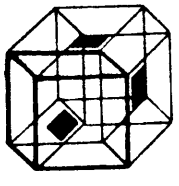
XYZT COMPUTER DIMENSIONS

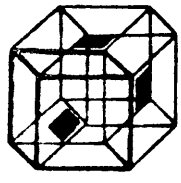
LIBRARY SUPPORT OPTION

The file management system
for the TRS-80 mod. I/III computers.

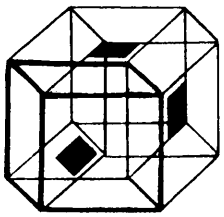


1983

T A B L E O F C O N T E N T S



XYZT COMPUTER DIMENSIONS, INC.



XYZT COMPUTER DIMENSIONS, INC.

2 PENN PLAZA, SUITE 1500, NEW YORK, N.Y. 10121
(212) 244-3100

Special for the NEWDOS/80 ICL and LSO users!

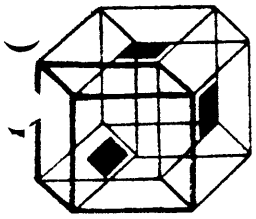
NEWDOS/80 is currently the only operating system with the true MINIDOS facility. Since MINIDOS is very convenient ICL was designed to allow use of procedures in MINIDOS environment.

However MINIDOS does not allow execution of /CMD modules. Therefore LSO users cannot enter MINIDOS and execute LDIR/CMD to display library directory which is often desirable while using wordprocessor, etc.

In order to bypass this obstacle we added new command to ICL - &CMD. This command will load and execute /CMD file. Its Syntax is similar to that of &DOS command. The main advantage of it is that &CMD can be used under MINIDOS. The /CMD module can be executed in the ICL reserved user area without damaging your current environment.

Your diskette contains ICL2C/CMD. (Probably it will be next release - 2-C). This version of ICL contains &CMD command. There are also MDIR/ICL procedure that can be used while in MINIDOS to display library directory instead of LDIR module, and LD/CMD module that will execute in ICL2C reserved area.

We hope you will find this useful.



XYZT COMPUTER DIMENSIONS, INC.

2 PENN PLAZA, SUITE 1500, NEW YORK, N.Y. 10121
(212) 244-3100

Dear ICL user,

XYZT came up with another exciting product - LIBRARY SUPPORT OPTION (LSO). LSO is a file management system which allows you to create and maintain libraries of files or programs.

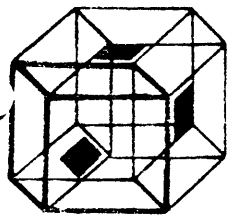
In order to give you enough information about LSO we placed LSO demonstration programs on you ICL disk and enclosed LSO documentation. Besides ICL files you will discover the following files on diskette:

DEMOLSO/CMD - LSO demo program,
CMD/LIB - library containing /CMD modules,
ICL/LIB - library with ICL procedures.

DEMOLSO performs the same as the main LSO module described in the manual but in read-only mode. You can read files from library, rename, kill, copy from the library to disk, etc. However you can not add new files to libraries, since all write functions are suppressed.

The purpose of the DEMOLSO is to give you some real experience. You can use ICL/LIB - it contains the most useful ICL procedures. Note that ICL/LIB contains 9 files and occupies only 4 granules. In a regular system those files would occupy 9 full granules. Savings in disk space are over 50% in this case.

If you decide to purchase LSO, we are ready to ship it to you as soon as you ask for it!



XYZT COMPUTER DIMENSIONS, INC.

2 PENN PLAZA, SUITE 1500, NEW YORK, N.Y. 10121
(212) 244-3100

1. I N T R O D U C T I O N

Welcome to the LSO! This package adds library management functions to the operating systems developed and used with the TRS-80 Mod I/III computers. Now you can combine your files into libraries and use them directly same way you are using regular files.

The advantages are significant:

1. Space economy. Typically the minimal file size is one granule - 5 sectors (1280 bytes). If you have number of small files - such as relocatable files produced by the MICROSOFT compilers, or other small /CMD files, etc. your diskette space is simply wasted, since a granule is allocated for each file even if the file is only hundred bytes long. Under LSO the minimal file size is one sector. As a result of that approximately 5 small files can be stored in the same space as 1 regular file. Plus compress option saves additional space by converting repeating string of the same characters into 3 byte groups - this is important for ASCII files produced by most of word processors.

