


---

Title:

MUSIL COMPILER  
Operators Guide  
Revision 2

---

 **REGNECENTRALEN**

RC SYSTEM LIBRARY: FALKONERALLE 1 DK-2000 COPENHAGEN F

---

RCSL No: 43-GL8538  
Edition: February 1979  
Author: Marie Louise Møller

---

Keywords:

RC3600, MUS, MUSIL, Compiler, Operators Guide.

---

Abstract:

This manual describes the parameters to the MUSIL compiler.

---

Copyright A/S Regnecentralen, 1978  
Printed by A/S Regnecentralen, Copenhagen

Users of this manual are cautioned that the specifications contained herein are subject to change by RC at any time without prior notice. RC is not responsible for typographical or arithmetic errors which may appear in this manual and shall not be responsible for any damages caused by reliance on any of the materials presented.

---

CONTENTS	PAGE
1. INTRODUCTION.....	1
2. INPUT/OUTPUT DEVICES.....	2
2.1 Input/output devices in the MUS version.....	2
2.2 Input/output devices in the DOMUS version.....	2
2.3 Conversion.....	3
3. OPERATOR COMMUNICATION IN THE MUS VERSION.....	4
3.1 Compiler Commands.....	4
3.2 Check of Commands.....	6
3.3 Status Errors.....	7
4. OPERATOR COMMUNICATION IN THE DOMUS VERSION.....	8
4.1 Error Messages.....	9
5. COMPILER DIRECTIVES.....	10
5.1 \$COPY <name>.....	10
5.2 \$END.....	11
5.3 \$PAGE.....	11
5.4 \$PAGEF.....	12
6. THE LISTING.....	13
6.1 Error Messages.....	13
APPENDIX A - REFERENCES.....	16

This page is intentionally left blank.

1. INTRODUCTION.

1.

This guide is an update of RCSL: 43-GL1849, MUSIL COMPILER, Operators Guide. The features described here are included in the MUSIL COMPILER VERSION 10 and newer revisions.

Each revision of the compiler is now available in two versions, one which is able to run in all MUS systems and one for the DOMUS operating system. The MUS version is operated like older revisions taking commands after loading, while the DOMUS version runs as a DOMUS utility program [1]. The description in the following is for both versions if nothing else is mentioned.

It is now possible to output paged programs from the compiler. Page shifts are generated automatically, but it is also possible to enforce them by means of compiler directives. Part of the program may be fixed in core during execution. The page size may be selected by the operator. The default page size is one sector. Two frames are set aside for not fixed pages during execution.

The relocatable binary output may consist of both normal relocatable and extended relocatable code, thus making it possible to apply the extended memory in RC3803. If specified in the call the main program and the procedures will be output as extended relocatable code while constants, variables, codeprocedures and the process descriptor are output as normal relocatable code.

## 2. INPUT/OUTPUT DEVICES.

2.

### 2.1 Input/output devices in the MUS version.

2.1

The compiler reads the source file from a device specified as parameter to a command. Furthermore it is possible to select an output device for the object module and to specify a list device.

When a peripheral device is selected, the parameter contains its driver name with a § sign in front. Further, if the device is file-oriented, the file number should follow as :<fileno> after the driver name, e.g. file number 7 on tape unit 0 must be typed as §MT0:7. The compiler checks that the driver is loaded. If the selected device is a disc, the name of the disc area must be used, but without a § sign in front. In this case the disc driver and the catalog system must be loaded. The compiler initiates the selected disc kit, using the old disc catalog (use e.g. the editor to initiate a new disc) and updates this catalog when the compilation is finished. A disc output file is created if it does not exist. An input file must exist. A file on a disc unit different from unit 0 must contain a :<unitno> after the file name, e.g. the disc file LIBRY on unit 1 must be typed as LIBRY:1.

In MUSIL COMPILER 5 all devices are accepted. The devices PTR, PTP, MT0, MT1, CDR, RDP, CT0, CT1, FD0, FD1, are treated separately. All other devices are opened in mode 1 if input and mode 3 if output and with kind 1 (= character oriented) and share-length 512 bytes.

Notice that the CDR driver should be CR002 and the RDP driver should be RP001 or newer versions. If no object code or listing is wanted, <nl> may be typed after the OUT or the LIST command.

