MISOSYS

CATALOG OF SERIOUS SOFTWARE

CAT: 82-1

EDAS - Version IV

EDAS is a powerful disk-based combined editor and assembler supported under Model I and Model III TRS-80s running under LDOS. A Model II TRSDOS version will be available in fall 1982. Among its features are direct assembly from one or more source disk files or memory buffer, conditional assembly, macro assembly, extensive cross reference listings, and a comprehensive line editor that supports upper and lower case text entry.

EDAS ASSEMBLER FEATURES

EDAS assembles absolute core-image object code to disk as a directly executable load module (CMD). Source code can exist in memory as well as included disk files when using the *GET assembler directive. *GET files can be nested to five levels. EDAS uses default file extensions of "ASM" for source and "CMD" for object code files to guard aginst inadvertant over-write of a source file with object code. EDAS also respects HIGH$.

A powerful "*SEARCH filespec" assembler directive will invoke automatic search of the Partitioned Data Set (PDS) "filespec" containing a library of source code. The PDS directory will automatically GET any PDS member that would resolve an undefined label reference. This process can be correlated to a relocating assembler's resolving references at link time. In EDAS, the source library is ISAM accessed for minimal I/O overhead. PDS is required to construct your own subroutine libraries.

Conditional assembly is supported with five pseudo-ops; IF expression; IFLT expression1, expression2; IFEQ expression1, expression2; IFGT expression1, expression2; IFDEF label; IFNDEF label; and IFREF label. Conditional assembly also supports the "IFx ELSE ENDIF" construct. Conditional expressions can be nested to 16 levels.

The expression evaluator supports left-to-right evaluation of the following operators: "+" addition; "-" subtraction; "*" 16-bit by 8-bit integer multiplication; "/" 16-bit by 8-bit integer division; ".MOD" modulo division; "<" shift; "&" or ".AND." logical AND; "!" or ".OR." logical OR; ".XOR." logical exclusive OR; ".NOT." unary one's complement; ".NE." logical not equal; and ".EQ." logical equal.

Pseudo-ops DEFB and DEFM are synonymys. EDAS also accepts DS, DW, DB, and DM as well as DEFS, DEFW, DEFB, and DEFM. EDAS provides for binary, octal, decimal, hexadecimal, and string constants. Constant declarations can be concatenated on one line, by separating terms with commas. This permits complex expressions such as:

```
DB 1,2,'Buckle your sho','e'.OR.80H,'I can''t'
```

Labels may be up to 15 characters long. Labels must start with A-Z, "@", or "$". Positions 2-15 may also use "?" and "_". The "*MOD" assembler directive is available to provide a unique character string substitution for the "?" character appearing in labels of all files accessed via *GET. The string value will increment each time *MOD is commanded. This will provide "local label" support for routines read off of disk.

A logical origin pseudo-op, LORG, will assemble load module files with the load addresses offset to the LORG address while execution addresses are based on the ORG address. When using EDAS to assemble applications that block move sections of code, the LORG can be used to assemble the entire job at once.

The EDAS assembler provides many switch options. These invoke: "-IM" assemble output to memory; "-LP" list to printer; "-NC" suppress false conditional blocks from listings; "-NE" suppress constant expansions on listing; "-NM" suppress listing of macro expansions; "-WE" wait on error; "-WO" assemble with object code; "-WS" generate a sorted symbol table listing; "-XR" generate a cross reference data file for downstream processing by XREF.

Single level MACROs are supported with both positional parameters and parameters by keyword. Values can be applied to any parameter at MACRO definition time to allow for expansion time defaults if a parameter is omitted at the time a MACRO is referenced. MACROs can be defined in memory or source files but must be defined prior to being referenced. Local labels are supported with the provision of a string substitution for the "?" character in labels. The string will provide a unique value for each MACRO expansion. The MACRO "?" substitution takes precedence over any *MOD substitution.

Additional pseudo-ops are provided for enhanced operation: "COM" will allow a comment line to be written to the load module. These comment records will not be loaded when executing the module, but will merely provide an easy way to store such things as copyright messages in your object deck files; "TITLE" will paginate your listing with a title string including the current date and time, and an incrementing page number; "SUBTTL" lists the sub-title string after each title; "PAGE" ejects a listing to a new page; "SPACE" generates additional line feeds during listings for highlighting modules.

A sorted symbol table listing is available during the assembly. A complete CROSS REFERENCE listing is available by a downstream processing utility, XREF. Once an XR data file is generated, XREF will produce a listing identifying all defined labels, the line number containing the definition, its value, and the file name of the source file containing the definition ($CORE is used to designate labels defined in memory). For each defined label, all references to the label are listed by line number and source file containing the reference. XREF lists statistics on the quantity of defined labels and references. XREF can also be used to generate a file containing EQUates (or DEFLs) for all symbols or a subset of symbols (those including a special character). The EQU file is useful for interfacing separately executable modules to a resident module (such as in overlay applications).

EDAS EDITOR FEATURES

The EDAS editor operates on text in memory and uses a command syntax identical to BASIC for intra-line editing. Lines hacked to null length will be automatically deleted.

EDAS will "Load" and "Write" text buffers from/to disk with text file concatenation in memory. The standard source file will be un-headered and un-numbered which saves approximately 20% of disk file storage requirements. However, EDAS will AUTOMATICALLY recognize and properly read a file that is headered and/or numbered whether through "Load" or "*GET" input. Two switches are provided in the "Write" command to generate a header or line numbers when saving a text buffer to disk.

You can input text in upper or lower case. In the case-converted mode, all assembler source input is properly converted to upper case, AUTOMATICALLY. In the case consistant mode, text remains as it was input. Thus, the editor can be used for assembler source, or source for other languages such as PASCAL and C.

The editor supports relocating a block of lines with the "<M>ove start,end,to" command. Global changes to character strings can be made throughout the text buffer or to only a designated range of lines with the "<C>hange /string1/string2/start,end" command. Want to copy a block of lines? The "<C>opy start,end,to" command will duplicate the block numbered from "start-end" to follow the line numbered "to".

A "<F>ind string" command will search the text buffer starting from current line+1 for the next occurrance of "string". String may be up to 15-characters in length. If "string" is null, the next occurrance of the previous "find string" will be searched for.

Single line scrolling is supported with the <UP-ARROW> and <DOWN-ARROW> keys. The <SHIFT-CLEAR> key invokes a "warm-boot" which aborts the current operation, clears the screen, and re-initializes line numbering while maintaining the current text buffer.

A "<U>sage command displays buffer status (in use and remaining), and the first available in-memory address. The latter is useful for assembling into memory then executing a "<B>ranch" to the in-memory object program for debugging purposes.

EDAS provides MiniDOS-type directory "<Q>uery" and file "<K>ill" functions. A "<V>iew" command will list a source file to the screen without affecting the buffer contents.

When all things are considered, if you are writing system software, support software, applications - big or small, EDAS will provide the power to make your assembly job easier, faster, and more worthwhile. It does everything but teach you how to program. EDAS comes complete in a three-ring binder with extensive documentation of over 100 pages of useful information (not OP-code explanations). A Z-80 quick reference card is included. EDAS, for LDOS equipped Model Is and IIIs (and Model IIs).

LDOS

LDOS is a new generation of operating system for the TRS-80. It is a totally device independent system, capable of device linking, routing, setting, and filtering. LDOS will support up to eight logical drives, including 5" (up to 80 tracks) and 8" floppies, single/double density, single/double sided, and all available step rates are supported. Hard disks are supported, up to 13 megabytes as a single drive. Hard drives may be partitioned to represent up to eight logical drives, depending on the number of heads on the drive. (Note: specific hardware may be required)

Model I/III LDOS disks can be either single or double density, and can be read or written on either machine. Model I LDOS supports double density with the Lobo LX-80 interface, or the Radio Shack, Percom, or Aerocomp double density adaptors.

LDOS is completely documented in an extensive operating manual in excess of 300 pages, containing both user instructions and a large section with technical information. Numerous examples are given to detail all operating functions.

