


AAGAARD

Title:

DOMUS, User's Guide, Part II

 **REGNECENTRALEN**

RC SYSTEM LIBRARY: FALKONERALLE 1 DK-2000 COPENHAGEN F

RCSL No: 43 - RI0432 (P1)

Edition: 77.09.15

Author: Dan Andersen.

CONTENTS

Page

PREFACE

1.1	UTILITY CALL	1-1
1.2	PARAMETER FORMAT	1-1
2.1	DEVICE HANDLING	2-1
2.2	USE OF DEVICE DESCRIPTORS	2-2
3.1	SYSTEM MESSAGES	3-1
3.2	MESSAGE GENERATION	3-5
4.1	STANDARD CONVERSION	4-1
5.1	UTILITY PROGRAMS	5-1
6.1	REFERENCES	6-1

APPENDICES:

Appendix	A	STANDARD UTILITY PROCEDURES	A-1
Appendix	B	STANDARD UTILITY MESSAGES	B-1
Appendix	C	STANDARD DEVICE DESCRIPTORS	C-1
Appendix	D	CONVERSION TABLE FORMAT	D-1

PREFACE

This manual describes the utility programs running under the RC 3600 DOMUS system.

Use of utility programs is based on the DOMUS s-function 'Utility program load', and can be reviewed as an extension of the S-commands.

The utility programs are designed in order to ease the administration of the disc files, the possibility to make backup and hardcopy of the files and to help the MUS-programmer with a set of tools in his program production.

Last the procedures used to standardize the interface between the utility programs and the environments are described.

1.1 UTILITY CALL.

The utility call is based on the s-function:

Utility program load given by the format

< filename> [< params>]

As seen from [1] the function is defined by the filename which is not a normal s-function.

The function is executed by loading the process placed in file < filename >, and transferring the < params > section to the process after fundamental syntax analyse and packing.

Further detailed analyse and interpretation is left to the utility program.

1.2 PARAMETER FORMAT.

In order to get a simple and standardized parameter interpretation all utility programs, with few exceptions, use the same format and check procedure.

The programs are at generation time born with a fixed number of parameters each specified with one of four types, a mnemonic parameter name and an expected placing in the < params > string.

The four types are NAMES, NUMBERS, TEXTS and booleans, where the first three is defined as in [1], and the latter is a name with either NO or YES as value.

The structure of < params > can then be defined as:

$$\begin{aligned} \langle \text{params} \rangle &::= \{ \langle \text{sep} \rangle [\langle \text{parameter name} \rangle .] \langle \text{parameter} \rangle \}^* \\ \langle \text{parameter name} \rangle &::= \langle \text{name} \rangle \\ \langle \text{sep} \rangle &::= \langle \text{space} \rangle \{ \langle \text{space} \rangle \}^* \\ \langle \text{parameter} \rangle &::= \langle \text{filename} \rangle | \langle \text{number} \rangle | \langle \text{text} \rangle | \langle \text{dummy} \rangle | \text{NO} | \text{YES} \\ \langle \text{filename} \rangle &::= \langle \text{name} \rangle [: \langle \text{number} \rangle] \\ \langle \text{dummy} \rangle &::= * \end{aligned}$$

As seen each parameter is called by a unique mnemonic name, which can be typed in front of the actual parameter followed by '.', hereby changing the position in the sequence of the parameters. This way of naming the parameters can be omitted, if all parameters are assigned in the sequence predefined by the program, and need only to be used if one or more parameters are skipped, or the defined sequence is broken.

Skip of a single parameter can be done by typing '*' instead of the parameter.

At program generation time each parameter has been assigned a default value too. These default values are taken if the parameters are not defined by the operator in the call, either because they have been skipped by the dummy parameter or one of the next parameters in the sequence has been selected by its name.

If any parameter of type name is used to define a catalog entry, the catalog-unit on which the entry resides can be transferred to the program by typing the delimiter ':' followed by the unit number (unit number 0 is default).

If any conflict between the parameters and parameter names typed and the expected format arise, the utility function is terminated with an error message on the console (***) PARAM).

For further explanation take following fictive utility-program MAIN:

DOMUS-UTILITY: MAIN

FORMAT: MAIN IN. < FILE > COUNT. < NUMBER >
OP. < BOOLEAN > TX. < TEXT >

FUNCTION: Undefined.

PARAMETERS: Default values of call are IN .\$PTR COUNT.20
OP.YES TX. '<0>...<0>'
Max textlength for parameter TX is 10.

EXAMPLE: MAIN \$ LPT 100 TX. 'ABCD'
MAIN \$ LPT 10 OP.NO
MAIN XX:3 TX.'AB'
MAIN IN.\$MT0
MAIN TX.'QV' OP.NO
MAIN \$LPT * YES

SPECIAL MESSAGES: 4000 *** UNDEFINED : < name >

INTERNAL EXECUTION: AS ABOVE

CORE REQUIREMENTS: 400 bytes dec

OTHER REQUIREMENTS: NONE.

