January 1, 1984

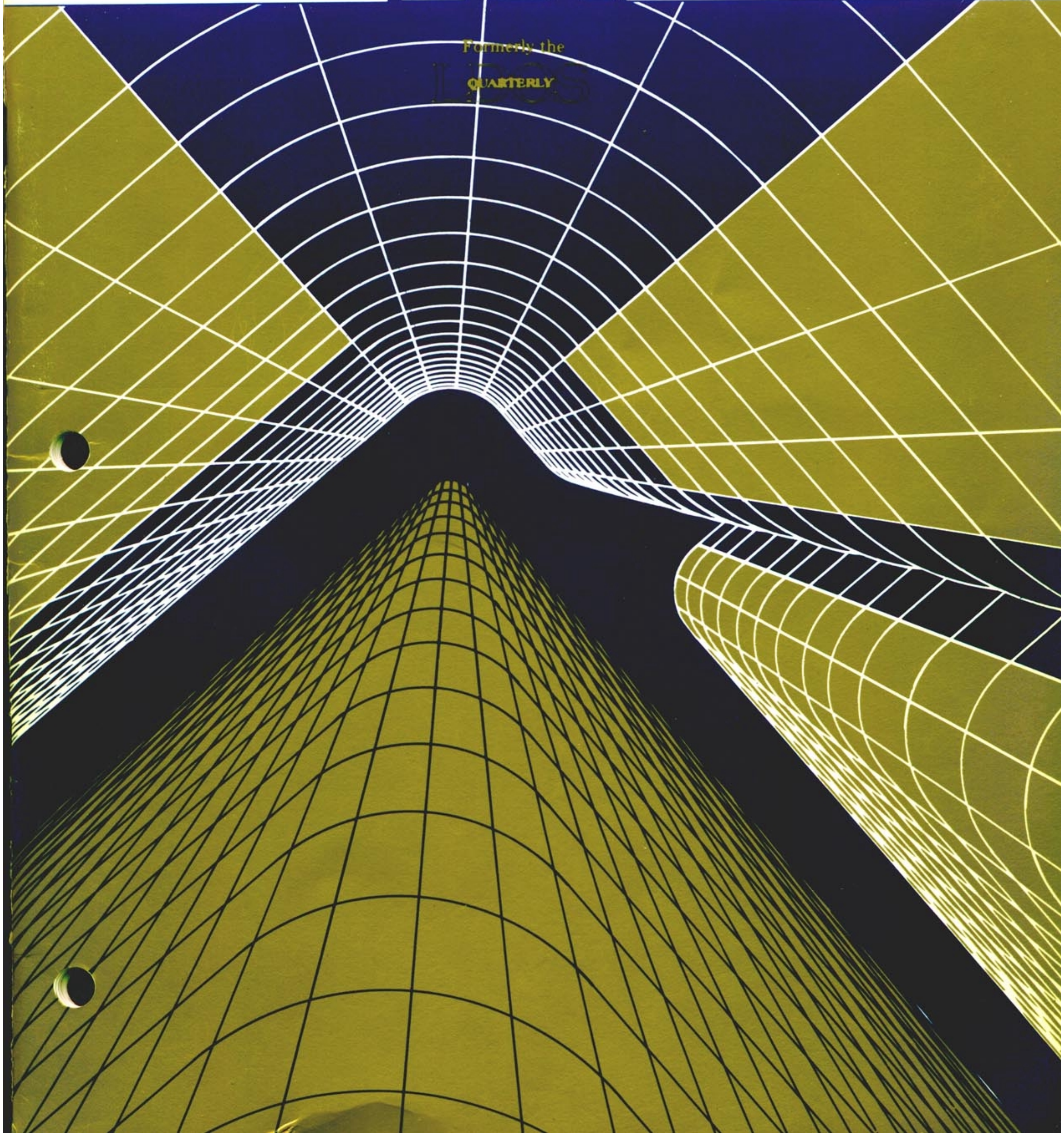
Volume 2, Number 5

LSI JOURNAL

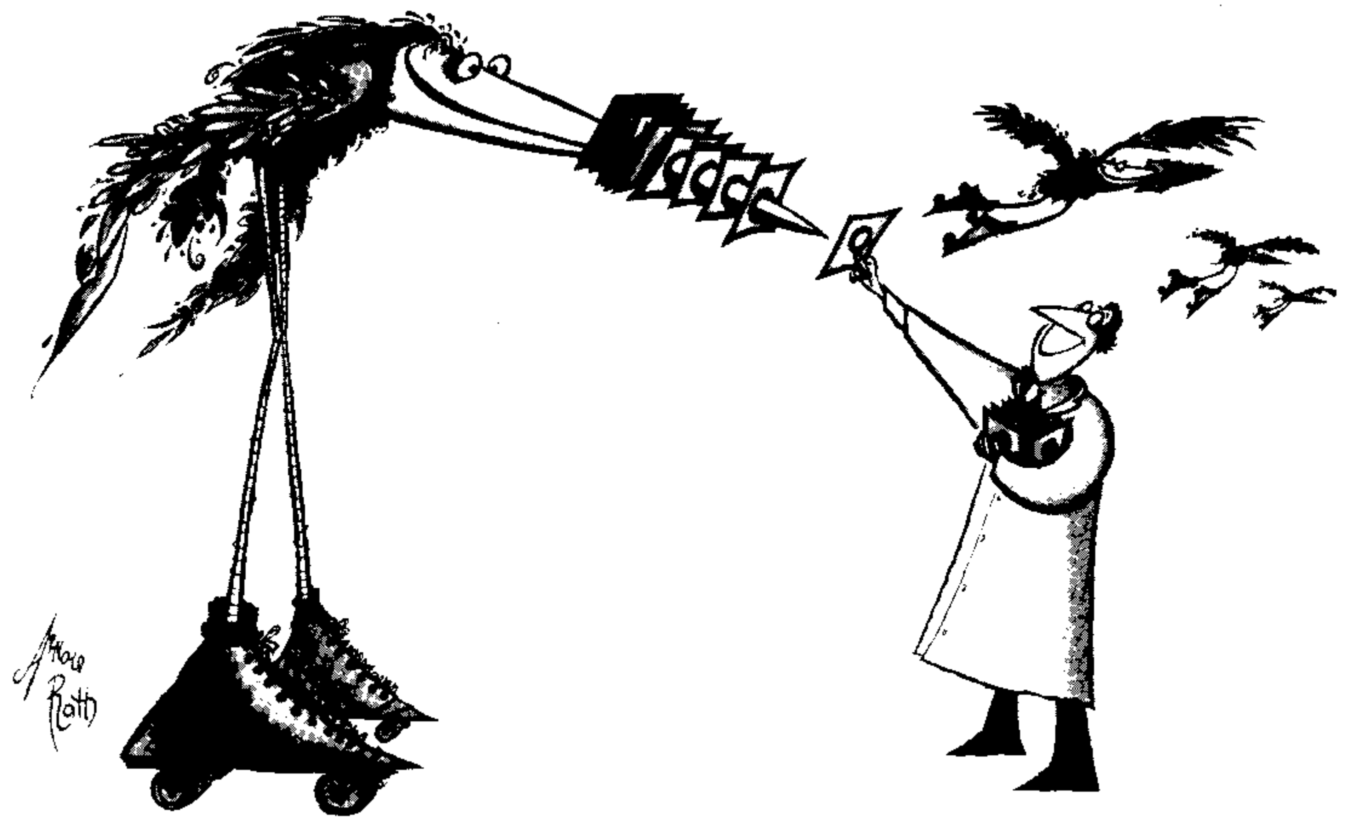
ISSN 0737-9161

$4

Formerly the LDOS QUARTERLY



Smallware™



Our software is making a name for itself.

Smallware. That's what we've named our unique software designed for microcomputers. Smallware offers much more than ordinary software: high quality, customer support and a complete product line. You can buy software anywhere. But for the special features of Smallware, The Small Computer Company is your one and only source.

The Small Computer Company is known to many as the company who developed the filing system software Profile III Plus—LDOS and TRSDOS versions—for Radio Shack.

Now, whether you're a microcomputer end-user, dealer or manufacturer, you can order our Smallware directly from us.

Here are just some of the enhancements we offer to Model III LDOS and TRSDOS users:

FORMS: If you prepare forms that require several lines of data, from invoices to shipping instructions, Forms is invaluable. It allows you to print individual forms (up to 13" x 11") with graphics, trademarks, logos, underlining, subscript and superscript functions $125

ARCHIVE: Lets you maintain up-to-the-minute, clean files by removing inactive records and transferring them to a pre-determined list or file; split an existing data base into any number of specialized files; free substantial disk storage space $150

PROPACK™: A tool that lets BASIC programmers more easily customize Profile systems. The resulting programs are shorter, easier to write and faster running. Propack also gives the BASIC program indexed access to Profile data $75

We also have other enhancements for the TRSDOS version of Profile III Plus and for Profile Plus, which runs on the Models II and 12. Call or write for our complete product line brochure.

The Small Computer Company does more than create award-winning Smallware. Our commitment to the customer extends to custom design as well as system consultation.

To order any of these products, call our toll-free sales number, 1-800-847-4740 (in New York state, call 1-212-398-9290).

The Small Computer Company, Inc.

230 West 41st Street, Suite 1200, New York, New York 10036

80 readers' choice micro 1982

Smallware, Propack, Quikback and filePro are trademarks of The Small Computer Company, Inc.
CP/M is a registered trademark of Digital Research Inc. Profile is a registered trademark of Radio Shack

THE LSI JOURNAL   January 1984   Volume 2. Number 5

Table of Contents

The LDOS Quarterly policy on the submission and payment for articles is as follows:

Articles sent for consideration must be submitted in the following format:

1. A cover letter, summarizing the content and intent of the article.
2. A printed hardcopy of the article. Desired print effects and formatting should be indicated where necessary.

A diskette with--

3. A 'plain vanilla' ASCII text file containing the article. The text should be free-form (without "hard" carriage returns), but any tables or other structured data should be formatted as 87 characters per line. Do NOT send SuperSCRIPSIT or Newscript files. Also, please do not embed print effects.
4. If the article involves assembly language programs, include both the source code, and the object code.
5. Any other necessary files or patches should also be supplied in machine readable form.

Please do not send in printed text without a diskette, as it will NOT be considered for publication. Payment will be made in the form of an LSI product, or $40 per published page in the current Quarterly format. The size of the article will determine the value of the LSI product available as payment.

Please include your name, address, telephone number and LDOS serial number with your submission, firmly attached to your hardcopy printout, and affixed to the diskette you submit.

LSI is extremely interested in seeing submissions from our users, and is open to suggestions on any ideas for the Quarterly.

Submissions should be sent to:

The LDOS Quarterly Editor
c/o Logical Systems, Inc.
8970 N. 55th Street
P.O. Box 23956
Milwaukee, Wisconsin 53223

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

by Bill Schroeder

Well, another year has come to an end, and I thank you all once again for being supporters of LSI, and the LSI product line. With your support LSI has made it through yet another year.

1984 will be a year of rapid and massive change in the microcomputer industry. Here are a few of my quickie predictions for the year ahead:

Tandy will try their hand at competing with IBM (no easy task there).

AT&T will enter the market place with a whole new concept.

Commodore will continue to dominate the low end market.

Digital Research will increase its presence in the systems software arena.

The REAL "Peanut" will be introduced by IBM.

TeleVideo will become a major force in the market.

Removable cartridge hard drives will become more prevalent.

Microsoft will fall slightly from its present lofty position.

Apple will begin to have serious trouble.

Over one hundred small computer product manufactures will go out of business.

The Coleco ADAM will prove to be a failure.

Over one-half of the TRS-80 software suppliers will falter or fail.

Over 25 microcomputer publications will fail.

It is quite probable that none of the above events will take place, but they are interesting possibilities - - -

Now, on to what I know will be changing at LSI in 1984:

LSI will not be publishing the LSI Journal, but the people at BASIC COMPUTING will be! That's right, starting with the April issue of BASIC COMPUTING, the LSI Journal will be incorporated as a special section in that magazine. This arrangement with these folks was made so that LSI could get out of the business of publishing a "magazine". All the same authors will be writing for us, and we will be providing the material to BASIC COMPUTING for publication, on at least a quarterly basis. All existing subscribers will have their subscriptions filled by BASIC COMPUTING. If you already subscribe to BASIC COMPUTING, your subscription will be extended the proper number of issues. For those who are not subscribers to either publication, a subscription order card is included in this issue.

This means that this is the final issue of the LSI Journal published by LSI. The major bulk of technical and other information will now be imparted through BASIC COMPUTING. But--- there will be a new publication from LSI. This will be the "LSI Newsletter". It will be published for the purpose of announcing new products, special offers and the like to our valued customers. All registered LSI customers will receive this newsletter at NO CHARGE. It will be published on an as required basis, but we expect at least several times per year. It will contain little, if any, technical information.

The LSI Hotline phone number has been disconnected due to lack of interest. There have been very few phone calls, far less than we expected. I can't justify the cost of the phone line, the answering equipment and the creation of the content for such a small audience. This service was discontinued effective January 1st. To those out there who called and appreciated the LSI Hotline, my apologies.

S U P E R   S A L E S   a n d   V A L U E S

DEAL #1: While our supplies last, any product LSI has in stock that is not manufactured or published by LSI will be sold-out for 40% OFF of the suggested retail price. Don't miss this chance. LSI will no longer be selling software that is not published and/or manufactured by LSI. We are liquidating our inventories of these products, to your benefit. All <IN STOCK> products from MISOSYS, MICROPRO, POWERSOFT, MOLIMERX, ... are to be sold at 40% off of suggested retail. We will NOT fill backorders or give rain checks for these products. We have dozens of some items, and only a few of others. These will be shipped on a first-come, first-served basis, and any orders for items that are already sold-out will be promptly returned/refunded. To take advantage of this offer, you must indicate that you are taking advantage of Deal #1 on your order or over the phone. Please note that the special introductory offer for WordStar and MailMerge is over, and that they have returned to their regular price of $395 and $249, respectively.

Products we have that are not listed in the LSI catalog:

PowerMail Plus, Model 1/3 version by PowerSoft	Sugg. retail $150
PowerMail Plus, Model 4 version by PowerSoft	Sugg. retail $150
ZSHELL, for LDOS 5.1 from MISOSYS	Sugg. retail $ 40

The 6.x versions of EDAS (PRO-CREATE) and DSMBLR III (PRO-DUCE)

For the most part, these and all the other discontinued products are fine products from excellent companies. Our decision to discontinue, marketing, promotion, support, endorsing, etc. of non-LSI products should in no way reflect on the quality of these products or companies.

DEAL #2: Special LSI 30% discount. For orders postmarked between March 1st, 1984 and March 15, 1984, take a 30% discount on any LSI-manufactured product. This includes LDOS, FED II, LED and all our other excellent LDOS support products (don't forget diskDISK). This offer is not good in conjunction with any other offer, but you may have both Deal #1 and Deal #2 items on the same order. This offer is also good on phone orders placed in this period, but again you must indicate Deal #2 over the phone (or on your order).

DEAL #3: FREE LSI Journal Issues. With any order totaling $50 or more (net amount after discount), get the previous Volume 2 LSI Journal/LDOS Quarterly issues at no charge. This includes Volume 2, numbers 1 through 4. Number 4 is in short supply, so if we run out, you will get only numbers 1 through 3. Take advantage of this order quickly if you need all four. Again, you must note this special offer on your order (or over the phone). This offer is good until we run out of back issues.

DEAL #4: How about the full LDOS 5.1.4 operating system at 1/2 price? That's right, <u>FULL</u> LDOS 5.1.4 for just $64.50. This special offer is available to CONVERTS from both NEWDOS and DOSPLUS. Now is the time for all your friends to convert to the power of LDOS. Here's how it works:

From February 1, 1984 until June 30, 1984, LSI will provide the complete LDOS 5.1.4 system for the price of $64.50, plus $5.00 shipping and handling, to anyone that trades in their NEWDOS-80 or DOSPLUS 3.4/3.5/4 operating system. To take advantage of this special, just send your original MASTER disk and MANUAL to LSI along with $69.50, and we will rush a fresh LDOS 5.1.4 operating system right out. This offer is not good in conjunction with any other offer. Don't forget to mark your order as Deal #4 or the LDOS Trade-In offer. Oh, one more thing... I will even take our own LDOS 5.0 systems in trade!

DEAL #5: We have several extra Radio Shack Hard Disk systems. These are in good, used condition, but are being sold on an as-is basis due to the fact that we do not have any of the manuals or cables (other than the power cable). Because of this, we recommend these to experienced users who already own a RS HD system. The price, you ask? Well, these are a steal at $995 for a primary or $695 for a secondary, plus shipping and handling. None of the above discounts apply, but we will throw in Deal #3.

All these "Deals" are good only directly from LSI, and not from any of our dealers.

In the future, LSI will again market software not created by LSI, hopefully by the middle of 1984. When we do, these will be very carefully selected products that are manufactured and supported by LSI, even though originally written outside. We will no longer offer products that do not bear the LSI name.

We had originally planned a new catalog for January '84, but due to needed product line changes, including the changes outlined above, we have decided to wait until June of 1984 to publish our new catalog. This will be sent to all registered LSI customers at no charge. We think our new catalog will be well worth the wait. At that time we will be implementing many new policies and pricing changes along with the revamped product line. For the time being, (the first half of '84) all LSI prices and policies will remain as stated in our current catalog.

Here's an interesting thought: "Beware of pre-release software". When a software company sends out BETA test (the second test phase) copies of software, they sometimes appear on the "Underground Software Exchange" overnight. Beware-- if anyone offers you a test copy of a new or updated product, you may be getting a free time bomb. That product is in BETA testing because the producing company is still in the process of locating and correcting errors. I know of one incidence of a fellow that received a clandestine copy of an update to a popular spread-sheet program. He was very excited about this find because this was the product he used everyday in his business, and now he had "THE HOT NEW VERSION". (Note: Apparently he never purchased the original version in the first place.)

After about three days of playing with it, he had managed to destroy all of his existing data files (several hundreds of hours of work). He called the company and they very nicely told him to go to hell. He was not a legitimate owner of their product in the first place, had no right to have the BETA product he obtained, and found out that the product was expected to destroy his files. After all, it was a BETA TEST version and not a released product. So beware, it may not be a good idea to be the first on your block with a neat, new product, if that is before it is even released.

