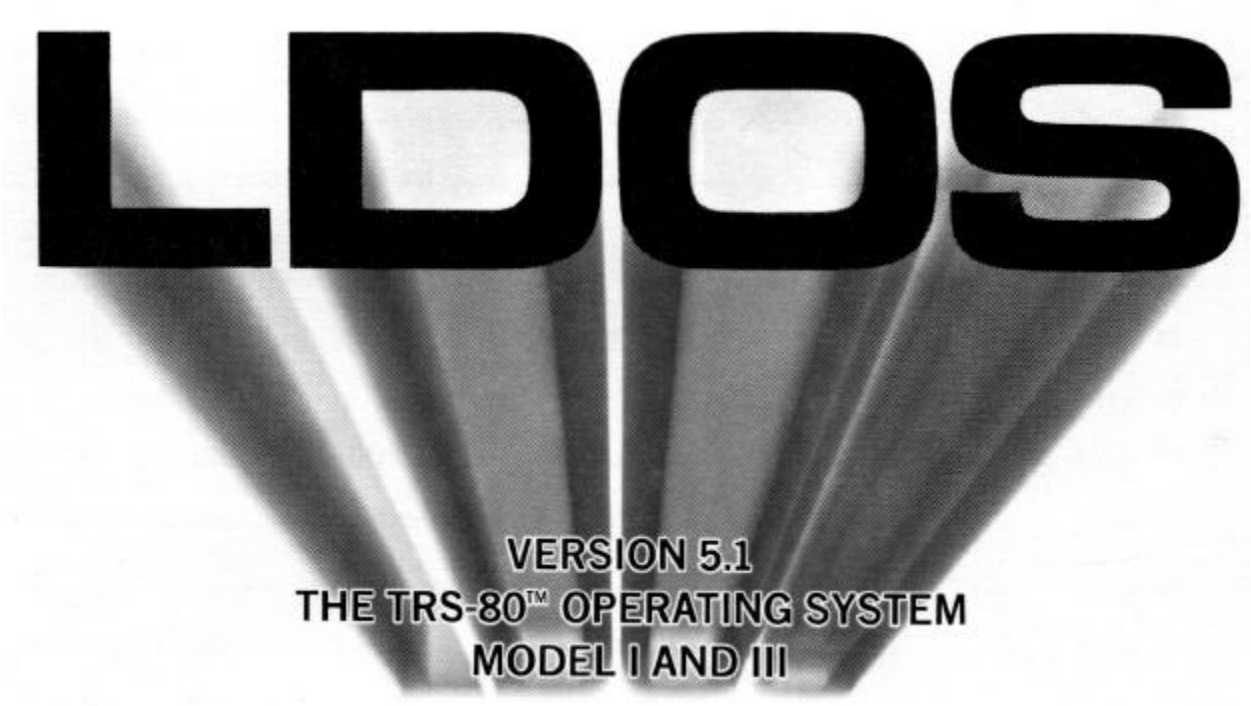


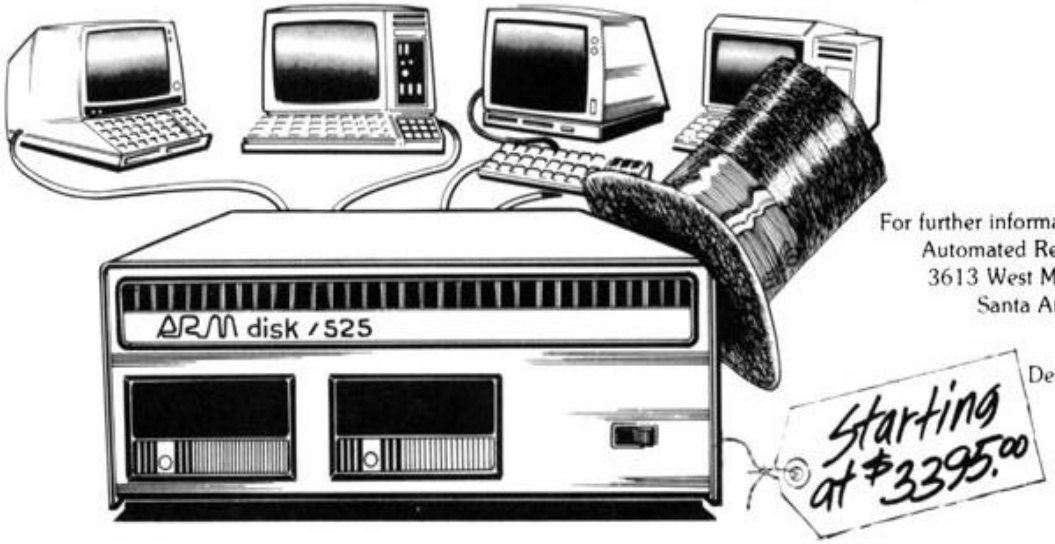
LOGICAL SYSTEMS INC.

Is your floppy a flop?

Book the new ARMdisk/525 Winchester subsystem with your personal or small business computer and enjoy a performance no floppy can equal. Hard disk capacity of up to 30 megabytes keeps you packin' them in where most other systems are playing to a full house. And eight-count'em-heads team up to cut data accessing time by as much as 80%. Backstage, the ARMdisk/525 uses an intelligent controller to help ensure data integrity and an error correction code that's stored on disk to make data loss practically impossible.

On the supporting bill is the LDOS operating system to provide for media backup as well as ***intelligent*** file save and restore. With a MTBF of 8,000 hours and a multiplex feature that allows up to four hosts to share the same unit, the ARMdisk/525 has the kind of star quality that legends are made of.

You'll applaud the price/performance of the Winchester disk from ARM.



For further information, please contact:
Automated Resource Management
3613 West Mac Arthur Boulevard
Santa Ana, California 92704
(714)850-9792

Dealer inquiries invited.

Table of Contents

VIEW FROM THE BOTTOM FLOOR ..... Page 2

FREE SOFTWARE FROM LSI! ..... Page 5

5.1.3 RELEASE ..... Page 5

WHAT'S NEW ? - LDOS COMPATIBLE SOFTWARE ..... Page 6

NEW LSI PRODUCTS ..... Page 8

ARTICLES AND REVIEWS FROM OUR USERS:

REVIEW - Earle Robinson reviews MICROCOMPUTER MATH ..... Page 10

PARITY = ODD by Tim Daneliuk ..... Page 11

..er.. - Earle Robinson on assembly language programming .... Page 14

LDOS - IT'S GREEK TO ME by Charles Knight ..... Page 15

HINTS AND TIPS contributed by our users ..... Page 18

FROM THE LSI STAFF:

ITEMS OF GENERAL INTEREST - Includes the "We Surrender" Patch to disable /BAS in LBASIC ..... Page 18

RTC - Roy describes the LDOS Task Processor ..... Page 20

THE JCL CORNER by Chuck ..... Page 34

LES INFORMATION by Les Mikesell ..... Page 40

USER CONTRIBUTED PROGRAMS:

RENAME - ADD THOSE DEFAULT EXTENSIONS ..... Page 41

VC3/FIX - A patch for Model III Visicalc ..... Page 43

FIX512 - KSM and KI/DVR patches for Version 5.1.2 ..... Page 44

LATE BREAKING NEWS AND OTHER RANDOM ITEMS ..... Page 45

11520 N. Port Washington Rd. Mequon WI 53092
(414) 241-3066

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

Well, LSI has now taken over all the available space in our office building and is still looking for more. Thanks to our dedicated users and their promotion of our products, LSI has grown beyond our wildest dreams. As we have grown, things have changed (sometimes too often) and we have tried to keep our customers in mind as these changes have occurred. One change of late is a major addition to our staff, Les Mikesell. Les will be in our systems software group working on maintenance of our LDOS product line as well as on new projects. We are sure Les will be a valuable addition to our staff.

Speaking of changes, we are considering relocating our company and are looking for suggestions. We are looking to move to a friendly climate, with minimal tax burdens. If you or someone you know is involved in your local Chamber of Commerce or City Planning organization, please get us what information you can. You may be living in the perfect place. If so, let us know where that is.... we are looking for that "perfect" place.

In the last Newsletter I made reference to our new hot-line service. This is a pilot program and we will have to see how it works out. There are no promises that this will be a permanent service. The phone number is (414) 241-4100 and will be available 24 hours a day starting the last week of July. This Hot-Line will have 1 to 3 minutes of LDOS announcements and info available to anyone. The Information will change as required, probably every week or so.

All registered LDOS owners have received this newsletter, but for owners whose support has expired or are new owners not on the Extended Service Agreement (ESA) this will be the last newsletter that they receive. If you wish to continue getting the publication and get on or remain on MicroNET, and have forgotten or misplaced your ESA, there is one in the back of this issue for your use.

Radio Shack has finally released their double density modification for the Model I. It seems to work very well and we have released the driver as RDUBL, which is now included on the Model I 5.1.3 LDOS. That's right - a 5.1.3 version of LDDS is now available. This 5.1.3 update is available to 5.1 owners (Mod I and III) for just $10.00 or $5.00 if they are on the Extended Service Agreement plan. The 5.1.3 release of LDOS was required to provide additional compatibility to upcoming Model III type software and to add several small enhancements and corrections to LDOS. A new vector was added called RAMDIR, which is in TRSDOS Model III. This is not a patchable update as the entire system was reassembled, so you will have to send in for an update. Complete information about this update will be included when we return your updated disk.

Lobo Drives in Goleta, California has announced a new product called the MAX-80. This is a 64K Z-80 computer that is Model I and Model III type compatible as well as capable of being run in a "full ram" mode. This machine has a programable character generator, a full blown keyboard, controller for 5 and 8 inch drives, a hard disk host adaptor, and 64 x 16 and 80 x 24 video. Even dual serial ports are standard, along with many other exciting features. A special LDOS system will be provided with this machine. The big news about this computer is the price..... under $1,000 retail !!! This could easily be

the best computer buy available at this time..... If you or a friend are considering a new Z-80 machine, check this one out. Deliveries are expected to begin in early fall (maybe late summer). I'm sure there will be quite a backlog for this machine, so get your order in early to get on the list. Call or write LOBO at 354 5. Fairview, Goleta, Ca. 93117 - Toll free - 800/235-1245 (Calf) 800/322-6103.

Some of our customers may know of Kim Watt, but for those who do not, he is the author of Super Utility. Kim and the company he works for, POWERSOFT in Dallas Texas, are two of the biggest supporters of the LDOS system and LSI. For this, many thanks. Powersoft offers a line of support products designed to run on LDOS only. These useful utilities include disk repair and reconstruction programs as well as programming tools and utilities. There are a total of six packages offered, each containing a group of associated utilities and priced at just $29.00. Most of the programs are self documenting and simple to use; many even have built-in HELP type features. So if you need that little "special something", give Powersoft a call. They probably have what you need, or you may just want to get their catalog and browse. I personally use several of these high level utilities on a regular basis when maintaining and testing LDDS. So give one or two a try. You will certainly get your money's worth. They can be reached at Powersoft, 11500 Stemmons X-way, Dallas, Tx. 75229 - (214) 484-5783.

In the last newsletter there was an offer to any registered LDOS owner to purchase a smal-LDOS. This was a special introductory offer and this product is no longer available for retail sale. It is available to hardware manufacturers and software publishers only. However, the smal-LDOS manual is available for $20 to any registered LDOS user. This manual is in the form of a paperback book (about 160 pages) and serves as an excellent "reference guide" for the day to day LDOS user.

One of our new products for this quarter, The Basic Answer (TBA), is one of the biggest advancements in BASIC programming since Microsoft. This is a product that will allow a Basic programmer to write basic code WITHOUT LINE NUMBERS and with up to 14 CHARACTER VARIABLES!! This product should revolutionize the way that you write your basic programs. Now the source code will make sense.

How TBA works is very simple. You write basic code in the same manner that you do now except that all routines have names instead of line numbers and variables can be up to 14 characters in length. You can even have variables declared as local or global in nature. You can write your code with or without the leading line numbers for each line, so you can use a word processor or the TRS-80 basic itself to write programs. Use as many tabs, spaces, comments or linefeeds as needed to make the source self-explanitory. When you are done you "compile" or process your source code through TBA with listing, cross referrenced variables and conditional compilation supported. The output of TBA is a compressed, optimised and ready to run BASIC program that is Microsoft compatible. Your source code is easy to follow and self documenting but the runtime code is very difficult to understand, so it makes it very hard for unauthorized persons to "play" with your code. Moving procedures or subroutines around in your source or adding new blocks of code is never a problem as line numbers are meaningless in the source. If you program in basic, give this product a try; you'll never go back to the old way. There is more information on TBA elsewhere in this newsletter.

The official editor assembler that is endorsed by LSI is EDAS from MISOSYS. This assembler is a great product as it is..... but it's getting better. Within the next couple months there will be a new, even more powerful version of EDAS. This new version will have nested conditionals, nested *GETS and a partial implementation of macros. Many other enhancements are being added including the ability to automatically deal with line numbered or unumbered source files. This package with all new documentation is expected to be priced at $100. Contact MISOSYS at 5904 Edgehill Dr Alexandria, Va. 22303 - (703) 960-2998 for additional information.

We have several active LDOS users submiting articles to this newsletter, but we need many more. Why not try your hand at authoring and article or two. If we accept a submission for publication you may have your choice of $25 per printed page or, depending on the size of the article, any LSI product as payment. Send any submissions in hardcopy and on disk in Scriptsit compatible format directly to LSI, attn: Quarterly Editor. If any program listing is involved it must be included on the disk. Any subject matter is acceptable as long as it has some connection to the LSI product line.

LSI is proud to announce the aquisition of Galactic Software. Due to the heavy interaction between LSI and Galactic in the past year it became more and more evident that LSI should aquire Galactic rather than contract their services. Effective on May 1, 1982, LSI became the owner of Galactic Software Ltd. This aquistition is being treated as a merger with some of the Galactic's resources being directed towards application software. We hope to bring sophisticated applications out under the LSI name by the end of this year. The next newsletter will contain announcements regarding our first major application products.

One thing about our support of controllers, printers, drives, speed-up kits, memory mods, and the like. Please, if you are thinking of getting into any non-Radio Shack items that we have not stated official support for, give us a call. We may not be able to give you a positive yes or no that something will work, but we will try and will pass along what other users may have told us about the product involved. We want to help and we try to have very few (if any) secrets at LSI. Also if you do get an item that we do not "officially" support (usually because we don't have one) and you find it works fine with LDOS or you have found a way to make it work, LET US KNOW. This information can be of great value other LDOS users.

Thank you all again for your active support of our products and our company. I sincerely hope that you and your family have a safe and pleasant summer....

****
*
*
* **ATTENTION!! FREE SOFTWARE FROM LSI!!**
*
*
****

Want a FED or a LED? How about a Filter package or an EDAS? You name it, it's yours, as long as . . .

Well, almost free. The LDOS Quarterly is always on the lookout for well written articles about almost anything having to with LDOS and any other LSI products. To get your free software, all you have to do is submit an article and have it accepted for publication. Upon acceptance, you will be notified and will be allowed to select from a group of software packages. You won't even have to wait for the article to be published - we'll send the software as soon as you make your choice. If you already have all the software you want, we'll give you a choice of $25 per page instead.