1. Example

Values used by program MAIN are:

```

IN      =   $LPT
COUNT =   100
OP      =   YES          default
TX      =   'ABCD <0> <0> <0> <0> <0> <0>'

```

2. Example

Values used by MAIN are:

```

IN      =   $LPT
COUNT =   10
OP      =   NO
TX      =   '<0> <0> <0> ....<0>'  default

```

3. Example

Values used are:

```

IN      =   XX          file on unit 3
COUNT =   20          Default
OP      =   YES          Default
TX      =   'AB <0> <0> <0> <0> <0> <0> <0> <0>'

```

4. Example

Values used are:

```

IN      =   $MTO

```

and the rest default.

5. Example

Values used are:

IN	=	\$PTR	Default
COUNT	=	20	Default
OP	=	NO	
TX	=	'QV <0> <0> .. <0>'	

This call is an example of a different sequence of parameters than defined by the program, therefore the parameter names must be specified.

6. Example

Values used are:

IN	=	\$LPT	
COUNT	=	20	Default as it is skipped by *
OP	=	YES	
TX	=	'<0> ...<0>'	Default.

2.1 DEVICE HANDLING.

Only few utility programs have been designed to use special devices, and to avoid each program from keeping track of the device characteristics in the handling, a general way of file I/O initialization has been implemented.

Documents which are not disc files are defined in catalog entries, called device descriptors, which holds information of the driver name to use, the document kind, the mode of operation, the status bits to giveup on and last the document position, if actual, given by the filename and block-number.

Supplied with the DOMUS system are a number of predefined devicedescriptors which can be found in appendix C of this guide, but the user can define new entries using the utility program SET which creates new descriptors or modify existing descriptors.

Format of the SET call is:

```
SET < entry name > < driver name > < filevalue > < blockvalue > < mode >
      < kind > < giveup mask >.
```

The kind is easily given in radix 2, and can be interpreted after the scheme:

- bit 15 : character oriented devices i.e.
lineprinters, paper tape reader/punch
- bit 14 : block oriented devices i.e.
magtapes, card readers, disc files
- bit 13 : positionable devices i.e.
magtapes, disc files
- bit 12 : repeatable devices i.e.
devices supporting error recovery
t.ex. magtapes, disc files.
- bit 11 : disc files only.

For mode and giveup-mask it is difficult to give rules as it depends on the specific device driver, but normally mode = 1 means binary input and mode = 3 binary output, and for simple devices a giveup-mask with all bits set except bit 3,4 and 5 (soft status bit) is sufficient.

All utility programs initialize input/output with the procedure:

1. The entry defined by the parameter is looked up in the catalog. If no unitnumber is specified unit = 0 is taken as default.
2. If the entry exists and the entry is a device descriptor the file is expected on the device defined by the entry.
3. If the entry does not exist or the entry describes a normal disc file the input/output is transferred to/from the disc-file defined by the parameter. In case of output a new entry is created. In case the file exists already the program terminates to prevent overwrite of existing data.
4. The input/output is normally done with 512 bytes blocks, unless the block size can be changed by use of parameters.

2.2 USE OF DEVICE DESCRIPTORS.

1. Define the device descriptor \$PTR to describe the document punched tape with even parity:

```
SET $PTR PTR 0 0 9
```

2. Define the descriptor describing file 10 on magtape MT0 with input:

```
SET $MT MT0 10 1 1 2'1110
```

Kind is set to repeatable, positionable and blocked.

3. Describe the operator console used for both input and output

```
SET $TTY TTY 0 0 1
```

4. Describe the file PIP on catalog unit 10 with a descriptor on unit 0 with the same name:

```
SET PIP PIP : 10 0 0 1 2'11110 -1
```

Kind must be set to disc and giveup-mask must have all bits set.

5. Copy file number 4 on magtape station MT2 to file number 6 on magtape station MT0:

```
SET MTIN MT2 4 1 1 2'1110
FINIS SET
SET MTOUT MT0 6 1 3 2'1110
FINIS SET
COPY MTIN MTOUT
FINIS COPY
```

6. For devices enabling both input and output it is sufficient to specify the mode for input as the utility programs in case of out-put always will or the mode found with the output mode 3.

7. Descriptor \$MT created in 2) can be used to define a new file on same station by:

```
SET $MT MT0 11
```

as all non given parameters are taken from the existing descriptor.

3.1 SYSTEM MESSAGES.

All system messages resides on the command file SSYSE on catalog unit0, and each message is identified by a unique number in the interval 1 to 9999.

