consuming basic routines with assembler modules. Compared to other compilers, this one is "different" but very cost effective. It is only $69.95. Contact Breeze/Quality Software Distributing at 11500 Stemmons, Suite #125, Dallas, Tx. 75229, for additional information. Also from these same people is a product call "Script Plus 3.0" which adds many enhancements to Radio Shack's Scripsit package, and sells for just $39.95. Speaking of Quality Software, I should tell you that they have made a major change in their company as of late, consummating in a merger of themselves and Breeze Computing (Kim Watt's company). So if any of you are wondering where Kim is, he is now in Dallas sharing an office complex with QSD. Kim moved there from Detroit in November (a nice time to go south). Kim's NEW Super-Utility now supports LDOS completely, including the new LDOS data address marks.

There are several music generating systems for the TRS-80, but without a doubt the most popular was the ORCHESTRA-80 and now ORCHESTRA-85.

This new product (ORCH-85) will allow you to create beautiful music with your computer in STEREO NO LESS. This system allows for the creation and playing of music in up to five parts. Its like having a band in your TRS-80. ORCH-85 lists for just $129.00 and is LDOS compatible. For additional information contact SOFTWARE AFFAIR, Rubis Dr., Sunnyvale, Ca., 94087 (408) 295-9195.

From Walonick Associates comes a product called StatPac. This is a super statistical analysis package that is available in a special LDOS version. For additional information contact Walonick Associates at 5624 Girard Ave. South, Minneapolis, Mn 55419.

Now there is a clock-calendar board supported by LDOS. This is the "T-TIMER", detailed in an article later in this newsletter.

For a complete TAX PREPARATION system to use with LDOS, you can contact Stenholm & Quint CPA, 129 Concord St., Framingham, Ma. 01701 (617) 879-8330. This CPA firm produces a package for tax preparation used by professionals. Their system is call the "SQ1 Tax Preparation System".

MISOSYS has available a CPM to LDOS convert utility which will move files from certain 8 or 5 inch CPM disks to an LDOS type disk. Another product from MISOSYS is their popular DISASSEMBLER which is compatible with LDOS and EDAS. For information contact MISOSYS at 5904Edgehill Dr. Alexandria, Va. 22303.

Aerocomp has now been delivering their doubler for several months and from all reports it seems to be working very well. Their doubler was designed and engineered by Wayne & Skip at Aerocomp. Wayne, one of the designers of the PERCOM Doublers I and II, applied the knowledge he had gained in the creation of those products to build the proverbial "better mousetrap". So if their doubler isn't a great product, it should be. Aerocomp calls their doubler the "DDC". Aerocomp also has a very "trick" product they call the "DDS". This is a piggy back data clock separator which is designed to plug into the PERCOM and LNW doubler boards to make them a lot more reliable in double density. This product will correct most of the problems created by these boards. For more information contact Aerocomp at Hanger #8, Redbird Airport, Dallas,Tx. 75224.

Many of our users want a good Disk Catalog program that is designed for LDOS. Earle Robinson, a very competent assemblerprogramer, has written a very extensive catalog program just for LDOS. This catalog program has all the features you could want in this type of utility.Earle has started a new company called "softERware" at 16007 Miami Way, PacificPalacades, Ca. 90272 just to handle the sale of this and his future LDOS compatible products. This program is called "DISCATER" and comes with documentation for just $39.95.

Roy Soltoff, the Systems Analyst on the LDOS product, has come up with a very interesting new package. Many LDOS users have asked for a method of creating their own "LIBRARY" of commands. After some investigation, Roy has come up with a construct for the user to be able to create his own libraries. He calls these Partitioned Data Sets or PDS for short. Roy has put this concept to work in a unique package that allows creation, listing and changing of these library like modules. Now you can have several small programs in just one file, saving disk space and directory entries. This product is detailed by Roy elsewhere in this newsletter and is available through MISOSYS at 5904 Edgehill Dr., Alexandria, Va. 22303

LED is the name of the LDOS TEXT EDITOR. This is a new product from LSI and will be available shortly. This editor will allow the processing of line numbered or unnumbered ASCII text of almost any type. It has most of the functions of a screen oriented word processor but adds several specialized features to deal with certain types of text. It does not provide for printer output from within the editor, as the LDOS LIST command can be used. The Editor does support the entire ASCII character set. This product is available directly from the LSI distributor nearest you (do not order directly from LSI). The price for LED is $30.00. Contact an LSI distributor for more information if you are interested in this package.

LSI will release "LC", our integer version of the "C" programming language in early '82. The price for "LC" has been set at $150.00. This package will include the LDOS TEXT EDITOR "LED" for creation and maintenance of LC source files. LC will be available from LSI distributors or from your local LDOS dealer. There is a full description of this package elsewhere in this newsletter.

Now for a real "biggy". This is a new super LBASIC enhancement package from the people at SNAPP Inc. This is an LDOS implementation of their famous SNAPPWARE BASIC. Any serious basic programmer should not be without this package. It is not cheap, but is worth every penny you pay for it. It will pay for itself in saved time in short order. I could write a book just on this package, but there will be a review in the next issue of this newsletter. For now, call or write SNAPP Inc. 3719Mantell, Cincinnati, Ohio 45236 phone: (800) 543-4628.

LOBO Drives has made a large change in the pricing of their powerful LX-80 interface. LOBO is now offering this interface for just $449.00 with a full 32K of ram. This is the interface that runs 8", 5", Single density, Double density and double sided drives. With the LX-80 you can even boot a MOD I Double density system disk. This is an excellent bargain on a proven product. We use six of these constantly at LSI and have no problems. LOBO has also announced a FREE upgrade to the LX-80. To make the LX-80 even more reliable they have added a satelite data separation board and are offering to retrofit this board into all earlier LX-80s. If you already have an LX-80 contact LOBO for instructions on how to get this new board installed in your interface. For more information contact LOBO drives at 354 S. Fairview Dr. Goleta, CA. 93117.

Now available from HEXAGON SYSTEMS is a powerful, LDOS compatible spelling checker for text files. This is the second version of a successful product and provides many enhancements over the original version. This "proof reader" is available for just $99.00 from HEXAGON SYSTEMS, P.O. Box 397, Station A, Vancouver, B.C. Canada V6C 2N2 (604) 682-7646. See their ad in this newsletter.

Another very good spelling checker that is LDOS compatible is MICRO PROOF from CORNUCOPIA SOFTWARE at P.O. Box 5028, Walnut Creek, Ca. 94596. Prices start at just $89.50.

UPDATE NEWS - MODEL I, 5.0.3A

The following patches are for Model I, Version 5.0.3A. If you have an earlier version, you should send in for an update. If your 5.0.3A dates are 11/15/81 or later, then these patches are already installed.

Use the BUILD command to type in the following JCL file and patches. When executing the DO command to compile and execute this JCL, be sure to specify the drivespecs S and D in the DO command line.

```
. FIX503A/JCL update to 5.0.3A
. s = source drive
. d = destination drive
//asign p=RS0LT0FF
patch backup/cmd.#p#:#d# backupm/fix:#s#
patch ksm/flt.#p#:#d# ksmb/fix:#s#
patch twoside/cmd.#p#:#d# twosidea/fix:#s#
patch sys2/sys.#p#:#d# sys2a/fix:#s#
```

```
. BACKUPM/FIX
. This patch will cause LDVR$ to be properly loaded at
. all times
X'5441'=CD C6 55
X'55C6'=E6 07 4F 32 08 43 C9
. EOP
******
```

```
. KSMB/FIX
. This patch will cause KSM to accept only upper case
. letters
X'5467'=18
. EOP
*****
```

```
. TWOSIDEA/FIX
. This patch will correct twoside to work with 5.0.3
D00,99=33
X'52E2'=B2
. EOP
*****
```

```
. SYS2A/FIX
. This patch will increase the timer for check drive
. to 500ms from 275ms
D03,A2=14
. EOP
*****
```

UPDATE NEWS - MODEL III, 5.1.0 A

The following patches are for Model III, Version 5.1.0A. If you have an earlier version, you should send in for an update. If your 5.1.0A dates are 11/15/81 or later, then the patches are installed except for the last two, and you should DO the JCL file starting at @NEW. If your dates are 12/15/81 or later, then all patches are installed.

Use the BUILD command to type in the following JCL file and patches. When executing the DO command to compile and execute this JCL, be sure to specify the drivespecs S and D in the DO command line.

```
. FIX510A/JCL Update to 5.1.0A
. s = source drive
. d   destination drive
//assign p=RS0LT0FF
.
. The first set of patches are for 5.1.0A with dates earlier
. than 11/15/81
patch lbasic/cmd.#p#:#d# lbasicd/fx3:#s#
patch lcomm/cmd.#p#:#d# lcommb/fx3:#s#
patch sys7/sys.#p#:#d# freea/fx3:#s#
patch backup/cmd.#p#:#d# backupc/fx3:#s#
patch format/cmd.#p#:#d# formate/fx3:#s#
patch pr/flt.#p#:#d# pra/fx3:#s#
patch sys0/sys.#p#:#d# sys0f/fx3:#s#
patch patch/cmd.#p#:#d# patcha/fx3:#s#
patch ki/dvr.#p#:#d# kib/fx3:#s#
patch sys2/sys.#p#:#d# sys2b/fx3:#s#
. The following patches are for 5.1.0A with dates
. earlier than 12/15
@NEW
patch sys11/sys.#p#:#d# sys11a/fx3.#s#
patch backup/cmd.#p#:#d# backupd/fx3:#s#
```

```
. LBASICD/FX3
X'5DED'=00
.EOP
****
```

```
. LCOMMB/FX3
X'543C'=F7
.EOP
****
```

```
. FREEA/FX3
L22
x'5276'=20
.EOP
****
```

```
. BACKUPC/FX3
X'545C'=CD E1 55
X'55E1'=E6 07 4F 32 27 44 C9
.EOP
****


. FORMATE/FX3
D06,47=F2 00 4E 3D 4E F4 00 0C 18 07 13 02 0E 1A 09 15
D06,57=04 10 1C 0B 17 06 12 01 0D 19 08 14 03 0F 1B 0A
D06,67=16 05 11 1D 00 00 00 00 00
.EOP
****


. PRA/FX3
.Allows a 0 to be sent to the printer
X'5518'=00 00 20
X'5531'=00 20
.EOP
****


. SYS0F/FX3
.Allows a 0 to be sent to the printer
X'41E5'=38 03 C2 4B 04 C3 C2 03
.EOP
****


. PATCHA/FX3
. This patch will allow patch to work without a closing
. paren
D00,BD=8D 52
. EOP
*****


. KIB/FX3
. This patch will prevent the "type" buffer from being
. filled by the repeat function.
X'546F'=02
X'5476'=02
.EOP
****

. SYS2B/FX3
. This patch will cause check drive to allow at least
. two passings of the index hole before returning with
. a drive unavailable indication.
X'4E7A'=0F
. EOP
****
```

```
. SYS11A/FX3
. This patch will correct the // WAIT in JCL
.
D02,D7=C2 FE 0A C2 5C 4F C5 CD 64 51 71 ED A0 C1 10 F1
D02,EB=7E
D02,F6=23
.EOP
****
```

```
. BACKUPD/FX3
. This patch allows a backup of :1 to :0 (X)
.
D05,87=B3
.
.EOP
****
```

UPDATE NEWS - MODEL 1, 5.1.1

The following patches are for Model I, Version 5.1.1. If your 5.1.1 dates are 12/10/81 or later, then the patches are already installed except for the CMDFILE patch, and you should DO the JCL at @NEW. Dates of 12/15/81 or later have all patches installed.

Use the BUILD command to type in the following JCL file and patches. When executing the DO command to compile and execute this JCL, be sure to specify the drivespecs S and D in the DO command line.

```
. FIX511/JCL update to 5.1.1
. s = source drive
. d = destination drive
//asign p=RS0LT0FF
patch lbasic/ov3.#P#:#D# ov3a/fx1 :#s#
patch backup/cmd.#P#:#D# backupa/fx1 :#s#
patch sys6/sys.#P#:#D# reseta/fx1:#s#
. If file dates are earlier than 12/15, also DO the
. next patch
@NEW
patch cmdfile/cmd.#P#:#D# cmdfilea/fx1 :#S#
```

```
. 0V3A/FX1
. This patch corrects the released LBASIC/0V3 file
. to match the LBASIC/CMD file.
.
D00,7B=7C
D04,D9=16
D04,E1=29
D04,FA=1D
D05,00=36
D05,06=3C
D05,0F=79
.
.EOP
****
```

```
. BACKUPA/FX1
. This patch allows a backup of :1 to :0 (X)
.
D05,87=B3
.
.EOP
****
```

```
. RESETA/FX1
. This patch will cause global reset to turn off the
. @ICNFG vector.
.
D2B,AC=32 03 43 11 02 00 21 00 42 CD 77 47 C2 F7 53 2E 70
.
.EOP
****
```

```
. CMDFILEA/FX1
D04,C8=61
D05,2A=C3 73 59 00 E5 CD 14 03 22 6C 5F CD F8 01 E1 CD 27 59 E5 21 74 58 CD
67 44 E1
.
.EOP
****
```

I WAS AN LDOS BETA TESTER by Tim Daneliuk

This is the story of LDOS 5.1.1, the long awaited upgrade for the TRS-80 Model I. It started rather innocently one fateful day in August. I was to do a product evaluation of LDOS 5.0.2 for a major computer magazine, and had arranged a trip to visit the authors of LDOS who would tell all. LSI Inc. is cleverly concealed on the bottom floor of a building in the scenic Wisconsin countryside north of Milwaukee. After a grueling 2 hour trip from Chicago, I found my desination and proceeded to announce myself. I spent the full day there and was amazed, mystified andagast on the way home. How was I going to describe this product adequately in 40,000 words or less? Clearly, LDOS was a major force in the microcomputer industry and was soon to become the standard of excellence in TRS-80 systems software.