When submitting articles, you should send double spaced typewritten or lineprinted copy. An ASCII text file, a Scripsit file, or a SuperScripsit (when available) file MUST accompany the printed copy! As an alternative, you can leave the file on MNET, and we will pick it up there. So all of you authors (and that's almost anyone out there), get those articles coming! Send all correspondence to:

LDOS Quarterly Editor
11520 N. Port Washington Rd.
Mequon, WI 53092

5.1.3 UPDATE FOR MODEL I AND III

Several new programs and a new doubler for the Model I have made it necessary to release LDOS Version 5.1.3. While we were at it, we added and/or changed parameter defaults in a couple of areas. A new system vector, @RAMDIR, was also added. For those of you who sent in for updates near the end of June, your disks were returned with the 5.1.3 version so you already have the update sheets. Model I owners should send in both disks, as substantial changes were made in the program layout on the disks. For the rest of you, here is a brief explanation of the changes.

NEW OR CHANGED PARAMETERS

COPY - The CLONE parameter now defaults to ON, meaning that the password status, mod flag, and date will now be carried automatically during a COPY.

DEVICE - Three new parameters allow the disabling of certain parts of the disk drive, device, and options displayed.

LIST - A new parameter was added to allow all 8 bits of a character to be displayed during an ASCII list.

PURGE - To standardize using the command with and without parameters, PURGE now requires the use of a drivespec.

NEW OR IMPROVED PROGRAMS

BASIC - A short program to translate the command "BASIC" into the appropriate "LBASIC" command. It also forces an EXT=OFF parameter to be executed.

RDUBL - A Model I driver for the new Radio Shack doubler.

PDUBL - A revised version to cure the "silent death" problem.

SYS12/SYS - A new system module containing the execution code to service the @DODIR and the new @RAMDIR vectors. Note that the @DODIR code has been removed from SYS10.

NEW SYSTEM VECTOR

@RAMDIR - This vector was defined by Radio Shack on the Model III, but was not implemented by LDOS. To provide compatibility with upcoming Radio Shack products, this vector was added to both the Model I and III. It deals with getting a directory listing from inside an application program, and is similar to @DODIR. Unfortunately, sometime last spring when the 5.1 version was being developed, a typographical error on a spec sheet put the LDOS @CKDRV vector on the Model III at X'4290'. The true location was supposed to have been X'4209'. You guessed it - @RAMDIR on the Model III is defined by RS to be at X'4290'. So, to provide compatibility with programs written for TRSDOS, the @CKDRV vector has been moved to X'4209'. To clear up any confusion:

```
Model I   - @RAMDIR = X'4396'   (new)
Model III - @RAMDIR = X'4290'   (new)
Model III - @CKDRV  = X'4209'   (moved)
```

This change may affect current assembly language programs that are running on the Model III. By contacting the program author with the above information on the new @CKDRV vector, a one byte patch should cure any problems. The most usual type of error will be a "Illegal drive number" or a "Device not available".

WHAT'S NEW ?

APL*PLUS

From STSC, Inc. comes the announcement of APL*PLUS/80, a version of the APL language for the Model III TRS-80. This version of the APL language is said to be compatible with STSC's APL*PLUS running on the IBM and PDP VAX series computers. Extensive documentation is provided. Included in the promotional information we received from STSC was a statement from a user who was running the system with LDOS and a 5 Meg hard drive. Information can be acquired from STSC, Inc., 2115 E. Jefferson Street, Rockville, MD 20852 (301) 984-5000.

MODEM80

From Les Mikesell comes a communications package called MODEM80. It is a disk oriented system designed to allow data transfer with a wide variety of other computers or terminals. All programs will run in either a Model I or III TRS80 with at least 32K and disk.

Programs are included to permit:

(1) Remote operation of a TRS-80 Model I or III from a terminal or a second TRS-80 through a telephone link - files may be transferred with the unattended computer.

(2) Error free file transfers with another TRS80 or a computer that can use the protocol of the CP/M program "Modem" which is widely used on computer bulletin boards (and available on CP/M user group disk #25).

(3) File transfers with many other types of computers with the TRS-80 acting as a terminal. The communication parameters, character set and control characters may be re-defined as desired.

The terminal program is set up to be easy to use, with most commands available both from menus and control keys. A single line may be transmitted from a file, allowing a more flexible response to prompts from the remote computer. Both transmit and receive files may be active at the same time, allowing the instructions to the remote to be transmitted from one file, while downloading its responses to another file. DOS commands and programs which execute in the lower 16K of memory may be executed while maintaining positions in the transmit and receive files. The previous screen contents are restored when the DOS command is completed.

To order, send $39.95 + $2.00 shipping to:

Leslie Mikesell
4117 W. Hawthorne Trace Rd. Apt 207
Brown Deer, WI 53209

Or send c/o Logical Systems.

LX80 owners note that a modified version which uses the LDOS R5232L/DVR is also available for an additional $10.

ZGRAPH

ZGRAPH by Karl Hessinger is a comprehensive editor for the TRS-80 screen graphics published and distributed by MISOSYS.

The package contains:

1) ZGRAPH - a BASIC program with a machine language support utility.

2) BINCONV - a utility to convert the finished display into several useful formats, including disk load module, packed BASIC statements, BASIC DATA statements, and EDAS source code.

3) DOSAVE - a filter for *KI that will permit dumping the video contents to a disk file useable by ZGRAPH from any program that uses the system *KI driver.

4) BINPRINT - a utility program to print the graphic files using a printer that supports compatible bit graphics.

5) Example files of screen graphics.

Designs may be created by simply using the cursor positioning commands to set or reset the screen graphic blocks, and text can be interspersed as desired. Several complex functions are available to draw lines, rectangles, circles, or to move or duplicate parts of the display. The <R>everse command will reverse all the black/white graphic areas without affecting the text on the screen. Other commands will produce horizontally or vertically Inverted mirror images, or translate all occurrences of any specific character to another character. Partially completed screens may be merged with previously saved files or the contents of any of the 4 buffers used by the program for temporary storage. To assist in the use of the program, there are two helpful menus of commands which can be invoked without losing the screen contents.

Files output by the ZGRAPH program are always written as an image of the screen memory, 64 characters per line, terminated with a carriage return. The utility prog0am BINCONV can be used to convert these files (saved with a /BIN extension) into forms which may be integrated into programs, or the BINPRINT program will print them on a printer with bit graphics (Epson, Okidata, etc.).

If the goal is to modify an existing screen display from some other program, the DOSAVE filter will greatly simplify the task by copying the video contents into a file which may be loaded into ZGRAPH for editing. The ability to convert the the output files into forms which may be integrated into BASIC or machine language programs makes this package useful to anyone wishing to integrate graphics into their programs. ZGRAPH is available from MISOSYS, P0 Box 4848, Alexandria, VA 22303-0848 (703) 960-2998.

NEW PRODUCTS FROM LSI

THE BASIC ANSWER

Logical Systems is proud to announce a new product, called The BASIC Answer. Targetted release date for this product is scheduled for September '82, and the tentative price has been set at $69.

This software package is a BASIC text processing utility, and is designed to allow the BASIC programmer to build his/her programs in a structured manner. "Source" code is written by the user either with a word processor (such as SCRIPSIT), or can even be written in BASIC. After the creation of this source code, The BASIC Answer is used to generate "Object" code, which is the same as a normal "interpretive BASIC" program.

Several advantages are gained by using The BASIC Answer to generate programs as opposed to writing programs in the BASIC environment. The use of absolute line numbers in your BASIC program will be eliminated. Descriptive Labels (up to fourteen characters in length) can be used to reference points of transfer in your program. This concept does away with having to concern yourself with remembering the line number associated with a given routine. By using labels, program code is arranged in a structured manner, allowing for the easy determination of program logic (even if the program has been sitting on the shelf for several light years). Also, if the Label method is used properly, the user may even have the capability of writing BASIC code that is truly relocatable.

In addition to the use of labels, The BASIC Answer introduces a new concept for the use of variables. Variable names may contain up to fourteen characters, all of which are significant to defining the variable. This allows selection of more descriptive variable names. Coupled with this is the implementation of LOCAL and GLOBAL variables. Using local and global variables will eliminate the need to "chase down" variables when determining if the use of a new variable will cause a conflict.

Complete documentation will be provided with the program, and is written in such a manner that it will serve as a tutorial on using The BASIC Answer to its fullest capacities (along with explaining all of the functions of the program in an easy to understand dialogue). Printed output is also provided by the program, and will allow you the option of generating cross-reference tables.

For more detailed information on The BASIC Answer, contact Logical Systems.

QUIZMASTER

Rather than being an LDOS utility, the QuizMaster Series from Logical Systems is an application package made up of several modules. The scheduled release date of this product is September of '82. The initial release will be the "personal" version, and will sell for $39.00.

QuizMaster is an educational/informational question and answer program, and can also be used as a game. Its basic operation is to display a question and four possible answers, and score the operator's response based on the speed of selecting the correct answer and on the number of incorrect choices made. There are three skill levels available, so the operator can choose the speed at which to play. To prevent memorization of the question and answers, the display order of both is randomized every time the program is started. Scoring above a certain value will provide extended operation which can continue as long as no incorrect choices are made. High scores for each speed can be saved on the question/answer set disk.

Five support programs are provided to create or extend, edit, print, and maintain the question/answer set files. Also included is a program to reconstruct a file that has been damaged by disk I/O errors or faulty disk drives or media. Creating files is a simple procedure, requiring only the entering of the question, four possible answers, and the correct answer number.

For ease of entry, an "input editor" allows for full transparent cursor motion and character insert, delete and type over during both the Add and Edit modes. Each question/answer set file may contain up to 255 entries. Three question/answer sets will be provided, containing 100 questions each and covering U.S. Information, General trivia, and one other as yet undecided category. A user can create as many new files as desired.

The main QuizMaster program and the file maintenance, print and recover programs are written in assembly language. The add and edit programs are written in BASIC and use assembly language USR routines to provide quick operation.

An "educational" version of the series is also being developed, and will deal with student names and numbers, and other types of information for classroom use. Additional question set files are also being developed. For more information on the QuizMaster series, contact Logical Systems.

REVIEW by Earle Robinson of:

Microcomputer Math by William Barden, Jr.

Many of us know Bill Barden's well deserved reputation as a writer from the books has he done on the Z80. In fact, in my opinion, his introductory book, published by both Radio Shack (at a very low price of $4.95) and by Sams (at a somewhat higher price) is the best one available. Bill Barden's books are also noteworthy for having been carefully proofed and for containing few serious errors, typographical or others. Bill has recently written a book on math for micros appropriately titled "Microcomputer Math". It is also published by Sams, and the price, as can be expected from that house, is on the high side, $11.95.

For those who are interested in obtaining an understanding of binary arithmetic and wish to acquaint themselves with the hexadecimal base, this book may assist them. It is clearly, though in my view a bit too flippantly, written. As in his earlier works, it is reasonably carefully crafted. Bill knows how to present his subject in such a way that the beginner is not overwhelmed with the jargon which many of us use too often and too freely. We forget that many people don't understand us.

Unfortunately, "Microcomputer Math" neglects to cover binary coded decimal operations. It also neglects to provide concrete examples for coding with the major processors, especially the Z80 and the 6502. Such a book should help its readers to use the knowledge that they have obtained. The absence of such examples detracts from the very real merit of yet another fine book by Bill Barden. Another caveat concerns the illustrations which accompany the text. The publisher has made virtually no effort to 'landscape' the drawings and tables. For a book such as this which will be widely read, this demonstrates a degree of greed which, alas, is a mark of Howard W. Sams & Co, the publisher.

----- PARITY = ODD -----
Subtitle: A MESSAGE to HACKERS

By Tim Daneliuk

You might call this installation of the column "Tim's Soapbox"! In the course of doing product reviews, I come across a wide spectrum of software products. They naturally fall into two groups: SYSTEMS software and APPLICATIONS software. The former are things like operating systems, drivers, and utilities. The latter are packages like database managers, word-processors, and accounting software. Each, of course, has it's place and (hopefully) an audience of users. However, I've recently noticed a TERRIBLE practice which seems to be pervading the applications software industry. If you write applications programs, READ THIS!!! If you buy such programs, BEWARE!!!

Before I describe this great programming evil, I really ought to tell you how I came to write this Quarterly's column. I was innocently reviewing a database manager for this issue, when I noticed that all of a sudden KSM wasn't working. Further inspection revealed that the program had taken over the printer, keyboard, and video devices with it's own drivers. That wasn't so bad, except that DEVICE showed the respective DCBs for these devices pointing to an area in memory well below HIGH$! That's right folks, the next program that loaded into memory below HIGH$ overlayed these "drivers" (and I use the word loosely) and caused a SYSTEM (BLOW-UP)! Naturally, this tended to bias my review of the software somewhat. A call to the author yielded a "We never had that problem before." It seems that the program in question had been used only with other programs that had their own internal drivers. Consequently, the problem never showed up.

So, here's my gripe: "APPLICATIONS SOFTWARE HAS NO BUSINESS FIDDLING WITH, MODIFYING, OR "IMPROVING" THE OPERATING SYSTEM...EVER!!!" (Well, "Hardly Ever!" to quote Gilbert and Sullivan). Friends, the whole point of the operating system is to insulate you from the harsh realities of the hardware. LDOS especially gives you all the entry points any reasonable and competent programmer should need. In 99% of the cases, the programmers at LSI have presented you with code as good as or better than your own. In those rare instances where you must go directly to the hardware itself, the very least you can do is restore the system to what it was before you started changing it. In the particular example cited above, everything would have been fine if the programmer had thought ahead a little and restored the DCB pointers when exiting his program.

Now, don't get me wrong. If you're writing programs just for yourself, do whatever feels good! But if you're going to peddle the ultimate program, here are a few thoughts to consider (by the way, this also applies to you BASIC fanatics who insist on destroying the DOS with PEEKs and POKES!!!):

TIM'S TIPS FOR TERRIFIC, TRANSPARENT, TRAUMA-FREE BIT TWIDDLING

1) HONOR THE DCB VECTORS WHENEVER POSSIBLE (which should be well over 90% of the time). Try filtering the device instead of re-writing the driver.

2) IF YOU MUST WRITE YOUR OWN DRIVERS, MAKE THEM "TRANSPARENT" TO THE SYSTEM. That is, restore the old DCB vectors when your program exits.

3) ALWAYS HONOR THE HIGH MEMORY POINTER (HIGH$). Even if all you do is check to see if your program fits into the available space (and aborts if it doesn't), this is better than the usual "try it and see" mentality of many program "philosophies".

4) BE SURE YOU KNOW WHAT YOU'RE DOING IF YOU DO GO INTO THE HARDWARE DIRECTLY. EXAMPLE: I don't know how many programs which check the printer directly on Model I systems do it this way:

```
LOOP    LD      A,(37E8H)
        CP      3FH
        JR      NZ,LOOP
```

This is WRONG!!!!!!! It is so bad, that it may even be criminal!! The reason is, that only the upper nibble (bits 4-7) actually return the printer status. The four lower bits' values will vary with different hardware. In a Tandy interface, a printer ready will show a X'3F', while in the LOBO LX-80 it will be X'3C' unless a hard disk is being used. In the latter case, the lower nibble will take on different values as the hard disk is accessed. The point is that the four low bits should be masked. For example:

```
LOOP    LD      A,(37E8H)
        AND     0F0H
        CP      30H
        JR      NZ,LOOP
```

This is the way the Model I ROM does it. I am really very tired of disassembling "review" software so I can patch it to work on the LX80. This sort of thing is just sloppy technique and has no place in software to be sold.

