EMERGENCY COVER

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

We Now Have A Toll Free Orderline! (800) 248-3535
See What You Can Do With DOTWRITER 4.0!
Now available for the Model 4, too!

This 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 (80 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 these 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 Grafix, MX-100 with Grafix, Plus, RX-80, and FX-80; the C. ITOH 851D/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
(Model I, III) $79.95
(Model 4) 99.95
Letterset Design System (LDS) 39.95
Special: DOTWRITER and LDS for Models I, III 99.95
for Model 4 119.95
(Model 3 LDS) 179.95 and 24.95

ORDER NOW, TOLL-FREE
(800) 824-7885, oper. 577

PROSOFT
Dept. C, Box 560, N. Hollywood, CA 91603
(818) 764-3121 Information and Same-Day Processing
TERMS: VISA, MC, cheq., COD. Please add 9.00 shipping in U.S. or Canada. 850.00 overseas. Sales tax in CA.
Most orders filled within one day.
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.)
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** A 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 an 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>
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.


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=uncleared 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 908
(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 B6 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 PED (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 (+) 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.
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 (whoops that $ 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-site 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 ATST 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 USTORS - 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.

<table>
<thead>
<tr>
<th>Listing 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>START: LD DE,FSPEC ;header to find</td>
</tr>
<tr>
<td>LD A,83 ;GTMOD</td>
</tr>
<tr>
<td>RST 28H</td>
</tr>
<tr>
<td>FSPEC: DEFM 'INKY4'</td>
</tr>
<tr>
<td>DEFB 0DH</td>
</tr>
</tbody>
</table>

Locating high memory routines under LDOS/TRSDOS 6.X
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 into E
INC HL ; it 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

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(X))
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 bytes 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 in 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**

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 TRSDOS 6 "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 L/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.
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. 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.)

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:

\[ a[n] \ \text{is equivalent to} \ *(a+n) \]

\[ *(\&a[n]) \ \text{is equivalent to} \ *(a+n) \]

\[ \&a[n] \ \text{is equivalent to} \ \&a[0]+n \ \text{is equivalent to} \ a+n \]

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

\& \ \text{means the address}

\* \ \text{means the data at}

\[ a[n] \ \text{means element n of array a} \]

\[ \&a[n] \ \text{means the address of element n of array a} \]

\[ *(\&a[n]) \ \text{means the data (element) at the address of element n of array a} \]

This says the same thing as element n of array a

\[ *(a+n) \ \text{means element n of array a} \]

\[ a+n \ \text{means element n of array a} \]

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 \[ \text{argc} \] and \[ \text{argv} \] are not new to you, they were described in Part II. As you noted, \[ \text{argv} \] is treated as an array and as a pointer in the above program. Are you curious about the \[ \text{argv}[0][0] \] expression used in the \[ \text{printf} \] 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 \[ \text{test} \], 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

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

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

The \[ \text{argv} \] 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: \[ \text{argv}, \ \*\text{argv}, \ \text{argv}[0], \ \*(\text{argv})[0], \ \text{argv}[0][0]. \] Having some trouble “visualizing” what each represents? Look at a possible storage map (chart) of \[ \text{argv} \]:

<p>|---V---| |---V---|</p>
<table>
<thead>
<tr>
<th>argv</th>
<th>address</th>
<th>*argv[0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>argv[0]</td>
<td>address</td>
<td>argv[0][0]</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>string</td>
<td>address</td>
<td>string</td>
</tr>
<tr>
<td>data</td>
<td>address</td>
<td>data</td>
</tr>
</tbody>
</table>

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 \[ \text{argv} \]. 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 \[ \text{struct} \] keyword is used to declare a structure. An optional name or tag can follow the \[ \text{struct} \] keyword. In this example, I used the tag of \[ \text{payroll} \]. 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 \[ \text{payroll} \], it might be done as follows:

```c
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:

```c
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

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

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 hit 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

Table 1 - C Language Pitfalls

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

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 hit 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

