Title:

DOMUS Linkage Editor

Revision 01

A/S REGNECENTRALEN

RC SYSTEM LIBRARY: FALKONERALLE 1 DK-2000 COPENHAGEN F

RCSL No: 43-GL 5319
Edition: October 1977
Author: Marie Louise Møller

Keywords: DOMUS, Macro Assembler, Linkage Editor.

Abstract:

This manual describes the linkage editor for the disc operating system DOMUS for RC3600 line of computers.

Printed by A/S Regnecentralen, Copenhagen

CONTENTS

FORWORD

The purpose of a linkage editor is to handle external references between relocatable binary modules. The DOMUS Linkage Editor (LINK) takes a number of relocatable binary modules produced by the DOMUS Macro Assembler [3] and outputs a single program in relocatable binary or absolute binary form, having filled in inter module references and relocated appropriately.

The input modules are taken from disc files. When linking a disc work file named '.INK' is used. This file is removed automatically when the linking has finished.

1. PARAMETER FORMAT. 1.

The format of a call of the Linkage Editor is as described below.

Call:

LINK OUT.<outfile> LOG.<logfile> TITLE.<title>
ENTRY.<integer> CHECK.<boolean> FORM. <binspec>
IN.<infile>+

where

<outfile> is the name of the file or entry to which the binary code is output. This parameter must always be specified. If not, error message number 200 NOT ENOUGH ARGUMENTS will be displayed.

<logfile> is the name of the file or entry to which the log information is output. If not specified no log is produced.

<title> is the title of the output module. Only relevant if parameter FORM is R or P. If not specified, the title is MAIN.

<integer> is an integer specifying the maximal number of entries defined in the input modules. The default value is 255.

<boolean> is either YES or NO. It specifies if check of location overwrite is wanted. The default value is YES.

NOTE: If the parameter is NO and overwrite occurs in connection with a normal external the linkage editor may loop.

<binspec> is an identifier specifying the form of the binary output. Only the first letter is significant and only one of the following letters is allowed.

R: Relocatable binary output.

A: Absolute binary output. Zero relocatable code is output starting in word 50_8 and normal relocatable code is output to word 400_8 and on.

B: Basic system. This is absolute binary output where start addresses from the input modules are inserted from word 402_8 and on in the order they are met, and the sequence is terminated by -1. The start address for the output is 377_8.

P: Paged program. When loading a paged program some restrictions concerning the order of the normal relocatable code and the virtual code should be fulfilled [4]. The binary output meets these demands.

<infile> is the name of a file containing one relocatable binary module. Only disc files are accepted as input. The maximal number of input files is 60. The modules are linked in the order specified.

Default:

```
LINK OUT.<0 LOG.<0> TITLE.MAIN ENTRY.255:
:CHECK.YES FORM.R IN.<0>
```

1.1 Examples of calls. 1.1

Link the modules from the files IN1, IN2, and IN3 and produce relocatable binary output with title TEST on file BIN. The entry $LPT describing the lineprinter driver exists on the disc, and the log is printed on the lineprinter.

Check of overwriting is performed and the number of entries does not exceed 255.

```
LINK BIN $LPT TEST IN.IN1 IN2 IN3
```

Link the modules from TEXT1 and TEXT2 and output absolute binary on file ABSB1. The entry $TTY describing the teletype driver exists on the disc. The log is listed on the console.

```
LINK ABSB1 $TTY FORM.A TEXT1 TEXT2
```

Link the modules from files TEXT1 and TEXT2 and produce relocatable binary output on file PROG. Check of overwriting is not wanted and the number of entries exceeds 255 but not 350. The title of the output module is MAIN and no log is wanted.

```
LINK PROG ENTRY.350 NO IN. TEXT1 TEXT2
```

2. THE LOG.

This chapter describes the information output if a log device or file is specified.

The linking is carried out in three passes. In pass 1 the syntax of the input modules is checked and the entries defined are inserted in a symbol table. For each module the title is logged, followed by a listing of the names and values of all entries defined in the module. In addition all error messages found in pass 1, if any, are logged. After each module the following values may be logged if relevant for that module:

SADR NNNNNN : Logged if the module has a start address. NNNNNN is the start address in octal.

ZMAX NNNNNN : Logged if the module contains page zero relocatable code. NNNNNN is an octal number specifying the last page zero relocatable address used.

NMAX NNNNNN : Logged if the module contains normal relocatable code. NNNNNN is an octal number specifying the last normal relocatable address used.

In pass 2 the values of external references are inserted. Again the title of each module is logged followed by error messages found in pass 2, if any.

