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 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

Smallware, Propack, Quickback, and Reface are trademarks of The Small Computer Company, Inc. COM is a registered trademark of Digital Research, Inc. Profile is a registered trademark of Radio Shack.
Table of Contents

INTRODUCTION FROM LSI:

ARTICLE SUBMISSION POLICY ................................................ Page 2
VIEW FROM THE BOTTOM FLOOR ............................................... Page 3
NEW PRODUCT ANNOUNCEMENTS ................................................ Page 6

FROM OUR USERS:

EZEDIT/FLT - The LDOS command line editor ............................... Page 7
PRPARAM - Change those PR/FLT parameters ............................... Page 13
SYNONYM - The LDOS command synonym processor ........................ Page 14
Mod 3 RTC Interrupts - discussed and modified .......................... Page 22
Profile One Plus - A special treat for Mod 1 users ....................... Page 26

REGULAR USER COLUMNS:

*** PARITY = ODD *** - Tim Daneliuk ................................. Page 28
'C' What's Happening - Earl Terwilliger ............................... Page 31

FROM THE LDOS SUPPORT STAFF:

ITEMS OF GENERAL INTEREST ................................................. Page 35
LDOS: HOW IT WORKS - The BACKUP utility ............................... Page 36
Who, what, where, why, when (and how) discussed

Sending Printer codes - A special application .......................... Page 38

THE JCL CORNER - by Chuck ............................................... Page 39

Special Feature:

The LSI Journal and LDOS Quarterly Index ............................... Page 39

by Scott Loomer

Copyright (c) 1983/1984 by Logical Systems, Incorporated
8970 N. 55th Street P.O. Box 23956
Milwaukee, Wisconsin 53223
Main switchboard: (414) 355-5454
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

UNIX™ is a trademark of Bell Laboratories
MAX-80™ is a trademark of LOBO Systems, Inc.
PC-DOS™ and IBM-PC™ are trademarks of IBM Corp.
TRSDOS™ is a trademark of Radio Shack/Tandy Corp.
MS-DOS™ and XENIX™ are trademarks of Microsoft, Corp.
WordStar™ is a trademark of MicroPro International Corp.
CP/M™, CP/M-80™, and CP/M-86™ are trademarks of Digital Research, Inc.

The LDOS Quarterly is copyrighted in its entirety. No material contained herein may be duplicated in whole or in part for commercial or distribution purposes without the express written consent of Logical Systems, Inc. and the article's author.
VIEW FROM THE BOTTOM FLOOR

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.

SUPER SALES and VALUES

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-CREATR) 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 offer 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, FULL 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.
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.
The first part of the coding loads an initial message
checks that *KI has been referenced, and for KI/DVR and MINIDOS being active.

The first part of the coding loads an initial message
checks that *KI has been referenced, and for KI/DVR and MINIDOS being active.
A good place to start
Put *KI DCB into
IX register for later
This section of code corrects all model specific references in the code.

A good place to start
Put *KI DCB into
IX register for later
This section of code corrects all model specific references in the code.

See if *KI specified
in the filter command
error if not.
Point to message....
.and print it.
System flag area
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.
00950  LD HL,ZAPMSG ;Point to message....
00960  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 $-2
01320  NO_DO LD (VECTAD),DE ;Store in EZ-Edit
01330  LD HL,(HIGH1$) ;HL = Present High Memory
01340  LD HL,(LAST-START) ;Get length of filter
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 ;to new start
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 ;recovery start
01580  PUSH HL ;and keep going
01590  LD DE,CONTENT-START
ADD HL,DE
LD (MODIFY3),HL
LD (MODIFY8),HL
POP HL
PUSH HL
LD DE,XORVAL-START
ADD HL,DE
LD (MODIFY4),HL
LD (MODIFY5),HL
POP HL ;DE = Start of EZ-Edit
PUSH HL ;save it. DE = To
LD HL,START ;and HL = From
LDIR ;BC = Count, so move it.!
POP DE ;DE = Start of EZ-Edit
PUSH DE ;save it. DE = To
LD HL,START ;and HL = From
LDIR ;BC = Count, so move it.!
DI ;Don't interrupt a minute
LD A,(SFLAG1$) ;Test for JCL again
M4 EQU $-2
BIT 5,A
JR NZ,DO_ON ;Skip if ACTIVE, else
LD (IX+01H),L ;Load KIDCB with address
LD (IX+02H),H ;of EZ-Edit
JR OUT
DO_ON LD (KIJCL1$),HL ;JCL, so load KIJCL$
M9 EQU $-2
OUT EI ;Interrupts on, and
LD B,62 ;how many spaces
LD HL,INBUF1$ ;point to inbuf$
M15 EQU $-2
LD A,20H ;space
HERE LD (HL),A ;store it
INC HL ;point to next
DJNZ HERE ;go until done
LD A,0DH ;CR
LD (HL),A ;put at end of buffer
JP @EXIT ; now back to DOS

Table of zaps for MINIDOS zapping routine
TABLE DB 21H,10H,00H,0EDH,7AH,0F9H,0E1H,0C9H,00H,00H

;Error if KI/DVR not established
KINOTON LD HL,KIOFFMS ;Point to message
GO_OUT CALL @LOGOT1 ;LOG and DISPLAY
M12 EQU $-2
JP @ABORT ;Abnormal exit to DOS

;Error if *KI not referenced
WRONGDV LD HL,NOTKIMS ;Point to message
GO_OUT ;Log, display, and DOS

; Messages
LOGMSG DB 'EZ-EDIT - LDOS command line editor. Version 5.1',0AH,'(c) 1983 by G.Brown. All rights reserved',0DH
KIOFFMS DB 0AH,'KI/DVR not installed!',0DH
NOTKIMS DB 0AH,'Filter ONLY using *KI device!',0DH
ZAPMSG DB 0AH,'MINIDOS detected - patching...',0DH

