

April 1984

Volume 2 Number 6

# JOURNAL

ISSN 0737-9161

\$4.

Formerly the  
LDS  
QUARTERLY

**EMERGENCY  
COVER**

(Originally Scheduled  
to Appear in the April  
Issue of Basic Computing.)

We Now Have A Toll Free Orderline!

(800) 248-3535

LOGICAL  
SYSTEMS  
INC.  
□-□-□-□

**NEW!**

**OUTLINE**    **CAMEO**    Pretorian  
**Rotunda**    LOMBARDIAN    celtic  
**Nostalgia**    INCISED TRAJAN    Playbill  
 LIQUID CRYSTAL    Old English    **CHAINED**  
**MOON LITE**    CLIMBING    **ANTIQUÉ**  
**BANNER**    Pump    **REV BIAN**  
**SHADOW**    Elegant    Mini Cubes  
 Chancery Medium    ROMAN    Small Boldface  
    **JULY 4**    Small Bold Italics

These are printed by DOTWRITER just as you see them.

## See What You Can Do With DOTWRITER 4.0!

Now available for the Model 4, too!

**T**his new, fast version of DOTWRITER is just what you need to turn your dot-matrix printer into a versatile typesetting machine. Written entirely in "machine language," our latest release offers even more features to help you produce beautiful, eye-catching results.

### What Is DOTWRITER?

DOTWRITER is a full-function text printing program. It lets you print distinctive letterheads, brochures, flyers, catalogs, invitations, or even a book. It does superb right-justified proportional printing, including "kerning" (tucking small letters under big ones to achieve a really professional result). DOTWRITER handles type sizes from 1/8 to 1 inch, can magnify text until each letter fills the page, intermixes type styles, and even does reversals (white on black).

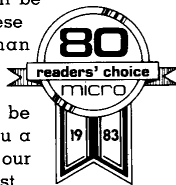
After writing your text with any popular TRS-80 Word Processor, such as NEWSSCRIPT, just insert the necessary layout commands, save it to disk, and DOTWRITER will do the rest.

### What's Included?

DOTWRITER includes the printing program and fourteen complete sets of type faces (60 to 90 characters in each set). The 60-page manual has a step-by-step tutorial, a Table of Contents, and an Index. And, of course, you will have on-going support directly from PROSOFT.

### DOTWRITER Can Grow With You

15 "Font disks," each with 3-12 complete typeface sets (60 to 95 characters in each set), are available separately, and three more will be released soon. These disks cost less than \$25.00 each, and may be purchased at any time. We'll be happy to send you a free sample of all our typefaces on request.



### Design Your Own Typefaces

If you just want to use some of the many typefaces we carry, then DOTWRITER is all you need. If you want to be able to modify our typefaces or even design new ones, then you will also want to order the "Letterset Design System." We offer it at a reduced price when you order it along with DOTWRITER.

### Supports Models I, III, and 4

One version of DOTWRITER 4.0 is for the TRS-80 Models I and III (also LNW and MAX-80), and another is for the TRS-80 Model 4 (yes, in native Model 4 mode). At least 48K and two disk drives are needed.

The Letterset Design System works only on Models I and III, but it can run on a Model 4 in Model III mode.

Versions are available for the Epson MX-80 with Graftrax, MX-100 with Graftrax-Plus, RX-80, and FX-80; the C. ITOH 8510/1550; the Microline 84/92/93; and Radio Shack's DMP series 200/400/500, and 2100. Please specify your printer and computer when ordering!

### How to Order

Limited ad space allows us to show you only a few of the 120 DOTWRITER fonts, but free print samples are available on request. If you want the best in graphics printing, we suggest you order DOTWRITER today, toll-free.

DOTWRITER 4.0 (Models I, III)	\$79.95
DOTWRITER 4.0 (Model 4)	99.95
Letterset Design System (LDS)	39.95
Special:	
DOTWRITER and LDS for Models I, III for Model 4 (Model 3 LDS)	99.95
Additional Fonts (4-12 per disk)	17.95 and 24.95

ORDER NOW, TOLL-FREE  
(800) 824-7888, oper. 577

**PROSOFT**

Dept. C, Box 560, No. Hollywood, CA 91603  
(818) 764-3131 Information and Same-Day Processing  
TERMS: VISA, MC, checks, COD. Please add \$3.00 shipping in U.S. or Canada, \$15.00 overseas. sales tax in Ca. Most orders filled within one day.

# CONTENTS

**INTRODUCTION FROM LSI:**

LATE BREAKING NEWS ..... Page 2

NEW PRODUCTS ANNOUNCEMENTS ..... Page 3

ITEMS OF GENERAL INTEREST ..... Page 4  
 patches and update information

Profile III Plus and LDOS (revisited) ..... Page 7

**LSI Journal/BASIC Computing Insert:**

View from the Bottom Floor ..... Page 9

Locating high memory routines under LDOS/TRSDOS 6.X ..... Page 11

REGULAR USER COLUMNS:

\*\*\* PARITY = ODD \*\*\* - Tim Daneliuk ..... Page 12

'C' What's Happening (Part 6) - Earl Terwilliger ..... Page 14

FROM THE LDOS SUPPORT STAFF:

LDOS: HOW IT WORKS - an introduction to COMM and LCOMM ..... Page 16

Les Information - Fast Machine Language Disk 1/0 ..... Page 17

LSI Journal and LDOS Quarterly Index - by Scott Loomer ..... Page 22  
 (Please note that due to the lead times involved,  
 the index in Vol2, #5 is actually more complete.)

Copyright © 1983/1984 by Logical Systems, Incorporated  
 8979 N. 55th Street P.O. Box 23956  
 Milwaukee, Wisconsin 53223

Main Switchboard and Customer Service: (414) 355-5454  
 Toll Free Order Number (899) 248-3535

## LATE BREAKING INFORMATION

By Bill Schroeder

Yes ... this is the issue of the LSI Journal that was to have appeared in the pages of the magazine BASIC Computing. Alas, it seems that this will never come to pass.

About the first week of March, I was informed that there would never be an April issue of BASIC Computing. Citing ongoing and growing losses, Irv Schmidt informed me that BASIC Computing was going to cease publication. At this writing, they have stated that they are making arrangements to have another publication fulfill their subscription obligations as they are financially unable to do so. If you have any questions about your subscription, contact BASIC Computing at 3838 S. Warner St., Tacoma, Washington, 98409 - (206) 475-2219.

On March 9th, 1984, the material that was to have been published in the April issue of BASIC Computing was returned to us. We did a mad scramble to put the issue together ourselves, and this is the result. Most of the material is presented as it would have appeared in BASIC Computing. We are sending this issue to all registered LDOS owners to explain the situation. We sincerely apologize for this development. Please understand that we had NO KNOWLEDGE WHATSOEVER of the upcoming demise of BASIC Computing.

## LSI ANNOUNCES TOLL FREE ORDER SERVICE

(899) 248-3535

For your convenience, LSI has installed Toll-Free phone service to our order desk. Now you can place orders with LSI "on our nickel". The number will be (800) 248-3535 and will go into service about March 30th, 1984. We hope you will enjoy the convenience of this new service. Please remember that this number is for orders only. Customer Service calls will continue to be taken at the main switchboard number of (414) 355-5454.

## LSI "Such a Deal" Update

Here is the current status of the Deals offered in the last Journal:

### **DEAL #1:**

For those of you that did not receive the previous issue of the Journal, LSI was closing out all products not manufactured by LSI. All of the non-LSI software is gone, except for MicroPro's MailMerge, Captain 747 from Molimerx and the IJG book "How to do it on the TRS-80", by William Barden.

For those who own the LDOS version of WordStar, now is the time to pick up a copy of MailMerge. For details on the capabilities of MailMerge, see your WordStar manual (the MailMerge documentation is included in it). There are less than twenty copies left, so place your order right away. Under Deal #1, MailMerge is \$149.40, plus \$3 shipping and handling. We also have a limited quantity of extra WordStar/MailMerge manuals available for \$20 each plus \$4 shipping and handling (binder not included).

"How to do it on the TRS-80" by William Barden is an excellent book for all TRS-80 owners and LDOS users. This book talks about both hardware and software (including LDOS) in a very readable manner. Mr. Barden has always received great praise for his books on microcomputers and assembly language programming, and has done it again with this book. This is a must for all TRS-80 owners. Whether you buy it from us at a discount, or pay full retail, don't boot up without it.

The price, you ask? Well, Deal #1 was only for software products, but "Such a deal we have for you". The normal retail price is \$29.95, but order it from LSI for just \$25 by itself, or for \$20 with any order over \$50 in other merchandise. Enclose your check with your order, and LSI will pick up the added shipping charges. Otherwise, add \$2 for shipping in the United States. Now how many do you want?

**DEAL #2:**

is over! Please note that the new LSI catalog will be available Real Soon Now. We will probably start mailing it out by the beginning of June. Unfortunately, along with the new catalog will come some price increases (except for LS-LED, which will go down to \$49). This makes \*now\* the time to place that order you've been holding off on. Orders postmarked before June 1st, 1984, or placed by phone before that date will be honored at the old prices.

**DEAL #3:**

continues! We said "until we run out" and we meant it (and still do!). With any order totaling \$50 or more, LSI will include a set of Volume 2 Journals/Quarterlies, Numbers 1 through 4 at no extra charge. There is still a reasonable supply left of all four issues. Even if you already have all these issues, please take advantage of this offer and give them to a less fortunate friend.

**DEAL #4:**

Speaking of that less fortunate friend, how many of your friends are not using LDOS on their TRS-80? Tell them about Deal #4. They can take advantage of it and trade in that "other" operating system. Until June 30th, 1984, LSI will take either NEWDOS80/v1/v2 or DOSPLUS 3.4/3.5/4 in trade towards the purchase price of the LDOS 5.1 operating system. Send your old MASTER diskette and MANUAL to LSI, and pay only \$64.50 plus \$5 shipping and handling for a brand-new copy of the LDOS 5.1.4 operating system (specify model). The only operating system/version other than those mentioned above that qualifies for this Deal is 5.0 Model 1 LDOS. It may be traded in on LDOS 5.1.4 for the Model 1.

**DEAL #5:** Sorry, all the extra Radio Shack Hard Disk Systems are gone.

**NEW PRODUCT ANNOUNCEMENTS**

LSI Publishes LDOS Source Code

Logical Systems, Inc. announces publication of the complete, commented, assembler source code for the LS-DOS/TRSDOS 6.2 operating system. This may be the first time that the complete source code for such a sophisticated operating system has been made available to the public at a reasonable price.

The publication is entitled LS-DOS/TRSDOS 6.2 "THE SOURCE", and is published in three 8 and 112 by 11, soft bound volumes:

- Volume #1 - THE SYSTEM           L-60-011
- Volume #2 - THE LIBRARIES       L-60-012
- Volume #3 - THE UTILITIES       L-60-013

Each volume is available for \$99, or the complete set of three as L-60-020 for \$249. Delivery of all volumes will begin in June of 1984.

New TRSDOS 6 BASIC Utilities

Logical Systems, the authors of the TRSDOS 6.x operating systems licensed to Tandy/Radio Shack for the Model, 4 have announced a new product package for use by BASIC programmers, called "BSORT/MOD324".

This package includes a very powerful SORT utility, called "BSORT", which is executed from BASIC to sort arrays. BSORT is written entirely in assembler using advanced sorting techniques for super fast operation. Tag arrays, index arrays, string and numeric arrays, mid-string sorts, ascending and descending sorts and much, much more are supported. BSORT requires TRSDOS 6.1.2 or 6.2.0 for proper operation. The TRSDOS 6.1.2 release includes a new version of BASIC, and should be available from Radio Shack by the time you read this. Radio Shack's catalog number for TRSDOS 6.1.2 is 700-2246.

A BASIC program conversion aid called MOD324 completes the package. Again, MOD324 is written totally in assembler for speed. This product will convert a Model 3 BASIC program to a Model 4-type format. MOD324 is even capable of adjusting print locations on the screen ("PRINT @" and "PRINT TAB") as well as pointing out the lines that need further attention after the convert. BSORT/MOD324 is available as L-32-210 for \$49.

#### LSI Announces FED86

FED86 is the LSI all-purpose File and disk Editor for the IBM-PC and PC-compatible machines (\*not\* the Tandy 2000) running under MS-DOS or PC-DOS (version 2.X required).

Any byte in any given file or disk can be displayed and/or altered. The display information in file mode includes: 256 byte record display, file name and drive number, record number and relative byte number within the current sector. In addition, the value of the byte under the cursor is displayed in hex, decimal and binary. In disk mode, the disk relative sector is displayed instead of the file name.

FED86 also includes commands for: searching for hex or ASCII strings, a case-independent text search, sending a record or group of records to the printer and modification of the displayed data (either in hex or ASCII).

All in all, FED86 is a tool that no MS-DOS user should be without. Even if these operations sound difficult, FED86 makes them a snap!