I finished the product review and settled down to enjoy my new-found DOS when the phone rang. It was LSI and they wanted to know if I'd be interested in field testing the new 5.1 version of LDOS on the Model I. Would I! Promptly, the software was sent and I was officially a "Beta Test Site". Surrounded by 1 computer, 2 Disk Drives, 1 Printer, 2 Cats, and 1 Wife, I calmly proceeded to boot LDOS 5.1.1 and make a backup. Fire-extinguishers and paramedics were available should the experience prove overpowering to either the CPU or myself. Neither was required and what follows is a brief overview (by no means complete) of this new LDOS.

LDOS 5.1.1 is not radically different than 5.0.3. Rather there is a set of subtle enhancements to the system which add to its already appreciable power. These were enumerated in the last LDOS quarterly so I'll only deal with my particular favorites.

One of the nicest features of 5.1.1 is the SYSRES command. This SYSTEM feature allows certain /SYS files to be loaded into high memory. This speeds up DOS operation since overlays are not being called as often. Also, certain features are added if SYRES is used. For example, by SYSRESing SYS2, SYS3, SYS8, and SYS10 into memory, you can do BACKUP by Class between two non-system diskettes on a two drive system. Another application of SYSRES is the single drive system user. By residing /SYS files into memory they can be purged from the disk and that space is now available for user files.

Several new features have been added which make LDOS even easier to use than before. It is now possible to enter a command with parameters and leave off the closing parenthesis. Directories are alphabetically sorted which makes finding a particular file in a listing a lot easier. As with Model III LDOS, 5.1.1 has a MINIDOS filter which allows the commonly used DOS commands to be entered in one keystroke sequence. Another favorite command of mine is the software write-protect feature. This allows you to prevent the operating system from writing to certain drives without having to put a write-protect tab on the disks. LBASIC has also been enhanced. One of the nicest features here is a CMD function which sorts a string array.

Should you upgrade to 5.1.1? In all probability, yes. Though the improvements found in this new release are subtle, they are not trivial. I've found my efficiency using LDOS dramatically improved with 5.1.1 (no mean feat considering how efficient 5.0.3 was to use). In my estimation, 5.1.1 has the single greatest feature of being even more "user-friendly" than previous LDOS releases, and certainly than other so-called "advanced" TRS-80 operating systems.

ITEMS OF GENERAL INTEREST

Those of you trying to use KSM to set up control sequences to be sent to a lineprinter have need of the semicolon character as other than an imbedded carriage return. Use one of the KSM/FLT patches to change the semicolon to a character of your choice. The patch will be a two part patch; changing the character and character offset value. The value (nn) is the ASCII value of the character to act as the embedded <ENTER>. The offset (oo) is the value (nn-X'0D'). Both (nn) and (oo) should be represented as hex digits.

```
   Model I, Version 5.0  - X'5490'=nn:X'5494'=oo
   Model III, Version 5.1.0 - X'555B'=nn:X'555F'=oo
   Model I, Version 5.1.1 - X'558E'=nn:X'5592'=oo
```

Those of you who don't have lower case hardware installed may run into a perplexing problem when doing comparisons on keyboard inputs. If KI/DVR is set, the keyboard will automatically be in the lower case mode, even though upper case is displayed on the video. Characters from the keyboard will not match up when compared to upper case characters. Be sure to do a <SHIFT><0> and then sysgen the system to lock yourself in the caps mode.

Model I owners should be sure NOT to use configuration files created under 5.0.x with version 5.1.x!

LDOS provides the <SHIFT><BREAK> key sequence as a means to restart a timed out drive. However, this will only work if the interrupts are on. Since FORMAT and BACKUP both disable the interrupts, a timed out drive during either procedure will hang up the system. Some versions of the Radio Shack interface (both the model with the buffered cable and the one without) have an official Radio Shack modification to increase the drive select time by changing a resistor value from 200k to 270k ohms. Also, as mentioned on page 49 of the December 80-Microcomputing, the capacitor used with the resistor to provide the select timing can go bad and may have to be replaced. The article gives component numbers for the capacitor of C-48 (C-62 in the newer interfaces) and C-12 for the LNW interface. The original value was 33mF, and the article recommends that a 47mF or a 68mF, 16 volt, tantalum capacitor be used to replace it. If you are experiencing drive timeout, you should have these two components checked.

The new 5.1 manual mentions a version 5.1 ROM for LX-80 owners. Hopefully, this ROM will be available in early 1982. Among other things, this ROM will allow software write protect, correctly pick up the DAM of the directory (eliminating the need to log a drive), and provide more head settling time for 8" drives. For details on upgrading your ROM, contact LOBO Drives directly.

Single drive owners of 5.1 can get a directory of visible files on a data disk by using the following procedure. Use the SYSTEM (SYSRES=) command to reside SYS modules 1 and 10. Filter the keyboard with the MiniDOS filter. Then type in the MiniDOS command <CLEAR><SHIFT><Q>. When the Q prompt appears on the screen, insert the data disk in drive 0 and press enter.

Model I LDOS owners upgrading 5.0 disks to 5.1 may have a problem after doing a BACKUP (OLD) from a 5.1 system disk to a 5.0 system disk. The 5.1 version SYS6 increased in length to 50 sectors. Depending on the location of SYS6 on the 5.0 disk, it is possible that it will get broken into 3 extents by the backup. This will result in certain library commands not working. If this is the case, try killing SYS6 on the 5.0 disk and then backing up (not COPYing) SYS6 from the 5.1 to the 5.0 disk. If this does not work, it will be necessary to make a mirror image backup of the 5.1 disk and then move the necessary files from the 5.0 to the 5.1 disk.

LDOS FILTER PACKAGE

LSI is proud to announce the first in a series of extension packages for the LDOS product line. This package is FILTER oriented, and will contain many useful modules. This section of the newsletter will detail the filters that will be incorporated into this package.

XLATE/FLT........ A complete translation filter system, for input and output. Included are a complete EBCDIC translate system and also a DVORAC keyboard translator. The user can very easily build any other translate tables that are needed for special use.

LISTBAS/FLT......	A filter which will format the output of a Basic program. All program lines which contain multiple statements (i.e. statements separated by colons) will have their appearance reformatted when displayed. All new statements encountered in a physical program line will be displayed on a new line, and will be indented. Also, special formatting will be done to change the display of PRINT statements and statements involving parentheses.
STRIP7/FLT.......	Strips bit seven (the high bit) off of each character.
STRIPCNT/FLT.....	A filter which will replace an output character above X'7F' or below X'20' with a pound sign (#).
MONITOR/FLT......	A filter similar to STRIPCNT/FLT, with the exception that characters less than X'20' will be displayed as a percent sign (%) followed by an ASCII representation of the actual character value + X'41'. in addition, characters greater than X'7F' will be displayed as either an <UP ARROW> or a <LEFT BRACKET>.
TITLE/FLT........	A printer filter that will print a user defined title after each Top-Of-Form character (X'0C') is encountered.
UPPER/FLT........	Converts every alphabetic character (a-z) to UPPER case.
LOWER/FLT........	Converts every alphabetic character (A-Z) to lower case.
SLASH0/FLT.......	Will cause a printer that is capable of backspacing to do a backspace and type a "/" over every 0 (numeric zero) that is encountered.
TRAP/FLT.........	Will trap and throw away a certain character each time that the character tries to go through the filter. Any character (00 - FF) may be "trapped".
LINEFEED/FLT.....	Either add or remove a linefeed after each carriage return.
PAGEPAWS/FLT.....	Will pause after each Top-Of-Form character is printed and wait until <ENTER> is pressed to continue.
CALC/FLT.........	A keyboard filter to perform Hex/Decimal/Binary conversion. Hex addition and subtraction may also be done.
REMOVE/CMD.......	Removes each occurrence of a specified byte from a file.

F E D - The Ultimate File Editor - by Doug Kennedy

FED is an all-purpose, screen oriented file editor to be used with the LDOS operating system. Its wide range of capabilities make it excellent for the advanced user, but its simplicity makes it easy to use for the novice user. The editor supports both Model I and III, upper and lower case, single or double density, or anything readable by LDOS. Some things to clarify:

This is a file editor, NOT a file copier, text editor, or word processor. It is for displaying, printing, and modifying existing files. Fed works on a file level, not a track/sector level.

FED was not designed to repair damaged disks, or recover lost files, but it could be used to do so by the experienced LDOS user. You cannot create or extend files with FED, only modify existing ones. FED is intended to run with the LDOS operating system only.

FED is available for $40.00 from Galactic Software, 11520 N. Port Washington Rd, Mequon, WI 53092. Here is a brief description ofFED's capabilities:

1) Complete editing capabilities, including Hexadecimal and ASCII modifying. Direct disk patching becomes a simple matter with FED. It is possible to write machine language code directly to disk. Small changes in files can be made instantly. No need to read in a large source file and reassemble just to change one character.

2) Record advancing, backspacing, and positioning. Move through files quickly either forward or backward. The user need not know how the file's directory records are stored, how many sectors pergran are on the disk, or how many granules per cylinder the disk has, or what density is used. Just know the filespec and password (if it has one).

3) Global ASCII and Hex string searching, with a command to position to the next occurrence of that string. FED searches the entire file, not just one record like most editors. It allows searching for 30 character ASCII strings (upper/lower case), and 30 digit (15 byte) Hex strings. FED saves that string and you can go to the next occurrence of that string from the currently displayed position in the file.

4) Locate a Hex address in a load module format file, and calculate the load position of a specified byte. A MUST for assembly language programmers. No more tedious hours spent going through a load module file manually to locate or change a byte at some memory location. Just type in the load address and FED points you at that byte. Another extremely powerful feature is the reverse of the address location command. FED calculates where in memory a specific byte pointed to by the cursor will load. With these two features it is possible to write machine language routines directly to disk. Direct patches are made quickly and easily. Even X-patches are easily installed by the experienced programmer.

5) Listing of a file or individual records to a printer, with many safeguards added to make it difficult to LOCK-UP the system if a printer is deselected, out of paper, etc.

6) Includes a 256 byte display mode, and an extended 128 byte mode. Editing utilities in the past allowed for 256 byte displays only. By using this format exclusively, the variations of an ASCII/HEX display are limited. But by having a 128 character display mode, the extra space makes it more visually appealing. The filespec, drivespec, record number, input & output can be displayed horizontally instead of vertically.

ELSIE - THE CONTENTED COMPILER by JimFrimel

LC, LSI's soon-to-be-released C compiler (nicknamed Elsie), will give LDOS users many new ways to "milk" their system. LC provides a substantial subset of the C programming language of Bell Labs fame, the main language used under the UNIX (TM Western Electric) operating system. LC was written to be compatible with UNIX programs. The C standard library is supplied with the compiler. LC programs which use the standard library can be compiled and run under UNIX. Programs written under UNIX which only use statements implemented by LC are also portable to LC and LDOS. A large amount of existing software, both commercial and public domain, will be directly useable by LC owners.

For those of you who are not familiar with the C programming language, here is a brief introduction. C is a structured, portable language. A C program is a collection of functions arranged hierarchically (they call each other). C functions can be recursive and re-entrant, as local variables are created and stored in a stack (in LC, the Z80 machine stack). All machine-dependent features needed, such as I/O, are not implemented in the language; rather, they are placed in the standard library. Thus, only the implementation of the standard library changes from installation to installation, and C programs are written in machine-independent ways. The language itself provides ways of expressing program structure, and of giving arithmetic and logical expressions. C is known for having one of the most powerful expression capabilities available in any language. C statements supply the WHILE, DO-WHILE, FOR, IF, and SWITCH-CASE constructs. C also provides powerful pointer capabilities to enable direct access to memory and variable storage.

LC is an integer-only implementation of C, which provides all C statements except "struct", "union", "goto", and "typedef". All data types except "float" and "double" are implemented; "long" and "short" declarations are accepted, but 16 bit fields are used for all integers. LC accepts multiple input files, with 4 levels of nesting for "#include'd" files. The compiler generates a Z-80 source file which is then assembled and linked to the standard library to generate the executable program. The LC standard library provides such functions as standard I/O redirection, dynamic memory allocation, automatic standard I/O opening and closing, and program chaining. In addition, functions specific to LDOS and the TRS-80 are supplied in an installation library, and provide access to graphics and LDOS entry points.

LC supports separate compilation; programs may be compiled in segments, and frequently used functions can be pre-compiled. External variables are supported with the "extern" statement.

Optionally, the compiler will generate external declarations automatically without any "extern" statements. Users can create their own libraries of commonly used functions - user libraries - which need not be compiled for each subsequent use. Elsie supports both the Microsoft Macro-80 assembler and the MISOSYS EDAS disk editor-assembler. Under M80, the standard and function libraries are in relocatable format; under EDAS, the standard library is implemented as a partitioned data set composed of relocatable object deck modules which use an LC-supplied linker program while the user function libraries will be in Z-80 source form, to be assembled along with the program using the "*get" file include facility of EDAS. Supplied with LC is a program, "blib", which allows users to create their own relocatable load module libraries under the Microsoft package.