5) Point four brings up another thought: DON'T RELY ON UNDOCUMENTED HARDWARE OR SOFTWARE FEATURES TO MAKE YOUR PROGRAM RUN. In the example above, no one ever guaranteed what the low four bits would be, so good practice dictates "defensive" programming.

6) PRESERVE EVERY REGISTER YOU USE. Just like the DCB pointers, your program should handle the registers in a "transparent" fashion.

7) FOR MAXIMUM PORTABILITY, STAY OUT OF THE ALTERNATE REGISTERS. Just because they aren't used now, doesn't mean they never will be. Many systems, like interrupt driven multi-user schemes, will positively DIE if you foul the alternate registers with your code.

8) ALL HIGH MEMORY OPTIONS SHOULD BE SELF-RELOCATING. They should also reset HIGH$ accordingly. This is not that difficult to do, and saves the end user a lot of frustration and time.

9) TRY TO TEST NEW SOFTWARE ON ALL OF THE TYPES OF SYSTEMS IT WAS DESIGNED FOR. This will probably eliminate the majority of what I call "retail-time debugging"!

This list is by no means exhaustive. I think, however, that you offenders out there get the point. There is nothing more frustrating for the non-technical user than to buy a premium DOS (LDOS of course!) and find that applications programs won't run. If you do have a piece of software that has problems on a system it claims to be compatible with, you should do two things. First, contact the manufacturer and ask them to correct their problem. Be very sure, however, it really is THEIR problem. No one likes to hear that their programs won't run because you can't or won't read a manual. Secondly, if the problem is not resolved, pass the word around. Don't be slanderous, but if you're sure a problem really exists, let other people know. If more people did this, the dishonest software manufacturers would be forced to fix their code or go out of business.

What about this issue's program review? Well, when I called the author, he told me that a new version of the program was soon to be released, and that he would try to fix the problem. Since the version of the program I have will no longer be available, there isn't much point in reviewing it now. I have, however, been promised a copy of the new version. I will review that release. Hopefully, he'll have read my "TIPS" by then......

While I don't have any complete software reviews for this issue, a few products have come to my attention which I'd like to mention in passing. First of all, Earle Robinson has released a new version of his DISCATER disk cataloging system. This version has many new features including "wild-card" listings, room for more file entries (a real must, for those who run DDEN double-sided drives, hard disks, or 8" floppies), and listings of every occurrence of a file where it appears on multiple disks. Earle has also managed to optimize his sorting routine a bit more. This is an excellent program that runs smoothly under LDOS. There is also an upgrade policy for those of you who had the old version of the product. DISCATER is available from:

softERware
16007 Miami Way
Pacific Palisades, CA 90272

You can also reach Earle on the LDOS bulletin board. His ID# is 70135,141.

Powersoft has released a set of seven LDOS Utility Disks for systems running under LDOS. These perform tasks such as disk reformatting, directory checking, and providing the user with an "on-line" help file. I've just gotten these in, so I have not had much chance to really evaluate the products. Those that I have used seem to work well. Each disk is $29.95 and they are available from:

Powersoft
11500 Stemmons Expwy.
Suite 125
Dallas, TX 75229

Finally, I must plug the FILTER Disk from LSI. Though this product has been mentioned before, it deserves further notice. This disk sells for $60 and comes with many useful filters on it. LSI has also included the Source Code for each filter so you can learn how to write your own. One of my favorites in this package is CALC/FLT.

This filter allows you to perform base conversion and arithmetic from within an applications program. I use this all the time to do decimal/hex conversions from within an editor-assembler. Another interesting filter is XLATE. This allows full bi-directional translation of data in any I/O path. Using XLATE, you can make your TRS-80 "speak" EBCDIC or BAUDOT instead of ASCII. There are many time saving filters on this disk, and if you do any serious programming whatsoever, you should own this product.

.....er.....

(EDITOR'S NOTE: This is the first in what will hopefully be a regular series of articles on assembly language programming by Earle Robinson.)

As a veteran of Hewlett Packard programmable calculators, I learned to program very tightly indeed. Do any of you remember the HP-65 which allowed the user to have up to 100 lines, or the HP-67 which permitted 224 lines, and had 23 memory locations? If I tell you that I once wrote a program for the latter for the evaluation of a 6 x 6 matrix, you will appreciate the constraints.

Therefore, when I began to program on the Mod I, first with only 16 K of RAM in the Expansion Interface, then with 32 K, I retained the habit of using tightly written code. In fact, there are three criteria for any program development which should be kept in mind, and my comments in this article, as well as those which may follow should be read with them in mind. These criteria are:

1) Document your program
2) Write tight code
3) Write code which is fast

As some of you may surmise these three can conflict with each other all too often, especially because tight code is not always the fastest. Further, you can (and I have, I must admit) waste a lot of time in saving those last few milliseconds.

The first step in improving your programming skills, and I am assuming that you already can write something, is to study the instruction set and note where the T cycles are the longest. A lot of hot air is wasted by people talking about the number of megahertz at which the processor operates. The reason for which the 6502 chip, used in the Apple, is often faster, is because many of its instructions use fewer T cycles; although the Z80 does have a better instruction set, it is often slower than the 6502 due to the T cycles. Also, the number of bytes used per instruction doesn't tell you everything at all.

Here is an example. Let us assume that you are comparing strings addressed by the HL and the DE registers. After each compare, you will increment each of them, with the 'INC HL' and 'INC DE' instructions. Each is a one byte instruction. However, if you could use 'INC L', and 'INC E' instructions instead, and you are doing a number of such comparisons, you will save a lot of time. These instructions are also one byters, and use 4 T cycles versus the 6 T cycles used by the former. Of course, you must be certain that

neither of the increments will reach the value of FF Hex for reasons that I will leave the reader to ponder. It is also manifestly ridiculous to go to this trouble if you are only doing doing such a comparison on a single, short string.

A second example will show how tight code can be slower. Look at the number of T cycles for an absolute jump. It is 10. Now, look at the cycles for a relative jump. There are 12. However, the absolute jump is a three byte instruction; the relative jump is a two byte instruction. Naturally, common sense will tell you that it silly to worry much about this unless such a jump is repeatedly made within the program. Further, it is more difficult to relocate a program containing absolute jumps.

In the next issue of the Quarterly I'll try to provide some more insights on programming, using a horror story as the basis for the discussion.

LDOS - It's Greek to Me - by Charles Knight

Both LDOS and the LDOS filters disk offered for it have added immeasurably to my wife's happiness. About the time LDOS 5.1 was first released, she began studying New Testament Greek in her spare time. As is the case with any well disciplined student, she forces herself to translate not only from Greek to English as in most Greek courses at seminaries and universities, but also from English into Greek. I keep telling her that she's a glutton for punishment, but she keeps plodding along doing her thing.

When the LDOS Filters disk became available, I bought it thinking about its utility to the operation of my computer and never gave a thought that it might help my wife with her studies. Since I speak Greek about as well as I do Sanskrit, Russian, or Campa indian (not at all), I never thought to bring all my resources together until my wife asked if it were possible to type Greek into Scripsit. She expected a resounding "No"! She received a somewhat hesitant "Maybe". I had purchased a General Scientific printwheel for my Diablo printer some time ago at a sale where I got it for $2.00 and bought it simply because it was so cheap, never thinking that I'd someday have a use for it.

The first point to note is that the character set available on this wheel was not intended for the Greek language, but rather for scientific applications; therefore the characters are not necessarily in the best place on the print wheel and only a complete set of lower case Greek letters is available. Uppercase letters are present, but not a complete set of them. A period is also missing, but a raised dot is available that makes a pretty good substitute.

Putting our two heads together hoping that the total intelligence might approximate that of a half-wit, we designed the following table for use by XLATE/FLT:

```
.table for use with XLATE/FLT and Greek print wheel
"A"="a" "B"="b" "G"="q" "D"="w" "E"="e" "Z"="z"
"H"="h" "V"="r" "I"="i" "K"="k" "L"="g" "M"="m"
"N"="n" "C"="u" "O"="o" "P"="!" "R"="p" "S"="s"
```

```
"T"="t" "U"="y" "F"="d" "X"="v" "Y"="c" "W"="l"
"g"="q" "d"="w" "v"="r" "l"="g" "c"="u" "p"="!"
","="^" "."=22 "'"="["
. ^ = right arrow on video, clear + on kbd
. [ = up arrow on video, clear < on kbd
.end table
```

This was simple to do and enabled an elementary sort of Greek word processing to be done. The next thing my wife said she wished she could do is to place an iota subscript in her text. What the heck is an iota subscript? I asked, (I thought from her pronunciation she was saying Yoda subscript and had spent too much time listening to Starwars records)! At any rate, we decided that the iota to be subscripted should be assigned to the capital "I". First we had to change the "I"="i" in the translation table to: "I"="I" effectively bypassing translation of this character. It was also necessary to modify the SLASH0/FLT source code thoughtfully provided with the filters disk. The modification was easy, requiring only the addition of 18 lines of code and the changing of a few of the messages. (This was fortunate, since I program in assembly language only slightly better than I speak Greek!) Unfortunately, it is impossible to have right justification of text at the same time as the iota subscript, so be sure to include the format line, J=N somewhere near the top of the text. The following lines were either changed or added. The ones that have been added have line numbers not ending in zero:

```
00100 ;****    IOTA/FLT
00501          LD    (OUTP4+1),HL
00502          LD    (OUTP5+1),HL
00503          LD    (OUTP6+1),HL
00750 SIGNON   DB    31,'IOTA/FLT - LDOS  Line  Printer Filter  - Version
1.0',0AH
00810 IOTA     JR    START
00840 FNAME    DB    'IOTA'          ;new filter name
00890          CP    'I'             ;Capital IOTA?
00930 OUTCF    LD    C,08            ;send backspace
00950          LD    C,27            ;send escape
00970          LD    C,'U'           ;1/2 linefeed
00971 OUTP4    CALL  0000H           ;send it
00972          LD    C,'i'           ;send iota
00973 OUTP5    CALL  0000H           ;send it
00974          LD    C,27            ;escape
00975 OUTP6    CALL  0000H           ;send it
00976          LD    C,44H           ;D neg 1/2 linefeed
```

When the source is first loaded into EDAS, issue the command: N,100,10 and the text will be given numbers to match these. Then change or add each line shown. You might need to send a different escape code sequence, depending on your printer.

After this was done, my wife had another chore for me to do. She wanted me to make a filter to change the comma and dot characters into a

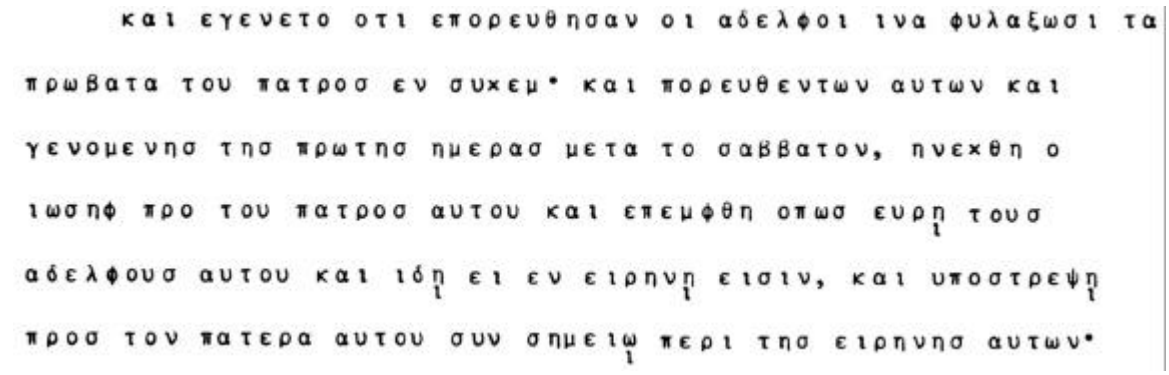
semicolon which, in Greek, is used as a question mark. We assigned this character to the question mark on the keyboard rather than to the semicolon. To accomplish this, the following lines of code were added to or changed from the original unmodified SLASH0/ASM source code:

```
00470 SIGNON    DB     31,'QMARK  -  LDOS  Line  Printer Filter - Version
1.0',0AH
00830 FNAME     DB    'QMARK'
00880           CP    '?'             ; ?  ?
00921           LD    C,5EH           ;Comma character
00940           LD    C,08H           ;Backspace
00960           LD    C,'"'           ;Period character
```

With the three filters residing on the disk and the Translation table which we have named GREEK/XLT also present, it is necessary to install the filters in the following order. If the order isn't right, neither will the output to the printer be right. We have made a /JCL file for this:

```
RESET *PR
FILTER *PR IOTA
FILTER *PR QMARK
FILTER *PR XLATE GREEK(OUTPUT)
SCRIPSIT
```

Now all that is necessary to do Greek word processing is to type: DO = GREEK at LDOS Ready. While it's not yet perfect, we'll have to wait for Diablo or Qume to bring out a more appropriate wheel for the Greek language to get any better. It is indeed fortunate that I own two computers, for now that she can do her studies on the computer, I'd hardly get to use mine if I didn't have two. It would be nice if she were taking the course at a university so she could see the instructor's face when he is handed a perfectly typed piece of Greek homework. My wife now tells me that she will study Hebrew when she has finished her Greek. Now, if anyone has a Hebrew daisy wheel for sale cheap....



HINTS FROM OUR USERS

From Jerry Latham of Midwest City, Oklahoma, comes the following on the MX-80 with the Graftrax ROM.

******

Epson MX-80 printer users who have the Graftrax-80 option installed can now stop patching, filtering, and otherwise messing around with either their software or hardware in an effort to get the Epson to properly print the TRS-80 Model I graphics codes. On page 11 of the Graftrax manual is documented the (apparently little known) command <CTL><:>. That is hex 1B and 3A or decimal 27,58. However you get it to the printer, be it by POKEing or LPRINTing the result is the same . . . the printer will now do the proper shifting of graphics codes from the Model I to the equivalent Epson code. The added benefit you get is that no other printer functions are lost. You will no longer have the command codes listed in the original manual on page 82, but those are all duplicated below 20H anyhow.

Using this method of controlling the printer will in no way effect any other printer functions in effect when the codes are sent. It does not reset the printer (that is reserved for <CTL><@> sequence). Remember that this function will be destroyed if you turn off the printer or send it the <CTL><@> (1B 40 hex, 27,64 decimal) sequence. There is one other way to return the printer to its power up graphic code status. That is by sending the <CTL><;> sequence to the printer. That is 1B 3B hex, or 27,59 decimal. That sequence simply causes the MX-80 to revert back to its original group of graphics codes at 160 to 223 decimal.
******

From Peter C. Trenholme of Montclair, New Jersey comes this information about using the MODEM-80 hardware and the included DTERM program. This patch makes DTERM's buffer honor HIGH$, so that high memory drivers such as PDUBL can be used. It is for the Model I.