For those who are interested, the current version of the TRSDOS 6.x system for the Model 4 and 4P is 06.01.01, and should be available from your local RS store. I think its a freebie. They should also have the hard disk drivers and formatter for TRSDOS 6.X (it is available off-the-shelf in the Midwest Computer Centers, anyway). The latest version of LDOS for the Model 1, 3 and MAX-80 is 5.1.4 with file dates of 10/01/83.

The Model 2/12 version of TRSDOS 6.x is complete (and in final testing) as of this writing. This will be known as LS-DOS 6.2.0 (or TRSDOS if purchased through Tandy). The pricing and public availability of this product are still in question. If you are interested in this product, please contact LSI in March of 1984 for complete and final details. Whatever the final decisions may be, you WILL be able to purchase this product in April of 1984.

This new product will allow for complete transportability of software between the Models 4/4P and 2/12, along with any other machine that has 6.x implemented for it. A bit of caution here--- you MUST have used ONLY the official information contained in the Model 4 technical manual (26-2110) from Tandy, or information distributed by LSI when interfacing to the 6.x system to achieve this compatibility. Use of any other source of information on the 6.x system will most likely result in current or future incompatibilities. At this time, LSI and Tandy are the only official sources regarding technical specification for the TRSDOS 6.x and LS-DOS 6.x systems.

New Product Announcements

Remember LED, the LDOS EDitor? This is the official LDOS 5.1 Text Editor, and is used here at LSI for program source code maintenance, KSM file editing, and many other editing needs. Well, LED has been vastly enhanced and is now one of the best full screen editors available for your TRS-80 Model 4/4P (and in the future 2/12). LED is available now for $99 plus $3 shipping and handling as LS-LED for the 6.X operating system ONLY. We will NOT be doing a Model III version of this product (there just isn't enough room). If you use your Model 4 for programming, you will love this powerful text editor.

Note that LS-LED is not a word processor, but is a flexible, easy to use screen oriented text editor. LS-LED is capable of doing most word processor type functions, but there are no print formatting facilities, or indeed any printing capabilities (Of course, the DOS library command LIST with the "print" parameter can be used for hardcopy, if desired). Two major features that have been added to LS-LED are: writing a marked block to disk, and the insertion of the contents of a disk file at the current cursor position. With these two commands, the production and maintenance of subroutine libraries and other forms of "boiler-plate" operations become a snap.

Of course, the original version of LED for LDOS 5.1 is still available for $29 plus $3 shipping and handling.

LSI is now shipping LS-Host/Term, a comprehensive communications utility for the 6.X operating system. This package allows a Model 4 running under 6.x to emulate an ADDS-25 terminal, and provides for error-free file transfer between 6.x systems, or other systems supporting the Modem7 protocol. The host system may be unattended during the transfer. Speaking of hosting, the host capabilities of this package are very sophisticated, and include password protection and remote cursor positioning using two different protocols. LS-Host/Term is only $199 plus $3 shipping and handling.

FED, the most popular "zapping" program around for the LDOS system will be available for the IBM (or any MS-DOS 2.x, PC compatible) machine in March or April of 1984. The price will be $99, plus $3 shipping and handling. This is a very valuable tool for any serious user of an MS-DOS machine.

LSI will also introduce the most versatile data handling and management system available for the PC-DOS (and MS-DOS) world. This product will be announced in June of 1984, and more details will be available at that time.

LSI Quick Hint #1

Question: How can I move a file from LDOS 5.1 or TRSDOS 6.x to TRSDOS 1.3?

Quite simple, actually. Here is the procedure: First, under LDOS 5.1 or TRSDOS 6.x, format a five inch diskette as thirty-five track, one side, single-density. Now, copy the desired file to this special diskette. You may now re-boot under TRSDOS 1.3, and the TRSDOS 1.3 "CONVERT" utility will read this disk just as though it were a Model 1 TRSDOS 2.3 diskette.

Here's the catch-- some Model 4 computer systems have a newer type of controller board, and will not format properly in single-density with LDOS 5.1.3 or before, or TRSDOS 6.0. This problem was "bypassed" in software, and these machines will format correctly with LDOS 5.1.4 and TRSDOS 06.01.01.

EZEDIT/FLT - a command line edit utility

by Graham M Brown, 20 Paddock Close, Castle Donnington, Derby DE7 2JW, England

How many times have you typed in a long command line from DOS, only to find that you spelled a parameter incorrectly, and DOS asks you to do it all again? Well certainly I have done it enough times, and it is a nuisance. So EZ-Edit was born. Imagine typing:

```
FILTER *PR PR(M=10,I=10,C=80,L*60,P=66,F,T)
```

There is an error in the "L" parameter. With EZ-Edit, just type in <SHIFT><CLEAR><O> and your command comes back on the screen - position the flashing cursor over the character in error, correct it, press <ENTER>, and bingo!

EZ-Edit is a keyboard filter which intercepts a <SHIFT><CLEAR><O> and allows you to edit the previously entered command line. The left and right arrows move through the line, and extend it if required. Any other key will overtype existing text. <ENTER> will cancel editing, and process the command as shown. <SHIFT><CLEAR> will delete the command from the cursor to the right, and then execute the remainder. <BREAK> will return to DOS ready. In the above example, if the cursor was over the first open bracket, pressing <SHFT><CLR> would result in the command **FILTER *PR PR**.

Just as KSM does not work properly with MINIDOS (see LDOS Quarterly, Jan '83), EZ-Edit has a similar effect on the "R" command of MINIDOS. I have developed a similar patch for MINIDOS when EZ-Edit is running. EZ-Edit will perform this alteration as a temporary patch in memory when it loads. This program will, of course, work without MINIDOS. EZ-Edit takes less than 256 bytes when relocated in high memory.

EZ-Edit requires KI/DVR to be present and active. This program should work under all 5.1 releases of LDOS, but has only been tested on 5.1.3 and 5.1.4. Originally, the program was written to accept <clear><shift><E>, but this was changed to prevent a conflict with KSMPLUS. If desired, the value on line 2340 may be changed to any other valid character. The value (EF) is also underlined in this BINHEX listing.

```
0506 4544 4954 2020 0102 0052 D5DD E13A 2501 FE49 2048 2111 4422 A252 22AA 5221 2B44
2290 5222 E152 2125 4222 F253 228F 5422 9F54 22F6 5221 BE42 2298 5222 F052 2199 4222
A254 2189 4222 6252 218A 4222 1253 21D5 0022 7852 2120 0122 7F52 2115 40AF ED52 C217
5321 1C53 CD67 4421 1F44 CB66 CA0E 53CB 6E28 1E21 AC53 CD67 442A FA4D E511 D900 1936
12E1 1124 0119 EB21 0453 010A 00ED B0ED 5B16 403A 0F43 CB6F 2804 ED5B BE43 ED53 D953
01E1 002A 4940 5D54 AFED 4222 4940 23E5 ED53 CE53 1109 0019 2208 5422 1754 221B 54E1
E511 0A00 1922 0D54 2232 54E1 E511 0B00 1922 1154 2214 54D1 D521 CC53 EDB0 E1F3 3A0F
43CB 6F20 08DD 7501 DD74 0218 0322 BE43 FB06 3E21 1843 3E20 7723 10FC 3E0D 0102 0053
77C3 2D40 2110 00ED 7AF9 E1C9 0000 2175 53CD 7B44 C330 4021 8D53 18F5 455A 2D45 4449
5420 2D20 4C44 4F53 2063 6F6D 6D61 6E64 206C 696E 6520 6564 6974 6F72 2E20 5665 7273
696F 6E20 352E 310A 2863 2920 3139 3833 2062 7920 472E 4272 6F77 6E2E 2041 6C6C 2072
6967 6874 7320 7265 7365 7276 6564 0D0A 4B49 2F44 5652 206E 6F74 2069 6E73 7461 6C6C
6564 2021 0D0A 4669 6C74 6572 204F 4E4C 5920 7573 696E 6720 2A4B 4920 6465 7669 6365
210D 0A4D 494E 4944 4F53 2064 6574 6563 7465 6420 2D20 7061 7463 6869 6E67 2E2E 2E0D
180A 0000 0445 6469 7400 0000 CD00 00FE EFC0 E5D5 C5DD E5ED 5B20 407B E6C0 5FED 5320
40D5 D521 1843 7EFE 0D28 0623 CD33 0018 F53E 01AF 0054 0FCD 3300 D13E 5F32 D553 6F1A
32D6 53AD 32D7 5321 D753 3AD5 53AE 32D5 5312 0150 01D5 CD2B 00D1 B720 050B 78B1 20F3
28E3 F53A D653 12F1 FE08 2823 FE09 2815 FE0D 2835 FE1F 2821 FE01 285A FE20 38B7 FE80
30B3 127B E63F FE3F 3001 1318 A87B E63F FE01 3801 1B18 9E3E 1ED5 ED53 2040 CD33 003E
0FCD 3300 D17B F63F 5F1A 1BFE 2028 FA13 13ED 5320 407B E63F 4F06 00E1 1118 43ED B03E
0D12 CD33 00DD E1C1 D1E1 2118 43C3 0544 E1DD E1C1 D1E1 C32D 4002 0200 52
```

```
00100         TITLE '<EZEDIT/FLT by: G M Brown>'
00110 ;*=*=*
00120 ;Some code alteration is necessary in MINIDOS/FLT for
00130 ;the "R" command to function properly. So EZ-Edit
00140 ;should be installed AFTER MINIDOS. Should MINIDOS be
00150 ;installed after EZ-Edit, then the "R" command is not
00160 ;guaranteed to work at all.
00200 @KBD    EQU     002BH
```

```
@DSP    EQU     0033H
KIDVR   EQU     4016H
CURSOR  EQU     4020H
@EXIT   EQU     402DH
@ABORT  EQU     4030H
HIGH1$  EQU     4049H
HIGH3$  EQU     4411H
SFLAG1$ EQU     430FH
SFLAG3$ EQU     442BH
INBUF1$ EQU     4318H
INBUF3$ EQU     4225H
KIJCL1$ EQU     43BEH
KIJCL3$ EQU     42BEH
@CMNDI1 EQU     4405H
@CMNDI3 EQU     4299H
DFLAG1$ EQU     441FH
DFLAG3$ EQU     4289H
@DSPLY  EQU     4467H
@LOGOT1 EQU     447BH
@LOGOT3 EQU     428AH
;The first part of the coding loads an initial message
;checks that *KI has been referenced, and  for KI/DVR
;and MINIDOS being active.
        ORG     5200H           ;A good place to start
        PUSH    DE              ;Put *KI DCB into
        POP     IX              ;IX register for later
;       This section of code corrects all model specific
;       references in the code.
        LD      A,(0125H)
        CP      'I'             ;it's a 3 if true
        JR      NZ,CONT         ;go if Mod 1
        LD      HL,HIGH3$
        LD      (M1),HL
        LD      (M2),HL
        LD      HL,SFLAG3$
        LD      (M3),HL
        LD      (M4),HL
        LD      HL,INBUF3$
        LD      (M5),HL
        LD      (M6),HL
        LD      (M7),HL
        LD      (M15),HL
        LD      HL,KIJCL3$
        LD      (M8),HL
        LD      (M9),HL
        LD      HL,@CMNDI3
        LD      (M10),HL
        LD      HL,DFLAG3$
        LD      (M11),HL
        LD      HL,@LOGOT3
        LD      (M12),HL
        LD      HL,00D5H
        LD      (M13),HL
        LD      HL,0120H
        LD      (M14),HL
CONT    LD      HL,4015H        ;See if *KI specified
        XOR     A               ;in the filter command
        SBC     HL,DE
        JP      NZ,WRONGDV      ;error if not.
        LD      HL,LOGMSG       ;Point to message....
        CALL    @DSPLY          ;..and print it.
        LD      HL,DFLAG1$      ;System flag area
M11     EQU     $-2
```

```
00880         BIT     4,(HL)          ;Test for KI/DVR, and
00890         JP      Z,KINOTON       ;error if NOT set
00900         BIT     5,(HL)          ;Test for MINIDOS, and
00910         JR      Z,MDISOFF       ;bypass zapping if NOT there.
00930 ;If MINIDOS active, then it has to be zapped, otherwise
00940 ;this bit is skipped over.
00960         LD      HL,ZAPMSG       ;Point to message....
00970         CALL    @DSPLY          ;...and display it.
00980         LD      HL,(4DFAH)      ;Get MINIDOS address
00990         PUSH    HL              ;and save
01000         LD      DE,00D9H        ;Offset for zap
01010 M13     EQU     $-2
01020         ADD     HL,DE           ;get actual address, and
01030         LD      (HL),12H        ;zap MINIDOS
01040         POP     HL              ;restore start address
01050         LD      DE,0124H        ;for next offset
01060 M14     EQU     $-2
01070         ADD     HL,DE           ;get actual address, and
01080         EX      DE,HL           ;put in DE register pair
01090         LD      HL,TABLE        ;Point to zap table
01100         LD      BC,0AH          ;10 bytes to zap, so..
01110         LDIR                    ;...do it
01130 ;The next part of the code intercepts the *KI driver
01140 ;address, and stores EZ-Edit's start there. The old
01150 ;contents are loaded into EZ-Edit for continuation.
01170 ;High memory is also altered.
01190 ;EZ-Edit does not bother to indicate to the system (or
01200 ;other programs) that it has loaded. Indeed, by finding
01210 ;the proper header, it would be possible to get EZ-Edit
01220 ;to locate itself at the same address after a *KI reset.
01230 ;This has been left for you to do if you require.
01250 MDISOFF LD      DE,(KIDVR)      ;Get *KI driver address
01260         LD      A,(SFLAG1$)     ;See if JCL is
01270 M3      EQU     $-2
01280         BIT     5,A             ;active at present
01290         JR      Z,NO_DO         ;skip if not, else
01300         LD      DE,(KIJCL1$)    ;change DE to KIJCL$
01310 M8      EQU     $-2
01320 NO_DO   LD      (VECTAD),DE     ;Store in EZ-Edit
01330         LD      BC,LAST-START   ;Get length of filter
01340         LD      HL,(HIGH1$)     ;HL = Present High Memory
01350 M1      EQU     $-2
01360         LD      E,L             ;Put into register
01370         LD      D,H             ;pair DE
01380         XOR     A               ;Clear the carry
01390         SBC     HL,BC           ;get new High memory,
01400         LD      (HIGH1$),HL     ;and set it.
01410 M2      EQU     $-2
01420         INC     HL              ;Point to EZ-Edit start
01430         PUSH    HL              ;and save it
01440         LD      (OLDHI),DE      ;Put old HIGH$ in filter
01450         PAGE    OFF
01480 ;Relocate calls    and jumps in the filter
01490 ;Restore *KI DCB driver address, and relocate
01500 ;EZ-Edit to below HIGH$
01520         LD      DE,ONOFF-START  ;Get offset
01530         ADD     HL,DE           ;add to new start
01540         LD      (MODIFY2),HL    ;and modify
01550         LD      (MODIFY6),HL
01560         LD      (MODIFY7),HL
01570         POP     HL              ;recover start
01580         PUSH    HL              ;and keep going
01590         LD      DE,CONTENT-START
```

```
01600          ADD     HL,DE
01610          LD      (MODIFY3),HL
01620          LD      (MODIFY8),HL
01630          POP     HL
01640          PUSH    HL
01650          LD      DE,XORVAL-START
01660          ADD     HL,DE
01670          LD      (MODIFY4),HL
01680          LD      (MODIFY5),HL
01690          POP     DE              ;DE = Start of EZ-Edit
01700          PUSH    DE              ;save it.  DE = To
01710          LD      HL,START        ;and HL = From
01720          LDIR                    ;BC = Count, so move it.!
01730          POP     HL              ;recover EZ-Edit's start
01740          DI                      ;Don't interrupt a minute
01750          LD      A,(SFLAG1$)     ;Test for JCL again
01760 M4       EQU     $-2
01770          BIT     5,A
01780          JR      NZ,DO_ON        ;Skip if ACTIVE, else
01790          LD      (IX+01H),L      ;Load KIDCB with address
01800          LD      (IX+02H),H      ;of EZ-Edit
01810          JR      OUT
01820 DO_ON    LD      (KIJCL1$),HL    ;JCL, so load KIJCL$
01830 M9       EQU     $-2
01840 OUT      EI                      ;Interrupts on, and
01850          LD      B,62            ;how many spaces
01860          LD      HL,INBUF1$      ;point to inbuf$
01870 M15      EQU     $-2
01880          LD      A,20H           ;space
01890 HERE     LD      (HL),A          ;store it
01900          INC     HL              ;point to next
01910          DJNZ    HERE            ;go until done
01920          LD      A,0DH           ;CR
01930          LD      (HL),A          ;put at end of buffer
01940          JP      @EXIT           ; now back to DOS
01960 ;Table of zaps for MINIDOS zapping routine
01980 TABLE    DB      21H,10H,00H,0EDH,7AH,0F9H,0E1H,0C9H,00H,00H
02000 ;Error if KI/DVR not established
02020 KINOTON  LD      HL,KIOFFMS      ;Point to message
02030 GO_OUT   CALL    @LOGOT1         ;LOG and DISPLAY
02040 M12      EQU     $-2
02050          JP      @ABORT          ;Abnormal exit to DOS
02070 ;Error if *KI not referenced
02090 WRONGDV  LD      HL,NOTKIMS      ;Point to message
02100          JR      GO_OUT          ;Log, display, and DOS
02120 ; Messages
02140 LOGMSG   DB     'EZ-EDIT - LDOS command line editor. Version 5.1',0AH,'(c) 1983
by G.Brown. All rights reserved',0DH
02150 KIOFFMS  DB      0AH,'KI/DVR not installed !',0DH
02160 NOTKIMS  DB      0AH, 'Filter ONLY using *KI device! ',0DH
02170 ZAPMSG   DB      0AH,'MINIDOS detected - patching...',0DH
02190 ;EZ-Edit : The actual relocated filter
02200 ;The filter conforms to the LDOS standard header.
02230 START    JR      BEGIN
02240 OLDHI    DW      0000H
02250          DB      4,'Edit'
02270 ;Local storage area
02290 ONOFF    DB      00H             ;Flash character store
02300 CONTENT  DB      00H             ;The contents of DE before flash
02310 XORVAL   DB      00H             ;The XOR character for the flash
02320 BEGIN    CALL    0000H           ;Call to KI/DVR, and MINIDOS
02330 VECTAD   EQU     $-2
02340          CP      0EFH            ;See if SHIFT+CLEAR+O
```

```
02350         RET     NZ              ;Ret if not
02370 ;Got a SHIFT+CLEAR+O here
02390         PUSH    HL              ;Save what we use
02400         PUSH    DE
02410         PUSH    BC
02420         PUSH    IX
02430         LD      DE,(CURSOR)     ;Get the cursor location
02440         LD      A,E             ;and make sure that
02450         AND     0C0H            ;its at the far left of
02460         LD      E,A             ;the screen.
02470         LD      (CURSOR),DE
02480         PUSH    DE              ;Save the address of the
02490         PUSH    DE              ;line - TWICE
02510;Get the last command from INBUF$, and put on screen
02520;Turn the cursor off afterwards.
02540         LD      HL,INBUF1$      ;Point to input buffer
02550 M5      EQU     $-2
02560 GETLOOP LD      A,(HL)          ;and get all characters
02570         CP      0DH             ;up to but NOT including
02580         JR      Z,GOT_CR        ;a CR, and put on the
02590         INC     HL              ;screen
02600         CALL    @DSP
02610         JR      GETLOOP
02620 GOT_CR  LD      A,0FH           ;Cursor OFF now
02630         CALL    @DSP
02640         POP     DE              ;Restore line address
02650         PAGE    OFF
02670 ;Scan for an input, and flash the character with the
02680 ;cursor (CHR$(95)). Allowable inputs are:
02690 ;LEFT ARROW  - backspace without erase
02700 ;RIGHT ARROW - cursor right without erase
02710 ;SHIFT+CLEAR - erase all from cursor to the right and
02720 ;              process the command to the left.
02730 ;ENTER       - process the line  as is. Characters to the
02740 ;              left and right of the cursor make up the
02750 ;              command
02760 ;BREAK       - Back to DOS
02780 FLASH   LD      A,5FH           ;Cursor character
02790         LD      (ONOFF),A       ;store it
02800 MODIFY2 EQU     $-2
02810         LD      L,A             ;and save in L
02820         LD      A,(DE)          ;Get character in DE
02830         LD      (CONTENT),A     ;and save for later
02840 MODIFY3 EQU     $-2
02850         XOR     L               ;XOR to get flash value
02860         LD      (XORVAL),A      ;and save it.
02870 MODIFY4 EQU     $-2
02890 TEXTLP  LD      HL,XORVAL       ;Point to store area
02900 MODIFYS EQU     $-2
02910         LD      A,(ONOFF)       ;Get one character
02920 MODIFY6 EQU     $-2
02930         XOR     (HL)            ;XOR
02940         LD      (ONOFF),A       ;save back for flash
02950 MODIFY7 EQU     $-2
02960         LD      (DE),A          ;and put on screen
02970         LD      BC,150H         ;Timer for scanning
02980 LOOP    PUSH    DE              ;save DE
02990         CALL    @KBD            ;and look at the keyboard
03000         POP     DE              ;get DE again
03010         OR      A               ;Any input ???
03020         JR      NZ,INPUT        ;Jump if so, else....
03030         DEC     BC              ;see if BC = 0
03040         LD      A,B
```

```
03050         OR      C
03060         JR      NZ,LOOP          ;Loop if not, else...
03070 INPUT   JR      Z,TEXTLP         ;no input = back to flash
03080         PUSH    AF               ;save input
03090         LD      A, (CONTENT)     ;restore contents to
03100 MODIFY8 EQU     $-2              ;original character
03110         LD      (DE),A
03120         POP     AF               ;recover input, and
03130         CP      08H              ;is it LEFT ARROW ??
03140         JR      Z,LARROW
03150         CP      09H              ;RIGHT ARROW ??
03160         JR      Z,RARROW
03170         CP      0DH              ;ENTER ??
03180         JR      Z,ENTPRES
03190         CP      1FH              ;SHIFT+CLEAR ??
03200         JR      Z,CLEARPR
03210         CP      01H              ;BREAK  ??
03220         JR      Z, BREAK
03230         CP      20H              ;Is it ASCII 2$H-7FH
03240         JR      C,FLASH
03250         CP      80H
03260         JR      NC,FLASH
03270         LD      (DE),A           ;if so, then put on
03280 RARROW  LD      A,E              ;the screen. Test for
03290         AND     3FH              ;extreme right, else
03300         CP      3FH              ;increment DE (type to
03310         JR      NC,NOADD         ;the right).
03320         INC     DE
03330 NOADD   JR      FLASH            ;Back for next input.
03350 ;Deal with the input. Editing allowed on one line only.
03370 LARROW  LD      A,E              ;Test for column 1
03380         AND     3FH              ;If less, then cannot
03390         CP      1                ;move further left.
03400         JR      C,NODEC
03410         DEC     DE               ;else DECrement
03420 NODEC   JR      FLASH
03430 CLEARPR LD      A,1EH            ;Erase to end of line
03440         PUSH    DE               ;Save DE for a moment
03450         LD      (4020H),DE       ;tell the cursor where,
03460         CALL    @DSP             ;and clear the line.
03470         LD      A,0FH            ;Cursor OFF
03480         CALL    @DSP
03490         POP     DE               ;and restore DE
03510 ;How long is the command ?? - go to right of line, and
03520 ;find first non blank character on the left.
03540 ENTPRES LD      A,E              ;Point DE to end of
03550         OR      3FH              ;the line
03560         LD      E,A
03570 BKLOOP  LD      A,(DE)           ;and look back....
03580         DEC     DE
03590         CP      20H
03600         JR      Z,BKLOOP
03610         INC     DE
03620         INC     DE               ;DE = space after command
03630         LD      (CURSOR),DE      ;load the cursor
03650 ;Move the new command into INBUF$, and execute via
03660 ;@CMNDI address.
03680         LD      A,E              ;Get column number into A
03690         AND     3FH
03700         LD      C,A              ;Load length of the
03710         LD      B,0              ;command into BC
03720         POP     HL               ;Point HL to line start
03730         LD      DE,INBUF1$       ;DE = Destination
```

```
M6      EQU     $-2
        LDIR                    ;Move the command
        LD      A,0DH           ;finish off with a CR
        LD      (DE),A          ;in the buffer, and....
        CALL    @DSP            ;on the screen.
        POP     IX              ;Restore original
        POP     BC              ;register values
        POP     DE
        POP     HL
        LD      HL,INBUF1$      ;Point to new command
M7      EQU     $-2
        JP      @CMNDI1         ;and execute it.
M10     EQU     $-2
BREAK   POP     HL              ;Keep the stack tidy
        POP     IX
        POP     BC
        POP     DE
        POP     HL
        JP      @EXIT           ;and exit to DOS
LAST    EQU     $
        END     5200H
```