To give you an idea of how simple LC programs can perform complex functions, here is a simple example program:

```
#include stdio/csh  /* standard I/O definitions */
/*  XFER - copy standard input to standard output   */

int  c, bytes, lines;
FILE *fp;

main()
{    while( (c=getchar()) != eof)
     {      putchar(c);
            ++bytes;
            if (c == eol) ++lines;
     }
     fp = fopen("*do","w");
     fprintf(fp, "%d characters , %d lines were copied", bytes, lines);
}
```

This program just copies standard input to standard output, then gives a character and line count to the user on the video display. Here is an explanation of the program, going through the program from top to bottom. The standard header file, "stdio/csh", is processed as if it were part of the program source file, and contains system dependent and machine dependent information, such as the constants "eol" and "eof" (end-of-line and file). "FILE", also defined in the standard header file, provides a system-independent way of defining a file pointer variable. "main" is always the name of the main function, which is the top of the hierarchy of the structured C program. The brace characters ({,}) group statements into compound statements. Thus, the body of the function is one compound statement. The "while" statement provides looping for the program. The conditional expression in parentheses shows the power of complex C expressions. The innermost expression, "c=getchar()", gets a character from the standard input, and puts it into the variable, "c". [All functions in C designate parameters within the appending parentheses and the parentheses are always required regardless of the absence of any parameter.]

The value of the character is then compared to the constant, "eof", to determine if end-of-file has been reached. Within the loop the character "c" is output, and the byte counter is incremented. The line counter, "lines", is incremented if an end-of-line character was read. Finally, when the loop is exited, the video display is opened as an output file and "printf" is used to print statistics on the display.

This normally trivial program becomes a powerful utility when LC's standard I/O redirection feature is invoked. Standard I/o redirection allows the user to specify at execution time where standard input and output are to be read and written.

Once XFER is compiled, assembled and linked, typing the following line into LDOS:

```
XFER
```

will copy the keyboard to the video display, the defaults for standard input and output. However, both input and output may be redirected, using the > (for output), and < (for input) characters. Thus, the command:

```
XFER <xfer/ccc
```

will list the contents of the file, "xfer/ccc", on the video display. Since devices can be opened as files under LC and LDOS, the command:

```
XFER >*PR
```

will let you type directly from the keyboard to the printer. The command:

```
XFER >*pr <xfer/ccc
```

lists the source code of the XFER program on the printer. And, as you could probably guess by now, files can be copied by redirecting both input and output to and from disk files, as follows:

```
XFER <TEST/JCL >TEST/BAK
```

LC will probably find its niche most readily among those who would like to program at the "systems" and utilities level, but really don't relish assembly language programming. Those LDOS users who already write assembly language programs will find that LC increases their productivity for system tasks, and executes fast enough for most system work. LC readily interfaces with assembly language for critical portions of code. However, users will find that this is usually not necessary.

LC will be available for shipment in February of 1982, if all goes well, for a cost of $150.00. Orders may be placed with the domestic US LSI distributor nearest you.

MISOSYS ANNOUNCES PARTITIONED DATA SETS by Roy Soltoff

Partitioned Data Sets (PDS) are not new to LDOS. The two library files, SYS6/SYS and SYS7/SYS, are PDS structures.Katzan, in OPERATING SYSTEMS, A

PRAGMATIC APPROACH, defines a partitioned data set as "a data file that is divided into sequentially organized members."Katzan further states, "Each PDS includes a directory that points to the beginning of each member. Data sets of this type are most frequently used to store object programs - each member corresponds to a single object program. The PDS as a whole is referred to as a library. Operating system libraries and user libraries are stored in this fashion." This definition describes exactly the two LIB files in LDOS.

Some LDOS users have already explained how to find the member "number" corresponding to a LIB command by searching through the PARM table of SYS1/SYS. This table is used by the LDOS Command Interpreter, which parses the command line and checks if the "filename" entered by the user matches up with a LIB command listed in the table. When a match is found, linkage is established with the system loader in SYSRES to denote the LIB file and the specific member entry satisfying the LDOS command entered.

The system loader, in reading the LIB file, discovers that it is a PDS and reads through the member map table stored in the LIB file. This map table is a directory containing information relevant to each member in the file. Once the system loader finds the appropriate entry, it positions the file to the starting point of the member, and then loads and executes the module.

The PDS structure has provided a technique for combining separately executable object programs into one file, thereby saving directory slots. It also saves time by not having to load an entire 10K-15K file just to get a few hundred bytes or a few thousand bytes of program loaded if all LIB commands were just one big file. The overhead of having to read and search the member directory is minimal. This technique has been used for years on mainframe and mini computers. LDOS is the only known DOS supporting PDS structures on the TRS-80.

Up until now, only the system library has been blessed with this support. However, MISOSYS has now implemented User PDS structures. The PDS command can be used to create custom user libraries. A library could be a collection of a dozen utility programs - all stored under one name but directly executable by specifying the library name followed by the member name. Consider for a moment, that I have built a library containing CMDFILE, DSMBLR, FED, BINHEX, EDAS, and XREF. The library name MYLIB was chosen. I can then execute EDAS by entering:

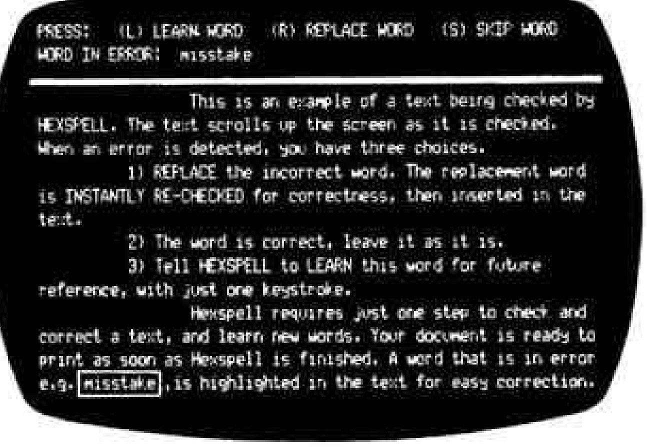
```
        MYLIB(EDAS)
```

at the LDOS ready prompt. If I wanted to build a custom LDOS command library, I could use CMDFILE to extract DIR, COPY, KILL, DEBUG, ROUTE, and RESET from SYS6/SYS and SYS7/SYS and build them into a user SYSLIB. Then I could kill off SYS6 and SYS7 which would save about 15K from my "custom" SYSTEM disk. When I wanted to do a directory, I would only need to type:

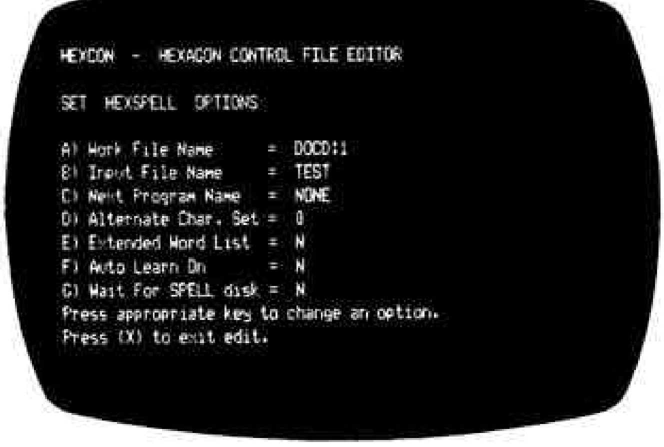
```
        SYSLIB(DIR) :2 (A,I)
```

to achieve the same result as if I had typed DIR :2 (A,I) on a regular SYSTEM disk. Albeit I could have named my user library, "S" and save the entering of five characters each time I wanted to execute a member of the library. That would let me use "S(DIR)"! How's that for you BASIC/S fans? ... cont. page 33

Looking for a
SPELLING CHECKER?



```
PRESS:   (L) LEARN WORD   (R) REPLACE WORD   (S) SKIP WORD
WORD IN ERROR:  misstake
------------------------------------------------------------
                  This is an example of a text being checked by
HEXSPELL. The text scrolls up the screen as it is checked.
When an error is detected, you have three choices.
          1) REPLACE the incorrect word. The replacement word
is INSTANTLY RE-CHECKED for correctness, then inserted in the
text.
          2) The word is correct, leave it as it is.
          3) Tell HEXSPELL to LEARN this word for future
reference, with just one keystroke.
                  Hexspell requires just one step to check and
correct a text, and learn new words. Your document is ready to
print as soon as Hexspell is finished. A word that is in error
e.g. misstake ,is highlighted in the text for easy correction.
```



```
HEXCON  -  HEXAGON CONTROL FILE EDITOR

SET  HEXSPELL  OPTIONS

A) Work File Name       =  DOCD:1
B) Input File Name      =  TEST
C) Next Program Name    =  NONE
D) Alternate Char. Set =  0
E) Extended Word List   =  N
F) Auto Learn On        =  N
G) Wait For SPELL disk =  N
Press appropriate key to change an option.
Press (X) to exit edit.
```

Get more than just a spelling checker —

HEXSPELL 2™

the EVERYTHING checker.

HEXSPELL CHECKS:
Product codes • zip codes • French words • engineering formulae • typesetting commands • German words • numbers • addresses • chemical formulae • Greek words • names • amounts • printer commands • Japanese words • acronyms • dimensions • mathematical formulae • values . . . and of course plain old English words.

Hexagon Systems is proud to announce another first in text checking — an everything checker. Hexspell 2 checks not only dictionary words, but learns and checks codes, formulae and numbers which are so essential in many commercial and technical documents. With Hexspell 2 you define which characters make up a "word", then teach Hexspell the new "words" it needs to check your text. This advanced system builds on the unique features of the original Hexspell (the first TRS-80 spelling checker).

When we introduced Hexspell nearly a year ago, it was the world's first adaptive learning spelling checker. Hexspell constantly adapts its wordlist to your usage. No complicated editing of the wordlist. Hexspell remembers words as long as you use them. If you never use a word again Hexspell will eventually forget it to make room for new words. Extending this feature in Hexspell 2 gives you a text checker that can learn to check everything that you write.

Hexspell 2 is the product of nearly three years research and experience. Unlike other microcomputer spelling checkers Hexagon Systems didn't simply rewrite the multi-step batch process found on large computers, we invented a simple one-step process that capitalizes on the interactive strengths of the personal computer. Hexspell's simple one-step process makes it practical for daily use on all your letters and documents.

HEXSPELL VERSION 1 USERS — GET THESE HEXSPELL 2 ENHANCEMENTS:
Programmable character set · Disk error trapping · Extendable word list, up to 50,000 words · Speed control · Longer words, up to 40 characters · Auto learn function · User settable system options · Control file editor · Built in lower case driver.

Hexspell 2 upgrade for registered owners of Hexspell Vers. 1
— $35 from Hexagon Systems or your nearest dealer.

US $99.
Manual $12

VISA

Hexspell requires a TRS-80 Mod I or Mod III with 2 drives and 48K. Works with most word processors.
TRS-80 is a trademark of Tandy Corp.

HEXAGON SYSTEMS

P.O. Box 397, Station A
Vancouver, B.C. Canada V6C 2N2
Telephone (604) 682-7646
Electronic Mail-Micronet 70235,1376

Expand your TRS-80*

With the LOBO LX-80 Expansion Interface

Now you can realize all the power and Potential of your TRS-80*, Model 1. If it's Add on memory you need, your LX-80 can Accommodate up to four 5 ¼-inch, single- or double-density 35, 40, or 80 track mini-floppies, four 8-inch floppies (single or double sided), and up to eight Winchester fixed disk drives (5 ¼", 8", 14").

LOBO's powerful new LDOS™ operating System, provided with your LX-80, allows For the use of any eight drives, in any Combination, single or double density. And there's more ... lots more. There are Two parallel ports (standard) two serial ports (optional), a keyboard ROM override switch, and a 32K memory expansion (optional). Send for a free LX-80/TRS-80 cost performance comparison chart.

For the full story on how the LX-80 can expand your TRS-80, see your nearest LOBO dealer, or write or call:

*TRS-80 is a registered trademark of Radio Skack, a Tandy Company.

LOBO drives INTERNATIONAL

LOBO DRIVES INT'L
354 South Fairview Ave.
Goleta, CA 93117
(805) 683-1576

If you *liked* MICROPROOF, then you'll *love* SON OF MICROPROOF:

ELECTRIC WEBSTER T.M. *DICTIONARY SOFTWARE*

The ultimate spelling checker.

IMPROVED:

- One-step proofing and correcting.
- Lists errors to screen or printer — can display errors in context.
- Can display dictionary to locate correct spellings.
- New precise symbolic dictionary will not miss an error.
- Remarkably compact (50,000 word dictionary will fit on one 5 inch disk).
- Even FASTER than MICROPROOF (formerly the fastest available).
- Simple Grammatical Checking (Optional Feature).
- Hyphenates automatically (Optional Feature available for some Word Processing programs).

```
SELECT APPROPRIATE RESPONSE:
CORRECT MISSPELLED WORD:        ENTER CORRECT WORD
LEAVE WORD "AS IS":             HIT <ENTER> KEY
DISPLAY WORD IN CONTEXT:        ?
DISPLAY DICTIONARY:             @
ADD WORD TO DICTIONARY:         +
EXIT:                           !
WORD:        (Your error)
RESPONSE:
```

Here's what the reviewers have to say about MICROPROOF:

"I have already found that the use of (MICROPROOF) has greatly enhanced the quality of my letters and reports. This is a very useful product and should be obtained by anyone who uses a word processor."
Michael Tannenbaum, CPA
80 Microcomputing, August 1981

"The summary review of this program? One word — Excellent. I highly recommend it for anyone using a word processor for any need — articles, manuals, reports, and even letters of substantial length."
A. A. Wicks - Program Previews
Computronics, September 1981

In a comparative review of proofreading programs (with smaller dictionaries), MICROPROOF was found to be considerably faster than all the others, when tested against a 400 word sample document.
Phillip Lemmons
BYTE Magazine, November 1981

"(MICROPROOF) operates with good speed and efficiency. A 1500 word document took 26 seconds to load, process, and proof when the program was run on a TRS-80 Model II under CP/M."
"Once the program is integrated, it is very friendly and any person able to use a word processing program can master it in moments."
Frank Derfler
Info-World (to appear shortly)

EASY TO USE: Prepare your text on any Z-80 based microcomputer, using any of a number of popular word processing programs. When you are finished, enter the appropriate command, and ELECTRIC WEBSTER proofreads your document, displaying misspellings and typos on the screen. Then correcting ELECTRIC WEBSTER can display each error separately, requesting you to enter the correct spelling for each. You are also given the option of displaying errors in context or adding words to ELECTRIC WEBSTER's 50,000 word vocabulary. If you do not know the correct spelling you may also ask ELECTRIC WEBSTER to look it up for you and display the dictionary. Finally, ELECTRIC WEBSTER corrects your document. All in less than a minute.