To facilitate handling large numbers of files, all files created under LDOS carry their date of creation or last modification, and are marked with a "MOD" flag, if modified since their last backup. Many LDOS commands and utilities can manipulate files by user specified file extension, full or partial file name (including the use of wildcard characters), by MOD flag, or by date or range of dates.

LDOS comes with an extensive Job Control Language (JCL). This is a compiled language that allows the user to input commands and Job Control conditionals and execution statements into a file which will control the computer's job stream. Execution can be tied to the setting of the real-time-clock, and can provide both visual and audible alerts. Variables and labels may be assigned by the user at run time to select the actual Job Control execution and starting position in the JCL file.

LDOS comes complete with an RS-232 driver program, a terminal communications utility which includes automatic file transfer, a complete ASCII keyboard driver with 128-character type-ahead, a KeyStroke Multiply filter for key re-definition, full printer spooling to memory and/or disk, a printer output formatting filter, a disk-modifying extended DEBUG utility, a file PATCH utility, a feature to reside /SYS files in memory for super fast operation, and much, much more.

LDOS includes a Minidos keyboard filter that provides constant access to certain system functions such as directory, free space, kill, debug, character printing and top-of-form.

A Job Log driver can be used to send a list of all commands and system error messages along with a time stamp to a specified file or device. This is generally found only in main-frame systems.

LDOS includes an enhanced disk BASIC with high-speed load and save, run multiple programs with common variables, blocked files of 1-256 LRL, execute LDOS commands from BASIC, string array sort, RESTORE nnnnn, line re-numbering and cross-reference listings, additional file modes for opening "old" or "new" files, and more.

An extensive set of LIBRARY commands are stored and rapidly invoked from an ISAM accessed partitioned data set. These include APPEND {concatenate two files or device to file}, ATTRIB {set or alter file or disk pack attributes}, BUILD {create ASCII or packed hex files}, COPY {transfer an image of a file/device to another file/device}, CREATE {pre-allocate file space that will NOT shrink}, DEVICE {display status of all enabled disk drives, devices, and user options}, DIR {obtain SORTED disk drive directory data including attributes, file space, date, LRL, extents,...}, DO {compile and/or execute a predefined series of commands and keystrokes stored in a JCL file}, DUMP {save a core-image in ASCII or CMD format}, FILTER {invoke a device transfer function}, FREE {obtain a free-space map or line listing for all disk packs}, LINK {couple two devices together}, LIST {obtain a file listing in ASCII or hex format}, MEMORY {display and/or set HIGH$, directly alter/clear memory, jump to an address}, PURGE {selectively delete files by screen prompting}, RESET {return a device or configuration to power up status}, ROUTE {redirect the I/O path of a device to another device or file}, SET {establish a new device}, SPOOL {invoke the system's printer spooler}, SYSTEM {alter many system parameters such as: BREAK key; blinking cursor; drive write protect, step rate, delay; invoke SVC processor, DATE/TIME prompts; establish disk drivers}, and more.

The BACKUP utility performs mirror-image or by-class transfer of disks/files. More options than you can shake a stick at, including {only those files not already backed up, backup by date range, backup only visible/invisible/system files, backup only files currently existing or not existing on the destination disk, backup between disks of different configurations with swap disk prompts when the destination becomes full}.

CMDFILE is included to transfer SYSTEM tapes to CMD files and vice versa. Also concatenates two or more CMD files. CMDFILE will perform load module offsets, as specified by you.

CONV is a utility to transfer files from Model III TRSDOS. The REPAIR utility is provided to turn non-LDOS single-density disks into LDOS disks usable on Model I or Model III.

The FORMAT utility will write formatting information on 5" or 8" floppy diskettes in either single or double density, or one or two sided (with appropriate hardware).

The LCOMM program is an advanced communications package that allows machine to machine communications, supporting the keyboard, display, printer, and transfer of disk files. LCOMM is switch controlled with menu display.

LDOS is THE system. If you are looking to step up to an advanced DOS, or are just plain frustrated with your present DOS, this is the solution. Logical Systems also has a one-year warranty program available that includes minimum cost updates, four issues of the LDOS QUARTERLY, MicroNet LDOS SIG, and more. Please specify Model I or Model III.

LC COMPILER

One of the high-level languages getting a great deal of attention lately is the "C" language. This is due in part from the knowledge that UNIX (*), a powerful operating system for minicomputers, mainframes, and now micros, is written in the C-language. Why did they choose C? Because the UNIX designers realized that application software and system code could be both created and maintained more easily when written in the high-level C-language. Another reason for C's growing popularity, is that it is a language rich in the use of expression operators, functions and structured code.

If you would like to get started in C, or you are a C expert just waiting for the perfect Model I or Model III release, your waiting days are now over. "LC" (pronounced 'elsie'), a C-language compiler, is now available for use with your LDOS. LC provides a substantial subset of the C programming language as described in, "THE C PROGRAMMING LANGUAGE" by Kernighan and Ritchie. LC was written to be compatible with UNIX programs. LC programs using the standard library (supplied with the compiler) can be compiled and run under UNIX. Programs written under UNIX which use only statements supported by LC are also portable to LC. A large amount of existing software, both commercial and public domain, will be directly usable by LC owners.

C is a structured, portable language. A "C" program is a collection of functions arranged hierarchically. C functions can be recursive and re-entrant, as local variables are created and stored in a stack. All machine-dependent features needed, such as I/O, are not implemented in the language; rather, they are placed in the standard library. Thus, only the implementation of the standard library changes from installation to installation, and C programs are written in machine independent ways. The language itself provides ways of expressing program structure, and of giving arithmetic and logical expressions. C is known for having one of the most powerful expression capabilities available in any language. C statements supply the WHILE, DO-WHILE, FOR, IF, and SWITCH-CASE constructs. C also provides powerful pointer capabilities to enable direct access to memory and variable storage.

LC is an integer-only implementation of C which provides all C statements except "struct", "union", "goto", "switch-case", and "typedef". All data types except "float" and "double" are implemented; "long" and "short" declarations are accepted, but 16-bit fields are used for all integers. In LC, "char" variables are implicitly unsigned. Single-precision and double-precision floating point operations are supported via functions supplied in the FP/LIB library included with the LC compiler. LC accepts multiple input files, with four levels of nesting for "#include'd" files. The compiler generates an EDAS Version IV assembler source file which is then assembled with the standard library and any other libraries needed to resolve function references in order to generate the executable program. The value in generating assembler source is twofold. First, you can obtain a complete machine code source listing which could prove invaluable in debugging complex code. Second, local optimization of assembler source code can be performed as required by the experienced assembler programmer. The LC standard library provides such functions as standard I/O redirection, dynamic memory allocation, automatic standard I/O opening and closing, and program chaining. In addition, functions specific to LDOS and the Model I/III are supplied in an installation library, to provide access to such functions as graphics and system entry points.

LC supports separate compilation; programs may be compiled in segments, and frequently used functions can be pre-compiled. You can create your own library of commonly used functions with the Partitioned Data Set utility (PDS is not included with LC but is available as a separate package). The assembler source code output by LC is designed to use the extensive SEARCH and conditional assembly support in EDAS Version IV. The assembler and companion assembler cross-reference utility are supplied with the LC package. You need nothing more to start writing and running C-language programs except your LDOS-equipped computer and a copy of "THE C PROGRAMMING LANGUAGE". A 48K-RAM two-drive Model I or Model III is required.

Some highlights of the "elsie" compiler are:

- o Integer subset of the C language.
- o Access to floating point routines in ROM via function calls.
- o All statements supported except STRUCT, UNION, TYPEDEF, SWITCH-CASE, GOTO.
- o All operators supported except "->", ".", SIZEOF, and (TYPENAME).
- o UNIX-compatible standard I/O library.
- o Standard I/O redirection with complete device independence.
- o Input using FGETS or GETS functions support LDOS Job Control Language.
- o Dynamic memory management (ALLOC, FREE, SBRK).
- o Sequential files open for READ, WRITE, and APPEND.
- o Generates Z-80 EDAS Version IV source code as output.
- o User libraries in Z-80 source ISAM-accessed PDS files.
- o Compact one-line invocation of the compiler.
- o LC's interactive friendly interface provides easy way to learn LC options.
- o Supports separate compilation of functions.
- o Compiled programs run under both Models I and III without modification.
- o Installation library gives access to graphics and LDOS entry points.
- o Supplied with example programs and utilities in source form.
- o LC/LIB includes: FPRINTF, PRINTF, ALLOC, FREE, SBRK, and String functions.
- o The LC package is Model I/III LDOS compatible and includes **LC/CMD, LC/LIB, FP/LIB, IN/LIB, EDAS-IV, XREF,** and more than 200 pages of documentation.

