LK-80 OPERATOR'S GUIDE

PRELIMINARY

Digital Research, Inc.
P.O. Box 145
37 N. Auburn Avenue
Sierra Madre, CA 91024

213-355-4211

Third Printing: January, 1982

TABLE OF CONTENTS

1. INTRODUCTION TO LK-80

LK-80tm is a linkage editor combining relocatable object (REL) modules into an executable file and optional overlay files. LK-80 is designed for use with the CB-80tm compiler, and also links modules created by a relocatable assembler such as RMACtm. When LK-80 is used with CB-80, a composite program is produced by combining the language's default library with the REL modules.

LK-80 links any program occupying less than 64K bytes of memory unless the length of symbols exhausts the space reserved for the symbol table.

This manual describes version 1 of LK-80 operating with Digital Research's CP/M-80 or MP/M-80tm operating systems. LK-80 is provided with CB-80. The End User Licensing Agreement for CB-80 covers the use of LK-80. Refer to the CB-80 Licensing Guide for information concerning support for LK-80 and the distribution policy for composite programs. If you do not have a Licensing Guide please contact Digital Research at (213)355-4211.

The Licensing Guide explains how to use CBCK to verify your copy of LK-80 is correct and unaltered during disk copying or due to hardware or software failure. It also contains copies of all applicable licensing agreements.

2. OPERATION OF LK-80

This section describes the operation of LK-80. The general form of a LK-80 command line is shown below:

LK80 [<fn>=]<fn.ft>{,<fn.ft>} { ([<fn>=]<fn.ft>{,<fn.ft>}) }

The brackets ([]) denote optional portions of the command. The braces ({}) indicate the enclosed section may be repeated zero or more times. The symbol "fn" indicates a file name without a type extension. The symbol "fn.ft" represents a file name with an optional type extension and optional command toggles.

Each LK-80 command line option is explained in detail in the remainder of this section.

2.1. Linking Modules

LK-80 is executed by typing LK-80 followed by the name of the module to link.

```
LK80 TEST
```

This command links the REL module "TEST.REL" producing an executable file "TEST.COM". A symbol location (SYM) file "TEST.SYM" is also produced. The SYM file may be used with Digital Research's symbolic debugging program SID[tm].

When linking CB-80 programs, LK-80 automatically searchs the default disk for the CB-80 runtime library "CB80.IRL". Any library modules required by the program being linked is combined into the executable module produced by LK-80. The combination of one or more REL files with a language library forms a composite program.

LK-80 prints information on the display regarding the module being linked. The following example shows the results of linking a simple program. The CB-80 program below, "TEST.BAS", can be compiled to produce a REL file "TEST.REL":

```
PRINT "THIS IS A TEST PROGRAM"
PRINT "IT IS USED TO DEMO LK80"
STOP
```

The module "TEST.REL" is then linked using the following command:

```
LK80 TEST
```

The information printed on the display by LK-80 is shown below:

```
A>LK80
-------------------------------------------------
LK80 Version 1.2 Serial No. 123-4567  Copyright (c)
1981 Digital Research, Inc.  All rights reserved
-------------------------------------------------

code size:      1173 (0100-1273)
common size:    0000
data size:      0168 (1280-13E7)
symbol table space remaining:   0A4C
```

The amount of memory allocated to code, common data and local data is shown next. In this example there is no common data. All the values are hexadecimal numbers.

The amount of symbol table space remaining provides an indication of the number of additional symbols that could be added to the modules being linked without running out of symbol table space.

Normally LK-80 produces a COM file with the same name as the REL file. For instance linking the example above results in an executable file "TEST.COM" being placed on the default drive.

```
LK80 PAYROLL=B:PAY_
```

The command above links the module "PAY.REL" from drive B but creates an executable file "PAYROLL.COM" on the default drive.

```
LK80 B:PAYROLL = B:PAY
```

This command produces the same executable file as the previous example but the file "PAYROLL.COM" is placed on the B drive instead of the default drive.

The names of the modules being linked may have type extensions.

```
LK80 A.REL,B.C
```

Regardless of the type extension, LK-80 assumes the file is a REL file unless the type extension is IRL. This means that, in the previous example, the module "B.C" is read assuming it is a relocatable object module. This does not mean the type extension "C" is ignored; it only means that what is in the file is treated as a REL file.

2.2. Linking Multiple REL Files

Multiple REL modules may be combined into one executable file by listing a group of REL modules separated by commas.

```
LK80 A, B, C, D
```

This command links the four REL modules, "A.REL", "B.REL", "C.REL", and "D.REL", to an executable file "A.COM". The first name in the list is used as the name of the COM file.

