UNIVERSITY OF EDINBURGH
Edinburgh Regional Computing Centre

User Note 96

(September 1986)

Title:

Modifying Object Files on EMAS-3

Author: Sandy Shaw

Contact: Advisory Service

Software Support Category: See Note 15

Synopsis

This Note describes how to use the command MODIFY to alter attributes of an EMAS-3 object file. It is likely to be useful where large object files with many entries and references are handled. The reader is assumed to be conversant with the principles of EMAS program loading as described in the EMAS-3 User's Guide and User Note 32.

Keywords

bound file, data entry, object file, procedure entry

Edinburgh Regional Computing Centre

James Clerk Maxwell Building, The King's Buildings, Mayfield Road, Edinburgh, EH9 3JZ. Telephone 031-667 1081

The following types of operation are available:

- changing the name of procedure or data entries and references.
- suppressing procedure and data entries.
- changing static procedure references to dynamic and vice versa.
- merging the code and shareable symbol tables areas or the GLA and unshared symbol tables areas.
- binding the object file.

The BIND operation is intended to alleviate the problem of lengthy loading time for large packages. Essentially, it processes an object file to produce a module which can subsequently be loaded at minimal cost. To do this it assumes fixed sites for the code and GLA, and hence can satisfy all relocation requests. It also resolves all cross references in the object file.

The program is called by:

Command: `MODIFY object file [,output object file] [,report file]`

The input (and output) object file may be a pd file member.

If the second parameter is omitted, the modified object file overwrites the input object file.

If the third parameter is omitted, the report, which contains details of the operations performed, is sent to T#MODLIST. Fault messages are sent to the terminal and do not appear in the report.

The program issues the prompt *Operation:*, the replies to which are given below. Most operations require further input and issue appropriate prompts.

The *Operation:* prompt is reissued until the reply `CLOSE` is given. This terminates the modification and causes the output file to be generated.

All input may be given in upper or lower case; spaces are not significant.

In any of the commands which operate on entries or references, a mask may be specified (similar to that used in the Subsystem FILES command) rather than an explicit name, e.g.

Operation: `SUPPRESS` *Proc ent pair:* `*BR*`	- suppresses all entries containing `BR`
Operation: `RENAME` *Proc ent pair:* `NEW*, OLD*` *Proc ent pair:* `*, *Z` *Proc ent pair:* `.END`	- changes all entries beginning with `NEW` to begin with `OLD`, appends `Z` to every entry
Operation: `REDIRECT` *Proc ent pair:* `S*Y, S*` *Proc ent pair:* `.END`	- removes `Y` from every procedure reference starting with `S` and ending in `Y`

The parameters for the first five operations consist of one or more lines, each containing a pair of names separated by a comma, the list being terminated by `.END`, e.g.

Operation: RENAME
Proc ent pair: DREAD, READDATA
Proc ent pair: DPRINT, PRINTDATA
Proc ent pair: .END
Operation:

1. **RENAME**

   Changes the names of procedure entries.

   Parameters: procedure entry, newname

2. **REDIRECT**

   Changes the names of procedure references.

   Parameters: procedure reference, newname

3. **RENAME DATA**

   Changes the names of data entries.

   Parameters: data reference, newname

4. **REDIRECT DATA**

   Changes the names of data references.

   Parameters: data reference, newname

5. **ALIAS**

   Allows a copy of a procedure entry to be made and given a different name.

   Parameters: procedure entry, copy name

The parameters for operations 6-15 below consist of a list of items separated by commas, or the keyword .ALL. The list is terminated by newline unless the last character is a comma, in which case it continues on the following line. The remaining operations (16-18) take no parameters, except BIND which takes two optional parameters.

Examples:

Operation: MAKEDYNAMIC
Proc ref list: A1,A2,A3,
Proc ref list: A4
Operation:

Operation: SUPPRESS DATA
Proc ent list: .ALL
Operation: BIND
Operation:

6. **MAKE DYNAMIC**

Makes static procedure references dynamic.