### 2.2 Input/output devices in the DOMUS version.

2.2

Parameters are specified in the load command as for other utility programs, and peripheral devices are selected and accessed by means of device descriptors. For further information see ref. [1].

2.3 Conversion.

2.3.

It is possible to use conversion tables for the list device and the input devices. These conversion tables are written as MUSIL programs. Paged programs must not be used as conversion tables. The name of the conversion table for the list device is CLTAB, for the normal input device it is called CITAB, and for the copy sources it is called CCTAB. The format of the conversion table programs is

```
const
table = # XXX
        X
        XXX
        #;
.....
begin
        .....
end;
```

### 3. OPERATOR COMMUNICATION IN THE MUS VERSION.

3.

When the compiler is ready to start a compilation, the text MUSIL READY is written on the operator device. Now a command followed by a parameter may be typed on the operator device. All the compiler commands have a standard value for the parameter.

#### 3.1. Compiler Commands.

Input source	IN	<name>	The standard name is \$PTR
Output	OUT	<name>	The standard name is \$PTP
Listing	LIST	<name>	The standard name is no device
Input code procedures	INCOD	<name>	The standard name is no device All the code procedure must be loaded from the device or file specified by <name>, e.g. one may have all the code procedure needed or more in a library disc or magtape file. When a code procedure is loaded, its name is written on the operator device.  Error messages during the loading: STATUS EM a code procedure is missing INCOD ERROR no load device is specified END ERROR a start block is missing SUM ERROR sumcheck error ILL ERROR inconsistent block
Process name	NAME	<name>	The standard process name of the object code is MAIN. Only the first 5 characters of name are used.
Identification	IDENT	<ident>	The standard ident is blank. Only the first 5 characters in the ident are used. The ident is output as an ascii text in front of the object code.



Operators device	OPCOM <device>	The standard device is TTY. The name, max. 5 characters, is the driver name of the object code's operator device without a \$-sign in front.
Modification	MODIF <chars>	The standard modification is empty. <chars> is a string of 1 to 6 letters or digits, where the following are allowed: <ul style="list-style-type: none"> <li data-bbox="890 537 1484 616">B One extra message buffer is added per file.</li> <li data-bbox="890 638 1508 716">C The program is a coroutine program. Modification B is included in C.</li> <li data-bbox="890 728 1484 806">N The object code is output without a process descriptor.</li> <li data-bbox="890 817 1508 952">X The main program and the procedures are output as extended relocatable code. Not allowed with modification P.</li> <li data-bbox="890 963 1492 1086">P The program output is a paged program containing two frames. Not allowed with modification X. <ul style="list-style-type: none"> <li data-bbox="890 1108 1364 1142">1 The page size is 1 segment</li> <li data-bbox="890 1153 1380 1187">2 The page size is 2 segments</li> <li data-bbox="890 1198 1380 1232">4 The page size is 4 segments</li> <li data-bbox="890 1243 1380 1276">8 The page size is 8 segments</li> </ul> </li> </ul> Modifications 1, 2, 4, and 8 have only effect if P is also specified. If P is specified but no page size, 1 is used, and if more than one digit is specified the last one is used.
Display	DISP	The command will display the contents of the other commands.
Initialization	INIT	The command will set the contents of the other commands to the standard values.
Start of compilation	START	The command will start the compilation, using the specified parameters.

An example of a set of compiler commands:

```

IN      TEST      ; source input from disc unit 0 file
                    TEST
OUT     $MT0:7    ; object code output on MT0 file 7
LIST    $LPT      ; listing on LPT
INCOD   $PTR      ; code procedure input from PTR
MODIF   B         ; one extra buffer per file
NAME    TEST      ; name of object code
IDENT   VER07     ; ident of object code
START                   ; starts the compilation with the
                    given parameters

```

### 3.2. Check of Commands.

If an error is made in giving commands, a message is printed on the operator device.

The messages are:

```

ILLEGAL COMMAND      , the command does not exist
DRIVER MISSING       , the driver or disc catalog system is not loaded
ILLEGAL DEVICE       , the device cannot be used for input or output
NO INPUT FILE        , the input disc file does not exist
FILE NO MISSING      , no file number is given for a file-oriented
                    device
ILLEGAL MODIF ITEM   , other characters than B, C, N, X, P, 1, 2, 4,
                    or 8 are used.

```

### 3.3. Status Errors.

3.3.