As many as 40 REL files may be linked at one time by LK-80. However, the total length of the command line is limited to 128 characters. Thus it may be necessary to rename some REL files to shorter names when a large number of files are linked.

The modules are linked in the order of appearance in the LK-80 command. If no drive reference is specified, the files are read from the default drive. However REL files are linked from any drive.

```
LK80 AP,B:APMENUE,A:APSCN
```

When multiple modules are linked together the executable filename may be specified in the command line.

```
LK80 MYPROG = TEST,RTN
```

In this example the modules "TEST.REL" and "RTN.REL" are linked together forming an executable module "MYPROG.COM".

2.3. Producing Overlays

LK-80 produces overlay files loaded and executed by a CB-80 CHAIN statement. The CB-80 Language Manual contains detailed information on using the CHAIN statement.

LK-80 produces overlay (OVL) files preserving variables in COMMON including dynamically created data such as arrays or strings. To place a REL module in an overlay the name of the REL file is placed in parentheses.

```
LK80 A(B)
```

When this command is executed, LK-80 produces two files:

A.COM

B.OVL

The COM file is the root. "B.OVL" is an overlay file loaded only by a CHAIN statement contained in the root "A.COM".

Chaining to an overlay is different than the conventional concept of loading overlays. When the root chains to an overlay the root itself is replaced by the overlay. Likewise when the overlay chains to another overlay or back to the root, the new overlay replaces the currently executing overlay.

CB-80 ensures all library routines are contained in the root. Chaining preserves the libraries used by the overlay files. This reduces the size of overlays and decreases the time required to load an overlay file.

An overlay file returns to the root that loaded it by chaining back to the COM file, or the overlay may load another OVL as long as the second overlay was also linked with the same root.

```
LK80 A(B)(C)
```

This example produces a root "A.COM" and two overlays "B.OVL" and "C.OVL".

When LK-80 produces an overlay it places all the library routines required by the root and an overlay into the root. This minimizes the size of overlays and ensures the same library is not contained in multiple overlays.

Up to 40 overlays can be created by LK-80. However the total number of REL modules linked may not exceed 40. A particular overlay contains multiple REL files.

```
LK80 A(B)(C,D,E)(F)(G)
```

The name of each overlay as well as the name of the root is specified in the command.

```
LK80 A=ROOT(B=OV1)(C=OV2)
```

The command above produces a root "A.COM" and two overlays: "B.OVL" and "C.OVL".

2.4. LK-80 Toggles

Information is passed to LK-80 by placing toggles between square brackets. The command below passes the Q toggle to LK-80:

```
LK80 TEST[Q]
```

The Q toggle causes LK-80 to place symbols beginning with a question mark into the SYM file. A symbol beginning with a question mark is normally used by language developers for library names. Using the Q toggle adds approximately 100 symbols to the SYM file.

If the Q toggle is not specified the SYM file normally contains only the symbols defined in the linked programs.

2.5. LK-80 Error Messages

When LK-80 detects an error it prints a phrase describing the error onto the display. If it is a fatal error control is returned to CP/M.

The following error messages may occur:

```
Unresolved external: <symbol name>
```

The symbol name is defined as an external symbol but is never defined as a public symbol.

```
Out of Directory Space
```

LK-80 ran out of directory space while writing the root or overlay file.

```
Disk Full
```

LK-80 ran out of disk space while writing the root or overlay file.

```
Multiple Definition: <symbol name>
```

The symbol name is defined twice.

Too many overlays

More than 40 overlays were specified in the command line.

Too many modules

More than 40 modules were specified in the command line.

Symbol table overflow

There is not sufficent memory for the symbol table.

Cannot open source file

A source file, specified in the command line, cannot be opened.

3. LINKING WITH ASSEMBLY LANGUAGE

LK-80 links modules produced by the RMAC relocating Macro Assembler with REL files created by CB-80. The same commands explained in the previous section apply. The programmer should be aware that using assembly language routines makes a program machine dependent.

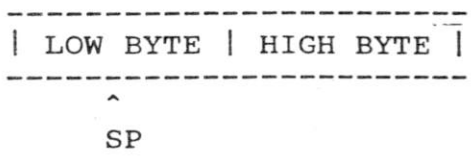
An assembly language module linked with CB-80 must not contain any initialized data. This restriction is necessary because of the runtime environment required by CB-80. Any data that must have initial values are placed in the code segment.

3.1. Passing Parameters

CB-80 passes all parameters to the 8080 hardware stack. The top of, or last entry on, the stack contains the return address. Parameters are stored below the return address. When a routine is called, the first, or left most, parameter is placed on the stack first. Each remaining parameter from left to right is then placed on the stack.