;EZ-Edit : The actual relocated filter
START JR BEGIN
OLDHI DW 0000H
DB 4,'Edit'
ONOFF DB 00H ;Flash character store
CONTENT DB 00H ;The contents of DE before flash
XORVAL DB 00H ;The XOR character for the flash
BEGIN CALL 00000H ;Call to KI/DVR, and MINIDOS
VECTAD EQU $-2
CP 0EFH ;See if SHIFT+CLEAR+O
; Got a SHIFT+CLEAR+O here
PUSH HL ; Save what we use
PUSH DE
PUSH BC
PUSH IX
LD DE,(CURSOR) ; Get the cursor location
LD A,E ; and make sure that
AND 0C0H ; its at the far left of
LD E,A ; the screen.
LD (CURSOR),DE
PUSH DE ; Save the address of the
PUSH DE ; line - TWICE
; Get the last command from INBUF$, and put on screen
; Turn the cursor off afterwards.
LD HL,INBUF1$ ; Point to input buffer
MS EQU $-2
GETLOOP LD A,(HL) ; and get all characters
CP 0DH ; up to but NOT including
JR Z,GOT_CR ; a CR, and put on the
INC HL ; screen
CALL @DSP
JR GETLOOP
GOT_CR LD A,0FH ; Cursor OFF now
CALL @DSP
POP DE ; Restore line address
PAGE OFF
; Scan for an input, and flash the character with the
; (CHR$(95)). Allowable inputs are:
; LEFT ARROW - backspace without erase
; RIGHT ARROW - cursor right without erase
; SHIFT+CLEAR - erase all from cursor to the right and
; process the command to the left.
; ENTER - process the line as is. Characters to the
; left and right of the cursor make up the
; command
; BREAK - Back to DOS
FLASH LD A,5FH ; Cursor character
LD (ONOFF),A ; store it
MODIFY2 EQU $-2
LD L,A ; and save in L
LD A,(DE) ; Get character in DE
LD (CONTENT),A ; and save for later
MODIFY3 EQU $-2
XOR L ; XOR to get flash value
LD (XORVAL),A ; and save it.
MODIFY4 EQU $-2
LD HL,XORVAL ; Point to store area
MODIFY5 EQU $-2
LD A,(ONOFF) ; Get one character
MODIFY6 EQU $-2
XOR (HL) ; XOR
LD (ONOFF),A ; save back for flash
MODIFY7 EQU $-2
LD (DE),A ; and put on screen
LD BC,150H ; Timer for scanning
CALL @KBD ; and look at the keyboard
POP DE ; get DE again
OR A ; Any input ???
JR NZ,INPUT ; Jump if so, else....
DEC INPUT ; see if BC = 0
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 $-2 ;original character
03110 LD (DE), A
03120 POP AF ;recover input, and
03130 CP 08H ;is it LEFT ARROW ?
03140 JR Z,LARROW
03150 CP 09H ;RIGHT ARROW ?
03160 JR Z,RARROW
03170 CP 09H ;ENTER ?
03180 CP 1FH ;SHIFT+CLEAR ?
03190 JR Z,CLEARPR
03200 CP 01H ;BREAK ?
03210 JR Z,BREAK
03220 CP 20H ;Is it ASCII 2$H-7FH
03230 LD A,0FH ;Cursor OFF
03240 CALL @DSP
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.
03340 ;Deal with the input. Editing allowed on one line only.
03350 LARROW LD A, E ;Point DE to end of
03360 AND 3FH ;the line
03370 CP 1 ;move further left.
03380 JR C,NODEC
03390 DEC DE ;else DECrement
03400 NODEC JR FLASH
03410 CLEARPR LD A, 1EH ;Erase to end of line
03420 CALL @DSP ;and clear the line.
03430 LD A, 0FH ;Cursor OFF
03440 CALL @DSP ;and restore DE
03450 POP DE ;and restore DE
03460 ;How long is the command ?? - go to right of line, and
03470 ENTER LD A, E ;find first non blank character on the left.
03480 OR 3FH ;the line
03490 LD E, A
03500 BKLOOP LD A, (DE) ;and look back....
03510 DEC DE
03520 CP 20H
03530 JR Z,BKLOOP
03540 INC DE ;DE = space after command
03550 LD (CURSOR), DE ;load the cursor
03560 LD (4020H), DE ;tell the cursor where,
03570 CALL @DSP ;and clear the line.
03580 LD A, 0FH ;Cursor OFF
03590 CALL @DSP ;and restore DE
03600 ;Move the new command into INBUF$, and execute via
03610 A ND DE ;@CMNDI address.
03620 LD A, E ;Get column number into A
03630 AND 3FH
03640 LD C, A ;Load length of the
03650 LD B, 0 ;command into BC
03660 POP HL ;Point HL to line start
03670 LD DE, INBUF1$ ;DE = Destination

Page 12
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


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:

```
0 5 0 65 0 5 25 0 4 15 2 4 D0 1 0 20 0 5 2E 5 2 16 3 5 3C D 6 74 4 3 A2 5 0 1F E 4 92 0 3 12 1 1 F4 4 2 24 0 5 22 1 8 9 4 2 2 24 8 5 22 1 5 44 4 2 26 C 5 23 E 0 13 2 C 85 2 2 18 A 4 22 2 5 E5 3 3 AF F 5 23 D 3 2F F 5 23 2 B 75 2 3 A0 8 5 3 3 D 3 20 8 5 33 2 B A5 2 3 A3 E 4 0F E 5 1C 2 5 65 3 2 11 F 4 4C B 5 EC A 5 A5 3 F D2 A F 64 D E 17 E F E0 D 2 80 7 F E 2 0 2 00 B 2 31 8 F 43 E 0 D3 2 0 75 5 C 3C C 5 21 1 6 B5 5 C D7 6 4 4C 2 5 25 3 D D2 1 2 54 0 2 1F F F FC D 1 E5 3 2 80 3 F D7 7 1 72 1 F FF F C D1 E 5 32 8 0 3 F D7 7 1 A2 1 F FF F C D1 E 5 32 8 0 3F D7 7 0 32 1 F FF F 2 37 C B 52 8 0 72 B F D7 4 6 DF D 7 57 1 2 10 0 0 07 C B 5 2 8 9 E3 E 0 CC D 3 B0 0 3 E0 0 D D7 7 0 42 6 0 0F D 6 E1 7 1 1A 5 5 4C D 2 F5 3 F D6 E 1 91 1 B E5 4 C D2 F 5 3F D 6 E 1 A 1 18 C 5 4C D 2 F5 3 3 A2 8 4 06 F 1 1F E 5 4C D 2 F5 3 3 A2 A 4 06 F 1 13 0 5 5C D 2 F5 3 F D6 E 6 D0 1 0 20 0 5 3 1 1 3 F5 5 C D2 F 5 3F D 6 E7 1 1 16 0 5 5C D 2 F5 3 2 15 8 5 4C D 6 74 4 2 11 D 5 5C D 6 74 4 C 32 D 4 02 3 2 42 5 2 0 2 E 7 CB 5 C 82 8 7 DC 9 6 40 0 0 A0 0 0 10 0 D D2 1 2 95 3 A FD D 4 60 1 D D4 E 0 0B 7 E D4 2 3 80 3 3 C1 8 F 90 9 C 6 3 0 1 21 3 7 9F E 0 1C 8 D D2 3 D D2 3 1 8E 2 E 12 1 0 75 4 D D2 1 C 95 3 D D2 1 E 75 3 C D7 B 4 4C 3 3 04 0 0 A5 0 5 2 5 0 4 15 2 4 D2 0 2 D2 0 7 06 1 7 26 1 6 D6 5 7 46 5 7 22 0 6 D6 F 6 46 9 6 66 9 6 57 2 2 06 6 6 F7 2 2 05 0 5 22 F 4 6 C 4 5 4 2 06 9 7 32 0 6 E6 F 7 42 0 7 96 5 7 42 0 6 96 E 7 37 4 6 10 1 0 20 0 5 46 C 6 C6 5 6 42 0 2 10 D 0 A0 A 5 06 1 7 2 6 1 6 D6 5 7 46 5 7 22 0 6 57 2 7 26 F 7 22 0 2 D2 0 7 47 2 7 92 0 6 16 7 6 16 9 6 E2 0 2 10 A 4 16 C 6 C6 F 7 76 1 6 2 6 C 6 52 0 7 06 1 7 26 1 6 D6 5 7 46 5 7 27 3 2 06 1 7 26 5 3 A2 0 4 D2 C 4 92 C 4 32 C 4 C2 C 5 02 C 4 64 6 2 C6 1 6 E 6 4 2 07 0 6 17 2 6 16 D 6 5 7 4 6 57 2 7 32 0 6 17 2 6 52 0 3 A0 A 0 A2 0 2 02 0 2 02 0 4 D4 1 5 24 7 4 94 E 2 03 D 2 05 8 5 85 8 2 07 3 7 06 1 6 3 6 5 7 30 A 2 02 0 2 02 0 2 02 0 4 3 4 8 4 15 2 5 32 0 8 15 2 4 32 0 2 02 0 2 02 0 8 15 2 4 94 E 4 44 5 4 E5 4 7 65 2 4 92 0 2 02 0 2 02 0 7 65 2 4 C4 9 4 E4 5 5 3 2 0 9 75 2 4 C2 0 2 02 0 2 02 0 9 75 2 5 04 1 4 74 5 2 02 0 A 25 2 5 02 0 2 02 0 2 02 0 A 25 2 4 64 6 2 02 0 2 02 0 B C 5 2 5 84 C 4 15 4 4 52 0 A D5 2 5 82 0 2 02 0 2 02 0 A D5 2 0 00 2 0 20 0 5 2
```
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:

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

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'541B'=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 dl dir :1 &
2 do 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 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")
6 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 0000 00 with CD67 44.