There are other proofreading programs available to choose from. Since MICROPROOF became available in December of 1980, a number of companies have announced programs with small dictionaries. It took us almost two years to develop MICROPROOF. During that time we were able to compress our full 50,000 word dictionary into a manageable size (fits on one single density 5¼ inch disk). We were able to design a proofing program which operates remarkably fast. And now with ELECTRIC WEBSTER we have perfected the ultimate spelling checker.

SPEED is the single most important factor in a dictionary program. All dictionary programs will find your potential errors but if the program is too slow, you are not likely to use it. ELECTRIC WEBSTER's speed is outstanding. It can proofread a several page letter in 20 seconds.

LOW PRICES: Standard MICROPROOF is available for $69.50 (TRS-80). Standard ELECTRIC WEBSTER is available for either $89.50 (TRS-80 Model I or III, Apple) or $149.50 (CP/M, TRS-80 Model II and all others). The optional correcting feature can be added at any time for an additional $60. Correction feature can be ordered with patch to integrate with your word processing software. For each patch, optional Grammatical Checking feature, or optional Hyphenation feature, add $35. (Integration patch not necessary for Wordstar.™

ELECTRIC WEBSTER's FULL 50,000 WORD VOCABULARY saves you time and allows you greater confidence in the lists of potential errors that ELECTRIC WEBSTER identifies. The mini-dictionary programs, with their 10,000 and 20,000 word vocabularies, have many correctly spelled words omitted from their vocabularies. Consequently, they identify as potential "errors" many words that are actually spelled correctly; five to ten times as many such words as does ELECTRIC WEBSTER. So, when you use ELECTRIC WEBSTER, you will have far fewer extra words to evaluate, a major time savings. There will be less need to look up words in order to verify that they are in fact spelled correctly. The extra 30,000 words in ELECTRIC WEBSTER's vocabulary assures you confidence in the error lists that ELECTRIC WEBSTER generates.

NOW ALSO AVAILABLE FOR APPLE!

See your local microcomputer dealer or write to:

CORNUCOPIA SOFTWARE

Post Office Box 5028, Walnut Creek, California 94596 • (415) 524-8098

80-U.S.

THE TRS-80 USERS JOURNAL.

If you own a TRS-80® Model I, Model II, Model III, the Color Computer, or the new Pocket Computer, YOU NEED 80-U.S.

80-U.S. Journal

Programs for your enjoyment and enlightenment! Every issue contains several BASIC or MACHINE language program listings. It contains BUSINESS articles and program listings. No matter where you are, there is something for YOU in the Journal!

and...

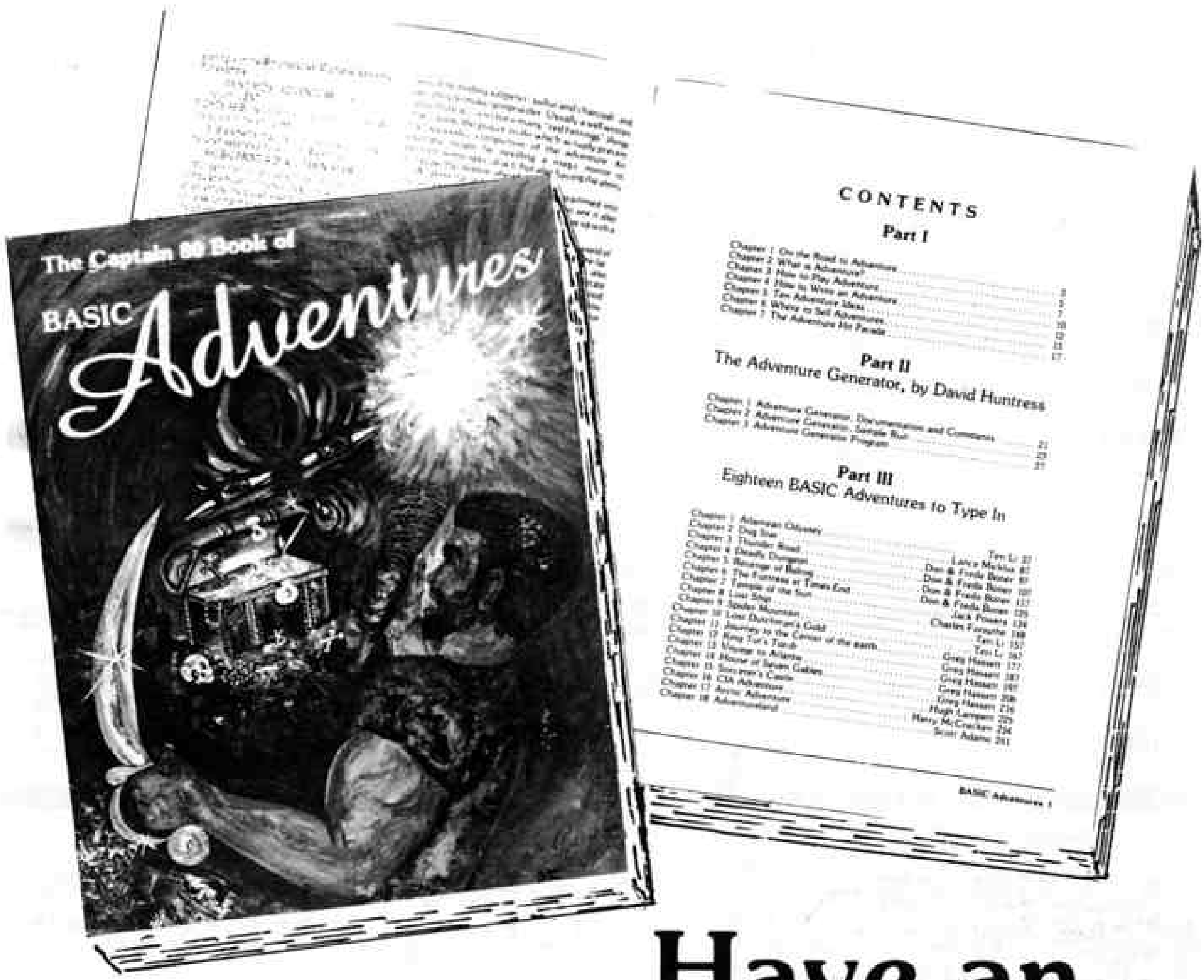
The Journal contains reviews of hardware and software. Our "Evaluation Reports" will help you make the best choice in selecting additions to your system.

80-U.S. is the most informative MONTHLY TRS-80 magazine. Subscribe now for only $16 a year and you will save $20 off the cover price and you will be receiving a wealth of useful information for *YOUR TRS-80 Microcomputer!*

Send to:
80-U.S. Journal, Dept. L
3838 South Warner St.
Tacoma, WA 98407
(206) 475-2219

Name__________
Address__________
City__________
State, Zip__________
Visa/MC__________
Exp.Date________ Payment Enclosed______

1 Yr. $16, 2 Yrs. $31, 3 Yrs. $45
Canada & Mexico $25 per year
Foreign Surface $30 -- Airmail $72 per year



CONTENTS

BASIC Adventures 1

Have an adventure!

- ***The first book of its kind!***
- ***Just $19.95 (+ 2.05 s/h)***
- ***Contains a unique adventure generator!***
- ***Famous adventures by famous authors (see table of contents, above).***
- ***See it at your local dealer, or order direct from:***

80-Northwest Publishing Inc
3838 South Warner Street
Tacoma, Washington 98409
(206) 475-2219

Dealer inquiries invited

Please send _____ copy(s) of the ADVENTURE BOOK for only $19.95 each, plus $2.05 shipping and handling. Check, M.O., VISA/MC# enclosed. COD orders accepted.

Name ____________________

Address ____________________

City ____________ *State* _____ *Zip* _______

VISA/MASTERCARD# ____________________

Exp Date __________ *Signature* __________

Send to: 80-Northwest Publishing Inc. 3838 South Warner Street, Tacoma, Washington 98409

LOGICAL SYSTEMS ORDER FORM

This order form can be used to order any of the Logical Systems products listed below. To place an order, fill in the quantities for each product. Shipping and handling will be listed for each individual product. Orders Should be sent to the LSI Distributor nearest you. If the distributor is in The same state, add the proper sales tax.

Company Name:______________________________ Individual:________________________

Address:___________________________________ City:______________________________

State:_______ Country:___________________ Zip Code:__________________________

LDOS Serial Number:_______________________ Phone No.:_________________________

Ship Via: F/C Mail ( ) UPS ( ) UPS AIR – Add $5.00 ( )

Method of payment: COD ( ) Check/Money order ( ) Master ( ) Visa ( )

Credit Card #:______________________________________ Expires: ________________

Cardholder Signature:___________________________________ Date: _______________

Qty	Description	Net Each	U.S. & Can.	Foreign	Total Amount
	LDOS Model I 5.1.1	169.00	6.00	25.00	
	LDOS Model III 5.1.0	169.00	6.00	25.00	
	FILTER PACKAGE	60.00	4.00	8.00	
	LC (LDOS “C” Language)	150.00	6.00	20.00	
	LED (LDOS TEXT EDITOR)	30.00	4.00	8.00	

Distributors:

ialactic Software Ltd
.1520 N. Pt. Washingtor
lequon, WI 53092

Lobo Drives Intl.
354 S. Fairview
Goleta, CA 93117

IISOSYS
904 Edgehill Dr.
lexandria, VA 2230

Molimerx
1 Buckhill Rd, Bexhill
Sussex, ENGLAND

Sub total: ____________

S & H: ____________

Sales Tax: ____________

Total Due: ____________

o not send orders directly to LSI. Send to the distributor nearest yo
hecks or money orders must be made out to the distributor, NOT TO LS

TRS-80™ PROGRAMS SUPER SOFTWARE AT SUPER PRICES

THE ULTIMATE DISK-OPERATING SYSTEM

LDOS™

A completely documented 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. Full customer service provided with a toll free 800 number. Excellent documentation in a tab indexed manual (over 250 pages). Call or send for more information today.

Available for Model I or III . . . . only $169.00

LDOS™ is a product of Logical Systems Inc.

ALSO AVAILABLE

EDAS 3.5 Model I and III. . . $79.00
EDAS 5.0 Model II. $179.00
Host Model II was. $199.00
now. $99.00

BUSINESS SYSTEMS

INVENTORY MASTER SYSTEM

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. More information available upon request.

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

galactic software ltd.
11520 North Port Washington Road
Mequon, Wisconsin 53092

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. Series II is compatable with Series I data files.

Model I system will handle up to 600 names.
Model III system will handle up to 1200 names.
Either system available for only. $159.00
Also available for the Model II. $199.00

THE ULTIMATE IN MAILING SYSTEMS MASS/MAIL SUBSCRIPTION SYSTEM AVAILABLE FOR MODEL II.

"I think the system will stand the test of time . . . it is probably the most sophisticated system around for the micros today"
Mike Schmidt (EDITOR) *80-US Jan/Feb pg. 94*

Galactic's Mass/Mail System was designed with the needs of the large mailing list in mind. This system has capabilities of handling up to 10,500 records, with access times of less than 15 seconds and usual access of about one second. All adds, deletes and edits are instant for the operator and

MAILING SYSTEMS

are then completed later in a "batch mode". The System comes complete with extensive documentation and ongoing support. . . . .

Contact Galactic Software for detailed specifications for your exact needs. Available only for the TRS-80 Model II.

ULTRA-TREK GAME

Can you master this trip through the galaxies?

Climb aboard the Star-Ship Enterprise and head for a universe of unknown encounters, enemies and stellar battles. As commander, will you be able to keep the Enterprise and its crew en route and on patrol? Can you destroy the hostile base? Order today.

Model I or III version only $19.95

All of Galactic's products have a one year limited warranty with customer support included with each purchase. Warranty information is available upon request in writing.

