Consulair
Linker

CONSULAIR

LINKER LIBRARIAN

VERSION 1.0, AUGUST, 1985

140 Campo Drive
Portola Valley, CA 94025
(415)851-3272

Copyright:
This manual and the software described in it contain proprietary information which is protected by copyright. Under the copyright laws, this manual or the software may not be copied, in whole or part, without written consent of Consulair, except in the normal use of the software or to make a backup copy. The same proprietary and copyright notices must be affixed to any permitted copies as were affixed to the original. This exception does not allow copies to be made for others, whether or not sold. Under the law, copying includes translating into another language or format.

140 Campo Drive
Portola Valley, CA 94025

Limited Warranty on Media and Manuals:
If you discover physical defects in the diskettes on which this software is distributed, or in the documentation distributed with the software, Consulair will replace the media or documentation at no charge to you, provided you return the item(s) to be replaced with proof of purchase to Consulair or an authorized Consulair dealer during the 90-day period after you purchased the software.

All implied warranties on the media and documentation, including implied warranties of merchantability and fitness for a particular purpose, are limited in duration to ninety (90) days from the date of the original retail purchase of this product.

Consualir has tested the software and reviewed the documentaion, but Consulair makes not warranty or representation, either express or implied, with respect to this software or documentation, its quality, performance, merchantability, or fitness for a particular purpose. As a result, this software and documentation is sold "as is," and you, the purchaser are assuming the entire risk as to its quality and performance.

In no event will Consulair be liable for direct, indirect, special, incidental, or consequential damages resulting from any defect in the software, its manuals or any additional documentation, even if advised of the possibility of such damages. In particular Consulair shall have no liability for any programs or data stored in or used with Consulair products, including the costs of recovering such programs or data.

The warranty and remedies set forth above are exclusive and replace all others oral or written, express or implied. No Consulair dealer, agent or employee is authorized to make any modification, extension, or additions to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or conswquential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Support:
Contact your dealer first for assistance in installing or using the Software. After you return the Registration Card, we will provide you reasonable telephone or electronic mail support, during our published customer support hours, on questions not answered by your dealer or the Linker Manual. If you licensed the Product directly from us and have returned your Registration Card, then contact us directly for assistance. We will not be responsible for user support on products other than Consulair products or on programs written using Consulair products.

Again thanks for choosing Consulair products.

Please record your serial number(s) and include them on any correspondence.

CONTENTS

VERSION	REVISION HISTORY	PRINT DATE
1.0	Original Printing	August, 1985

The information in this manual is subject to change without notice.

INTRODUCTION

The Consulair Linker for the Macintosh is an optimizing linker and library manager. It is fully compatible with the Apple Macintosh 68000 Development System (MDS) and the Consulair "Mac C™" compiler. The Linker removes unused code from the program being linked ("dead code stripping"), and can be used to produce pure "double-clickable" Macintosh applications, resource files, driver and "PROC" resources, and plain M68000 code (suitable for use on any machine with a Motorola 68000 processor). It can merge existing resources with the program being linked, so that the resultant application is complete without a separate resource merging operation. The symbol map file produced by the linker is used by MacDB, the MDS two Macintosh debugger, to provide fully symbolic program debugging. A complete listing of the final application code (annotated by the assembler source code) can be produced if required.

The library manager portion of the linker allows the creation, modification, and listing of library files. Since library files are a superset of MDS relocatable binary files, library files may be combined to produce new libraries. A full set of library management operations makes the Consulair Linker a powerful library management tool.

DESCRIPTION

The linker gets its commands from a Link Control File, which is a text file created by the user (normally with Edit). There is a unique Link Control File for each application to be linked and each library to be created. It contains the names of the relocatable binary files and libraries to be used in the linking process, and the control statements which tell the linker what to do. Given a Link Control File which contains link commands and the names of libraries and relocatable binary files, the linker reads all of the files and, after resolving all external symbol references, combines the relocatable binary files and the library modules actually referenced by the program into an application. Unused code sections are not included in the application. A code section corresponds to a function in Consulair "Mac C", a library module from a library file, or a relocatable binary file from the MDS assembler.