Mr. Brown also sent us the following program to alter PR/FLT parameters "on the fly". Unfortunately, we don't have room to run the source code. If you desire a hardcopy of the source, send us a request along with a self-addressed, stamped envelope (large, with 37 cents postage)

Although PR/FLT as supplied with LDOS is a very fine general purpose printer filter, I have found it somewhat restricting in that there is no facility for changing any of the parameters once set. In fact, should you wish to change the margin, it would be necessary first to issue a RESET *PR, followed by re-filtering PR/FLT with your new parameter values in the command line (as normal). This is all very well, but if you had your printer output routed, or indeed had other printer filters installed, the RESET would ruin everything. Certainly from my point of view, I needed a program that would find where PR/FLT was in memory and then change any of the parameters that I wanted.

The program is called from DOS ready by entering the command:

```
  PRPARM (parm,parm,parm,...)
```

where parameters are nearly the same as for PR/FLT:

MARGIN	number of spaces for the left margin
INDENT	-:- -:- indent on line wrap
CHARS	number of characters per line
LINES	number of printed lines per page required
PAGE	page length in lines (at 6 lines per inch)
XLATE	use the format XLATE=X'aabb' (see the PR/FLT documentation)
FF	issue a form feed

Abbr: MARGIN=M,INDENT=I,CHARS=C,LINES=L,PAGE=P,XLATE=X

Example: PRPARM (M=30,C=50,FF)

Run PRPARM and reset the left margin to 30 and only print 50 characters on each line. Issue a form feed character to the *PR device as well.

A typical display would be:

```
PRPARM - parameter modifier for PRIFLT. Version 5.1.3
Copyright (c) 1983 by Graham M Brown.
```

```
The current PR/FLT parameters are
MARGIN = 010 spaces
INDENT = 036 spaces
CHARS  = 082 characters per line (0 = no count made.)
PAGE LENGTH = 066 lines
LINES/PAGE  = 066 lines
000 (dec) is being translated to 000 (dec).
```

By typing in PRPARM only, the current parameters as stored in PR/FLT are displayed and no alterations are done. A check is made for LDOS 5.1.x and also that PR/FLT is in fact installed. The program is written for the Model 1 and 3. Here is the BINHEX code:

```
0506 5052 5041 524D 0102 0052 E521 6353 CD67 443A 2501 FE49 2031 211F 4422 4052 2189
4222 4852 2154 4422 6C52 3E01 32C8 5221 8A42 225E 533A FF52 3D32 FF52 32B7 523A 0853
3D32 0853 32BA 523A 3E40 FE51 C256 5321 1F44 CB5E CA5A 53FD 2AF6 4DE1 7EFE 0D28 07FE
2020 0B23 18F4 3E0D 3207 55C3 CC52 116B 55CD 7644 C252 53DD 2125 4021 FFFF CD1E 5328
03FD 7717 21FF FFCD 1E53 2803 FD77 1921 FFFF CD1E 5328 03FD 771A 21FF FFCD 1E53 2803
DD77 0521 FFFF CD1E 5328 03DD 7703 21FF FF23 7CB5 2807 2BFD 746D FD75 7121 0000 7CB5
289E 3E0C CD3B 003E 00DD 7704 2600 FD6E 1711 A554 CD2F 53FD 6E19 11BE 54CD 2F53 FD6E
1A11 8C54 CD2F 533A 2840 6F11 FE54 CD2F 533A 2A40 6F11 3055 CD2F 53FD 6E6D 0102 0053
113F 55CD 2F53 FD6E 7111 6055 CD2F 5321 5854 CD67 4421 1D55 CD67 44C3 2D40 2324 2520
2E7C B5C8 287D C964 000A 0001 00DD 2129 53AF DD46 01DD 4E00 B7ED 4238 033C 18F9 09C6
3012 1379 FE01 C8DD 23DD 2318 E2E1 2107 54DD 21C9 53DD 21E7 53CD 7B44 C330 400A 5052
5041 524D 202D 2070 6172 616D 6574 6572 206D 6F64 6966 6965 7220 666F 7220 5052 2F46
4C54 2E20 5665 7273 696F 6E20 352E 312E 330A 8383 8383 8383 2020 2043 6F70 7972 6967
6874 2028 6329 2031 3938 3320 6279 2047 7261 6861 6D20 4D20 4272 6F77 6E2E 0D0A 0A50
6C65 6173 6520 7573 6520 4C44 4F53 2035 2E31 2E33 206F 6E6C 792E 0D0A 0A50 522F 464C
5420 6973 206E 6F74 2079 6574 2069 6E73 7461 0102 0054 6C6C 6564 2021 0D0A 0A50 6172
616D 6574 6572 2065 7272 6F72 202D 2074 7279 2061 6761 696E 2021 0A41 6C6C 6F77 6162
6C65 2070 6172 616D 6574 6572 7320 6172 653A 204D 2C49 2C43 2C4C 2C50 2C46 462C 616E
6420 584C 4154 450D 0A54 6865 2063 7572 7265 6E74 2050 522F 464C 5420 7061 7261 6D65
7465 7273 2061 7265 203A 0A0A 2020 2020 204D 4152 4749 4E20 3D20 5858 5820 7370 6163
6573 0A20 2020 2020 494E 4445 4E54 203D 2058 5858 2073 7061 6365 730A 2020 2020 2043
4841 5253 2020 3D20 5858 5820 6368 6172 6163 7465 7273 2070 6572 206C 696E 6520 2830
203D 206E 6F20 636F 756E 7420 6D61 6465 2E29 0A20 2020 2020 5041 4745 204C 454E 4754
4820 3D20 5858 01D6 0055 5820 6C69 6E65 7320 2D20 544F 4620 6861 7320 6265 656E 2072
6573 6574 0D20 2020 2020 4C49 4E45 532F 5041 4745 2020 3D20 5858 5820 6C69 6E65 730A
2020 2020 2058 5858 2028 6465 6329 2069 7320 6265 696E 6720 7472 616E 736C 6174 6564
2074 6F20 5858 5820 2864 6563 292E 0D4D 2020 2020 208C 524D 4152 4749 4E8C 5243 4841
5253 2081 5243 2020 2020 2081 5249 4E44 454E 5476 5249 2020 2020 2076 524C 494E 4553
2097 524C 2020 2020 2097 5250 4147 4520 20A2 5250 2020 2020 20A2 5246 4620 2020 20BC
5258 4C41 5445 20AD 5258 2020 2020 20AD 5200 0202 0052
```

<u>SYNONYM -- The LDOS Command Line Synonym Processor</u>

By Henry Melton, 2511 Dovemeadow Drive, Austin, TX 78744

The world of the professional programmer can get positively confusing at times. In the course of a single business day, I am required to deal with a half-dozen or so different operating systems, some quite modern and some positively ancient. And then I come home to work with my LDOS system . . . Not only do I have to constantly adjust to differing keyboards and screen formats, but more significantly, I have to remember which operating system command does what. Just to get a listing of files, my fingers might type LIST, DIR, FILES, CAT, LISTC, or L. Some of the newer systems are helping us poor users by allowing command synonyms to be used, so that more than one reserved word might be used for the same function. I decided to add that same capability to my LDOS.

The SYNONYM utility is an addition to the error trapping system of LDOS that allows for extended flexibility in operator actions for Model I and Model III TRS-80 systems running under LDOS 5.1. SYNONYM locates itself into high memory following the standard LDOS convention and is SYSGENable. Once loaded, it intercepts any PROGRAM NOT FOUND errors generated from keyboard or JCL command line inputs. Once the error is detected, a special text file SYNONYM/TXT is read and the match-and-substitute record lines in it are used to determine if the rejected command line has a valid substitute. If so, the reconstructed command line is placed in the command buffer and re-executed.

The result of SYNONYM processing is to add another level to the two existing levels of command interpretation. Under LDOS, the first word of a command line is first checked for an LDOS reserved word. If none of the reserved words match, then the word is treated as the name of a command file to be executed. With SYNONYM, if the first two levels fail, this third possibility for command information exists.

Like the JCL language, much of the utility of SYNONYM is up to the imagination of the user. The synonym library file can be created and modified with any word-processor that will produce a plain, un-numbered file of text lines.

Each record line in the file contains three items of information:

The first character of each line is a single numeric digit in the range of 1 through 9. This is the minimum number of characters that must be the same to indicate a match.

The second item is a word that is allowed as a valid synonym.

The third item is the remaining text on the line. This is the text that will replace the rejected command line. The special character '&' may be used to represent the text of the original command line after the first word.

Here are some examples:

Command line	SYNONYM record	Resulting Command line
fi /vc	2 files dir &	dir /vc
f	1 free free &	free
calc 365/12	4 calc lbasic ?&:cmd"S"	lbasic ?365/12:cmd"S"
c01 syn/mac	3 c$1 copy &:$:1	copy syn/mac:0 :1
d0	2 d$ dir &:$	dir :0
d1 sample	2 dl dir &:1	dir sample:1
bk work	2 bk copy & bk&:3	copy work bkwork:3
a syn	3 asm do it(@asm,name=&)	do it(@asm,name=syn)
ban	3 banner lbasic run "banner"	lbasic run"banner"

If there is no acceptable synonym, then the PROGRAM NOT FOUND error is displayed normally. If the resulting synonym command is itself invalid, the whole process repeats using the new command line as input.

Synonyms can be used within JCLs and synonyms may call JCLs. In fact, using a DO command line as a synonym may easily be the most powerful use of this utility. My personal JCL file (IT/JCL) is quite large, but remembering all the parameters and syntax considerations has been my main trouble. When I want to invoke a function of the system, I really don't want to puzzle over whether I am invoking a system function, a /CMD program, a LBASIC program or a JCL -- I want type the word and have it done. SYNONYM makes this a reality.