With LC, in no time at all you will be writing C programs such as:

```
#include stdio/csh /* standard I/O definitions */
/*  XFER - copy standard input to standard output  */
int  c, bytes, lines;
FILE *fp;
main()
{   while( (c=getchar()) != eof)
    {   putchar(c)+
        ++bytes;
        if (c == eol) ++lines;
    }
    fp = fopen("*do","w");
    fprintf(fp,"$d characters , $d lines were copied", bytes, lines);
}
```

This program copies standard input (*KI) to standard output (*DO) while it counts the number of characters and lines. However, with LC's I/O redirection, input and/or output can be changed to any other device or file. Type directly into a file or copy a file to a printer!

Modernize your programming skills and begin writing maintainable applications. Get C - get LC!

* - UNIX is a trademark of Bell Laboratories

PDS

Katzan, in OPERATING SYSTEMS, A PRAGMATIC APPROACH, defines a Partitioned Data Set (PDS) as "a data file that is divided into sequentially organized members." Katzan further states, "Each PDS includes a directory that points to the beginning of each member. Data sets of this type are most frequently used to store object programs - each member corresponds to a single object program. The PDS as a whole is referred to as a library. Operating system libraries and user libraries are stored in this fashion." This definition describes exactly, the two LIB files in LDOS, SYS6/SYS and SYS7/SYS.

The PDS structure has provided a technique for combining separately executable object programs into one file thereby saving directory slots. It also saves time by not having to load an entire 10K-15K file just to get a few hundred bytes or a few thousand bytes of program loaded if all LIB commands were just one big file. The system overhead of having to read and search the member directory is minimal.

Up until now, only the system library has supported the PDS structure. Now, with this PDS utility from MISOSYS, you can have "user" PDS structures. The PDS command can be used to create custom libraries. A library could be a collection of a dozen utility programs - all stored under one name but directly executable by specifying the library name followed by the member name. Consider for a moment, that you have built a library containing CMDFILE, DSMBLR, FED, BINHEX, EDAS, and XREF. The library name MYLIB was chosen. You can then execute EDAS by entering:

```
MYLIB(EDAS)
```

at the LDOS ready prompt. If you wanted to build a custom LDOS command library, you could use CMDFILE to extract DIR, COPY, KILL, DEBUG, ROUTE, and RESET from SYS6/SYS and SYS7/SYS and build them into a user SYSLIB. Then you could kill off SYS6 and SYS7 which would save about 15K from your "custom" SYSTEM disk. When you want to do a directory, you would only need to type:

```
SYSLIB(DIR) :2 (A,I)
```

to achieve the same result as if you had typed DIR :2 (A,I) on a regular SYSTEM disk. Albeit you could have named your user library, "S" and save the entering of five characters each time you wanted to execute a member of the library. That would let you use "S(DIR)"! PDS also allows you to abbreviate the member's name to as few characters as uniquely identify it. If "DIR" was the only member starting with the letter "D", you could even have entered, "S(D)..."

The PDS command is itself a Partitioned Data Set and supplies the following functions via installed members:

o APPEND - Appends a new member to the existing PDS and updates the member directory and ISAM table records.

o BUILD - Creates a new Partitioned Data Set. The PDS is composed of a Front End Loader program, a MEMBER directory, and an ISAM table.

o COPY - Transfers an image of a PDS member from the PDS to a designated file.

o DIR - Provides a directory listing for each member with its name, type, date of addition, and file space occupied.

o KILL - Makes a member inaccessible for access.

o LIST - Will list a specific member in standard hex format or ASCII format.

o PURGE - Removes killed member(s) from the PDS and compresses the file to reclaim the space previously occupied by the killed member(s).

o RESTORE - Restores a killed file to accessibility.

Here is a sample PDS directory:

```
PDS: U/CMD          Ø7/Ø7/82    Size:  45K Members: 15/ 16
convcpm  P 13-Mar-82  1597      dct      P 13-Mar-82  362Ø
debugger P 13-Mar-82  2398      dircheck P 13-Mar-82  2137
dirlist  P 27-Mar-82   957      doconfig P 3Ø-Apr-82   459
dsmblr2  P 13-Mar-82  5724      edas     P 13-Mar-82 1Ø123
fed      P 13-Mar-82  73Ø8      led      P Ø7-Apr-82  5699
monitor  P 13-Mar-82  1814      reformat P 13-Mar-82   614
strip    P 13-Mar-82   767      unhash   P 13-Mar-82   346
xref1    P 13-Mar-82  2127
```

Note that it is sorted, shows the size of each member (in bytes), and has the date that each member was added to the PDS. This is an excellent tool to organize your disk space and unclutter your directories! PDS is distributed on a single-density 35 track data diskette. It requires LDOS Model I or Model III Version 5.Ø.3 or later.

MSP-Ø1

The MSP-Ø1 package is a collection of four utility programs to further enhance the use of your LDOS. These programs are entitled: DOAUTO, DOCONFIG, MEMDIR, and PARMDIR. Each program functions under Model I or Model III LDOS; the package is distributed on a 35-track single density diskette.

How many times do you suppress the execution of an "AUTO" command by holding down the <ENTER> key on boot-up only to later decide you want to execute the AUTO. However, you really don't remember the specific syntax of that complicated LBASIC command line that auto-executes. Wouldn't it nice to be able to command the AUTO to execute without having to type BOOT or hit the RESET button? The **DOAUTO** command is a short program that will execute the "AUTO" command buffer located on ANY drive - not just the SYSTEM drive. It's as easy as typing "DOAUTO :2".

DOCONFIG is a major enhancement of the configuration capabilities of your LDOS. DOCONFIG works in one of two ways. You can SAVE the current configuration of your system to ANY file of your choice on any drive of your choice. You can also restore the machine's configuration at any time from any of the configuration files you created. The configuration file is constructed identically to the LDOS CONFIG/SYS file, except that now YOU control configurations without having to re-boot your machine.

DOCONFIG can even be executed from a Job Control Language file to either SAVE or RELOAD a configuration file while the JCL is executing. This will work even if a re-loaded configuration changes the drive assignment for the drive currently executing the JCL file - be it the system's SYSTEM/JCL file or your own execute-only JCL file. DOCONFIG is smart enough to correct the JCL interfacing being done by LDOS if drive assignments are switched. If the JCL is SAVING a configuration, the CONFIG file will not reflect JCL as being active. The use of DOCONFIG now gives JCL more power to run job streams that require revised high-memory configurations for selected applications. Wow, dynamic reconfiguration - on the fly!

Ever wonder what in the world was up in high memory when you execute a MEMORY command and it says HIGH$=X'E123'? Where did all that memory go? No need to wonder any more. **MEMDIR** is here to give you a directory of high memory. It tells you what program/module is there, where it resides, and how long it is. MEMDIR makes use of the front end linkage protocol as documented by Logical Systems in the January 1982 LDOS QUARTERLY and requires all high-memory modules to adhere to that standard.

The biggest part of the MSP-Ø1 package is **PARMDIR.** This is a tough one to explain. Essentially, PARMDIR is a specific-purpose report writer that uses the on-line disk directories as a data base of information. PARMDIR was originally written to automatically generate Job Control Language files based on tests of data contained in the directory. For instance:

```
PARMDIR /DOC:3 REN:Ø (A="RENAME ",X="/SCR")
```

will produce a JCL file containing an entry for all files on drive 3 that have an extension of "/DOC". Each JCL line of the file, REN/JCL, will appear as: "RENAME filename/DOC:3 /SCR". If the parameters were entered as "(A,X)", then each JCL line would appear as: "#A# filename/SCR #X#". Thus, at JCL compilation time, parameters may be substituted for "A" and "X".