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

VISA master charge

Can't wait? call
414/241-8030

TRS-80 A Trademark of Tandy Corp.

```
MISOSYS MISOSYS MISOSYS MISOSYS MISOSYS MISOSYS MISOSYS MISOSYS MISOSYS
I                                                                     Y
S                                                                     S
O                                                                     M
S MISOSYS announces a special offer to LDOS owners.  Now you can pur- I
Y                                                                     S
S chase the same ASSEMBLER used to create LDOS – but a special one  O
M                                                                     S
I time only price reduction.  EDAS 3.5.2 is available for $10 off ofY
S                                                                     S
O the regular  price of $79 when purchased  direct from MISOSYS usingM
S                                                                     I
Y the coupon included with this  advertisement.   EDAS is a disk onlyS
S                                                                     O
M assembler which produces  absolute core-image code from one or moreS
I                                                                     Y
S source disk files and/or memory buffer. Supported features are conM
O                                                                     I
S ditional assembly, paged & titled listings, global search & replaceS
Y                                                                     O
S 14-character labels, lower case text entry, block move,  cross refS
M                                                                     Y
I erence utility, DIR FREE LIST & KILL from EDAS,  R/W Macro80 filesS
S                                                                     M
O                                                                     I
S User *PARTITIONED DATA SETS* are now possible with the PDS commandS
Y                                                                     O
S The LDOS user will find User PDS structures  useful for customizingS
M                                                                     Y
I SYSTEM disks and other CMD libraries. PDS is available as an add onS
S                                                                     M
O to your LDOS direct from MISOSYS.  Introductory priced at $40 untilI
S                                                                     S
Y March 1, 1982. The PDS structure is described elsewhere in this NLO
S                                                                     S
M                                                                     Y
I                                                                     S
S CONVCPM is the Utility the LDOS     MISOSYS accepts payment in theM
O                                                                     I
S user can use ======================================== form of CashS
Y                                                                     O
S to transfer  | LDOS EDAS DISKMOD DSMBLR CMDFIL TUTIL | Master CardS
M                                                                     Y
I CPM files to |                  o                   | VISA, CheckS
S                                                                     M
O LDOS. Works  | /-\_/-\_/-\/-\   / \   /-\/-\_/-\_/-\ | or COD.  AddI
S                         C O U P O N                                 S
Y with 8” SDEN | \_/-\_/-\_/\_/  / C \  \_/\_/-\_/-\_/ | for S&H, $4O
S                                                                     S
M or OMIKRON’s |               / N O \                | EDAS, $1 allY
I                                                                     S
S 5” CPM disk. | CONVCPM DUTIL / O P U \  THE-B00K PDS | other itemsM
O                                                                     I
S ===>$30<===  ======================================== listed here. S
Y                                                                     O
MISOSYS MISOSYS MISOSYS MISOSYS MISOSYS MISOSYS MISOSYS MISOSYS MISOSYS
```

Okay, what capabilities are included with PDS? The PDS command will itself be a PDS. Members provided will implement the following functions:

o ALIAS - Will provide the capability of defining more than one reference name to a PDS member.

o APPEND - Will append a new member to the existing PUS and update the member directory and map tables accordingly.

o CREATE - Will provide the capability of creating a new partitioned data set with a maximum number of members. The PDS is composed of a front end loader program, a MEMBER directory, a MAP table, and one or more executable object deck files as specified by the user. The object deck (/CMD files) list can be entered either by on-line prompts or via a listing in a data file. The capability of multiple entry points to a single module (separate member names just like COPY and APPEND or SET and FILTER) is implemented only when the data file list option is specified.

o DIR - Will provide a directory of members listing the member name, date of entry, location in the PDS, transfer address, and file space occupied.

o EXTRACT - Will transfer a copy of a PDS member from the PDS to a target diskette as a standard /CMD file. The member will not be deleted from the PDS.

o KILL - Will remove a member from the PDS and compress the file to delete the space previously occupied by the deleted member.

o LIST - Will list a specific member in standard hex format.

PDS is available from MISOSYS, 5904Edgehill Drive, Alexandria, VA 22303. PDS is priced at $50. Orders received prior to March 1, 1982 may take advantage of an introductory offering at $40.

LDOS 5.1 - Enhancements and New Features - by DickKonov

The following pages of the newsletter will describe the differences between LDOS-5.0 and LDOS-5.1. In the last issue of the newsletter, a similar article was written. Since that time, additional features have been added to LDOS-5.1. This article will highlight the most significant changes which have been made to the operating system.

1.> Any LDOS command parameters which requires parentheses may be entered without the closing parenthesis.

2.> The CONV utility has been added to the system. This command will allow you to transfer files created on the Model III under TRSDOS 1.2 or 1.3 to an LDOS diskette. Note that Model I owners who wish to utilize CONV must be capable of reading double density with their system.

3.> Some LDOS drivers, filters and Library commands will reuse their original memory allocations if they are established, turned off, and re-established. They are the SPOOLer, the Keyboard driver (KI/DVR), the MiniDOS Filter, the Printer filter (PR/FLT), and the KSM Filter. In addition, drivers such as PDUBL, TWOSIDE and the above mentioned will only be allowed to be initiated once.

4.> The Keyboard (*KI) now has its own driver. This driver activates the use of the <CLEAR> key, and must be established if advanced keyboard features are to be utilized (e.g. KSM). In addition, the KI/DVR allows you to enter all ASCII characters (0-127) directly from the keyboard. Also, the KI/DVR allows you to establish an extended character set.

5.> The MiniDOS filter has been added. This is an additional keyboard filter which allows the keyboard driver to intercept certain keyboard inputs and immediately act on them. MiniDOS commands are issued by depressing <CLEAR>, <SHIFT> and an alphabetic key. The following functions are available when using the MiniDOS filter: toggle the CLOCK display on or off, enter the system DEBUGger (if activated), display FREE space for all active drives, KILL a file, display a disk's DIRECTORY, send a hex character to the printer, REPEAT the last DOS command, and issue a Top of Form to the lineprinter.

6.> A SuperVisory Call (SVC) table can now be loaded into high memory for use by assembly language programmers. It contains most documented system entry points and routines. It is established using the SYSTEM library command.

7.> Some SYStem modules may be resided in high memory by using the SYSTEM library command. This will allow you to perform certain functions that normally require these SYStem modules to be on a disk. For instance, the user of a two drive system will be allowed to perform a BACKUP by class using non-system diskettes, provided the proper SYStem modules have been made resident. Residing system modules also increases the speed of the system.

8.> Any Physical drive in your system may be established as Logical drive 0. This can be accomplished by using the SYSTEM library command.

9.> For Radio Shack interface owners only, drives may be software write protected. This is accomplished using the SYSTEM Library command.

10.> The parameters BSTEP and CYL have been added to the SYSTEM library command. BSTEP is global in nature, while CYL is used with the DRIVE= parameter. These parameters will allow you to set your own default values for the FORMAT utility regarding the bootstrap step rate and number of cylinders to be formatted.

11.> The SYSTEM library command will also have added to it the parameters DATE= and TIME=. These parameters are used to enable or disable the DATE and TIME prompts on power up or reboot. If not altered, you will be prompted only for the date on power-up.

12.> The SPOOLer will despool at a faster speed, and will work in conjunction with the LSCRIPT patch for SCRIPSIT.

13.> APPEND has two additional parameters. The ECHO parameter will echo characters to the screen if a "devspec to -->" APPEND is performed. The STRIP parameter will "strip" off the last character of the file being APPENDed to before the APPEND is performed. This can be useful if you wish to, for instance, APPEND two SCRIPSIT files together.

14.> The COPY Library command has also had the ECHO parameter added to it.

15.> The DEVICE Library command will display additional information. Included in this is whether or not the diskette is write protected (software or physically), options which are currently active (e.g. KI/DVR, MiniDOS, TYPE, etc.) and SYStem modules which have been resided in high memory.

16.> A "Not" partspec may be specified for some Library commands and utilities. For example, by entering the following command:

```
    DIR -/CMD:0
```

you will be shown a Directory of drive 0 of all files that do NOT have the extension "/CMD". This "Not" partspec is available with the DIR and PURGE Library commands and the BACKUP utility.

17.> A specific date, or a range of dates may be specified when utilizing the DIR and PURGE Library commands, as well as the BACKUP utility. The system will use the file's last modification date to determine whether or not that file is to be included in the specified operation.

18.> Along with the above mentioned modifications to DIR, additional enhancements have been made. Unless specified, all files in a directory will be displayed in sorted order. Also, a MOD parameter has been added to allow you to produce a directory of modified files only.

19.> Along with the above mentioned modifications to BACKUP, two additional parameters have been added. The NEW parameter will BACKUP only those files not already on the destination disk. The OLD parameter will BACKUP only those files already existing on the destination disk.

In addition, if the (QUERY) parameter is specified, a backup by class will always be the result, whether or not the backup is a mirror image. If the (QUERY) parameter is specified, you will be shown the MOD date and flag of each file.

20.> One enhancement was made to the FREE Library command. If the FREE command is given with a drivespec, a FREE space map will be displayed for that particular drive.

21.> The FORMAT utility has had two parameters added to it. The (QUERY) parameter will query you as to the type of FORMAT you require. If not specified, the (QUERY) parameter will default to ON. The (SYSTEM) parameter has been added, and deals with formatting a hard drive. Also, changes have been made to the prompt questions and how these questions are dealt with.

22.> LCOMM has been modified substantially. The modifications made to LCOMM are too numerous to mention here. Two of the most significant enhancements will be discussed.

LCOMM will allow you to input from the keyboard all ASCII characters (0-127). Also, a Dump To Disk (DTD) function has been added. This will allow you to receive a file in a RAM buffer and dump it to disk after the transmission has been completed. This feature is important to Radio Shack interface owners, as it will guard against the random dropping of bytes when receiving files. Many other features have also been added.

23.> The PATCH Utility has been enhanced, and will allow you to enter Direct ('D' type) patches from the command line.

24.> There will be two SCRIPSIT patches included with LDOS-5.1. The SCRIPT/FIX patch will be identical to the SCRIPSIT/FIX patch that was included with LDOS-5.0.

Many new features have been added to SCRIPSIT if the LDOS-5.1 LSCRIPT/FIX patch file is applied. Operation of SCRIPSIT when using the new patch will differ greatly from operating under the LDOS-5.0 patched version. Many of the enhanced LDOS operations (such as the use of theSPOOLer and MiniDOS) will be accessible from the SCRIPSIT environment.

25.> The RS232/DVR driver has had its name changed to RS232R/DVR. Both RS232 drivers have had enhancements made to them.

The parameters BAUD, WORD, STOP and PARITY may be abbreviated to their first characters.

As specified by standard RS232 conventions, a TRUE condition means a logic 0, or a positive voltage. A FALSE condition means a logic 1, or a negative voltage. The RS232 driver will now treat the line condition parameters DSR, CD, CTS and RI in the following manner:

> If specified ON, the driver will observe the lead and wait for a TRUE condition before sending each character.
>
> If specified OFF, the driver will observe the lead and wait for a FALSE condition before sending each character.
>
> If not specified, the lead will be ignored.

In addition, a (BREAK) parameter has been added to the RS232 drivers. This parameter determines whether the driver can set the system BREAK, PAUSE or ENTER bits.

26.> LBASIC has had several enhancements. The time it takes to load or save LBASIC programs has been decreased drastically. Also note that LBASIC does not need to be created by applying patches to Radio Shack Basic; it will be resident on your Master Diskette.

A machine language string sort has been added to LBASIC. This will allow you to sort a string array contained in RAM.

In the CMD"N" function, the last parameter will now represent the last line number to be renumbered. It used to represent the first line number above those to be renumbered minus one.

The RESTORE command has had an optional parameter added to it. Entering the command RESTORE nnnn (where nnnn represents a line number in an LBASlC program) will restore all DATA statements in lines whose numbers are greater than or equal to the line number specified in the RESTORE statement.

The RUN command has had two parameters added to it. These parameters come into play when using the RUN command as an LBASIC program statement which will chain programs together. The (V) parameter will allow you to save any variables which may have been established in one program, and utilize them in another program. The (line number) parameter will allow you to specify a line number dealing with where execution of the program is to begin.

A new command has been introduced into LBASIC. This command is SET EOF# (where # refers to the buffer # associated with an open disk file). This command may be used to reset the End Of File (EOF) marker of a random file. This is a very convenient way of shortening a random file that may contain unwanted records at the end.

27.> The file MOD1/EQU is included on your Master Diskette. It is an equate file, and may be used by assembly language programmers.

28.> The file VC/FIX is also included on your Master Diskette, and is a patch file to be used with VISICALC.

LDOS Supports the T-TIMER (tm) by Roy Soltoff

The T-TIMER, a bus-connected circuit board that provides a calendar and clock in real time, is now supported by LDOS. This board is manufactured by the BAR-W-HARDWARE-RANCH. The T-TIMER keeps the time and date via a MSM-5832 real time clock/calendar chip. It uses a power backup supply of 2-AA size batteries used for when the CPU is turned off. The T-TIMER is currently available for the Model I and plugs into either the expansion port (screen printer port) or the bus connecting the CPU to the expansion interface. It has a bus extender socket. My test version is connected to my LX-80 development system on the CPU. The LX-80 is then plugged into the T-TIMER. This timer has been in use for over two months without any problem. The pleasure of accurate date and time without having to enter date and time on each powerup is great. The only thing I have found wrong with this unit is that when plugged into the CPU bus, my little finger scrapes the edge of the T-TIMER circuit board when I press the TRS-80's RESET button. Perhaps I should just plug it into the expansion port and be done with it. The only bad thing about this device is that now I am spoiled. The office has four machines but only one T-TIMER. Come on BAR-W, get me a Model III version! The Model I T-TIMER is priced at $89 and is available from BAR-W at Box 1631, Hurst, TX 76053. Herewith are the LDOS patches:

```
. TTIMER Model I Version 5.0.1 patch
. Copyright (C) 1981 by Roy Soltoff, All rights reserved
. PATCH SYS0/SYS.WOLVES
D04,72=D8 45 ED 78 0D CD A1 47 ED 78 0D E6 0F 85 12 1B
D04,82=C9 11 43 40 01 C5 03 CD C9 45 10 FB
D0D,D3=21 46 40 01 CA 01 CD AA 4E 06 03 CD AA 4E 01 CC
D0D,E3=0F CD AA 4E EB 13 DB CB E6 03 21 9E 50 20 01 34
```

```
D0D,F3=18 39 ED 78 0D A0 07 57 0707 82 57 ED
D0E,00=78 0D E6 0F 82 77 2B C9
. end of 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
D04,82=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 0707 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
D04,82=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 13 DB CB E6 03 21 B9 50 20 01 34
D0D,6E=18 21 ED 78 0D A0 07 57 0707 82 57 ED 78 0D E6
D0E,7E=0F 82 77 2B C9
. end of patch
```

LDOS Device I/O and Independence by RoySoltoff

A very powerful and frequently used feature of LDOS is its relatively complete device independence. This feature is available through the FILTER, LINK, RESET, ROUTE, and SET commands. It is also available to the Assembly Language programmer due to the correlated structuring of the Device Control Blocks (DCB) as used for byte I/O devices {*KI, *DO, *PR, *CL, etc.} and the File Control Blocks (FCB) as used for all disk files.