******
```
X'5936'=C3 7E 59 00 00 00 00
X'597B'=3F 20 00
X'597E'=D5 EB 2A 49 40 EB 23 AF 77 DF 20 FA D1 C3 3D 59
X'56BC'=2D 40
```
******

ITEMS OF GENERAL INTEREST

This is the TTimer patch for the new 5.1.3 version.

```
.TTIMER Model I Version 5.1.3 patch
D04,08=D2 45 ED 78 0D CD A6 47 ED 78 0D E6 0F 85 12 1B
D04,18=C9 11 43 40 01 C5 03 CD C3 45 10 FB
D0D,51=2l 46 40 01 CA 01 CD B8 4E 06 03 CD B8 4E 01 CC
D0D,61=0F CD B8 4E EB DB CB E6 03 21 B9 50 20 01 34
D0D,70=l8 21 ED 78 0D A0 07 57 07 07 82 57 ED 78 0D E6
D0D,80=0F 82 77 2B C9
```

WE SURRENDER! Following are the patches to Version 5.1.2 and 5.1.3 to make the /BAS extension parameter default to OFF. However - we would rather have you rename your current LBASIC programs to have a /BAS extension, thus providing instant ID at a glance and allowing you to view, move or remove them as a class with the DIR, PURGE and BACKUP commands. There is a program in the USER PROGRAM section of the newsletter that provides a "mass rename" capability that should help with the task. The new BASIC/CMD program also can be used to enter LBASIC, as it forces a EXT=OFF condition. But, if you really must, apply the following patch to LBASIC/CMD:

```
. MODEL I patch to make EXT=OFF the default
. Good for Versions 5.1.2 and 5.1.3
D11,3C=00 00
.EOP


. MODEL 3 patch to make EXT=OFF the default
. Good for Versions 5.1.2 and 5.1.3
D11,25=00 00
```

For Model I users with 5.1.2 dated 5/25, and for version 5.1.3, the following patch to PDUBL or RDUBL will allow you to write the old DAM. The patch to SYS0 is the same as for the original 5.1.2 release.

```
.Patch PDUBL/CMD
X'5476'=A9

.Patch RDUBL/CMD
X'54AA'=A9

.Patch SYS0/SYS
X'467C'=A9
```

Requests from those of you running bulletin boards with LDOS brought about the following patch. When implemented, the ability to access a file with the LDOS Master Password will be disabled. This should allow you to keep people out of files they don't belong in. Of course, PURGE and BACKUP by Class will not function with password protected files.

```
.Patch SYS2/SYS, Model I
D02,0B=FE

.Patch SYS2/SYS, Model III
D02,29=FE
```

Roy's Technical Corner

It seems that everytime I sit down to write another article in this technical series on LDOS, I want to start it out with, "One of the most important and least understood aspects of LDOS is..." Is it because LDOS is so complex that very little about its inner workings has been known? Or perhaps is it that I feel many things about LDOS are so very important that I wish every good programmer would be knowledgeable about the entire system. I have begun to get feedback on this series; albeit requests for more information. It is a very good idea to put forth your comments. Need I expand on a previous article's discussion? Is there something you want discussed? The QUARTERLY is a two-way street. If you share your views and opinions, I may be able to clear up some misunderstood point.

This issue's RTC stems from such a user request. How do we use the task processor? To begin such a topic, it is best to agree on the terms I will be using. This is as good a place as any to present the concepts of "foreground" and "background". To those in data processing, you already have some idea what they mean. This idea may be in contrast to how we use these terms for LDOS, so pay attention. In LDOS, any activity that uses most of the CPU's time is called a foreground activity or foreground task. This could be a running LBASIC program, an LSCRIPT editing session, or the development of an assembly language program using EDAS. While you may think that your task is utilizing 100 percent of the machine, in reality, other activities (or tasks) are taking place.

Notice that cursor blinking in front of you (of course it's not blinking in front of you since you are reading this article from paper; however if you were looking at the LSCRIPT file on your computer as I am, you would see the cursor blinking - so just imagine it...). Some activity is causing it to blink at periodic intervals. Turn on the clock display. Notice how it keeps near perfect time (once it is set) and updates the video display every second! How about recognizing that every time you depress the keyboard, LDOS is saving your entries until your application program (or LDOS itself) requests keyboard input. What each of these activities has in common is that they all are occurring "behind our backs", so to speak. Each activity takes but a brief moment to accomplish its task, essentially sneaking its time by interrupting the foreground task. They are taking such a small amount of time compared to the amount of time our foreground tasks are taking, they are called "background" tasks.

There is a module within LDOS that I call the task processor. This is the boss that supervises all of the background tasks so that each gets its turn at sneaking CPU time. The task processor works in concert with and depends on part of the machine's hardware that generates interrupts to the CPU. These interrupts are the way in which a background task can gain control from the foreground task. It's therefore a good idea to touch on some of the hardware aspects of interrupts.

There is an input to the Z-80 (pin 16 to be exact and labeled INT*) that is used to specify a hardware interrupt request. The Z-80 has two program instructions that are used to accept or ignore these requests. If the Disable Interrupt (DI) instruction has been executed, the Z-80 will completely ignore any activation of the lead. An Enable Interrupt (EI) instruction is used to have the Z-80 honor this lead. (CONTINUED Page 29). . .

Super Utility Plus I/III (LAST CHANCE at THIS price!). 49.95
Super Uitlity Tech manual 14.95
Inside SU+ Manual 19.95
PowerDRAW 39.95
PowerPRINT 29.95
PowerTERM 29.95
SCRIPLUS 3.0 ..... 39.95
The BASIC/S Compiler System 89.95
Make/80 Mod I 14.95
Make/80 Mod III 24.95

Utility Disks for LDOS(tm) Mod I or III by Kim Watt... 29.95 ea.
Utility Disks 2-8 all in one package (special) 149.95
contains:

PMOD/CMD	PCHECK/CMD	PFIX/CMD	PREFORM/CMD
PFIND/CMD	PFIND/CMD	PCOMPARE/CMD	PVU/CMD
PERASE/CMD	PCLEAR/CMD	PSS/CMD	PMAP/CMD
PMOVE/CMD	PDIRT/CMD	PASSGO/CMD	PUN/CMD
PEX/CMD	PMX/FLT	PHELP/CMD	PBOOT/FIX
PFILT/FLT	DVORAK/FLT	DVORAK/JCL	CODE/JCL
DECODE/JCL			

SNAPP Enhancements for LBASIC are HERE!! Mod I as well as III!

SNAPP-II Extended BASIC 79.00
SNAPP-III Extended Build-in Functions 69.00
SNAPP-IV Extended Mapping Support 69.00
SNAPP-V   Extended File Mapping Support 59.00
SNAPP-VI College Educated Garbage Collector 69.00
SNAPP-VII Reverse Compression 39.00

SNAPP-WARE LDOS Trial Package (ALL the above)..... 10.00

Specify Mod I or III LDOS!!

NOTE: This trial package is a great way to explore the power of "Snapped" up LBASIC. The $10 is fully deductable from the future purchase price of any LDOS Snapp package. The trial package is NOT backupable, and is only good for a certain number of LBASIC executions before it expires. (Don't worry... you get plenty!)

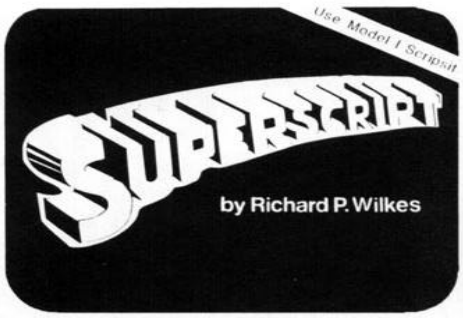
BASF 40trk-DD certified BOX of 10 ..... 29.95
Mod I or III Green Screen (specify) 17.50

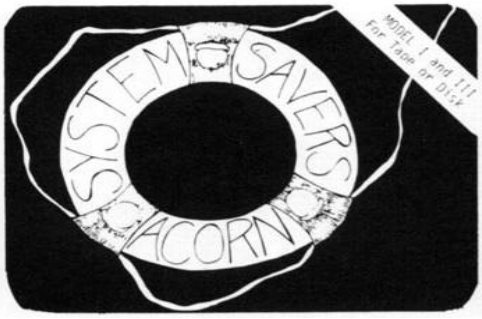
Write for full catalogue. -- Dealer inquiries welcome!

POWERSOFT (tm)

11500 STEMMONS EXPRESSWAY. SUITE 125
DALLAS, TEXAS 75229
PHONE (214) 484-2976
MICRONET 70130,203

POWERSOFT IS A DIVISION OF BREEZE/QSD, INC.





SuperScript
By Richard Wilkes

An enhancement program to Radio Shack's Scripsit, SuperScript turns a good word processing system into a great one!
Depending on your printer's capabilities you can superscript, subscript, underline, boldface, select 10/12 pitch and slash zeroes. Brackets, braces and carets can now be entered from the keyboard. You can get a Directory and kill files within SuperScript without losing text. You can pause while printing and insert text into unjustified lines. Eleven drivers are included with SuperScript -- one of which should work with almost any type of printer. However, not all features are available on all drivers ... and not all features are possible on all printers. Serial drivers are provided which use the ETX/ACK protocol for 1200 baud communications. Special drivers are provided for the NEC 5510, NEC 5530, Daisy Wheel II, Lineprinter IV (Centronics 737), Diablo printer, and Epson MX80 (Graftrax owners can get underlining and italics). Custom serial and parallel drivers are included which can be modified to provide some or all features on most standard printers. Both Model I and Model III versions require Model I Scripsit. Model III owners use the Model III TRSDOS "CONVERT" utility. SuperScript for Model I is designed to work with TRSDOS; Model III version also is **LDOS COMPATIBLE**.

32K Disk $50.00 plus $2.00 shipping and handling.

System Savers
By Tom Stibolt

Two machine language utility programs designed to make your use of SYSTEM format tapes easier and more enjoyable -- You can make backup copies of standard SYSTEM tapes on **either** tape or disk.

System Savers has two different programs on the cassette: FLEXL lets you merge two or more SYSTEM tapes into a single tape, merging machine language routines into one file. On the model III, baud rates can be changed, allowing low baud rate tapes to be re-written to take advantage of Model III's high baud rate. FLEXL enables the user to make and verify backup copies of programs written in the TRS-80 SYSTEM format.

TDISK allows the user to save programs from SYSTEM format tapes onto disk. It's specifically designed to allow saving and running disk programs that reside in the same location as TRSDOS. TDISK will automatically load programs with non-contiguous blocks.

16k Model I/III Tape (transferable to disk)
$19.95 plus $2.00 shipping and handling.

Structured Basic Translator
By Gene Bellinger

Structured Basic Translator allows a whole new world of computer programming. No longer will you be lost in a confusing sea of line numbers, GOTOs and GOSUBs. No longer will you repeatedly type subroutines that you use often. No longer will you be confused by BASIC programs that are inadequately documented due to memory restrictions. Instead, you can begin to program in a clear straight-forward manner that yeilds source programs that will be as easy to understand five years from now as they are when you write them.

Structured Basic Translator allows you to develop this new way of programming -- actually a new way of thinking -- without having to learn a whole new language. SBT utilizes the features of BASIC that you already know and provides a means to make program writing and documentation clearer and more efficient.

See the product review in **LDOS Quarterly** v1 #4.

32k Disk Model I (Model III owners use CONVERT utility.)
$49.95 plus $2.00 shipping and handling.

L S I P R O D U C T G U I D E

O P E R A T I N G S Y S T E M S

STOCK #	ITEM NAME (TITLE)	RETAIL	S & H
L-10-010	LDOS 5.1.x Model 1	$ 129.00	$5 + 1 The ULTIMATE DOS
L-10-030	LDOS 5.1.x Model 3	$ 129.00	$5 + 1 The ULTIMATE DOS
L-11-010	smal-LDOS Mod 1 W/Man	$ N/A	** Available to OEMs ONLY!! **
L-11-015	smal-LDOS Mod 1 WO/Man	$ N/A	NOTE: NOT TO BE SOLD RETAIL
L-11-030	smal-LDOS Mod 3 W/Man	$ N/A	Special pricing is available
L-11-03b	smal-LDOS Mod 3 WO/Man	$ N/A	for orders over 100.

L A N G U A G E S

STOCK #	ITEM NAME (TITLE)	RETAIL	S & H	
M-20-010	EDAS 3.5.x Mod 1 & 3	$ 79.00	$4 + 1	MISOSYS Editor/Assembler
M-20-015##	EDAS x.x.x Mod 1 & 3	$ 100.00	$4 + 1	## Not yet released
L-20-020	EDAS 4.0.x Model 2	$ 99.00	$4 + 1	Model II Editor/Assembler
L-20-022##	EDAS 5.0.x Model 2	$ 199.00	$4 + 1	## Not yet released
L-2l-010##	LC - "C" Compiler	$ 150.00	$5 + 1	## Not yet released
A-21-020	ALCOR Pascal	T/B/A		## To be announced
L-21-030##	The BASIC Answer	$ 69.00	$3 + .50	## Released Sept. 82
S-25-010	Snapp Trial Model 1	$ 10.00	$1.50+.50	Trial BASIC package
S-25-030	Snapp Trial Model 3	$ 10.00	$1.50+.50	Trial BASIC package
S-25-040	Extended Basic Mod 1	$ 79.00	$1.50+.50	Snapp's extended BASIC
S-25-050	Extended Basic Mod 3	$ 79.00	$l.50+.50	Snapp's extended BASIC

U T I L I T I E S

STOCK #	ITEM NAME (TITLE)	RETAIL	S & H	
L-30-010	FED - LDOS File Editor	$ 40.00	$2 + 1	
L-30-020	LED - LDOS Text Editor	$ 40.00	$2 + 1	
L-30-030	I/O Monitor	$ 25.00	$2 + 1	Disk I/O error monitor
L-30-040	MemDISK	$ 39.00	$2 + 1	Build a disk in memory
L-32-050	Filter Package #1	$ 60.00	$2 + 1	General purpose filters
L-32-060##	Filter Package #2	$ T/B/A		## Not yet released
L-32-070##	Utility Package #1	$ 50.00	$2 + 1	## Not yet released
L-32-080##	Utility Package #2	$ T/B/A		## Not yet released
M-35-200	Disassembler 2.x	$ 25.00	$2 + 1	MISOSYS disassembler
M-35-210	PDS	$ 40.00	$2 + 1	Partitioned data sets
M-35-220	CONV/CPM	$ 30.00	$2 + 1	CP/M to LDOS utility
M-35-230	CONV800	$ 50.00	$2 + 1	8080 to Z80 source xlater
M-35-240	SOLE (boot DDEN)	$ 25.00	$2 + 1	Boot DDEN on Model I
M-35-250	HELP	$ 25.00	$2 + 1	HELP program and Q/R card
M-35-260	Graphic support pack	$ 50.00	$2 + 1	MX-80 and MX-100 utility