#### LSI Announces the LS-Utility Disk

The LS-Utility Disk can be considered as "the best of 5.1 for 6.X". It includes the majority of the most popular filters and utilities from our LDOS 5.1 Filter Disks #1 and #2, and our Utility Disk #1, reconfigured for use under LDOS/TRSDOS 6. Projected availability is June 1984, and the programs included with the LS-Utility Disk are expected to be:

PRCODES/FLT	On printers that will backspace, provides for easy control of boldface and underlining, and also provides slashed zeros.
TRAP/FLT	Traps and throws away any user-defined character.
MAXLATE/FLT	A complete translation filter system for input or output devices. Includes tables for EBCDIC and DVORAK translation. Custom tables are easily built for any application, and a string of characters may be specified to replace a "match".
CALC/FLT	*KI filter - HEX/DEC/Binary conversion and HEX add/subtract.
KSMPLUS/FLT	Same as KSM, but allows key redefinition "on-the-fly" from the keyboard. Also includes "command repeat" and more.
RDTEST/CMD	Non-destructive forced read of an entire diskette.
READ40/CMD	Allows read-only access to forty cylinder diskettes in a eighty cylinder drive by double-stepping (96 TPI only).
TYPEIN/CMD	Combines the functions of JCL and KSM. Allows most programs that won't normally accept JCL to be controlled automatically, if they honor the device 1/0 structure of LDOS/TRSDOS 6.

#### Items of General Interest

The current release of LDOS for the TRS-80 Models 1, 3 and the LOBO MAX-80 is 5.1.4, with file dates of 10/01/83.

The latest (known) release version of TRSDOS 6 is 06.01.02. In this release, there are no operating system changes from 06.01.01, but there is a new version of Microsoft BASIC supplied by Radio Shack. This version can be obtained from any Radio Shack Computer Center upon proof-of-purchase (of a Model 4 or 4P). Your old 06.00.00 to 06.01.01 DOS disk should serve. The Radio Shack stock number for this update is 700-2246. Given the heavy demand for this item, your Computer Center may not have it in stock, but can order and/or reserve one for you.

Radio Shack Hard Disk Owners who have not yet received the hard disk drivers to allow the use of the hard disk under TRSDOS 6 should get a copy of 700-2247. This is TRSDOS 06.01.01 along with the hard disk drivers (and formatter). You will also need to get the above mentioned 700-2246 so that you have the new version of Microsoft BASIC. Again, the Computer Center may not have a 700-2247 in stock for you, but can order it.

The following patches have been requested to alter the time/date prompts and commands to accept a period instead of a colon or slash as a delimiter. This patch accepts a period only (not almost any character as in 5.1.4). The first patch is for the boot-up prompts, and should be applied to SYS0. The second is for the DATE and TIME commands, and should be applied to SYS7.

```
.T611SYS0/FIX
. Patch to TRSDOS 06.01.01 as distributed by Radio Shack
. Use the BUILD command or an editor to type in this patch
. as shown, and then PATCH SYS0/SYS.LSIDOS T611SYS0/FIX
.
. This patch forces boot-up date and time prompts to accept
. the period as a delimiter instead of the slash or colon
.
. This part changes the date prompt
D0E,B3=2E
F0E,B3=2F
.
. This part changes the time prompt
D0F,9E=2E
F0F,9E=3A
. End of patch

. T611SYS7/FIX
. Patch to TRSDOS 06.01.01 as distributed by Radio Shack
. Use the BUILD command or an editor to type in this patch
. as shown, and then PATCH SYS7/SYS.LSIDOS T611SYS7/FIX
.
. This patch forces the DATE and TIME commands to accept
. the period as a delimiter instead of the slash or colon
.
. This part changes the date command
D01,EB=2E
F01,EB=2F
.
. This part changes the time command
D03,05=2E
F03,05=3A
. End of patch
```

The Model 1 to Model 3 upgrade/replacement board mentioned previously by Tim Daneliuk in his \*\*\* Parity = Odd \*\*\* column is available from a company called Northern Technology. They are located in Elk Grove Village, Illinois, and their phone number is (312) 860-1772. By now you may have already seen their ads in the TRS-80 magazines.

The following corrections to the article "Profile One Plus" in the January '84 LSI Journal were prepared by Joseph J. Kyle-DiPietropaolo:

The patch files for RM, CM, EFC8, EFCC and EFCM needed correction. Using the new patch files on the next page with the rest of the previously published patch files will result in a properly operating version of Profile One Plus. Please note that these patches are for 01.00.00 or 01.00.01 only. With version 01.01.00, all patches except EFCM/FIX seem correct, but have not been tested extensively. With all versions, PR/FLT should be installed with the SLINE=1 parameter to ensure proper report pagination:

**FILTER \*PR PR/FLT (SLINE=1)<enter>**

```

.CM/FIX *
X'7BC2' =67 44
X'74B8' =18 43
X'7652' =18 43
X'765E' =18 43
X'74BB' =19 43
X'7661' =05 44
X'7832' =05 44
X'761A' =49 40
X'761E' =49 40
X'7617' =01 30
X'782C' =01 30
X'7ABE' =01 30
X'7B3F' =01 39
X'7151' = "ONE"
X'73C5' =DE 41
X'74F6' =63
-----
.RM/FIX *
X'771A' =05 44
X'7770' =05 44
X'779C' =05 44
X'7956' =05 44
X'7598' =18 43
X'770F' =18 43
X'7717' =18 43
X'7765' =18 43
X'776D' =18 43

X'778E' =18 43
X'7796' =18 43
X'759B' =19 43
X'75EO' =63
X'7C63' =01 30
X'7BE2' =01 30
X'7950' =01 30
X'76CE' =01 30
X'714B' = "ONE"
X'7CE6' =67 44
X'7591' =DE 41
X'7799' =DE 41
-----
.EFC8/FIX *
X'56F2' =67 44
X'543C' =01 30
X'5668' =01 30
X'56E9' =01 30
X'707B' =01 30
X'70AD' =01 30
X'7BAF' =01 30
X'7BE7' =01 30
X'7BDE' =18 43
X'7BED' =18 43
X'8DE7' =18 43
X'8DFE' =18 43
X'8E51' =18 43

X'5442' =05 44
X'70D4' =05 44
X'7BF0' =05 44
X'8DD9' =49 40
X'9730' =49 40
X'6DF8' =C4 44
X'6E06' =C4 44
X'5764' =5A 44
X'564C' =5D 44
X'8DD2' =DE 41
X'74C9' =60
-----
.EFCC/FIX
X'56E7' =67 44
X'9625' =67 44
X'97E9' =67 44
X'9FFD' =67 44
X'A01F' =67 44
X'565D' =01 30
X'56DE' =01 30
X'73B1' =01 30
X'73D5' =18 43
X'73DD' =18 43
X'9228' =18 43
X'9248' =18 43
X'9298' =18 43
X'73E0' =05 44

X'7406' =05 44
X'9225' =49 40
X'5641' =5D 44
X'5759' =5A 44
X'7100' =C4 44
X'710C' =C4 44
X'587B' =60
-----
EFCM/FIX *
X'8CFF' =67 44
X'7379' =96 43
X'7601' =05 44
X'8843' =05 44
X'8986' =05 44
X'76EC' =C2
X'7423' =18 43
X'75D2' =18 43
X'75FE' =18 43
X'8835' =18 43
X'8840' =18 43
X'882B' =01 30
X'8980' =01 30
X'8BFB' =01 30
X'8C7C' =01 30
X'8772' =DE 41
X'8D7D' =60

```

## BUSINESS CHECK LEDGER + ADVANCED BOOKKEEPING SYSTEM \$99.50

Checks ledger, expense accounts ledger, deposit accounts ledger, deposit ledger, accounts payable ledger, payee ledger, bank file, add — edit — display — print for each file.

Handles three banks with three series of checks numbers per bank. Can split checks into nine different expense accounts. List checks by check number, payee, or expense. Reconciles bank balance by clearing checks and deposits. Requires two disk drives, printer and 48K RAM. Integrates with RS disk payroll 26-1556. Model I, III, 4 or MAX80. Manual contains step-by-step procedures for all features and entries.

### Data Stored in Business Check Ledger +

#### BANK FILE <Max 3 banks>

(1) Name of bank (2) address of bank (3) city, state, zip of bank (4) balance (5) first check number entered in computer (6) last check number entered in computer (7) last check number cleared

#### CHECK FILE <Max 9 — three banks and three series each bank>

(1) Check number (2) check ID <1=current 2=cleared 3=split expense 4=void> (3) amount of check (4) payee code <payee file number> (5) expense code <expense file number> (6) date of check

#### DEPOSIT ACCOUNT FILE <Max 100>

(1) Deposit account number (2) deposit name (3) quarter to date <QTD> (4) year to date <YTD>

#### EXPENSE ACCOUNT FILE <Max 100>

(1) Expense account number <general ledger account number> (2) expense name (3) quarter to date <QTD> (4) year to date <YTD>

#### PAYEE FILE <Max 998>

(1) Payee name (2) payee address (3) payee city, state, zip (4) expense account number normally associated with payee (5) quarter to date <QTD> (6) year to date <YTD>

#### DEPOSIT FILE <No max>

(1) Deposit ID <1=uncleared 2=cleared> (2) deposit amount (3) date of deposit (4) account deposited to

#### ACCOUNT PAYABLE FILE

(1) Date due (2) amount due (3) payee code (4) expense code

FOR DETAILS WRITE

**JAMES RUSSELL**

110 Beechmont Dr., Carmel, IN 46032, (317) 846-8553



The following optional patch for LDOS 5.1.3 or 5.1.4 on the Model 1 will allow the system time and date routines to utilize a hardware clock such as the Alpha Products Newclock80 or Timedate80, and most other clock devices that use the MSM5832 clock chip. A similar patch is not possible on the Model 3 as the real-time-clock service routines are in ROM. A new service routine could be written as a RTC Task, however.

This patch assumes that the clock is addressed as ports B0 through BF. If your particular clock module is addressed differently, the underlined bytes may be changed. For instance, if your clock uses ports C0 to CF, change the underlined B5 to C5, and alter the rest of the underlined bytes in a similar manner.

```
. Patch to SYS0/SYS.SYSTEM to allow use of a clock device
. Fix the timer interrupt routine
D04,08=D2 45 ED 78 OD CD A6 47 ED 78 OD E6 OF 85 12 1B
D04,18=C9 11 43 40 01 B5 03 CD C3 45 10 FB
. Fix the date initialization on BOOT
D0D,58=21 46 40 01 BA 01 CD C6 4E 06 03 CD C6 4E 01 BC
D0D,68=0F CD C6 4E EB D6 50 47 DB BB E6 03 11 45 40 21 48 40 20
D0D,7B=26 CB FE 18 22 ED 78 OD A0 07 57 07 07 82 57 ED
D0D,8B=78 OD ED OF 82 77 2B C9
. End of patch
```

### Using Profile 3 Plus under LDOS (revisited)

by Joseph J. Kyle-DiPietropaolo

Profile 3 Plus and LDOS are a very powerful combination. There are, however, several things that can cause trouble if you are not careful. Let's take them one at a time.

First, what version should you buy? Well, the only version that will run under LDOS is the "Hard Disk" version. Don't let that phrase scare you, as it works quite well on floppy disk also. At least two double-density drives are required, and if you are going to do any fancy sorting, three would be better. Double-sided and/or eighty cylinder drives can be used if you have them.

The "Hard Disk" version is \$100 more than the "floppy" version, but this is more than offset by the fact that PROSORT is included. PROSORT is Small Computer Company's disk virtual sort and enhanced selection module, which normally sells for \$150 by itself. If you already own the "floppy" version, Radio Shack will sell you an upgrade to the "Hard Disk" version for \$100 and proof of purchase for the "floppy" version. Their catalog number for the upgrade package is 700-6203.

In terms of actually running Profile 3 Plus, there are three things to keep in mind:

1) If you have the \*KI driver installed, all references to the <clear> key in the Profile documentation should be changed to <shift><clear>. If you are using math fields, do not use type-ahead, as Profile does not seem to calculate properly if <shift><clear> is struck before the calculation is complete.

2) Profile checks the printer status by looking directly at the hardware. This means that if you are using the spooler, the printing rate will not really improve much. This can be corrected by searching through all the EFC programs (and RM/CM) for the byte sequence 3A E8 37 and replace it with 3E 30 00. This is a good excuse to learn how to use FED (included with the LDOS 5.1.4 update). Making these changes will disable the check of printer status. Since the system printer driver (which contains its own "Printer \*BUSY\* test") is used to actually output the data, there will be no conflicts. Also, you will be able to route printed reports to a disk file without having a printer ready and on-line.

3) Lastly, contrary to what the manual says, Profile will \*not\* work with "DO files" under LDOS. This is because of the way keyboard input is requested by Profile. This difficulty is neatly fixed by using the TYPEIN utility from the LSI Utility Disk #1 (Order L-32-070, \$39 direct from LSI).

When creating a file of commands to be used with TYPEIN and a Profile user menu, put the entire command sequence in the data file. This includes the EFC line. Build the file by -putting in the exact keystrokes you would type if performing the procedure by hand. Then, invoke the procedure with TYPEIN. Do not attempt to pass the name of a DO file to the EFC module by placing it in parenthesis as stated in the Profile manual, as it will not work.

For example, let's say that we want to be able to expand our existing data file by 100 records by pushing one key. First, create a user menu that has an entry called "Expand the MAIL file by 100 Records". Set up the user menu so that this entry is executed by pushing "E". When "E" is pushed, it should execute the command: **TYPEIN EXPAND/KYS**. The TYPEIN data file you would have to build (with the name **EXPAND/KYS**) is given below. Note that <enter> means the depression of the key marked "ENTER".

Contents of the file EXPAND/KYS:

```
EFC7<enter>
MAIL<enter>
100<enter>
```

LSI's LED, the LDOS text EDitor (L-30-020, \$29) is an excellent tool for generating and maintaining these keystroke files for TYPEIN. LED also has a special HEX mode that will allow you to insert the <shift><clear> code (X'1F') directly into a TYPEIN keystroke file. This character is needed to terminate "extended mode" functions. Without LED, it can be done but it is not easy.