However, PARMDIR goes light years beyond this simple example. You can have any of the parameters A,B,C,X,Y,Z be constructed with directory data information for each filespec selected. The information is positioned according to key-word assignment within the parameter string. For example,

"(A="$NAM $EXT $LRL $REC")"

will recover in each output line, the file name, extension, logical record length, and number of records. Keywords are available also for protection level ($PRO), ending record number ($ERN), file date ($DAT), end-of-file byte location ($EOF), drive spec ($DRV), volume name ($VNM), volume date ($VDT), or the entire volume id ($VID).

Each of the keywords (except filename/ext) may be tested for value comparisons in order to select the directory record for output. The comparison is constructed as a complex "IF expression" syntax. For example:

IF="$LRL <= 18 & $REC < 3"

selects those directory records with a logical record length of from 1-18 only if the number of records is less than 3. If you make an error in the syntax of the expression, PARMDIR will tell you exactly what character was in error - that's friendly!

The output can be directed to any file or device and the output is SORTED by filename/extension. Since PARMDIR can make extensive use of parameters, you can enter parms in the command line OR from any file or device. You can create a PARMSLIB disk file that contains NAMED parameter procedures and refer PARMDIR to the specific procedure of parameters for a particular execution of PARMDIR - just like JCL can use a PROCLIB with named JCL procedures. PARMDIR even permits you to type in parameters from the keyboard at execution time if you select PARMS="*KI" as the parameter input device. There is no limit to the amount of parameters that can be entered from a parameter file or device input - only the command line limits its entry to 63 characters max.

When PARMDIR generates its JCL file, all of your parameter entries are written as comments to the output. You are even provided a parameter to suppress these "notes". PARMDIR can automatically generate a Partitioned Data Set (PDS) "MAP" file as easy as "PARMDIR /TXT:2 ROYS/MAP (MAP)".

PARMDIR can access the directory information of a specific drive or all on-line drives. You can use PARMDIR to construct customized directory listings. Use it to mechanize your JCL file construction. In essence, PARMDIR is the most versatile program to come along that lets you tap the data contained in directories.

PARMDIR is worth the cost of the entire package. However, you get PARMDIR, MEMDIR, DOCONFIG, and DOAUTO in the MSP-Ø1 package. You won't know how you ever got along without it.

M I S O S Y S P R I C E L I S T

Effective August 1, 1982

CON8ØZ - translate 8Ø8Ø to Z-8Ø $5Ø

CONVCPM - Transfer CP/M files to LDOS $4Ø

DSMBLR-II for Model I/III disk $2Ø

EDAS Version IV Model I/III ..... $1ØØ +

EDAS Version IV Model II (includes PDS lib) ..... $2ØØ +

FED - LDOS File Editor $4Ø

FILTER PACK-I $6Ø

GRASP - Graphics Support Package $5Ø

HELP/QRC - for LDOS 5.1 (requires l/c video) $25

I/O MONITOR $25

LC Compiler (LC, EDAS-IV, XREF, LOBLES) ..... $15Ø *

AVAILABLE 9/82

LDOS 5.1 Mod I or III (deduct $35 if ordering both) ... $129 *

LED - LDOS Text Editor $4Ø

MEMDISK $4Ø

MSP-Ø1 (DOAUTO, DOCONFIG, MEMDIR, PARMDIR) $5Ø

PDS - Partitioned Data Set utility $4Ø

SOLE - DDEN booting LDOS Model I $25

THE BØØK, Vol I or II - while they last $1Ø

ZGRAPH - Graphic Screen Editor $4Ø

Shipping: Items marked "*", add $5. Items marked "+", add $4
All others, add $1.5Ø plus $Ø.5Ø per item. COD add $1
"IF LDOS or COD then UPS else USPS"
VA residents, please add 4% Sales Tax

M I S O S Y S O R D E R F O R M

Company Name: _____ Date: _____

Individual: _____ P.O.#: _____

Address: _____ Apt#: _____ Phone: ___+___-_____

City: _____ State: _____ ZIP: _____

Payment: Check/MO { } MC/VISA { } Signature _____ COD { }

Credit Card #: _____ Bank #:_____ Expires: _____

Qty	Description	Net Each	Total
	LDOS 5.1 Model I { } Model III { }		
	EDAS-IV Model I/III { } Model II { }		
	LC Compiler (LC, EDAS-IV, XREF, libraries)		
	GRASP - Graphics Support Package		
	ZGRAPH - Graphic Screen Editor		
	MSP-Ø1 (DOAUTO, DOCONFIG, MEMDIR, PARMDIR)		
	Partitioned Data Set utility (PDS)		
	SOLE - DDEN booting LDOS Model I		
	HELP/QRC - for LDOS 5.1 (requires 1/c video)		
	DSMBLR-II for Model I/III disk		
	CONVCPM - Transfer CP/M files to LDOS		
	CON8ØZ - translate 8Ø8Ø to Z-8Ø		

Send to: MISOSYS [7Ø3+96Ø-2998] Sub-Total | |
P.O. Box 4848
Alexandria, VA 223Ø3-Ø848 VA Residents Sales Tax | |

>>>>>>>>>>>>>>>>>>>><<<<<<<<<<<<<<<<<<<< Shipping Charge | |
>>> $1.ØØ Credit when your order is <<<
>>> placed using this order form!!! <<< Total | |
>>>>>>>>>>>>>>>>>>>><<<<<<<<<<<<<<<<<<<<

GRASP

The GRAphics Support Package (GRASP) is a collection of programs, filters, and drivers that will enhance the capabilities of your Epson MX-80 Graftrax or MX-100 printer. GSP implements customized character sets which include standard ASCII characters, TRS-80 graphics blocks, and Model III special character symbols.

A screen-oriented character editor makes it easy for you to modify or create any character font you desire up to a size of 16 vertical by 12 horizontal dots. If you use the double-character mode, your character font can occupy a width of up to 24 dots. The editor displays an individual character in a visual matrix made up of large graphics blocks. By manipulating the graphics cursor within the matrix, you control exactly what "dots" will be present in your character.

Filters are provided to toggle underlining and invoke selected double-width characters intermixed with standard width. Another filter gives you the capability of printing the Model III special characters with a minimum of high-memory usage.

A program is provided to easily set the custom functions of your MX-80G or MX-100 from the DOS Ready mode instead of having to write complex PRINT CHR$ instructions.

GRASP works with both a Model I or a Model III TRS-80 or compatible microcomputer running under LDOS. The GRASP package includes ALTCHAR/CMD, ALRCHAR/DVR, ALTLINE/FLT, ALTWIDE/FLT, MOD3CHAR/FLT, GPD/DVR, SETMX80G/CMD, SETMX100, and UNDRLINE/FLT - 9 programs in all.

The ALTCHAR/CMD program is a special-purpose graphics editor for you to use in constructing and customing entire character sets for use with the ALTCHAR/DVR printer driver.

ALTCHAR comes supplied with seven already defined character sets which are: STD10/12 - a 10/12 pitch character set of "standard" characters, block graphics, and Model III special characters; TYPE10/12 - a 10/12 pitch set of typewriter like characters, block graphics, and Model III special characters; SCI10/12 - a 10/12 pitch set derived from STD10 which includes greek characters plus superscripted and subscripted numerals; and OLDENG - a 10 pitch double-width character set of Olde English characters.

ALTCHAR/DVR implements the printer support drivers that will use the character files to generate the customized character sets on your printer. The driver options include the following parameters: ADDLF will cause a line feed to be sent after each carriage return; SPACE will cause the output of an extra one-half line feed between each line of text; WIDTH establishes the number of characters to print on a line; DOUBLE will cause the interpretation of the character set as being "double-width". HIGH will allow the printing of only characters with an ASCII value less than or equal to the value specified. Only the necessary portion of the character data set will be read and stored in memory, thus allowing you to cut down on the ALTCHAR driver high memory requirements; LENGTH will set the page form length in one sixth inch lines.