/REL
/CMD
1 SECTOR
COMPRESS

2. Increased speed. In most cases LSO increases speed of processing. Each library file has its own directory and search for a file in library directory is faster than in diskette directory. Another nice feature - permanently opened libraries (for files such as /CMD or /ICL) saves time on every opening of the file.

directory
permanently opened

3. Ease of file manipulation. With LSO you can copy, kill, backup, move all library files with as little as one command. Just copy library file and all the files contained in the library will be copied. And you can logically group your files - you can create separate libraries to keep all your business letters in one library, personal in another, all BASIC programs in third and so on.

copy
kill
backup

The LSO was tested with the following software products - BASIC, ICL, ELECTRIC PENCIL, M80, L80, EDIT/CMD, SUPERZAP and presumably LSO will work with any software package using standard DOS I/O routines. Its operation is transparent to the DOS, only few new procedures are required to use LSO. The package is ready to run and you can start using it immediately!

All comments and suggestions are welcomed!

2. L S O I N S T A L L A T I O N

The diskette shipped to you contains following files:

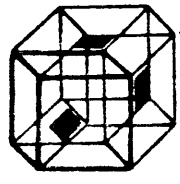
File name	Size	Description
LSO/CMD	4g	Main program.
CMD/LIB	4g	Library of utility modules.
		Includes LCREATE/CMD
		LXTEND/CMD
		LNAME/CMD
		LDIR/CMD
		LSET/CMD
ICL/LIB	3g	** ICL users only **. Library
		containing ICL procedures to
		enhance LSO processing.
		Includes LQDIR/ICL
		LCOPY/ICL

At first you should copy LSO/CMD and CMD/LIB to your diskette. If you are ICL user as well copy ICL/LIB too. After that start LSO - simply enter LSO. The package will load and relocate itself into high memory and return control to DOS. Now you can find out what is inside CMD/LIB. Enter - LDIR CMD. This will display library directory (very similar to the DOS DIR command). As you can see the LDIR/CMD module that was used to display library contents is in this library itself. The rest of utility modules are described later.

This ends installation of the LSO package. We suggest you make a backup copy of the original diskette.

/LIB
LSO
LDIR <->

LSO 2 kinnen FØCA



3. H O W T O U S E L S O

3.1. General information.

What is a library? A Library is a file containing members (smaller files). Same as a diskette contains a number of files and a directory, a library file contains several members and a directory to reference them. In fact a library is like having a microdisk on a diskette.

From the DOS point of view (if it has any) a library file is a regular file. This is obvious. More interesting is the fact that due to the LSO design library members are also treated the same way as regular files. We'll explain here just the idea, all details are given in the technical section of this manual.

Suppose you wish to have a library to keep all (or some) of your BASIC programs in it. Such files have extension /BAS. To do that you create a BAS/LIB library and copy your programs into it. The file MYPROG/BAS will become a member MYPROG in that library, TEST/BAS will become a member TEST and so on. By the logic of LSO operation all members in the library are treated by DOS as regular files with name same as the member's name and extension equal to the name of the library. How this is done? The LSO intercepts all I/O requests. When the system wants to open a file the LSO first checks whether there is a library with the same name as the file extension. If such a library does not exist LSO returns control to the operating system and lets it continue. In case such a library does exist LSO will search this library directory for the specified file and open it. The FCB (file control block) will be marked as a special library FCB so that LSO will know what to do with it when requests for READ or WRITE will be issued for that FCB. Since library FCB has the same structure as a regular FCB, the application programs can use it without problems. As a result of that the LSO operation is transparent to the DOS and to application software as well, which makes LSO easy and convenient to use.

Intercepts Overlay-Control bei 400C: JP F2E3 (RST 5(EF))

3.2. Library creation.

First of all we'll see how to create a library. There are several special utility modules in the LSO package handling maintenance functions, one of them - LCREATE is used to create a library. (All utility modules are fully described in the next section of the manual).

The needed information is the number of files the library will contain and total size of library. Just for example we'll create a small library - 10 sectors size of library and 9 directory entries. To do that enter:

LCREATE BAS/LIB,10,9

and library file BAS/LIB(size 10 sectors, 9 directory entries) will be created on your diskette.

Now we can use this library to store files in it.