If you must use BUILD, and you need to get a <shift><clear> keystroke into the TYPEIN file, do the following: Make sure that KI/DVR is installed when doing the BUILD. When you get to the point at which you need to insert the <shift><clear> keystroke, actually put a <clear><shift><enter> character there. That character is produced by pressing the <clear> key, and while still holding is down, press the <shift> key, and while still holding both of them, press the <enter> key and then release them all. A Plus/Minus symbol ( $\pm$ ) or square block on the Model 1 and MAX-80 will appear on the screen. Now continue with the remainder of the key data. Whenever you need to execute this kind of a TYPEIN procedure, use the following format: **TYPEIN SELECT/KYS (X1=X'7F1F')**. Each <clear><shift><enter> will now be translated to the required <shift><clear> when executing. This TYPEIN command can, of course, be placed in a user menu.

All prices mentioned above that pertain to LSI products are valid only through May 31st, 1984. Radio Shack pricing policies and product availability is subject to change. Please contact your local Radio Shack Computer Center for more information.

### LSI Quick Hint #3

Don't forget about the LDOS SIG (Special Interest Group) on CompuServe. If you are already a CompuServe member, simply GO PCS-49. If are not a member, any Radio Shack Computer Center can set you up with a membership kit (note: RS232 and modem required). There are many useful public domain programs and utilities in the LDOS SIG databases, and questions about LDOS and TRSDOS 6 will be answered promptly, often both by the LDOS Support staff and experienced users. This is an excellent way to keep abreast of the latest in the LDOS world. For most people, CompuServe is only a local call away, and that local call plus CompuServe's modest \$6 per hour fee is a lot less than a long distance call during the day to LSI.

# JOURNAL

## View from the bottom floor

Bill Schroeder, Logical Systems, Inc.

Hello to all *LSI Journal* subscribers and readers of *Basic Computing*. From this point forward, the contents of the *LSI Journal* will be presented in *Basic Computing*. We feel that this will be of great benefit to both our tens of thousands of users, and to the existing fans of *Basic Computing* magazine. If you are a subscriber to the *LSI Journal*, your subscription will be fulfilled by *Basic Computing*, which will contain this *LSI Journal* section. Our subscribers gain the additional information, ads, and interesting articles from the rest of the magazine, while *Basic Computing* subscribers gain technical information and articles of special interest concerning LDOS 5.1.x, TRSDOS 6.x, LSI products, and last (and least), my ramblings.

Just so rumors don't start to fly ... *LSI* was *not* bought out by *Basic Computing*, nor were they bought by *LSI*. There is NO connection between our companies. The name *LSI Journal* still belongs to *LSI*, but *Basic Computing* has been granted the right to use the name as needed for promotional purposes both in and outside of this publication. *LSI* provides all the editorial material in our corner of *Basic Computing* magazine. No money changed hands between the folks at *Basic Computing* and us here at *LSI*.

This arrangement is purely one of convenience. *LSI* is a software company, *Basic Computing* is a magazine. Both companies are very good at what they do best, and less efficient at "the other guy's specialty". So, *Basic Computing* will be our publisher.

Mike Schmidt (the owner of *Basic Computing* magazine) and I have been discussing the possibility of this arrangement for well over a year now.

Being from a conservative German background, I tend to move very slowly on this type of major decision. To slow things down even more, Mike comes from the same conservative German background. Between the two of us, we were almost moving backward. The final result is, from now on, the *LSI Journal* will be incorporated in (about) every third issue of *Basic Computing* (read that as at least that often).

Should you have occasion to write or call regarding information presented in this area of the magazine, please contact *LSI* directly at: 8970 N. 55th St. PO Box 23956, Milwaukee, WI 53223 (414) 355-5454. Of course you can always contact *Basic Computing* also, but I would like to make it very clear that *LSI* provides all of the material contained in this section of *Basic Computing*.

Now down to what's happening at *LSI*. There have been many changes at *LSI* as we prepare to enter the MSDOS world (IBM & compatibles). We *will not* be providing an operating system for this market... at this time. Our first product will be a truly "RELATIONAL" data base manager. This product is very dynamic in concept for a micro. It will be easier to use than any existing product of similar capabilities, faster, more versatile and will be very "friendly" to the average user. The name and price for this product remain undecided at the time of this writing, but should be known by the time you read this. Please call for additional information if you are interested. Note: This product will be for use with MS-DOS 2.x only.

**Question: What is AT&T up to!?**

Answer: Who knows?

One thing is certain: with *IBM* having sold about 850,000 PCs in 1983, and projecting about 2,500,000 (that's right--2.5 million!) to be sold in 1984, it will take a very heavy hitter to slow them down. If there is a company that can actually go head to head against *IBM* in the small business and professional market, it is most certainly *AT&T* (whoopsthat \$ must have been a slip). *AT&T* is one of the largest and richest companies in the world (much larger than even *IBM*).

*AT&T* could have kept their antitrust case of the Seventies in court for another 20 or 30 years if they wanted to. But lo and behold, all of a sudden they offer a settlement to the U.S. government. What is most amazing is the fact that the airheads in Washington, D.C. accepted it. *AT&T* drew their own draft for a break-up and reorganization of their own operations. They would agree to this "massive" break-up only if they would be allowed to enter "new markets", that they had (years earlier) agreed to stay out of. One of those markets is, of course, computers and other end-user DP equipment and services.

For many years, the fellows up at *Bell Labs* have been far ahead of private industry in computer technology. They have perfected practical bubble memory, 32-bit microprocessors and some of the best concepts (and implementations) in the software industry. Now, *AT&T* owns *Bell Labs*, and gets to take the cost of operating it as a TAX DEDUCTION (by a special Act of Congress). Boy, I wish *LSI* could write off our costs of software development that way.

*AT&T* has more ready cash, a larger manufacturing capability, a larger support

system, and a larger marketing system than any company in the microcomputer field today. For that matter, make that ANY company at all, in any field.

#### **And now AT&T cometh**

Let us imagine an AT&T manufactured, marketed, and supported microcomputer system. The word "system" is very important here, and is going to encompass a lot more than what IBM, Apple or Tandy calls a "system" today. Think more of something like the United States telephone "SYSTEM". That, of course, was the last AT&T controlled accomplishment. The U.S. phone system (and other types of communications associated with this "system") are without a doubt a major reason for the prosperity, industrial growth and power of the U.S.

I believe that the next AT&T system will begin to take shape in 1985, and be providing full services to the top 50 U.S. population centers by 1990, with full coverage of the North American continent by 2000. Let's take a look at what this next "system" might bring.

With the present cable, microwave, satellite, and radio network under the direct and indirect control of AT&T, it is possible today to communicate in "full duplex" with almost every residence and business in the United States, and many foreign countries. Much of the capacity of this network is already utilized for data transmission, collection and distribution. This usage is for many different reasons, and by many varied interests ranging from Government activities to medical education to airline pilot weather data. Note: at present, the list of data accumulation and distribution uses for the AT&T system is growing at an accelerating rate, faster than ever before, and this rate will continue to accelerate for a long time to come as the true potential of the system, and the hunger for "DATA" is exploited by our friends at AT&T.

I believe that AT&T is about to introduce a new computer intended for everyone, from the home user, to the president of a large company, to the bank around the corner. This computer will become part of a system of distributive processing the likes of which the world has never dreamed of. These machines will be entirely "solid state" in that they will contain no moving parts (other than the keyboard system until full voice control is perfected).

These machines will have a megabyte or more of bubble memory, and at least 256K of regular RAM (bubble memory is relatively slow). There will be a 9- or 12-inch monitor, with color and graphics capability, and TV interface options. A keyboard of 90-plus keys will be provided and initially a 1200 to 4800 baud communication system. This will all be available to the customer as components, CPU, keyboard, and monitor, with several options for each component.

The systems will initially rent for between \$50 and \$250 per month, with all maintenance provided by AT&T. By 1990, the price should fall to around \$25 to \$75 (or the equivalent at the time, adjusted for inflation). The initial thrust of this product will be aimed at the business market place. As production distribution and profits rise, they will decrease the costs to truly enter the "home" market at around the cost of present phone service.

#### **Now for the "magic" - the SOFTWARE system**

The most expensive and the most volatile part of any computer system is the software. Not the operating system itself, but the user applications. The mass market for "software" does not exist! But the "mass use" potential of software does exist. Think about it. Any particular piece of music is enjoyed by many more people than own the recording. Movies are watched by tens of thousands more than own a legitimate copy of the movie. Libraries are used by many more people than own complete libraries, or for that matter, those that even own a significant number of books.

For the want of a better term, let's call this concept "mass-use software". All the items mentioned above are "authored", "edited", "produced", "published", "stored on media", "reproduced" and then "marketed". Computer software is handled in a similar manner until it reaches the "marketed" stage. In most cases you must buy or license a physical copy of the reproduced media to use the product. That would be like having to buy a "print" of a movie so you could watch it, or marketing a book and NOT allowing it to be added to libraries.

This is not efficient, and would have greatly reduced the mass acceptance of products from the aforementioned

industries. In the early days of each of these industries, the one to one marketing concept was all that existed, just like the handling of software today. I believe AT&T is about to change that, and bring computing in the U.S. to a new age of maturity.

AT&T could easily provide fifty or more supercomputers, properly placed around the country. When you turned on your own AT&T computer, it would automatically establish a link to the local supercomputer. You would then select the software you wished to use for the day (or as long as your machine stays on). The software and all support files would be sent to your machine. The host system would automatically know who you are and provide you with your personal data files at the same time. It would then disconnect from your computer. When you complete your work and sign off at the end of the day, your machine would automatically call the host and send back your updated data files, and then shut itself off.

Initially, data transmission speeds will be limited to 4800 bits per second, or less. This will change rapidly in the 1990s as fiber optic cables are laid, and fiber optic links enter most houses in the U.S. This "optical cable" will carry your T.V. as well as your phone and computer services, at up to hundreds of times faster than existing data transmission speeds.

There are several very strong points in favor of such a system. First, software piracy is eliminated (or made very difficult), as only the host has your data, and you will not have on-sight permanent storage. Handling of diskettes, making proper backups of data files and the like would not be the user's problem (most users don't handle this job correctly anyway).

Second, the customer would always have the most recent version of the software to run. Then, if a bug is reported to a vendor, he simply corrects the bug and updates the product on any one of the host supercomputers. In the middle of the night, the host machines exchange updated files. The next morning, all software is corrected for EVERY user. The same concept applies to documentation and updates.

Third, you will be able to send data to anyone in the country in just hours or maybe minutes. The postal service would

finally be dealt a long deserved death blow.

Fourth would be the reduced cost of such a system for the user. How much software do you own that you DO NOT USE? With this system, you could select between dozens (or hundreds) of word processors or spreadsheet programs, and only pay when you use them. The "Software Usage Fee, For End Reception" ("SUFFER" for short) would probably range from 25 cents to as much as \$25 per day for "rental". AT&T would keep a percentage of this fee as a "distribution" charge (20 to 30% seems reasonable). The SUFFER charges would be levied on your regular phone bill, along with: your mail charges, T.V. usage, data file storage charges, and of course your charges for local equipment, options and services. Sounds like AT&T will be getting a lot more out of us than when they were "just the phone company".

One last point is that AT&T would probably have little or nothing to do with applications software (too much trouble). They would accept most professionally written packages for installation on the system. It would be up to the developer to promote and advertise the product to generate usage. Programs with very limited use would be removed from the system. A good software product will make millions for the authors under this system. Users will get much better quality software, as "JUNK MERCHANTS" will not be able to survive under this type of system. The computer owners would have to use a product repeatedly to make it a success.

One big negative is that this whole system will make 1984 (the "Big Brother" syndrome) a reality, and turn Orwell into -a prophet. The government will be able to get their hands on "anything and everything", from payroll data to the letter

you write to your mother, and how much you owe on your house, directly through the system. Tax collections should skyrocket. This is the main reason why I feel this overall concept will become reality. The government will want access to the tremendous database created by this type of system. Complete electronic banking will become reality (there will be no cash transactions to avoid taxes) and no one will have any truly private information.

The end of an era and the dawn of the information revolution is upon us. Whatever the outcome, our civilization is about to undergo radical change.

For all that the AT&T Mega-lith has done for the American people in the past, AT&T, we thank you. One can only wish that the results of your next accomplishment will be as benevolent to the American people. I hope so.

## Locating high memory routines under LDOS/TRSDOS 6.X

**Richard Schulman, MagiComp  
2710 W. Country Club Rd., Philadelphia, PA 19131**

A short while after my Mod 4 came rolling in, I got the tech manual and turned to the SVC section to find out how to convert my Mod 3 subroutines. One of the first SVCs I looked for was the one to get USTOR\$ - the LDOS 8 byte storage area allocated to the user. I store the addresses of my subroutines there so that I can find them from LBASIC. Unfortunately, I never found the SVC -because there isn't one. So I plunged in to find out how to locate my routines.

LSI has adopted a convention for a header to precede high memory routines. Using this header allows an SVC to locate the routine for you. Listing I demonstrates the technique by fetching the address of INKY4.