ALTLINE is a filter to implement character underlining using a toggle character. The ALTLINE filter works in conjunction with the ALTCHAR driver to allow the printing of a continuous underline with little user intervention. Upon receipt of the switch toggle character, ALTLINE will underline all characters until either the end of the line is reached or the switch toggle character is detected. The toggle character is not printed.

This ALTWIDE filter provides the capability of printing selected characters in double width while all others are printed in standard width. It could be used, for instance, to print all capital letters in double width.

MOD3CHAR/FLT is a filter that adds the capability to print the special video characters as displayed on the Model III without the high memory overhead needed by ALTCHAR. If you only need the special characters, this filter will do it; however, ALTCHAR is still needed for your custom character sets.

GPD/DVR allows the use of all dot addressable graphics on the Epson printers. GPD/DVR replaces the printer driver routines located in the TRS-80 ROM. The TRS-80 ROM printer driver routines convert some characters and trap others. GPD/DVR eliminates this problem. When GPD/DVR is set, ALL codes will be passed unmodified to the printer.

The SETMX80G and SETMX100 utilities permit conviently setting the printer options for the Epson MX-80G or MX-100 printers. Command line options for MX-80G are:

RESET	- reset to defaults	RSmode	- Radio Shack mode
Paper	- paper transfer mode	Emph	- emphasized mode
Comp	- compressed mode	eXpand	- expanded mode
Italic	- italics mode	MSB	- MSB function
Double	- double strike mode	Space	- line spacing
Form	- form length in lines	Lines	- lines per inch
Margin	- restores PR/FLT left margin		

The SETMX100 program also supports the following:

US/FRench/GErman/ENglish/DAnish/SWedish/ITalian/SPanish			
SKip	- skip over perfs	COLumn	- column width

UNDRLINE/FLT is used to provide an easy means of underlining on any printer that will backspace (without erasing) and print an underline character (ASCII 95). This filter will work with the Epson MX-80 w/Graftrax but not with the Epson MX-100. The character specified by the parameter, CHAR, will be used to start and stop (toggle) underlining.

GSP is complete for your Model I/III LDOS machine and Epson printer. Seven character sets are provided with GSP to get you on your way to developing your own custom fonts. Herewith are some examples:

ZGRAPH

ZGRAPH is a graphics editor that allows creation of graphic images. ZGRAPH possesses two sets of commands, primary commands and secondary command functions. A 'help' list of commands at both levels is available by typing <H> for primary commands or <F><H> for secondary functions.

The video display screen of the TRS-80 consists of 1024 bytes of memory arrayed as 16 rows of 64 columns. Each memory location is capable of displaying one ASCII or special character or any combination of the six (2 wide by 3 high) graphic dots referred to as pixels. ZGRAPH allows any of the 160 (224 on the Model III) possible characters (ASCII, graphic and special) to be displayed at any point on the screen.

Cursor movement depends on the mode that ZGRAPH is in. In the graphics mode, movement is achieved using the number keys 1-4 and 6-9. If you go off the screen to the left, you will reappear on the right. The same is true of the top and bottom.

In the DRAW mode, the cursor will leave a trail of bright graphic pixels everywhere it goes. In the ERASE mode, the graphic pixels will be turned off everywhere the cursor is moved. The MOVE mode is a non-destructive means of moving the cursor. While in the text INSERT mode, cursor movement is via the arrow keys. The cursor is non-destructive of both graphics and text. Simply move the cursor to the desired position and start typing text.

The entire screen can be reversed (graphic on/off) via the REVERSE command. Text will not be reversed. The XFLIP command will create a mirror image of the screen about the Y-axis. The graphics will be a true mirror image and the order of text characters will be reversed. The YFLIP is similar to the XFLIP except rotation is about the X-axis.

ZGRAPH has five in-memory screen buffers in addition to the video display screen. Four of these buffers are general purpose buffers and are available to the user to store displays. This is useful when creating a large graphic consisting of several ZGRAPH images or in creating those images using the MERGE function. ZGRAPH can also load and save images to disk files. All data moving to and from the disk passes through the primary video display. The fifth internal display buffer is used for error recovery in case you make a mistake (perish the thought!).

GET is the function for loading the video display screen from a disk file or one of the buffers. Any one portion of the screen can be saved to a buffer or file by using the SAVE command. MERGE allows you to superimpose one graphic image over another. If you want to exchange the screen display with a buffer, use the XCHANGE command.

ZGRAPH provides functions to make your graphics generation easier. The DUPLICATE command replicates a block defined by markers to another area of the screen. LINE will establish the best fitting line between the marker SET and the current cursor position. The marker position will be updated to the current cursor position after each line is drawn providing an easy way to construct lines connected end-to-end. The RECTANGLE command creates a rectangle with opposing diagonals being the SET marker and current cursor position. The CIRCLE function "rounds out" the ZGRAPH graphics functions by drawing a circle or an arc around the current cursor position.

While in the WINDOW mode, the entire screen display will move in response to the arrow keys. Any part of the image moved off of the edges of the screen is erased. This command is very useful to reposition an entire image on the screen.

To allow ZGRAPH created displays to be used in other applications, the BINCONV post-processing program is provided. ZGRAPH's standard file format is a pure binary representation of the screen display. Each line of the screen memory is saved as the values of the memory bytes terminated by a carriage return. BINCONV converts its standard file formats to:

<1> - ZGRAPH to Load Module in order to create an executable /CMD file that will place your image on the screen;

<2> - ZGRAPH to Packed BASIC - creates a file of packed graphics strings with each line consisting of the string {ZG$(#)="packed value of one line of your image"} starting with an index (#) of Ø, line number of 1ØØ and line number increment of 1Ø;

<3> - ZGRAPH to BASIC Data which creates BASIC data statements of 16 decimal numbers representing the sequential values of your screen image;

<4> - ZGRAPH to EDAS creates a file in assembler source format of DEFB statements with 16 decimal values per statement representing the values of the bytes of your image. This file may then be merged into an EDAS assembler program.

The ZGRAPH graphics package also includes a keyboard filter, DOSAVE, that is similar to the LDOS screen print function. However, where the screen print directs an image of the screen to the printer, DOSAVE will direct the screen image to a disk file specified by the user at the time you depress <CLEAR><SHIFT><S>. These screen files may be loaded into ZGRAPH for further operations. Also included is the BINPRINT program which provides the capability of printing a binary graphic file to a printer that supports compatible bit graphics (MX-8Ø/Graftrax, MX-1ØØ).

You get ZGRAPH, BINCONV, DOSAVE, and BINPRINT in this graphics editor. In no time at all, you will be using images such as:

I/O MONITOR

MONITOR is a disk I/O error monitor designed to run exclusively with LDOS. Its purpose is to intercept a disk read or write error and offer the operator certain options. MONITOR will display a brief message informing you that an error occurred. For example:

```
-->> MONITOR ACTIVATED:

Error Code #4, Drive :2, Track: X'25', Sector: X'14'
SYS8 Resident, Buffer at X'6AØØ', Returns to X'4789'

    <A>bort - <C>ontinue - <I>gnore - <R>etry ?
```

advises you of a "parity error during read" of sector 2Ø on track 37.

MONITOR will be especially useful when using programs that do not incorporate sophisticated error trapping to manipulate files. With MONITOR installed, disk I/O errors which would normally abort processing, may be intercepted, giving YOU the ability to abort or continue. Thus, decisions on error handling may now be made by YOU rather than the program, and may be based on the conditions existing at runtime.

Four options will be available for handling the error:

1> ABORT: This option will abort the I/O operation and attempt to return to LDOS Ready.

2> CONTINUE: will pass control back to the module that originated the I/O operation request. This will result in the error code being passed back to the calling program to permit it to take whatever action it can muster.

3> IGNORE: will cause MONITOR to cancel the I/O error code and continue with the operation back to the calling program as though no error occurred. IGNORE could get you through a BACKUP with a troublesome drive; however, any file written is not normally usable.