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 hit 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
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 I/O 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...
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 ;Check mod1/3 ROM
00140 @CLOSE EQU 4428H
00150 @DEFLY EQU 4467H
00160 @ERROR EQU 4409H
00170 @EXIT EQU 402DH
00180 @FSPEC EQU 4411H ;Model 3
00190 @INIT EQU 4049H ;Model 1
00200 @KEYIN EQU 0040H
00210 @OPEN EQU 4424H
00220 @READ EQU 4436H
00230 @WRITE EQU 4439H
00240 HIGH$ EQU 4411H
00250 HIGH1 EQU 4049H
00260 ;
00270 ORG 5200H
00280 ;Put file buffer first to force location
00290 ;on memory page boundary
00300 BUFRLO DS 256
00310 BUFRHI DS 256
00320 ;Machine specific code:
00330 BEGIN: LD A, (125H) ;Check mod1/3 ROM
00340 CP . 'I'
00350 LD HL, (HIGH$) ;Mod 3 location
00360 JR Z, SETHI ;Go if mod 3
00370 LD HL, (HIGH1) ;Mod I location
00380 SETHI: LD (MYMEM), HL ;Store correct value
00390 ;
00400 ;End of LDOS 5.1 specific code
00410 ;****************************
00420 ;Special chars
00430 ETX EQU 03H
00440 CR EQU 0DH
00450 LF EQU 0AH
00460 ;
00470 ;FCB offset definitions
00480 BUFRLO EQU 3 ;Buffer address
00490 BUFRHI EQU 4
00500 ERNHI EQU 13 ;Ending record #
00510 ERNLO EQU 12
00520 EOF EQU 8 ;Offset of last byte
00530 NRNLO EQU 10 ;Next record pointer
00540 NRNHI EQU 11
00550 ;
00560 START: LD HL, LOGON ;Log on
00570 DISKIN: LD HL, MSG1 ;Prompt for input file
00580 CALL INPFSP ;Get answer
00590 CALL @OPEN ;Open the file
00600 JR Z, GOTINP ;Go if successful
00610 CALL SHOERR ;Else report error
00620 ;File is open, check if it contains any records
00630 GOTINP: LD HL, (FCB1+ERNLO)
00640 JR Z, ASK ;Go if file has data
00650 LD HL, BUFHR1 ;=>disk buffer
00660 CALL @OPEN ;Open the file
00670 JR Z, GOTINP ;Go if successful
00680 CALL SHOERR ;Else report error
00690 JR DISKIN ;And ask again
00700 ;
00710 ;This part is for LDOS 5.1
00720 ;operating system entry points:
00730 @ABORT EQU 4030H ;Check mod1/3 ROM
00740 @CLOSE EQU 4428H
00750 @DEFLY EQU 4467H
00760 @ERROR EQU 4409H
00770 @EXIT EQU 402DH
00780 @FSPEC EQU 4411H ;Model 3
00790 @INIT EQU 4049H ;Model 1
00800 @KEYIN EQU 0040H
00810 @OPEN EQU 4424H
00820 @READ EQU 4436H
00830 @WRITE EQU 4439H
00840 HIGH$ EQU 4411H
00850 HIGH1 EQU 4049H
00860 ;Machine specific code:
BEGIN

00890 JR ASK ;And re-prompt

00900 ;

00910 ;Set up for add or remove LF's

00920 ASK1A: LD HL,MSG3 ;Remove linefeeds?

00930 LD DE,RLF

00940 CALL SGETYN ;Prompt, set flag

00950 JR Z,AGAIN ;If yes, skip 2nd

prompt

00960 ;

00970 ASK1B: LD HL,MSG4 ;Add linefeeds?

00980 LD DE,ALF

00990 CALL SGETYN ;Prompt, set flag

01000 JR NZ,ASK1A ;Must do one or the

other

01010 ;