Parameters: static procedure reference list *or* .ALL

7. **MAKE STATIC**

Makes dynamic procedure references static.

Parameters: dynamic procedure reference list *or* .ALL

8. **SUPPRESS**

This operation allows procedure entries to be suppressed. Other operations such as SATISFY REFS will still be able to find and use a suppressed entry - the only effect is that the suppressed entry will not appear in the load data for the output object file. Note that a main program entry cannot be suppressed.

Parameters: procedure entry list *or* .ALL

9. **RETAIN**

This cancels a SUPPRESS request. Since by default all entries are retained, this operation is provided so that SUPPRESS with parameter .ALL can be followed by explicit RETAIN calls for the procedure entries which are actually required. Note that an object file must have at least one main entry, procedure entry or data entry.

Parameters: procedure entry list *or* .ALL

10. **SUPPRESS DATA**

This command allows data entries to be suppressed (see SUPPRESS).

Parameters: data entry list *or* .ALL

11. **RETAIN DATA**

This cancels a SUPPRESS DATA request (see RETAIN).

Parameters: data entry list *or* .ALL

12. **SATISFY REFS**

For each procedure reference given, a search is made of the list of procedure entries. If a matching entry is found, then the reference is satisfied and removed from the load data. This operation is performed as a side effect of BIND, and is hence redundant if BIND is also called.

Parameters: external procedure reference list *or* .ALL

13. SATISFY DATA

For each data reference given, a search is made of the list of data entries. If a matching data entry is found, then the data reference is satisfied and removed from the load data. This operation is performed as a side effect of BIND, and is hence redundant if BIND is also called.

Parameters: data reference list *or* .ALL

14. NO CHECK

When a procedure reference is satisfied, a check is normally made that the number and size of the parameters agree with those recorded for the procedure entry. This operation suppresses checking of all references to the named procedure entries.

Parameters: procedure entry list *or* .ALL

15. COMMON REF

When BIND is used, all data references to COMMON areas are satisfied by MODIFY, and the necessary relocations performed. This operation instructs MODIFY to leave such references intact, so that the loader can subsequently process them in the normal way. This is required where a BOUND program is to access COMMON created by another program.

Parameters: COMMON definition list *or* .ALL

16. BIND

This command performs pre-loading operations on the object file so that it can be loaded subsequently at reduced cost. To do this it uses fixed sites for the code and GLA, and hence can satisfy all relocation requests. It also satisfies procedure and data cross-references and COMMON definitions. To achieve the greatest savings in load time, the BIND command should be preceded by calls on the commands to suppress procedure and data entries. BIND automatically attempts to satisfy all procedure and data cross-references, so the explicit commands for these operations need not be called where BIND is used.

Bind may be supplied with two parameters to specify which fixed sites (segment numbers) are to be used for the code and GLA. These should be in the range 150-230.

If no parameters are supplied, BIND will select a segment number at random in the range 190-230 for the code. The GLA is fixed at the next free segment following the code.

Although a bound file is loaded most efficiently at its fixed sites, if either of these sites is already in use (because some other file is connected there) loading can still proceed. The Subsystem loader will perform the necessary rebinding when a preferred site is not available. However the restriction will remain that once bound, an object file cannot subsequently be modified or linked.

Further advice on the use of bound files is given in User Note 32.

Parameters: null *or* code segment, GLA segment

17. NO HISTORY

This operation causes the history information, displayed when the file is analysed, to be discarded.

Parameters: None

18. CLOSE

This terminates the user input. A confirmatory message is printed if the modification is completed successfully.

Changes from EMAS 2900

Modify has changed its behaviour from the EMAS 2900 version in two areas. Firstly, the treatment of COMMON areas has been simplified, and the three commands which dealt with this replaced by one, COMMON REF. When a file is bound, the user allows MODIFY to process all references to COMMON, or uses COMMON REF to cause COMMON references to be processed in the normal way at load time.

The FUSE commands have been removed. These had the effect of concatenating certain object file areas to improve paging behaviour. However, these operations are now performed automatically by LINK.