M A N U A L S

STOCK #	ITEM NAME (TITLE)	RETAIL	S & H	
L-40-010*	LDOS 5.1.x Model 1 Without Binder	$ 49.00	$3 + 1	
L-40-030*	LDOS 5.1.x Model 3 Without Binder	$ 49.00	$3 + 1	
L-40-035*##	LOOS Operators guide	$ 10.00	$2	
L-40-039*	LDOS exchange 1 & 3 Without Binder	$ 29.00	$3 + 1	
L-40-040	smal-LDOS 5.1.x	$ 20.00	$2 + 1	
L-40-050	LBASIC 5.1.x	$ 9.00	$2	
L-40-060*##	Ref. Card for 5.1.x	$ 5.00	$1	
L-40-600	Other product manuals	$	50% of the retail price of the product.	
L-40-700*	Replacement Manual	$	25% of the retail price of the product.	
L-44-010##	LSI 3-ring binder Sm.	$ 7.00	$3	
L-44-020	LSI 3-ring binder Lg.	$ 10.00	$4	
L-44-025	TAB Index set (any)	$ 2.00	$1	
L-49-101	Vol 1-1 Jul '81	$ 5.00	$0	LDOS Quarterly newsletter
L-49-102	Vol 1-2 Oct '81	$ 5.00	$0	
L-49-103	Vol 1-3 Jan '82	$ 5.00	$0	
L-49-104	Vol 1-4 Apr '82	$ 5.00	$0	
L-49-105	Vol 1-5 Jul '82	$ 5.00	$0	

A P P L I C A T I O N S

STOCK #	ITEM NAME (TITLE)	RETAIL	S & H	
L-50-010	Mail/File II Mod 1	$ 159.00	$5 + 1	Mailing list, 0 names
L-50-020	Mail/File II Mod 2	$ 199.00	$5 + 1	Mailing list, 2500 names
L-50-030	Mail/File II Mod 3	$ 159.00	$5 + 1	Mailing list, 1200 names
L-50-040	Mass/Mail Mod 2	$ 795.00	$0	Mailing list, 10500 names
L-50-042	M/M Zone Count opt.	$ 100.00	$0	2nd class mail count
L-50-044	M/M Country opt.	$ 100.00	$0	Foreign list option
L-50-046	M/M Reconstruct	$ 100.00	$0	Recover damaged diskettes
L-50-049	M/M Demo Package	$ 50.00	$4 + 1	
L-50-050	Inventory Mod 1	$ 159.00	$5 + 1	Retail users, 2700 items
L-50-060	Inventory Mod 3	$ 259.00	$5 + 1	Retail users, 2700 items
L-50-070##	Inventory Mod 2	T/B/A		## To be announced
M-50-210	ZGRAPH Mod 1 & 3	$ 40.00	$2 + 1	Screen graphics package
L-50-410	Stock Market Mod 1	$ 99.00	$4 + 1	
L-50-420	Stock Market Mod 3	$ 99.00	$4 + 1	
L-51-500##	Quiz-Master Mod 1 & 3	$ 39.00	$2 + .50	Game or educational
L-51-500##	Quiz-Master Series "E"	$ 99.00	$5 + 1	Educational quiz program
L-51-610##	Q/M - Amer. History	$ 20.00	$2	Quizmaster Q & A sets
L-51-620##	Q/M - Geography	$ 20.00	$2	
L-51-630##	Q/M - Solar System	$ 20.00	$2	
L-51-640##	Q/M - Music Trivia	$ 20.00	$2	
L-51-650##	Q/M - Math General	$ 20.00	$2	
L-51-660##	Q/M - Science General	$ 20.00	$2	
L-51-699##	ANY THREE Q/M MODULES	$ 20.00	$3	
L-55-010	Ultra-Trek Disk 1 & 3	$ 20.00	$2	Star-trek program

S E R V I C E S

STOCK #	ITEM NAME (TITLE)	RETAIL	S & H
L-70-010	Update 5.0 to 5.1	$ Calculated	
L-70-015	Five Dollar Update	$ 5.00	N/A
L-70-020	Ten Dollar Update	$ 10.00	N/A
L-70-030	Special Update	$ As required	
L-70-040	Spc. disk With Media	$ 10.00	N/A
L-70-045	Spc. disk W/O Media	$ 5.00	N/A
L-71-010	Technical services	$ 50.00 per hour	
L-71-020	Programming Basic	$ 40.00 per hour	
L-71-030	Programming Assembler	$ 50.00 per hour	
L-71-040	Design & Analyze	$ 60.00 per hour	
L-71-050	Author Tech. Doc.	$ 40.00 per hour	
L-72-100	Out of town, Per day	$ 300.00 plus expenses	
L-72-900	Special Services	$ By qoute	
L-75-100	5.1.x Extended support	$ 25.00	N/A

M I S C E L L A N E O U S I T E M S

STOCK #	ITEM NAME (TITLE)	RETAIL	S & H
L-90-015	5 Inch floppy disk	$ 5.00	N/A
L-90-018	8 Inch floppy disk	$ 8.00	N/A
L-90-019	8 Inch Dbl. Sided	$ 10.00	N/A

* - Indicates that the product is available to registered owners ONLY.

- Indicates that the product has not yet been released.

All Prices and Specifications are subject to change without notice.

==========
Shipping to foriegn countries -

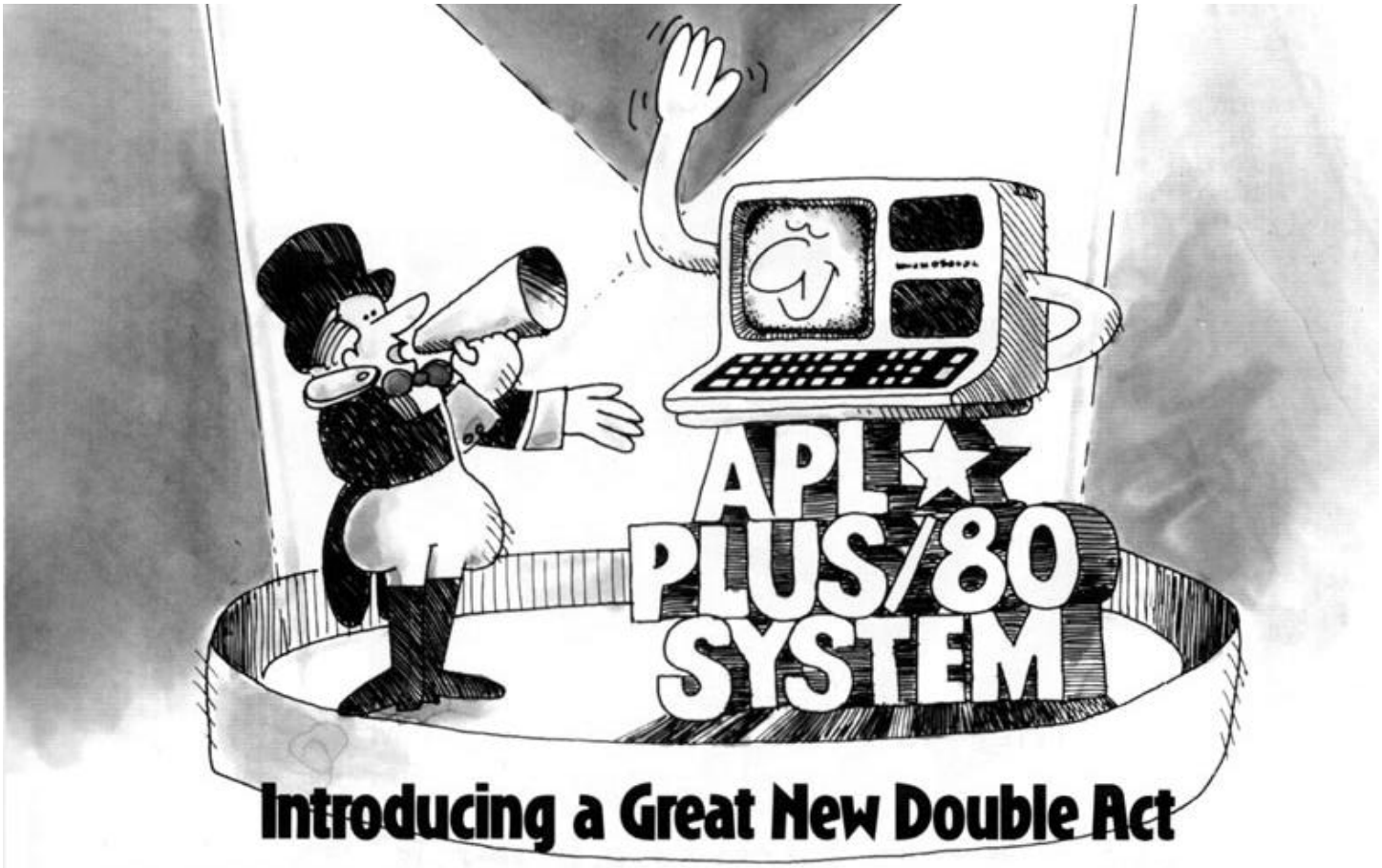
Shipping to Canada and Mexico: Multiply shipping times TWO
Shipping elsewhere: Multiply shipping times FOUR
==========

LSI will pay normal shipping and handling charges on any order over $100.00 that is pre-paid with a check or money order.

Most domestic orders are shipped standard UPS. Foreign orders are shipped via air mail. Special shipping via UPS Blue Label and overnight express is also available. Special shipping charges will be determined by the shipper's current rate chart.

A V A I L A B L E M E D I A

All LSI products are supplied on LDOS compatible media. For the Model I, single density 35 track media is provided. For the Model III, double density 40 track media is provided. For an additional charge of $5 (USER MEDIA) or $10 (LSI MEDIA) all products are available on the following LDOS compatible diskette types: 5 inch 80, 40 and 35 track in single or double density. 8 inch 77 track, single or double density. Although LDOS WILL support double sided diskettes LSI products will NOT be provided on double sided diskettes at this time.



Introducing a Great New Double Act

Faster, easier, more powerful programming! STSC's APL★PLUS®/80 System brings the productivity of APL to your TRS-80® Model III. Here's your escape from the restrictions and wordiness of BASIC.

With our **APL★PLUS/80 System,** you can develop and maintain programs in one-fourth to one-tenth the time needed with BASIC, because one symbol in APL often does as much as an entire statement in other languages. So you can write applications in APL that you'd hesitate to attempt in BASIC.

A Complete Application Development System

It's our language support features that put the PLUS in **APL★PLUS/80.** You get:

- complete APL, upwards compatible to our mainframe APL systems
- powerful output formatter
- array-oriented file system
- access to regular LDOS® (or TRSDOS®) files and subroutines
- communications as both a "smart" and a "simple" terminal
- traditional APL symbols *or* mnemonic keywords
- utility program libraries
- convenient access to many TRS-80 features
- complete documentation, including *APL is Easy!* (an introductory tutorial), an APL textbook, *and* a set of four detailed user's guides and reference manuals—everything to support the beginning user to the experienced APL programmer.

APL★PLUS/80 runs under LDOS 5.1 (or TRSDOS 1.3) on a 48K RAM TRS-80 Model III with two disk drives. The **APL★PLUS/80** comes with custom APL character ROM.

Join the software consultants who have already selected **APL★PLUS/80** to develop and deliver solutions. Mail in the coupon below with your payment and we'll rush you the **APL★PLUS/80 System**—all you need to run APL under LDOS (or under TRSDOS) on your TRS-80 Model III right away.

We've prepared a free information package to answer your questions about APL, **APL★PLUS/80,** and their advantages to you as a TRS-80 user. Just send in the coupon.

We're STSC, Inc., the largest supplier of APL software and services in the United States. Our APL★PLUS systems have been serving the business and professional world for more than 12 years.

Challenge to Basic

Draw a bar graph on a cleared screen with up to 13 vertical bars 5 pixels wide (separated by 5 pixels) using heights given by keyboard input.

Using APL:

```
     ∇ BARGRAPH;I;N
[1]    'ENTER UP TO 13 HEIGHTS (FROM 0 TO 48 EACH):'
[2]    I←⍳N←⍎⍞ ◇ ⎕TCFF ⍝ INPUT, CLEAR.  PLOT, LOOP:
[3]  LP:(↓N[I])∘.⎕SPOT(10×I-1)+↕5 ◇ →(0<I←I-1)⍴LP
     ∇
```

This APL program with this numeric input

```
32 6 41 25 48 2 19 0 36 48 45 12 9
```

draws the entire graph in under five seconds. How fast does your BASIC program draw the graph, and how complicated is the program?
A detailed explanation of this APL solution and a comparable BASIC solution are included in the information package.

stsc

Attn: APL★PLUS/80 Distribution
STSC, Inc., 2115 East Jefferson Street
Rockville, Maryland 20852 (301) 984-5000 (orders only)

Yes, send me the APL★PLUS/80 System, postpaid.

☐ My check for $295 is enclosed. *Add sales tax where applicable.
☐ Charge my MasterCard Account # _____
Bank # _____
☐ Charge my VISA Account # _____
Expiration date: _____
*State sale tax applicable in MD (5%), FL (4%), and CT (7½%).

I'd like to know more about the APL★PLUS/80 System.

☐ Send me your free information package.

Name _____
Address _____
_____
City _____ State _____ Zip _____
Phone (_____) _____

LDQ-782

LDOS

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

WE ARE NOW NUMBER 1

OUR MASSIVE VOLUME HAS ALLOWED US TO REDUCE OUR PRICE TO JUST

New Low Price

$129.00

REDUCED FROM 169.00

FOR THE POWERFUL LDOS 5.1 OPERATING SYSTEM.

The Ultimate In Operating Systems For Model I & III

- Model I 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 & 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 Rd.
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

DEALER INQUIRIES WELCOME. LDOS is a product of LSI. TRS-80 & Radio Shack are trademarks.

LOGICAL SYSTEMS INC.

Mequon, WI 53092
(414) 241-3066

One very important point to make here is that once the Z-80 receives an interrupt after it has executed a EI instruction, it will behave as if it executed a DI - that is so because the action of processing the interrupt automatically disables the Z-80 interrupt flip-flop until the Z-80 executes an EI instruction. The TRS-80 operates in interrupt mode one (IM1). Without going into specifics, if the INT* lead is activated under EI and IM1, the Z-80 will execute a RST 38H (RST 56) upon completing the currently executing instruction.

In the Model I, there is an interrupt latch memory mapped to location 37E0H. If you have access to an Expansion Interface schematic, you will be able to observe that the INT* lead (E/I pin 21) receives an input from both the INTRQ line of the Floppy Disk Controller and the Real Time Clock circuitry. A closer examination will show that the INTRQ lead of the FDC is also brought out to D6, which is bit 6 of the data bus (E/I pin 24). The RTC also goes to D7, which is bit 7 of the data bus (E/I pin 20) [Note, some E/I schematics erroneously depict this lead as going to D5]. The two lines are passed through gates which are closed only when memory location 37E0H is read (for example by issuing a "LD A,(37E0H)" instruction). A factory Model I, by the way, does not use bit 0 through bit 5 of the interrupt latch.

What could happen is that if either the FDC INTRQ or RTC goes active, it will activate the INT* line and a RST 56 instruction will be executed. Somewhere down the road, the task processor will take control and read the interrupt latch so it can determine what device is causing the interrupt. Before I explain what happens after the RST 56 is executed, let me cover the Model III interrupt latch.