If a status error occurs on a device in use, the compiler writes

```
<devicename> STATE : <statusword> REP?
```

If the answer is NO, the compilation is skipped and the compiler is ready for the next compilation. On all other answers than NO the compiler performs a repeatshare except on <statusword> = 010020, (file already exists). If an already existing disc file is specified as binary output file the error message

```
<file name> STATE : 010020 REP?
```

is displayed. Here alle other answers than NO will cause the file to be overwritten unless it is writeprotected.

The error message COREOVERFLOW means that more core is needed for the compilation. The error message STACKOVERFLOW means that the program contains a too complicated expression or too many nested statements.

4. OPERATOR COMMUNICATION IN THE DOMUS VERSION.

4.

MUSIL IN.<in> OUT.<out> LIST.<list>!  
 ! INCOD.<codep> NAME.<name> IDENT.<ident>!  
 ! OPCOM.<opname> MODIF.<modif> BLOCK.<outblk>

<in> the name of the file where the input source is located.

<out> if specified, the object code is output on file <out>

<list> if specified, the source text is listed on file <list>

<incod> the name of the file where codeprocedures are loaded from

<name> the process name of the object code

<ident> if specified, the ident is output as an ascii text in front of the object code.

<opcom> the driver name of the object code's operator device

<modif> one to six characters denoting special functions.

The following are allowed:

B. one extra message buffer is added for each file in the program.

C. the program is a coroutine program. Modification B is included.

N. The object code is output without a process descriptor

X. The main program and the procedures are output as extended relocatable code. Not allowed with modification P.

P. The program output is a paged program containing two frames. Not allowed with modification X.

1 The page size is 1 segment

2 The page size is 2 segments

4 The page size is 4 segments

8 The page size is 8 segments

Modifications 1,2,4 and 8 have only effect if P is also specified. If P is specified but no page size, 1 is used, and if more than one digit is specified the last one is used.

<outblk> an integer specifying the blocklength on the output file. The maximal blocklength allowed is 512 bytes. The parameter has no effect if a disc file is used as output file.

Default values of call are:

MUSIL IN.\$PTR NAME.MAIN OPCOM.TTY BLOCK.512  
i.e. neither binary output nor listing is produced.

#### 4.1 Error Messages.

4.1

The standard DOMUS error messages are applied. Only two should be mentioned here:

0205 \*\*\* SHORT OF CORE STORAGE

means that more core is needed for the compilation.

0265 \*\*\* STACK OVERFLOW

means that the program contains a too complicated expressions or too many nested statements.

If an error occurs the error is displayed and the compilation is finished.

5. COMPILER DIRECTIVES.

5.

5.1. §COPY <name>.

5.1

The compiler directive §COPY makes it possible to insert a piece of program source at a given place in the normal program source. This copy source must come from a file different from the device specified by the IN-command. If the directive §COPY <name>, where <name> specifies a load device or file with the syntax described in ch. 2, is put into the program source, the compiler will insert the source read from the device or file until a §END is met. The compiler then returns to the normal input source and continues with this source. It is only possible to have one level of copy source. A conversion table called CCTAB may be used for these sources, (see ch. 2.3.).

The directive §COPY <name> must be terminated by a new line character.

```
Ex.:      :      ! normal input on file with !
          :      ! name different from PIP !
PROCEDURE XXX;
BEGIN
:
END;

§COPY PIP      ! insert procedure PIP !
:
:      ! procedures !
BEGIN      ! main program !
:
END
```

The file PIP contains the following:

```
PROCEDURE PIP;
BEGIN
:
END;

§END      ! terminate with a new line character!
```

5.2.    \$END.

5.2

This directive must terminate a copy input source, (see 5.1.).  
When read the compiler returns to the normal input source. The  
directive must be terminated by a new line character.

5.3.    \$PAGE.

5.3.

In paged programs the compiler automatically starts on a new  
page if the old one is filled not taking into account where it  
would be most optimal as regards speed of the program execution.  
On the listing is marked where the page shifts are inserted. By  
means of the directive \$PAGE it is possible for the programmer  
to enforce a page shift, e.g. to ensure that a loop will not be  
divided.

```
Ex.:  .
      .
      .
      i: = 0;
      $PAGE           ! start on new page !
      repeat
      .
      .
      .
      i: = i + 1
      until i < = maxno;
```

The directive has only effect if compilation is performed with  
modification P.