01020 ;Read file into memory - set (MORE)=OFFH if it

doesn't fit

01030 AGAIN: LD HL,(MYMEM) ;=>end of free space

01040 LD BC,BUFFER ;=>working buffer

start

01050 OR A

01060 SBC HL,BC

01070 LD B,H ;# of sectors that

will fit

01080 LD HL,BUFFER ;Start of buffer

01090 LD DE,FCB1 ;=>input file buffer

01100 RLOOP: LD (FCB1+BUFRLO),HL ;Set load

address

01110 CALL @READ ;Get a sector

01120 JP NZ,CKEND ;End or error

01130 INC H ;Back up to last

full sector

01140 ADD HL,DE ;Add in contents of

last sector

01150 ;Now check to see if the last sector contained the

EOF

01160 ;To be sure all data loaded belongs in file

01170 PUSH HL ;Save current posn

01180 LD HL,(FCB1-RRNLO) ;Check if this is

the buffer?

01190 LD DE,(FCB1+RRNLO) ;Last sector

01200 OR A

01210 SBC HL,DE ;Is NRN=ERN?

01220 POP HL ;Get end ptr

01230 JR Z,ISEOF ;Go if this is EOF

01240 LD A,OFPH ;Set flag for more

input

01250 LD (NORR),A ;Save current postn

01260 JR FINIS ;And start

processing

01270 ;

01280 CKEND: CP 1CH ;EOF?

01290 JR Z,ISHOF ;Yes - processing finished

01300 CP 1DH ;Do past EOF?

01310 JP NZ,DOERR ;Quit if other error

01320 ISHOF: LD A,(FCB1+EOF) ;Get EOF offset byte

01330 DEC A ;Point to end byte

01340 LD E,A

01350 LD D,0 ;DE-offset of EOF

01360 DEC H ;Back up to last

full sector

01370 ADD HL,DR ;Add in contents of

last sector

01380 INC HL ;HL=>1 past buffer

01390 ;

01400 FINIS: EX DE,HL ;Pointer to DE

01410 LD BC,BUFFER ;START OF DATA AREA

01420 ;

01430 ;Memory is loaded, check for changes

01440 ACHAR: LD A,D ;Is this the end of

the buffer?

01450 CP B

01460 JP NZ,NOTEND ;Go if not LF

01470 LD A,E

01480 CP C

01490 JR Z,CHECKM ;End, check for more

input

01500 NOTEND: LD A,(BC) ;Get a character

01510 INC BC ;bump pointer

01520 CP LF ;LF?

01530 JR Z,NULF ;Go if not LF

01540 JP NZ,NULF ;Go if LF

01550 LD D,(RLF) ;Removing line

feeds?

01560 OR A ;Test

01570 JR Z,DOLF ;Not removing line

feeds

01580 JR ACHAR ;Skip LF

01590 CP CA ; carriage RET?

01600 JP NZ,WRBYT ;Write if not CR

01610 CALL PUTTER ;Write CR

01620 LD A,(ALF) ;Adding line feeds?

01630 OR A ;Test

01640 JR Z,ACHAR ;Skip if not wanted

01650 DOLF: LD A,LF ;Load the line feed

01660 WRBYT: CALL PUTTER ;Write char to file

01670 JP ACHAR ;Loop through buffer

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),O ;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 LD A,(FCB2+MORE) ;Else report

01810 JR NZ,DOERR ;Go if error

01820 CALL @DSPLY ;Issue prompt

01830 ;And go back to DOS

01840 ;