In pass 3 the binary code is output.

After each location value the relocation property is indicated by one of the following symbols: 2.

space	absolute
'	normal relocatable
''	normal byte relocatable
-	page zero relocatable
=	page zero byte relocatable.

3. ERROR MESSAGES.

The Linkage Editor may produce two kinds of error messages; the numbered DOMUS error messages displayed on the console, and the error messages occurring on the log, indicating errors in the relocatable binary input modules.

The DOMUS error messages are listed in the DOMUS Manual. Only the following special error messages from the Linkage Editor should be mentioned here:

0250 *** CHECKSUM ERROR, FILE XXXXX

indicates a checksum error in the specified input file.

0251 *** OVERFLOW IN ENTRY TABLE

indicates that the input modules contain more entry definitions than specified in the parameter ENTRY.

0252 *** FATAL ERROR, LINKAGE EDITOR

indicates that fatal errors (see 3.1) are found during linking. This means that no binary output is produced.

0253 *** WARNING, LINKAGE EDITOR

indicates that a warning is listed on the log. (See 3.2). Binary output has been produced.

The error messages occurring on the log are of two kinds, fatal errors and warnings. If a fatal error has occurred no binary output has been produced.

3.1 Fatal Errors. 3.1

M XXXXX : Multiple defined entry. XXXXX is the entry.

D XXXXX : Displacement overflow. The value of the displacement with name XXXXX is greater than 255 or the relocation property is neither absolute nor page zero relocatable.

Note: The value of a displacement external used in a JMP instruction is not checked. The reason for this is that the instruction code for the JMP instruction is 0. An external defined as displacement (EXTD) may be used as a normal external as well. If the left byte in the word, where the external appears, is zero (as for JMP) the use is considered to be as normal external, and then there is no restriction on the value.

R XXXXXX : Relocation error. XXXXXX may either be an octal number or an entry name.

If it is a number, it specifies an address where the relocation bits indicate neither an absolute address nor a normal or page zero relocatable address.

If it is an entry name, it specifies that the relocation bits for this entry are outside the range 1 to 5. (For relocation bits see ref. [3]).

START BLOCK MISSING:

No start block terminates the module.

TITLE BLOCK MISSING, FILE XXXXX:

The first block of the file is not a title block. The rest of the file is skipped.

3.1 SYNTAX, BLOCK TYPE NNNNNN: 3.1

The order of blocks is not as described in ch. 4.1. XXXXXX is an octal number specifying the block type breaking this sequence. The rest of the file is skipped.

SPECIFICATION ERROR, FORM = P:

The specification word of the program produced as output does not contain a one in bit 6. This bit should be set in paged programs [4]. The linking is stopped and no output is produced.

OVERWRITE NNNNNN:

The octal location NNNNNN is overwritten.

3.2 Warnings.

U XXXXX : Undefined symbol. The external symbol XXXXX is undefined.

ILLEGAL BLOCK, BLOCK SKIPPED, BLOCK TYPE NNNNNN:

A block type different from the types mentioned in ch. 4.1 is read. NNNNNN is an octal number specifying the block type.

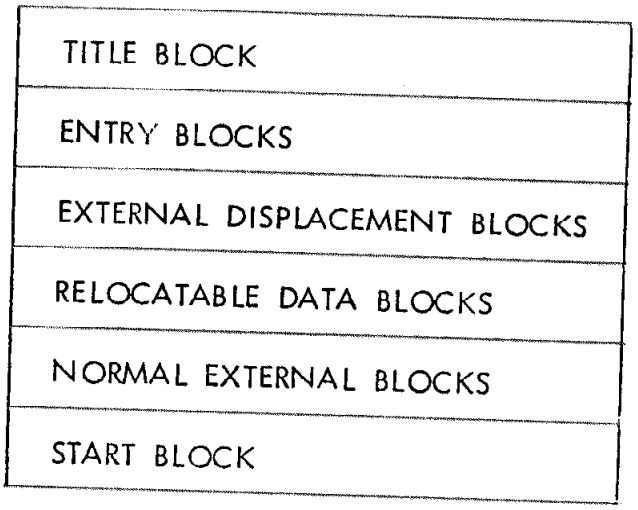
4. BINARY FORMATS.

4.

4.1 The Format of an Input Module.

4.1

Input to the Linkage Editor is relocatable binary modules produced by the DOMAC assembler. Each module is divided into series of blocks, the order of which is shown in the figure following.



Other types of blocks will be skipped of the linkage editor with a warning on the log.