### Listing 1

```
START: LD    DE,FSPEC      ;header to find
        LD    A,83        ;GTMOD
        RST  28H
FSPEC: DEFM  'INKY4 '
        DEFB  0DH
```

DE is loaded with the address of the name of the routine you want to find, and A with the number of the SVC (in this case 83). The name must be in UPPER CASE characters and terminated

### Listing 2

```
PUSH HL           ;Save the parameter address
LD E, (HL)       ;Load the actual parameter
INC HL           ;itself into DE
LD D, (HL)
LD HL, 18
ADD HL, DE       ;Add 18 to DE
EX DE, HL        ;Address of 'INKY4' to DE
LD A, 83         ;SVC number
RST 28H
POP DE           ;Recover parameter location
EX DE, HL        ;Move routine location to DE
LD (HL), E       ;Move LSB to parameter
INC HL
LD (HL), D       ;Move MSB to parameter
RET
DEFM 'INKY4'
DEFB 0DH
```

### Listing 3

```
14 DIM US(11):FOR X=0 TO 11:US(0)=0:NEXT X
15 DATA 24293,22051,4641,6400,16107,-4269,
-5167,9075,-13966,20041,22859,3380
16 FOR X=0 TO 11:READ US(X):NEXT X:
X=VARPTR(US(0))
17 CALL X(X):INKY4=X '** SAVE INKY4 ADDRESS
20100 '** INKEY ROUTINE
20105 CALL INKY4(Z)
```

### Listing 4

```
;*****
;** HEADER **
;*****
BEGIN: JR START
DEFW LAST-1 ;HIGHEST MEMORY BYTE
DEFB 5
DEFM 'INKY4'
MODDCB: DEFW $-$
DEFW 0
START: PUSH HL ;SAVE LOCATION OF PARAMETER
```

with a character whose code is in the range 0-31. After executing the SVC with RST 28H, the starting address of INKY4 is in HL, and the Z flag is set (NZ if it wasn't found).

Of course, that doesn't entirely solve the problem of how to get this address into BASIC so that you can CALL (or USR) the routine.

The method I chose involves passing a parameter to the routine, and sending the address back in the parameter. The key to finding the routine is to know where to find the name of the routine (so it can be loaded into DE). The program is shown in Listing 2. The parameter passed to this program is the ADDRESS of the program itself.

That is the secret to finding the location of the name of the routine you wish to locate. The program is exactly 18 bytes long. Therefore adding 18 to the address of the program (the parameter we passed) gives us the location of the name of the routine we are searching for. That is the purpose of the LD HL,18 and ADD HL,DE instructions. The 18 instructions can be loaded into an integer array as shown in the BASIC program in Listing 3.

On line 16, X is set to the location of X(0) in memory. It is both the location of the routine CALLED in line 17 and the parameter passed to it. The integer INKY4 is the location of the high memory routine.

There is one more secret to success. You have to translate the name of the high memory routine into integers. The letters I,N,K,Y and 4 are represented in memory by the ASCII codes 73,78,75,89 and 52 respectively. To find out what integers to use, let's look at I and N. Those two letters make up one integer. Set I%=0. Then poke the codes for the letters into I%: POKE VARPTR(I%), 73: POKE VARPTR(I%)+1,78. Then PRINT I% and you will find that I%=20041. The value for NK is 22859, and the value for 4 + carriage return (code 13) is 3380. Those are the last three values in the DATA statement at line 15.

If your routine name has an even number of characters you can terminate the name with an integer value of 0-31 (assuming an extra byte of value 0 following the terminating character). I just use 13 from habit.

### High Memory Headers

None of this is possible without the proper header. My header for INKY4 is shown in Listing 4. The first two bytes jump to the start of the actual routine. Next is a two byte integer with the address of the highest byte of memory occupied by the routine. Then one byte which gives the length of the name of the routine, 5 in this case INKY4. There follows two byte reserved for the address of a Device Control Block if the routine is associated with a device and two bytes that are reserved for I don't know what.

### Do it Again

You can reuse the routine in lines 14-17 in the same program. Line 18 would reset US(9) to (11) or however many elements after US(8) need to be changed. For example, let's find the routine FLASH. Line 18 would be

US(9)=19526:US(10)=21313:US(11)=72 (I terminated 'FLASH' with a 00), followed by X=VARPTR(US(0)):CALL X(X):FLASH=X. You can do it as many times as you need in the same program. And since you can wipe out arrays in BASIC under LDOS (TRSDOS) 6.x, you can ERASE US when you're done to reclaim its space and have the addresses of your routines in the integers INKY4 and FLASH (and whatever else).

Be sure to put X=VARPTR(US(0)) before each CALL X. BASIC moves things around dynamically and the address of

US(0) might change between calls to X. And be absolutely certain that X is an integer. I can promise you from experience that you will not like the results if X is not an integer.

That's all there is to it. Simply MERGE lines 14-17 with your BASIC program, then remove the last three integers and replace them with the name of the high memory routine you are looking for.

## \*\*\*Parity = Odd\*\*\*

© 1983, Tim Daneliuk, TH Communications Associates

Welcome to PARITY = ODD! This may be the first time you've seen this column, so introductions are in order. PARITY = ODD originated in the early days of the LDOS Quarterly. At the time, I was reviewing TRS-80 related products for several magazines, as well as beta testing products for the LDOS operating system. In talking with Bill Schroeder of Logical Systems, it became clear that LDOS was soon to become a dominant force in the TRS-80 industry. And what a force it is! There are LDOS products for the TRS-80 Models 1 and 3, and the latest machines in that family (the Models 4 and 4P) use a descendant of LDOS 5.1 as their primary DOS. Yes folks, TRSDOS 6 is a version of LDOS. In fact, LDOS and TRSDOS 6 are so closely related in concept that most of the major commands work almost identically.

Originally, PARITY = ODD was created as a forum for examining topics of particular interest to the LDOS user. In this column you'll see product reviews, discussions of programming technique, solutions to user's problems, and almost anything else that strikes my fancy. If you have a particular question or want to see a specific product reviewed, let me know. Moreover, feel free to bring up topics

unrelated to LDOS directly. In the coming installments you'll probably see items concerning CP/M and MSDOS now that Tandy will be distributing these operating systems.

I can be contacted by mail and via CompuServe. If you choose to write, please send your letter to this magazine and my attention. If you want to "talk" by electronic mail, you'll find me skulking about on the LDOS Special Interest Group (SIG) of Compuserve. My PPN # is 70745,1520. Either way, there are some vital pieces of information you should always include. First, *please* include your name and address (a telephone number is helpful too). Second, *always* include a description of the system you are using. Please state explicitly which TRS-80 or TRS-80 "work alike" you are using as well as the types and sizes of disks drives, controller, etc., you have. Finally, if you are requesting help with a problem, please try to describe the problem in a logical, step-by-step manner. Be as explicit as possible. I can't much with a problem like "It doesn't work right".

Remember, this column reflects my opinion. I try to responsibly evaluate those products about which I write, but errors

can and will happen. If you find such an error, let me know.

### **Chips**

Tandy recently unveiled their MS-DOS machine, the Tandy 2000. The 2000 is notable because it is one of the first personal computers to use the Intel 80186 microcomputer chip. To understand the power of the '186 we need to take a quick look under the hood of the modern microprocessor.

The much-touted IBM-PC uses the Intel 8088 microprocessor. IBM will tell you that this makes the PC a "sixteen bit" computer. Horsefeathers! Internally, the 8088 contains data registers which are indeed 16 bits wide. BUT -externally (i.e. those places where the 8088 "talks" to the rest of the computer and the outside world) data is transferred 8 bits at a time. This is a situation which is much like the 8 bit Z-80 microprocessor. The Z-80 also has 16 bit internal registers and it transfers data to the rest of the system a byte at a time. In fact, the only significant advantage that a 8088 has is that it can address up to 1 Megabyte of memory directly compared to the Z-80's 64K Byte maximum. This 16 Bit internal/8 Bit external architecture is a large part of the reason the IBM-PC is so wretchedly slow! That's just my opinion of course, but

if you want an eye-opening experience, try benchmarking a program on a TRS-80 Model 4 or LOBO MAX-80 against an IBM-PC. The PC will generally be slightly faster, but not enough in my judgement to justify the 30-50 percent price difference between these machines.

So, what's an 80186? It is an enhanced version of the Intel 8086 microprocessor. The 8086 is a processor which has the same instruction set as the 8088. However, the '86 not only has internal 16-bit data paths, it also has EXTERNAL 16-bit data paths. For this reason alone, it generally runs quite a bit faster than the '88. The '186 enhances this further by including almost an entire computer on one chip, and offering a faster clock speed. In addition to the 8086 microprocessor, the 80186 includes such things as a DMA (Direct Memory Access) controller, and memory chip select logic all on one piece of silicon. This makes the '186 a very cost effective part when you're designing a complete computing system. The 80186 also has some new instructions which the '88 and '86 don't have. The '186 also has the advantage of being compatible with all the instructions of the '88 and '86. This means that programs written for the latter two will also run on the 80186!

The bottom line is that you can expect the Tandy 2000 to be quite a performer. Though it does not appear to be "compatible" with the IBM-PC, most software which uses the MS-DOS operating system for I/O (and doesn't try to "talk" to the hardware directly) ought to work with the Tandy 2000. The one potential weakness of the 2000 is that it uses "quad density" (doublesided, double-density eighty track) disk drives. While this gives an enormous amount of storage, it may not be easy to move programs and files to and from "normal" MS-DOS forty track drives. I'm expecting a review machine fairly soon - I'll let you know what I find.

#### **Disk, Disk, Disk...**

If you've followed the evolution of LDOS at all, you probably realize that Logical Systems also publishes many useful add-on products for their operating system. The latest such product is called diskDISK, and will be especially useful for those of you using LDOS on a hard disk

drive. To understand how diskDISK can be used, we need to step back and look at a hard disk system.

By virtue of its tremendous storage capacity, a hard disk can contain literally hundreds of files. While all that storage is great, it becomes difficult to keep track of all the files on the disk after a while. One partial solution to this dilemma is to use hard disk "partitions". A partition is simply part of the hard disk treated as a separate logical drive. For instance, if you had a disk drive with four platters (storage surfaces) each capable of storing 2 Megabytes, you could organize the drive one of several ways. You could have one eight Meg, two four Meg, four two Meg or any combination of the above which totals 8 Megabytes of storage. By the way, partitioning is only possible if the hard disk driver program is written to do so. The point is that one large hard disk is made to "look" like several smaller disk drives. The advantage of this approach is that you break the mass storage into smaller, more easily managed "chunks" of storage.

Even with disk partitioning, there are still times when you will be limited by the large number of files you have to manage on a hard disk. It would be ideal to go even further and sub-divide a given PARTITION of a hard disk. The diskDISK utility does just that. With diskDISK, you can create a "logical" storage area on a hard disk which has the capacity of a 5 or 8 inch floppy. In other words, diskDISK creates a file which is large enough to store as much as say, a single 5" floppy. Then this FILE is installed in LDOS as a logical disk drive.

For instance, I generally use a LOBO 1850 hard disk as the principal drive of my system. This drive stores 8 Megabytes, and is partitioned into two 2 Megabyte partitions (Drives 0 and 1) and one 4 Megabyte partition (Drive 2). On Drive 1, I have a file called C/DD. This is a diskDISK file which ordinarily looks just like any other file to the system. If I issue the command "DD :3 C/DD" from LDOS Ready, this FILE is installed as logical Drive 3. From then on, any LDOS operation on Drive 3 (DIR, FREE, KILL, etc.) takes place physically in the file C/DD. This is entirely invisible to the user. The diskDISK drivers make C/DD "look" like a floppy disk drive. In the case of this particular file, I used diskDISK to "format" it to look like a double-density, double-sided, 40 track disk drive. This

gives me about 360K of storage on this logical drive, more than enough to store a C compiler and my current C program source files.

You can create a different diskDISK file for each major type of file in your system to help organize your files. For instance, I can have one diskDISK set up for BASIC programs, another for PROFILE, and still another for utilities. There's a hidden benefit in all this too. The granule size for an LDOS hard disk system is always at least 4K. Even if your file only has 2 bytes of data in it it will still occupy at least 4K of disk space. When you save a file on a diskDISK, the granule size is that of the floppy disk you're emulating. For instance, files stored on a SSDD 5" diskDISK use 1.5 K granules. Your two byte file will occupy 1.5 K instead of 4 K, for a net savings of 2.5 K of disk space. If you have many small files, the savings are tremendous. A special diskDISK format (called type 1) uses 256 byte granules for the most efficient storage possible.

Another advantage is that a diskDISK only occupies one directory entry on the hard disk, no matter how many files actually exist inside. If you have a lot of small files on a hard disk or 8 inch floppy, you can easily run out of directory space before you run out of disk space. In a sense, diskDISK gives LDOS the capacity of having sub-directories similar to MD-DOS 2.x and UNIX.

diskDISKs can be created on any type of LDOS compatible media. This product is available for either the Model 4 running TRSDOS 6 or the LDOS 5.1.x family of DOS products. Either package costs \$99 and is available from Logical Systems Inc., 8970 N. 55th Street, P.O. Box 23956, Milwaukee, WI 53223.

#### **Random Items of Interest**

Because of the volume of products sent to me for review, it isn't always possible to look at each one in depth.

From time to time, you'll see mini-reviews like these. First, if you're an LDOS 5.1.x user and need a great disk cataloging program, take a look at ZCAT from MicroConsultants. It is written entirely in assembly language and is the best of its type I've seen to date. It costs only \$35 and is available from MicroConsultants, 7509 Wellesley Drive, College Park, MD 20740-3037 (301) 474-8486.