After the code and resource sections of an application are linked (where resources are provided in relocatable binary format), the linker merges extant resources from other files into the final application.

THE PIECES OF AN APPLICATION

A file on the Macintosh contains two forks, a data fork and a resource fork. The code for an application is stored in the resource fork as a collection of resources of type 'CODE'. Two of these resources, ID = 0 and ID = 255, contain special information relating to the application. Code Resource 0 contains the jump table for the

application, and Code resource 255 contains the global data initialization segment. The remaining 'CODE' resources contain the actual instructions for the application, where each resource is a different segment in the application. Normal code segments are absolutely limited to 65536 bytes by the segment loader. Segments greater than 32768 bytes in size may require some care in constructing, since PC relative jumps and function calls are limited to an addressing range of -32768 to +32766 bytes.

The jump table (Code segment 0) is built by the linker to resolve references between code segments. The fact that this is done entirely in the linker, means that segmentation of a program is done at link time, and need not be a consideration in the source code (other than to force the unloading of unused segments with the UnloadSeg trap). See the Segment Loader section of Inside Macintosh for details.

The global data initialization segment contains encrypted information which is used by the Consulair Mac C library during run-time initialization. It is not necessary to know anything about this segment other than the fact that it is an important part of an application written with Consulair Mac C.

The gobal data for an application (static and extern variables in "C", and storage defined with the "DS" directive in the Apple 68000 Assembler) is stored below the address register A5, and is normally addressed as an offset from A5. The linker locates the global data area so that it ends at the byte just before the data global area address, which is placed at -$100 by default to allow room for the quickdraw globals which are normally placed immediately below A5. A link control directive allows the global data address to be set to something else. The linker calculates the size of the global data area, and stores it into a special location in the jump table segment, which is subsequently used by the Macintosh Segment Loader to allocate space for it when the application is run. The practical limit to the size of the global data area is approximately 32K, since a negative A5 offset (used for addressing global variables) can be no greater than -32768.

RUNNING THE LINKER

The Linker may be run from the Finder, from the Exec (or Make), or from Edit. When run from Edit or Exec, it may be on any mounted volume, so long as the name of that volume is reflected in the Edit Menu or the Exec Job file. If a link control file is specified when the linker is started (explicitly in an Exec Job file, as the frontmost file in Edit, or as a multiple icon selection in the Finder), it will use that file as input, and perform the specified operations. If a link control file is not specified, a Standard File dialog box will be offered to select a link control file.

If the linker finishes without errors, it returns control to the calling environment (the Finder, Exec or Edit). If there are errors, an error file is generated and control is passed to Edit or the application specified in an Exec Job file.

THE LINK CONTROL FILE

The link control file is a standard text file (normally created by Edit), which contains commands directing the linker, and the names of files to be linked. It must have the extension ".link". Each command is on a separate line, and lines beginning with the character ';' are comment lines. Any line which is not recognizable as a comment or a linker directive is assumed to be the name of a relocatable binary file (which may, in turn, be a library file). Thus file names may contain any legal Macintosh file name character, and may begin with any legal character other than the characters "! < [ ] ( ) $ / ;", which are used as linker directives in the MDS, and double quotes, which are used to enclose a file name when a possible ambiguity exists. File names are not required to have the ".Rel" extension in the link control, e.g. the relocatable binary file "Foo.Rel" may be referred to simply as "Foo". A link control file may contain either library management commands or application link commands, but not both. The name of the application produced by a link control file is given the "root" name of the link control file by default, e.g. the link control file "Foo.Link" produces the application named "Foo".

LINK COMMANDS

These commands are used when linking an application. Any line which is cannot be interpreted as a command or a comment is asummed to be the name of a ".Rel" file, which may either be a library file or a file produced by the MDS or Consulair Mac C compiler.