Any system that supports total device independent structures should be realized as an OS with possible channel input/output for each distinct device {such as console, error output, data sets, etc.} on the machine side. On the device side, there should exist no discernable transmission difference among all the peripherals attached to the machine. Furthermore, you should be able to easily interconnect any machine channel to any peripheral, possibly to multiple peripherals. The system should also be capable of installing filters (massaging functions) anywhere in the I/O channel. The operating system running on a TRS-80 most resembling this structure is LDOS. Functions lacking from this "total device independent structure" are its inability to filter a channel connected to a data set (disk file) as occuring from the command, "ROUTE *DD TO FILESPEC" and a somewhat simplified scheme in eliminating filters and channel connections once made (i.e. via RESET). These limitations generally stem from the lack of adequate memory available to the operating system in a 48K environment. Consider, for a moment, that UNIX - a most device independent system - takes upwards of 100K for its system implementation.

Enough of the philosophical. Let's begin to examine the methods used in LDOS to implement device independence and reflect on how this ties in with the programming of drivers and filters. You will first need a basic understanding of the data fields associated with the DCB and FCB. Adequate explanations are covered in the technical section of the LDOS manual and will not be repeated here. I will expand on the information where applicable.

Device independence has its roots in what I will call "byte I/O". The term shall apply to any I/O passed through a channel, one byte at a time. You get one byte when you scan the keyboard (albeit early TRS-80s were afflicted with kkeyybboouncce but that was not multiple byte input). One byte is passed to a printer. A data set generally has I/O data blocks of 256 bytes; however, we can certainly characterize a blocked file with a record length of one - that provides byte I/O with a data set channel.

Three primitive routines are available at the assembly language level for byte I/O. Primitive is not used here to imply rudimentary but rather elementary. Just as the atom is considered a basic building block of molecules, these byte I/O primitives can be used to build larger routines. The three are called @GET, @PUT, and @CTL. Their vector entry points are in the Level II ROM. @GET is used to input a byte from a device or file. @PUT is used to output a byte to a device or file. @CTL is used to communicate with the driver routine servicing the device or file. A reasonable illustration of these routines is:

```
;*=*=*
;       Example ROM Byte I/O Routines
;*=*=*
@GET    PUSH    BC              ;Save this register
        LD      B,00000001B     ;Set the mask code
        JR      GO2DVR
@PUT    PUSH    BC              ;Save this register
        LD      B,00000010B     ;Set the mask code
        JR      GO2DVR
@CTL    PUSH    BC              ;Save this register
        LD      B,00000100B     ;Set the mask code
        JR      GO2DVR
        .
        .
GO2DVR  JP      DVRBGN
```

Observe the similarity of these routines. They are each identical except for the value loaded into register B which establishes a mask code. If we examine a few routines that use these primitives as building blocks, the illustration will become more clear. Three other routines are in the ROM having vectors labeled @KEY (scan the keyboard and return the value of any depressed key), @DSP (display a character on the video screen), and @PRT (output a character to the line printer. These appear as follows:

```
@KBD    LD      DE,KIDCB$       ;P/u keyboard DCB
        JR      @GET            ;Go to input
@DSP    LD      DE,DODCB$       ;P/u the video DCB
        JR      @PUT            ;Go to output
@PRT    LD      DE,PRDCB$       ;P/u the printer DCB
        JR      @PUT            ;Go to output
```

Again we discover some interesting similarities among these three routines. First, each loads register pair DE with a pointer to a specific Device Control Block - the DCB assigned for use by the device. Second, where we expect to input a byte, we go to the @GET routine but where we want to output a byte, we go to the @PUT routine.

These are not just whimsical procedures. The rule is that to input a byte from a device, we load register pair DE with the pointer to that device's control block then go to the @GET routine. If we want to output a byte to a device, we point DE to the control block and go to the @PUT routine. Similarly, if we want to "talk" to the driver routine servicing the device (i.e. RS232/DVR), we can use the @CTL vector. Later on I will show exactly how a driver routine can be written to sort out these primitives.

If we go back and examine the three primitives, we notice that each one collects into an instruction that jumps to some common driver beginning. Again, this is a routine that exists in the TRS-80 ROM. Since the Model III routine is different from the Model I routine, both will be shown here. For one reason, it is important to see how Tandy screwed up the Model III device independence and made it more difficult for such an implementation in LDOS. For another reason, it is what forced the limitation of a maximum of four active ROUTES in the Model III. First the Model I routine:

```
     ;*=*=*
     ;       Model I Driver Initialization for hooks
     ;*=*=*
     DVRBGN  PUSH    HL               ;Save register HL
             PUSH    IX               ;Save register IX
             PUSH    DE               ;Transfer the DCB or 0CB
             POP     IX               ;   address to IX
             PUSH    DE               ;Save register DE
             LD      HL,DVRRET        ;Set up returnrestoral
             PUSH    HL               ;  onto stack
             LD      C,A              ;Xfer char to output
             LD      A,(DE)           ;P/u DCB/FCB "type" byte
             AND     B                ;Mask with primitive code
             CP      B                ;Can DCB handle the call?
             JP      NZ,IOHOOK        ;Go to DOS hook if not
             CP      2                ;Set flags for direction
             LD      L,(IX+1)         ;P/ulo-order vector
             LD      H,(IX+2)         ;P/u hi-order vector
             JP      (HL)             ;Go to device driver
    DVRRET   POP     DE               ;Reg restoral routine
             POP     IX
             POP     HL
             POP     BC
             RET
```

The most important lines of code are:

```
        LD      A,(DE)           ;P/u DCB/FCB "type" byte
        AND     B                ;Mask with primitive code
        CP      B                ;Can DCB handle the call?
        JP      NZ,IOHOOK        ;Go to DOS hook if not
```

This piece of code masks the TYPE byte with the primitive mask and checks for a match. For example, if you peek at address X'4015', the TYPE byte for the keyboard, you will see a value of X'01' or 00000001B. If the @GET primitive was entered by the @KBD vector, the mask value would be 00000001B. Thus the comparison would be a match and the jump to IOHOOK would not take place. What if the @PUT primitive was entered with register pair DE pointing to the *KI DCB? The TYPE value of 00000001B would beANDed with the mask value of 00000010B with a resultant value of 0. The zero value would not match the mask value causing a jump to the IOHOOK routine. Obviously, an output request to the keyboard makes no sense! In fact, the keyboard driver could not even handle such a request!! We thus see that the first three TYPE bits (0, 1, and 2) are used by the three primitives to identify whether the device driver has any code to deal with each specific primitive. If you examine the TYPE codes in the *KI, *DO, and *PR device control blocks, you will see that *KI supports @GET, *DO supports @GET, @PUT, and @CTL, while the *PR device supports only @PUT and @CTL. Where a conflict occurs, the IOHOOK routine is supposed to deal with the error. In Level II, this will generate an illegal function call error.

What would happen if the bits 0-2 each contained a zero but one of the bits 3-7 contained a one? If this were the case, then the comparison would never match the primitive mask and the jump to IOHOOK would always happen. Notice that the technical documentation states a ROUTED device has its TYPE byte bit 4 set to a one. A NIL device has its TYPE byte bit 3 set to a one. Therefore, any device which is ROUTED or NIL, will have all three primitive byte I/O calls directed to the IOHOOK routine. Also, if you examine the technical information on file control blocks, you will see that an FCB for an OPEN file, will have FCB+0 bit 7 set. This was not arbitrary. If register pair DE contains an FCB pointer for the primitive call (@GET or @PUT), then the DVRBGN routine will likewise vector to IOHOOK. This little feat enables us to get out of the ROM into a routine that we control in the operating system for any primitive call with a ROUTED or NIL device or an open file. More on that later.

Before we go on to the Model III DVRBGN routine, let's take a peek at one last compare. After the masking is performed, if we don't jump to IOHOOK, we see a "CP 2" instruction. What's the purpose of this? The masked value will be either 1 for @GET, 2 for @PUT, or 4 for @CTL. The following table illustrates the FLAG register conditions prevailing after the compare for each primitive:

TABLE I - FLAG SETUP

1 CP 2	**C,NZ**	**= @GET primitive**
2 CP 2	**Z,NC**	**= @PUT primitive**
4 CP 2	**NZ,NC**	**= @CTL primitive**

After the compare, the DVRBGN routine picks up the vector pointing to the driver routine's entry point, places it into register pair HL, and performs an indirect jump instruction (this causes a jump to the address contained in register pair HL).

We then see that when the DVRBGN routine passes control over to the device driver routine, the flag conditions are unique for each different primitive. This provides a method that the drivers can use to establish what primitive was used to access the routine - and the primitive established the direction of the request - input, output, or control!

Now, the corresponding routine in the Model III will be shown. Prepare for the pain. The horrible instructions are preceded by triple asterisks:

```
;*=*=*
;       Model III Driver Initialization for hooks
;*=*=*
DVRBGN  PUSH    HL              ;Save register HL
        PUSH    IX              ;Save register IX
        PUSH    DE              ;Transfer the DCB or FCB
        POP     IX              ;   address to IX
        PUSH    DE              ;Save register DE
        LD      HL,DVRRET       ;Set up returnrestoral
        PUSH    HL              ;  onto stack
        LD      C,A             ;Xfer char to output
        LD      A,(DE)          ;P/u DCB/FCB "type" byte
***     BIT     7,A             ;If not a file, don't
***     JR      Z,DVRB1         ;  permit the IOHOOK
        AND     B               ;Mask with primitive code
        CP      B               ;Can DCB handle the call?
        JP      NZ,IOHOOK       ;Go to DOS hook if not
DVRBl   AND     B               ;Mask with primitive code
        CP      2               ;Set flags for direction
        LD      L,(IX+1)        ;P/ulo-order vector
        LD      H,(IX+2)        ;P/u hi-order vector
        JP      (HL)            ;Go to device driver
DVRRET  POP     DE              ;Reg restoral routine
        POP     IX
        POP     HL
        POP     BC
        RET
```

Tandy got their sticky little fingers into the ROM and created two lines of code that kept us from exiting through IOHOOK anytime bit 7 was reset. Ugh! Double ugh! Thus, neither the ROUTE bit nor the NIL bit could be used to force control to IOHOOK. This forced us to either implement a route driver in high memory (which would have eliminated device routes from LBASIC under the current scheme of memory management) or a route table in SYSRES (which would limit the number of active routes to some maximum number). The latter method was chosen as the lesser of two evils with four being arbitrarily set as the route table maximum. Four popped out due to space limitations.

Since so much has been stated about the IOHOOK jump, let's take a look at a representative sample of such a routine. This one originates from the Model I LDOS:

```
     ;*=*=*
     ;       LDOS Model I IOHOOK routine
     ;*=*=*
     IOHOOK0 PUSH    HL              ;Xfer pointer to IX
             POP     IX
     IOHOOK  LD      L,(IX+1)        ;P/ulo-order vector
             LD      H,(IX+2)        ;P/u hi-order vector
             LD      A,(IX+0)        ;P/u DCB/FCB type
             CP      00010000B       ;Is this DCB routed?
             JR      Z,IOHOOK$       ;Vector points to new DCB
             JP      NC,BYTEIO       ;Disk I/O on bits 7,6,5
             AND     00001000B       ;Is this DCB NILed?
             XOR     00001000B       ;Set Z-flag if so
             RET     Z               ;Ignore CALL if NIL
             LD      A,B             ;Xfer the mask code
             CP      2               ;Set up the flags
             JP      (HL)            ;  and go to vector addr
     ;*=*=*
     ;       Byte I/O routine
     ;*=*=*
     BYTEIO  EQU     $               ;Tests for direction to
                                     ;  read or write 1 byte
```

Consider what would happen if we performed the command:

```
ROUTE *PR *DO
```

The *PR TYPE byte would contain the value X'10' indicating a routed device. Also, the ROUTE library command would replace the vector address in the *PR DCB with a pointer to the *DO device control block. That is, the first three bytes of the *PR DCB would be [10 1D 40]. If you trace through a call to @PRT (request to output a byte to the printer), you will discover that a jump to IOHOOK is performed. Trace through the IOHOOK routine. We first pick up the vector to *DO's DCB from (IX+1) and (IX+2) and put it into register pair HL. We then pick up the TYPE byte, compare it to 00010000B (X'10') and discover that the device is routed. The match causes a jump to IOHOOK$ which transfers the vector in HL (remember HL contained the *DO DCB address) This then falls through to IOHOOK which grabs the *DO vector and TYPE byte. If you examine the remaining code of IOHOOK, you will discover that in this case, the last three instructions get executed which are essentially identical to the ROM DVRBGN exit discussed earlier. Notice that if *DO was itself routed, IOHOOK would again loop to IOHOOK$ and continue the loop until it got to a device in the "chain" that was not routed.

With the NIL bit set, it is easy to observe that the code in IOHOOK that appears as:

```
AND    8
XOR    8
RET    Z
```

will perform an immediate return with the Z-flag set on a NIL device. This function constitutes the bit-bucket.