Another use of synonym is to execute one-liners from LBASIC. See CALC and REVTOF in the sample SYNONYM file to see what I mean. LBASIC is powerful. The ability to use its power at LDOS Ready is very handy at times. The limitations on this use is that the function has to fit within LBASIC and :CMD"S, leaving only 50 characters to get the job done. Anything longer has to route through a JCL or use a real /BAS program file. To keep screen clutter to a minimum, I patched LBASIC on my Model I system to eliminate the sign-on banner when LBASIC is executed.

```
. Patch to remove the execution banner from LBASIC - Model 1
X'541E'=00 00 00 00 00 00

. Patch for Model 3
X'5446'=00 00 00 00 00 00
```

Sample SYNONYM/TXT file -- Total line length should not exceed 80 characters. The resulting command line should not exceed 63 characters.

```
2 dirt dir &
2 files dir &
1 asm do it(@asm,name=&)
2 debug debug (e
3 out debug (n
3 off system (drive=3,disable)
3 onn system (drive=3,enable)
1 v dir /vc
1 d dir /aaa
1 bye do it(@back)
1 free free &
2 d1 dir :1 &
2 d0 dir :0 &
3 device device
2 looker lbasic run"looker
3 c01 copy &:0 :1
3 c03 copy &:0 :3
1 hex list & (H,LRL=256)
3 nosynonym memory (add=x'400d',word=X'4bcd')
5 synoff nos
2 peek memory (add=x'&')
3 zap purge & (S,I,Q=N)
4 hide attrib & (inv)
3 unhide attrib & (vis)
4 calc lbasic ?&:cmd"S"
3 tof lbasic lprint chr$(12);:cmd"s"
6 revtof lbasic for i=0 to 65:lprint chr$(27)+chr$(13);:next i:cmd"S
3 jmp memory (go=X'&')
3 cls jmp 1C9
3 erase kill &
3 delete kill &
2 type list &
3 catalog dir &
4 page tof
3 num list & (num,A8
3 tab list & (num,tab,A8
3 slow filter *do dospeed
4 fast reset *do
3 b01 backup &:0 :1
3 b03 backup &:0 :3
5 name0 attrib :0 (name="&")
5 name1 attrib :1 (name="&")
7 format1 format :1 (name="&",dden,abs,mpw="PASSWORO")
4 sys1 system (system=1)
6 sysgen system (sysgen)
8 clonesys do it(@clonesys)
```

Here is the BINHEX code for SYNONYM. If you wish the "processed" DOS command to be displayed on execution, replace the <u>0000 00</u> with <u>CD67 44</u>.

```
0506 5359 4E4F 4E59 0102 0052 21AD 52CD 6744 3A2D 40FE C320 0921 FC52 CD67 44C3 2D40
3A25 01FE 4920 2F21 1144 2251 5222 7952 2285 5221 1D42 2294 5222 A252 2322 9A52 22A8
5221 2542 229E 5322 7B54 2B22 F953 2199 4222 A453 DD21 2E53 2A49 4022 5C53 1156 56B7
ED52 444D 3E16 DD6E 00DD 6601 235E 2356 EB09 EB72 2B73 DD23 DD23 3D20 E9ED 5B49 4021
5656 01FD 02ED B8ED 5349 402A 0D40 226B 5321 6453 220D 403A 0343 3292 542A 0443 2293
543E C332 0343 218C 5422 0443 C32D 4053 594E 4F4E 594D 202D 2D20 4C44 4F53 2063 6F6D
6D61 6E64 206C 696E 6520 7379 6E6F 6E79 6D20 7072 6F63 6573 736F 7220 0A43 6F70 7972
6967 6874 6564 2031 3938 332C 2048 656E 7279 204D 656C 746F 6E0D 4F6E 6C79 0102 0053
2076 616C 6964 2061 7420 4C44 4F53 2052 6561 6479 202D 2069 6E73 7461 6C6C 6174 696F
```

```
6E20 6162 6F72 7465 6421 0A0D 8D52 8A52 8C54 9652 E053 9C52 A452 8553 8A53 8F53 9653
9953 A653 A953 AC53 B553 B853 CD53 D053 EB53 2954 7754 1808 0000 0553 594E 4F4E F5FE
8628 04F1 C300 0033 3333 33F1 FE5F 2809 F53E 863B 3B3B 3B18 EA3B 3B3B 3B3B 3BCD A653
20DF CDCD 5320 DACD EB53 2802 18F4 CD25 54CD 7754 F121 1843 0000 00C3 0544 CDB5 5321
C754 1195 5406 00CD 2444 C921 C153 1195 5401 0C00 EDB0 C953 594E 4F4E 594D 2F54 5854
0D21 0756 1195 54CD 1300 C077 FE0D C8FE 2038 F1E5 1157 56ED 52E1 28E8 2318 E521 0756
7ED6 3047 237E FE20 28FA 1117 4313 1AFE 2028 0197 0054 FA1A AEE6 DFC0 2313 10F7 1AFE
0DC8 FE20 C87E FE20 280D FE0D 2809 1AAE E6DF C023 1318 E7B7 C9DD E5D5 E521 C755 D11A
FE20 2803 1318 F813 1AFE 2028 FA77 FE0D 2833 FE26 2805 2313 1A18 F2DD E1DD E5DD 7E00
FE20 2806 381A DD23 18F3 DD23 DD7E 00FE 2038 0D28 F577 23DD 23DD 7E00 FE20 30F5 2B18
D1D1 DDE1 C921 C755 1118 4306 3F7E 12FE 20D8 2313 10F7 3E0D 12C9 2164 5322 0D40 C900
0001 0557 5648 4A4D 0202 0052
```

```
00110 ; SYNONYM -- Command line synonym processor for LDOS
00120 ;      Copyright 1983 by Henry Melton
00140 ;  The synonym processor resides in high memory and is linked into the error
00150 ;  reporting chain. When the error code 95 appears, indicating that a Program not
00160 ;  found error is to be flagged, SYNONYM reads the command buffer and uses the
00170 ;  first word of the contents as a substitution key for an acceptable alternate
00180 ;  command string.  A file with the name SYNONYM/TXT is needed that contains the
00190 ;  substitution data.
00460 ;  Synonym records are searched sequentially til either a match or EOF occurs. If
00470 ;  the new command line fails, then the processor tries again with this new
00480 ;  input. When & substitution occurs, there is the possibility of the resulting
00490 ;  command line exceeding the 64 character command buffer. If so, the line will
00500 ;  be truncated to 64 characters.
00530 ECHO      EQU       1          ;Display the generated command = 1
00550 @DSPLY    EQU       4467H
00560 @EXIT     EQU       402DH
00570 @GET      EQU       0013H
00580 @OPEN     EQU       4424H
00590 ; Model 1 Equates
00600 HIGH1$    EQU       4049H
00610 INBUF1$   EQU       4318H
00620 @ICNFG1   EQU       4303H
00630 @CMNDI1   EQU       4405H
00640 ; Model 3 Equates
00650 HIGH3$    EQU       4411H
00660 INBUF3$   EQU       4225H
00670 @ICNFG3   EQU       421DH
00680 @CMNDI3   EQU       4299H
00690 ;
00700           ORG       5200H
00710 START::   LD        HL,BANNER      ;Display the program name and copyright
00720           CALL      @DSPLY
00730           LD        A,(@EXIT)
00740           CP        0C3H              ;is it a 'jump'?
00750           JR        NZ,OKGO           ;if so, we aren't at LDOS Ready
00760           LD        HL,ABORTMS        ;and can't install/modify the system
00770           CALL      @DSPLY
00780           JP        @EXIT
00790 OKGO:     LD        A,(0125H)         ;pick up type of machine flag
00800           CP        'I'               ;will be "I" if Model 3
00810           JR        NZ,MOD1           ;continue if Mod 1
00820           LD        HL,HIGH3$         ;Pick up Mod 3 locations and
00830           LD        (MODCH1),HL       ;adjust all references
00840           LD        (MODCH2),HL
00850           LD        (MODCH3),HL
00860           LD        HL,@ICNFG3
```

```
00870           LD        (MODCH4),HL
00880           LD        (MODCH6),HL
00890           INC       HL
00900           LD        (MODCH5),HL
00910           LD        (MODCH7),HL
00920           LD        HL,INBUF3$
00930           LD        (MODCH8),HL
00940           LD        (MODCH11),HL
00950           DEC       HL
00960           LD        (MODCH10),HL
00970           LD        HL,@CMNDI3
00980           LD        (MODCH9),HL
01000 ; all Mod 1/3 conversion is done, let's get on with the good stuff...
01020 MOD1:     LD        IX,RELOTB        ;Using the relocation table and the
01030           LD        HL,(HIGH1$)      ;High Memory location, patch the
01040 MODCH1    EQU       $-2
01050           LD        (STORE),HL       ;load module for operation in high
01060           LD        DE,LAST-1        ;memory.
01070           OR        A
01080           SBC       HL,OE
01090           LD        B,H
01100           LD        C,L
01110           LD        A,ENTRYS
01120 PTLOOP:   LD        L,(IX+00H)
01130           LD        H,(IX+01H)
01140           INC       HL
01150           LD        E,(HL)
01160           INC       HL
01170           LD        D,(HL)
01180           EX        DE,HL
01190           ADD       HL,BC
01200           EX        DE,HL
01210           LD        (HL),D
01220           DEC       HL
01230           LD        (HL),E
01240           INC       IX
01250           INC       IX
01260           DEC       A
01270           JR        NZ,PTLOOP
01290           LD        DE,(HIGH1$)      ;Move the module to its high memory
01300 MODCH2    EQU       $-2
01310           LD        HL,LAST-1        ;location
01320           LD        BC,LAST-FIRST
01330           LDDR
01350           LD        (HIGH1$),DE      ;Put the new HIGH$ value in storage
01360 MODCH3    EQU       $-2
01370           LD        HL,(400DH)
01380 REL0:     LD        (VECTOR),HL      ; Plug the primary RST 40
01390 REL1:     LD        HL,BEGIN         ; vector with the SYN processor
01400           LD        (400DH),HL       ; address.
01410           LD        A,(@ICNFG1)      ; Do the same for the CONFIG
01420 MODCH4    EQU       $-2
01430 RELX:     LD        (CMD),A          ; initialization chain so the
01440           LD        HL,(@ICNFG1+1)   ; SYN processor can be SYSGENed
01450 MODCH5    EQU       $-2
01460 RELY:     LD        (ADDRES),HL
01470           LD        A,0C3H
01480           LD        (@ICNFG1),A
01490 MODCH6    EQU       $-2
01500 RELZ:     LD        HL,INIT
01510           LD        (@ICNFG1+1),HL
01520 MODCH7    EQU       $-2
01530           JP        @EXIT            ;end of the loading operation
```

```
BANNER:   DEFM      'SYNONYM -- LDOS command line synonym processor '
          DEFB      0AH
          DEFM      'Copyrighted 1983, Henry Melton'
          DEFB      0DH
ABORTMS:  DEFM      'Only valid at LDOS Ready - installation aborted!'
          DEFB      0AH
          DEFB      0DH
RELOTB:   DEFW      REL1       ;Table of addresses to relocate to HIGH$
          DEFW      REL0
          DEFW      INIT
          DEFW      RELX
          DEFW      RELXX
          DEFW      RELY
          DEFW      RELZ
          DEFW      INTERCPT
          DEFW      LOOP1
          DEFW      REL2
          DEFW      PROCESS
          DEFW      REL3
          DEFW      OPENSYN
          DEFW      REL4
          DEFW      REL5
          DEFW      LOADFCB
          DEFW      REL7
          DEFW      READLN
          DEFW      LOOP2
          DEFW      COMPARE
          DEFW      REL8
TBLEND:   DEFW      MOVELN
TBSIZE    EQU       TBLEND-RELOTB+2
ENTRYS    EQU       TBSIZE/2
FIRST:    JR        BEGIN
STORE:    DEFW      0              ;to receive the old HIGH$ value
NAME:     DEFB      BEGIN-TEXT
TEXT:     DEFM      'SYNON'
BEGIN:    PUSH      AF
          CP        86H               ;A true error code has
          JR        Z,CHECK2          ; an 86H here.
QUIT:     POP       AF                ; so if not, go about
          DEFB      0C3H              ; your business.
VECTOR:   DEFW      0
CHECK2:   INC       SP
          INC       SP
          INC       SP
          INC       SP
          POP       AF
          CP        95                ; A PROGRAM NOT FOUND
          JR        Z,INT             ; error will have a 95
          PUSH      AF                ; here.
          LD        A,86H
          DEC       SP
          DEC       SP
          DEC       SP
          DEC       SP
          JR        QUIT
INT:      DEC       SP                ; make sure the stack
          DEC       SP                ; is ordered properly
          DEC       SP                ; to minimize side effects
          DEC       SP
          DEC       SP
          DEC       SP
INTERCPT: CALL      OPENSYN           ; open SYNONYM/TXT
          JR        NZ,QUIT
```

```
02200 LOOP1:    CALL      READLN            ; read in a line
02210           JR        NZ,QUIT
02220 REL2:     CALL      COMPARE           ; test for a match
02230           JR        Z,PROCESS         ; if so, process it
02240           JR        LOOP1             ; or else loop
02250 PROCESS:  CALL      BUILDLN           ;build the replacement
02260 REL3:     CALL      MOVELN            ; command line and move
02270           POP       AF                ; it back into the
02280           LD        HL,INBUF1$        ; command buffer and
02290 MODCH8    EQU       $-2
02300           IF        ECHO              ; execute it.
02310           CALL      @DSPLY
02320           ELSE
02330           NOP
02340           NOP
02350           NOP
02360           ENDIF
02370           JP        @CMNDI1
02380 MODCH9    EQU       $-2
02400 OPENSYN:  CALL      LOADFCB
02410 REL4:     LD        HL,DSKBUF
02420 REL5:     LD        DE,FCB
02430           LD        B,0
02440           CALL      @OPEN
02450           RET
02470 LOADFCB:  LD        HL,FILENAME
02480 REL7:     LD        DE,FCB
02490           LD        BC,CR-FILENAME+1
02500           LDIR
02510           RET
02530 FILENAME: DEFM      'SYNONYM/TXT'
02540 CR:       DEFB      13
02560 READLN:   LD        HL,LINEBF         ;Read a line of text into
02570 LOOP2:    LD        DE,FCB            ; a local buffer until
02580           CALL      @GET              ; a CR, skipping all nulls
02590           RET       NZ                ; or other control characters.
02600           LD        (HL),A
02610           CP        13
02620           RET       Z
02630           CP        20H
02640           JR        C,LOOP2
02650           PUSH      HL
02660 RELXX:    LD        DE,LAST           ; Make sure the text does
02670           SBC       HL,DE             ; not exceed the line
02680           POP       HL                ; buffer.
02690           JR        Z,LOOP2
02700           INC       HL
02710           JR        LOOP2
02730 COMPARE:  LD        HL,LINEBF         ;Get the match count
02740           LD        A,(HL)            ; from the first
02750           SUB       30H               ; character in the
02760           LD        B,A               ; synonym record line.
02770 LOOP3:    INC       HL
02780           LD        A,(HL)            ; skip spaces in the
02790           CP        20H               ; syn record
02800           JR        Z,LOOP3
02810           LD        DE,INBUF1$-1
02820 MODCH10   EQU       $-2
02830 LOOP4:    INC       DE
02840           LD        A,(DE)            ; skip leading spaces
02850           CP        20H               ; in the original
02860           JR        Z,LOOP4           ; command line
02870 LOOP5:    LD        A,(DE)
```

```
02880           XOR       (HL)              ;compare loop.
02890           AND       0DFH              ; ignoring upper/lower
02900           RET       NZ                ; case distinctions,
02910           INC       HL                ; compare for the full
02920           INC       DE                ; match count, rejecting
02930           DJNZ      LOOP5             ; for any mismatch.
02940 LOOP5A:   LD        A,(DE)
02950           CP        13                ; After the count,
02960           RET       Z                 ; if the original ends
02970           CP        20H               ; first, match is okay.
02980           RET       Z
02990           LD        A,(HL)            ; But if the SYN key
03000           CP        20H               ; ends first, then the
03010           JR        Z,NOMATCH         ; match is rejected.
03020           CP        13
03030           JR        Z,NOMATCH
03040           LD        A,(DE)
03050           XOR       (HL)
03060           AND       0DFH
03070           RET       NZ
03080           INC       HL
03090           INC       DE
03100           JR        LOOP5A
03110 NOMATCH:  OR        A
03120           RET
03140 BUILDLN:  PUSH      IX            ;Save IX on general principles.
03150           PUSH      DE            ;Save the pointer into INBUF$
03160           PUSH      HL            ;Save the pointer into the SYN record
03170 REL8:     LD        HL,NEWLN      ;Start at the beginning of the
03180           POP       DE            ; new line buffer,
03190 LOOP6:    LD        A,(DE)        ; skip until a space
03200           CP        20H
03210           JR        Z,LOOP7
03220           INC       DE
03230           JR        LOOP6
03240 LOOP7:    INC       DE            ; skip spaces until replacement
03250           LD        A,(DE)        ; text is encountered.
03260           CP        20H
03270           JR        Z,LOOP7
03280 LOOP8:    LD        (HL),A        ; copy replacement text
03290           CP        13            ; until EOL
03300           JR        Z,BLDEX
03310           CP        '&'           ; if    '&' is in replacement
03320           JR        Z,SUBSTIT     ; text then substitute.
03330 RETUR:    INC       HL
03340           INC       DE
033S0           LD        A,(DE)
03360           JR        LOOP8
03380 SUBSTIT:  POP       IX            ;Pick up the pointer to the
03390           PUSH      IX            ; remainder of the original
03400 LOOP9:    LD        A,(IX)        ; command line, and copy it
03410           CP        20H           ; to the new command line.
03420           JR        Z,LOOP10
03430           JR        C,NOSUB       ; Skip leading spaces.
03440           INC       IX            ; If no no-space characters,
03450           JR        LOOP9         ; then & vanishes.
03460 LOOP10:   INC       IX
03470           LD        A,(IX)
03480           CP        20H
03490           JR        C,NOSUB
03500           JR        Z,LOOP10
03510 LOOP11:   LD        (HL),A
03520           INC       HL
```

```
                INC       IX
                LD        A,(IX)
                CP        20H
                JR        NC,LOOP11
      NOSUB:    DEC       HL
                JR        RETUR
      BLOEX:    POP       DE
                POP       IX
                RET
      MOVELN:   LD        HL,NEWLN
                LD        DE,INBUF1$    ; copy up to 63 text
      MODCH11   EQU       $-2
                LD        B,63          ; characters and one
      LOOP12:   LD        A,(HL)        ; CR to the original
                LD        (DE),A        ; command buffer.
                CP        20H
                RET       C
                INC       HL
                INC       DE
                DJNZ      LOOP12
                LD        A,13
                LD        (DE),A
                RET
      INIT:     LD        HL,BEGIN      ; This is the CONFIG
                LD        (400DH),HL    ; initialization routine.
      CMD:      DEFB      0C9H
      ADDRES:   DEFW      0
      FCB:      DEFS      50
      DSKBUF:   DEFS      256
      NEWLN:    DEFS      64            ; Result line buffer
      LINEBF:   DEFS      80            ; Syn line buffer
      LAST:     DEFM      'HJM'
                END       START
```

Modifying the Model 3 Real-Time-Clock Interrupts

by Andrew Gransden, c/o 68 St Annes Grove, FAREHAM, Hampshire, England, UK TS15 9TB

This article is primarily aimed at those TRS-80 Model III owners, living outside North America, with machines adapted to work with 50 Hertz mains power. This is not to say that other readers will not be interested in the experiences described below. (With some modification, this approach could be used to correct the clock on a Model 4 running in the Model 3 mode under 5.1.4 at the 4MHz speed - ed.)

This all started by replacing my unreliable Mod I-type machine (Video Genie {UK}/PMC-80 {USA}) with a Mod 3 from Tandy. I stayed with the TRS-80 line only because I am hooked on LDOS, and on the fact that I could use all my LDOS compatible software (with only minor patching) and disks on my new machine. I was extremely pleased with my Mod 3 operating under LDOS apart from the annoying fact that the Real Time Clock (RTC) lost 10 seconds every minute. As I had always used the RTC in my programs, and with system functions like JOBLOG, I set forth to find a means of correcting this problem.

The internal clock must run at the proper frequency to ensure that the display is not affected by jitter or flickering. In the Mod 3, the internal clock frequency is divided to yield a RTC interrupt frequency half that of the mains. In a 60Hz machine the RTC interrupt occurs at 30Hz, or every 33.33 milliseconds, while in the 50Hz machine it occurs at about 25Hz (25.381Hz to be precise) or approximately every 40 milliseconds. Unfortunately, TANDY did not compensate for this difference by replacing the System ROMs. TRSDOS 1.3 can be patched, but under LDOS the clock management is left to the ROM. This means that the RTC goes uncorrected when using LDOS.