Model 4 owners may be interested in the new "PRO-CESS" utility from MISOSYS. This product is very similar to the CMDFILE program included with LDOS 5. 1, except that PROC-CESS runs under TRSDOS 6. This product also has some nice new twists, like being able to sort load module records by address and conversion of X-type patches to D patches. PRO-CESS is \$40 and can be obtained from MISOSYS, P.O. Box 4848,

Alexandria, VA 22303-0848, (703) 960-2998.

Finally, if you use SuperScripsit, there are two sources of printer drivers you should know about. One is softERware, and the other is PowerSOFT. I've used the softERware product which seems to work fine, but I've not seen the PowerSOFT driver. Contact these companies for more details: PowerSOFT, 11500 Stemmons Freeway, Suite 125, Dallas, TX 75229 (214) 484-2976, or softERware, 300

Grenola St., Pacific Palisades, CA 90272 (213) 459-3414.

### The Finishing Touches

That about wraps it up for this installment of PARITY = ODD. Hopefully the next time you read this column, I'll have something to report on the Tandy 2000, as well as the usual collection TRS-80 reviews and trivia! So long for now.

## The "C" Language, part 6

Pointers, arrays, structures and common errors

**Earl "C." Terwilliger, 647 N. Hawkins Ave., Akron, Ohio 44313**

Could you use some POINTERS on how to STRUCTURE better C programs? In this part, Part VI, structures will be introduced and the discussion on pointers and arrays will continue. Oh? You thought when you read the word POINTERS and the word STRUCTURE that this part would really be discussing techniques for improving your C code? Ha! Well, okay, not to disappoint you, included in this part is a discussion of the most common errors or "things" not to do in a C program. Will that help?

First, let's continue on from the last part with pointers and arrays.

In the last part, you saw an expression `*ptr++`. Were you puzzled? Remember back when the `++` and `--` operators were introduced? It was stated that `++` added one to its operand and `--` subtracted one from its operand. Be careful applying this to pointers. The "one" referred to which is added to or subtracted from a pointer is actually a scale factor. This scale factor is dependent on the type the pointer points to. That means it is scaled by a size equal to the data type length. This holds true for all "pointer arithmetic". (For example, in a Z80 based machine the scale factors are 1 for char, 2 for int.)

There are some rules to follow when doing arithmetic in C using pointers. It is legal to:

- add an integer to or subtract an integer from a pointer
- subtract a pointer from a pointer
- compare a pointer to another pointer

All other conceivable arithmetic, including shifting or masking is illegal. Note: a pointer containing NULL or 0 is a

special case. The C language guarantees that if a pointer points to valid data, it will not contain 0. The 0 value is usually used to indicate an error condition. An example of this would be when a storage allocate function is called. This function may have been designed to return a non zero pointer to the beginning of the allocated storage. If storage can not be allocated it could return a NULL (zero) value indicating an error of some type occurred. Consider the statements below for the discussion following:

```
char *ptr;
static char a[5] = "test";
ptr = a;
++ptr;
```

`ptr` is a pointer to type character. `ptr` is initially set to the address of the array `a`. This is written as `&a[0]` or simply `a`. Next, `ptr` is incremented to point to the next element of the array. This is written as `ptr++`. (Other possible ways to code it, in this example, could have been `*ptr++`, `*++ptr`, `*(++ptr)` or `*(ptr++)`. Note that `(*ptr)++` would create a different undesired result than `*ptr++`. The `++` and `*` operators are of equal precedence and associate right to left.) From the above statements, you can conclude that array subscripting can be done by incrementing a pointer. You can also conclude that the following two expressions are equivalent:

```
ptr = a;
ptr = &a[0];
```

(Note that, in effect, an array name is a pointer expression. Note also that using pointers rather than array subscripting usually results in more efficient code.)

As general rules:

- a[n] is equivalent to \*(a+n)
- \*(&a[n]) is equivalent to \*(a+n)
- &a[n] is equivalent to &a[0]+n is equivalent to a+n

Perhaps if I spelled out how to "pronounce" some of the expressions used in the general rules above, these rules might become more clear?

- & means the address
- \* means the data at
- a[n] means element n of array a
- &a[n] means the address of element n of array a
- \*(&a[n]) means the data (element) at the address of element n of array a  
This says the same thing as element n of array a
- \*(a+n) means element n of array a
- a + n means element n of array a

Did the above help?

If you have been looking at some sample C programs, you may have seen by now that sometimes an array name is written as a[] or \*a when used as parameters in a function. Rather nice don't you think? The function, when passed an array name, can treat it as an array, as a pointer or both. If you have some doubt, look at the C code in Listing 1.

The arguments argc and argv are not new to you, they were described in Part II. As you noted, argv is treated as an array and as a pointer in the above program. Are you curious about the argv[0][0] expression used in the printf function? What will print is a single character, the first character of the command line argument after the program name. If the above program was called test, to invoke it and pass it an argument, you might type:

```
test -l myfile/dat
```

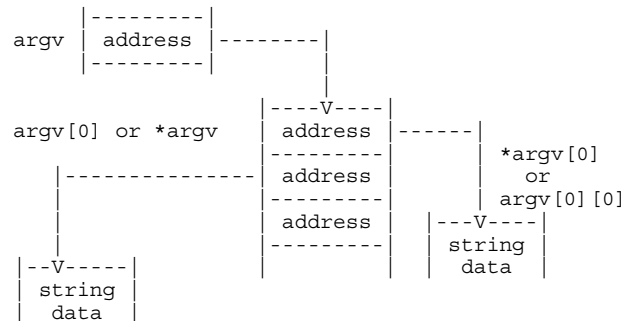
If you compile it and try it using the above invocation, you should see the - printed. (Try it with different argument values and different numbers of arguments.) Of what value is this?

**Listing 1**

```
main (argc, argv)
int argc;
char **argv;
{
    if (argc < 2) {
        printf("Error - no parameter was
            given!");
        exit(1);
    }
    ++argv;
    printf("%c\n", argv[0][0]);
}
```

Well, actually this program might be used as part of a larger program and the argv[0][0] could be used to test for a "switch" such as + or - in front of a parameter. In the example invocation above, I included the myfile/dat parameter to suggest some possibilities for you to ponder!

The argv function parameter, as mentioned in Part II, is a pointer to an array of pointers. Here is a list of possible ways or forms in which you might see it used: argv, \*argv, argv[n], \*argv[0], (\*argv)[0], argv[0][0]. Having some trouble "visualizing" what each represents? Look at a possible storage map (chart) of argv:



I hope the above map will be of some aid. Try to fit into the above map all of the ways of using the function argument argv. Enough of this for awhile! Let's switch topics and introduce structures.

A nice feature for a language is the ability to group variables of different types and treat them as one. This grouping of variables, called a structure in C, is called a record in other computer languages. Listing 2 is an example of the declaration of a sample C structure.

The struct keyword is used to declare a structure. An optional name or tag can follow the struct keyword. In this example, I used the tag of payroll. The tag names the structure and can be used as a shorthand method for the complete structure declaration. For example, to declare two more structures of type payroll, it might be done as follows:

```
struct payroll person1, person2;
```

The variables declared in the structure are referred to as members. Structure members or tags can have the same name as other simple variables. The C compiler can tell them apart due to the way they are used. Members of a structure are referenced as follows:

```
structure-name.member
```

The "." is called the structure operator. It connects the structure name to a member name. More will be said about structures in the next part!

Now, as promised, Table I is a list of many of the most common errors found in a C program. Keep these "mistakes" in mind as you code in C. Looking out for these pitfalls will help you design a more "bug free" program.

As you read some of the above most frequent C coding errors didn't you say to yourself, "Yes, I have done that before."? If you did, you are not alone! Some of these errors are caught by the C compiler, many are not. Another program for detecting possible errors in C code is called LINT. It typically better enforces the rules of C and reports more possible errors than does the C compiler.

In the next part, the C programming environment will be discussed along with many functions which are part of (or should be part of) the "standard" library. Structures will also be covered in more detail. Practice your C coding techniques until then!

## Listing 2

```
struct payroll { char name[30];
                int age;
                char sex;
                int pay;
                }
```

## Table 1 - C Language Pitfalls

- Using = instead of == in an if statement
- Thinking arrays start at index 1 instead of 0
- Unclosed braces or brackets
- Forgetting a ;
- Using / instead of \
- "Off by one" errors in looping or array indexing
- Declaring function arguments after the {
- Forgetting the precedence of operators
- Thinking C has built in string comparisons
- Using ' instead of "
- Using ( ) instead of [ ]
- Function arguments placed in the wrong order
- Not reserving an array element for the terminal \0
- Forgetting about "side effects"

# LDOS: How it works - an introduction to COMM and LCOMM

Joseph J. Kyle-DiPietropaolo

LCOMM (on LDOS 5.1.x) and COMM (on TRSDOS 6.x) are both very powerful communications packages, but few people use them to their full advantage. Admittedly, the manual(s) are a bit terse in regards to these packages, but with a little help everyone should be able to use these utilities. If you need any help in regards to physically connecting your modem to your computer, contact your vendor for the proper cables and instructions.

Getting started-- First, type the commands below at DOS ready:

```
Under TRSDOS 6.x:
SET *CL COM/DVR<enter>
SETCOM (DTR,W=8,P=N)<enter>
COMM *CL<enter>
```

```
Under LDOS 5.1.x:
SET *KI KI (T,J)<enter>
SET *CL RS232T
(DTR,W=8,P=N)<enter>
LCOMM *CL<enter>
```

Note that <enter> means to press the "ENTER" key. If you have a Model 1, on the second line type: "SET \*CL RS232R<enter>" instead. Your system is now set up for RS232 communication at 300 baud, the most common mode when using a modem over the telephone line. This setup should work for most modems, including the Radio Shack modems and their DC Modem II.

When you enter COMM/LCOMM in this manner, you are immediately in the terminal mode. Anything you type will be sent to the modem, and whatever is received will be displayed on your screen. If you use a JCL (Job Control Language) procedure to set this up automatically for you, don't forget to add a line reading "//STOP" as the last line of the JCL file. If you don't, control will be returned to DOS if you attempt to execute certain COMM/LCOMM commands.

Now to actually communicate. Let's take CompuServe as an example. Dial the phone number for your local CompuServe node. If you have a "smart"-type modem, refer to your operators manual for dialing instructions. Once the phone number has been dialed, put your modem on-line. For most "dumb" modems, this will mean flipping the switch on, or placing the phone handset into the acoustic cups. Most "smart"-type modems automatically enter the on-line mode after dialing.

When the "CD" or carrier detect light comes on, type a <control>-C. On the Models I and III, this is the combination <Shift><Down Arrow> (meaning "control") and (while still holding them down) then a <C>. On the Model 4, you have a <control> key. Depress this, then (while still holding it down) type a <C>. In either case, CompuServe should respond with "User Id:". This is your first prompt. Type in your user number (provided in your sign-up package -- obtain from Radio

Shack). For example, the user number for the people here at LSI is 76703,437. We would type in "76703,437".

The semi-colon is very important. Until you tell CompuServe otherwise, they assume that you are using what is called a "Videotex-compatible terminal". When using LCOMM or COMM, this will cause all sorts of nasty things to appear on your video display. The semi-colon prevents this from happening. Once you are logged on, if you change your terminal type (tell CompuServe that you have an "other"-type terminal) this will not happen and you may omit the semi-colon.

CompuServe will now respond with "Password:". Type your password from the sign-up package and hit <enter>. After a few seconds, CompuServe will respond

with its first menu. Congratulations! You have successfully communicated!

Now we know how to establish a communications channel from our computer, let's look at how we give instructions to COMM/LCOMM. All commands begin with a <clear><keystroke> or <clear><shift><keystroke> sequence. For instance, pressing <clear> and holding it while you press <8> will display the COMM/LCOMM command menu. The <clear> key is used as a second type of "control" key, one that has special meaning to COMM/LCOMM (and many other DOS utilities).

But-- what does all this menu information represent? Let's take a look at an example. \*PR stands for the "logical"

printer device, and the word "ON" is relatively self-explanatory. If you hit <clear><3> and then <clear><:;>, (the keystrokes that represent "\*PR" and "ON" respectively) the \*PR device will be "turned on". Typing <clear><8> will re-display the menu. Do you see the difference? There is now a "\*" below the \*PR device, indicating that it is "ON". Now, any characters received by your computer will be sent to the printer in addition to the video display.

So far, we have covered the initial "set-up" phase of COMM/LCOMM. In future installments, we will cover more advanced features of these utilities, such as file uploading and downloading.

## Les information - faster file access

### Les Mikesell

When accessing data files and devices in machine language, there are many different techniques that can be used. The operating system @GET and @PUT calls (these are called the "Byte 1/0 calls" because they move one byte at a time) are easy to use, and allow use of either devices or files. See past issues of the LDOS Quarterly (volume 2, numbers 2 and 3) for more information on this method.

@GET and @PUT are convenient, but there is a speed penalty as compared to full sector operations. The single byte operations can only access one sector per disk revolution, while full sector operations may be able to handle an entire track in two or three revolutions (depending on the disk type and the processing time needed between sectors). Thus, it may be possible to speed up disk operations considerably by buffering as much as possible in memory using full sector disk I/O. One significant drawback to this approach is that the program then becomes responsible for observing or setting the proper end-of-file offset when the data file does not end on a sector boundary.

The sector interleave on floppy disks is designed to allow just enough time to move a sector of data in or out of the file buffer

before the next consecutive sector passes under the read/write head. Any additional processing at this point will usually cause the next sector to be missed, and necessitate a wait until the next revolution of the disk. When using a hard drive or MemDISK, the interleave factor is not critical, but programs will still benefit from the reduced overhead of full sector operations.