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 8222 2A52 2322 9A52 22A8 5221 2542 229B 5322 7B54 2B22 F953 2199 4222 A453 DD21 2EB5 2A49 4022 5C53 1156 56B7 ED52 444D 3E16 DD6E 00DD 6601 235E 2356 EB09 EB72 2B73 DD23 DD20 89ED 5B49 4021 5656 01FD 02ED 88ED 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 666E 6520 7379 6E6F 6E79 6D20 7072 6F63 6573 736F 7220 0A43 6F70 7922 6967 6874 6564 2031 3938 332C 2048 656E 7279 204D 656C 746F 6B0D 4F6E 6C79 0102 0053 2076 616C 6964 2061 7420 4C44 4F53 2052 6561 6479 202D 2D20 6D63 6E77 7461 6C6C 6174 696F
; SYNONYM -- Command line synonym processor for LDOS
; Copyright 1983 by Henry Melton

; The synonym processor resides in high memory and is linked into the error
; reporting chain. When the error code 95 appears, indicating that a Program not
; found error is to be flagged, SYNONYM reads the command buffer and uses the
; first word of the contents as a substitution key for an acceptable alternate
; command string. A file with the name SYNONYM/TXT is needed that contains the
; substitution data.

; Synonym records are searched sequentially til either a match or EOF occurs. If
; the new command line fails, then the processor tries again with this new
; input. When & substitution occurs, there is the possibility of the resulting
; command line exceeding the 64 character command buffer. If so, the line will
; be truncated to 64 characters.

ECHO EQU 1 ; Display the generated command = 1
@DSPLY EQU 4467H
@EXIT EQU 402DH
@GET EQU 0013H
@OPEN EQU 4424H

HIGH1$ EQU 4049H
INBUF1$ EQU 4318H
@ICNFG1 EQU 4303H
@CMNDI1 EQU 4405H
HIGH3$ EQU 4411H
INBUF3$ EQU 4225H
@ICNFG3 EQU 421DH
@CMNDI3 EQU 4405H

ORG 5200H

START:: LD HL,BANNER ; Display the program name and copyright
CALL @DSPLY
LD A, (@EXIT)
CP 'I' ; will be "I" if Model 3
JR NZ,MOD1 ; continue if Mod 1
LD HL,HIGH1$ ; Pick up Model 1 locations and
LD (MODCH1),HL
LD (MODCH2),HL
LD (MODCH3),HL
LD HL,HIGH3$ ; Pick up Mod 3 locations and
LD (MODCH1),HL
LD (MODCH2),HL
LD (MODCH3),HL
LD HL, @ICNFG3

OKGO: LD A, (0125H) ; Pick up type of machine flag
CP 'I' ; will be "I" if Model 3
JR NZ,OKGO ; if so, we aren't at LDOS Ready
LD HL,ABORTMS ; and can't install/modify the system
LD (MODCH1),HL
LD (MODCH2),HL
LD (MODCH3),HL
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 MODCH2 EQU $-2
01130 PTLOOP: LD L,(IX+00H)
01140 INC HL
01150 LD E,(HL)
01160 INC HL
01170 LD D,(HL)
01180 EX DE,HL
01190 INC HL
01200 EX DE,HL
01210 LD (HL),D
01220 INC HL
01230 LD (HL),E
01240 INC IX
01250 INC IX
01260 INC A
01270 JR NZ,PTLOOP
01290 MODCH3 EQU $-2
01300 LD DE,(HIGH1$) ;Move the module to its high memory
01310 MODCH4 EQU $-2
01320 LD HL,LAST-1 ;location
01330 LD BC,LAST-FIRST
01350 MODCH5 EQU $-2
01360 LD (HIGH1$),DE ;Put the new HIGH$ value in storage
01370 MODCH6 EQU $-2
01380 REL0: LD HL,(400DH) ; Plug the primary RST 40
01390 REL1: LD HL,BEGIN ; vector with the SYN processor
01400 REL2: LD (400DH),HL ; address.
01410 REL3: LD A,(@ICNFG1) ; Do the same for the CONFIG
01420 MODCH4 EQU $-2
01430 JP @EXIT ;end of the loading operation
01540 BANNER: DEFM 'SYNONYM -- LDOS command line synonym processor '
01550 DEFB 0AH
01560 DEFM 'Copyrighted 1983, Henry Melton'
01570 DEFB 0DH
01580 ABORTMS: DEFM 'Only valid at LDOS Ready - installation aborted!'
01590 DEFB 0AH
01600 DEFB 0DH
01620 RELOTB: DEFW REL1 ;Table of addresses to relocate to HIGH$
01630 DEFW REL0
01640 DEFW INIT
01650 DEFW RELX
01660 DEFW RELXX
01670 DEFW RELY
01680 DEFW RELZ
01690 DEFW INTERCPT
01700 DEFW LOOP1
01710 DEFW REL2
01720 DEFW PROCESS
01730 DEFW REL3
01740 DEFW OPENSYN
01750 DEFW REL4
01760 DEFW REL5
01770 DEFW LOADFCB
01780 DEFW REL7
01790 DEFW READLN
01800 DEFW LOOP2
01810 DEFW COMPARE
01820 DEFW REL8
01830 TBLEND: DEFW MOVELN
01840 TBSIZE EQU TBLEND-RELOTB+2
01850 ENTRYS EQU TBSIZE/2
01870 FIRST: JR BEGIN
01880 STORE: DEFW 0 ;to receive the old HIGH$ value
01890 NAME: DEFB BEGIN-TEXT
01900 TEXT: DEFM 'SYNON'
01910 BEGIN: PUSH AF
01920 CP 86H ;A true error code has
01930 JR Z,CHECK2 ; an 86H here.
01940 QUIT: POP AF ; so if not, go about
01950 DEFB 0C3H ; your business.
01960 CHECK2: INC SP
01970 INC SP
01980 INC SP
01990 INC SP
02000 INC SP
02010 POP AF
02020 CP 95 ; A PROGRAM NOT FOUND
02030 JR Z,INT ; error will have a 95
02040 PUSH AF ; here.
02050 LD A,86H
02060 DEC SP
02070 DEC SP
02080 DEC SP
02090 DEC SP
02100 JR QUIT
02120 INT: DEC SP ; make sure the stack
02130 DEC SP ; is ordered properly
02140 DEC SP ; to minimize side effects
02150 DEC SP
02160 DEC SP
02170 DEC SP
02180 INTERCPT: CALL OPENSYN ; open SYNONYM/TXT
02190 JR NZ,QUIT
02200 LOOP1: CALL READLN ; read in a line
02210 JR NZ,QUIT ; test for a match
02220 REL2: CALL COMPARE ; if so, process it
02230 JR Z,PROCESS ; or else loop
02240 JR LOOP1 ; ;build the replacement
02250 PROCESS: CALL BUILDLN ; command line and move
02260 REL3: CALL MOVELN ; command buffer and
02270 POP AF ; it back into the
02280 LD HL,INBUF1$ ; EXECUTE IT.
02290 MODCH8 EQU $-2
02300 IF ECHO ; execute it.
02310 CALL @DSPLY
02320 ELSE
02330 NOP
02340 NOP
02350 NOP
02360 ENDF
02370 JP @CMNDI1
02380 MODCH9 EQU $-2
02400 OPENSYN: CALL LOADFCB
02410 REL4: LD HL,DSKBUF
02420 REL5: LD DE,FCB
02430 LD B,0
02440 CALL @OPEN
02450 RET
02470 LOADFCB: LD HL,FILENAME
02480 REL7: LD DE,FCB
02490 LD BC,CR-FILENAME+1
02500 LDIR
02510 RET
02530 FILENAME: DEFM 'SYNONYM/TXT'
02540 CR: DEFB 13
02560 READLN: LD HL,LINBUF ;Read a line of text into
02570 LOOP2: LD DE,FCB ; a local buffer until
02580 CALL @GET ; a CR, skipping all nulls
02590 RET NZ ; or other control characters.
02600 LD (HL),A
02610 CP 13
02620 RET Z
02630 CP 0AH
02640 JR C, LOOP2
02650 INC HL
02660 RELXX: LD DE,LAST ; Make sure the text does
02670 SBC HL,DE ; not exceed the line
02680 POP HL ; buffer.
02690 JR Z, LOOP2
02700 INC HL
02710 JR LOOP2
02730 COMPARE: LD HL,LINBUF ;Get the match count
02740 LD A,(HL) ; from the first
02750 SUB 0AH ; character in the
02760 LD B,A ; synonym record line.
02770 LOOP3: INC HL
02780 LD A,(HL) ; skip spaces in the
02790 CP 0AH ; syn record
02800 JR Z, LOOP3
02810 LD DE,INBUF1$-1
02820 MODCH10 EQU $-2
02830 LOOP4: INC DE
02840 LD A,(DE) ; skip leading spaces
02850 CP 0AH ; in the original
02860 JR Z, LOOP4 ; command line
02870 LOOP5: LD A,(DE)
02880 XOR (HL) ; compare loop.
02890 AND 0DFH ; ignoring upper/lower
02900 RET NZ ; case distinctions,
02910 INC HL ; compare for the full
02920 INC DE ; match count, rejecting
02930 DJNZ LOOP5 ; for any mismatch.
02940 LOOP5A: LD A, (DE) ; After the count,
02950 CP 13 ; if the original ends
02960 RET Z ; first, match is okay.
02970 CP 20H ; ends first, then the
02980 RET Z
02990 LD A, (HL) ; But if the SYN key
03000 CP 20H ; ends first, then the
03010 JR Z,NOMATCH ; match is rejected.
03020 CP 13
03030 JR Z,NOMATCH
03040 LD A, (DE)
03050 XOR (HL)
03060 AND 0DFH
03070 RET NZ
03080 INC HL
03090 INC DE
03100 JR LOOP5A
03110 NOMATCH: OR A
03120 RET
03130 BUILDLN: PUSH IX ; Save IX on general principles.
03140 PUSH DE ; Save the pointer into INBUF$.
03150 PUSH HL ; Save the pointer into the SYN record
03160 REL8: LD HL, NEWLN ; Start at the beginning of the
03170 POP DE ; new line buffer,
03180 LOOP6: LD A, (DE) ; skip until a space
03190 CP 20H
03200 JR Z, LOOP7
03210 INC DE
03220 JR LOOP6
03230 LOOP7: INC DE ; skip spaces until replacement
03240 LOOP8: LD (HL), A ; copy replacement text
03250 CP 13 ; until EOL
03260 CP 20H
03270 JR Z, LOOP7
03280 LOOP9: LD A, (IX) ; command line, and copy it
03290 CP 20H ; to the new command line.
03300 JR Z, LOOP10
03310 CP '&' ; if '&' is in replacement
03320 JR Z, SUBSTIT ; text then substitute.
03330 RETUR: INC HL
03340 INC DE
03350 INC IX
03360 JR LOOP8
03370 SUBSTIT: POP IX ; Pick up the pointer to the
03380 PUSH IX ; remainder of the original
03390 LOOP9: LD A, (IX) ; command line, and copy it
03400 CP 20H ; to the new command line.
03410 JR Z, LOOP10
03420 JR C, NOSUB ; Skip leading spaces.
03430 INC IX ; If no no-space characters,
03440 JR LOOP9 ; then & vanishes.
03450 LOOP10: INC IX
03460 LOOP11: LD (HL), A
03470 INC HL
03530 INC IX
03540 LD A,(IX)
03550 CP 20H
03560 JR NC,LOOP11
03570 NOSUB: DEC HL
03580 JR RETUR
03590 BLOEX: POP DE
03600 POP IX
03610 RET
03620 MOVELN: LD HL,NEWLN
03630 LD DE,INBUF1$ ; copy up to 63 text
03640 MODCH11 EQU $-2
03650 LD B,63 ; characters and one
03660 LOOP12: LD A,(HL) ; CR to the original
03670 LD (DE),A ; command buffer.
03680 CP 20H
03690 RET C
03700 INC HL
03710 INC DE
03720 DJNZ LOOP12
03730 LD A,13
03740 LD (DE),A
03750 RET
03760 INIT: LD HL,BEGIN ; This is the CONFIG
03770 LD (400DH),HL ; initialization routine.
03780 CMD: DEFB 0C9H
03790 ADDRESS: DEFW 0
03800 FCB: DEFS 50
03810 DSKBUF: DEFS 256
03820 NEWLN: DEFS 64 ; Result line buffer
03830 LINEBF: DEFS 80 ; Syn line buffer
03840 LAST: DEFM 'HJM'
03850 END START