Each module must contain at least a title block and a start block. Presence of one or more of the other types of blocks will depend upon the source text.

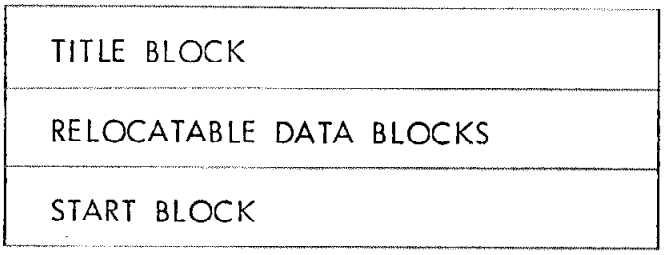
The contents of each of the block types is described in [3].

4.2 The Format of The binary Output.

4.2

The Linkage Editor can produce relocatable binary or absolute binary output. The relocatable binary output consists of a title block followed by one or more relocatable data blocks and

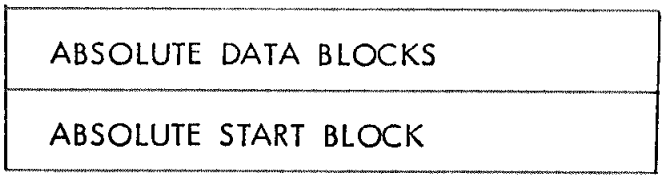
4.2 terminated by a start block.



The contents of each of the block types is described in [3].

The title stored in the title block of the relocatable binary output may be operator specified in the parameter TITLE. If not specified the title is MAIN. If one or more of the input modules contained a start address the last one of these will be start address of the relocatable binary output. If no start address appeared at input the output will not have a start address either.

Absolute binary output is divided into a number of absolute data blocks terminated by an absolute start block.



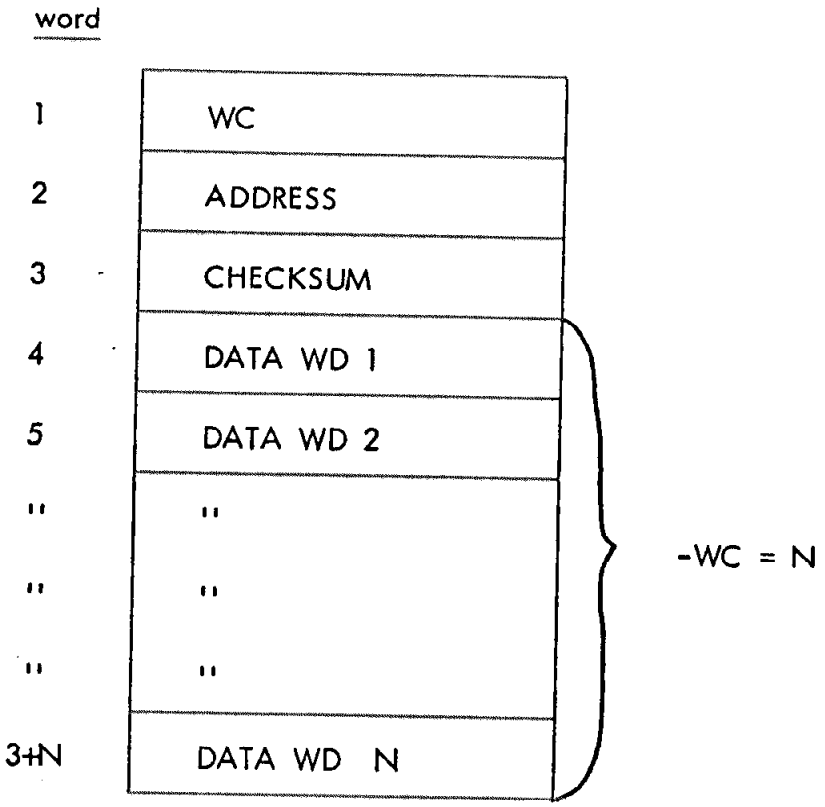
The first non-null character indicates the start of a new block.

The block type is determined by the first word of the block.

The first word, WC, of an Absolute Data Block is in the range

$$-20_8 \leq WC < 0.$$

4.2 Its format is:



WC is given in the first word. Normally sixteen data words will be punched per data block, but the .END and .LOC pseudo-ops to the Assembler may cause short blocks to be punched. The second word contains the address at which the first data word is to be loaded. The third word contains a checksum. This number is computed so that the binary sum of all words in the block should give a zero result.