The Model III interrupt latch is ported to PORT 0E0H. This means that to read the latch, an "IN A,(0E0H)" instruction is used. As an aside, this latch uses logic reverse of the Model I. For example, on the Model I, an interrupting device will present a one to its corresponding bit whereas the Model III will present a zero to its corresponding bit. The Model III uses the following assignments for each bit:

```
Bit 7 = Undefined
Bit 6 = RS-232 Error interrupt
Bit 5 = RS-232 Received character available
Bit 4 = RS-232 Transmit holding register empty
Bit 3 = IOBUS interrupt
Bit 2 = Real Time Clock interrupt
Bit 1 = 1500 Baud cassette falling edge
Bit 0 = 1500 Baud cassette rising edge
```

Interrupts for each device are only enabled if its corresponding bit is output to the port. A mask for this port showing what has been enabled is located at address 4213H. Note that the Floppy Disk Controller does not have its INTRQ output presented to this latch. On the Model III, the FDC INTRQ is tied to the Non-maskable interrupt (NMI) on the Z-80. The NMI cannot be suppressed. If generated, a RST 66H instruction is executed.

The task processor in LDOS is essentially the same on the Model I and the Model III. If you examine the instruction at ROM location 0038H (that's where the RST 56 goes, folks), you will see an instruction that causes a jump to address 4012H. At address 4012H, you will find another jump instruction.

The two-level jump instruction is used to provide an exit from the ROM to an address where we can alter the jump vector if we want to. Some articles appearing in the media have documented a way of interfacing to the interrupt processor by hooking into the vector at this address. I would like to discourage any interfacing in this manner because it could be detrimental to the proper operation of your LDOS. Besides, there is a much easier and more powerful way of having LDOS manage your own tasks that you will discover by reading further.

From here on in, I will refer to the task processor as "TP". Once an interrupt is detected by the Z-80, according to the discussion above, the TP gains control and the Z-80 is automatically inhibited from accepting further interrupts (we don't want our interrupt routine interrupted, do we?). Its first goal is to find out what peripheral is causing the interrupt. The interrupt latch is read and its image stored in RAM at INTIM$. A table of vectors located at RAM address INTVC$ contains a branch vector corresponding to each bit of the latch. If a corresponding latch bit is unused (or undefined), the vector entry will point to an "RET" instruction just in case a hardware glitch occurs and makes it appear as if that bit was active. The TP will scan each bit of the latch image. If it locates an active bit (at least one of them must have been active), it will save all primary registers and index register IX and then branch to the routine according to the corresponding vector in the INTVC$ table. The servicing routine ends with an "RET" instruction which comes back to the TP and restores the registers. The TP will then continue to examine the saved interrupt latch image until all eight bits have been examined. In this manner, if two or more peripherals interrupted simultaneously, they all will get serviced.

One of the hardware interrupts implemented in the TRS-80 is the Real Time Clock. This is the infamous "heartbeat" which caused such consternation in the early days of TRS-80 disk systems (boy, it seems strange to be able to talk about the early days of the TRS-80). In the Model I Expansion Interface, the RTC circuitry is an oscillator that produces a pulse precisely every 25 milliseconds - that's 40 per second. On the Model III, the RTC is synchronized to the AC line frequency and pulses at 30 pps or every 33.3 milliseconds. When the TP identifies the RTC as the source of the interrupt, a major part of the TP is put into action.

One important function is to maintain a counter with the pulsing so that clock time can be derived. Another function is to execute some background task routine. The system maintains a Task Control Block vector table at TCB$ (it is actually at 4500H on both Models but the exact address is unimportant). This table contains 12 vectors - one for each of 12 possible tasks numbered from zero through eleven. One of the first eight (0-7) is selected each time the RTC interrupts. They each take their turn in rotation while the system keeps track of which one was last executed so it can select the proper one at the next RTC interrupt. Since there are eight selections at 25/33 millisecond intervals, each of the eight are executed every 200/267 milliseconds (the numbers are Model I/Model III). These eight task slots are called low priority tasks - executing at 4-5 times per second.

TCB slots 8-11 are executed at every RTC interrupt and thus are executed every 25/33 milliseconds - which is 30 to 40 times a second. Since these tasks execute much more frequently, they are termed high priority tasks. A choice between whether a task should be placed in high or low priority is

made by analyzing the criticality of its timeliness. For instance, the TRACE function is a low priority task because it would be rather difficult to see an address value that is changing 30 to 40 times a second. As it is, there are complaints that the 4-5 per second rate is too fast. Type-ahead is a high priority task because 4-5 scans of the keyboard per second would most likely lose some of your key strokes.

Well, anyway, now that you know about this table, how do you use it? A good question deserves a good answer. LDOS contains four system entry points that manage the task vectors. These and their functions are:

```
@ADTSK = Adds a task to the TCB$ table
@RMTSK = Removes a task from the TCB$ table
@KLTSK = Removes the currently executing task
@RPTSK = Replaces the TCB address for the current task
```

I will go into a little more detail concerning the use of these system functions; however, it is interesting to note that Model I TRSDOS 2.3 contains the identical functions and a similar Task Processor. The entry addresses are the same as in Model I LDOS.

I need to make sure that the next point is completely understood since it has caused confusion to many attempting to learn how to interface to the TP. Both the INTVC$ and TCB$ contain vector pointers. The INTVC$ vectors actually point to the servicing routine; however, the TCB$ vectors POINT TO A 16-BIT LOCATION IN MEMORY WHICH CONTAINS THE VECTOR OF THE SERVICING ROUTINE. Thus, the TCB$ tasks are indirectly addressed. Make sure you keep this in mind! When you are programming an interrupt service routine, the entry point of the routine needs to be stored in memory. If we call this storage location the beginning of a Task Control Block (TCB), the reason for the indirect method of vectoring interrupt tasks will become more clear. Let's illustrate an example TCB.

```
MYTCB   DEFW    MYTASK
COUNTER DEFB    10
TEMPY   DEFS    1
MYTASK  RET
```

This is obviously an extremely useless task since all it does is return from the interrupt. However, note that I have defined a TCB location as "MYTCB" and this location contains the address of the task. I have also defined a few more data bytes immediately following the task address storage. If you think about other control blocks in LDOS (such as a Device Control Block), you will observe that control blocks are contiguous data areas that contain vectors and possibly data for a driver routine. The DCB starts with a TYPE code, followed by a vector pointing to the device driver routine, followed by data bytes - the TCB starts with the vector. We also should remember from a previous article that upon entry to a device driver, register IX is pointing to the first byte of the control block. This is also true in task processing. UPON ENTRY TO AN INTERRUPT TASK DRIVER, REGISTER "IX" WILL CONTAIN THE ADDRESS OF THE TCB. You, therefore, can address any TCB data using index instructions as in "DEC (IX+2)" which will decrement the value contained in "COUNTER".

Let's expand the routine slightly.

```
MYTCB   DEFW    MYTASK
COUNTER DEFB    10
TEMPY   DEFB    0
MYTASK  DEC     (IX+2)
        RET     NZ
        LD      (IX+2),10
        RET
```

Here we have made use of the counter. Each time the task executes, the counter is decremented. When the count reaches zero, the counter is restored to its original value. This task still is pretty worthless for its function except for its illustration of data referencing. The big question is how does this task get added to the task control block table (TCB$)? We use @ADTSK for that. Assuming we have decided that the task will be low priority, we choose an unused low-priority task slot, say slot 2. The following code will add such a task to TCB$:

```
        LD      DE,MYTASK
        LD      A,2
        CALL    @ADTSK
```

We just point register "DE" to the TCB, load the task slot number into the accumulator, then issue the system call. The task, most likely, would have been placed into high memory and protected by lowering (HIGH$). Chuck's article on RELOCATING CODE appearing in the January 1982 LDOS QUARTERLY should be consulted if you are not knowledgeable on placing routines into high memory.

Once a task has been activated, it is sometimes necessary to deactivate it. This can be done in two ways. The most often way is to use the @RMTSK system call in the following manner:

```
        LD      A,2
        CALL    @RMTSK
```

What could be more simple? We identify what task slot to remove by the value placed into the accumulator, then issue the system call. Another method can be used if we want to remove the task WHILE WE ARE EXECUTING IT. Consider the routine modified as follows:

```
MYTCB   DEFW    MYTASK
COUNTER DEFB    10
TEMPY   DEFB    0
MYTASK  DEC     (IX+2)
        RET     NZ
        CALL    @KLTSK
```

The @KLTSK system routine will remove the currently executing task. Since this task is currently executing, it is the one that gets removed from the TCB$ table. The system will not return to your routine but will continue as if you had executed an "RET" instruction. Therefore, the "CALL @KLTSK" should be the last instruction you want executed. In this example, MYTASK will decrement the counter by one on each entry to the task. When the counter

reaches zero, the task will be removed from slot 2 (remember it was placed in slot 2).

One additional TP system call is @RPTSK. The function is easy to say in words; however, its function is best illustrated. The @RPTSK function will update the TCB storage vector (the vector address in your task control block) to be the address immediately following the CALL @RPTSK instruction. This is also another case where the system will NOT return to your servicing routine after the CALL is made but rather continues on with the TP. To illustrate how this TP function is used in a program, the following example should be examined:

```
        ORG     9000H
        @ADTSK  EQU     4410H
        @RPTSK  EQU     4416H
        @RMTSK  EQU     4413H
        BEGIN   LD      DE,TCB
                LD      A,0
                CALL    @ADTSK
                JP      402DH
        TCB     DW      TASK
        COUNTER DB      5
        TASKA   CALL    @RPTSK
        TASK    LD      A,'|'
                LD      (3C3FH),A
                DEC     (IX+2)
                RET     NZ
                LD      (IX+2),5
                CALL    @RPTSK
        TASKB   LD      A,'-'
                LD      (3C3FH),A
                DEC     (IX+2)
                RET     NZ
                LD      (IX+2),5
                JR      TASKA
                END     BEGIN
```

First, I would like to point out that this task routine contains no method of relocating it to protected RAM. The statements starting at label, BEGIN, add the task to TCB$ slot zero and return to LDOS Ready. The task contains a one second down counter (Model III users would most likely use a count of four to approximate one second) and a routine to stuff video RAM (64th character of row 1) with a character. At one second intervals, the character toggles between '|' and '-'. The toggling is achieved by toggling the execution of two separate routines which perform the stuffing. Use is made of the @RPTSK TP call to implement the routine toggling. Examine this task closely to ascertain the functioning of @RPTSK.

By firmly understanding the functions of each of the four TP calls discussed, you will become proficient at integrating interrupt tasks into your applications. A final note is to be aware of the task slots already used by LDOS utilities. Consult the reference manual for details. See you next quarter and keep those cards and letters coming in, folk's.

THE JCL CORNER by Chuck

Not much has happened in the world of JCL since April's issue. No new symbols, no new macros. The change made to the 5.1.2 KI/DVR program kept the //INPUT macro from working properly with the 5.1.2 releases dated earlier than 5/25/82. The fix for this can be found in the 512 FIXES section in this issue. As stated last quarter, this month's column will be devoted to keyboard input within a JCL, and keyboard substitution of JCL lines within application programs. If you're going to be trying any of these examples, be sure to patch the KI/DVR program, if necessary.

JCL keyboard macros are //KEYIN and //INPUT, both execution macros. They function very differently. //KEYIN is used to present a pre-programmed set of choices, and then react to a numeric key response to make the selection. //INPUT is used to take any type of input at execution time, temporarily turning keyboard control back to the user, and then passing the resultant keyboard input to the program or "level" that requested it. Each macro has its own definite purpose.

When executing at the LDOS Ready level, //KEYIN menus generally will consist of a number of comment lines followed by the //KEYIN statement. For example:

```
.  1 = PAYROLL
.  2 = LEDGER
.  3 = STARTREK
.
//KEYIN - Please indicate your choice, 1 to 3
```

This sequence of lines would display the three program choices and the //KEYIN line, and then wait until a key was pressed. To evaluate the key you would use the other two macros associated with //KEYIN, the //CHARACTER and the ///. In the //CHARACTER macro, CHARACTER can be any single numeric character 0 to 9. Try adding the following lines to the above example (the comments in parentheses are for explanation only; do NOT include them in the JCL file):

```
//1           (first //CHARACTER block)
LBASIC
RUN" PAYROLL/BAS"
//STOP
//2           (second //CHARACTER block)
LBASIC
RUN"LEDGER/BAS"
//STOP
//3           (third //CHARACTER block)
LBASIC
RUN" STARTREK/BAS"
//STOP
///           (this /// marks the end of all //KEYIN response blocks)
```

OK, just what exactly have we got here. First, you can see there are three character blocks, each corresponding to one of the numeric keys you are allowing. Each block is a complete set of commands, entering LBASIC, running the program, and ending with the //STOP to return keyboard control to the

program and end JCL execution. The triple slant /// tells JCL to stop looking for a match to the key pressed, and to start executing any following lines. In other words, if a key other than 1, 2, or 3 were pressed, execution would continue with whatever commands you had after the ///.

IMPORTANT POINT: Each character block extends from the //CHARACTER line to the next //CHARACTER line, or until a /// line.

You may notice that the individual character blocks appear repetitous, with each containing an LBASIC and a //STOP line. Remember the initial display, consisting of a series of comments and the //KEYIN? If you were to go into LBASIC first (so you could remove the LBASIC command from each block), and then try to use the same lines to display the menu, you would see a "Syntax Error" message when the first comment was displayed. LDOS will tolerate comment lines starting with a period - LBASIC won't. You could remove the LBASIC command line and placed it between the menu comments and the //KEYIN macro. Doing that, however, would show the LBASIC sign on message between the menu comments and the //KEYIN line. Acceptable, but not very appealing to the eye. It would also mean that you would be at the LBASIC level if a key other than 1, 2, or 3 was pressed, so the line following the /// had better be an acceptable LBASIC command or the JCL will abort.

With this small amount of menu choices, it's possible to re-write the menu lines and do the whole //KEYIN from the LBASIC mode. Try the following:

```
LBASIC
%1F//KEYIN 1=Payroll, 2=Ledger, 3=StarTrek - Please select
//1
RUN" PAYROLL/BAS"
//2
RUN" LEDGER/BAS"
//3
RUN" STARTREK/BAS"
///
//STOP
```

Using this JCL file will enter LBASIC and then display the //KEYIN line and the three choices. The %1F will clear the screen before displaying the keyin line. Pressing one of the indicated keys will start the selected program running and then execute the //STOP after the triple slash, turning keyboard control back to the program and ending the JCL execution. Pressing a key other than 1, 2, or 3 would execute the //STOP, ending the JCL execution and displaying the LBASIC Ready prompt. At this point, the you could type in your own RUN"PROGRAM" command or any other valid LBASIC statement.