; any text	Comment Line
/Start label	Identify starting location for application. "label" must be an externally defined label in the application (use the XDEF directive in the Assembler). If a start label is not specified, the application will use location 0.
/Undefined	Allow unresolved external symbol references in the linked application.

/NoUndefined	Do not allow unresolved external symbols. [Default]
/Strip	Remove unused code from the program being linked. [Default]
/NoStrip	Do not remove unused code from the program being linked.
/Output <filename>	Specify a name for the output file.
/Type '<Type>' '<Creator>'	Set the type and creator of the output file. <Type> and <Creator> are both 4 character strings [Default = 'APPL', 0]
/Globals -$<number>	Specify data global address. Note that this specifies the end of the area as an offset from A5, and it must be a negative number (even if it is -0). [Default = -$100]
/Bundle	Set the bundle bit in the Finder Byte of the output file.
/Resources	Stop linking the normal code section of the application and begin linking resources. The first file following this directive MUST begin with a RESOURCE pseudo-op (in the original ASM file). All files following this directive in the link control file will be placed in specific resources as directed by the RESOURCE pseudo-ops in the ASM source files.
/Include <filename>	Includes all resources from the extant resource file <filename>.

/Data	Stop linking code or resource section of application, and begin placing output into the data fork. All output will go into the data fork without any segmentation or other control information.
/Segment	Start a new segment
/Animate	Display the contents of each library while linking. [Default]
/NoAnimate	Do not display the contents of the libraries while linking.
/End	End of Link Control File

Map File Controls

/References	Include all symbol references in Map file.
/NoReferences	Do not list symbol references. [Default]
/Locals	List local labels in Map file.
/NoLocals	Do not list local labels. [Default]
/Code	Start code listing in Map file. Include ASM source if file was assembled with .VERBOSE option.
/NoCode	Stop code listing in Map File. [Default].

Duplicates for certain commands are provided for backwards compatibility with the MDS Linker. These are:

MDS Command	Consulair Linker Command
!	/Start
/UndefOK	/Undefined
/NoUndef	/NoUndefined
/Verbose	/References
/NoVerbose	/NoReferences
$	/End
(	/Locals
)	/NoLocals
[	/Code
]	/NoCode

[	/Code
]	/NoCode

Library Commands

A library module (LM) is the building block used in library management. A relocatable binary file (".Rel" file) contains at least one library module, whose name is the same as the file name root. It may contain more, created either through library management or assembler pseudo-ops (not available in the MDS assembler). Unused code is removed on the basis of Library Modules. All Library Modules have names, which must be unique within a library file.

/Library <filename>	Identify a library link control file. <filename> is the name of the library while be modified or created by the operations specified by this control file. This directive must be the first non-comment line.
/Create	This will create a new library with the name specified in the /Library directive. Any previous contents of the file will be destroyed. This command must precede any library file names.
/Delete <LM name >	Delete the named library module from the library file.
/Replace	Allows replacement of identically named modules in the library. When this option is selected, duplicate module names will not be flagged as errors. The most recent module will replace the previous one.
/NoReplace	Flag duplicate module names as errors. [Default]
/Contents	List library modules and labels from the current library file onto Map file.

/Rename <old name> to <new name>
Changes the name of the external label named <old name> to <new name>.

<filename>	Add all of the Library Modules contained within the specified file. If the library already contains Library Modules with the same names, they will be replaced by the new ones.

APPENDIX A:
QUICK REFERENCE

Linker

Setting the program start.

/Start <label>

<label> must be an externally defined label in one of the input files. For Mac C programs, it is normally Start, AltStart, or QuickStart.

Naming the Output File

/Output <filename>

Specifying a relocatable binary or library file

<filename>

Files are linked into the application in the order that they are listed in the link control file.

If <filename> identifies a relocatable binary file, it is linked into the program. If it is a library file, the library modules in the file are linked.