4> RETRY: will repeat the I/O attempt that caused the error. Infinite retry is possible as long as you have the patience to enter the <R>. Usually, if a soft error has occurred, it takes only one or two re-attempts to satisfactorily complete an I/O operation.

End your I/O error woes! MONITOR is great on those troublesome 8Ø-track double headed drives. Even if soft errors are caused by your old drives migrating out of alignment, MONITOR may save the day. If soft errors continue at an excessive rate, it is recommended that you investigate possible hardware or diskette problems. MONITOR will work on a Model I or Model III LDOS system.

LED

LED is a screen-oriented text editor that is designed to work with LDOS, Model I or Model III. It was written for Logical Systems by Rick Wilkes, of SuperScript fame. Although very versatile, the LED commands are easy to learn. Those familiar with the LDOS LSCRIPT version of Scripsit will notice a similarity in the command key layout. This is the LED command menu, and can be displayed at the bottom of the video screen while using LED.

```
INDNT  FIND CHANGE HEX  UNMRK DNP  UPP  ALL  AGN      NAME EXIT
 =1=   =2=   =3=   =4=   =5=  =6=  =7=  =8=  =9=      =:=  =-=
INSRT  LIN   DEL   WRD   BLK  END  TOP  SPA  TAB      MENU SAVE
                           {TEST/TXT:Ø-R}   (Ø):X'ØØ'|35751
```

The display contains the name of the file currently being edited, the current cursor column, the hex value of the character under the cursor, and the available memory in the text buffer.

Since LED uses the LDOS keyboard driver, type-ahead, and all keyboard filters are available for use with LED. Also, the entire ASCII character set is available directly from the keyboard (LED uses the extended cursor mode of the LDOS KI/DVR).

Cursor positioning is done in the normal manner, with the four arrow keys controlling the cursor motion. The <CLEAR><ARROW> keys will move to the top or bottom of the text, or to the left or right end of a line. The <SHIFT><LEFT> and <SHIFT><RIGHT> arrows also perform movement to the ends of a line unless tabs are set. Then, they position either to the next tab location or back to the previous one. There are four different cursor characters, depending on the mode you are in (typeover, insert, insert line, or delete). The UPP and DNP commands are used to move the display buffer up or down a full page at a time. If the file to be edited has a /KSM extension, LED will automatically display the alphabetic letter before each line assigned to that letter.

LED can be used on many different types of files. The FIND and CHANGE commands make it handy for doing global changes in LBASIC programs (saved in ASCII). The AGN and ALL commands let you find or change things one at a time or all at once.

A very useful feature is the HEX mode. This mode is available either when overtyping or when inserting. It allows you to input characters as two hexadecimal digits over the entire X'ØØ' to X'FF' range, making possible direct editing or inputing of graphics characters.

Certain parameters may be specified when first entering LED. TABS will cause any X'Ø9' tab character to be expanded. Tabs normally appear as a small graphics block. SAVE="filespec" will save a file under a name different from that which was used to load the file. XLATE=X'fftt' will perform a character translation when loading and saving a file. Two other parameters, END=X'ØØ' and WP deal with word processor files that use an X'ØØ' to mark the end of a file.

Another nice feature is an automatic SAVE prompt. If you request an exit back to LDOS Ready, and have modified the text buffer, LED will automatically ask you if you want to save the file. If no modifications have been made, an immediate exit to LDOS is done without the prompt. This text editor is a definite bargain.

FED

FED is a powerful file "zapping" utility that will work on Model I or Model III LDOS systems. Its wide range of abilities make it an excellent tool for the advanced user, but its simplicity makes it easy to use for the novice. The editor supports upper and lower case, and all drive types and sizes supported by LDOS. FED can be used for displaying, printing, and modifying existing files. FED works on a file level - not a track/sector level. You can neither create nor extend files with FED; however, its ease of use will make modifications to existing files such a dream that you will wonder how you existed before using FED. Read the following descriptions of FED commands then order your copy today!

FILE DISPLAY - FED works by displaying a single 256-byte record of the specified file. The display will show both the ASCII and hexadecimal equivalents of the bytes in that record. Full cursor positioning makes it possible to position quickly to any byte in the record. FED also provides a 128-byte mode, displaying a user-selected 128-byte window of the current record. In this mode, the decimal and hexadecimal equivalents of the byte at the cursor are displayed. The following is a sample display:

```
..SYS1  .2Copyri.ØØ> Ø5Ø6 5359 5331 2Ø2Ø 1F32 436F 7Ø79 7269.Ø.S
ght (C) 1981 by .1Ø> 6768 742Ø 2843 292Ø 3139 3831 2Ø62 792Ø.Ø.Y
Logical Systems .2Ø> 4C6F 6769 6361 6C2Ø 5379 7374 656D 732Ø.Ø.S
Incorporated...N.3Ø> 496E 636F 727Ø 6F72 6174 6564 Ø1Ø2 ØØ4E.Ø.1
.p..((. ($.@..O..4Ø> E67Ø FE1Ø 2828 FE2Ø 2824 FE4Ø CAC1 4FFE.../
P.5P.@..P.Ø..%B..5Ø> 5ØCA 355Ø FE6Ø CAFA 5ØFE 3ØCØ 1125 42D5.  S
.?...>....#.Ø@1..6Ø> Ø13F ØØED BØ3E Ø312 E118 23C3 3Ø4Ø 31E5.  Y
A.:+D.>.8..2.D!..7Ø> 41FB 3A2B 44Ø7 3EC3 38Ø1 AF32 Ø544 21E5.  S
Q.gD!%B.?.@.8....8Ø> 51CD 6744 2125 42Ø6 3FCD 4ØØØ 38DD CDF1.  :
Q.-@. ..(-.! .2q.9Ø> 51Ø1 2D4Ø C57E FE2E 282D FE21 2ØØ4 3271.  Ø
N#..D..O ...*(...AØ> 4E23 1185 44CD C14F 2ØC1 1AFE 2A28 BCE5.
.Roy...N..P(.!.Q.BØ> 8752 6F79 Ø8Ø1 A84E CDB2 5Ø28 1221 E251.
.5P..3D:+D.o(....CØ> CD35 5ØE1 C333 443A 2B44 CB6F 28B6 C9E1.
.2.Dz....C.>.Ø.<.DØ> AF32 Ø544 7AØ7 D5DØ D143 Ø73E 8830 Ø13C.
..WO.^G.APPEND1..EØ> EFAF 574F CD5E 47C7 415Ø 5Ø45 4E44 318Ø.>ØØ
ATTRIBQ.AUTO  ...FØ> 4154 5452 4942 51CØ 4155 544F 2Ø2Ø 11CØ.C:
```

FILE POSITIONING - FED allows for record advancing, backspacing, and direct positioning. You may page through a file quickly, either forward or reverse. FED will also position directly to the first or last record, and FED will also indicate the true end-of-file byte. You need not know any diskette information such as track number, sector number, diskette density, number of sides, or other drive data as FED automatically handles all spanning of sectors, tracks, and extents. The only thing that is required is knowledge of the proper filespec, and perhaps its password!

FILE MODIFICATION - FED supports complete editing of a file in both ASCII and Hexadecimal modes. Modifications may be done to any type of file - data files, load module format (CMD) files, BASIC programs, LSCRIPT or other ASCII file. When modifying a file, changes are made to a memory buffer containing the desired record, which can then be saved to disk with a single command.

FILE SEARCHING - FED provides for two types of character string searches. These searches allow ASCII strings of up to 30 characters and hexadecimal strings up to 15 bytes in length. The search modes will search the entire file, starting at the cursor location in the currently displayed record. If the target string is found, the cursor will be positioned to the start of the string in the appropriate record. A single key command will also continue to the next occurrence of the search string.

LOAD MODULE FILE SEARCHING - You will be allowed to locate a load address in a load module format file, or calculate the load position of a specified byte. This feature will facilitate the inspection and editing of a load module file. Just type in the load address in question, and FED will position the display to that byte. Another extremely powerful feature is the reverse of the address location command. FED will calculate where in memory a specified byte pointed to by the cursor will load. These two features are worth the entire price of FED.