3.3. Using library.

Start BASIC and enter some small program, for example following

```
10 ' THIS IS A TEST PROGRAM
20 FOR I=1 TO 10
30 PRINT "MESSAGE ";I;" > I'M RUNNING"
40 NEXT
50 PRINT "MESSAGE ";I  ;" > FINISHED"
```

Save it to disk under the name "TEST/BAS". Then clear it by entering NEW and reload it. You may even try to run it!

Now exit back to DOS and examine directory. The file is nowhere in sight?!! That is right, LSO saved it to the BAS/LIB. You can run the LDIR utility to see what is in the library:

enter LDIR BAS

You will see the following display

```
___   ==> BAS/LIB      BLKS 010/008    DIRS 009/008

TEST     ........  ........  ........  ........  ........

DOS READY
```

Here it is right in the library. You can use regular DOS commands with this file - for example you can make a backup copy of it:

enter **COPY TEST/BAS TEST1/BAS**

and run LDIR again. You will see second member in the library directory - TEST1. And you can always extract file from the library and put it on disk as a regular file -

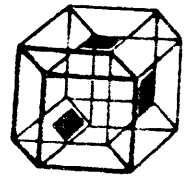
enter **COPY TEST/BAS TEST/AAA**

and file TEST/AAA will appear on regular diskette directory. Similarly you can copy regular diskette files to library. Besides copying you can use other DOS commands such as LIST, PRINT, KILL, LOAD, etc. to process library members.

Several important points.

1. Library files always have priority over regular files - if a corresponding library exists the file will always be searched in the library at first.
2. LSO supports both regular and library files. If a file is not found in the library search will continue thru regular files. It is possible to have some files in library and some on diskette.
3. Regular DOS commands such as COPY, KILL, LOAD, LIST, PRINT... can be used to process library members. Most of software will work with library files without any special procedures.
4. You can force file to be processed as regular by preceding filename with '#' sign. DOS command will reject it since they check commands very carefully, but in most of other software this will work, for example BASIC does not check the file name so meticulously and you can force regular file processing this way.

*) Bei COPY Multi/ICL:3 TST2/ICL:1 beim 2. Filenamen kann vor das # angegeben werden, wenn mehrere ICL/LIB's vorhanden



3.4. Library options.

As if you have not had enough to bruise your mind, here is some more. The LSO gives you special features - permanent open option and compress option. Let's discuss them one at a time.

Permanent open option.

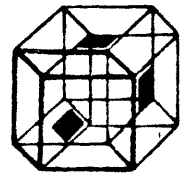
Usually the library is opened during search for the file and close when DOS command or application program (/CMD file) is completed. Since opening the library requires certain time, the whole processing can be optimized by specifying library to be opened permanently. In that case once the library is opened it will remain opened until the next BOOT so it will not be necessary to open it every time. This will save substantial time for frequently used libraries. Note permanently opened libraries should not be on the removable diskettes, since LSO expects the library to be present on disk and therefore if diskette is removed an error can occur.

Permanent open option can be set/reset via LSET utility. For more information refer to the UTILITIES section of the manual. LSET

Compress option.

The library can be designated as a compressed. This means that all members of the library will be compressed when they are written to the library and decompressed when read from the library. Compression deals with the strings of repeated characters(4 or more same characters in a row). Such strings are replaced with a 3 byte group indicating compression, character and length of the string. this operation is entirely invisible for the application programs and can save substantial amount of disk space. It's main purpose is for word processors producing ASCII files also it may be used with any file.

The compress option can be set/reset via LSET utility. For more information see UTILITIES section of this manual.



The LSO package provides several utility modules for maintenance purposes. The LCREATE and LXTEND utilities used to create and increase library size, LSET is used to set library options, LDIR and LNAME replace DOS commands DIR and RENAME which are not applicable to libraries. Every utility is decribed below with examples how to use it and possible limitations. The general rule applies to all utilities - default extension /LIB will be inserted into the library name filespec if no extension is specified.

4.1. LCREATE - create library file.

This utility checks that library to be created does not exist already and then creates a library according to given parameters.

Format:

LCREATE <filespec>,S,D

where: filespec - library file name,
S - total library size in sectors,
D - number of directory entries.