01850 ;Set flag according to 'Y' response

01860 SGGETYN: PUSH DE ;Save flag address

01870 CALL @DSPLY ;Issue prompt

01880 LD HL,YN$ ;Add Y/N?

01890 CALL @DSPLY ;Give upper case

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?
01940 POP HL ;Get flag address
01950 RET NZ ;Return if not "Y"
01960 LD (HL),0FFH ;Set flag if "Y"
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 ;Add a byte to disk buffer/write if full
02080 ;Note that disk buffer must end on XXFFH boundary
02090 ;So the INC L will set the Z flag when sector
02100 ;Buffer is full. (PUTPTR) is a pointer to the next char
02110 ;Position in the sector buffer
02120 ;Point to buf pos
02130 ;Move to buffer
02140 ;Bump buffer ptr
02150 ;Save for next
02160 ;If not full
02170 ;Write a physical sector to disk
02180 ;That was easy
02190 ;Report any error
02200 JP @ABORT ;And quit
02210 ;Buffers & Pointers
02220 ;BUFFERS & POINTERS
02230 ;BUFFERS & POINTERS
02240 ;BUFFERS & POINTERS
02250 ;BUFFERS & POINTERS
02260 ;BUFFERS & POINTERS
02270 ;BUFFERS & POINTERS
02280 ;BUFFERS & POINTERS
02290 ;BUFFERS & POINTERS
02300 ;BUFFERS & POINTERS
02310 ;BUFFERS & POINTERS
02320 ;BUFFERS & POINTERS
02330 ;BUFFERS & POINTERS
02340 ;BUFFERS & POINTERS
02350 ;BUFFERS & POINTERS
02360 ;BUFFERS & POINTERS
02370 ;BUFFERS & POINTERS
02380 ;BUFFERS & POINTERS
02390 ;BUFFERS & POINTERS
02400 ;BUFFERS & POINTERS
02410 ;BUFFERS & POINTERS
02420 ;BUFFERS & POINTERS
02430 ;BUFFERS & POINTERS
02440 ;BUFFERS & POINTERS
02450 ;BUFFERS & POINTERS

Listing 2 LDOS 6.x version

00100 ;*******************************
00110 ;Header to convert programs with 5.1 CALLS to 6.x
00120 ;
00130 ORG 2600H
00140 BUFFR1: DS 256 ;put on page boundary
00150 BUFFR2: DS 256
00160 ;DI
00170 LD (STACK),SP ;save SP at entry
00180 PUSH HL ;Save ptr to CMD buffer
00190 LD A,101 ;set up IY
00190 PUSH (MYMEM),HL ;Store for later
001A0 ;
001B0 ;BH2IN: DI
001C0 LD (STACK),SP ;save SP at entry
001D0 PUSH HL ;Save ptr to CMD buffer
001E0 LD A,103 ; Disable break vectoring
001F0 POP AF ;EOF byte to A
00200 ;
00210 LD A,101 ;set up IY

Page 21
Index
Index to LDOS Quarterly/LSI Journal, issues 1,1 to 2,4

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
"Alcor Pascal" - Scott Loomer  V2N1P18
Allocation, disk  V2N4P38
AM Electronics, disk controller  V1N6P16
"APL*PLUS/80 A System Overview" - Daniel Lofty & Lee Rice  V1N6P20
"At Large" - Earl Terwilliger  V1N5P14
"Automatic Chaining with JCL" - Jim Kyle  V1N5P16
"ASCII File Listing Utility for The BASIC Answer" - Jeffrey Brenton  V2N3P19
Assembly language basic concepts  V1N6P38
Assembly language patching  V2N3P19
Assembly language programming tips  V2N3P19
"Beta Tester, I was an LDOS" - Tim Daneliuk  V1N6P38
"BASIC Concepts - The RUN,V Command" - Dick Konop  V1N6P38
"BASIC and File Structure - A Beginner's View" - Wes Goodnough  V1N6P38
"Beta Tester, I was an LDOS" - Tim Daneliuk  V1N6P38
".Byte I/O under LDOS 6.0"  V1N6P38
"C' graphics  V1N6P38
"C' graphics, The" - Earl Terwilliger:  V1N6P38
  general introduction  V1N6P38
"Article, An" Charlie Butler  V1N6P38
"ASCII File Listing Utility for The BASIC Answer" - Jeffrey Brenton  V2N3P19
Assembly language basic concepts  V1N6P38
Assembly language patching  V2N3P19
Assembly language programming tips  V2N3P19
"At Large" - Earl Terwilliger  V2N3P19
"Automatic Chaining with JCL" - Jim Kyle  V2N3P19
"BASIC and File Structure - A Beginner's View" - Wes Goodnough  V2N3P19
"BASIC Concepts - The RUN,V Command" - Dick Konop  V2N3P19
"Beta Tester, I was an LDOS" - Tim Daneliuk  V2N3P19
".Byte I/O under LDOS 6.0"  V2N3P19
"C' graphics  V2N3P19
"C' graphics, The" - Earl Terwilliger:  V2N3P19
  general introduction  V2N3P19
Manual story, with corrections to 1st edition