Modifying the Model 3 Real-Time-Clock Interrupts
by Andrew Gransden, c/o 68 St Annes Grove, FARHAM, 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. TRSOS 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'1E0') 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 HIGHS; and then reduces HIGHS 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 HIGHS$. One word of warning: as should be the case with all programs used with LDOS, HIGHS 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.

\begin{verbatim}
0 5 0 65 4 4 94 D 4 54 2 4 90 1 0 20 0 5 22 1 9 65 2 C D6 7 4 43 A 2 50 1 F E4 9 C 21 8 5 32 A 1 34 0 3 EA 5 B DC 2 1 D 5 3 3 E4 4 B CC 2 1 D5 3 2 A7 9 4 43 E 4 FB C D A2 2 5 32 A 1 14 4 D D2 1 C 95 3 D D7 5 0 1D D 7 40 2 D D2 1 C D5 3 D D7 5 0 1D D 7 40 2 3 E1 5 7 72 B D D2 1 D E5 3 D D7 5 0 1D D 7 40 2 D D2 1 E 25 3 D D7 5 0 1D D 7 40 2 D D2 1 E B5 3 D D7 5 0 1D D 7 40 2 3 E2 4 7 72 A 7 94 4 2 20 A 5 42 A 1 14 4 2 2B 9 5 30 1 5 70 0 A F E D 4 22 2 1 14 4 2 3F 3 2 27 9 4 4E B 2 1B 7 5 3E D B 0F B 2 1F 7 5 2C D 8 A4 2 C 32 D 4 05 4 4 94 D 4 53 5 3 02 0 2 D 2 0 4 C4 4 4 F5 3 2 05 2 6 56 1 6 C2 0 5 46 9 6 D6 5 2 04 3 6 C6 F 6 36 B 2 03 5 3 04 8 7 A2 0 4 36 F 7 27 2 6 56 3 7 46 9 6 F6 E 2 05 5 7 46 9 6 C6 C 6 56 4 2 06 1 6 E6 4 2 06 F 7 06 5 7 26 1 7 46 9 6 F6 E 6 16 C 7 26 5 6 16 4 7 92 0 6 96 E 7 37 4 6 16 C 6 C6 5 6 40 D 5 44 9 4 D4 5 3 53 0 2 04 1 6 26 F 7 27 4 6 56 4 2 10 D 1 80 9 0 15 3 B B5 3 0 65 4 4 94 D 4 53 5 3 03 A 1 64 2 F E1 6 2 03 5 3 A 0 C 5 43 D 3 20 C 5 4F E 0 02 0 0 A3 E 1 53 2 0 C5 4 3 E1 E 3 21 6 4 23 A 0 D5 4 3 D3 2 0 D5 4 F E0 0 2 01 5 3 E2 4 3 20 D 5 43 E 0 15 F 3 A1 6 4 29 3 F E0 6 3 00 2 3 E0 6 3 21 6 4 23 A 1 64 2 E E0 6 2 00 4 3 C3 2 1 64 2 C 30 0 0 00 2 0 2 0052
\end{verbatim}

00100 ;* TIME50/ASM - Version 4.1 - 28 Oct 83
00110 ;* TITLE: TIME50 50Hz Real Time Clock Correction Routine
TIME50 routine corrects for errors in the TRS-80 Model III Real Time Clock
when operating on a 50Hz Version Machines under the LDOS Disk Operating System Version 5.1.x.