Examples:

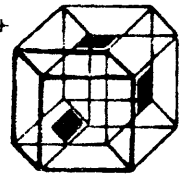
```
LCREATE BAS,20,13
LCREATE ASM:1,30,10
LCREATE TXT/LIB:0,200,25
```

Note: maximum library size is 255 sectors(51 granules), maximum number of directory entries is also 255.

You should keep in mind that directory entries acquire certain space from the total library size. The first 13 entries occupy 1 sector and for every subsequent 16 entries additional sector is allocated.

Below is the table with recommended size/directories.

Number of files in library	Average file size(sectors) 1-2	3-5	5-10	10-20
10	15/13	40/13	80/13	200/13
15-25	40/29	90/29	200/29	255/19
30-40	70/45	160/45	255/45	-
50-80	100/93	255/93	-	-
100-150	255/157	-	-	-



4.2 LXTEND - increase size of library.

This utility checks that library to be extended already exists and then increases library size according to given parameters.

Format:

LXTEND <filespec>,X

where: filespec - library file name,
X - additional size (in sectors)

Examples:

```
LXTEND  BAS,10
LXTEND  ASM:1,30
LXTEND  TXT/LIB:0,5
```

Note: you can increase only library size, not the number of directory entries.

4.3. LSET - set library option(s).

vorher LSET <filespec>, NO

This utility is used to set or reset library options, Note that if the library is opened at the time options are set/reset it has to be closed and then opened again so that new set of options will become effective.

Format:

LSET <filespec>,<option>,<option>

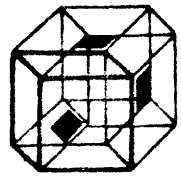
where: filespec - library file name,
option - one of the following
O - permanent open
C - compress
NO - reset O option
NC - reset C option

Examples:

```
LSET    CMD,O
LSET    PCL:1,C
LSET    BAS,C,NO
```

Compress benutzt 03H als Character zum Anzeigen der Kompression

Vorsicht: Beim Herauskopieren mit copy wird die Kompression nicht rückgängig gemacht!



4.4. LDIR - display library directory

This utility displays library directory, available space and directory entries, and library options set for this library.

Format:

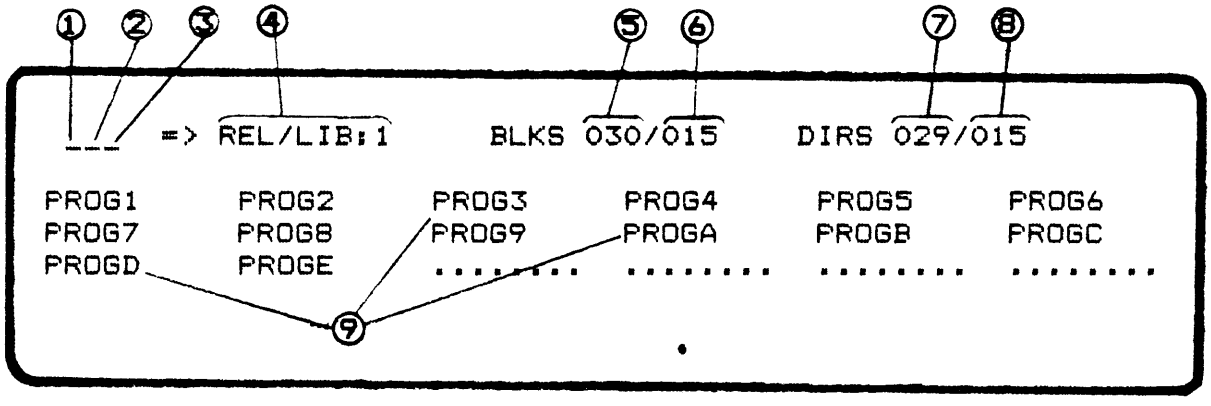
LDIR <filespec>

where: filespec - library file name.

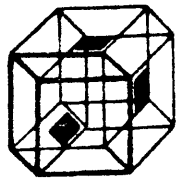
Examples:

```
LDIR   BAS
LDIR   ASM:1
LDIR   TXT/LIB:O
```

Below is the typical display produced by the LDIR



where:
- 1 - diagnostic character. If any errors asterisk will appear in this position.
- 2,3 - option setting. If permanent open O will appear, if compress - C.
- 4 - name of the library.
- 5 - total library size (in sectors).
- 6 - available sectors.
- 7 - total number of directory entries.
- 8 - number of free(unused) directory entries.
- 9 - library members



4.5. LNAME - rename library member

This module performs same as the DOS command RENAME to rename library members. It's syntax is the same as for RENAME only if second file extension is specified it must be same as first.