Each error occurring in the system is converted to an appropriate message number, and the text to output is fetched from the common file. This enables the user to add, translate, change or reformulate the error-text connected to the number.

In order to avoid conflicts in use of the texts they have been split in different groups:

Number	Used by
0 - 99	DOMUS Operating system
100 - 1999	Standard utility programs
2000 - 2999	Standard device errors Each text is found by adding a base connected to a specific device with the number of the left- most status bit set
3000 - 7999	Standard application programs
8000 - 8999	Informative texts
9000 - 9999	Customer available texts, free to use for any customer designed texts or messages.

In the groups from 0 - 7999 the text delivered is allways defined with the heading XXXX ***, where XXX is the textnumber.

- 0111 *** FILE DOES ALREADY EXIST, FILE < name >
The entry to create is already
present in the catalog on the
selected unit.
- 0112 *** INDEX BLOCK FULL, FILE < name >
The maximum filelength has
been reached on file < name >

Message numbers in case of errors in the initialization of input/output or data transfer to files are found by adding base 120 to the leftmost status bit number after removal of bit 3 and bit 4:

- 0120 *** CATALOG I/O ERROR; FILE < name >
See error 100.
- 0121 *** FILE DOES NOT EXIST, FILE < name >
See error 101.
- 0126 *** FILE IN USE, FILE < name >
The file to work on is reserved
for exclusive use by another process,
or a process with name < name > is
present in core.
- 0127 *** NO FREE AREA PROCESS TO FILE < name >
The common pool of area processes
is empty. The pool can be extended
by loading more area processes.
- 0131 *** END MEDIUM ON FILE < name >
Physical end of file found before
an expected logical end of file.

0132 *** MAP/FILE EXCEEDED, FILE < name >
The maximum filelength has been
reached on file < name > or no
more free segments due to a con-
figuration error on the disc.

Message numbers used in case of disc errors or device errors are found by adding the base 2000 to the number of the leftmost bit set after removal of bit 3 and bit 4:

2000 *** DISCONNECTED, FILE < name >
The device is not ready or
set ready by the operator, or the
hardware is missing.

2001 *** OFF-LINE, FILE < name >
The device is or has been set
local by the operator.

2002 *** BUSY, FILE < name >
The device is not ready to
accept input/output transfers.

2006 *** RESERVED OR ILLEGAL OPERATION, FILE < name >
The driver is reserved by another process,
or the operation is unknown to the
driver or write has been attempted
on a writeprotected device.

2007 *** END OF FILE, FILE <name>
See error 0131.

2008 *** BLOCKLENGTH ERROR, FILE <name>
The block read was too big
to be held in the used buffer
size.

The block output was too big
to be held on the document.
Format error in input.

- 2009 *** DATA LATE, FILE <name>
The CPU was too busy to
respond on a memory reference
from device <name>. High speed
devices only.
- 2010 *** PARITY ERROR, FILE <name>
One or more characters read
had a parity error.
- 2011 *** END MEDIUM, FILE <name>
See error 0131.
- 2012 *** POSITION ERROR, FILE <name>
The drive is unable to find
the position requested.
- 2013 *** DRIVER MISSING, FILE <name>
The driver to use is not loaded
by the operator.
- 2014 *** TIMEOUT, FILE <name>
The device did not respond on
the requested operation within a
maximum time.

3.2 MESSAGE GENERATION.

Supplied with the system is the common text file SSYSE. The source text out
of which the file is generated is placed in the file ERMES.

Modification of the sourcefile is done by the Text Editor, and the formatted file SSYSE is generated by use of utility program GENER (see this).

Unused text numbers will not occupy disc space.

Maximum messagelength is 502 characters, and undefined messagenumbers result in the text *** UNREGISTERED ERROR if fetched from the SSYSE file by DOMUS.

4.1 STANDARD CONVERSION.

When programs output to printers data are send as ASCII characters.

As the RC 3600 lineprinters can run with a number of different print-drum character-sets all lineprinter drivers support use of standard conversion.

This conversion is initiated by connecting a conversion table placed in core to a specific printer driver by use of the utility program STACO.

The function of STACO is to connect a conversion-table placed in a core item to the selected driver and to prevent the conversion-table from removal without the drivers knowledge. A list of coreitems after the STACO call will show that the driver is placed as owner of the coreitem containing the conversion-table.

Regret of standard conversion can only be done by a kill of the owner driver, and a reload of the process.

Standard conversion is only used by the drivers if the reserver program does not specify its own conversion-table, hereby enabling application programs with own defined tables to run without change.

DOMUS-UTILITY: APPEND

FORMAT: APPEND OUT. <file> IN. <file> <file> ...

FUNCTION: The command activates the utility program append which copies from IN. <file> to OUT. <file>. Trailing zeroes in each file are skipped. Up to 10 input files may be specified.