FILE PRINTING - FED provides for sending a listing of a single record, or the entire file, to a line printer. Forgot to turn your printer on? Ran out of paper? FED will not lock up if the printer becomes unavailable. With a functioning printer, the printout will contain the filespec and drive number, the record number, and the ASCII and hexadecimal equivalents of the 256 byte record.

Forget a command? Just depress <ENTER> and FED will display a menu of its commands just to jog your mind:

<;>	Advance File Record	<BREAK>	Cancels command
<->	Back up File Record	<N><ENTER>	New File
<B>	Beginning Record of File	<S><ENTER>	Save Record
<E>	Ending Record of File	<X><ENTER>	Exit FED
<R>	Position to Record	<H>	Hexadecimal Modify
<Z>	Zip through File blocks	<A>	ASCII Modify
<M>	Calculate Load Address	<T>	Toggle Display modes
<C>	Find ASCII String	<F>	Find Hex string
<L>	Locate Hex Address	<G>	Go next occurrence
<D>	Dump File to Printer	<O>	Output top-of-form
<P>	Send Buffer to Printer		

If you need to do any file "zapping", then you need FED!

FILTER PACKAGE

Now from Logical Systems comes the first in a series of extension packages for the Logical Disk Operating System (LDOS). This package is FILTER oriented, contains many useful modules, and comes with complete SOURCE code.

CALC/FLT - A keyboard filter to perform hex/decimal/binary conversion. Hexadecimal addition and subtraction may also be done.

LINEFEED/FLT - Either add or remove a linefeed after each carriage return.

LISTBAS/FLT - A filter which will format the output of a BASIC program. All program lines which contain multiple statements separated by colons will have their appearance reformatted when displayed.

LOWER/FLT - Converts every alphabetic character (A-Z) to lower case (a-z).

MONITOR/FLT - A filter similar to STRIPCNT/FLT, with the exception that characters less than X'2Ø' will be displayed as a percent sign (%) followed by an ASCII representation of the actual character value + X'41'.

PAGEPAWS/FLT - Will pause after each top-of-form character is printed and wait until <ENTER> is depressed to continue.

REMOVE/CMD - A program to remove each occurrence of a specified byte from a disk file.

SLASHØ/FLT - Will cause a printer that is capable of backspacing to do a backspace and type a slash (/) over every Ø (numeric zero) that is encountered.

STRIP7/FLT - Strips bit seven off of each character.

STRIPCNT/FLT - A filter which will replace an output character above X'7F' or below X'2Ø' with a pound sign (#).

TITLE/FLT - A printer filter that will print a user-defined title after each top-of-form character (X'ØC') is encountered.

TRAP/FLT - Will trap and discard away a certain character each time that the character tries to go through the filter.

UPPER/FLT - Converts every alphabetic character (a-z) to UPPER case (A-Z).

XLATE/FLT - A complete translation filter system, for input/output. Included are an EBCDIC translate system and a DVORAC keyboard table. You can easily build any other translate tables that are needed for special use.

HELP

The HELP series of utility programs provides prompting notes on the LDOS 5.1.2 system commands and syntax. It is supported under LDOS on lower case equipped Model I and Model III machines. The HELP utility contains two types of files; HELP and SYN. The HELP files contain detailed descriptions of the system functions including explanation of parameters and their default values. The SYN files will be useful to the more experienced user. They consist of the syntax necessary to invoke the function without an explanation of terms. Both the HELP and SYN functions can be executed from LDOS Ready or from within LBASIC using a statement of the class: CMD"HELP_(command)".

The HELP programs provide adequate explanations of the specified commands. Each HELP file also contains a HELP function that will yield a menu of the helps within the particular HELP file.

The HELP and SYN files are implemented using the Partitioned Data Set (PDS) utility. Each member of the data sets is an executable machine language program that consists of text that is loaded directly into screen memory, a short routine to position the cursor and a transfer to the LDOS Ready prompt. This manner of implementation allows quick access to any help member within the files with minimal memory requirements and rapid response time. When you type HELP_(command), only a small front end loader actually loads into memory. It, in turn, clears the screen then loads the HELP explanation directly into the screen memory.

Two files are included with the HELP utility package that will assist you in creating your own "HELP" type files. The program, TEXTCMD/BAS, provides an easy means of converting text files created with a text editor into executable CMD files. PDSHELP/FIX is a patch to the PDS front-end-loader which adds the function of clearing the screen. You will need the PDS utility in order to create new HELP files; however, PDS is NOT needed in order to use any of the HELP files included in this package.

The HELP utility also comes with a Quick Reference Card. The QRC is a ten-panel foldout card that identifies all LDOS library commands, utilities, drivers, filters, Disk BASIC, and Job Control Language. With HELP at your disposal and the QRC at your side, keep you LDOS manual on the shelf and consult it when you need in-depth information. Over 90% of your reference needs could be rapidly satisfied with the HELP series. Do you need HELP?

SOLE

LDOS is a sophisticated operating system. The folk's at Logical Systems have expended great efforts in producing such a powerful DOS for the TRS-8Ø users. Paramount in their implementation was the concept of standardization. LDOS makes every attempt at standardizing functions and media whenever possible. The media format chosen for double density operation on the Model I was an entire diskette formatted in double density. Since the TRS-8Ø Model I cannot begin to BOOT a diskette unless the BOOT sector (track Ø, sector Ø) is formatted in single density, the standard LDOS double density diskette cannot be BOOTed. As a result, some users have taken LSI to task for not providing a means of booting a double density disk on the Model I.

Operation of the RESET button causes the Z-8Ø CPU to begin execution at address Ø. The Model I eventually finds its way to a ROM routine which attempts to read a disk booting routine stored on sector Ø in track Ø into a buffer at X'4200'. The problem here is that this ROM routine can only read the boot sector if it is in single density. Since LDOS has a double density track Ø when a disk is formatted in double density, the ROM cold start routine doesn't like it.

The disk boot routine is supposed to read in the resident system, known as SYSØ/SYS. If SYSØ has been read successfully (that means no disk error in big letters), then the booting routine passes control to SYSØ. The resident system initialization routine does its thing, loads in a CONFIG/SYS file if one is available, and finally brings in SYS1 to display the "LDOS Ready" prompt and await your command. A lot of work has been done to get to this point. If you are lucky to have a working double density adaptor, then you would have liked all of this work to take place on a double density diskette.

SOLE is an application to accomplish that goal. It will create a double-density booting SYSTEM diskette for use with LDOS on a Model I. It essentially constructs a single density track Ø on a previously formatted double density diskette. It then proceeds to add a second BOOT routine and double density READ ONLY disk driver to be used to read SYSØ. This SOLE BOOT routine and driver is what the sector Ø BOOT routine will read. Since the track Ø is single density, the ROM can read sector Ø. The SOLE additions are also placed on track Ø so the BOOT routine in sector Ø, which expects to see a single density diskette, actually winds up reading only single density sectors. The sector Ø BOOT passes control to the SOLE BOOT after it successfully loads the SOLE BOOT.

The SOLE BOOT routine interfaces with a double density driver that can do only one thing - read sectors. It reads the SYSØ which is obviously positioned on some double density track. After SYSØ is loaded and before passing control to SYSØ, the SOLE BOOT slides its booting drive code table into the standard drive Ø position. Then when SOLE passes control to the SYSØ initialization, SYSØ is interfaced to the double density read-only disk driver. The requirement here is that an LDOS double density driver needs to be loaded. That is accomplished by having it in a configuration file. Thus, when the initialization part of SYSØ loads in the CONFIG/SYS file, the LDOS double density driver is loaded into high memory and the drive code table data is updated.

SOLE supports PERCOM-type double density adaptors and the Radio Shack type adaptor. SOLE is for Model I LDOS only.