If we performed the command:

```
ROUTE *PR to filespec
```

the TYPE byte of the *PR DCB would be set to X'10' indicating a route. However, the vector bytes would be set to the file control block in high memory established by the ROUTE command. The file would have been opened so the FCB would be in an open state (remember, bit 7 is set). What happens if we @PRT a character?

The DVRBGN routine will again vector to IOHOOK. IOHOOK will discover that *PR is routed as previously shown. When IOHOOK$ does its job, it will now point IX to the FCB of the file *PR was routed to. The TYPE byte of the FCB (the FCB+1 byte) will have bit 7 set causing the jump to the BYTEIO routine. This routine is not going to be detailed here. It essentially ascertains from the mask byte whether the request was @PUT or @GET and performs the appropriate byte transfer for the file.

Now that we have examined how LDOS directs I/O to either drivers or file access routines, let's take a peek at how we can establish control blocks for device independence. That's the real easy part. When you are requested to enter a specification, you generally can provide a device specification or a file specification. If a program is written to always provide for a 32-byte FCB, you will always have enough space for an 8-byte DCB. The system @OPEN and @CLOSE routines (@INIT included) detect the specification of device or file via the presence or absence of the asterisk prefix required in device specifications. If a file specification is given, @OPEN opens the file and constructs the open FCB. If a device specification is given, @OPEN copies the DCB into the FCB space (in the Model III we also must establish a pseudo-route table within the FCB space so that IOHOOK can gain control. Thus a standard linkage such as:

```
          LD      HL,BUFFER
          LD      DE,FCB
          LD      B,0
          CALL    @OPEN
```

doesn't care if FCB contained a true file specification or a device specification. At the conclusion of the @OPEN statement, the FCB contents are filled with data needed to support calls for byte I/O such as:

```
          LD      DE,FCB
          CALL    @GET
```

or

```
          LD      DE,FCB           ;Point to control block
          LD      A,CHAR           ;P/u the char to output
          CALL    @PUT             ;Issue the PUT call
          CALL    NZ,TSTERR        ;Test if valid error
          JP      NZ,IOERR         ;Go service the error
TSTERR    EX      DE,HL            ;If a file, its an error
          BIT     7,(HL)           ;Test for FCB
          EX      DE,HL            ;Z=DCB, NZ=FCB
          RET
```

The byte I/O vector is independent of the device type provided the device can support that direction of I/O. The error testing procedure is needed if any output requests are made in an environment that is subject to either device or file specifications (independence). Since the TRS-80 ROM driver routines do not complete their output function and return unilaterally with the ZERO FLAG get, a @PUT to a device could be interpreted as an error. However, it is most important to test for an error return if the @PUT was to a disk file! It is for this reason that program authors who do not take this into consideration when writing their application software will cause havoc to those LDOS users that ROUTE *PR TO FILESPEC and lockup the system when "printer" output is re-directed to a disk file on a diskette that becomes FULL.

Want to have some fun? Try going into LBASIC and running the following mini-program:

```
 5 FO$="*PR"
10 OPEN"O",1,FO$
20 PRINT#1,"This is a test"
30 CLOSE1
```

Find a use for that? Okay, let's say you have a program that writes data to a sequential file. You want to test the program and ascertain what is being written without actually having to generate the file and list it back out. You could of course duplicate your PRINT#1 statements as LPRINT statements. You could0also have substituted the *PR device name in lieu of the file name in the OPEN statement. If your program contained numerous PRINT#1 statements, which would be easier? How about this one?

```
 ROUTE *D0 NIL
 ROUTE *D1 FILE1
 ROUTE *D2 FILE2
 ROUTE *D3 FILE3
 LBASIC
 10 CMD"ROUTE *D0 *D1
 20 GOSUB180
 30 PRINT#1,"Bunch of data"
 40 CLOSE1
 50 CMD"ROUTE *D0 *D2
 60 GOSUB180
 70 PRINT#2,"Another bunch of data"
 80 CLOSE1
 90 CMD"ROUTE *D0 *D3
110 GOSUB180
110 PRINT#1,"The last bunch of data"
120 CLOSE1
130 CMD"S
140 CMD"RESET *D1
150 CMD"RESET *D2
160 CMD"RESET *D3
170 END
180 OPEN"0",1,"*D0":RETURN
```

And they said I didn't know BASIC! Ingenuity will find a use for this procedure. One such use could be to segregate output for different kinds of preprinted forms - such as tax forms.

Now we will move on to the device driver linkage used to separate out the @PUT, @GET, and @CTL calls. It is extremely important to remember the FLAG register direction conditions that were set according to the primitive byte I/O routine that got us to the driver. These conditions were presented in TABLE I. Consider the following protocol for the "front end" of a driver:

```
ENTRY   JR      BEGIN     ;Branch around linkage
        DW      $-$       ;Last byte used by driver
        DB      BEGIN-ENTRY-5,'MODNAME'
BEGIN   JR      C,WASGET  ;Go if @GET request
        JR      Z,WASPUT  ;Go if @PUT request
        .                 ;Must have been @CTL request
```

At the entry of the driver, an absolute jump instruction executes which causes a branch around some data. Ignore, for a moment, the data area which will be discussed shortly. At the label, BEGIN, a test is made on the CARRY FLAG. If the CARRY was set, then it only could have gotten itself into that condition if the disk primitive was an input request (@GET). Thus, an input request could be directed to a part of the driver which only handles INPUT from the device.

If the request was not from the @GET primitive, the CARRY will not be set. The next test is if the ZERO FLAG is set. The ZERO condition prevailed when a @PUT primitive was the initial request. Thus the jump to WASPUT can go to a part of the driver that deals specifically with OUTPUT to the device.

If neither the ZERO or CARRY flags are set, the routine falls through to the next instruction (which is not shown). What would follow would be a part of the driver that would handle @CTL calls. For instance, you may want to have an RS-232 driver handle a BREAK by issuing a @CTL call 80 that the RS-232 driver emits a true modem break, but a CONTROL-A (SHIFT+DOWNARROW+A) would @PUT a X'01'. You might ask why we don't check for a "NOCARRY" flag? Well if we just ascertained that the CARRY is not set, then it must be the other condition - there is only one CARRY flag - it is either SET or RESET. Thus the only other case could be a @CTL request.

Some drivers are written to assume that @CTL requests are to be handled exactly like @PUT requests. This is entirely up to the function of the driver and the author thereof. If the driver is the LDOS byte I/O disk driver, a @CTL call is output to the disk file just as if the byte I/O primitive was a @PUT.

Now, the front end linkage shown above has been implemented in LDOS supplied drivers and filters starting with release 5.1.1. This was to provide a uniform way of identifying the name of a driver when it was resident in memory as well as the last memory address used by the driver. The PDUBL/CMD double density disk driver released with 5.1.1 also has this linkage. PDUBL itself uses it to search through all drivers vectored from the Drive Code Table (DCT) prior to relocating itself into high memory in case PDUBL was already resident from a previous PDUBL command.

Under LDOS 5.0.x, if you entered PDUBL twice, two copies of PDUBL would be in high memory - one just wasting away. This will not happen under 5.1.1. TWOSIDE also uses this technique. Although LDOS 5.1.0 had all of its filters and drivers implemented with a technique for determining system residency, they have been revised to also use this front end linkage for the sake of uniformity at the expense of a few extra bytes taken up in RAM. It will be interesting to observe that if every module using space in high memory used this technique, then one could check a device chain and locate the starting and ending point of each filter, driver, and whatnot. In fact, a memory management routine could be written to start from (HIGH$+1) and search through all of high memory presenting a directory of space utilization. Have I raised a few eyebrows? If you write drivers and filters, you can use this technique as easily as I can within the system.

One last topic needs to be discussed relating to filters. A filter is inserted between the DCB and driver routine (or between the DCB and the current filter when applied to a DCB already filtered). The usual linkage for a filter is to access the chained module by calling the address that was in the DCB at the time of the filter installation. However, since the driver expected the FLAG register contents to designate the I/O direction of the request, it is ABSOLUTELY ESSENTIAL that each filter maintain the integrity of those flags when issuing that CALL instruction. If you check out the TRAP filter described in the technical section of your LDOS manual, that is the purpose of the PUSH AF - POP AF sequence. Another pitfall to watch out for is that routines handling output requests expect the output character stored in the C register when the routine takes over. At the conclusion of the routine, the character is generally placed in the A register. If your filter is going to massage output, adhere to those specifications. This will ensure compatibility amongst all filters and drivers.

Questions and/or comments concerning this article may be addressed to me at LSI corporate headquarters.

RELOCATING CODE FOR LBASIC USR ROUTINES by Chuck

One of the most useful features of LDOS is the capability for a user to insert blocks of code in high memory and have them protected from the system. The FILTER and SET library commands operate in this manner. It is also possible to put blocks of code into memory to be called as USR routines from LBASIC. However, since the high memory location of the code can vary, any absolute references in the code to be moved will be incorrect after the relocation. Also, if the code is to be called as a USR routine, a method must be used to store the entry point for LBASIC to recover and assign with a DEFUSR statement. This article will describe three steps - how to relocate the code and set the new high memory protect, how to relocate absolute addresses in the code, and how to access this code from LBASIC.

Normally, a front end program module is used to relocate the user code into high memory. This module normally can beORGed at X'5200'. However, if the program is to be executed with the RUN (X) command, it should beORGed at X'5300'. The normal order of code in the program will be as follows:

1) * The memory protect routine.
2) * The address relocater.
3) * The block move relocation.
4) * The table of relocation addresses.
5) * The actual code to be moved.

The first thing you should do is to find the current HIGH$ value and the length of the code you wish to move. You can then calculate the new HIGH$ value, and the location in high memory where your code will reside. The next example assumes the label LENGTH is equated to the length of your code (a method to do this is shown in a later example).

* The memory protect routine

```
00200        LD       HL,(HIGH$)   ;get current memory protect
00210        LD       BC,LENGTH    ;length ofrelocatable code
00220        OR       A            ;clear Carry flag
00230        SBC      HL,BC        ,calculate new HIGHS
00240        LD       (HIGH$),HL   ;store this value
00250        INC      HL           ;pt to 1st memory location
```

Now that HIGH$ has the correct value to protect the code to be moved and the HL register is pointed at the start of the relocation area, you can see about fixing up all absolute addresses in the code. The method described here was chosen because it fulfils two requirements - it is very easy to use and modify, and it adds no extra code in high memory. This method requires the construction of a table containing the absolute instruction addresses. The easiest way to do this is to label all absolute instructions with common labels, such as REL1, REL2, etc.

For example, suppose you had the following routine assembled at X'5500':

```
5500 C30655  00000      JP       Z,OUT
5503 3E0A    00000      LD       A,10
5505 C9      00000      RET
5506 E5      00000 OUT  PUSH     HL
```

The first line of code assembles as an absolute jump on zero to address X'5506'. When this block of code is moved to high memory, a jump to X'5506' is not what the user really wants. This same problem occurs with CALL statements, a LD of registers from memory locations, or a LD of a memory location from a register. Of course, jumps, calls, or loads outside of the relocatable code will not be a problem.

There are many different ways to get around this problem when writing your own code. Relative jumps can be used instead of absolute jumps, alternate registers can be used instead of memory locations, etc. However, there will be times when you will have to use absolute instructions. The following block of code will be used as an example to show how to label the instructions and construct the table. This code serves no actual purpose other than as an example.

* The actual code to be moved

```
5500          02000         ORG     5500H
5500 CD7F0A   02010 START   CALL    0A7FH
5503 ED5B1755 02020 REL1    LD      DE,(STOR1)
5507 CDC901   02030         CALL    01C9H
550A CD1355   02040 REL2    CALL    LOOP1
550D CA1455   02050 REL3    JP      Z,ENDIT
5510 221855   02060 REL4    LD      (STOR2),HL
5513 C9       02070 LOOP1   RET
5514 C3C901   02080 ENDIT   JP      01C9H
5517 00       02090 STOR1   DEFB    00
5518 00       02100 STOR2   DEFB    00
0018          02110 LENGTH  EQU     $-START
5500          02110         END     5500H
```

As you can see, only those instructions that reference addresses inside the user's block of code need to be relocated. Calls or Jumps to ROM or LDOS system addresses outside of the relocatable code should not be altered!

The relocating table will be a series of addresses. The label RELTBL should be placed on the first entry in the table. Also, a terminating pair of 0 bytes should be the last entry in the table.

```
* The table of relocation addresses

01000 RELTBL  DEFW    REL1+2
              DEFW    REL2+1
              DEFW    REL3+1
              DEFW    REL4+1
              DEFW    0             ;end of table indicator
```

The table entries point to the locations in memory where the absolute addresses reside. Notice that the REL1 label entry is +2 from the start of the instruction. This is because the LD DE,(STOR1) instruction is a 4 byte instruction with the address being the last two bytes. All other of the example addresses follow single byte instructions and therefore are offset by only 1 byte. If you are unsure of the instruction length, it would be advisable to run an assembled listing and check the offsets used in the table.