Using Segments

/Segment

Tells the Linker to begin a new segment at this point in the application. All files which are listed after this line in the link control file will be placed in the new segment (until another /Segment directive is encountered, or the end if the link control file is reached.

Finding Undefined Symbol References

/References

This directive causes the linker to list ALL references to global variables, whether they are defined or not, in the Map file. It is useful for finding which modules reference a symbol.

Getting a complete program listing

/References
/Locals
/Code

Files to be linked should have been assembled with the .VERBOSE option in the assembler source. This set of directives will produce an annotated listing of the application in the Map file.

Including an existing resource.

/Include <filename>

Looks in the extant resource file <filename> for resources, and includes them into the application being linked.

Allowing Undefined Symbol references

/Undefined

Tells the linker to produce an executable application even if there are undefined external symbols.

Producing non-Macintosh output

/Data

Any code linked after this directive will be placed in the data fork of the output file as pure M68000 object code. The code listing in the Map file will reflect its contents.

Librarian

Building a New Library

```
/L:ibrary <filename>
/Create
<filename 1>
<filename 2>
        .
        .
        .
<filename n>
```

Builds a library of the name <filename> from the modules contained in <filename 1> through <filename n>.

Finding out what's in an existing library

```
/Library <filename>
/Contents
```

Lists the modules and their global symbols from the library file <filename> onto the map file, <filename>.Map.

Adding to an existing library

```
/Library <filename>
<filename 1>
<filename 2>
        .
        .
        .
<filename n>
```

Adds the module(s) in <filename 1> through <filename n> to the library named <filename>.

Replacing part of an existing library

```
/Library <filename>
/Delete <modulename>
<filename>
```

Replaces <modulename> with the modules in <filename>.

Reducing an existing library

/Library <filename>
/Delete <modulename>

Removes <modulename> from the library.

APPENDIX B:
ERROR MESSAGES

A RELATIVE ADDRESS WAS TOO BIG AT LABEL <LABEL>
The legal range is from NNN to NNN.
The actual value was NNN.

This error message is produced when the linker encounters a PC relative address which cannot be contained in the opcode field allocated to it. It typically results from a BSR, BRA, BSR.S, or BRA.S instruction. When you get this message, you need to change a short branch to a long (if it is a .S instruction), re-organize your link control file so all PC relative references are within 32K bytes, or segment your program into smaller pieces (using the /Segment directive).

COULD NOT ADD RESOURCE TO FILE
An input/output failure occurred when attempting to add a resource to an application with an Include directive.

COULD NOT OPEN FILE <FILE NAME>
The indicated file could not be opened.

COULD NOT WRITE LIBRARY FILE
An input/output failure occurred when attempting to output a library file.

DATA SEGMENT LARGER THAN 32K
The segment required for storage of global and static data has grown to be larger than 32K bytes. This is illegal in applications using A5 as a base register for addressing global data. (This includes applications produced with Consulair Mac C.)

DISK FULL
The diskette being used for output is full.

DISK I/O ERROR
An input/output error has been reported by the Macintosh operating system.

DISK WRITE PROTECTED
The write protect tab on the diskette is open.

FILE LOCKED
The output file is open by another application.

ILLEGAL LIBRARY FILE NAME
The file name used in a Library directive is illegal.

INPUT FILE NOT FOUND <FILE NAME>
The indicated file was specified as an input file in the link control file, but could not be found.

INVALID NUMBER
A number has been scanned which has an invalid syntax.

I/O MEMORY ERROR
There was insufficient memory to complete an input/output operation.

LABEL USED IN /START NOT DEFINED: <NAME>
The indicated label was specified as the starting location for the application in the link control file, but was not defined in any of the linked modules.

LIBRARY MODULE NAME TOO LONG
A library module name is longer than 255 characters.