Before explaining how the //INPUT macro can be used to provide additional choices when combined with //KEYIN, let's discuss what SHOULD NOT be done when using //KEYIN. Since //KEYIN is an execution macro, it cannot be used to evaluate //IF macros or to //SET or //ASSIGN tokens at execution time. Consider the following example:

```
(Lines of menu choices . . .)
//KEYIN with your comments
//1          (first //CHARACTER block)
//SET A
```

```
//ASSIGN BIG=LARGE
//2           (second //CHARACTER block)
//SET B
//3           (third //CHARACTER block)
//IF B
//SET C
//END
///
```

This JCL example does NOT do what it appears to do on the surface! For instance, if you think pressing 1 will ignore the //SET B and the //IF statement in the other two option blocks, you're wrong! Since the //SET, //ASSIGN, and //IF macros are compilation macros, they will be evaluated during the compile phase. In this example, the compile phase will ALWAYS set the token A true, assign BIG=LARGE, set B true, and set C true. So, the important point to remember is - DO NOT USE COMPILATION MACROS INSIDE //KEYIN BLOCKS. That's not the way JCL works! You must be sure that any logical testing is done before the //KEYIN blocks start.

You can use the //INPUT macro to provide an additional run time option in case one of the //KEYIN choices was not selected. Of course, this depends on the construction of the //CHARACTER and /// blocks. The LDOS manual shows how to use //INPUT and //KEYIN from the LDOS level to provide an alternative in case no valid //KEYIN selection is made. Refer to your manual for that type of example. If you want to allow an additional input from within the JCL environment in LBASIC, you can use the following format. The important point to note is the use of the //STOP macro within the individual //CHARACTER blocks:

```
LBASIC
%1F//KEYIN    (your menu choices)
//1
RUN"PROGRAM1"
//STOP
//2
RUN"PROGRAM2"
//STOP
//3
RUN"PROGRAM3"
//STOP
///
//INPUT  Enter your own command as RUN"PROGRAM"
//STOP
```

From the top - the screen clears, the //KEYIN shows the menu choices and then waits for a keyboard input from the user. If 1, 2, or 3 is pressed, the corresponding program is run and the following //STOP halts JCL execution, returning keyboard control to the user. HOWEVER . . . if any other key is pressed, the //INPUT line will be displayed. This will allow the user to type in a RUN"PROGRAM", or any other valid LBASIC statement, executing the command and then the //STOP, giving keyboard control back to the user. If, for example, the user pressed <ENTER> in response to the //INPUT, the LBASIC Ready prompt would appear and the JCL execution would halt.

That about wraps up the basic uses of //KEYIN The important points to remember are:

1) //KEYIN can only respond to the numbers 0 to 9.
2) DON'T use compilation macros inside //KEYIN blocks.
3) Structure your //KEYIN so you exit at the proper level if no choices are selected.

One use of the //INPUT macro was demonstrated in the previous //KEYIN example, that being to input an LBASIC command. The same method would be equally valid if the //KEYIN exited at the LDOS level (as demonstrated in the JCL section of the LDOS manual). The main use for //INPUT is to allow the computer operator to input a command or response that may vary at runtime. The most important point to remember is that the operator's response MUST be valid at whatever level the JCL is at. Keeping the previous menu selection theme going, you could use the following as an AUTO DO file to select certain /CMD files from the LDOS level every time you powered up:

```
%1F. This disk contains the following program files -
. Be SURE the proper data disks are installed, then select.
.
. *=* PROFILE *:* VC *:* LSCRIPT *:* EDAS *=*
.
//INPUT     Please type in your choice, or <BREAK> to abort
//STOP
```

This example would clear the screen (the %1F), then display all the comment lines and the //INPUT line. At this point, JCL execution would stop, waiting for the operator to input a command line. Since execution paused at the LDOS level, entering any of the listed programs, or any other program or command that could execute directly from the LDOS level is perfectly acceptable. You could even run an LBASIC program by entering a command line such as LBASIC RUN"PROGRAM". See any pitfalls in this example? No? Look Again. HINT: think about what the //STOP would do if an LDOS library command were entered.

Entering any program name at the //INPUT would execute the program, halt JCL execution, and relinquish keyboard control to the program, exactly as desired. Entering an LDOS library command would execute the command, halt the JCL, and relinquish keyboard control to . . . . Well, you would actually be at the LDOS Ready level, except that the //STOP would keep the "LDOS Ready" prompt from being displayed!! It may appear as if the system had hung up. This is similar to the way the MiniDOS filter returns when executed from the LDOS level, with the cursor displayed but no visible LDOS Ready prompt.

This example shows how //INPUT works, and demonstrates how the //STOP can produce different (unexpected, abnormal, undesired, whatever) results if the JCL is supposed to //EXIT rather than //STOP in response to an //INPUT. It also shows that this type of menu function could probably be handled better by the //KEYIN macro. To demonstrate the real power of //INPUT, we'll have to get into using pre-arranged keyboard responses in a JCL file. The following will try to be practical, using BASIC programs as examples.

As is stated in the LDOS manual, JCL works by "taking control of the keyboard". Actually, this should read "taking control of any LINE request from the keyboard". With the TRS-80, there are two visible types of "language" inputs, those from BASIC (LBASIC) and those from an assembly language program. To allow a JCL file to provide an input, use the following rules:

1) From LBASIC, use either the INPUT or the LINEINPUT statement. INKEY$ will NOT pull in characters from a JCL file.

2) In assembly language, use the @KEYIN call (X'40') rather than the single key (X'2B' or X'49') calls to input a keyboard response.

OK. The subject is "How to control programs totally with JCL". Starting with LBASIC programs, use rule #1. Use the LBASIC "LIST" command to determine where keyboard entries are requested, and which statement(s) are used for the input. A good rule of thumb is that any program that shows a flashing cursor and a field length marked by periods or graphics characters is using the INKEY$ statement to handle keyboard input. If the keyboard requests show this flashing cursor or field display, you can generally assume that all inputs will be taken through the same subroutine. Try and isolate the GOSUB statement that does this. Once you find the GOSUB line number, you can edit that program line to be an INPUT or LINEINPUT statement, thereby allowing the answer to any prompt to come from a JCL file. If there are certain prompts that you wish to FORCE the user to answer at runtime, don't worry! This is where the //INPUT macro really comes into its own.

In this following example, you should assume that you want to give a data file name, and then stack the rest of the responses to any future prompts in the JCL file.

```
LBASIC
RUN"Report12/bas"
//INPUT
15
JUNE
FIRST
//EXIT
```

Now, let's assume that the first input request in the program "Reportl2/bas" was a LINEINPUT statement to request a filename:

```
100 LINEINPUT "Enter the name and drive of the data file ";D$
110 OPEN"R",1,D$,128
```

When the program runs, the line 100 LINEINPUT message will be displayed, and the JCL processor will attempt to feed in the next line from the JCL file. Since the next line is an //INPUT, the keyboard will now become "alive", allowing the operator to type in the filespec of the data file to be used by the program. This filespec will be assigned to the variable D$ in line 100, and then be used as the filename and drive number in the OPEN statement in line 110. Assuming that there are three more prompts in the program, the 15, JUNE, and FIRST lines in the JCL file will be used to answer them.

Although this example used only one //INPUT, it's allowable to use as many as you want anywhere in the program. Just be sure that you keep the order of the program's input request in mind when creating the JCL file of this type. Believe me, there's nothing more frustrating than typing in a large JCL file with pre-arranged keyboard responses, only to discover that a program input request was missed somewhere along the way. This can definitely mess up data files, so be sure to test any new JCL with backup copies of the data!

The same procedure described above can be used when running an assembly language program with a JCL file. Again, the method of input will have to be the line input handler at X'40' (hereafter referred to as @KEYIN). This precludes using a JCL file to control such programs as Scripsit and Visicalc, as these programs all use a single key input routine. As last issue's column mentioned, the CMDFILE utility does use the @KEYIN call for all inputs. If you assembly language programmers are repetitively using CMDFILE, you can make a run through the procedure, writing down the necessary responses to the prompts. These responses can then be typed into a JCL file, using the //INPUT macro whenever an unpredictable answer may be needed.

HEY! I heard those moans and groans from some of you when you discovered that Scripsit, etc., can't be controlled with a JCL file. In fact, I've discussed this on the phone with a handful of users who have called about this. How would y'all like a patch to Scripsit and Visicalc that would let you run them from a JCL file? Sorry, but nothing like that exists. However...

The KSM filter can be put to good use to control these types of programs. Although not a JCL function, I'll mention it here as it is, in most cases, a very acceptable substitute (this is assuming that you are using the LSCRIPT fix to Scripsit). To start the explanation, assume that you have three Scripsit files to be loaded and printed, called TEXT1/SCR, TEXT2/SCR, and TEXT3/SCR (how original). As long as you have Scripsit, use it to build the following KSM file line:

```
#L TEXT1;#P;#L TEXT2;#P;#L TEXT3;#P;
```

Save this file using the ASCII option of Scripsit, giving it a /KSM extension. Now comes the slightly tricky part. You must, in some manner, change every "#" to the character X'1D', necessary to get you into the SPECIAL COMMAND? mode of LSCRIPT. If you have LED, you could use the HEX mode to insert this character directly as you were creating the file. If you have FED, you can use the H modify option to change the # to the proper character. If you have neither, you can use the "LIST filespec (H)" command to list the file to the video, noting the location of every #. Then the PATCH utility can be used to change the # characters to the necessary X'1D' value. With this method, you can create command lines up to 255 characters in length to control Scripsit, and other assembly language programs, with a single keystroke and the KSM filter. In this example, assume that you have built the KSM file and assigned the line to the "A" key. Enter LSCRIPT, and then press a <CLEAR><A>. The three files will now be automatically loaded and printed, with no further need for operator intervention.

LES INFORMATION

HELLO - from Les Mikesell

Frequenters of the MicroNet bulletin boards will know that I have long been an LDOS fanatic (yes, I even liked VTOS, if anyone can remember that far back), so perhaps it was inevitable that I would migrate to Mequon to become the newest member of the LDOS support and development team. You will be seeing more from me on these pages in the future in the Les Information column. The name is just in keeping with the general spirit of the computer business; actually there may be some facts included from time to time. Since my main involvement with the TRS-80 so far has been the Modem80 communication program (see What's New in this issue), the first column will deal with some of the basics of data communication.

First, assuming that the remote computer is more than 50 feet away, you will need a modem to allow transmission via the phone lines. Most modems require that the TRS-80 be equipped with the RS232 interface, although a few connect directly to the TRS-80 expansion buss. For use with a Model 1, the buss-connecting types may be a good choice, but the Model 3 RS232 interface has proven to be reliable and has the advantage of being able to generate an interrupt to the CPU when a character is received allowing more dependable operation under LDOS. There are acoustically-coupled modems with rubber cups in which to place a telephone handset and direct-connect modems that plug into the phone line. In the latter class, there are also auto-answer modems that will operate unattended. The newest innovation is to build a microprocessor into the modem itself to allow features such as your choice of touch tone or pulse dialing controlled by character strings transmitted from the computer. Virtually all of the common 300 baud modems use the standard Bell 103 tones and can communicate with each other, so the choice can simply be based on cost versus the desired features. Special programs may be required to activate some of the non-standard functions, though, so you should be sure that what you need is available unless you write your own programs. The standard LDOS drivers will operate the Radio Shack RS232 interface and systems that are "software compatible", and the serial ports of the LOBO LX80. The system of "logical devices" will allow anyone familiar with assembly language to write a driver for non-standard equipment.

Now, just dial the phone and..... well, next you need someone at the other end. Most large cities have one or more bulletin board type computer systems which can be used free of charge for exchanging messages (and sometimes public domain programs). It is difficult to keep an up-to-date directory of available systems, but one publication that attempts to do so is The On-Line Computer Telephone Directory (OLCTD), published quarterly by Jim Cambron, P.O. Box 10005, Kansas City, MO 64111. A year's subscription is $9.95, or single copies are available for $2.85. Most local systems maintain a list of numbers for other bulletin boards currently active, so finding the first one is the hardest part.

You may also want to explore the Compuserve Information service, an enormous data base and message system with local phone access in many cities. Information on accessing this system is available from: Compuserve, Information Service Division, 5000 Arlington Centre Blvd., Columbus, OH 43220, or by phone at 614-457-8600 (in Ohio) or 800-848-8990 (outside Ohio).

Of special interest to LDOS users is the bulletin board maintained by the LDOS support group where users can discuss new applications and get answers to technical questions. Access is limited to those who have sent in their LDOS registration form with their Compuserve user number listed. If you get a Compuserve number after you register, call or write with the number and you will be logged as a member on our bulletin board.

There are two main "modes" of operation on Compuserve. The information service is generally accessed through menu selections, which is slow but fairly easy for new user. This mode can be distinguished by the exclamation point prompt for each entry. The data base entries accessable from the menus have unique page numbers, and can also be reached by typing GO nn (where nn is the desired page number) at any exclamation point prompt, speeding things up after you become familiar with the system. A small magazine is published for subscribers to the service with an index to the available page numbers, although the index is also on-line and constantly being updated. For the more adventurous, there is also the MicroNet area, distinguished by the OK prompt. In this mode, you are actually operating the DEC PDP-10 computer and can write and run your own programs using any of several available editors and languages. Each user has his own 128K filespace which can be used for uploading, downloading or exchanging files with other users. The commands for the MicroNet area are more complex and generally require a MicroNet users guide. The MicroNet area can be reached through the menus or by typing GO PCS 71 at any of the exclamation point prompts. To get back to the information service, type R DISPLA at the OK prompt. The DEFALT program can be used to permanently change the area you enter when you log on to the system, as well as your terminal configuration.

The Public Access area is a fairly recent addition to the service that allows downloading of user-contributed material. This is about a 9 megabyte area, so be prepared to spend a long time just reading the current directory. It may be reached through the menus, or from MicroNet by entering R ACCESS. Like most of the other system programs, there is a built in HELP command to assist new users. There are some files relating to LDOS under my user number <70010,266> in this area that are accessable to any Compuserve user.

USER CONTRIBUTED PROGRAMS

LDOS, beginning with release 5.1.2, has LBASIC assigning a default extension of /BAS to every file SAVE or LOAD or RUN. This is in full accordance with the standard adopted with TRSDOS for command files from DOS Ready, but is a bit foreign to LBASIC users and users of other systems. At first I didn't think I'd like it this way, but after a couple of hours I decided that it was the only way to go. I now love it!.

The problem is that I had a lot of files which needed renaming. My previous practice was to name BASIC programs without an extension and put one on most other files. This works nicely, but I wanted to take full advantage of the default extensions within LDOS so I wrote the following utility to help rename the large number of files that did not have an extension. Perhaps you can use it too.

The program is short, so it should present little difficulty in keying it in. One thing to watch is that there is no error trapping. Be sure that you have the write protect tab off the disk before putting it in the drive or you'll get an error.

The program will ask for an extension to be applied to all files. Then the directory is read and you are asked if you wish to rename each file. If you answer "Y", the file is immediately renamed, if you answer "N", the file is left alone. When the program has finished, you are offered an opportunity to run it again or exit to LBASIC or to LDOS Ready.