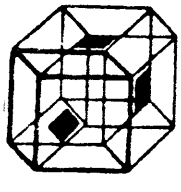
Format:

LNAME <filespec1>,<filespec2>

where: filespec1 - full name of the member to be renamed.
filespec2 - new name for that member.

Examples:

```
LNAME  TESTPROG/BAS,OLDTEST
LNAME  OLDTEST/BAS:1,TEST1/BAS
```

5. S P E C I A L F O R I C L U S E R S

LSO was first conceived as supporting product for ICL since typically ICL procedures are small files and it is quite wastefull to allocate for each of them full granule. Number of users asked for solution and we came up with LSO.

This package contains ICL/LIB with two procedures - LCOPY and LQDIR. LCOPY is used to copy all of the files from one library to another, it is similar to MKILL and RNAME in the ICL package. The main use for LCOPY is to merge libraries, in all other you can simply copy library file instead of copying members one at a time.

LCOPY format:

LCOPY <fromlib> <tolib>

where: fromlib - source library - name:drive#
tolib - destination library - name:drive#

Examples - LCOPY BAS:0 BAS:1
LCOPY BAS:0 OLD:0

The LQDIR procedure loads library directory into queue. It is analog of the QDIR procedure in the ICL package. LQDIR is used by. LCOPY and also can be used by other procedures to provide information about library directory. LQDIR executes LDIR internally to display library directory, receives display output and loads file names into queue in format: "name extension". Extension is taken from the library name. Same as with QDIR five asterisks - "*****" follow last entry.

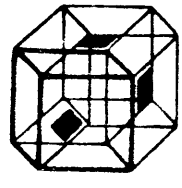
LQDIR format:

LQDIR <libname>

where: libname - library name - name:drive#

Examples: &RUN LQDIR FOR
&RUN LQDIR BAS:1

We suggest you create a library for all your ICL procedures and include these two procedures. Please note - permanent open option may be used with ICL, but not the compress option.



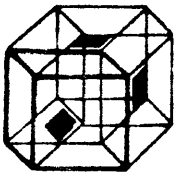
6. I N C O M P A T I B I L I T I E S

For all practical purposes LSO is almost 100% compatible with the regular operation. However due to DOS design there are several minor incompatibilities. Some of them may never be encountered by user.

6.1 "Source and destination is the same file" type of error.
Occurs sometimes when copying file from one library to another, or when copying regular file to/from library. Even though files belong to different libraries DOS displays error. The reason is that files occupy similar directory entries in libraries. DOS checks that files are on the same diskette, use same directory entries(though in different libraries) and decides it is an error.
To bypass this obstacle use another destination file name, then after copy kill first destination file(unsuccessful) and after that rename second(successfull) destination file to desired name.

6.2 "No memory" type of error. May occur if number of simultaneously opened libraries exceeds specified limit. If you assigned too many libraries to be permanently opened LSO will not have enough working storage to process all of them. In that case either change library open option, or start LSO with greater number of libraries to be opened simultaneously - for example, enter LSO 7 when starting LSO to be able to work with up to 7 libraries at the same time. The default number of libraries is 5. Note: every opened library requires 40 bytes.

6.3 Certain errors may happen if you replace diskette containing currently opened library with another diskette. This may happen with permanently opened libraries, or even with regular libraries, for example COPY :1,TEST/BAS TO TEST/BAS may cause error if both diskettes contain BAS/LIB libraries. To avoid this NEVER REMOVE DISKETTE with currently opened library if you will continue work with that library while diskette is removed.



This section describes library file organization and control blocks.

7.1 Library file.

Library file consists of:

Header record	1 sector
Directory sectors	0-15 sectors
Data sectors	1-254 sectors

Number of directory sectors is determined by the number of directory entries in library - first 13 fit into header record, every next 16 require 1 additional sector.

Number of data sectors is equal to the library size minus 1 sector for header minus directory sectors.

7.2. Library header.

The header is the first sector in library file. It contains control information(lib size, number of directory entries, options), bit table of sector allocation, and up to 13 directory entries.