After opening a file, its size can be determined from the contents of the open FCB (file control block). The ending record number (ERN) is stored at FCB+12 & 13, and the end of file offset (EOF) is at FCB+8. If the ERN is 00, the file is empty. Otherwise, ERN-1 is the number of full sectors in the file, and EOF is the number of bytes included in the ending sector (where 0 = 256). Thus EOF-1 is the offset of the last valid byte in the file buffer when the last sector is read. This may be easier to remember by keeping in mind that these three bytes are always maintained as a pointer to the next record to write to extend the file.

For many operations, a file may be read into memory until the DOS error 1CH (end of file) or 1DH (past end of file) occurs, then the buffer pointer adjusted back to the correct byte offset in the previous sector. However, it may sometimes be necessary to

# LSI Journal

determine if the current sector contains any data past the end of file before reading the next sector (which would return the EOF error). In this case, the next record number (NRN) at FCB+10 & 11 may be compared to the ERN after the read. If the NRN is the same as the ERN, the sector just read contains the end-of-file.

When writing sequential data using the full sector operations, it is necessary to update the EOF byte before closing the file. If @POSN has been used, it is necessary to update the ERN, since the system will then consider the file to be "random-access" and update the length only if it has been extended. Moving the NRN into the ERN in the FCB will set the current position as the end of file even if the previous ERN was larger.

Listing 1, a program to either add line-feed characters after carriage returns, or remove line-feeds, demonstrates some of the techniques of handling byte data with the DOS sector operations. Note that the input routine simply moves the buffer pointer in the FCB for each sector read rather than moving the data from the file buffer.

The output routines are a little slower, and will miss the disk interleave on a Model I or III with the standard CPU speed. This could be avoided by processing the data into a larger buffer space, then writing several sectors at once. Pre-allocating the disk space before the write would increase the speed also, by reducing the number of times the system has to go to the disk directory. This may be done simply by using @POSN and @WRITE to write (anything) to the last sector of the output file, then re-position to record 0.

This program may also be assembled for use with the TRSDOS/LDOS 6.x system by deleting the beginning lines between the asterisks, and including the standard 6.x header. For TRSDOS/LDOS 6.x operation, delete lines 100 to 410 and insert the code from Listing 2.

## Listing 1 LDOS 5.1 version

```

00100 ;*****
00110 ;This part is for LDOS 5.1
00120 ;operating system entry points:
00130 @ABORT EQU 4030H
00140 @CLOSE EQU 4428H
00150 @DSPLY EQU 4467H
00160 @ERROR EQU 4409H
00170 @EXIT EQU 402DH
00180 @FSPEC EQU 441CH
00190 @INIT EQU 4420H
00200 @KEYIN EQU 0040H
00210 @OPEN EQU 4424H
00220 @READ EQU 4436H
00230 @WRITE EQU 4439H
00240 HIGH$ EQU 4411H ;Model 3
00250 HIGH1 EQU 4049H ;Model 1
00260 ;
00270 ORG 5200H
00280 ;Put file buffer first to force location
00290 ;on memory page boundary
00300 BUFR1 DS 256
00310 BUFR2 DS 256
00320 ;Machine specific code:
00330 BEGIN: LD A,(125H) ;Check mod1/3 ROM
00340 CP 'I'
00350 LD HL,(HIGH$) ;Mod 3 location
00360 JR Z,SETHI ;Go if mod 3
00370 LD HL,(HIGH1) ;Mod I location
00380 SETHI: LD (MYMEM),HL ;Store correct value
00390 ;
00400 ;End of LDOS 5.1 specific code
00410 ;*****
00420 ;
00430 ;Special chars
00440 BTX EQU 03H
00450 CR EQU 0DH
00460 LF EQU 0AH
00470 ;
00480 ;
00490 ;FCB offset definitions
00500 BUFRLO EQU 3 ;Buffer address
00510 BUFRHI EQU 4
00520 ERNHI EQU 13 ;Ending record #
00530 ERNLO EQU 12
00540 EOF EQU 8 ;Offset of last byte
00550 NRNLO EQU 10 ;Next record pointer
00560 NRNHI EQU 11
00570 ;
00580 START: LD HL,LOGON ;Log on
00590 CALL @DSPLY
00600 DISKIN: LD HL,MSG1 ;Prompt for input
file
00610 CALL INPFSP ;Get answer
00620 LD DE,FCB1
00630 CALL @FSPEC ;Move filename
00640 LD B,0 ;LRL=0 (256)
00650 LD HL,BUFR1 ;=>disk buffer
00660 CALL @OPEN ;Open the file
00670 JR Z,GOTINP ;Go if successful
00680 CALL SHOERR ;Else report error
00690 JR DISKIN ;And ask again
00700 ;
00710 ;File is open, check if it contains any records
00720 GOTINP: LD HL,(FCB1+ERNLO)
00730 LD A,H ;Is ending record 0?
00740 OR L
00750 JP NZ,ASK ;Go if file has data
00760 LD HL,NODAT ;Else report empty
file
00770 CALL @DSPLY
00780 JP @ABORT ;And quit
00790 ;
00800 ASK: LD HL,MSG2 ;Prompt for output
file
00810 CALL INPFSP ;Get answer
00820 LD DE,FCB2
00830 CALL @FSPEC ;Move filename
00840 LD B,0
00850 LD HL,BUFR2
00860 CALL @INIT ;Create file
00870 JR Z,ASK1A ;Continue if good
init
00880 CALL SHOERR ;Else report error

```

```

00890      JR      ASK          ;And re-prompt          01400 FINIS:  EX      DE,HL          ;Pointer to DE
00900 ;
00910 ;Set up for add or remove LF's          01410          LD      BC,BUFFER      ;START OF DATA AREA
00920 ASK1A: LD      HL,MSG3      ;Remove linefeeds?          01420 ;
00930          LD      DE,RLF
00940          CALL   SGETYN      ;Prompt, set flag          01430 ; Memory is loaded, check for changes
00950          JR      Z,AGAIN      ;If yes, skip 2nd          01440 ; BC=>current char, DE =>end+1
prompt
00960 ;
00970 ASK1B: LD      HL,MSG4      ;Add linefeeds?          01450 ACHAR:  LD      A,D          ;Is this the end of
00980          LD      DE,ALF
00990          CALL   SGETYN      ;Prompt, set flag          the buffer?
01000          JR      NZ,ASK1A      ;Must do one or the          01460          CP      B
other
01010 ;
01020 ;Read file into memory - set (MORE)=0FFH if it          01470          JP      NZ,NOTEND      ;Go if not
doesn't fit
01030 AGAIN:  LD      HL,(MYMEM)      ;=>end of free space          01480          LD      A,E
01040          LD      BC,BUFFER      ;=>working buffer          01490          CP      C
start
01050          OR      A
01060          SBC   HL,BC
01070          LD      B,H          ;# of sectors that          01500          JR      Z,CHECKM      ;End, check for more
will fit
01080          LD      HL,BUFFER      ;Start of buffer
01090          LD      DE,FCB1      ;=>input file buffer
01100 RLOOP:  LD      (FCB1+BUFRLO),HL      ;Set load          01510 NOTEND: LD      A,(BC)          ;Get a character
address
01110          CALL   @READ          ;Get a sector
01120          JP      NZ,CKEND      ;End or error
01130          INC   H          ;Bump ptr for next
01140          DJNZ  RLOOP          ;Stop if memory is          01520          INC   BC          ;bump pointer
full
01150 ;Now check to see if the last sector contained the          01530          CP      LF          ;a linefeed?
EOF
01160 ;To be sure all data loaded belongs in file          01540          JP      NZ,NTLF      ;go if not LF
01170          PUSH  HL          ;Save current posn          01550          LD      A,(RLF)          ;Removing line
01180          LD      HL,(FCB1-NRNLO) ;Check if this is          feeds?
01190          LD      DE,(FCB1+ERNLO) ;The last sector          01560          OR      A
01200          OR      A
01210          SBC   HL, DE          ;Is NRN=ERN?          01570          JR      Z,DOLF      ;Not removing line
01220          POP   HL          ;Get end ptr
01230          JR      Z,ISEOF      ;Go if this is EOF          01580          JR      ACHAR          ;Skip LF
01240          LD      A,0FFH      ;Set flag for more          01590 NTLF:  CP      CA          ; carriage RET?
input
01250          LD      (MORE),A
01260          JR      FINIS          ;And start
processing
01270 ;
01280 CKEND:  CP      1CH          ;EOF?          01600          JR      NZ,WRBYT      ;Write if not CR
01290          JR      Z,ISEOF
01300          CP      1DH          ;Or past EOF?
01310          JP      NZ,DOSERR      ;Quit if other error          01610          CALL   PUTTER          ;Write CR
01320 ISEOF:  LD      A,(FCB1+EOF) ;Get EOF offset byte          01620          LD      A,(ALF)          ;Adding line feeds?
01330          DEC   A          ;Point to end byte          01630          OR      A          ;Test
01340          LD      E,A
01350          LD      D,0          ;DE=offset of EOF          01640          JR      Z,ACHAR      ;Skip if not wanted
01360          DEC   H          ;Back up to last          01650 DOLF:  LD      A,LF          ;Load the line feed
full sector
01370          ADD   HL,DE          ;Add in contents of          01660 WRBYT: CALL   PUTTER          ;Write char to file
last sector
01380          INC   HL          ;HL=>1 past buffer          01670          JP      ACHAR          ;Loop through buffer
01390 ;
01680 ;
01690 ;End of buffer, is file done?
01700 CHECKM: LD      HL,MORE          ;Is there more to
read?
01710          LD      A,(HL)          ;Zero if done
01720          LD      (HL),0          ;Set for next time
01730          OR      A          ;Set Z if done
01740          JP      NZ,AGAIN      ;Loop till finished
with files
01750 ;
01760 ;Finished, close the output file
01770          CALL   LSTSEC          ;Flush buffer
01780          LD      DE,FCB2          ;=>output FCB
01790          CALL   @CLOSE          ;Close output file
01800          JP      NZ,DOSERR      ;Go if error
01810          LD      HL,MSG5          ;Else report
completion
01820          CALL   @DSPLY
01830          JP      @EXIT          ;And go back to DOS
01840 ;
01850 ;Set flag according to 'Y' response
01860 SGETYN:  PUSH  DE          ;Save flag address
01870          CALL   @DSPLY          ;Issue prompt
01880          LD      HL,YN$          ;Add Y/N?
01890          CALL   @DSPLY
01900          LD      B,1          ;Max input wanted
01910          CALL   KEYIN          ;Get 1 char answer
01920          AND   5FH          ;Force upper case
01930          CP      'Y'          ;Is it Y?

```

# LSI Journal

```

01940      POP      HL              ;Get flag address
01950      RET       NZ              ;Return if not "Y"
01960      LD        (HL),0FFH      ;Set flag if "Y"
response
01970      RET
01980 ;Get a filename:
01990 INPFSP: CALL  @DSPLY          ;Display prompt
02000      LD        B,1FH          ;Set max length
02010 ;Get user input
02020 KEYIN: LD      HL,INBFR       ;=>buffer to receive
input
02030      CALL     @KEYIN          ;Get input
02040      JP       C,@ABORT        ;Quit if BREAK
pressed
02050      LD        A,(HL)         ;Else pick up 1st
char
02060      RET
02070 ;
02080 ;Add a byte to disk buffer/write if full
02090 ;Note that disk buffer must end on XXFFH boundary
02100 ;So the INC L will set the Z flag when sector
02110 ;Buffer is full. (PUTPTR) is a pointer to the next
char
02120 ;Position in the sector buffer
02130 PUTTER LD      HL,(PUTPTR)    ;Point to buf pos
02140      LD        (HL),A         ;Move to buffer
02150      INC      L               ;Bump buffer ptr
02160      LD        (PUTPTR),HL    ;Save for next
02170      RET       NZ           ;If not full
02180 ;Write a physical sector to disk
02190 WSEC:  PUSH   DE
02200      LD        DE,FCB2
02210      CALL     @WRITE        ;Write
02220      POP      DE
02230      RET       Z             ;That was easy
02240 DOSERR: CALL  SHOERR          ;Report any error
02250      JP       @ABORT        ;And quit
02260 ;
02270 SHOERR: OR     OCOH          ;Mask for short
msg.,ret
02280      JP       @ERROR
02290 ;
02300 ;Fill remainder of sector buffer with 00's
02310 ;Set the EOF offset, and flag DOS to reset the
02320 ;File's ERN to the current sector
02330 LSTSEC: LD      HL,(PUTPTR)   ;Get posn in buffer
02340      LD        A,L           ;Did last write
02350      OR        A             ;Hit sector end'
02360      JR       Z,SETEOF        ;Finished on sec
boundary
02370      PUSH   AF              ;Save last char I
02380      XOR     A              ;Set A=0
02390 FLSEC: LD      (HL),A        ;Zero remaining
buffer
02400      INC      L
02410      JR       NZ,FLSEC       ;Pad sec w/nulls
02420      CALL     WSEC           ;Write the last
sector
02430      POP      AF           ;EOF byte to A
02440 ;
02450 SETEOF: LD      (FCB2+8),A    ;Set EOF offset
02460 ;Note: this step is actually not necessary unless
@POSN called
02470      LD        HL,(FCB2+NRNLO) ;Put NEXT record no.
02480      LD        (FCB2+ERNLO),HL ;Into ENDING record
no.
02490      RET
02500 ;
02510 ;
02520 ;END OF PROGRAM AREA
02530 ;ASCII DATA
02540 LOGON:  DB      LF,'TEXT FILE PROCESSOR ',LF,CR
02550 MSG1:   DB      'Input Filespec?'
02560 MSG2:   DB      'Output Filespec?'
02570 MSG3:   DB      LF,'Remove line-feed characters',ETX
02580 MSG4:   DB      'Add line-feed characters',ETX
02590 MSG5:   DB      LF,'File output completed -',CR
02600 NODAT: DB      'Input file is empty!',CR
02610 YN$:   DB      '(Y/N) ? ',ETX
02620 ;
02630 ;BUFFERS & POINTERS
02640 RLF:    DB      0             ;Remove LF flag
02650 ALF:    DB      0             ;Add LF flag
02660 MORE:   DB      0             ;More input flag
02670 EOFFLG: DB      0             ;Last sector flag
02680 MYMEM:  DW      0             ;HIGHS pointer
02690 PUTPTR: DW      BUFR2        ;Posn in output
buffer
02700 GETPTR: DW      BUFR1,255    ;Posn (-1) in input
buffer
02710 FCB1:   DS      32           ;File FCBs
02720 FCB2:   DS      32
02730 INBFR: DS      40           ;KB input buffer
02740 BUFFER  EQU    $             ;START OF DATA
BUFFER
02750      END      BEGIN