Variable and Label Declarations

LF EQU 0AH ; linefeed
CR EQU 0DH ; carriage return
SEC1 EQU 21 ; coarse correction count
COR1 EQU 8 ; coarse correction value
SEC2 EQU 36 ; fine correction count
COR2 EQU 1 ; fine correction value
ROMCHK EQU 0125H ; ROM check for Model III
INTVEC EQU 4012H ; interrupt vector
EXIT EQU 402DH ; LDOS return entry
ABORT EQU 4030H ; abnormal program exit to LDOS
HEAT$ EQU 4216H ; clock heartbeat counter
HIGH$ EQU 4411H ; highest useable memory
RTCVEC EQU 4479H ; jump vector to RTC routine

* start of TIME50 installing routine *

ENTRY LD HL,MSG1 ; point to message 1
CALL @DSPLY ; and display it
LD A,(ROMCHK) ; test for Model III
CP 49H
JP NZ,ERROR ; go if not
LD HL,(INTVEC+1) ; load interrupt vector
LD A,0A5H ; load A with known jump
CP L ; and compare with HL
JP C,ERROR1 ; exiting if incorrect
LD HL,(RTCVEC) ; load RTC vector
LD A,44H ; to error handing abort
CP H
JP NZ,ERROR1
LD HL,(RTCVEC) ; load RTC vector
LD A,4FH ; load max DOS area addr
CP H
JP C,ERROR2 ; go if greater
LD HL,(HIGH$) ; get current high mem bc
LD IX,P1 ; set pointer to allow
LD (IX+1),L ; correct addressing of
LD (IX+2),H ; storage in high memory
LD IX,P2
LD (IX+1),L
LD (IX+2),H
LD IX,P3
LD (IX+1),L
LD (IX+2),H
LD A,SEC1 ; set count to ? seconds
LD (HL),A
DEC HL
LD IX,P4 ; set pointer as above
LD (IX+1),L
LD (IX+2),H
00800 LD IX,P6
00810 LD (IX+1),L
00820 LD (IX+2),H
00830 LD A,SEC2
00840 LD (HL),A
00850 LD HL,(RTCVEC) ;get present mt vector
00860 LD (EXIT+1),HL ;and save
00870 LD HL,(HIGH$) ;reduce HIGH$ by
00880 LD (NXTMEM),HL ;store old high memory
00890 LD BC,LAST-START ;length of routine
00900 XOR A ;clear carry flag
00910 SBC HL,BC ;calculate new HIGH$
00920 LD (HIGH$),HL ;protect routine
00930 INC HL ;point to new start
00940 DI ;disable interrupts
00950 LD (RTCVEC),HL ;break into Int chain
00960 EX DE,HL ;transfer new START to DE
00970 LD HL,START ;load address of routine
00980 LDIR ;move routine to high ram
00990 EI ;enable interrupts
01000 LD HL,MSG2 ;point to message
01010 CALL @LOGOT ;display & log
01020 JP @EXIT ;return to LDOS
01040 ;* Messages *
01060 MSG1 DB 'TIME50 - LDOS Real Time Clock 50Hz'
01070 DB 'Correction Utility - Ver.4.1',LF
01080 DB 'Copyright (C) 1983 A W Gransden',CR
01090 MSG2 DB 'TIME50 installed and operational',CR
01110 ;* error handling *
01130 ERROR LD HL,ERMSG ;point to message
01140 JR EREXIT ;jump to error exit
01150 ERROR1 LD HL,ERMSG1 ;point to error message
01160 JR EREXIT ;jump to error exit
01170 ERROR2 LD HL,ERMSG2 ;point to error message
01180 EREXIT CALL @LOGOT ;display message
01190 LD HL,ERMSG3 ;load abort message
01200 CALL @LOGOT ;display & log
01210 JP @ABORT ;jump to abort routine
01230 ;* error messages *
01250 ERMSG DB 'For TRS-80 Model III use ONLY!'
01260 DB CR
01270 ERMSG1 DB 'Correction written to work under '
01280 DB 'LDOS Version 5.1.x ONLY!'
01290 DB CR
01300 ERMSG2 DB 'Correction already installed'
01310 DB CR
01320 ERMSG3 DB 'TIME50 Aborted!',CR
01340 ;* actual TIME50 routine to be placed in high ram *
01360 START EQU $ 
01370 JR J1 ;skip protocol block
01380 NXTMEM DS 2 ;high mem addr of next bk
01390 DB 06H ;6 bytes of protocol blk
01400 DB 'TIME50' ;routine title
01410 J1 EQU $ 
01420 LD A,(HBEAT$) ;get heartbeat count
01430 CP 1EH-COR1 ;just reset?
01440 JR NZ,TEST2 ;go if not
01450 P1 LD A,(COUNT1$) ;get seconds count
01460 DEC A ;decrease by one
01470 P2 LD (COUNT1$),A ;save
01480 CP 00H ;... seconds gone
01490 JR NZ,P4 ;go if not
01500 LD A,SEC1 ;reset seconds count
01510 P3 LD (COUNT1$),A ;save reset count
01520 LD A, 1EH ;increase heartbeat
01530 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:

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.

Page 28
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.COM 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 your 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 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:

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

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

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

```c
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:

```c
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:

```c
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:

```c
 /* ....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:

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

Notice that it is so much easier to use:

```c
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:

```c
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:

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

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:

```c
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:

```c
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:

```c
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:

```c
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?

```c
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 TRSOS 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'=DS 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 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.

---

**Sending Characters to a Printer Via the Keyboard with a Single Keystroke (Whew!)**

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 v1n5p14 v1n6p64
"At Large" - Earl Terwilliger v1n6p50
"Automatic Chaining with JCL" - Jim Kyle v2n4p52
"BASIC and File Structure - A Beginner's View" - Wes Goodnough v1n6p20
"BASIC Concepts - The RUN,V Command" - Dick Konop v1n6p69
"Beta Tester, I was an LDOS" - Tim Daneliuk v1n3p16
Byte I/O under LDOS 6.x v1n3p38 v1n6p76 v2n2p52
'C' graphics v2n4p16
"'C' language, The" - Earl Terwilliger: general introduction v2n1p15 functions, variables, constants, expressions v2n2p35 operators v2n3p39 logic, control & flow v2n4p34 initialization, 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
"Confessions of a Machine Language Addict" - Ray Pelzer
Configuring with non-relocatable code
Customer service tips
Cylinder term explained
DAM's, old
"Data Address Marks" - Roy Soltoff
Date conversions
"Device I/O and Independence, LDOS" - Roy Soltoff
Disk allocation schemes
Disk byte I/O
Disk controller, AM Electronics
Disk drive first access delay
"Disk Drive Control Linkages, LDOS" - Bob Bowker
Disk drives (8") on the Model III
Disk drives, non-Radio Shack
Disk drive poll
Disk drive select time hardware fix (Mod I)
"Disk I/O in Assembler" - Doug Kennedy
Disk speed (300 RPM delays)
"Double Sided Drives with LDOS" - Tim Mann & Roy Soltoff
"Easy LScript" - James Bruckart
"Easy VisiCalc" - James Bruckart
"EDAS IV 'Z' Command" - Earl Terwilliger
Editor, LDOS command line
Electric Webster with Newscript, LDOS and Sole
"Electronic Inbasket, The" - Gordon Thompson
"ELSIE - The Contented Compiler" - Jim Frimmel
Epson MX-80 tips
assembly language efficiency
printers, TBA, UTILZAP
printers, word processors, MNet, 'C', LDOS 6.x
'C' book, SS drivers, Model 4, PROMPT/CMD
UNIX, IBM PC, Telex
Expansion interface
EZEDIT - The LDOS command line editor - Graham Brown
"Fast Graphics for 'LC'" - Scott Loomer
File listing utility for TBA
File structure, BASIC
Filter linkage
"Fortran, Cobol and LDOS JCL" - Glen Rathke
Fortran with interrupts and SVC's
"Greek to Me, LDOS - It's" - Charles Knight
"Hayes Smartmodem, LDOS and a" - John Mullin
High memory, avoiding memory conflicts
High memory module header
Host, communications
Index, LDOS Quarterly/LSI Journal - Scott Loomer
"Inside the Expansion Interface" - Earle Robinson
"Interrupts and SVC's in Fortran, Using" - J. Bender
JCL chaining
"JCL Corner" - Chuck Jensen:
clear screen
file copying
compilation macros
hex codes, more compilation macros
keyboard input
DAM's, old
logical operators, keyboard filters, LScript patch
creating and defining JCL procedures
backups
JCL with FORTRAN and Cobol
Cumulative 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
"*** PARITY = ODD ***" - Tim Daneliuk:
  introduction, DATAENTR 200 & ISAM 200 v1n4p17
  tips for better programming, DISCATER, Filter Disk v1n5p11
  BASP drives, drive poll, TAS, MODEM80, HEXSPELL II, Utilities, HELP v1n6p29
  Tandy, MAX-80, AEROCOMP, Electric Webster, The BASIC Answer v2n1p42
  disk drive poll, gold plugs, LX-80, Proofreader, Scripsit Dict. v2n2p27
  magazines v2n3p35
  rumors, LX-80 software compatibility, TRSDOS 6.x software v2n4p31
  CP/M, LeScript, M4 vs. MAX-80 v2n5p28
  "Partitioned Data Sets, MISOSYS Announces" - Roy Soltoff v1n3p23
  "Pascal 80, LDOS and" - D. Hill v2n1p22
  Passwords for LDOS 5.1.x v1n1p03
  Patch, how to v2n1p52
  "PDS - Standard and Other Types of Uses" - Scott Loomer v2n1p24 v2n3p45
  "Performing Date Conversions in BASIC" - Dick Konop v2n4p60
  PR/FLT, changing parameters v2n5p13
  Printer codes, sending v2n5p38
  Printers v2n1p39
  Profile III Plus with LDOS v2n4p55 v2n5p26
  Profile One Plus (Pro3+ on the Mod 1) - E. Sturiale v2n5p26
  PRPARM - changing PR/FLT parameters - Graham Brown v2n5p13
 Quarterly reader survey results v2n2p06
  Radio Shack v2n1p42
  Recursive v2n5p42
  "Relocating Code for LBASIC USR Routines" - Chuck Jensen v1n3p47

Reviews:
  Alternate Source (TAS) - Tim Daneliuk v1n6p60
  APL*PLUS/80 (STSC) - Daniel Lofy & Lee Rice v2n1p09
  CHROMAtrs (South Shore Computer Concepts) - Scott Loomer v2n3p28
  DATAENTR200 & ISAM200 (Johnson Associates) - Tim Daneliuk v1n4p18
  EDAS IV (MISOSYS) - Marc Leager v1n6p09
  Electric Webster (Cornucopia) - Tim Daneliuk v2n1p45
  HELP (MISOSYS) - Tim Daneliuk v1n6p63
  HEXSPELL II (Hexagon) - Tim Daneliuk v1n6p61
  LDOS Utilities (Powersoft) - Tim Daneliuk v1n6p62
  LISP (Supersoft) - Lee Rice & Daniel Lofy v1n6p29
  MAX-80 (Lobo) - Tim Daneliuk v2n1p44
  Microcomputer Math book (SAMS) - Earle Robinson v1n5p10
  MODEM80 (LSI) - Tim Daneliuk v1n6p60
  Pascal (Alcor) - Scott Loomer v2n1p18
  Pascal-80 (New Classics) - D. Hill v2n1p22
  PDS (MISOSYS) - Scott Loomer v2n1p24
  Printers - Earle Robinson v2n1p39
  Profile III+ (Radio Shack) - Sam Goldberg v2n3p25
  Proofreader (Aspen Software) - Tim Daneliuk v2n2p30
  Scripsit Dictionary (Radio Shack) - Tim Daneliuk v2n2p30
  Structured BASIC Translator (Acorn) - Sue Ratkowski v1n4p21
  The BASIC Answer (LSI) - E. Cheatham v2n3p17
  The BASIC Answer (LSI) - Tim Daneliuk v2n1p46
  UOLISP (Far West) - Lee Rice & Daniel Lofy v1n6p30
  Routing a device v1n3p38

"Roy's Technical Corner" - Roy Soltoff:
  load module structure v1n4p42
  task processor v1n5p20
  error handling during byte I/O, @CKDRV, KFLAG$, @ICNFG, @KITSK v1n6p76
  LDOS 6.x v1n6p76
  LDOS 6.x v2n2p46
  RS-232 drivers v1n6p83
  RTC Interrupt, Model 3 v2n5p22

"Running 8-Inch Drives on the Model III Under LDOS" - Peter Simon v2n3p08

Sector I/O v2n5p55

Sending Printer codes v2n5p38
Cumulative Index to LDOS Quarterly Volumes 1 and 2 - Subject

"SOLEFIX - Fix that GAT Error" - Erik Ruf v2n2p14 v2n4p38
Speedup kits v1n1p13
SYNONYM - The LDOS command synonym processor - Henry Melton v2n5p14
Sysgen and type-ahead v1n1p04
Sysres, Quickly with JCL v2n4p12
System routines:
@ADTSK, @RMTSK, @KITSK, @RPTSK - task proc. v1n5p31 v1n6p76 v2n4p19