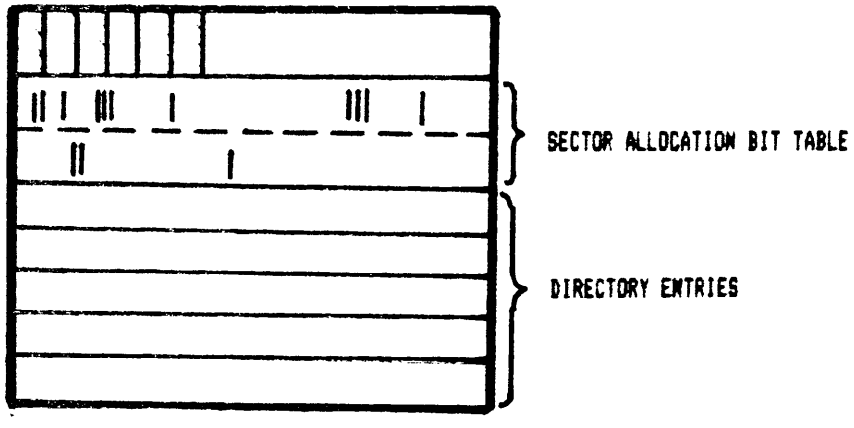
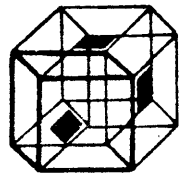


Figure 7.1. Header record.

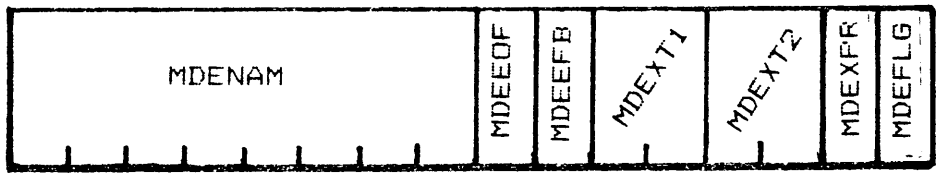
Field	Displ	Lngth	Description
LHDID	00h	1	Library ID - X'BA'
LHDSZ	02h	1	LIBRARY SIZE
LHDDIR	04h	1	Number of directory entries
LHDFLG	05h	1	Library options: bit 7 - compress option bit 6 - permanent open option
LHDSCT	10h	32	Sector allocation table. Each bit in this table represents 1 sector, if on - sector is used



7.3 Directory entries.

Directory sectors contain directory entries. Each entry is 16 bytes long. Directory entry can be primary - first directory entry for the member, or secondary - containing extent information. Primary directory entry contains information for the first two extents of the member. Each secondary entry contains information for 7 additional extents.

Primary directory entry



Secondary directory entry

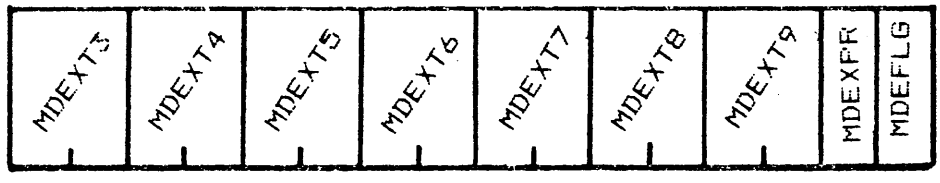
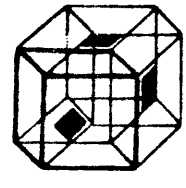


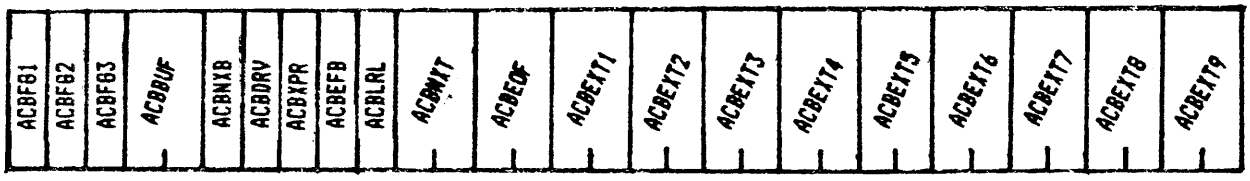
Figure 7.2. Directory entries - primary and secondary.