5.4. \$PAGEF.

5.4

As mentioned in 5.3. it is possible to enforce a page shift. The directive `$PAGEF` has the same effect as `$PAGE`, i.e. a new page is started, but in addition this new page will be fixed in core during execution. This is usefull e.g. if the program contains a procedure which is often used. The size of the program in core will be increased with one page for each fixed page.

```
Ex:  .
      .
      .
      $PAGEF          ! this page is fixed in core !
      procedure general;
      begin
      .
      .
      end;
```

The directive has only effect if compilation is performed with modification P.



6. THE LISTING.

6.

When a compilation starts the text MUSIL COMPILER VERSION X is written on the list device if you want a listing. After this text the compiler outputs the name and the ident of the object module. When the compilation is finished, the compiler writes either

NSIZE: XXXXX

or

NSIZE: XXXXX

XSIZE: ZZZZZ

or

NSIZE: XXXXX

PAGES: YYYYY

XXXXX is the decimal size (bytes) of that part of the compiled program, which must be loaded into the lower memory.

ZZZZZ is the decimal size (bytes) of the part of the program, which may be loaded into the high memory.

YYYYY is the decimal no. of pages in a paged program.

If no output is wanted, \*\*\* before size indicates that possible code procedures are not included in size.

In the case of errors the text ERRORS : XXXXX is written on the listing, and the object output is skipped.

In paged programs a + initiating a line on the listing indicates that a new page has been started when generating code for the preceding line.

6.1 Error Messages.

6.1

During compilation errors give rise to the following numbers on the listing or on the operator's console:

020202 Number overflow, a numeric constant exceeds 65535, or 16 bits.

020301 Illegal character in input.

030102 < appearing within a string is not followed by a numeric literal.

030202 The construct <number is not followed by a>.

030302 The number between < and > exceeds an 8-bit byte value.

030403 Core overflow, produced code exceeds available space.

030503 Core overflow, code contains too many relocation bits.

040105 Name conflict in Constant Section.

040205 Name conflict in Type Section.

040302 Syntax in Type Section, no = following an identifier.

040405 Name conflict in Variable Section.

040506 File variable with 0 buffers.

040602 Procedure head not followed by ,

040702 Procedure without legal identifier or with name conflict.

050102 Type is no identifier.

050202 ( is missing after string.

050302 Length undefined for string.

050402 String with length 255 declared.

050502 ) is missing after string.

050604 Undefined type identifier. Note that no forward declarations are allowed.

050702 Improper termination of type specification.

051002 Field of type different from string.

051102 Incorrect use of FROM.

051205 Name conflict in GIVEUP procedure.

051304 Conversion table undeclared.

051406 Conversion table type error.

060206 Double defined label.

060302 Variable is no identifier. Or undeclared.

060402 . is not followed by identifier or by undeclared field.

060504 Identifier undeclared.

060606 Type error with BYTE or WORD.

060702 Relational operator missing.

061002 Procedure statement with missing )

061102 Type error in procedure parameter.

061306 Illegal number of parameters.

061406 Type error with operator.

061506 Overflow of work registers. Expression too complex.

## Error Messages which cause skipping of program parts:

- 000040 Syntax in section delimiter.
- 000041 Syntax in constant declaration.
- 000042 Syntax in table declaration.
- 000043 Type specification incorrectly terminated.
- 000044 Variable declaration incorrectly terminated.
- 000045 Variable declaration incorrectly terminated.
- 000046 Procedure heading incorrect.
- 000051 Syntax in field list.
- 000052 Syntax in file declaration.
- 000063 Incomprehensible statement.
- 000064 Incorrect label declaration.
- 000065 Incomprehensible expression.

APPENDIX A - REFERENCES.

- [1] DOMUS, User's Guide, Part II.  
Keywords: DOMUS, MUS, Operating System, Guide.  
Abstract: This manual describes the utility system for the disc operating system DOMUS for RC3600 line of computers.
- [2] DOMUS, User's Guide, Part I.  
Keywords: DOMUS, MUS, Operating System, Loader, Disc.  
Abstract: This manual describes the disc operating system DOMUS for the RC3600 line of computers.
- [3] RC3600 Paging System, Operating Guide.  
Keywords: MUS Paging System, Survey, Operating System, Loader.  
Abstract: This manual describes how to operate the MUS Paging System, and lists the program modules accessible.