Having established the cause of the problem, I now had to find a way of applying my own correction. With reference to the Technical Information section of the LDOS Manual, the

Model III Technical Reference Manual (Cat No. 26-2109) and the excellent reference work 'Model III ROM Commented' (a fully commented disassembly of the System ROMs) and armed with EDAS 4.1, DSMBLRII and the LDOS DEBUGger, I set about my task.

One solution would have been to design, build, and install an additional crystal-driven clock operating at 30Hz. This would have been easy enough, but with the result that the warranty on my Mod 3 would have been voided. Another solution would have been to order, at some expense, an external battery-powered clock. The third, and the solution I chose was to correct the clock error using software. The following is a description of the Mod 3/LDOS RTC interrupt control chain, and how I solved the problem.

When the RTC pulse occurs, it interrupts the CPU and sets Bit 2 of the Interrupt Status Port (X'E0') to zero. An image of this Status Port is kept by LDOS in INTIM$ (X'4473'). The Jump vectors relevant to each Interrupt Status Bit are held in INTVC$ (X'4475' - X'4484'). In LDOS, the vector relating to Bit 2 of INTIM$ points to X'44A5' in SYS0, which in turn jumps to X'3529' in ROM where the RTC routines can be found. This routine decrements the system clock 'heartbeat' (X'4216') from 30 (X'1E') down to zero. When 'heartbeat' reaches zero, the time is incremented and, if selected, displayed in the corner of the screen. What I needed to do was to re-write this routine to operate properly at the actual 25Hz interrupt rate instead of the expected 30Hz. To totally re-write the ROM routine would have used a lot of high memory, and would have duplicated a lot of code. The solution was to write a relocatable routine which would apply a correction to the clock count, and then hand control back to the ROM RTC routine. Results from several experiments proved that I needed to apply a double correction using 2 counters. The final affect is to stretch every 20th second by approximately a third. The accuracy of the corrected RTC is within 3 seconds a day.

The program first loads into low memory; locates the current RTC interrupt vector and saves it; modifies the internal references; installs the routine in high memory below the current HIGH$; and then reduces HIGH$ to protect the routine. The correction routine uses only 87 bytes of high memory and obeys the 'front-end' protocol as defined by LSI. The program carries out a series of checks to ensure that you are using a Mod 3 with LDOS (Version 5.1.x) and you have not previously applied the correction. The program will abort with a suitable error message if any of these tests fail.

Once installed the routine can be SYSGENed to load every time your Model III is booted. This is because SYSGEN saves all system vectors including those contained in INTVC$ as well as the high memory area above HIGH$. One word of warning: as should be the case with all programs used with LDOS, HIGH$ should be respected, otherwise your system will crash FASTER than ever before (within 25 ms of clobbering the correction routine).

Below is the BINHEX dump for those of you without an Editor/Assembler.

```
0506 5449 4D45 4249 0102 0052 2196 52CD 6744 3A25 01FE 49C2 1853 2A13 403E A5BD C21D
533E 44BC C21D 532A 7944 3E4F BCDA 2253 2A11 44DD 21C9 53DD 7501 DD74 02DD 21CD 53DD
7501 DD74 02DD 21D6 53DD 7501 DD74 023E 1577 2BDD 21DE 53DD 7501 DD74 02DD 21E2 53DD
7501 DD74 02DD 21EB 53DD 7501 DD74 023E 2477 2A79 4422 0A54 2A11 4422 B953 0157 00AF
ED42 2211 4423 F322 7944 EB21 B753 EDB0 FB21 F752 CD8A 42C3 2D40 5449 4D45 3530 202D
204C 444F 5320 5265 616C 2054 696D 6520 436C 6F63 6B20 3530 487A 2043 6F72 7265 6374
696F 6E20 5574 696C 6974 7920 2D20 5665 722E 342E 310A 436F 7079 7269 6768 7420 2843
2920 3139 3833 2020 4120 5720 4772 616E 7364 656E 0D54 494D 4535 3020 696E 01BB 0053
7374 616C 6C65 6420 616E 6420 6F70 6572 6174 696F 6E61 6C0D 2131 5318 0821 5053 1803
218A 53CD 8A42 21A7 53CD 8A42 C330 4046 6F72 2054 5253 2D38 3020 4D6F 6465 6C20 4949
4920 7573 6520 4F4E 4C59 210D 436F 7272 6563 7469 6F6E 2077 7269 7474 656E 2074 6F20
776F 726B 2075 6E64 6572 204C 444F 5320 5665 7273 696F 6E20 352E 312E 7820 4F4E 4C59
210D 436F 7272 6563 7469 6F6E 2061 6C72 6561 6479 2069 6E73 7461 6C6C 6564 0D54 494D
4535 3020 4162 6F72 7465 6421 0D18 0901 53BB 5306 5449 4D45 3530 3A16 42FE 1620 353A
0C54 3D32 0C54 FE00 200A 3E15 320C 543E 1E32 1642 3A0D 543D 320D 54FE 0020 153E 2432
0D54 3E01 5F3A 1642 93FE 0630 023E 0632 1642 3A16 42EE 0620 043C 3216 42C3 0000 0202
0052
```

```
00100 ;*       TIME50/ASM - Version 4.1 - 28 Oct 83
00110 ;*       TITLE:  TIME50 50Hz Real Time Clock Correction Routine
```

```
00120 ;*      AUTHOR: Andrew W. Gransden
00130 ;*              c/o 68 St Annes Grove
00140 ;*              FAREHAM, Hampshire
00150 ;*              PO14 1JW   England  UK
00170 ;*      Copyright (c) 1983 A W Gransden
00190 ;*      TIME50 routine corrects for errors in the
00200 ;*      the TRS-80 Model III Real Time Clock
00210 ;*      interrupt handling routine when operating
00220 ;*      on a 50Hz Version Machines under the
00230 ;*      LDOS Disk Operating System Version 5.1.x.
00250 ;*      Variable and Label Declarations
00260 LF      EQU    0AH      ;linefeed
00270 CR      EQU    0DH      ;carriage return
00280 SEC1    EQU    21       ;coarse correction count
00290 COR1    EQU    8        ;coarse correction value
00300 SEC2    EQU    36       ;fine correction count
00310 COR2    EQU    1        ;fine correction value
00320 ROMCHK  EQU    0125H    ;ROM check for Model III
00330 INTVEC  EQU    4012H    ;interrupt vector
00340 @EXIT   EQU    402DH    ;LDOS return entry
00350 @ABORT  EQU    4030H    ;abnormal program exit to LDOS
00360 HBEAT$  EQU    4216H    ;clock heartbeat counter
00370 HIGH$   EQU    4411H    ;highest useable memory
00380 @DSPLY  EQU    4467H    ;display message
00390 @LOGOT  EQU    428AH    ;display & log message
00400 RTCVEC  EQU    4479H    ;jump vector to RTC routine
00420 ;*      start of TIME50 installing routine                   *
00440         ORG    5200H
00450 ENTRY   LD     HL,MSG1               ;point to message 1
00460         CALL   @DSPLY                ;and display it
00470         LD     A,(ROMCHK)            ;test for Model III
00480         CP     49H
00490         JP     NZ,ERROR              ;go if not
00500         LD     HL,(INTVEC+1)         ;load interrupt vector
00510         LD     A,0A5H                ;load A with known jump
00520         CP     L                     ;and compare with HL
00530         JP     NZ,ERROR1             ;exiting if incorrect
00540         LD     A,44H                 ;to error handing abort
00550         CP     H
00560         JP     NZ,ERROR1
00570         LD     HL,(RTCVEC)           ;load RTC vector
00580         LD     A,4FH                 ;load max DOS area addr
00590         CP     H
00600         JP     C,ERROR2              ;go if greater
00610         LD     HL,(HIGH$)            ;get current high mem bc
00620         LD     IX,P1                 ;set pointer to allow
00630         LD     (IX+1),L              ;correct addressing of
00640         LD     (IX+2),H              ;storage in high memory
00650         LD     IX,P2
00660         LD     (IX+1),L
00670         LD     (IX+2),H
00680         LD     IX,P3
00690         LD     (IX+1),L
00700         LD     (IX+2),H
00710         LD     A,SEC1                ;set count to ? seconds
00720         LD     (HL),A
00730         DEC    HL
00740         LD     IX,P4                 ;set pointer as above
00750         LD     (IX+1),L
00760         LD     (IX+2),H
00770         LD     IX,P5
00780         LD     (IX+1),L
00790         LD     (IX+2),H
```

```
        LD     IX,P6
        LD     (IX+1),L
        LD     (IX+2),H
        LD     A,SEC2
        LD     (HL),A
        LD     HL,(RTCVEC)           ;get present mt vector
        LD     (EXIT+1),HL           ;and save
        LD     HL,(HIGH$)            ;reduce HIGH$ by
        LD     (NXTMEM),HL           ;store old high memory
        LD     BC,LAST-START         ;length of routine
        XOR    A                     ;clear carry flag
        SBC    HL,BC                 ;calculate new HIGH$
        LD     (HIGH$),HL            ;protect routine
        INC    HL                    ;point to new start
        DI                           ;disable interrupts
        LD     (RTCVEC),HL           ;break into Int chain
        EX     DE,HL                 ;transfer new START to DE
        LD     HL,START              ;load address of routine
        LDIR                         ;move routine to high ram
        EI                           ;enable interrupts
        LD     HL,MSG2               ;point to message
        CALL   @LOGOT                ;display & log
        JP     @EXIT                 ;return to LDOS
;*      Messages                                             *
MSG1    DB     'TIME50 - LDOS Real Time Clock 50Hz '
        DB     'Correction Utility - Ver.4.1',LF
        DB     'Copyright (C) 1983 A W Gransden',CR
MSG2    DB     'TIME50 installed and operational',CR
;*      error handling                                       *
ERROR   LD     HL,ERMSG              ;point to message
        JR     EREXIT                ;jump to error exit
ERROR1  LD     HL,ERMSG1             ;point to error message
        JR     EREXIT                ;jump to error exit
ERROR2  LD     HL,ERMSG2             ;point to error message
EREXIT  CALL   @LOGOT                ;display message
        LD     HL,ERMSG3             ;load abort message
        CALL   @LOGOT                ;display & log
        JP     @ABORT                ;jump to abort routine
;*      error messages                                       *
ERMSG   DB     'For TRS-80 Model III use ONLY!'
        DB     CR
ERMSG1  DB     'Correction written to work under '
        DB     'LDOS Version 5.1.x ONLY!'
        DB     CR
ERMSG2  DB     'Correction already installed'
        DB     CR
ERMSG3  DB     'TIME50 Aborted!',CR
;*      actual TIME50 routine to be placed in high ram       *
START   EQU    $
        JR     J1                    ;skip protocol block
NXTMEM  DS     2                     ;high mem addr of next bk
        DB     06H                   ;6 bytes of protocol blk
        DB     'TIME50'              ;routine title
J1      EQU    $
        LD     A,(HBEAT$)            ;get heartbeat count
        CP     1EH-COR1              ;just reset?
        JR     NZ,TEST2              ;go if not
P1      LD     A,(COUNT1$)           ;get seconds count
        DEC    A                     ;decrease by one
P2      LD     (COUNT1$),A           ;save
        CP     00H                   ;... seconds gone
        JR     NZ,P4                 ;go if not
        LD     A,SEC1                ;reset seconds count
```

```
01510 P3      LD     (COUNT1$),A          ;save reset count
01520         LD     A,1EH                ;increase heartbeat
0i530         LD     (HBEAT$),A           ;save reduced heartbeat
01540 P4      LD     A,(COUNT2$)          ;carry out fine
01550         DEC    A                    ;correction
01560 P5      LD     (COUNT2$),A          ;and save
01570         CP     00H
01580         JR     NZ,TEST2             ;when COUNT2$ reaches 0
01590         LD     A,SEC2               ;reset correction count
01600 P6      LD     (COUNT2$),A
01610         LD     A,COR2
01620         LD     E,A
01630         LD     A,(HBEAT$)           ;get heart beat count
01640         SUB    E                    ;remove correction
01650         CP     06H                  ;finished?
01660         JR     NC,J2                ;go if less
01670         LD     A,06H
01680 J2      LD     (HBEAT$),A
01690 TEST2   LD     A, (HBEAT$)          ;get heartbeat count
01700         XOR    06                   ;at new bottom?
01710         JR     NZ,EXIT              ;go if not
01720         INC    A                    ;heartbeat=1
01730         LD     (HBEAT$),A           ;save reduced heartbeat
01740 EXIT    EQU    $
01750         JP     0000H                ;jump to RTC routine
01760 COUNT1$ DS     1                    ;reserve 2 bytes
01770 COUNT2$ DS     1                    ;for correction counts
01780 LAST    EQU    $
01790         END    ENTRY
```

Profile ONE Plus ??

by E. R. Sturiale, SASSCO Microcomputer Services
133 Falmouth St., Rochester, NY 14615 (716) 865-1622

Has Radio Shack forgotten about the Mod 1 user? If their marketing of Profile III+ is any indication, they have. Fortunately, when they made the wise decision to use LDOS as their hard disk operating system, the possibility arose to develop patches for Mod 1. Since we run a small software consulting firm which uses both Mod 1's and 3's, it became necessary to have compatibility between machines for Profile data bases.

As you can probably tell from the number of patches, the conversion of machine language programs is not simple. Also, there are just enough hardware differences between machines to generate Excedrin headaches 256 through 1023. The calls to the operating system were relatively easy to change after the required 896 pages of disassembling and cross referencing to decode the Profile modules.

Applying the patches

1) The minimum requirements to install these patches and run Profile I+ are:
   Double Density (any LDOS-supported form)
   Two Disk Drives
   PROFILE III+ HD (RS number 26-1593)

2) BACKUP your RS Profile III+ distribution diskette onto a working patch DATA disk. This can be accomplished after formatting by using the LDOS Backup utility with (X,VIS) or any other copy by file option. By all means leave the write protect tab on the distribution diskette.

3) Create a "CLEAN" LDOS system diskette with at least 50 K of free space.

4) Type in all the FIX and JCL files using the BUILD command (or some ASCII text editor). Store them on the "CLEAN" LDOS system diskette and double check your typing.

5) Place the Backup (working) PROFILE diskette in Drive 1 and the system disk with the fix files in Drive 0.

6) NOTE : If your distribution diskette is labeled Version 01.00.01 or if you have already applied the patches supplied by Radio Shack, then skip to Step 7.

   Now type: DO RSPATCH

   If the messages indicate all is O.K., then continue else check your RSPATCH/JCL file and then return to Step 2

7) This JCL will apply all the patches necessary for Profile 1 + to operate. The procedure may take as long as five minutes. Please watch the screen for errors in patching. If one does occur, then check the FIX files and go back to Step 2.

   To start the patch procedure, type: DO PROFIX

8) If you are going to use the system for Hard Disk operation, then copy all your Profile /CMD programs from the working disk on Drive 1 to your HD and begin hacking. If you want to use the system on floppy, then continue. This set of patches will add prompts for diskette swaps.

   Now type: DO PROMPT

9) At this point, create two more LDOS system diskettes and label the first "CREATION DISKETTE" and the other "RUNTIME DISKETTE"

   Copy the following patched files to the CREATION diskette:
   EFC1/CMD EFC2/CMD EFC3/CMD EFC4/CMD EFC5/CMD EFC6/CMD EFCE/CMD EFCM/CMD CM/CMD

   Copy the following patched files to the RUNTIME diskette:
   EFC7/CMD EFC8/CMD EFC9/CMD EFCA/CMD EFCB/CMD EFCC/CMD EFCD/CMD EFCF/CMD RM/CMD

   If all goes well, you should be ready to dive into the manual and get started.

There are several differences in PROFILE I+ that I should mention.

1) The Profile I+ programs will try to write the working modules such as screen formats on Drive 0. When working with the floppy version, it is usually more convenient to have these files somewhere else. The best way to do this is to use the LDOS SYSTEM (DRIVE=0,WP=ON). This forces the files to be written on the next available drive and not your CREATION diskette.

2) The cursor character on the DEFINE SCREENS option is different than standard. The only problem that may occur is if you try to use the special character from the <shift> @ display which is the same as the cursor. If the cursor passes over this character the special character will be erased. Since there are lots of other special characters that look like the cursor, you should not have any problems selecting another one.

3) Part of the patching procedure disables the Model III scroll protect option since the Model I does not have that feature. The easy way to do that was to change the memory loads required to a place where they will not do any damage. I chose location 3001H which is in non-existent memory in the Model I. If you are using a memory side-car that resides in that area then either un-plug it, or change all the patches in the FIX files that are "01 30" to a location that is not being used. By the way, I have not noticed any difference in operation or screen presentation by eliminating the scroll protect feature. Other than some characters being

different, due to the different character sets, all screen presentations seem to be O.K.

4) Using KI/DVR with the (TYPE) option is recommended. Use of Profile I+ with the other drivers or filters (except the HD drivers, PDUBL and RDUBL) has not been evaluated. If you wish to try some others, experimentation may be required. When using KI/DVR with Profile I+, any prompts that call for the <clear> key should be replaced with <shift><clear>.

5) None of the other modules that are offered by Small Computer Company have been patched or tested with Profile I+. We have plans to purchase them so if patches are necessary, you may see them in future LSI Journals.

6) After Profile I+ was working for awhile, we noticed that the programs gave excessive PRINTER NOT READY messages when everything was in fact O.K. Patches were added to lengthen the delay time the programs wait for the printer to respond to accommodate such functions as double striking or fast CPU's.

The patches are available in XA0 of the LDOS SIG on CompuServe, and are presented here starting on page 47.

***** PARITY = ODD *****

by Tim Daneliuk, T&R Communications Assoc., 4927 N. Rockwell St., Chicago IL 60625

Well, Winter is upon us. If that isn't bad enough, my friendly LSI Journal editor decided that he needed TWO columns absolutely ASAP. He said "I need two PARITY=RIDICULOUS columns by the deadline. Can you do it?" Well, I barely make it to work on time. So friends, I had to get moving. Translated, this means that what you are about to read is probably not up to the high journalistic standards you've come to expect from me. I must be held blameless, as I simply cannot rush two columns out in 12 weeks and be expected to maintain the excellence and humility you've all come to know and love...

CP/M and the TRS-80

Let's face it. CP/M is not a great operating system. It's not really even a GOOD operating system. There are too many systems out there which use CP/M as the DOS (?) to ignore "Pa Kildall's" favorite product. Even the folks at Radio Shack have announced their intention to bring out CP/M+ (aka CP/M 3.x) for the Mods 4, 4P and 12. So, as a public service, let's discuss a few CP/M-80 related products.