Field	Displ	Lngth	Description
MDENAM	00h	8	Member's name
MDEEOF	08h	1	Position of last sector
MDEEFB	09h	1	Position of last byte in last sector
MDEXT1	0Ah	2	First file extent: first byte - start of extent second byte - length of extent
MDEXT2	0Ch	2	Second file extent(same as first)
MDEXPR	0Eh	1	Points to the next directory entry (if file has more than 2 extents)
MDEFLG	0Fh	1	Status bits. bit 7 - entry used bit 6 = 0 - primary entry = 1 - secondary entry
MDEXT3... MDEXT9			Extents in the secondary entries.



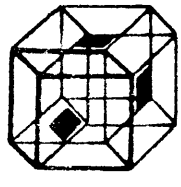
7.4. Access control block - ACB

ACB is the control block the LSO creates when the file is opened. It is same as the regular file FCB with several exceptions.



Field	Displ	Lngth	Description
ACBFG1	00h	1	System flags 1 bit 6 = 0 - regular FCB bit 6 = 1 - LSO member's ACB (other bits are same as for FCB)
ACBFG2	01h	1	System flags 2
ACBFG3	02h	1	System flags 3 bits 0 thru 5 are used
ACBBUF	03h	2	Buffer address
ACBNXB	05h	1	Next byte
ACBDRV	06h	1	Drive #
ACBXPR	07h	1	
ACBEFB	08h		EOF byte
ACBLRL	09h		LRL byte
ACBNXT	0Ah	2	RBA next
ACBEOF	0Ch	2	RBA of the EOF
ACBEXT1	0Eh	2	Extent 1 byte 1 - start of extent byte 2 - length of extent
ACBEXT2	10h	2	Extent 2 (extents 2-9 are same)
ACBEXT3	12h	2	Extent 3 (as extent 1)
ACBEXT4	14h	2	Extent 4
ACBEXT5	16h	2	Extent 5
ACBEXT6	18h	2	Extent 6
ACBEXT7	1Ah	2	Extent 7
ACBEXT8	1Ch	2	Extent 8
ACBEXT9	1Eh	2	Extent 9

Note: all important for application programs fields are same as in regular file FCB - flags, buffer address, next and EOF RBAs, drive number, etc. It was assumed that extent fields are not used by application programs and thus far this appears to be true.



7.5 Library file repair.

This paragraph describes how to interpret the library control information and how to fix(if possible) a library directory in case of errors. This information is mainly for the NEWDOS/80 and DOSPLUS users who have a zap program in their DOS which should be used for library examination and repair, though it may be useful for other DOSs users as well. However you should try to repair library only if you have sufficient understanding of library file organization.
OTHERWISE YOU MAY DESTROY ENTIRE LIBRARY!

First of all let us study the following data - some library header. That's how it will be displayed via SUPERZAP or DISKDUMP.

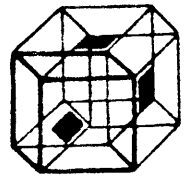
```
00 BA14 000D 0080 0000  0000 0000 0000 0000 ........ ........
10 130F 0000 0000 0000  0000 0000 0000 0000 ........ ........
20 0000 0000 0000 0000  0000 0000 0000 0000 ........ ........
30 5445 5354 0000 0000  00DC 0101 0000 0080 TEST.... ........
40 4F4C 4454 4553 5400  0000 0000 0000 0000 OLDTEST. ........
50 5445 5354 3100 0000  02A5 0401 0801 0680 TEST1... ........
60 5445 5354 3200 0000  0023 0B01 0000 0080 TEST2... .#......
70 1102 0000 0000 0000  0000 0000 0000 0040 ........ .......@
80 0A01 0000 0000 0000  0000 0000 0000 00C0 ........ ........
90 5445 5354 3300 0000  0189 0901 1101 0080 TEST3... ........
A0 0000 0000 0000 0000  0000 0000 0000 0000 ........ ........
B0 0000 0000 0000 0000  0000 0000 0000 0000 ........ ........
C0 0000 0000 0000 0000  0000 0000 0000 0000 ........ ........
D0 0000 0000 0000 0000  0000 0000 0000 0000 ........ ........
E0 0000 0000 0000 0000  0000 0000 0000 0000 ........ ........
F0 0000 0000 0000 0000  0000 0000 0000 0000 ........ ........
```

The first byte is x'BA' - a library ID. Next x'14' is a library size byte - 20 sectors. Then(skip one byte) is a directory size byte - x'0D' - number of directory entries - 13. After that(skip one more byte) is an option byte - x'80'. This byte means that library contains compressed files. That's all for the first line.