@CKDRV - determining 5.1.2 vs 5.1.3 for @CKDRV adjustment v1n6p78
@CKDRV - moved v1n5p06
@CMNDI - command interpreter v1n1p12
@CTL, @GET, @PUT - byte I/O v1n3p38
DAYS - day of the week v1n4p11
@GET and @PUT under LDOS 6.x v2n3p56
@ICNFG - configuration interfacing v1n6p81
@KITSK - keyboard task v1n6p82
KFLAGS$ - keyboard scanner v1n6p78
"@PARAM, Using" - Roy Soltoff v1n2p31
@PARAM under 6.x v2n4p61
"@PARAM, @DSPLY, @EXIT and INBUF$ for Everyone" - David Vinzant v2n2p23
@RAMDIR documentation correction v2n2p39
SVC's and Fortran v2n4p19
Tandy v2n1p42 v2n3p03
Task processor v1n5p20 v1n6p76
"T-Timer, LDOS Supports the" - Roy Soltoff v1n4p12 v1n5p18 v1n6p66
TRSDOS (Mod I) to LDOS (Mod III) transfer without REPAIR v1n4p58 v2n2p44
TRSDOS 6.x (see also LDOS 6x.) v2n2p46
Update policy explained - Bill Schroeder v1n2p02 v1n4p04
Upper case lock v1n4p11
Users group directory v2n1p06
Utilities:
CONV v2n2p44
COPY23B v2n2p44
PATCH v2n1p52
REPAIR v2n2p44
Version number explanation v1n2p02
VisiCalc made easy v2n3p23
"VisiCalc with LDOS, Using" - Roy Soltoff v1n2p20

Cumulative Index to LDOS Quarterly Volumes 1 and 2 - Programs

ADDLF/FLT - James Bruckart - add a line feed after carriage return v2n1p36
ALIVE/ASM - Earl Terwilliger - task processing demo v1n6p53
BEEP/FLT - Tim Mann - beep cassette port on each keystroke v1n1p22
BINHEX/BAS - Tim Mann - convert hex to binary v1n4p56
Improved version with checksum v2n2p42
CARD/BAS - Paul Tonini - file utility v2n2p18
CASE/CMD - Rick Tobias - Mod I case mode indicator v1n2p11
CIRCLE/CCC - Scott Loomer - fast graphic circle routine v2n4p16
CLFL/FLT - Les Mikesell - general purpose communications filter v1n6p87
CLS/CMD - clear screen command for use in JCL (very short!!) v1n6p67
COM/BAS - Earl Terwilliger - short BASIC dumb terminal program v1n6p50
COMM1/ASM - Earl Terwilliger - machine language dumb terminal program v1n6p51
CONFIG8/CMD - Peter Simon - 8" drives on the Model III v2n3p08
CTLP/FLT - Les Mikesell - link printer and display v2n3p57
DATE/BAS - Dick Konop - date conversions v2n4p60
DCAL/BAS - Tim Mann - disk drive speed check on Mod I v1n1p26
DELETE/CMD - Renato Reyes - multiple file killer v1n4p56
DIRMAP/BAS - Dave McLaughlin - granule map of disk allocation v1n2p41
DISKMAP/BAS - Bill Fields - alphabetized disk file mapper v1n4p50
EZEDIT/FLT - Graham Brown - LDOS command line editor v2n5p07
### Cumulative Index to LDOS Quarterly Volumes 1 and 2 - Programs

<table>
<thead>
<tr>
<th>Program</th>
<th>Description</th>
<th>Volume and Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAST/CMD</td>
<td>Tim Mann - fast clock on the LX-80</td>
<td>v1n1p24</td>
</tr>
<tr>
<td>FASTSPL/ASM</td>
<td>fast despooling for Model I 5.0.3</td>
<td>v1n6p68</td>
</tr>
<tr>
<td>FILL/CCC</td>
<td>Scott Loomer - non-recursive graphic fill routine</td>
<td>v2n4p16</td>
</tr>
<tr>
<td>INBASKET/BAS</td>
<td>Gordon Thompson - communications message collector</td>
<td>v2n4p15</td>
</tr>
<tr>
<td>IOTAP/FLT</td>
<td>Charles Knight - modification to SLASH0/FLT</td>
<td>v1n5p16</td>
</tr>
<tr>
<td>KILL/CCC</td>
<td>Earl Terwilliger - multiple file killer</td>
<td>v2n2p39</td>
</tr>
<tr>
<td>LBASIC/0V4</td>
<td>Alan Moyer - active variable dump for Model I</td>
<td>v2n3p15</td>
</tr>
<tr>
<td>LISTITBA</td>
<td>Jeffrey Brenton - file lister for TBA</td>
<td>v2n3p19</td>
</tr>
<tr>
<td>MINIDOS/FLT</td>
<td>Roy Soltoff - MiniDOS for 5.0.x Mod I users</td>
<td>v1n2p44</td>
</tr>
<tr>
<td>MODCMD/CMD</td>
<td>John Mullin - send commands to a Hayes modem</td>
<td>v1n6p34</td>
</tr>
<tr>
<td>MX80/FLT</td>
<td>Ken Roser - block graphics on MX-80 w/o Graftrax</td>
<td>v1n1p22</td>
</tr>
<tr>
<td>MX80/FLT</td>
<td>improved version of above</td>
<td>v1n4p59</td>
</tr>
<tr>
<td>NODAM/CMD</td>
<td>Chuck Jensen - read non-LDOS disks without REPAIR</td>
<td>v1n4p58</td>
</tr>
<tr>
<td>ONLINE/FLT</td>
<td>John Mullin - carrier detection filter</td>
<td>v1n6p37</td>
</tr>
<tr>
<td>PASSWORD/BAS</td>
<td>James Bruckart - password check for host systems</td>
<td>v1n1p36</td>
</tr>
<tr>
<td>PENCTL/FLT</td>
<td>Tim Mann - for Electric Pencil users</td>
<td>v1n1p22</td>
</tr>
<tr>
<td>PDUML/CMD</td>
<td>original Percom doubler driver (for illustration only)</td>
<td>v1n1p24</td>
</tr>
<tr>
<td>PROMPT/CMD</td>
<td>Earle Robinson - change the 'LDOS Ready' prompt</td>
<td>v2n3p35</td>
</tr>
<tr>
<td>PRPARM/CMD</td>
<td>Graham Brown - changing PR/FLT parameters</td>
<td>v2n5p13</td>
</tr>
<tr>
<td>REFLEX/FLT</td>
<td>Gordon Thompson - communications program</td>
<td>v1n6p16</td>
</tr>
<tr>
<td>RENAME/BAS</td>
<td>Charles Knight - BASIC file renamer</td>
<td>v1n5p42</td>
</tr>
<tr>
<td>SALVAGE/BAS</td>
<td>transfer data from TRSDOS 2.3B disks</td>
<td>v1n2p15</td>
</tr>
<tr>
<td>SLASH0/FLT</td>
<td>print slashed zeroes</td>
<td>v1n2p41</td>
</tr>
<tr>
<td>SLOGSTEP/DCT</td>
<td>Les Mikesell - head settling delay for some DS drives</td>
<td>v2n1p60</td>
</tr>
<tr>
<td>SOLBFIX/ASM</td>
<td>Erik Ruf - fix SOLEd disk for compatibility</td>
<td>v2n2p14</td>
</tr>
<tr>
<td>SYNONYM/CMD</td>
<td>Henry Melton - command synonym processor</td>
<td>v2n5p14</td>
</tr>
<tr>
<td>TCHRON/ASM</td>
<td>Roy Soltoff - time board support routine</td>
<td>v1n6p66</td>
</tr>
<tr>
<td>TERMINAL/BAS</td>
<td>dumb terminal program</td>
<td>v1n6p86</td>
</tr>
<tr>
<td>TIME05/CMD</td>
<td>Andrew Grandsen - modify M3 RTC processing (50Hz)</td>
<td>v2n5p22</td>
</tr>
<tr>
<td>TIMEDI/FOR</td>
<td>J. Bender - Fortran interrupt and SVC demo</td>
<td>v2n4p22</td>
</tr>
<tr>
<td>UPDATE/BAS</td>
<td>Tim Mann - update system disks (replaced by BACKUP)</td>
<td>v1n1p23</td>
</tr>
<tr>
<td>USRFREE/CMD</td>
<td>Chuck Jensen - LBASIC free memory command</td>
<td>v1n1p24</td>
</tr>
</tbody>
</table>

### Cumulative Index to LDOS Quarterly Volumes 1 and 2 - Patches

<table>
<thead>
<tr>
<th>Patch</th>
<th>Description</th>
<th>Volume and Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASIC compiler (RS) patch</td>
<td>v1n4p62</td>
<td></td>
</tr>
<tr>
<td>COBOL patch</td>
<td>v1n4p62</td>
<td></td>
</tr>
<tr>
<td>DEBUG ASCII mode patch</td>
<td>v1n5p45</td>
<td></td>
</tr>
<tr>
<td>DIRCHECK/CMD (LSI Utility Disk #1) patch for hard or DS drives</td>
<td>v2n2p44</td>