In all fairness, I should point out that CP/M was FIRST! For its day, it was a fine product which served the 8-bit micro industry well. Much of the popularity of early microcomputers was partly due to the existence of CP/M. Since it was first, CP/M is among the best outside vendor and user-supported operating system on the market. There are products which ONLY run under CP/N, so it's a good idea to be familiar with it.

First, let's look at Montezuma Micro's (hereafter known as MM) CP/M for the Model 4. Tandy announced CP/M availability this spring, and to date has not yet released it. The MM CP/M is a full blown CP/M 2.2 for the Mod 4. It incorporates all the usual CP/M commands (all 4 of 'em) and utilities, with extras. The folks at MM have done a real nice job on this implementation. The BIOS (Basic Input/Output System) is written to emulate an ADM-3A. The keyboard driver allows you to input special characters (curly braces {}, brackets [], and backslash \) which are not normally available on the Model 4 keyboard. MM has gone as far as cleaning up those hideous CP/M error messages. Now after an error, you get the choice of retrying the operation which caused the error, letting CP/M handle the error, or rebooting the system. Instead of the usual "BDOS ERROR" message, the MM CP/M tells you things like "Record Not Found" and the like.

The highlight of this CP/M implementation is that it is chock full of useful utilities. The most notable is INTERCHG.COM, a utility which reads "alien" disk formats. The only "standard" CP/M format is eight inch, single-sided, single-density. As a result, there is no "compatibility" of the CP/M system whenever you're using other types of media. This isn't so bad with eight inch floppy disks because if you can write anything else, you can almost always create single-density single-sided disks also.

With five inch CP/M, each manufacturer specifies the media format as they choose. This has created a wonderful mess of incompatible five inch CP/M disks. INTERCHG reads over 20 popular five inch CPIM media formats, including Osborne, Xerox, Lobo, Zenith, and NEC disks. I tested INTERCHG on Lobo and NEC PC-8001 five inch disks. I was able to read both with no difficulty whatsoever.

MM has also included the popular public-domain MODEM.CON program. This gives you communications ability under CP/M. MEMLINK.COM is a "RAMdrive" program which can use the extra 64K memory bank as a logical drive, like TRSDOS 6.x. The only thing missing in this CP/M implementation is the ability to use the Shack's hard disks under CP/M. On the other hand, why would anyone stop using LDOS and start using CP/M on a hard disk?

In the several months I've used the MM CP/M, it has run flawlessly with one exception. When you do a lot of disk I/O among several different drives, you get Record Not Found errors. I've detailed this bug to the manufacturer. By the way, if you're worried about support, fear not! Montezuma Micro is run by the same people who run Aerocomp. Aerocomp is one of the most reputable and helpful mail-order houses in the business. The Model 4 CP/M package comes with 36 pages of documentation about the implementation, as well as Dave Cortesi's EXCELLENT book "INSIDE CP/M". The latter is a 500+ page discussion of CP/M. MM CP/N costs $199 and is available from:

Montezuma Micro, P.O. Box 32027, Dallas, TX 75232 (214) 339-5104

Thanks to a local CP/M "guru", I have been introduced to a fabulous new product called MPC. If you fiddle around in assembler, you've probably had the urge at some point to write you're own operating system. No, you're not crazy! (Well, maybe a little crazy...) Of course, such a project is a large undertaking, and would require more time than the average individual has available. The next best thing would be to look at the source code for someone else's DOS. Unfortunately, most DOS authors frown on distributing the source for their system, and quite understandably!

You may have noticed that MPC is CPM reversed, and that's exactly what this product does. It "reverses" (disassembles) the code which makes up the CP/M kernel and produces a FULLY COMMENTED source file. If you're interested in how the BDOS (Basic Disk Operating System) or CCP (Console Command Processor) work, you can read and study this file. The comments alone are worth their weight in gold and will give you great insight as to how CP/M actually works. MPC costs only $35 and is available from:

CC Software, 2564 Walnut Blvd. #106, Walnut Creek, CA 94598, (415) 939-8153

THINGS FOR THE MAX-80

The folks at Powersoft are at it again! Apparently they really like the MAX-80 'cause they keep bringing out new products for it. First, a "MAXed" version of SU+, and now SETMAKER/SETWRITER. If you own a MAX, you already know that the video system on the MAX is quite versatile. The character set is programmable - i.e. you can make each character look as you wish. Unfortunately, this is a messy proposition involving time, effort, and assembly language.

SETMAKER allows you to create custom character fonts and graphics characters on the MAX-80. These can be saved as on disk or they can be directly loaded into LDOS itself.

You can boot the system with your own fonts and graphics in place. Powersoft has also included several examples of customized character fonts and graphics.

SETWRITER is a companion product to SETMAKER, and allows you to print your custom fonts and graphics on an Epson MX-80 or FX-80, just as you see them on the screen! If you're using an MX-80 you must have GRAFTRAX. I was not able to test SETWRITER since I don't have either of these printers. Judging from SETMAKER (which ran flawlessly), I don't hesitate to recommend these programs. They're in machine language, and are about as "bullet proof" as can be. There are plenty of on-line menus-- the documentation is almost unnecessary. These are $29.95 each, or $50 for both. They're from:

PowerSOFT, 11500 Stemmons Freeway Suite 125, Dallas, TX 75229 (214) 484-2976

MODEL 4 TOPICS

Speaking of Powersoft, they've also just released their LDOS utilities for TRSDOS 6.x on the Model 4. It's the "Toolbelt for TRSDOS 6" and costs $49.95. Considering the (more than 15) useful programs you get, this has got to be the best bargain in town. I use versions of these utilities under LDOS 5.1 and TRSDOS 6, and find them to be excellent products. Contact Powersoft for more details.

A new word processor from Anitek called "LeScript" is available. One version runs on Mod 3, 4, or MAX (the package also includes a Mod 1 version). It supports 80 col on the latter two, even when running the Mod 4 in Mod 3 mode! LeScript also uses the extended memory available in the Mod 4 and MAX as text buffer. When you fire LeScript up on a MAX, for example, you're greeted with the pleasant sight of around 80K of text space.

LeScript also supports virtually every printer known! If you have a printer from RS, Epson, NEC, C. Itoh, ... you'll find its features implemented in LeScript. For example, I was able to use italics, underline, super- and sub-scripts on my MX-100. LeScript even lets me print in proportional mode by using the Epson's bit-image graphics.

One especially delightful aspect of LeScript's operation is that it runs just great with LDOS 5.1 / TRSDOS 6. Although the program doesn't ordinarily use the system's DCBs (ahem!) it DOES know enough to not interfere with operating system features. For example, you can leave type-ahead on when you enter LeScript, and when you're done you won't be greeted with lines of garbage.

For $129 you'll be hard-pressed to find a better overall word processing product. Though LeScript isn't virtual (the maximum text you can edit at any one time is limited by memory), it should accommodate the vast majority of word processing chores you can dream up. If I sound enthusiastic, I am! You will be too when you see this product. For more information, contact:

ANITEK Software Products, P.O. Box 361136, Melbourne, FL 32936 (305) 259-9397

MACHINE WARS

It never fails, almost invariably someone asks the question, "What should I buy, a Model 4 or a MAX?" I've used both quite a bit and I'm ready to give my informed opinion - "It depends!" From a pure performance point of view, the MAX wins hands down. In some cases, the MAX runs rings around much more expensive machines like the IBM-PC or the TRS-80 Mod 12. I also prefer the versatility of the MAX. It gives me two serial ports, runs eight inch floppies, and can boot from any type of disk drive, including hard disks.

There's another side to this story. It's clear that new TRS-80 software will be for LDOS/TRSDOS 6. LDOS 6.x is not now available for the MAX-80, and may never be, due to hardware conflicts. If you use your machine in a commercial application and need to be compatible with the rest of the world, it seems that the MAX is not a viable choice unless you intend to use CP/M.

What about the Mod 4? Frankly, the Mod 4 never impressed me much. It has the same sleazy video that Tandy is notorious for, has no 8" drive capability, and only one RS-232 port. Worse yet, early Mod 4s apparently had flakey disk controllers and used wait states when accessing memory. The latter made this "4 MHz" machine run as much as 25% slower. (I am told that these problems have been resolved, and that current production Mod 4s work just fine.) But . . . just wait 'till you see the Mod 4P (P=Portable)! It has great video, no wait states, and is built like no TRS-80 I've ever seen. I spent a day with a 4P, and as jaded as I am, I was impressed! This is a wonderful machine, and shows that Tandy IS paying attention to the market. I still miss 8" floppies because an 8" disk drive is an excellent compromise between price and storage capability. Other than that, the 4P is a "dream" machine.

Another consideration is support. Though LOBO has one of the very best warranties, as a mail-order operation they're not in the position to give instant help. Radio Shack, on the other hand, can help you locally and is in a better position to answer questions.

All things considered, the bottom line is this: If you're a software "tinkerer" who is reasonably knowledgeable and a performance hound, get a MAX. You'll love it! If you're fairly non-technical, and need a lot of "hand-holding" and support, buy from Radio Shack. What do I use? A MAX-80! Though I enjoy using the Mod 4, I find the MAX a consistently overall better performer. Still, the Mod 4P is really tempting! Mebbe if I save my lunch money for a few years...

THE WRAP UP

That's it for now. Unfortunately, I've had to delay the review of the Model I to III upgrade I mentioned in last time. Hopefully I'll get another crack at it later in 1984.

The "C" Language (Part V)

Earl 'C' Terwilliger Jr., 647 N. Hawkins Ave., Akron, Ohio 44313

INITIALIZATION, BLOCKS, POINTERS, ARRAYS

As you can tell from the C commented title for Part V, the subjects for discussion are blocks, pointers, how variables can be initialized and an introduction to arrays. Shall we start with more on blocks? (Were you expecting a choice?)

Several computer languages are block-structured in the sense that they allow functions to be defined within other functions. C does not allow this. In C, functions are always "external" since they are not inside of other functions. I am alluding to the fact that functions are blocks of C code. Remember from previous parts that a block is enclosed via {}. These braces {} enclose functions and other blocks. After the { comes variable definitions, if any. Variables in C, are thus defined in a block-structured manner.

Variables can be declared following the { that begins any compound statement. Also after the { that begins a function, variables can be declared (defined). If more variables need to be declared, later in the function, they can be, by declaring them after the left brace which begins a block. These variables can even have the same name as other variables. Their declarations "supersede" the identically named variables in outer blocks. They exist only within the block in which they are declared. Don't forget or confuse what you have learned previously about variable storage class and what you are learning now. The above comments on variables declared within blocks hold true for external variables too. Now can we look at how variables can be initialized? (No freedom of choice, is there?)

If you would like to assign an initial value to a variable when it is defined, C will allow it. As an interesting point, C does initialize certain variable classes for you. If you do not specifically assign an initial value to an external or static variable, C will initialize them to zero for you. However, automatic and register variables are not initialized automatically for you by C. So, don't count on them containing anything worthwhile unless you specifically initialize or assign a value to them. An equals sign and a constant expression are used to initialize simple variables. (Arrays and structures are initialized differently, as we shall C later.) Here are some examples of simple initialization:

```
                  int a = 5;
                  int b = c = d = e = 0;
                  char g = 'x', h, i = 'y';
                  char f = 'f';
                  int d = 45 * 67;
```

As you can imply, this initialization saves "extra", sometimes unnecessary, assignment statements which assign a value to a variable. K&R call this shorthand for assignment statements. Remember what was just learned about blocks and how variables can be declared within them? Well, variables declared within these blocks (or functions) can also be initialized. This initialization takes place each time the function or block is "entered". External and static variables are initialized only once. (Are you wondering why this is? External and static variables are of different storage class and scope than automatic and register variables. Think about how and when these variables come into existence and when they go out of existence (if they do)?) Also, for automatic and register variables, the initialization can be done via any valid expression. This initializer is not limited to a constant expression.

Before I discuss how arrays can be initialized, shouldn't I discuss what they are and how they are declared (defined) ? For example:

```
                  int number[10];
```

This declares an array of size 10. In essence, this is a "block" of 10 integers together. Likewise:

```
                  char name[12];
```

declares a block (an array) of 12 characters. Each member of the array is called an element. Each element is numbered or indexed. In C the index starts at zero. In the number array above, the elements can be referred to individually via number[0], number[1], ..., thru number[9]. C also supports multi-dimensional arrays. For example:

```
                  int a[10][20];
```

This declares a two dimensional (rectangular) array. Elements of a multi-dimension array are stored by rows. Viewing storage as linear, elements of the array are seen in storage order if the right most index varies fastest. Now, how can arrays be initialized?

Arrays are initialized differently than other variables. Only external and static arrays can be initialized, automatic arrays can not be initialized. External and static arrays are initialized as shown in this example:

```
                  static int numbers[10] = { 0,1,2,3,4,5,6,7,8,9 };
```

Remember, in the absence of explicit initialization, all elements of external and static arrays are initialized automatically to zero.

In initializing external and static arrays, fewer initializers can be used than there are elements. In this case, the remaining elements will be zero. C also disallows more initializers than elements. Wouldn't it be nice to be able to repeat an initializer or just to initialize specific elements and ignore others? Well, sorry, C does not provide a means to do that.

Here is an example of a character array and its initialization:

```
                                 /*  ....5...10...15...20...25 */
                  static char me[] = "Earl C. Terwilliger Jr.";
```

Quick! How many elements does the array me have? (Use the comments ruler line to help you count.) Did you guess correctly with 24? Each character between the quotes is an element plus the \0 which is added by the C compiler to terminate the string. Did you notice that the size of the array, i.e., the number within the [] was omitted? If you do not include it, C will compute the size of the array for you based on the number of initializers. Another way to initialize a character array is as follows:

```
                  char name[] = { 'E', 'A', 'R' , 'L', '\0' };
```

Notice that it is so much easier to use:

```
                  char name[] = "EARL";
```

Are you thinking that the initialization of a character array is like a "string copy"? If so, be careful in your evaluation of the following statements:

```
                  static char msg[5];
                  msg = "TEST";
```

This is not a string copy! C does not provide any operator for string copying or dealing with an entire string of characters as a single unit. Also, msg is the name of an array, it is a constant. It is not an lvalue and the above expression using it as such is ILLEGAL! How then can elements of an array be assigned values? The answer is by individually assigning values to each element. To "blank out" a character array, examine the C code which follows:

```
                  char message[20];
                  ...
                  for (i=0, i<20, ++i) {
                       message[i] = ' ';
                       }
```

Also, note that the message array does not necessarily have to be external or static. It could be an automatic array!

Next, onward to pointers! A pointer is a C variable which contains the address of another variable. I can hear you thinking! You are no doubt asking, how does the pointer get the address? The unary operator & mentioned in an earlier part gives the address of its object. The & operator applies only to array elements and variables. Consider the following:

```
                  char a;
                  char *ptr;
                  ...
                  a = 25;
                  ptr = &a;
```

In the expression: ptr = &a, ptr is assigned the address of a. By the way, there is no such thing as just a pointer. In C, pointers are always pointers to a particular data type. As shown above ptr is a pointer to type character. The * operator denotes indirection, it treats its operand as an address. It accesses this address to obtain the contents stored there. For example:

```
                  char *ptr, a, b;
                  b = 'x';
                  ptr = &b;
                  a = *ptr;
```

In the above examples, b is assigned the value 'x'. ptr is assigned the address of b. a is assigned the value of the character pointed to by ptr, which is 'x'. *ptr is a C mnemonic declared in this example to be a character. The combination of the * and ptr denote a character just like the above variable b does. When a pointer is declared, the type of data it points to is stated. The pointer is limited to point to data of that type. Also, pointers and pointer references are lvalues and can appear on the left side of assignment statements. Above, the pointer ptr is seen appearing on the left of an assignment statement. Below, *ptr is shown on the left of an assignment:

```
                   char *ptr, a, b;
                   b = 'x';
                   ptr = &a;
                   *ptr = b;
```

After the above statements are executed, a will contain the same value as b! *ptr is a pointer reference. In the case above it actually references a. ptr contains the address of a and *ptr references the character stored at the address in ptr.

Having the address of a variable is very useful. Remember from a previous part that C passes copies of variables as arguments to a called function. This is "call by value". The called function can not alter a variable in the calling function. (Actually, it could if the variable used in both functions was an 'external" variable.) Now that you have learned about the & operand, you can use it to pass, as parameters to a function, addresses of (pointers to) variables. The called function can declare the arguments passed as pointers and alter the referenced data!

Looking back over the discussion on arrays, do you remember the problem of assigning values to an array? Consider this, now that you are familiar with arrays and pointers:

```
                   char *myname;
                   myname = "Earl C. Terwilliger Jr.";
```

This also is not a string copy! But it is a valid expression. myname is a pointer and it is assigned the address of the string! Comparing these two C statements with the ones shown to illustrate arrays, you should be wondering about the relationship between an array and a pointer. Actually an array name is a pointer expression. However, keep in mind that a pointer is a variable but an array name is a constant. If an array name is passed as an argument to a function, what is actually passed is the location (address) of the beginning of the array. (Using the & operator on just an array name is invalid. C does however, allow the & operator to take the address of an array element, for example &myname[4]. The & operator applies only to variables and array elements!)

A called function, when passed an array name as an argument, can declare the argument as a pointer and reference thru the elements of the array. Would you like an example?

```
         main() {
              static char myname[] = "Earl C. Terwilliger Jr.";
              char a;
              a = 'l';
              printf("%d\n",scount(myname,a));
         }
         scount (ptr,ch)
              char *ptr, ch;
         {
              int c = 0;
              while (*ptr != '\0') {
                   if (*ptr++ == ch) ++c;
              }
              return (c);
         }
```

The function scount will return the number of occurrences of a given character in a given character string (array). The two parameters passed to it are the address of the string to search and the character to search for. If you follow the logic, pay particular interest to the *ptr++ expression. The value printed after the above code is executed should be 3! (What? You don't believe me? Type in the code and try it out on your favorite C compiler.)

Next time, you will see more on pointers and arrays. Structures will be introduced and I will point out some of the most common errors found in C programs.

Items of General Interest

Here are corrections and additional information regarding subjects raised last time:

Page 12:

The patch to restore "random" allocation should have been listed as for Model 3 and MAX-80, LDOS 5.1.x only. The correct patch for Model 1, LDOS 5.1.x is:

```
PATCH SYS8/SYS.SYSTEM:0 (D00,FE=D5 CD C1 44 D1 6C)
```

Page 14:

The new name for the TRSOOS 6.x communications software package is LS-Host/Term, Catalog number L-35-281, $199 plus $3 shipping and handling.