4.2 The first word of a Start Block is 000001. Its format is:

word	
1	000001
2	S ADDRESS
3	CHECKSUM

The second word uses bit 0 as a flag. If S = 0, the loader will transfer control after loading to the address in bits 1 - 15 of the second word. The checksum is the same as that for a Data Block.

If a start address has been specified in more than one input module the last one met is used as start address in the binary output.

APPENDIX A - REFERENCES.

[1] DOMUS, User's Guide, Part 1

Keywords: DOMUS, MUS, Operating System, Loader, Disc.

Abstract: This manual describes the disc operating system DOMUS for the RC3600 line of computers.

[2] DOMUS, User's Guide, Part 2

Keywords: DOMUS, MUS, Operating System, Guide.

Abstract: This manual describes the utility system for the disc operating system DOMUS for RC3600 line of computers.

[3] DOMAC, Programmer's Reference Manual.

[3a] Introduction to DOMAC Assembler

Keywords: Beginners guide, DOMUS, DOMAC, RC3600, assembler.

Abstract: This manual contains a short introduction to the RC3600 assembler language description of how to invoke the DOMAC assembler, and a list of possible error messages from the DOMAC assembler.

[4] RC3600 Pagings System, System Programmers Guide.

Keywords: MUS, Paging System, Virtual Memory, Address Mapping.

Abstract: This manual describes how to use the RC3600 paging system from assembly programs under the MUS-system.

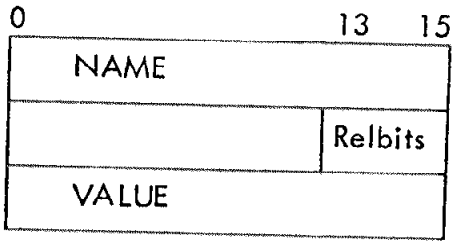
APPENDIX B - HOW LINK IS OPERATING.

The Linkage Editor operates in three passes, which will be described in the following.

Pass 1:

In pass 1 the syntax of each module is checked and all entry definitions are picked up in a symbol table.

The symbol table is build in a core item allocated in pass 1. The maximal number of entries in the table is determined by the value of the parameter ENTRY. An entry in the symbol table is described in three words:



The first word and bits 0 to 10 of the second work contain the name of the symbol in radix 50_8 representation. Bits 13-15 of second work contain the relocation bits of the value which is contained in the third word. Bits 11 and 12 in second word are not used.

Input modules with blocks of types 2, 3, 4, 5, 6, and 7 are accepted. All other block types are skipped with a warning on the log. The order of blocks is described in [3]. However, empty data blocks are accepted everywhere.

The checking of relocation bits for addresses in data blocks and for values of symbols defined in entry blocks is performed in pass 1 too. For entries the value of the relocation bits must be in the range $1 \leq$ re. bits ≤ 5. For addresses the value must be 1 (absolute), 2 (nrel.), or 4 (zrel.).

Pass 2:

In pass 2 external references are inserted. To do this the input modules are read again. For each module the external displacement symbols are numbered from 1 and on in the order met in the external displacement blocks, and this number is used in the relocatable data blocks to reference the displacement externals. From the external displacement blocks and the symbol table build in pass 1 a table is build containing the values and numbers for each displacement external in the module. The value can now be inserted when a number is referenced in a data block.

References to normal external symbols are linked in a chain, the head of which is placed in the normal external block. To be able to insert the values of normal externals a pseudo core image is build on the disc file '.INK'. This file is divided into blocks of 6 words. The first five words contain data and word number 6 contains relocation information of the five data words. Bits 13-15 are relocation bits for word number 1, bits 10-12 for word number 2, and so on. One segment contains 42 blocks and the last 4 words in a segment are not used. This means that each segment holds 210 core words. If a program contains absolute addresses these are kept at the first segments (as many as needed). Zero relocatable addresses, if any, occupy the next two segments and the last segments contain the normal relocatable addresses.

Pass 3:

The binary output is produced in pass 3. When relocatable binary or absolute binary output is specified, first all zero relocatable code is output, then all absolute addressed code and at last all normal relocatable code. For absolute output zero relocatable code is converted to absolute code starting in address 40 and normal relocatable code is converted to absolute starting in address 256. This means that when the output is loaded, first the zero relocatable code from all modules is loaded then all absolute code, and at last all normal relocatable code no matter what order the code appeared in the input modules. For paged programs all virtual code must be loaded before an initialization code placed in the space for frames. When FORM.P is specified LINK therefore first outputs the normal relocatable code from address nrel. 0 and to nrel 'first of frames' then all absolute code (virtual code) and at last the rest of the normal relocatable code.