CON80Z

Quite often when you need a specific tool, it is unavailable. For the Z-80 assembly language programmer, the need arises to maintain or modify programs written in 8080 code using Intel mnemonics. Since 8080 code is a subset of Z-80 code, a useful approach is to translate the 8080 code source file to Zilog mnemonics (Z-80) source code. You could hand translate your 8080 files to Z-80 files - a formidable task, indeed! An alternative would be to use a translator program. This tool should prove quite useful in such a task. CON80Z has been designed to facilitate the conversion of assembler source files written in 8080 Intel mnemonics to Z-80 Zilog mnemonics. CON80Z is a source translator to help you convert your 8080 files to Z-80 files - easily.

CON80Z consists of two programs: One, CON80Z/CMD, performs the necessary translations of code on a line by line basis. The translation is one-to-one. Each logical input line is replaced by one output line. The second program, UNNUMBER/CMD, is a preprocessor to CON80Z/CMD and is used to alter certain source files to conform with the requirements for the input file structure.

Although certain code sequences written in 8080 code can be optimized if the Z-80 extensions to 8080 code are utilized, CON80Z performs no such optimizations. CON80Z does help to transform the source into a file structure that can be loaded by your assembler's editor. Most 8080 assembler source files are structured as pure ASCII files with each line terminated by a Carriage Return (CR) followed by a Line Feed (LF). Source lines are also generally not line numbered as is the case with most TRS-80 assemblers. CON80Z will expect the source file to be un-numbered. The line feed may or may not be present.

Some 8080 assemblers support a logical line ending character, such as the exclamation mark (!), to create multiple source statements on one physical line. This is similar to the colon (:) separator in BASIC. By using the CR="c" parameter in the command line, the character "c" will be interpreted as a logical line end when found in the operand field of the source statement and not within single quotes.

Register nomenclature in 8080 code is always a single character. Eight-bit register references in 8080 assembler language are identical to Z-80 references {B, C, D, E, H, L}. The "(HL)" 8-bit memory reference is denoted in 8080 code as the single character M. The appropriate translation from "M" to "(HL)" will be made by CON80Z wherever necessary.

The 8080 16-bit registers available are denoted as B, D, and H with the OP code changed to "extended" to interpret the reference as 16-bit register use (e.g. LD changed to LDX). In addition, the Accumulator and FLAG register are referred to as "PSW" when used in PUSH and POP instructions (PSW is a carryover from main frames and stands for Program Status Word). CON80Z makes the appropriate translations on extended instructions and will translate B, D, H, and PSW to BC, DE, HL, and AF.

During the translation process, CON80Z will convert all comments in upper case characters to lower case characters except for the character immediately following the semicolon (;) comment indicator. CON80Z will also translate multiple blanks used as field separators to one tab (X'09').

CON80Z will perform translations on selected pseudo-ops where there is a similarity of usage on common TRS-80 assemblers. The following pseudo-op translations will be performed: <DB/DS/DW/SET> to <DEFB/DEFS/DEFW/DEFL>.

CON80Z requires LDOS Model I or Model III.

CONVCPM

The CONVCPM utility will allow you to transfer files from certain CP/M diskettes onto an LDOS formatted diskette. CP/M formats supported are standard 8" Single Density and 5" Single Density 128-byte sectoring (Omikron version and equivalent). Two drives are required.

The CONVCPM utility will allow you to move all or groups of files from certain CP/M disks onto your LDOS disks. It provides many different parameters to choose the files to be moved. The file specifications on the CP/M disk must conform to LDOS file specification standards. The filename and extension must begin with an alphabetic character {A-Z}. Subsequent characters of the filename and extension must be either one of the alphabetic characters or a numeric {A-Z,Ø-9}. CP/M apparently permits a filename to begin with a numeric {Ø-9} or certain other non-alphabetic characters. Any CP/M file not adhering to the LDOS standard must be renamed under CP/M prior to a transfer operation.

The CONVCPM utility has been designed to aid in transferring data files and other files that are not directly executable under CP/M {COM files are directly executable under CP/M and although transferable with CONVCPM, they are not "loadable" under LDOS}. Once moved to an LDOS diskette, the transferred file is an exact image of the file as it appeared on the CP/M diskette. The LDOS end-of-file mark is established as if the moved file ended on a sector boundary.

CP/M uses a sector skew translation scheme during disk I/O. CONVCPM has two sector translation tables for commonly used CP/M formats. The single-density 8" diskette structure supported is the Digital Research standard. Each company implementing a version of CP/M on other than 8" single density media chooses their own sector skew translation table. Thus, your version of CP/M that is on 5" media may or may not utilize the same translation table as that used in CONVCPM which is that implemented by Omikron. CONVCPM translation tables are Single Density 8" (1, 7, 13, 19, 25, 5, 11, 17, 23, 3, 9, 15, 21, 2, 8, 14, 2Ø, 26, 6, 12, 18, 24, 4, 1Ø, 16, 22) and Single Density 5" (1, 5, 9, 13, 17, 3, 7, 11, 15, 2, 6, 1Ø, 14, 18, 4, 8, 12, 16). A parameter provides the means for entering a different translation table into CONVCPM.

DSMBLR

This program is a machine language disassembler that produces an assembler source code using ZILOG mnemonics from Z-80 machine language resident in memory. This disassembler operates in two passes in order to incorporate symbolic labels in the source output. The symbolic labels are generated for address and 16-bit references within the start-to-end disassembly request.

References preceding the START address are output as EQUates. A reference is any relative instruction target address or a 16-bit target for load, call, jump, add, or subtract instructions.

Byte or Word values that begin in the range A-F are preceded with a Ø for proper assembly without error.

Output routed to the CRT is displayed in 16-line pages. The display will include the hex address, the hex code, a sequential line number, the OP code, operand, and displayable ASCII characters equivalent to the disassembled instruction's hex code. A page advance is user controlled by key entry.

Output routed to the PRINTER is paged at 56 lines per page. Each page has column headings, supports a user-entered TITLE, and is numbered for producing sophisticated print-outs that look identical to an assembler listing. Columns include ADDRESS, HEX CODE, LINE NUMBER, OPCODE, OPERAND, and ASCII equivalent of the hex code.

Output routed to the TAPE CASSETTE produces a source tape suitable for loading into the Radio Shack Editor Assembler. The tape is generated in blocks of 256 lines of code. A tab character is used between line number, opcode, and operand for best Editor Assembler input.

Output routed to DISK produces a disk file suitable for loading into EDAS, Disk-modified EDTASM, or Microsoft's ALDS (M-8Ø).

Machine language programs that would overlay the disassembler can be relocated by BASIC or a utility and conveniently disassembled with proper address references by using the RELOC feature.

DSMBLR functions under all popular operating systems and is supplied on a cassette tape easily transferable to disk. Here is a sample of the output:

```
MISOSYS Disassembler - Disk Version 2.4  Partial ROM  PAGE  ØØØØ1
ADDR CONTENTS LINE# LABEL  INSTRUCTION      ASCII

               ØØØØ1        ORG  ØØØØH
ØØØØ F3        ØØØØ2 ZØØØØ  DI                s
ØØØ1 AF        ØØØØ3 ZØØØ1  XOR  A             /
ØØØ2 C3153Ø    ØØØØ4        JP   Z3Ø15         C.Ø
ØØØ5 C3Ø4Ø     ØØØØ5 ZØØØ5  JP   Z4ØØØ         C.@
ØØØ8 C3Ø4Ø     ØØØØ6        JP   Z4ØØØ         C.@
ØØØB E1        ØØØØ7 ZØØØB  POP  HL            a
ØØØC E9        ØØØØ8        JP   (HL)          i
ØØØD C3123Ø    ØØØØ9        JP   Z3Ø12         C.Ø
ØØ1Ø C3Ø34Ø    ØØØ1Ø        JP   Z4ØØ3         C.@
ØØ13 C5        ØØØ11 ZØØ13  PUSH BC            E
ØØ14 Ø6Ø1      ØØØ12        LD   B,Ø1H         ..
ØØ16 182E      ØØØ13        JR   ZØØ46         ..
```

MISOSYS
P.O. Box 4848
Alexandria, VA 223Ø3-Ø848