```

## Listing 2 LDOS 6.x version

```

00100 ;*****
00110 ; Header to convert programs with 5.1 CALLS to 6.x
00120 ;
00130      ORG      2600H
00140 BUFR1: DS      256           ;put on page
boundary
00150 BUFR2: DS      256
00160 ;
00170 BEGIN: DI
00180      LD      (STACK),SP      ;save SP at entry
00190      PUSH   HL              ;Save ptr to CMD
buffer
00200      LD      HL,0
00210      LD      A,103          ;Disable break
vectoring
00220      RST    40
00230      EI
00240      LD      HL,0           ;get HIGH$
00250      LD      B,L           ;B=0
00260      LD      A,100
00270      RST    40
00280      LD      (MYMEM),HL     ;Store for later
00290      LD      A,101          ;set up IY

```

```

00300      RST      40          ;pointing to flag      00600      RET      Z
table
00310      PUSH     IY          ;trans to DE      00610      CP      42          ;LRL error
00320      POP      DE
00330      LD       HL, 'S'-'A' ;SFLAGS offset    00630 @READ  LD      A, 67
00340      ADD      HL, DE
00350      LD       (SFLAG), HL ;store away      00640      RST      40
00360      POP      HL          ;restore ptr to CMD 00650      RET
line
00370      CALL     START       ;execute program  00660 @WRITE LD      A, 75
00380 ; set up for exit..... 00670      RST      40
00390 @EXIT: LD       HL, 0      ;HL=0 if no error 00680      RET
00400 QUIT$: LD       SP, $-$    ;restore SP      00690 @CLOSE LD      A, 60
00410 STACK EQU      $-2
00420      RET
00430 @ABORT: LD      HL, -1     ;non-zero if abort 00700      RST      40
00440      JR       QUIT$
00450 ;use SVC's for operating system functions 00710      RET
00460 ;create label for each CALL used by program 00720 @KEYIN LD      C, 0
00470 @FSPEC LD      A, 78      ;SVC #            00730      LD      A, 9
00480      RST      40
00490      RET
00509 @OPEN  PUSH     HL          ;set inhibit bit  00740      RST      40
00510      LD       HL, $-$
00520 SFLAG EQU      $-2
00530      SET     0, (HL)       ;ignore LRL errors 00750      RET
00540      POP      HL
00550      LD       A, 59
00560      RST      40
00570      RET
00580 @INIT  LD       A, 58
00590      RST      40
00600      RET      Z
00610      CP      42          ;LRL error
00620      RET
00630 @READ  LD      A, 67
00640      RST      40
00650      RET
00660 @WRITE LD      A, 75
00670      RST      40
00680      RET
00690 @CLOSE LD      A, 60
00700      RST      40
00710      RET
00720 @KEYIN LD      C, 0
00730      LD      A, 9
00740      RST      40
00750      RET
00760 @ERROR PUSH     BC          ;save BC
00770      LD      C, A          ;trans error # to C
00780      LD      A, 26        ;display error
00790      RST      40
00800      POP     BC          ;restore BC
00810      RET
00820 @DSPLY LD      A, 10
00830      RST      40
00840      RET      Z          ;can return an error
00850      LD      H, 0          ;due to device
routing
00860      LD      L, A
00870      OR      0C0H
00880      CALL     @ERROR
00890      JP      QUIT$
00900 ;

```

# Index

Index to LDOS Quarterly/LSI Journal, issues 1,1 to 2,4

Scott Loomer

This is the subject index to the *LDOS Quarterly* and *LSI Journal* for the period from its inception in July, 1981 to the October, 1983 issue. Back issues for this period are still available, some as individual issues and some as sets. Contact LSI at (414) 355-5454, or write to Logical Systems, Inc., 8970 N. 55th St., Milwaukee, WI 53223. Volume 2, Number 5 (Jan.'84) is also available. It includes an expanded version of this index (which, by the way, was originally prepared by Scott Loomer).

## Subject/Author Vol/No/Page

"Active Variable Dump for LBASIC" - Alan Moyer V2N3P15 V2N4P38  
 "Alcor Pascal" - Scott Loomer V2N1P18  
 Allocation, disk V2N4P08  
 AM Electronics, disk controller V1N1P03  
 "APL\*PLUS/80 A System Overview" - Daniel Lofy & Lee Rice V2N1P09

"Article, An" Charlie Butler V1N4P40  
 "ASCII File Listing Utility for The BASIC Answer" - Jeffrey Brenton V2N3P19  
 Assembly language basic concepts V2N2P58 V2N3P46 V2N4P40  
 Assembly language patching V1N6P38 V2N1P46  
 Assembly language programing tips V1N5P14 V1N6P64  
 "At Large" - Earl Terwilliger V1N6P50  
 "Automatic Chaining with JCL" - Jim Kyle V2N4P52  
 "BASIC and File Structure - A Beginner's View" - Wes Goodnough V1N6P20  
 "BASIC Concepts - The RUN,V Command" - Dick Konop V1N6P69  
 "Beta Tester, I was an LDOS" - Tim Daneliuk V1N3P16  
 Byte I/O V1N3P38 V1N6P76 V2N2P52  
     under LDOS 6.0 V2N3P56  
 'C' graphics V2N4P16  
 " 'C' language, The" - Earl Terwilliger:  
     general introduction V2N1P15



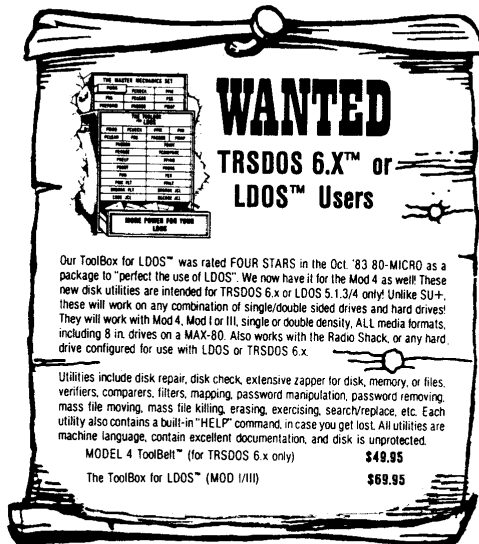
# LSI Journal

functions, variables, constants, expressions		V2N2P35	"Interrupts and SVC's in Fortran, Using" - J. Bender	V2N4P19
operators		V2N3P39	JCL chaining	V2N4P52
logic, control & flow		V2N4P34	"JCL Corner" - Chuck Jensen:	
"Card' Utility" - Paul Tonini		V2N2P18	clear screen	V1N1P03
"Case of Mis-Allocation, A" - Bill Schroeder		V2N4P08	file copying	V1N2P30
Case mode indicator		V2N2P11	compilation macros	V1N3P51
Changing operating systems		V1N4P40	hex codes, more compilation macros	V1N4P46
"Clock Speedup Kits with LDOS, Using" - Tim Mann		V1N1P13	keyboard input	V1N5P34
Cobol		V1N6P32	logical operators, keyboard filters, LScript patch	V1N6P72
"Color Comes to the TRS-80's" - Scott Loomer		V2N3P28	creating and defining JCL procedures	V2N1P54
"Communicating Micro, The" - Gordon Thompson	V1N6P12	V2N1P28	backups	V2N2P49
Communications	V1N5P40	V1N6P83	JCL With FORTRAN and Cobol	V1N6P32
"Communication Host" - James Bruckart		V2N1P35	LBASIC:	
"Confessions of a Machine Language Addict" - Ray Pelzer		V1N6P38	active variable dump	V2N3P15 V2N4P38
Configuring with non-relocatable code		V2N4P50	"CMD"O" implementation and possible uses"	V1N4P15
Customer service tips		V2N1P07	default extension patch	V1N5P19
Cylinder term explained		V1N1P04	file structure	V1N6P20
DAMs, Old		V1N1P19	notes (items missing from 1st printing of the LDOS manual)	V1N1P09
"Data Address Marks" - Roy Soltoff		V1N1P05	RUN,V command	V1N6P69
Date conversions		V2N4P60	USR routines (relocating to high memory)	V1N3P47
"Device I/O and Independence, LDOS" - Roy Soltoff		V1N3P38	LDOS 5.1.4, New features	V2N4P03
Disk allocation schemes		V2N4P08	LDOS 6.0	
Disk Controller, AM Electronics		V1N1P03	announced	V2N2P46
Disk byte I/O	V1N3P38 V1N6P76	V2N2P52	licensing	V2N4P07
Disk drive first access delay		V2N4P63	technical manual	V2N4P07
"Disk Drive Control Linkages, LDOS" - Bob Hawker		V1N6P43	patches	V2N4P64
Disk drives (8") on the Model III		V2N3P08	"LDOS: How it Works" - Joe Kyle-DiPietropaolo:	
Disk drives, non-Radio Shack		V2N3P55	PATCH utility	V2N1P52
Disk drive poll		V2N2P27	REPAIR, CONV and COPY238 utilities	V2N2P44
Disk drive select time hardware fix (Mod 1)		V1N3P18	non-Radio Shack disk drives	V2N3P55
"Disk I/O in Assembler" - Doug Kennedy		V2N2P55	configuring with non-relocatable code	V2N4P50
Disk speed (300 RPM delays)		V1N4P11	"Les Information" - Les Mikesell:	
"Double Sided Drives with LDOS" - Tim Mann & Roy Soltoff	V1N2P37	V1N3P56	communications	V1N5P40
"Easy LScript" James Bruckart		V2N2P22	RS232 drivers	V1N6P83
"Easy VisiCalc" James Bruckart		V2N3P23	SYSTEM(FAST) and SLOW commands	V2N1P58
"EDAS IV 'Z' Command" - Earl Terwilliger		V2N1P37	byte 1/0	V2N2P52
Electric Webster with Newscrip, LDOS and Sole		V1N6P55	byte 1/0 under LDOS 6.0, CTLP/FLT	V2N3P56
"Electronic Inbasket, The" - Gordon Thompson		V2N4P15	@PARAM under LDOS 6.0	V2N4P61
"ELSIE - The Contented Compiler" - Jim Frimmel		V1N3P21	"Let Us Assemble" - Rich Hilliard:	
Epson MX-80 tips		V1N5P18	the basics	V2N2P58
".... er ...." - Earle Robinson:			using DEBUG, sorts	V2N3P46
assembly language efficiency		V1N6P64	number base conversion	V2N4P40
printers, TBA, UTILZAP		V2N1P39	"Library, The" - Earle Robinson	V1N4P28
printers, word processors, MNet, 'C', LDOS 6.0		V2N2P26	Library commands;	
'C' book, SS drivers, Model 4, PROMPT/CMD		V2N3P33	LOAD documentation correction	V2N1P47
UNIX, IBM PC, Telex		V2N4P28	SYSTEM(FAST)	V1N1P13 V2N1P58
Expansion interface		V1N4P39	SYSTEM(SLOW)	V2N1P58
"Fast Graphics for 'LC' " - Scott Loomer		V2N4P16	Limited backup policy explained - Bill Schroeder	V2N3P04 V2N4P06
File listing utility for TBA		V2N3P19	"Linking to LDOS in Assembly" - Roy Soltoff	V1N1P12
File structure, BASIC		V1N6P20	"LISP Implementations for the Z-80" Lee Rice & Daniel Lofy	V1N6P26
Filter linkage		V1N3P47	Listing utility for TBA	V2N3P19
"Fortran, Cobol and LDOS JCL" - Glen Rathke		V1N6P32	LScript made easy	V2N2P22
Fortran With interrupts and SVC's		V2N4P19	"LScript Patches to Add Versatility" - Scott Lower	V1N6P45 V2N1P47
"Greek to Me, LDOS - It's" - Charles Knight		V1N5P15	Load module structure	V1N4P42
"Hayes Smartmodem, LDOS and a" - John Mullin		V1N6P34	Lower case lock	V1N4P11
High memory, avoiding memory conflicts		V1N4P13	LSI Journal submission and subscription policies	V2N4P02
High memory module header		V1N3P46	MAX-80 LDOS described	V2N1P50
Host, communications		V2N1P35	"MAX-80 Memory Map" - Chuck Jensen	V2N4P58
"Inside the Expansion Interface" - Earle Robinson		V1N4P39	Magazines	V2N3P35