Page 55:

It has been reported that the following patch will correct a "0 left" error with Model 1 SuperSCRIPSIT, Version 01.02.00: PATCH SCRIPSIT/CND (X'7E22'=FC)

Page 57:

Here is the equivalent of the ROM/CTL patch, but for the Model 1. Comments are the same as the Model 3 patch:

```
. ROM/FIX
. Patches to the Model 1 SS 1.2 DW2/CTL driver for system DCB usage
D00,91=3D BF
D02,0B=3E 30 00
D03,2D=CD 35 BF
D03,45=CD 35 BF
X'BF35'=D5 F5 CD 3B 00 F1 D1 C9
. End of patch
```

Page 64:

In the FDC driver patch, the first line should have ended ED A2, not ED A4. The SYS2 patch for drive timing is already present on most release versions of 6.1. The byte position of the SYS1 patch (for changing REMOVE back to KILL) should have been C8, not CB.

Patches, patches, patches ...

The following patch to LBASIC (versions prior to 09/31/83) will correct the operation of RUN"filespec",V for large programs:

```
. Patch to LBASIC/CMD (Model 1 ONLY!)
. corrects operation of RUN"",V
D0C,58=5E 64
D13,89=ED 62 39 D9 CD 4D 1B D9 F9 C9
. End of patch

. Patch to LBASIC/CMD (Model 3 and MAX-80 ONLY!)
. corrects operation of RUN"",V
D0C,6F=75 64
D13,A0=ED 62 39 D9 CD 4D 1B D9 F9 C9
. End of patch
```

The following patch to FM will correct a problem with not moving certain files:

```
. Patch to FM 5.1 to correct not moving certain HIT positions
D0F,4E=02
D19,62=62
. End of patch
```

```
. Patch to FM 6.x to correct not moving certain HIT positions
D0F,67=02
F0F,67=00
D19,C6=63
F19,C6=62
. End of patch
```

The following patch to QFB (all 5.1 versions) will provide for proper operation on double-sided media, and prevent a conflict with READ40 source drives:

```
. Patch QFB/CMD (5.1.x)
. corrects two-sided & READ40 operation
D02,26=00 00 00
X'5AAE'=CD C6 60
X'60C6'=CD 96 5C FD CB 03 A6 C9
. End of patch
```

The following patch corrects a problem in the version of XMODEM provided with the LS-Host/Term communications package:

```
. Patch to XMODEM from LS-Host/Term
. corrects problem with setting 8 bit word mode
D01,9E=C9
F01,9E=28
. End of patch
```

The following patch corrects an error in the SVC table for Model 1 LDOS 5.1.3 and 5.1.4

```
. Patch to correct SVC table entries for Model 1 LDOS 5.1.3 & 5.1.4
. this patch is to SYS7/SYS
D11,8B=44 30 44 33 44
. End of patch
```

LDOS: How it works - The BACKUP utility discussed

BACKUP functions and procedures discussed
or--- You can never have too many backups

by Joseph J. Kyle-DiPietropaolo

Long, long ago in a galaxy far, far away... oops- sorry. When the idea of a BACKUP utility was first implemented in a TRS-80 type DOS (Model 1 TRSDOS), the only designated purpose was to produce exact duplicates of existing diskettes. The BACKUP utility on LDOS 5.1.x and TRSDOS 6.x, however, wears many hats to serve a variety of purposes.

The first is, of course, to produce exact duplicates. One major difference between this mode of LDOS/TRSDOS 6.x BACKUP and the original variety is that LDOS/TRSDOS 6.x BACKUP (henceforth known as BACKUP) requires that the destination diskette be FORMATted first. The reason for this is simple: LDOS can handle many different disk drive setups. BACKUP can handle all of these, but only if the diskette was previously processed and made usable by LDOS through the FORMAT utility.

To produce an exact duplicate of a diskette, several things must be true about both the source and destination disks.

1) Both drives must be the same type. That is, they must both be five inch or both eight inch, the same density (single or double), and have the same number of sides.

2) Neither the source nor destination drive can be a hard disk system.

3) The destination drive must have an equal or greater number of cylinders than the source drive. For most people, a cylinder is the same as a track, but double-sided drives and hard disk systems actually have cylinders. A cylinder is a collection of tracks grouped together as one logical unit.

4) If the destination diskette has flaws (indicated during the FORMAT process), they cannot be on a cylinder that is occupied on the source drive. Generally, flawed diskettes should be discarded in any case.

When these conditions are met, and none of the special BACKUP parameters are specified (as described further on) BACKUP will be able to do what is called a "Mirror-image BACKUP". This is a misnomer-- the data is not reversed, as it would be in a mirror, but is copied identically from the source to the destination cylinder by cylinder.

This is the most common type of BACKUP. All other forms of BACKUP operations fall into the category of "BACKUP-by-class". If one or more of the above conditions are not met, then the BACKUP is done by copying each file on the source drive to the destination drive, one at a time. During this type of BACKUP, the BACKUP utility will display messages to indicate the type of BACKUP. "Backup-by-class invoked" means that the process was caused by a specification on the part of the user. "Backup-reconstruct invoked" means that BACKUP detected that one of the above conditions was not true.

Well, you may ask, what is a "BACKUP-by-class" good for? For this we must dig a little deeper into the parameters and specifications of BACKUP. One useful specification is the "partspec". A partspec is a portion of a normal LDOS file specification. For example, to move all files with the extension of /BAS, and that begin with "M". The command "BACKUP M/BAS:0 :1" could be used. The special partspec of $ means "all files".

To move groups of files, parameters can be used. For instance, "BACKUP :0 :1 (MOD)" would copy all files that had been modified since they were last backed up. In a DIR, this modified condition is indicated by a "plus" (+) sign next to the file.

Other parameters are available to backup files based on dates, visibility status, protection level, and whether or not the file already exists on the destination drive. With this introduction, all users should be able to use BACKUP more efficiently.

But what about frequency of backups? As a general rule, backups should be made at any significant break in a data processing procedure. That means at least every day a diskette is used. If a lot of processing is done, it wouldn't be a bad idea to perform backups more often, perhaps at mid-day in addition to at the end of the working day.

And how many sets of diskettes? Three is considered the absolute minimum. The sets should be used in a rotation system. For example, let's label the sets of disks A, B, and C. On the first day, set A is used. At the end of the day, set A is backed up onto set B. The next day, set B is used for processing. Set B is then backed up onto set C. Set C is used the next day, and at the end of the day, set C is backed up onto set A, and the cycle continues. In this manner, no set is used two days in a row, and new work is always done on the backup to ensure its integrity.

Many companies use five sets, labeled Monday through Friday. This helps prevent confusion as to what set is to be used, and provides additional backup protection. Each set is used on its labeled day, then backed up onto the next day's set. Six sets could be used if work is to be done on Saturdays.

What about re-formatting? Many people advocate periodically bulk-erasing and re-formatting the destination disk before a backup. This is a good idea, as this would be the only time that currently unused portions of the diskette would be checked for potential flaws.

What about diskettes themselves? Diskettes should be labeled with the date they are put into service. After a period of time, typically six months, they should be replaced with new diskettes, even if no difficulties were noted in their operation. The cost of even a premium diskette is trivial when compared to the value of the data it stores.

<u>Sending Characters to a Printer Via the Keyboard with a Single Keystroke (Whew!)</u>

by Dick Konop

An interesting customer service request prompted this article. The nature of the dilemma goes something like this: How can one pass an um-teen character control sequence to a printer directly from the keyboard with a single keystroke? One answer to this problem can be found in the use of KSM and MINIDOS.

The MINIDOS filter has a command (<CLEAR><SHIFT><P>) which will allow a (two character) hex byte to be entered from the keyboard. This byte is then sent to the printer. The KSM filter allows multiple keystrokes to be defined as a single key (each of the keystrokes <CLEAR><A> through <CLEAR><Z> can represent a different sequence of characters). To attain our final goal, a KSM file can be created which will invoke the MINIDOS filter, and pass it the hex control bytes.

The best way to illustrate this is by example. Let us assume that the bytes X'1B' and X'0F' need to be sent to the printer to produce the desired result. First, create a KSM file. One method of producing a KSM file is with the BUILD library command. If the BUILD command is used, the HEX parameter must also be specified. For example:

```
BUILD MOOSE/KSM (HEX)
```

After issuing the BUILD command, the prompt A --> will appear on the screen (if the extension for the filespec was /KSM). Respond to this prompt by entering the following characters (note that the spaces are for readability only, and must not be entered).

```
F0 31 42 3B F0 30 46 3B 0D
```

Once the KSM file has been built, the KSM and MINIDOS filters must be applied to the keyboard. The order in which the filters are applied is important. The KSM filter must be applied first, followed by the MINIDOS filter. For example:

```
FILTER *KI KSM MOOSE
FILTER *KI MINIDOS
```

After this has been done, depressing the <CLEAR><A> key sequence will cause the characters X'1B' and X'0F' to be sent to the printer.

A total understanding of what is happening is not required to use this concept. It is important to note that for each byte sent to the printer, four bytes are needed in the KSM file. The first byte will always be X'F0'. This is the character that will cause the MINIDOS "P" function to be activated. The next two bytes in the KSM file are the hex values corresponding to each hex digit in the byte being sent to the printer; That is to say, the "31" and "42" are the hex representations for the characters "1" and "B", respectively. These form the byte that will be sent to the printer (in this case X'1B'). The last byte in the four byte sequence will always be X'3B'. This is a semicolon character, and is translated by KSM into an <ENTER> (X'0D'). Finally, there must be a terminating X'0D' byte at the end of the KSM key assignment. This acts as a terminator for the KSM key definition. The X'0D' marks the end of all assignments made to the <CLEAR><A> KSM key.

Please note that when a KSM printer control key is pressed, the actual MINIDOS commands will appear on the screen (just as if they had been typed in). This type of printer control should be useable from within any program that allows use of KSM and MINIDOS.

THE JCL CORNER - by Chuck

For the last two years, I have attempted to use this column to shed some light on the subject of Job Control Language. Through examples both of my own design and also those of other readers, the many aspects of JCL have been examined and described. Rather than rehash all of this material again, I would rather devote this space to answering specific questions about the application of JCL procedures.

One particular question about using JCL procedures keeps coming to the attention of our customer service department. The question, "How can I use JCL to run (a particular program)?", can't always be answered with a simple set of instructions. Some programs as written can be run and controlled via a JCL procedure, while others can't. There are those that can be partially controlled, but still require some user keyboard input.

Future columns will attempt to deal with both previously mentioned subjects; answering specific user questions, and explaining how existing programs can be controlled with a JCL procedure. In addition, I will attempt to explain how a program can be designed to allow a JCL procedure to control it from start to finish. If any of you have questions, comments, or interesting uses for JCL, send them to LSI, attention "JCL Chuck".

Cumulative Index to LDOS Quarterly Volumes 1 and 2 - Subject

"Active Variable Dump for LBASIC" - Alan Moyer		v2n3p15	v2n4p38
"Alcor Pascal" - Scott Loomer			v2n1p18
Allocation, disk			v2n4p08
AM Electronics, disk controller			v1n3p03
"APL*PLUS/80 - A System Overview" - Daniel Lofy & Lee Rice			v2n1p09
"Article, An" - Charlie Butler			v1n4p40
"ASCII File Listing Utility for The BASIC Answer" - Jeffrey Brenton			v2n3p19
Assembly language basic concepts	v2n2p58	v2n3p46	v2n4p40
Assembly language patching		v1n6p38	v2n1p46
Assembly language programming tips		vln5p14	v1n6p64
"At Large" - Earl Terwilliger			v1n6p50
"Automatic Chaining with JCL" - Jim Kyle			v2n4p52
"BASIC and File Structure - A Beginner's View" - Wes Goodnough			v1n6p20
"BASIC Concepts - The RUN,V Command" - Dick Konop			v1n6p69
"Beta Tester, I was an LDOS" - Tim Daneliuk			v1n3p16
Byte I/O	v1n3p38	v1n6p76	v2n2p52
under LDOS 6.x			v2n3p56
'C' graphics			v2n4p16
"'C' language, The" - Earl Terwilliger:			
general introduction			v2n1p15
functions, variables, constants, expressions			v2n2p35
operators			v2n3p39
logic, control & flow			v2n4p34
initalization, blocks, pointers, arrays			v2n5p31
"'Card' Utility" - Paul Tonini			v2n2p18
"Case of Mis-Allocation, A" - Bill Schroeder			v2n4p08
Case mode indicator			v2n2p11
Changing operating systems			v1n4p40
"Clock Speedup Kits with LDOS, Using" - Tim Mann			v1n1p13
Cobol			v1n6p32
"Color Comes to the TRS-80's" - Scott Loomer			v2n3p28
"Communicating Micro, The" - Gordon Thompson		v1n6p12	v2n1p28
Communications		v1n5p40	v1n6p83
"Communication Host" - James Bruckart			v2n1p35

Cumulatlve Index to LDOS Quarterly Volumes 1 and 2 - Subject

LBASIC:		
active variable dump	v2n3p15	v2n4p38
"CMD"O" implementation and possible uses"		v1n4p15
default extension patch		v1n5p19
file structure		v1n6p20
notes (items missing from 1st printing of the LDOS manual)		v1n1p09
RUN,V command		v1n6p69
USR routines (relocating to high memory)		v1n3p47
LDOS 5.1.4, new features		v2n4p03
LDOS 6.x		
announced		v2n2p46
licensing		v2n4p07
technical manual		v2n4p07
patches	v2n5p35	v2n4p64
"LDOS: How it Works" - Joe Kyle-Dipietropaolo:		
PATCH utility		v2n1p52
REPAIR, CONV and COPY23B utilities		v2n2p44
non-Radio Shack disk drives		v2n3p55
configuring with non-relocatable code		v2n4p50
BACKUP utility		v2n5p36
"Les Information" - Les Mikesell:		
communications		v1n5p40
RS232 drivers		v1n6p83
SYSTEM (FAST) and (SLOW) commands		v2n1p58
byte I/O		v2n2p52
byte I/O under LDOS 6.x, CTLP/FLT		v2n3p56
@PARAM under LDOS 6.x		v2n4p61
"Let Us Assemble" - Rich Hilliard		
the basics		v2n2p58
using DEBUG, sorts		v2n2p46
number base conversion		v2n4p40
"Library, The" - Earle Robinson		v1n4p28
Library commands		
LOAD documentation correction		v2n1p47
SYSTEM (FAST)	v1n1p13	v2n1p58
SYSTEM (SLOW)		v2n1p58
Limited backup policy explained - Bill Schroeder	v2n3p04	v2n4p06
"Linking to LDOS in Assembly" - Roy Soltoff		v1n1p12
"LISP Implementations for the Z-80" - Lee Rice & Daniel Lofy		v1n6p26
Listing utility for TBA		v2n3p19
LScript made easy		v2n2p22
"LScript Patches to Add Versatility" - Scott Loomer	v1n6p45	v2n1p47
Load module structure		v1n4p42
Lower case lock		v1n4p11
LSI Journal submission and subscription policies		v2n4p02
MAX-80 LDOS described		v2n1p50
"MAX-80 Memory Map" - Chuck Jensen		v2n4p58
Magazines		v2n3p35
Manual story, with corrections to 1st edition		v1n1p16
Memory map		v2n1p34
Minimum configuration disk		v1n2p29
"Mixing Newscript, Electric Webster, LDOS and Sole" - Jerry Latham		v1n6p55
Modifying Mod 3 RTC Interrupts - Andrew Grandsen		v2n5p22
"My BASIC Answer" - E. Cheatham (TBA review)		v2n3p17
"Newscript 7.0 and REFLEX" - Gordon Thompson		v2n1p28
"Newscript and The BASIC Answer" - Jerry Latham		v2n2p20
"New Version - EDAS IV, A" - Marc Leager		v1n6p09
Number base conversion		v2n4p40
Old DAM's		v1n5p19
Parity errors		v1n1p04

Cumulative Index to LDOS Quarterly Volumes 1 and 2 - Subject

Cumulative Index to LDOS Quarterly Volumes 1 and 2 - Subject

Cumulative Index to LDOS Quarterly Volumes 1 and 2 - Programs

Cumulative Index to LDOS Quarterly Volumes 1 and 2 - Programs

FAST/CMD - Tim Mann - fast clock on the LX-80		v1n1p24
FASTSPL/ASM - fast despooling for Model I 5.0.3		v1n6p68
FILL/CCC - Scott Loomer - non-recursive graphic fill routine		v2n4p16
INBASKET/BAS - Gordon Thompson - communications message collector		v2n4p15
IOTA/FLT - Charles Knight - modification to SLASH0/FLT		v1n5p16
KILL/CCC - Earl Terwilliger - multiple file killer		v2n2p39
LBASIC/OV4 - Alan Moyer - active variable dump for Model I		v2n3p15
LISTITBA - Jeffrey Brenton - file lister for TBA		v2n3p19
MINIDOS/FLT - Roy Soltoff - MiniDOS for 5.0.x Mod I users		v1n2p44
MODCMD/CMD - John Mullin - send commands to a Hayes modem		v1n6p34
MX80/FLT - Ken Roser - block graphics on MX-80 w/o Graftrax		v1n1p22
MX80/FLT - improved version of above		v1n4p59
NODAM/CMD - Chuck Jensen - read non-LDOS disks without REPAIR	v1n4p58	v2n2p44
ONLINE/FLT - John Mullin - carrier detection filter		v1n6p37
PASSWORD/BAS - James Bruckart - password check for host systems		v2n1p36
PENCTL/FLT - Tim Mann - for Electric Pencil users		v1n1p22
PDUBL/CMD - original Percom doubler driver (for illustration only)		v1n1p24
PROMPT/CMD - Earle Robinson - change the 'LDOS Ready' prompt		v2n3p35
PRPARM/CMD - Graham Brown - changing PR/FLT parameters		v2n5p13
REFLEX/FLT - Gordon Thompson - communications program		v1n6p16
RENAME/BAS - Charles Knight - BASIC file renamer		v1n5p42
SALVAGE/BAS - transfer data from TRSDOS 2.3B disks	v1n2p15	v1n4p62
SLASH0/FLT - print slashed zeroes		v1n2p41
SLOSTEP/DCT - Les Mikesell - head settling delay for some DS drives		v2n1p60
SOLEFIX/ASM - Erik Ruf - fix SOLEd disk for compatibility		v2n2p14
SYNONYM/CMD - Henry Melton - command synonym processor		v2n5p14
TCHRON/ASM - Roy Soltoff - time board support routine		v1n6p66
TERMINAL/BAS - dumb terminal program		v1n6p86
TIME50/CMD - Andrew Grandsen - modify M3 RTC processing (50Hz)		v2n5p22
TIMEDI/FOR - J. Bender - Fortran interrupt and SVC demo		v2n4p22
UPCASE/FLT - Tim Mann - convert characters to upper case		v1n1p24
UPDATE/BAS - Tim Mann- update system disks (replaced by BACKUP)		v1n1p23
USRFREE/CMD - Chuck Jensen - LBASIC free memory command		v1n1p24

Cumulative Index to LDOS Quarterly Volumes 1 and 2 - Patches

BASIC compiler (RS) patch		v1n4p62
COBOL patch		v1n4p62
DEBUG ASCII mode patch		v1n5p45
DIRCHECK/CMD (LSI Utility Disk #1) patch for hard or DS drives		v2n2p44
DTERM patch - Peter Trenholme		v1n5p18
EDAS IV "Z" command patch - Earl Terwilliger		v2n1p37
FED patch		v2n1p47
File Manager patch for moving HIT entries		v2n5p36
File Manager patch for moving system files		v2n4p38
HIGH patch for screen full pause		v2n4p39
I/O bus disable for boot on Model III		v2n3p46
KI/DVR patch to extend keyboard debounce time		v2n3p45
LBASIC default extension patch		v1n5p19
LBASIC remove execution banner		v2n5p15
LBASIC RUN"",V fix		v2n5p35
LED xlate parameter patch		v2n1p48
Lower case driver forced on boot (Mod I)		v2n3p46
LScript patch to require filespecs for S or L		v2n1p62
LScript JCL patch		v1n6p13
LScriptX patches - Scott Loomer	v1n6p45	v2n1p47
MAP (LSI Utility Disk #1)		v2n1p47
MAX-80 patch to prevent printer driver from converting LF's to CR's		v2n2p39
MAX-80 patch to set LF after CR		v2n4p39
Microsoft patches		v1n2p16

Cumulative Index to LDOS Quarterly Volumes 1 and 2 - Patches

Cumulative Index to LDOS Quarterly Volumes 1 and 2 - Authors

Cumulative Index to LDOS Quarterly Volumes 1 and 2 - Authors

Developing an index turns out to be a very subjective procedure.