<td></td>
</tr>
<tr>
<td>DTERM patch - Peter Trenholme</td>
<td>v1n5p18</td>
<td></td>
</tr>
<tr>
<td>EDAS IV &quot;Z&quot; command patch - Earl Terwilliger</td>
<td>v1n1p37</td>
<td></td>
</tr>
<tr>
<td>FED patch</td>
<td>v1n1p47</td>
<td></td>
</tr>
<tr>
<td>File Manager patch for moving HIT entries</td>
<td>v2n5p36</td>
<td></td>
</tr>
<tr>
<td>File Manager patch for moving system files</td>
<td>v2n3p38</td>
<td></td>
</tr>
<tr>
<td>HIGH patch for screen full pause</td>
<td>v2n4p39</td>
<td></td>
</tr>
<tr>
<td>I/O bus disable for boot on Model III</td>
<td>v2n3p46</td>
<td></td>
</tr>
<tr>
<td>KT/DVR patch to extend keyboard debounce time</td>
<td>v2n3p45</td>
<td></td>
</tr>
<tr>
<td>LBASIC default extension patch</td>
<td>v1n5p19</td>
<td></td>
</tr>
<tr>
<td>LBASIC remove execution banner</td>
<td>v2n5p15</td>
<td></td>
</tr>
<tr>
<td>LBASIC RUN**,V fix</td>
<td>v2n5p35</td>
<td></td>
</tr>
<tr>
<td>LED xlate parameter patch</td>
<td>v2n1p48</td>
<td></td>
</tr>
<tr>
<td>Lower case driver forced on boot (Mod I)</td>
<td>v2n3p46</td>
<td></td>
</tr>
<tr>
<td>LScript patch to require filespecs for S or L</td>
<td>v1n6p45</td>
<td></td>
</tr>
<tr>
<td>LScript JCL patch</td>
<td>v1n6p13</td>
<td></td>
</tr>
<tr>
<td>LScriptX patches - Scott Loomer</td>
<td>v2n1p47</td>
<td></td>
</tr>
<tr>
<td>MAP (LSI Utility Disk #1)</td>
<td>v1n1p47</td>
<td></td>
</tr>
<tr>
<td>MAX-80 patch to prevent printer driver from converting LF's to CR's</td>
<td>v2n2p39</td>
<td></td>
</tr>
<tr>
<td>MAX-80 patch to set LF after CR</td>
<td>v2n4p39</td>
<td></td>
</tr>
<tr>
<td>Microsoft patches</td>
<td>v1n2p16</td>
<td></td>
</tr>
</tbody>
</table>
**Cumulative Index to LDOS Quarterly Volumes 1 and 2 - Patches**

- Microsoft MACRO-80 patches - Richard Deglin
- MiniLDOS patch to allow installation before KSM
- Mod 3 speedup patch
- Newscript patch to use with REFLEX - Gordon Thompson
- Password (master) disable patch
- PDUBL patch to write old DAM
- PR/FLT patch to disable chr$(6) trapping
- Profile patches - Dick Yevich
- Profile patch to fix buffer problem
- Profile 3 Plus, HD version - for use on Mod 1 - E. Sturiale
- PDUBL patch to write old DAM
- QFB - Quick Format and Backup fix for READ40 and DS drives
- SOLE patch to allow use of Super Utility - Erik Ruf
- SuperScripsit patch - Model 1 version 01.02.00
- SuperScripsit patches - Dennis Brent & Renato Reyes
- SuperScripsit patches - Joseph Kyle-Dipietropaolo
- SYSTEM(SLOW) and (FAST) patch
- TCHRON, TIMEDATE80 & TRSWATCH time & date boards patch
- The BASIC Answer patch for local variable DC
- T-Timer patches
- TRSDOS patch to read REPAIRed LDOS disks on Mod I
- VIDSAV/CMD (LSI Filter Disk #2) patch to allow JCL installation
- VisiCalc patches
- Mod III enhanced - Ray Pelzer
- RS332 transfer
- WRTEST (LSI Utility Disk #1)
- XMODEM - from LS-Host/Term - patch to fix 8-bit init

**Cumulative Index to LDOS Quarterly Volumes 1 and 2 - Authors**

- Bender, J. - "Using Interrupts and SVC's in Fortran"
- Bowker Bob - "LDOS Disk Drive Control Linkages"
- Brent, Dennis - SuperScripsit Patches
- Brenton, Jeffrey - "An ASCII File Listing Utility for TBA"
- Brown, Graham - "EZEDIT - The LDOS command line editor"
- "PRPARM - Changing PR/FLT parameters"
- Bruckart, James - "Communication Host" (PASSWORD/BAS & ADDLF/FLT)
- "Easy LScript"
- "Easy VisiCalc"
- Butler, Charlie - "An Article" (changing operating systems)
- Cheatham, E. - "My BASIC Answer"
- Danelluk, Tim - "I was an LDOS Beta Tester"
- **** PARITY = ODD ****: DATAENTR200 & ISAM 200
- **** PARITY = ODD ****: Message to Hackers
- **** PARITY = ODD ****: TASI, HEXSPELL II, MODEM80, HELP, Utilities
- **** PARITY = ODD ****: Tandy, MAX-80, Electric Webster, TBA
- **** PARITY = ODD ****: disk drive poll & spelling checkers
- **** PARITY = ODD ****: magazines
- **** PARITY = ODD ****: rumors, LX-80, TRSDOS 6.x software
- **** PARITY = ODD ****: CP/M, LeScript, M4 vs. MAX-80
- Deglin, Richard - Microsoft MACRO-80 patches
- Fields, Bill - DISKMAP/BAS
- Frimmel, Jim - "ELSIE, The Contented Compiler"
- Goldberg, Sam - "Review of Profile III+ for LDOS"
- Goodnough, Wes - "BASIC and File Structure - A Beginner's View"
- Hall, D. - "LDOS and Pascal-80" (New Classics review)
- Knight, Charles - "LDOS, It's Greek to Me" (Greek translation)
- "RENAME /BAS"
- Kyle, Jim - "Automatic Chaining with JCL"
- Latham, Jerry - "Mixing Newscript, Electric Webster, LDOS and Sole"
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'76BB'=02 7D
X'7D02'=CD 50 7C 21 2D 71 CD E5 7C 21 1C 7D 7B 02 23 CD 33
X'7D13'=00 7E FE 03 28 61 C3 0E 7D 0D 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 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 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 6E 20 43 6F 6E 77 69 73 74 75 63 72 69 73 74
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 2B 2E 2E 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 X'7BC2'=67 44
PATCH EFCF/CMD USING EFCF/FIX X'74B8'=18 43
PATCH EFC7/CMD USING EFC7/FIX X'7652'=18 43
PATCH EFC8/CMD USING EFC8/FIX X'74BB'=19 43
PATCH EFC9/CMD USING EFC9/FIX X'7661'=05 44
PATCH EFCA/CMD USING EFCA/FIX X'7832'=05 44
PATCH EFCD/CMD USING EFCD/FIX X'7617'=01 30
PATCH EFCC/CMD USING EFCC/FIX X'782C'=01 30
PATCH EFCE/CMD USING EFCE/FIX X'7ABE'=01 30
PATCH EFCM/CMD USING CPROMPT/FIX X'7151'="ONE"
PATCH RM/CMD USING RPROMPT/FIX X'74F6'=63
Networking FOR RADIO SHACK MODELS III & 4...NOW!

TYPICAL CONFIGURATION:

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 MULTILINK, 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 properly via 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

PSM-W5: Model 4 w/64k RAM, 408k DS Floppy, built-in 5 Mbyte hard disk, master network controller, LDOS 5.1.4, Technical Manual.

PSM-T/0: Model 4 w/64k RAM, boot PROM, satellite network controller, Tech Manual.

PSM-60: Upgrade your Model III/4 w/40k RAM to PSM-W5: incl'd 5mb hard disk, hard disk power supplies, Tech Manual.

PSM-NM: Technical Manual: Incl'd complete description of PSM networking, and application implementation. Tech Manual purchase may be credited toward later hardware purchase. 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

LDOS is a trademark of LOGICAL SYSTEMS, INC.