The next block of code will make the adjustments of the necessary addresses before the code is moved into high memory. It should immediately follow the block of code that calculates the new HIGH$. At that point, register pair HL contains the address where the relocatable code will start in high memory, and BC contains the byte count of the code to relocate. The address in HL will also be the entry point when the code is called as a USR routine from LBASIC. It is assumed that the label START is used on the first line of the code to actually be moved into high memory.

```
* The address relocater
```

```
00500 ;****   Relocate hard Jumps, Calls, and Addresses
00510 ;
00510         PUSH    HL              ;save entry point
00520         PUSH    BC              ;save byte count
00530         LD      BC,START        ;reloc. code start
00550         OR      A               ;clear CARRY flag
00560         SBC     HL,BC           ;calc. offset for reloc.
00561 ;
00562 ;     Register HL now contains the offset difference
00563 ;     between the current location and the location
00564 ;     in high memory.
00565 ;
00570         LD      C,L
00580         LD      B,H             ;offset into BC
00590         LD      IX,RELTBL       ;table of labels to adj.
00600 RELOOP  LD      L,(IX)          ;pre-move address
00610         LD      HL,(IX+1)
00620         LD      A,H             ;msb of table entry
00630         OR      A               ;see if it is a Zero
00640         JR      Z,DUNREL        ;end ofreloc. on Zero
00650         PUSH    HL              ;save loc. of address
00660         LD      E,(HL)          ;now get the CONTENTS
00670         INC     HL              ;of the memory location
00680         LD      D,(HL)
00690         EX      DE,HL           ;current absolute address
00700         ADD     HL,BC           ;add in the offset
00710         EX      DE,HL           ;new address into DE
00720         POP     HL              ;loc of address
00730         LD      (HL),E          ;store the offset address
00740         INC     HL              ;in the current instruction
00750         LD      (HL),D
00760         INC     IX              ;next table location
00770         INC     IX
00780         JR      RELOOP          ;reloc. next label
```

Now that all of the necessary adjustments are made, you can move the code into high memory. Remember that the entry point and byte count were pushed onto the stack at the start of the previous routine.

* The block move relocation

```
00800 DUNREL  POP     BC              ;recover byte count
00810         POP     DE              ;recover entry point
00820         LD      HL,START        ;start of user code
00830         LDIR                    ;move it
00840         JP      @EXIT           ;EXIT, all done
```

At this point everything is complete. The routine is loaded in high memory, the HIGH$ value is set to protect it, and all absolute addresses have be relocated. However, nothing has been done to make this routine available as a USR call from LBASIC. To do this, the location of the entry point in high memory must be stored in a fixed location so that the LBASIC program can pick it up.

There is an 8 byte user storage area provide in LDOS just for this purpose. This area is pointed to by the address in USTOR$, memory locations X'4DFE' and X'4DFF'. Consider the rewrite of the previous five lines:

```
   00800 DUNREL  POP     BC              ;recover byte count
   00810         LD      HL,USTOR$       ;get pointer to user area
   00820         LD      E,(HL)          ;now point DE at
   00830         INC     HL
   00840         LD      D,(HL)          ;user storage location
   00850         EX      DE,HL           ;0nto HL for load
   00860         POP     DE              ;recover entry point
   00870         LD      (HL),E          ;and store entry point
   00880         INC     HL
   00890         LD      (HL),D          ;for recovery byLBASlC
   00900 ;
   00910 ;           Now we continue the relocation move
   00920 ;
   00930         LD      HL,START        ;start of user code
   00940         LDIR                    ;move it into high memory
   00950         JP      @EXIT           ;EXIT, all done
```

This routine will work for USR programs with a single entry point. If more than one USR call will be made to the same block of code, the entry points can be stored sequentially in the user storage area. Since there are 8 bytes of storage space, up to four entry points can be used with this method. However, storing four entry points would require a slightly different routine to replace lines 860 to 890 in the previous example.

The LBASIC program can pick up the entry point as follows:

```
   100 DEFINT X
   110 X=PEEK(&H4DFE)+256*(PEEK(&H4DFF))
   120 REM ** It will be necessary to use the negative value
   130 REM ** when the entry point is above 32767.
   140 DEFUSR1= -1*(65536-(peek(X)+256*(peek(X+1))))
```

If more than one entry point is needed, addition lines similar to line 140 can be used, indexing off of the value in the variable X.

THE JCL CORNER by Chuck

A survey of all the customer support people revealed an interesting fact. Only one of them recalled ever being asked a question dealing with JCL compiling or the compilation macros. "Plenty of questions about the execution phase", they said, "but nothing about compiling". This tends to suggest one of two things - either everyone knows everything about the compiling phase of JCL, or no one is using it because they don't understand it. Assuming a point midway between these two conclusions, the next issues of the quarterly will try and explain the function of the compilation phase of JCL, and give some practical examples of its use.

The first question that comes to mind is "Why compile a JCL file ?" Basically, the compilation phase allows a single JCL file to be used for many different functions, and to have these functions selected at runtime. There are three terms that will be used throughout this discussion. They are LABEL, TOKEN, and MACRO. Labels are probably the easiest to understand. They are merely used to combine many smaller JCL files into one larger file to save disk and disk directory space. A token can be considered a variable. It will have either a logical true or logical false value. It may also have a string value assigned to it. A macro is a special JCL command statement. It is always in the format of two slashes (//) followed by the appropriate word.

So how can you construct a JCL file to make practical use of the compilation feature? A good place to start will be by gaining an understanding of the logical decision capabilities of JCL. This will include the three logical operator symbols and the three decision macros. Of course, the all important tokens will also be discussed.

```
   Logical operators      Decision macros
   -----------------      ---------------
      AND  &                  //IF
      OR   +                  //ELSE
      NOT  -                  //END
```

The decision macros can be used with or without the logical operators. The //IF macro is used to test the logical value of the statement following it. For example:

```
   //IF a          ;Tests to see if the token "a" is true
   //IF -a         ;Tests to see if the token "a" is false
```

A token can be declared to be true by specifying it on the DO command line. It does not have to be assigned a value. All of the following examples will assign a true value to the token "a".

```
   DO TEST/JCL (a)
   DO TEST/JCL (a=testfile)
   DO TEST/JCL (A)
```

You will notice that upper and lower case are evaluated identically, and there is no real difference between the first and third examples. Now that we know one method of assigning a true value to a token, we can determine how to assign it a false value.

There is literally nothing to assigning a false value to a token - simply do not specify it on the DO command line. Also, the //SET, //ASSIGN, and //RESET macros can assign or reset a token's logical value. They will be discussed later.

The //END macro is used to mark the end of a corresponding //IF block. There must be one //END used for every //IF. The //ELSE macro is used to provide an alternative course of action in case an //IF statement evaluates false. As you will see later, //IF conditional blocks can also be nested inside of //ELSE blocks. Now for some examples:

```
   Example #1    . TEST/JCL
```

```
                //if a
                . print this comment
                //end

   Example #2   . TEST/JCL
                //if a
                . print this comment
                //else
                . print nothing entered!
                //end

   Example #3   . TEST/JCL
                //if a
                . print this comment
                //else
                //if b
                . print it was b, not a
                //else
                . print neither a or b
                //end
                //end
```

Example #1 is easy to understand, If the token "a" was true, the comment line would be printed. Example #2 gives an alternate course of action in case the token "a" is false. Example #3 does the same thing, except that the alternate action also is a conditional test. Notice that there are two end statements in this example - the first //END ends the second //IF, and the second //END ends the first //IF.

The logical operators can be used with the //IF macro to do more complex testing. Logical operators all have the same precedence, and will be evaluated from right to left. Refer to the following:

```
   //IF a+b            ;if either a OR b is true
   //IF -a&-b&-c       ;if NOT a AND NOT b AND NOT c (if none is true)
   //IF a&-b           ;if a AND NOT b (if a is true and b is false)
   //IF b+-a           ;if b OR NOT a (if either b is true or a is not true)
```

As you can see, almost any combination of tokens and logical operators can be used with an //IF macro to determine the logical condition, and therefore the inclusion or exclusion of the following lines in the file up to the corresponding //END macro.

As was mentioned earlier, the //SET and //RESET macros can change the logical value of a token. //SET will assign a true value, and //RESET will assign a false value. //ASSIGN will assign a true value, as well as a character string value, to a token. One very powerful use for these macros is to allow a single control token to adjust the values of numerous other tokens for use by the rest of the JCL. This will reduce the number of characters needed on the DO command line to. For example:

```
                .TEST/JCL
                //if a
                //set b
                //set c
```

```
                 //assign d=testfile
                 //reset ex
                 //else
                 . a was not entered
                 //exit
                 //end
```

By entering a DO command "DO TEST/JCL (a)", you will assign the tokens "b" and "c" a logical true value, assign the token "d" a true value as well as the string value "testfile", and assign the token "ex" a logical false value, even if "ex" was entered on the DO command line. You could now have a series of //IF lines to do procedures based on the value of the tokens.

Perhaps you're wondering why a token can be assigned a string value since we have only discussed logical evaluations. They answer is that token values can be used as substitution fields, and can also be concatenated to build larger strings - probably the most powerful feature of the entire JCL language! Token substitution and concatenation is easy. Simply use the pound sign (#) to enclose the token name in the proper places. For example:

```
   COPY #f1#:1 to #f2#:2
   COPY #f1#:#s# to #f2#:#d#
   BACKUP GEN/#ex#:#s# :#d#
```

The first two examples have substitution fields for the filespec in the COPY command. Additionally, the source and destination drivespecs are substituted in the second example. The third example substitutes the file extension for a BACKUP command, as well as the drivespecs. Consider the following JCL examples:

```
   Example #1    . TEST/JCL
                 //if a
                 //assign ex=ASM
                 //end
                 //if -a
                 //assign ex=CMD
                 //end
                 //if -s
                 //assign s=1
                 //end
                 //if -d
                 //assign d=2
                 //end
                 BACKUP GEN/#ex#:#s# :#d#
                 //exit
```

Suppose you had two sets of files named GEN1 through GEN8, with one set having the extension /ASM and the other set having the extension /CMD. This file would allow you to backup these files sets and assign the source and destination drives. It also provides defaultdrivespecs in case they are not entered on the DO command line. The JCL files in the Update section of the newsletter even use a substitution field for the password in thefilespec to apply the patches. Refer to the following DO command line examples and the resultant line generated by the JCL compiler:

```
   DO TEST (a,d=3)     =>BACKUP GEN/ASM:1 :3
   DO TEST             =>BACKUP GEN/CMD:1 :2
   DO TEST (s=0,d=4)   =>BACKUP GEN/CMD:0 :4
```

One final point to end this month's JCL corner. IF you are going to experiment with JCL compilation, you may wish to use the special character $ in the DO command. This will do the compilation without actually executing the file. You can list the SYSTEM/JCL file to see the results of the compilation. If everything is satisfactory, you can then execute the SYSTEM/JCL file with the command DO *. Also, any compilation process should produce at least one line of executable code. Otherwise, the existing SYSTEM/JCL file may be executed. One way to avoid this problem is to start every JCL file with an execution comment listing the name of the file.

LATE BREAKING NOTES AND PATCHES

There are a few notes and extra patches that apply to the Model I, 5.1.1 version. If the file dates on your master is 12/15/81 or later, these patches have already been installed.

On page 5-5 of the 5.1.1 LScript addendum, there is an error in the MICROPROOF patch. The byte 22F in the second line of code should be a 2F.

There is a small patch for LBASIC/0V2 that correctly exits to LBASIC if the <BREAK> key is pressed.

```
   . LBASIC/0V2A
   . This patch will cause the break key to work correctly
   . during a CMU"X"
   .
   X'5682'=62 58
   X'5694'=6B 21 00
   X'5862'=CD 73 56 C3 F6 55
   .
   . EOP
```

Here is a short patch for the KI/DVR program, Model I, 5.1.1 only. It allows a zero character to be generated by a <CTRL><@> as stated in the manual.

```
   . KIA/FX1
   . This patch will allow CNT<@> to send a null properly
   .
   D00,FD=FE FF
   .
   . EOP
```

Both the Model I and III versions allow system overlays to be resided in memory, thereby freeing up disk space. When creating a minimum system disk, the only SYS files that normally need to be on the disk to boot the system are SYS0 and SYS2. However, if the CONFIG/SYS file has more than 4 extents, SYS8 must also be on the disk.

It is not advisable to ROUTE or LINK the printer to a disk file when using LCOMM or the printer SPOOLer. The results will vary depending on the particular hardware you are using. The most common result is a constant stream of zeros written to the file. As explained in Roy's article on Device I/O, the @CTL call can be used to check the status of a device. When the @CTL call is made to a device that is connected to a file, a zero character actually gets written to a file. In LCOMM, the status of the printer is checked as a high priority interrupt task. That means approximately 40 zeros per second will be written to the file!

Here is a short patch to KI/DVR for Model I, 5.1.1 owners who are using the LSCRIPT patch with Scripsit. It will increase the repeat rate of the keyboard, and thereby speed up the cursor movement in Scripsit.

```
. PATCH FOR KI/DVR to increase the keyboard repeat rate
.
D01,22=01
D02,F9=01
D03,00=01
```

One final note on double sided drives.... We have just re-tested TWOSIDE and PDUBL with double sided drives. We could find no problems in double or single density. However, one of our dealers who is also a hardware dealer has discovered that some Radio Shack interfaces had pins 32 and 34 connected together. Also, some disk drives have pin 34 grounded internally. When these two hardware elements come together in the same system, trouble results, especially with double sided drives.lf you are having difficulty, check the expansion interface and separate pins 32 and 34 if necessary.

The patch file VC/FIX on the 5.1.1 LDOSXTRA disk had extra carriage returns in the three longest lines in the file. These must be removed to keep from getting an error when applying the patch. NOTE: This patch is identical to the patch on the MicroNET board, and the one in the last newsletter.

When doing a backup from a source disk of larger capacity than the destination disk, it will be necessary to have multiple destination disks. BE SURE TREY ARE ALL FORMATTED THE SAME. Switching the number of tracks or sides can cause the backup to abort with a "Directory Read Error".

For those of you looking for an LDOS compatible data base, the MAXI MANAGER data base program should now be LDOS compatible. Instructions on converting the programs to LDOS will be supplied when purchased.