- apologies are hereby offered to anyone that feels slighted through omission or misstatement; it was unintentional -- Scott Loomer

LSI Quick Hint #2

With LDOS 5.1.4, using the library command "DIR partspec:n (A=N)" will provide the old-style "multiple-across" directory display without any patches to the system.

```
.CPROMPT/FIX
X'75F8'=CC 3A 7C CA 63 76 FE 6D CA 63 76
```

```
.RPROMPT/FIX
X'76BE'=02 7D
X'7D02'=CD 50 7C 21 2D 71 CD E5 7C 21 1C 7D 7E 02 23 CD 33
X'7D13'=00 7E FE 03 28 61 C3 0E 7D 0D 0D 0D 00 20 20 20 20
X'7D24'=20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 4D
X'7D35'=6F 75 6E 74 20 43 52 45 41 54 49 4F 4E 20 44 69 73
X'7D46'=6B 65 74 74 65 0D 20 20 20 20 20 20 20 20 20 20 20
X'7D57'=20 20 20 20 20 20 20 20 20 50 72 65 73 73 20 45 4E
X'7D68'=54 45 52 20 54 6F 20 43 6F 6E 74 69 6E 75 65 20 20
X'7D79'=03 AF CD 49 00 FE 0D 20 F8 C9
```

```
.RSPATCH/JCL
PATCH EFCM/CMD (X'886A'=C3 84 86)
PATCH EFCM/CMD (X'883C'=CD 78 86)
PATCH EFCM/CMD (X'8684'=AF 32 14 42 C3 2D 40)
PATCH EFCM/CMD (X'8678'=3E 20 EB 2B BE CA 7B 86 23 36 0D C9)
PATCH RM/CMD (X'716C'=31)
```

```
.PROFIX/JCL
. Auto patching for Profile + HD model III
.to operate on the Model I with either floppy or HD.
.Be sure the patch disk is in drive 0 and the
.disk containing the profile modules is in drive 1
//PAUSE Press <ENTER> to Begin
PATCH CM/CMD USING CM/FIX
PATCH RM/CMD USING RM/FIX
PATCH EFC1/CMD USING EFC1/FIX
PATCH EFC2/CMD USING EFC2/FIX
PATCH EFC3/CMD USING EFC3/FIX
PATCH EFC4/CMD USING EFC4/FIX
PATCH EFC5/CMD USING EFC5/FIX
PATCH EFC6/CMD USING EFC6/FIX
PATCH EFCM/CMD USING EFCM/FIX
PATCH EFCE/CMD USING EFCE/FIX
PATCH EFCF/CMD USING EFCF/FIX
PATCH EFC7/CMD USING EFC7/FIX
PATCH EFC8/CMD USING EFC8/FIX
PATCH EFC9/CMD USING EFC9/FIX
PATCH EFCA/CMD USING EFCA/FIX
PATCH EFCB/CMD USING EFCB/FIX
PATCH EFCC/CMD USING EFCC/FIX
PATCH EFCD/CMD USING EFCD/FIX
```

```
.PROMPT/JCL
.This patch allows FLOPPY usage
PATCH CM/CMD USING CPROMPT/FIX
PATCH RM/CMD USING RPROMPT/FIX
```

```
.CM/FIX
X'7BC2'=67 44
X'74B8'=18 43
X'7652'=18 43
X'765E'=18 43
X'74BB'=19 43
X'7661'=05 44
X'7832'=05 44
X'761A'=49 40
X'761E'=49 40
X'7617'=01 30
X'782C'=01 30
X'7ABE'=01 30
X'7B3F'=01 30
X'7151'="ONE"
X'7172'=31
X'74F6'=63
```

```
.EFC1/FIX                                    .EFCB/FIX              .EFCF/FIX
X'87FA'=67 44         X'5801'=5D 44          X'6681'=67 44          X'88AB'=67 44
X'846A'=05 44         X'5825'=5D 44          X'62F1'=05 44          X'7ADC'=18 43
X'8464'=01 30         X'5849'=5D 44          X'62EB'=01 30          X'7AF3'=18 43
X'86F6'=01 30         X'588E'=5D 44          X'657D'=01 30          X'7B01'=18 43
X'8777'=01 30         X'590D'=5D 44          X'65FE'=01 30          X'851B'=05 44
X'8397'=C2            X'5A03'=5D 44          X'5E4B'=C1 44          X'801B'=49 40
X'89B0'=60            X'60F4'=60             X'5E46'=C4 44          X'8515'=01 30
                                             X'56ED'=18 43          X'87A7'=01 30
                                             X'570B'=18 43          X'8828'=01 30
.EFC2/FIX             .EFC8/FIX              X'66FF'=60             X'82F3'=60
X'7E12'=67 44         X'56F2'=67 44
X'7A82'=05 44         X'543C'=01 30
X'74BE'=22 40         X'5668'=01 30          .EFCC/FIX              .EFCM/FIX
X'70C3'=5B            X'56E9'=01 30          X'56E7'=67 44          X'8CFF'=67 44
X'4022'=8F            X'707B'=01 30          X'9625'=67 44          X'7379'=96 43
X'4023'=00            X'70AD'=01 30          X'97E9'=67 44          X'7601'=05 44
X'7E90'=60            X'7BAF'=01 30          X'9FFD'=67 44          X'8843'=05 44
                      X'7BE7'=01 30          X'A01F'=67 44          X'8986'=05 44
                      X'7BDE'=18 43          X'565D'=01 30          X'76EC'=C2
.EFC3/FIX             X'7BED'=18 43          X'56DE'=01 30          X'7423'=18 43
X'8109'=67 44         X'8DE7'=18 43          X'73B1'=01 30          X'75D2'=18 43
X'7D79'=05 44         X'8DFE'=18 43          X'73D5'=18 43          X'75FE'=18 43
X'7CA6'=C2            X'8E51'=18 43          X'73DD'=18 43          X'8835'=18 43
X'8187'=60            X'5442'=05 44          X'9228'=18 43          X'8840'=18 43
                      X'70D4'=05 44          X'9248'=18 43          X'882B'=01 30
                      X'7BF0'=05 44          X'9298'=18 43          X'8980'=01 30
.EFC4/FIX             X'8DD9'=49 40          X'73E0'=05 44          X'8BFB'=01 30
X'7FE9'=67 44         X'9730'=49 40          X'7406'=05 44          X'8C7C'=01 30
X'7C59'=05 44         X'6DF8'=4C 44          X'9224'=49 40          X'8D7D'=60
X'7C53'=01 30         X'6E06'=4C 44          X'587B'=60
X'7EE5'=01 30         X'5764'=5A 44
X'7F66'=01 30         X'564C'=5D 44
X'8067'=60            X'74C9'=60             .EFCD/FIX
                                             X'5C40'=67 44
                                             X'58AA'=01 30          .RM/FIX
.EFC5/FIX             .EFC9/FIX              X'5B3C'=01 30          X'771A'=05 44
X'7D66'=67 44         X'E441'=67 44          X'5BBD'=01 30          X'7770'=05 44
X'79D6'=05 44         X'BFA3'=70 44          X'805D'=18 43          X'779C'=05 44
X'79D0'=01 30         X'E39C'=05 44          X'8074'=18 43          X'7956'=05 44
X'7C62'=01 30         X'7B7E'=18 43          X'8082'=18 43          X'7598'-18 43
X'7CE3'=01 30         X'C5F8'=5D 44          X'58B0'=05 44          X'770F'=18 43
X'81A1'=60            X'7B74'=00 00 00       X'5CBE'=60             X'7717'=18 43
                      X'E6B1'=60                                    X'7765'=18 43
                                                                    X'776D'=18 43
.EFC6/FIX                                    .EFCE/FIX              X'778E'=18 43
X'7AE8'=67 44         .EFCA/FIX              X'7D80'=67 44          X'7796'=18 43
X'7758'=05 44         X'7033'=67 44          X'79F0'=05 44          X'759B'=19 43
X'7B66'=60            X'6758'=70 44          X'791D'=C2             X'75E0'=63
                      X'6C9D'=01 30          X'79EA'=01 30          X'7C63'=01 30
                      X'6F2F'=01 30          X'7C7C'=01 30          X'7BE2'=01 30
.EFC7/FIX             X'6FB0'=01 30          X'7CFD'=01 30          X'7950'=01 30
X'6076'=67 44         X'5B55'=18 43          X'81B7'=60             X'76CE'=01 30
X'5CE6'=05 44         X'5B73'=18 43                                 X'714B'="ONE"
X'57D2'=5D 44         X'6CA3'=05                                    X'7CE6'=67 44
                      X'6A7B'=60
```

SUPERSCRIPSIT PRINTER DRIVERS $49/$59 (Dep. on printer)

With an ALPS printer driver, it is possible to attach non-Radio-Shack-supported printers to the powerful SuperSCRIPSIT Word Processing System. **More than 30 different printers are now supported** with customized drivers, which automatically handle the interface to the printers. Each driver supports all the normal text processing (margins, tabs, centering, etc.), plus whatever features the printer can support. The printer driver is a software product, which is easy to install and use.

FOUR MODEL 4 UTILITIES (FOR MODEL III MODE) $19

With the following ALPS Utilities, you can easily access some of the Model 4 hardware features while in Model III mode. The following utilities are designed to run on a TRS-80 Model 4 disk system equipped with either 64K or 128K of RAM memory, and executing in Model III mode. These utilities can be used on any operating system.

CPUFAST can be used to speed up processing speed. It sets the CPU clock speed to 4 Mz. This will cause programs to execute up to 50 percent faster than normal. The CPUFAST command need only be typed once. The Model 4 will remain in high speed mode until is is rebooted, or until the CPUSLOW command is executed. The CPUSLOW program allows returning processing speed back to the normal 2.0 Mz. rate (Model III compatible rate) without re-booting the system.

RAMTEST will test Model 4 memory by writing and reading data from each RAM location. It will continue endlessly until the BREAK key is presssed or until it encounters a bad location. The following areas of RAM are verified:
Std. RAM, ROM Overlay, First & Second 32K expansion bank (if 64K exp installed).

CLEARMEM will set all Model 4 memory (as described in RAMTEST above) to the value specified in BYTE='xx'. If no BYTE= parameter is specified, all memory is initialized to zero. This routine is useful for system developers.

BEEP will cause a beeping sound to be generated by the Model 4 sound generator circuitry. The optional LOUD parameter results in a harsher, louder beep. The optional CYCLES= parameter specifies the number of times that the beep cycle is repeated. Useful applications of BEEP are in /JCL files to signal an operator for keyboard input or to audibly indicate when a job has completed. BEEP can also easily be used from within BASIC programs via the BEEPSUB callable subroutine. The file BEEPTEST/BAS is a sample program which contains an example of the required calling sequence.

BASIC PROGRAM SEARCH UTILITY $19

Search for any string within a BASIC program with an easy CMD command. Display all occurrences on screen. With this Search Utility, you can scan your program automatically, rather than visually for greater accuracy. LDOS only. Model I/III/4(III mode).

Great for all BASIC programmers!

For example: **CMD "BS /QC/"** will reveal whether or not you have used the name "QC" in your BASIC program. Great way to prevent duplicate variable names.

For example: **CMD "BS /AB/"** will find all occurrences of the variable AB if you want to change the name to another name. Any missing lines that you know should be listed would indicate a mistyped line.

MEMDISK (MODEL 4 IN MODEL III MODE) $39

Easy to use. (LDOS only). Use MEMDISK as your System Drive or load other files into memory for Instant Access. Use **80K** of 128K memory. The MDISK utility creates a RAM memory disk drive using 80K of expansion memory avail- able in a Model 4 equipped with the 128K upgrade option. It operates under LDOS Version 5.1 in Model III mode only. MDISK creates a single 80K logical drive which acts nearly identical to a floppy disk but at greatly improved access speed. This should support all applications which do not directly interface with the floppy disk controller hardware, or do not depend on the 180K physical capacity of a floppy disk.

CREATE option is used to define a single 80K MDISK logical drive. The logical drive will be automatically formatted and can be used immediately after creating.

FLOPPY option is used to create a floppy disk logical drive. An existing MDISK drive could be redefined to its original floppy configuration using this command.

LOAD=... option is used to quickly build an MDISK from the contents of a file created by a previous SAVE. Note that a **High Speed MDISK System Drive** can be created. MDISK :0 (LOAD="SYS0") will load an MDISK drive saved with Filespec name "SYS0/MDK" directly into drive 0.

NULL option removes a logical drive completely from this system. An MDISK drive, if NULLed, can be recovered using the WARM option.

SAVE=... option is used to quickly save, in the form a single file, an image of the current MDISK drive. Only allocated cylinders of the MDISK drive are written to disk which greatly reduces the size of the generated file for MDISKs which are not full.

WARM option allows recovery of an MDISK drive when standard RAM memory has been changed, but expansion memory remains unaltered. This command can be used after a system reboot to recover a pre-existing MDISK drive.

MEMORY DUMP UTILITY $24

HEX/ ASCII screen display of memory pages (Incl. exp. 64K, low & std. RAM, ROM) with one easy command. For Model 4 (Mod III mode), any oper. sys. **(80X24 screen)**.

LOADMAP UTILITY $19

Obtain a graphic screen display of object program load address ranges. One command. Model III or Model 4 (Mod III mode), any operating system.

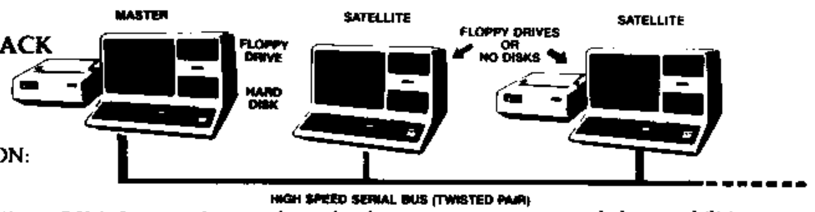
All of the above software products are immediately available with full documentation. You may order either by telephone or mail. Mastercard and Visa are accepted. (For UPS COD, add $3). Orders are sent via First Class Mail, unless otherwise requested.

A L P S, 23 Angus Road, Warren, New Jersey 07060
201-647-7230

Networking FOR RADIO SHACK

MODELS III & 4...NOW!!

TYPICAL CONFIGURATION:


MASTER
FLOPPY DRIVE
HARD DISK
SATELLITE
FLOPPY DRIVES OR NO DISKS
SATELLITE
HIGH SPEED SERIAL BUS (TWISTED PAIR)

After comprehensive field testing in medical offices, PSM, Inc. is releasing the technology necessary to expand the capabilities of your Model III/4 to MULTI-USER, MULTI-PROCESSING, SHARED DATA APPLICATIONS using Winchester 5.25" hard disks. Each workstation is a complete, independent computer which is tied to the hard disk through a high-speed serial bus. Workstations DO NOT NEED ANY DISK DRIVES; they will boot up properly through the serial bus from the hard disk. It will be running under LDOS as if the hard disk was its drive: 0 !

FEATURES

- **Protects existing software investment**
- **Data file sharing**
- **Compatible with LDOS 5.1.4**
- **Shared hard disk [5 to 20 megabytes]**
- **File/record lockout capability**
- **Used in active medical offices since 2/83.**

PSM-W5:	Model 4 w/64k RAM, 40trk DS Floppy, built-in 5 Mbyte hard disk, master network controller, LDOS 5.1.4, Technical Manual	$4,975.00
PSM T/0:	Model 4 w/64k RAM, boot PROM, satellite network controller, Tech Manual	$1,875.00
PSM-U5:	PSM upgrade of your Model III/4 w/64k RAM to PSM-W5: incld's 5Mb hard disk, hard disk controller, master network controller, LDOS 5.1.4, power supplies, Tech Manual	$2,625.00
PSM-U0:	PSM upgrade of your Model III/4 w/64k RAM to PSM T/0: incld's boot PROM, satellite network controller, Tech Manual	$665.00
PSM-NM:	Technical Manual: Incld's complete description of PSM networking, and application implementation Tech Manual purchase may be credited toward later hardware purchase	$20.00

PSM programming and systems management, inc.

All shipping and handling charges are the customers' responsibility. Technical consultation services available. Modification to Model III/4 to implement networking will void the Tandy warranty.

3866 INDIAN RIPPLE RD., DAYTON, OH 45440 [513] 426-8644