3.1.1. Integer Parameters

Integer numbers are passed onto the hardware stack as sixteen bit signed integers. The integers are stored with the low-order byte in the lower memory address.



3.1.2. Real Parameters

Real numbers are passed onto the hardware stack as eight byte floating point decimal numbers.

Each of the seven mantissa bytes contains two binary coded decimal digits. The left four bits of each byte in the mantissa contain the most significant digit in that byte. The mantissa is normalized so the most significant digit is always non-zero.

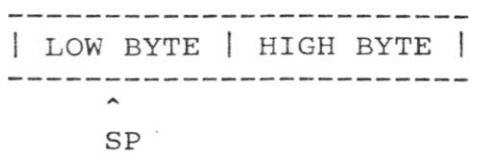
The left most bit of the exponent is the sign of the mantissa. If the bit is a one the mantissa is negative, and if it is a zero the mantissa is positive.

The remaining seven bits of the exponent represent the power of ten multiplier applied to the mantissa. The actual multiplier used is determined by subtracting 64 from the seven low-order bits of the exponent byte.

A number with a value of zero is represented by setting the exponent byte to 0. The mantissa is ignored. All eight bits of the exponent must be zero for the value to be zero.

3.1.3. String Parameters

Strings are passed by placing a pointer to the actual string on the hardware stack. The pointer is an unsigned sixteen bit integer.



If the value of the pointer is zero, the string is a null string. Otherwise the pointer is the address of the string. The first two bytes of the string contain an allocation bit and a fifteen bit string length. The left most bit of the first byte is the allocation bit.

If the allocation bit is a one, the string must be returned to the CB-80 pool of available storage prior to returning from the assembly language routine, and after all references to characters within the string have occurred. String space is returned to CB-80 using the ?RELS library routine. The ?RELS routine is explained in section four.

If the 8080 H and L registers contain the pointer to the string passed as the string parameter, the following assembly

language statements release a string with its allocation bit set:

```
MOV  A,H        ;IF PTR = Ø THEN
ORA  L          ;   NO RELEASE
RZ
MOV  A,M        ;GET HIGH BYTE OF LNG
ORA  A          ;IS ALLOC BIT = 1?
RP              ;IF NOT NO RELEASE
CALL ?RELS      ;RELEASE THE STRING
RET
```

If the allocation bit is a zero, the characters in the string should not be changed since the calling program still has access to the string. If the allocation bit is Ø the string cannot be released.

3.2. Returning Values to CB-80

An assembly language routine returns integer, real or string values to CB-80. Prior to returning to CB-80, all parameters passed onto the stack must be removed and the stack pointer adjusted accordingly.

3.2.1. Returning an Integer

An integer number is returned in registers H and L.

3.2.2. Returning a Real Number

Real numbers are returned by placing a pointer in registers H and L to an eight byte data area containing the real number to be returned.

The number being returned must be stored in the format described above. The H and L registers contain the address of the exponent byte.

3.2.3. Returning a String

Strings are returned by placing a pointer to the string in registers H and L. The string must be allocated by CB-80's dynamic storage management routines. Dynamic storage library routines are explained in section four.

The allocation bit of the returned string should be set to one to ensure the space is reclaimed when no longer required.

4. CB-80 LIBRARY ROUTINES

This section describes CB-80 runtime library routines called from assembly language programs.

4.1. Dynamic Storage Allocation Routines

The CB-80 runtime library provides four routines allowing a programmer to allocate and release memory and to determine the amount of space available for allocation.

The ?GETS routine allocates space. The number of bytes of memory required is placed in registers H and L. The maximum amount of space allocated is 32,762 bytes.

?GETS returns a pointer in registers H and L to a contiguous block of memory. There is no restriction on what is placed in the allocated memory, but the adjacent space at either end of the area may not be modified.

If sufficient space is not available, an "OM" error occurs.

The ?RELS routine releases previously allocated memory. The address of the released space is placed in registers H and L. ?RELS does not return a value.

The ?MFRE routine returns the size of the largest contiguous space currently allocated using the ?GETS routine. The value returned is an unsigned integer put in registers H and L.

The ?IFRE routine returns the total amount of dynamic space currently unallocated. The value returned is an unsigned integer placed in registers H and L.

4.2. Arithmetic Routines

The CB-80 runtime library provides routines for signed integer multiplication and division. The ?IMUL routine multiplies the signed integer in registers D and E by the signed integer in registers H and L. The result is placed in registers H and L.

The ?IDIV routine divides the signed integer in registers D and E by the signed integer in H and L. The result is placed in registers H and L.