Memory map

Minimum configuration disk

"Mixing Newscript, Electric Webster, LDOS and Sole" Jerry Latham

"My BASIC Answer" - E. Cheatham (TBA review)

"Newscript 7.0 and REFLEX" - Gordon Thompson

"Newscript and The BASIC Answer" - Jerry Latham

”New Version - EDAS IV, A” - Marc Leager

Number base conversion

Old DAMs (see also Data Address Marks)

Parity errors

-- PARITY = ODD - Tim Daneliuk:

introduction, DATAENTR 200 & ISAM 200

tips for better programming, DISCATER, Filter Disk

BASF drives, drive poll, TAS, MODEM80, HEXSPELL II, Utilities, HELP

Tandy, MAX-80, AEROCOMP, Electric Webster, The BASIC Answer

disk drive poll, gold plugs, LX-80, Proofreader, Scriptsp Dct.

magazines

rumors, LX-80 software compatibility, TRSDOS 6.0 software

"Partitioned Data Sets, MISOSYS Announces" - Roy Soltoff

"Pascal 80, LDOS and” - D. Hill

Passwords for LDOS 5.1.x

Patch, how to

"PDS - Standard and Other Types of Uses" - Scott Lower

"Performing Date Conversions in BASIC" - Dick Konop

Printers

Profile III Plus with LD03

Quarterly reader survey results

Radio Shack

"Relocating Code for LBASIC USR Routines" - Chuck Jensen

Reviews;

Alternate Source (TAS) - Tim Daneliuk

APL*PLUS/80 (STSC) - Daniel Lofy & Lee Rice

CHROMAtics (South Shore Computer Concepts) - Scott Lower

DATAENTR200 & ISAM20V (Johnson Associates) - Tim Daneliuk

EDAS IV (MISOSYS) - Marc Leager

Electric Webster (Cornucopia) - Tim Daneliuk

HELP (MISOSYS) - Tim Daneliuk

HEXPPELL II (Hexagon) - Tim Daneliuk

LDOS Utilities (Powersoft) - Tim Daneliuk

LISP (Supersoft) - Lee Rice & Daniel Lofy

MAX-80 (Lobo) - Tim Daneliuk

Microcomputer Math book (SAMS) - Earle Robinson

MODEM80 (LSI) Tim Daneliuk

Pascal (Alocr) Scott Lower

Pascal-80 IN ew Classics) - D. Hill

PDS (MISOSYS) Scott Lower

Printers - Earle Robinson

Profile III (Radio Shack) - Sam Goldberg

Proofreader (Aspen Software) - Tim Daneliuk

Scripsp Dictionary (Radio Shack) - Tim Daneliuk

Structured BASIC Translator (Acorn) - Sue Ratkowski

The BASIC Answer (LSI) E. Cheatham

The BASIC Answer (LSI) Tim Daneliuk

UOLISP (Far West) - Lee Rice & Daniel Lofy

Routing a device

"Roy's Technical Corner" Roy Soltoff:

load module structure

error handling during byte 1/0, @CKDRV, KFLAGS, @ICNFG, @KITSK

LDOS 6.0

RS-232 drivers

"Running 8-Inch Drives on the Model III Under LDOS" - Peter Simon

Sector I/O

"SOLEFIX - Fix that GAT Error" - Erik Ruf

Speedups kits

Syssgen and type-ahead

System routines:

@ADTSK, @RMTSK, @KLTSK, @RPTS: - task proc.