Manual story, with corrections to 1st edition	V1N1P16	Scriptit Dictionary (Radio Shack) - Tim Daneliuk	V2N2P30
Memory map	V2N1P34	Structured BASIC Translator (Acorn) - Sue Ratkowski	V1N4P21
Minimum configuration disk	V1N2P29	The BASIC Answer (LSI) E. Cheatham	V2N3P17
"Mixing Newsprint, Electric Webster, LDOS and Sole" Jerry Latham	V1N6P55	The BASIC Answer (LSI) Tim Daneliuk	V2N1P46
"My BASIC Answer" - E. Cheatham (TBA review)	V2N3P17	UOLISP (Far West) - Lee Rice & Daniel Lofy	V1N6P30
"Newsprint 7.0 and REFLEX" - Gordon Thompson	V2N1P28	Routing a device	V1N3P38
"Newsprint and The BASIC Answer" - Jerry Latham	V2N2P20	"Roy's Technical Corner" Roy Soltoff:	
"New Version - EDAS IV, A" - Marc Leager	V1N6P09	load module structure	VIN4P42
Number base conversion	V2N4P40	task processor	V1N5P20 V1N6P76
Old DAMs (see also Data Address Marks)	V1N5P19	error handling during byte 1/0, @CKDRV, KFLAGS, @ICNFG, @KITSK	V1N6P76
Parity errors	V1N1P04	LDOS 6.0	V2N2P46
-- PARITY = ODD - Tim Daneliuk:		RS-232 drivers	V1N6P83
introduction, DATAENTR 200 & ISAM 200	V1N4P17	"Running 8-Inch Drives on the Model III Under LDOS" - Peter Simon	V2N3P08
tips for better programming, DISCATER, Filter Disk	V1N5P11	Sector I/O	V2N2P55
BASF drives, drive poll, TAS, MODEM80, HEXSPELL II, Utilities, HELP	V1N6P59 V1N6P56	"SOLEFIX - Fix that GAT Error" - Erik Ruf	V2N2P14 V2N4P38
Tandy, MAX-80, AEROCOMP, Electric Webster, The BASIC Answer	V2N1P42	Speedup kits	V1N1P13
disk drive poll, gold plugs, LX-80, Proofreader, Scriptit Dict.	V2N2P27	Sysgen and type-ahead	V1N1P04
magazines	V2N3P35	System routines:	
rumors, LX-80 software compatibility, TRSDOS 6.0 software	V2N4P31	@ADTSK, @RMTSK, @KLTSK, @RPTSK - task proc.	V1N5P31 V1N6P76 V2N4P19
"Partitioned Data Sets, MISOSYS Announces" - Roy Soltoff	V1N3P23	@CKDRV - determining 5.1.2 vs. 5.1.3 for @CKDRV adjustment	V1N6P78
"Pascal 80, LDOS and" - D. Hill	V2N1P22	@CKDRV - moved	V1N5P06
Passwords for LDOS 5.1.x	V1N1P03	@CMNDI - command interpreter	V1N1P12
Patch, how to	V2N1P52	@CTL, @GET, @PUT - byte 1/0	V1N3P38
"PDS - Standard and Other Types of Uses" - Scott Loomer	V2N1P24 V2N3P45	DAYS - day of the week	V1N4P11
"Performing Date Conversions in BASIC" - Dick Konop	V2N4P60	@GET and @PUT under LDOS 6.0	V2N3P56
Printers	V2N1P39	@ICNFG configuration interfacing	V1N6P81
Profile III Plus with LD03	V2N4P55	@KITSK keyboard task	V1N6P82
Quarterly reader survey results	V2N2P06	KFLAG\$ keyboard scanner	V1N6P78
Radio Shack	V2N1P42 V2N3P03	"@PARAM, Using" - Roy Soltoff	V1N2P31
"Relocating Code for LBASIC USR Routines" - Chuck Jensen	V1N3P47	@PARAM under 6.0	V2N4P61
Reviews:		"@PARAM, @DSPLY, @EXIT and INBUFS for Everyone" David Vinzant	V2N2P23
Alternate Source (TAS) - Tim Daneliuk	V1N6P60	@RAMDIR documentation correction	V2N2P39
APL*PLUS/80 (STSC) - Daniel Lofy & Lee Rice	V2N1P09	SVC's and Fortran	V2N4P19
CHROMAtrs (South Shore Computer Concepts) - Scott Lower	V2N3P28	Tandy	V2N1P42 V2N3P03
DATAENTR200 & ISAM20V (Johnson Associates) - Tim Daneliuk	V1N4P18	Task processor	V1N5P20 V1N6P76
EDAS IV (MISOSYS) - Marc Leager	V1N6P09	"T-Timer, LDOS Supports the" - Roy Soltoff	V1N4P12 V1N5P18 V1N6P66
Electric Webster (Cornucopia) - Tim Daneliuk	V2N1P45	TRSDOS (Mod I) to LDOS (Mod III) transfer without REPAIR	V1N3P03
HELP (MISOSYS) - Tim Daneliuk	V1N6P63	TRSDOS (LDOS) 6.0 (see also LDOS 6.x)	V2N2P46
HEXSPELL II (Hexagon) - Tim Daneliuk	V1N6P61	Update policy explained - Bill Schroeder	V1N2P02 V2N4P64
LDOS Utilities (Powersoft) - Tim Daneliuk	V1N6P62	Upper case lock	V1N4P11
LISP (Supersoft) - Lee Rice & Daniel Lofy	V1N6P29	Users group directory	V2N1P06
MAX-80 (Lobo) - Tim Daneliuk	V2N1P44	Utilities:	
Microcomputer Math book (SAMS) - Earle Robinson	V1N5P10	CONV	V2N2P44
MODEM80 (LSI) Tim Daneliuk	V1N6P60	COPY238	V2N2P44
Pascal (Alcor) Scott Lower	V2N1P18	PATCH	V2N1P52
Pascal-80 IN ew Classics) - D. Hill	V2N1P22	REPAIR	V2N2P44
PDS (MISOSYS) Scott Lower	V2N1P24	Version number explanation	V1N2P02
Printers - Earle Robinson	V2N1P39	VisiCalc made easy	V2N3P23
Profile III (Radio Shack) - Sam Goldberg	V2N3P25	"VisiCalc with LDOS, Using" - Roy Soltoff	V1N2P20
Proofreader (Aspen Software) - Tim Daneliuk	V2N2P30		

# Much Needed Software from PowerSOFT!

For Users of LDOS™ and TRSDOS 6.x™



**WANTED**  
TRSDOS 6.X™ or  
LDOS™ Users

Our ToolBox for LDOS™ was rated FOUR STARS in the Oct. '83 80-MICRO as a package to "perfect the use of LDOS". We now have it for the Mod 4 as well! These new disk utilities are intended for TRSDOS 6.x or LDOS 5.1.3/4 only! Unlike SU+, these will work on any combination of single/double sided drives and hard drives! They will work with Mod 4, Mod I or III, single or double density, ALL media formats, including 8 in. drives on a MAX-80. Also works with the Radio Shack, or any hard drive configured for use with LDOS or TRSDOS 6.x.

Utilities include disk repair, disk check, extensive zipper for disk, memory, or files, verifiers, compilers, fillers, mapping, password manipulation, password removing, mass file moving, mass file killing, erasing, exercising, search/replace, etc. Each utility also contains a built-in "HELP" command, in case you get lost. All utilities are machine language, contain excellent documentation, and disk is unprotected.

MODEL 4 ToolBelt™ (for TRSDOS 6.x only) **\$49.95**  
The ToolBox for LDOS™ (MOD I/III) **\$69.95**



**WANTED**  
Your BASIC  
**Impakt!**

DO YOU USE DISK BASIC? WHAT DOS? TRSDOS? LDOS?

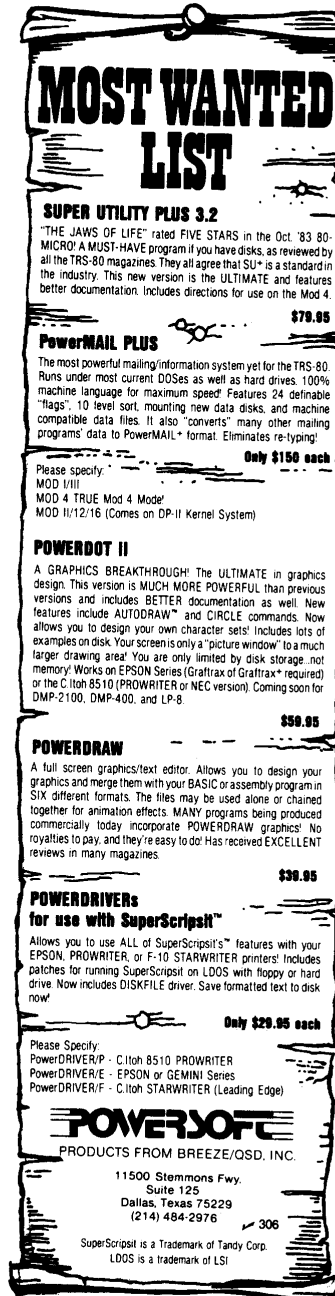
How'd you like with a single command from BASIC to: search and/or replace a particular keyword or string, remove remarks, remove extraneous spaces, remove extraneous colons, remove GOTOs following THENs, compress a program, decompress a program, renumber, trace a program without destroying the screen display either by line number or by step, set breakpoints in trace mode, join lines, copy lines from one part of the program to another with automatic renumber, move blocks of lines, check lines for validity, display memory usage statistics for a given program, decode packed string graphics, rescue programs lost by an inadvertent reboot or NEW, automatically save and load programs from disk, automatically convert uppercase strings to lower-case, and more! What? Get all those fancy trick features in the other DOSes, plus a LOT more, and use your existing DOS!

What Super Utility Plus is to disk utilities, IMPAKT! is to BASIC! This new self relocating machine-language module provides numerous extended functions for BASIC programmers. Unlike other BASIC extensions, it does not require special formats, BASIC commands, or any modification to BASIC/CMD or LBASIC/CMD, therefore programs written under IMPAKT! will run under regular BASIC. If you program heavily in BASIC you will appreciate the power that IMPAKT! gives you, plus the tremendous time it will save.

IMPAKT! You'll wonder how you ever got along without it! Compatible with most current TRS-80 compatible DOSes. A definite BOOST to TRSDOS Mod I or III and LDOS in particular. Beef up those BASICs! (Not TRSDOS 2.7DD or 6.x compatible)

**IMPAKT! on disk**  
**\$39.95 with complete users manual.**

US/Canada Please add \$2.50 Shipping/Handling-Foreign \$10. The above programs cannot be explained in this ad space. Please write for complete catalog with full details.



**MOST WANTED LIST**

**SUPER UTILITY PLUS 3.2**  
"THE JAWS OF LIFE" rated FIVE STARS in the Oct. '83 80-MICRO! A MUST-HAVE program if you have disks, as reviewed by all the TRS-80 magazines. They all agree that SU+ is a standard in the industry. This new version is the ULTIMATE and features better documentation. Includes directions for use on the Mod 4.  
**\$79.95**

**PowerMAIL PLUS**  
The most powerful mailing/information system yet for the TRS-80. Runs under most current DOSes as well as hard drives. 100% machine language for maximum speed! Features 24 definable "flags", 10 level sort, mounting new data disks, and machine compatible data files. It also "converts" many other mailing programs' data to PowerMAIL+ format. Eliminates re-typing!  
**Only \$150 each**

Please specify:  
MOD I/III  
MOD 4 TRUE Mod 4 Mode  
MOD II/12/16 (Comes on OP-II Kernel System)

**POWERDOT II**  
A GRAPHICS BREAKTHROUGH! The ULTIMATE in graphics design. This version is MUCH MORE POWERFUL than previous versions and includes BETTER documentation as well. New features include AUTODRAW™ and CIRCLE commands. Now allows you to design your own character sets! Includes lots of examples on disk. Your screen is only a "picture window" to a much larger drawing area! You are only limited by disk storage...not memory! Works on EPSON Series (Graftax of Graftax™ required) or the C.1toh 8510 (PROWRITER or NEC version). Coming soon for DMP-2100, DMP-400, and LP-8.  
**\$59.95**

**POWERDRAW**  
A full screen graphics/text editor. Allows you to design your graphics and merge them with your BASIC or assembly program in SIX different formats. The files may be used alone or chained together for animation effects. MANY programs being produced commercially today incorporate POWERDRAW graphics! No royalties to pay, and they're easy to do! Has received EXCELLENT reviews in many magazines.  
**\$39.95**

**POWERDRIVERS for use with SuperScript™**  
Allows you to use ALL of SuperScript's™ features with your EPSON, PROWRITER, or F-10 STARWRITER printers! Includes patches for running SuperScript on LDOS with floppy or hard drive. Now includes DISKFILER driver. Save formatted text to disk now!  
**Only \$29.95 each**

Please Specify:  
PowerDRIVER/P - C.1toh 8510 PROWRITER  
PowerDRIVER/E - EPSON or GEMINI Series  
PowerDRIVER/F - C.1toh STARWRITER (Leading Edge)

**POWERSOFT**  
PRODUCTS FROM BREEZE/QSD, INC.

11500 Stemmons Fwy.  
Suite 125  
Dallas, Texas 75229  
(214) 484-2976 **306**

SuperScript is a Trademark of Tandy Corp.  
LDOS is a trademark of LSI



Mention the LSI Quarterly in your order and deduct 15% from prices!