The second and the third lines contain a sector allocation bit table. Every byte in the sector bit table indicates status of 8 sectors. The first byte is for sectors 0-7, second for sectors 8-15, third for sectors 16-23 and so on.

There is a special rule to calculate which bit corresponds to a given sector. You must always start with the rightmost bit in the byte and then count to the left. That is so because Z-80 microprocessor used in TRS-80 treats byte in such way that bit 0 is the rightmost and bit 7 is the leftmost bits in the byte.

Therefore first byte in the second line - x'13' means that bits 0,1 and 4 are set, which indicates that sectors 0(header), 1 and 4 are used, while sectors 2,3,5,6 and 7 are available. Second byte in that line - x'0F' shows that sectors 8 thru 11 are used and sectors 12 thru 15 are available.



The whole table indicates that there are 7 sectors used - 0,1,4,8,9,10,11.

The rest of lines show the directory entries(lines 30-F0). Line 30 is a primary directory entry for the file TEST. It is an active directory entry since bit 7 in the last byte of line is on. Extent fields indicate that there is only one sector used by this file - sector 1. There are no secondary directory entries for this file - 15th byte is zero. The last byte in the last(in that case the only) sector in the file is on position x'DC'. That's basically all for that entry.

Next line - 40. Currently this entry is unused - the 16th byte is zero. Examining other bytes of the entry we can guess that it was used for a file OLDTEST which was killed.

Next line - 50. It contains information for the file TEST1. This entry is active and indicates that the file occupies sectors 4 and 8. It also points to the secondary entry - byte 15 contains x'06'. This secondary entry is line 80.(Line 30 is entry number 1, 40 - 2 and so on. Entry 13 is line F0, entry 14 if it existed will be in the next directory sector). This secondary entry shows that the file also occupies sector 10.

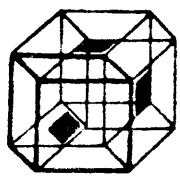
Line 60 is used for the file TEST2. Same as line 30.

Line 70 is currently unused. It looks as if it was used for some secondary directory entry(byte 16), maybe for the killed file OLDTEST.

Line 90 is used for the file TEST3. Same as lines 30 and 60. Lines A0 thru F0 are unused directory entries.

Upon close examination we can see that not everything is perfect with this library. The sector allocation bit table indicates sector 17 is not used, while line 90 shows that this sector is used by the file TEST3. Either the bit table is wrong, or the directory entry. In any case that will cause errors when library will be used.

To fix that problem set bit corresponding to sector 17 on and update header, then examine the file TEST3 to check it is ok, and if yes leave that bit on. Otherwise you might have to copy TEST3 to some other place and then delete TEST3.



Now the main question - how can it happen that a library file contains a bad data. There are several possibilities. First is a power failure. Directory entries and a sector table are not always updated with a single I/O, therefore it is possible that due to a power problem there is a discrepancy.

Second - swapping diskettes while the library is opened. All sorts of problems can happen because of that. Please be carefull and never remove diskette with a library being processed from the drive.

Third - LSO bug. This is quite unlikely, but there is such possibility. If you feel the problem is with the LSO please let us know, we'll try to fix it.

How you can determine that library contains bad data?
The LDIR module performs a very thorough check of the library file before displaying directory information. If any errors are detected an asterisk will be displayed in the left corner of the screen. If you see it then library contains some bad data. Below is the list of errors detected by the LDIR.

1. Library header ID is not x'BA'.
2. Sector is marked as used though it is not used by any file.
3. Sector is marked as unused though it is used by some file.
4. Sector is used by more than one file.
5. Directory entry flag byte is not one of the following:
   - x'80' - primary entry,
   - x'C0' - secondary entry,
   - x'40' - deleted secondary entry,
   - x'00' - unused.
6. The directory sectors are marked as unused.

The LDIR module at first reads and saves sector allocation table, after that it scans the directory and contructs another table according to the information in the directory entries, which is compared to the original to check correctness of the library. You may see the contents of both tables immediately after execution of the LDIR module (using DEBUG) at the following addresses:

original table - x'5A20'
reconstructed table - x'5A40'

Comparing these tables after LDIR can save you substantial time if you are trying to find wrong bit.