@CKDRV - determining 5.1.2 vs. 5.1.3 for @CKDRV adjustment

@CMNDI - command interpreter

@CTL, @GET, @PUT - byte 1/0

DAYS - day of the week

@GET and @PUT under LDOS 6.0

@ICNFG configuration interfacing

KFLAG$ keyboard scanner

"@PARAM, Using" - Roy Soltoff

@PARAM under 6.0

"@PARAM, @DSPLY, @EXIT and INBUFS for Everyone" David Vinzant

@RAMDIR documentation correction

SVCs and Fortran

Tandy

"T-Timer, LDOS Supports the" - Roy Soltoff

TRSDOS (Mod I) to LDOS (Mod III) transfer without REPAIR

TRSDOS (LDOS) 6.0 (see also LDOS 6.x)

Update policy explained - Bill Schroeder

Upper case lock

Users group directory

Utilities:

CONV

COPY238

PATCH

REPAIR

Version number explanation

VisiCalc made easy

"VisiCalc with LDOS, Using" - Roy Soltoff

VisiCalc made easy

VisiCalc made easy
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 US MICRO as a package "fit for the best of LDOS™. We now offer this for the best of TRSDOS as well. These new disk utilities are intended for TRSDOS 6.x or LDOS 5.1 and unlike Toolbox '83, these will work on any combination of single-sided and double-sided drives. The latest version is 1 MB in size, can edit up to 2 MB of memory, and includes all the features of Toolbox '83. We also offer a larger version (2 MB) that includes additional features.

Utilities include disk repair, disk rewrite, directory viewer, disk memory in use, file viewer, compare files, unerase, backup, directory manipulation, password security, partitioning, backup-to-tape, automatic backup to tape, and a user-definable user interface. All utilities are updated for use with LDOS or TRSDOS 6.x.

MODEL 4: Toolbox (for TRSDOS 6.x only) $45.95
The Toolbox for LDOS™ (MID: 110)

WANTED
Your BASIC
Impakt!

DO YOU USE DISK BASIC? WHAT DO YOU WELL WITH DISK BASIC?

You have a disk with a single command line: BASIC. To search another disk, you must first use a particular keyword in string search searches. Here's how you can do it. The following is a complete program that will search your disk for a string. If you use this program, you can search for a string in your disk. This program is automatically loaded into your disk, where it can be used by any disk program that supports string searches.

New features include ASCII-to-bit and bit-to-ASCII conversion, new search options, and automatic backup to tape. This program is also available in the TRSDOS 6.x version of Toolbox '83.

MODEL 2: IMPAKT! (for TRSDOS 6.x only) $29.95

Most Wanted List

SUPER UTILITY PLUS 3.2
"THE JAWS OF LIFE" super fast, simple, and easy to use. Works with the latest version of TRSDOS 6.x. Requirements include a compatible computer, memory, and disk. $78.95

POWERMAIL PLUS
The most powerful, full-featured mail system available for TRSDOS 6.x. Works with most versions of TRSDOS 6.x as well as some others. This program includes a full-featured mail system, complete mail facility, and batch mail facility. $89.95

POWERBOT II
A graphics breakthrough! The ultimate in graphics design. This version is much more powerful than previous versions, and includes better documentation and support. New features include: AutoCAD™ and X.Y.Z. commands. Now allow you to design your own character sets. New tools include: X.Y.Z. and AutoCAD™ commands. New features include: AutoCAD™ and X.Y.Z. commands. New tools include: X.Y.Z. and AutoCAD™ commands. New tools include: X.Y.Z. and AutoCAD™ commands.

POWERDRAW
A full-screen graphics editor. Allows you to design your graphical image with your terminal in assembly program via different tools. The tools may be used alone or in combination to produce a final image. This program is available in TRSDOS 6.x version of Toolbox '83.

POWERDRIVERs for use with SuperScript
Allow you to run all of SuperScript™, features with your EPSON Promoter. It is a fast, easy-to-use program that allows you to perform many of the functions of SuperScript using TRSDOS 6.x. Now includes a complete set of TRSDOS tools. $29.95

POWERSTUDIO
EPSON FL-210 PROMOTER PowerDriver for EPSON FL-210 PROMOTER PowerDriver for EPSON FL-210 PROMOTER $29.95

PowerSOFT PRODUCTS FROM BREEZE-IT INC.
11500 Stevenson Blvd.
Suite 125
Oakland, CA 94612
(214) 484-2976

This ad appears in the 11th issue of the LDOS Quarterly magazine

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