I have about four hundred disks with BASIC or parts of BASIC programs on each side of them, so this program has proven to be a real help to me. I hope you will find it equally useful.

```
20 REM ******************************************************
40 REM ****   LDOS FILE RENAMING UTILITY (C) 1982 by      ****
60 REM ****   Charles P. Knight  2708 Roberts Cir.        ****
80 REM ****   Arlington, Texas 76010  (817) 640-4452      ****
100 REM******************************************************
120 CLEAR10000
140 A$="FILE --> ":B$=" rename it to: ":DIMF$(128)
160 CLS:PRINT@512,"Which drive to rename files on";
180 INPUTTT$
200 D=VAL (TT$):IFD<0ORD>7THEN160
220 D$=" "+RIGHT$(STR$(D),1)
240 CLS:PRINT:PRINT:PRINT:PRINTTAB(18)"Select your extension:" PRINT
260 PRINTSTRING$(63,140):PRINTTAB(9)" 1. /BAS                2. /ASM
280 PRINTTAB(9)" 3. /TXT                4. /SCR
300 PRINTTAB(9)" 5. /CMD                6. /DAT
320 PRINTTAB(9)" 7. /REL                8. /CHN
340 PRINTTAB(9)" 9. /MAC                0. another choice
360 PRINTSTRING$(63,140):PRINT"Files are on drive"D
380 INPUT"Enter your choice";EX
400 IFEX=0THENINPUT"Enter your three letter extension";EX$:ELSE440
420 IFLEFT$(EX$,1)<>"/"THENEX$="/"+EX$:IFLEN(EX$)>4ORLEN(EX$)<1THEN240
440 IFEX>0THENONEXGOTO460,480,500,520,540,560,580,600,620ELSE660
460 EX$="/BAS":GOTO660
480 EX$="/ASM":GOTO660
500 EX$="/TXT":GOTO660
520 EX$="/SCR":GOTO660
540 EX$="/CMD":GOTO660
560 EX$="/DAT":GOTO660
580 EX$="/REL":GOTO660
600 EX$="/CHN":GOTO660
620 EX$="/MAC":GOTO660
640 GOTO240
660 X$="DIR/SYS.RS0LT0FF"+D$
680 OPEN"R",1,X$,32
700 FIELD1,1ASF1$,4ASD1$,8ASNA$,3ASE$,16ASD2$
720 Y=0
740 FORX=17TOLOF(1)
760 GET1,X
780 IFASC(F1$)AND128THEN960'IT'S FXDE
```

```
800 IFASC(F1$)AND64THEN960'IT'S SYSTEM FILE
820 IFNOTASC(F1$)AND16THEN960'IT'S NOT IN USE
840 IFASC(F1$)AND8THEN960'IT'S NOT VISIBLE
860 N$=NA$
880 IFRIGHT$(N$,1)=" "THENN$=LEFT$(N$,LEN(N$)-1):GOTO880
900 N1$=E$:IFE$<>"   "THEN960'ALREADY HAS EXT
920 F$(Y)=N$
940 Y=Y+1
960 NEXTX
980 CLOSE
1000 CLS:IFY>0THENFORX=0TOY-1ELSE1140
1020 PRINT@448,STRING$(63,140)
1040 PRINT@512,A$;F$(X);B$;F$(X);EX$;STRING$(8,32);
l060 PRINT@576,STRING$(63,140)
1080 IK$=INKEY$:IFIK$=""THEN1080
1100 IFIK$="Y"ORIK$="y"THENGOSUB1160
l120 NEXTX
1140 CLS:IFY>0THENPRINT@448,"***** Function Completed *****";:GOTO1260ELSEPR
INT@448,"**** No files to be renamed ****";:GOTO1260
l160 CM$="RENAME "+F$(X)+" "+EX$
1180 PRINT@960,CM$;STRING$(15,32);
1200 CMD CM$
1220 PRINT@960,STRING$(63,32);
1240 RETURN
1260 PRINT:PRINT:PRINT"1. Return to LDOS.  2. Exit to LBASIC   3. Run again"
1280 PRINT:INPUT"What shall it be";C$
1300 C=VAL(C$):IFC<1 OR C>3THEN1140
1320 IFC=1THENCMD"S"ELSEIFC=2THENSTOPELSERUN
```

FIXES FOR MODEL III VISICALC

```
.Patches to Model 3 Enhanced VisiCalc to run on
.LDOS 5.1.2. By Ray Pelzer, partly based on the
.Model 1/3 patches (c) 1981 by Logical Systems, Inc.
.These patches as of 05/15/82  (1st correction)
.These patches will make VisiCalc 3 run on a Model 1 or 3.
.WARNING! for Visicalc Version # VC-150Y0-T83 *ONLY*!!
.Patch in 002B kybd scan for typeahead, etc.
.X'557F'=CD 2B 00
.
. All the stuff for proper HIGH$ and Mod 1/3 conversion.
.X'521A'=CD 86 55
.X'52A6'=CD 86 55
.X'5586'=3A 25 01 FE 49 3E 00 2A 49 40 C0 21 90 42 22 A8 55
.X'5597'=21 64 4B 22 4A A7 21 93 42 22 69 A7 2A 11 44 C9
.
.Part of the @CKDRV call to see if a drive is active.
.X'55A7'=CD B8 44 C8 CD 51 A4 E1 C3 3F A7
.
.Add the ever-popular "/X-" screen-refresh command.
.X'7062'=5E 52
```

```
.Change the /V version display
.X'9235'=28 63 29 20 31 39 37 39 2C 38 31 20 56 69 73 69
.X'9245'=43 6F 72 70 20 56 43 2D 31 35 30 59 30 2D 54 38
.X'9255'=33 20 20 4C 44 4F 53 20 62 79 20 52 61 79 20 50
.X'9265'=65 6C 7A 65 72
.
.Now, make the changes to scan filespecs properly in LDOS
.When using a /S command withOUT a explicit filespec.
.X'A43D'=F5 3A 40 38 E6 04 28 06 CD 2D A4 F1 37 C9 CD 35 A4
.X'A44E'=F1 B7 C9 79 F5 78 4F F1 47 C9
.X'A720'=22 88 B5 CD 35 A4 3A 8F B5 FE FF 47 20 02 06 00
.X'A730'=CD 43 A7 D0 CD 3D A4 38 06 04 78 FE 08 38 F1 3E 03
.X'A741'=37 C9 CD 51 A4 CD A7 55 CD 65 4B 1E 01 21 D6 B5
.X'A751'=CD 45 4B CD 51 A4 20 55 21 D6 B5 7E E5 B7 C5
.X'A760'=28 3A 11 D6 B6 CD 51 A4 CD BB 44 EB 7E B7 28 3E
.X'A770'=7E FE 3A 28 39 FE 2F 23 20 F6 ED 5B 8A B5
.X'A77E'=06 03 1A FE 20 20 05 7E FE 3A 18 07 BE 20 21 13 23
.X'A78F'=10 EF 28 0C 3A 91 B5 B7 28 03 AF 18 02 3E 01 B7
.X'A79F'=20 0B CD 17 A8 38 03 C1 E1 C9 CD D8 A7 18 03
.X'A7AE'=CD C8 A7 C1 0C E1 23 CA 3F A7 18 A2
.end of patch (42 LINES)
```

512 FIXES

For Model I with file dates earlier that 5/25/82, apply the following patch to KI/DVR.

```
D02,C0=00
D02,C8=28 08 79 B7 28 6E 00 00 00 00
. EOP
```

For Model III with file dates earlier than 5/25/82, apply the following patch to KI/DVR.

```
D02,C3=00
D02,CB=28 08 79 B7 28 6E 00 00 00 00
. EOP
```

For BOTH Model I and Model III with files dated earlier than 5/25/82, apply the following patch to KSM/FLT.

```
D00,60=2A 8E 55
. EOP
```

About 100 Model I users had a bad KSM/FLT file put on their disk when it was updated to the 5/25 dated version. This will be evidenced by the message "Load file format error" when trying to establish a KSM file. If this is the case, apply the following patch to KSM/FLT.

```
D00,00=05 06
```

LATE BREAKING NEWS AND OTHER RANDOM ITEMS

To try and clear up any confusion about the new Model I RDUBL driver for the Radio Shack doubler, it should be noted that it requires the 5.1.3 update. This driver is not available for 5.1.2. All of our testing has been done on 5.1.3, and we are certain that the new driver works fine with this version of LDOS.

For those of you using DEBUG to work on program debugging, the following patch will force the display to initialize in the ASCII mode rather than in the HEX mode. This could be very useful for debugging programs that do alot of character string manipulation.

```
.SYS5 patch for ASCII display mode.
X'5147'=68
X'5148'=28
X'515C'=B7
X'515D'=CC
.EOP
```

We get calls with people having problems such as "I install PDUBL and the system locks up", or "I set the KI/DVR and the keyboard locks up". The common factor between PDUBL and KI/DVR, as well as any other driver or filter, is the area of memory that they use. Most of these calls come from Model I users. High memory (the top 32K) on the Model I is in the expansion interface. Access to this memory is made across a cable that has two weak points - the edgecards on the keyboard unit and on the EI. Also, the memory chips in the EI must be good quality chips. To see if you have memory problems when something like the above occurs, try the following. Use the MEMORY (HIGH=) command to set memory to X'BFFF' or X'7FFF'. This will lock out the top 16K or 32K, respectively. Then set the driver or filter that is causing problems. If the problem goes away, its time to clean the connectors or change the memory.

E X T E N D E D S U P P O R T A G R E E M E N T

This agreement is to provide additional support services to users of LDOS that have a desire for these services.

The following services will be provided to any registered LDOS owner for the sum of Twenty-five dollars ($25.00) paid to Logical Systems Inc. To execute this agreement you must be a REGISTERED LDOS OWNER, fill out this form and return it to LSI at 11520 N. Port Washington Rd. Mequon, Wis. 53092, along with a check for 0 The fee is $35.00 for all foreign users, including Canada and Mexico.

For ONE year from the date of this agreement LSI will provide the following services:

1> The users may send in his MASTER LDOS disk for upgrading at a fee of $5 instead of the $10 charged to owners not on this extended service agreement.

2> You will receive 4 issues (one year) of the LDOS QUARTERLY.

3> If you are a Compuserve subscriber, you will be given membership in the LDOS users group on MicroNET. (The LDOS bulletin board.)

4> Foriegn owners will have newsletters sent and updates returned via AIR MAIL.

Name ..... LDOS Serial # .....

Address .....

City ..... State ..... Zipcode .....

Country ..... Phone (.....) .....

Model I ..... Model III ..... Other .....

Enclosed $25 ..... $35 ..... MicroNET I.D .....

Charge my credit card account # .....

Mastercard ... VISA ... Expiration Date .../... Bank # .....

---------- ----------
Signature Date

ULTIMATE UPGRADES.

Buy direct from the manufacturer and save on high performance disk systems and other enhancements for Model I, II, and III.

Whether your TRS-80 is a Model I, II, or III, you've probably wished for more disk capacity. Now Lobo gives you that—and much more—at low, manufacturer-direct prices. With uncompromising quality, *and the protection of Lobo's unique 1-year warranty.*

Special for Model I owners: the LX-80 Expansion Interface

Radio Shack may have forgotten you, but Lobo hasn't! Our LX-80 expansion interface (plus LDOS operating system) gives your Model I more features and more expandability than a Model II or III. The sturdy steel enclosure fits under your monitor and adds:

- 32k additional RAM
- Interfaces for standard Radio Shack mini-floppy drives and Lobo high-performance disk systems
- Centronics-type printer port plus screen printer port
- Two RS-232C serial ports (optional)
- Plus a real-time clock, sockets for custom ROM, and a heavy-duty power supply for your keyboard unit

Discover the real power and potential of your Model I, with the bargain-priced Lobo LX-80!

LX-80 with LDOS operating system (required) $510.00
LX-80 without LDOS (for current LDOS users) $460.00
Dual RS-232C serial port option $100.00

LDOS: the ultimate TRSDOS-compatible operating system

One of the few software products ever to receive a perfect box score from *Infoworld* magazine. The reviewer said: "LDOS 5.1 is awesome! . . . It performs nearly perfectly . . . a straightforward and simple system to use . . . the best manual for software I've ever seen or reviewed, bar none . . . This DOS takes the TRS-80 from the hobby category and endows it with features that many a so-called business system does not have . . . LDOS offers unparalleled versatility and function."

LDOS includes a powerful extended disk BASIC, smart terminal emulator, and many other useful utilities that make it worth far more than its low price. It runs on any Model I or Model III with at least one disk drive.

LDOS operating system (specify Model I or Model III) $129.00

Add-on 8" floppies for Model II

Why pay Radio Shack prices to expand your Model II's disk capacity? The Lobo 8202C2 adds two 8" double-density floppy drives, for a total of 1.1 megabytes of additional storage. Installation and operation are identical, and you get the added benefit of Lobo's 1-year parts and labor warranty. **8202C2 dual-drive 8" floppy system for Model II $1269.00**

8" floppy systems for Model I and Model III

These rugged dual-drive systems attach to any Model I with LX-80 expansion interface, or any Model III, and add the mass storage you need for the big jobs. Double density recording stores 535kB on one side of the disk. Using the LDOS operating system (required) you get full compatibility with standard TRSDOS plus greatly increased capabilities.

8202C3 two single-sided drives (1.1 MB total) for Model III $1625.00
8202CX same as above, for Model I with LX-80 (sold separately) $1249.00
5202C3 two double-sided drives (2.2 MB total) for Model III $2025.00
5202CX same as above, for Model I with LX-80 (sold separately) $1749.00

Add-on minifloppy drives for Model I

Completely compatible with all Model I hardware and software, but with an extra 5 tracks for data storage. Requires a Model I with either the Radio Shack expansion interface or the Lobo LX-80 (see left).

4401C Add-on 5¼" drive for Model I $305.00

High-capacity minifloppy for LX-80

An economical way to get a big storage boost for your LX-80-equipped Model I. The double-sided, 96 track/inch drive stores 720 kB, and eliminates most tedious disk swapping.

Model 4801C high-capacity 5¼" drive for LX-80 $465.00

5¼" Winchester System

Compact and exceptionally reliable, with 4.8 megabytes of high-speed Winchester storage plus a 720 kilobyte floppy drive. The value leader in mass storage.

950T for Model III or Model I with LX-80 (sold separately) $3633.00

8" Winchester System

Over 9 million bytes of storage accessible in milliseconds: 8.2 MB on an 8" Winchester drive and another 1.1 MB on the backup floppy drive. Unsurpassed for maintaining very large data bases.

1850T for Model III or Model I with LX-80 (sold separately) $4459.00

Winchester disk systems for Model I and Model III

The ultimate mass storage devices! Enormous capacity and impressive speed give your system a dramatic performance boost. Add the impressive file-handling capabilities of LDOS (included), and you can outperform systems costing far more. IMPORTANT: Many Winchester disks now being sold have no provision for file backup. Lobo systems include a built-in high-density floppy drive that can store the entire contents of the hard disk on just 6 or 7 floppies. This backup drive is also usable for additional on-line storage of programs and data.

Ordering Information

All prices include shipping and handling. California residents add 6% sales tax. Credit card orders shipped within 24 hours. Personal checks require 2-3 weeks for clearance before shipment.

The Lobo Warranty

All Lobo hardware products carry a limited 1-year parts and labor warranty. Call or write for complete warranty statement.

TOLL-FREE ORDER NUMBERS:
U.S. (except California):
800-235-1245
In California: **800-322-6103 or 800-322-6104** Hours: 7AM–5PM Pacific Time

Write for free catalog:

LOBO drives INTERNATIONAL

Lobo Drives International
Dept. LQ7
358 S. Fairview Ave.
Goleta, CA 93117