LINK CONTROL FILE ERROR: <ERROR>
An error has been sensed in processing a link control file.
<error> will be one of the following:
Invalid / Command
Invalid Starting Label
Invalid File Name
Line Does Not End Properly
Resources cannot be segmented
Input File Not Found
Missing label in Rename
Missing = in Rename
Illegal include file name
File Name Too Long

MISSING LIBRARY MODULE: <NAME>
The indicated library module was referenced in a link control directive, and could not be found in the library or relocatable binary input files.

MULTIPLY DEFINED LIBRARY MODULE: <NAME>
The indicated library module was defined more than once.

NUMBER TOO LONG
A number is longer that 252 characters.

OUT OF MEMORY
The Linker has run out of available memory. If you are running with a Ram Disk, try making it smaller.

RESOURCES CANNOT BE SEGMENTED.
A /Segment directive has been found following /Resources in the link control file.

RESOURCE PSEUDO OP USED IN INPUT FILE BEFORE /RESOURCES DIRECTIVE.
A module has been named in the link control file, which contains a /Resource directive in the assembler source, and which does not follow a /Resources directive in the link control file.

RESOURCE TOO BIG <TYPE> <ID> [<NAME>]
The indicated resource could not be included in the final application produced by the linker because insufficient memory was available to hold it. If you get this error and your are using a Ram disk, try reducing the size of the Ram disk.

STRING TOO LONG
A string is longer that 252 characters.

SYMBOL MULTIPLY DEFINED: <NAME>
The label <Name> was defined in more than one link module.

SYMBOL NOT DEFINED: <NAME> REFERENCED IN <FILE NAME> <MODULE NAME>
None of the linked modules contained a definition for the external symbol <name>, which was referenced in the indicated modules.

SYMBOL TOO LONG
A symbol is longer that 252 characters.

UNKOWN I/O ERROR
An unspecified input or output error has been reported by the Macintosh operating system.

VOLUME LOCKED
The volume containing an output file is locked.

APPENDIX C:
STANDARD FILE EXTENSIONS

<filename>.Rel	A Relocatable Binary File	[Input].
<filename>.Lerr	A Linker Error File	[Output].
<filename>.Link	A Link Control File	[Input].
<filename>.Map	A Link Map FIle	[Output].

APPENDIX D:
EXAMPLES

(1) Create a library named "Lib1" from the relocatable binary files "file1" and "file2"

```
/Library Lib1
/Create
file1
file2
/End
```

(2) Link the application "Prog" with the library "Lib1", using the label "start" as the starting location.

```
/Start start
Lib1
Prog
/End
```

(3) Add "file3" to the library "lib1". "file3" contains an external label "Start", so we will rename this to be "file3_Start" to avoid conflict with "Prog".

```
/Library Lib1
file3
/Rename Start file3_Start
/end
```

(4) Link "Prog" with the new lib1, and turn off animation to speed things up.

```
/NoAnimate
/Start start
Lib1
Prog
/End
```

Linker Features:

- COMPATIBLE — Designed for the MDS by the creators of the MDS.
- EFFICIENT — Optimizes the final stage of your development process. Creates smaller applications by eliminating unused code.
- SAVES TIME — Improves project management by creating libraries of shared code and standard functions.
- FLEXIBLE — Create new libraries from existing object files (including software supplement code).
- RESOURCE MERGING — Directly link Macintosh resources without the need of RMaker or RMover.

Works With:

- Any Macintosh (128, 512, XL)
- Consulair's Mac C/Mac C Toolkit
- Apple's MDS
- External disk drive, Hard disks, Ram disks, Extended memory

unused code.
- SAVES TIME — Improves project management by creating libraries of shared code and standard functions.
- FLEXIBLE — Create new libraries from existing object files (including software supplement code).
- RESOURCE MERGING — Directly link Macintosh resources without the need of RMaker or RMover.

Works With:

- Any Macintosh (128, 512, XL)
- Consulair's Mac C/Mac C Toolkit
- Apple's MDS