PARAMETERS: The ident <file> is the name of the output file and the input files.

EXAMPLE: APPEND \$PTP DATA1 DATA2 DATA3
FINIS APPEN

The files DATA1, DATA2 and DATA3 are copied to file \$PTP.

SPECIALMESSAGES: 0200 *** NOT ENOUGH ARGUMENTS

INTERNAL EXECUTION: As described above.

CORE REQUIREMENT: 4410 bytes.

OTHER REQUIREMENTS: None.

DOMUS-UTILITY:

CATLIST.

FORMAT:

CATLIST MASK. <name mask> OUT. <file>

FUNCTION:

A sorted list of catalog entries is output on file <file>. In <name mask> \$ replace all possible characters.

Format of the output will be for normal entries:

*word**0-2*1: Entry name *+ UNIT NO.**6*

2: Entry attributes, with following interpretation:

C: Catalog entry

B: Big slice extension

L: Link entry

P: Permanent entry

W: Write protected file

E: Entry only *(d.v.s. ingen RB-cord)*

D: Device descriptor

V: Extendable file

F: Fixed length file

8

3: Segment number of index block

7

4: File length of file

9

5: Reserved length

10-11

6: Entry optional words 1 as ASCII string

12-15

7: Entry optional words 2 as ASCII string

In case of device descriptor entry:

1: Entry name

2: Entry attributes as for normal entry

3: Driver name

- 4: File number (decimal)
- 5: Block number (decimal)
- 6: Mode (decimal)
- 7: Kind (binary)
- 8: Giveup-mask (octal)

PARAMETERS:

Namemask is a mask selecting the entries to be printed. Any entry name that fits the mask is listed. Character \$ replace any character i.e. PIP\$ as mask will output all entries with name-length 4 where first three characters = PIP.

If unitnumber is specified on parameter MASK, the catalog on this unit is scanned for matching filenames.

Default values are:

CATLIST MASK.\$\$\$\$\$ OUT.\$TTY

i.e. all entries on unit 0 is listed on device described by \$TTY.

SPECIAL MESSAGES:

NONE

INTERNAL EXECUTION:

As above

CORE REQUIREMENTS:

16000 bytes dec

OTHER REQUIREMENTS:

NONE

DOMUS-UTILITY:

CHATR

FORMAT:

CHATR NAME. <file> ATT. <attributes>

FUNCTION:

The command activates the utility program CHATR which changes the attributes of a file with a specified name.

PARAMETERS:

The ident <file> is the name of the file to be changed.

The ident <attributes> is composed of file specification characters. Allowed specifications are: B (big slice extension), P (permanent file), W (write protected file), V (variable size i.e. extended-able), F (fixed size).

V is default value.

EXAMPLE:

```
CHATR PTP PW
FINIS CHATR
```

The file PTP is changed to have the file specifications permanent and write-protected.

```
CHATR WORK
FINIS CHATR
```

The file work is changed to not having the file specifications permanent and write-protected, and to be of variable size i.e. extendable.

SPECIAL MESSAGES:

```
0200 *** NOT ENOUGH ARGUMENTS
0202 *** ILLEGAL FILE SPECIFICATION
```

INTERNAL EXECUTION: As described above.

CORE REQUIREMENT: 1600 bytes.

OTHER REQUIREMENTS: None.

DOMUS-UTILITY:	CONFIGURATION
FORMAT:	CONFIGURATION LIST. <file>
FUNCTION:	All titles of modules placed in the current DOMUS basic system are listed on <file>.
PARAMETERS:	Only parameter is the list file name. Default value is operator console.
EXAMPLE:	CONFIGURATION XXX FINIS CONF XXX contains after call: *** DOMUS SYSTEM CONFIGURATION *** MUM04 --- --- MUI04
SPECIAL MESSAGES:	Only standard errors.
INTERNAL EXECUTION:	As above.
CORE REQUIREMENTS:	3254 bytes dec.
OTHER REQUIREMENTS:	Bootstrap-program DB000 or later versions must be used for system-start.

DOMUS-UTILITY: COPY

FORMAT: COPY IN. <file> OUT. <file> BLOCK. <size>

FUNCTION: The command activates the utility program COPY which copies from IN. <file> to OUT. <file>.

PARAMETERS: The ident <file> is the name of the output file and the input file. The integer <size> is the blocklength on the output file. Default value is 512.

EXAMPLE: COPY \$PTR \$PTP
FINIS COPY

The file \$PTR is copied to the file \$PTP.

SPECIAL MESSAGES: 0200 *** NOT ENOUGH ARGUMENTS

INTERNAL EXECUTION: As described above.

CORE REQUIREMENT: 4254 bytes.

OTHER REQUIREMENTS: None.

DOMUS-UTILITY:

CREATE

FORMAT:

CREATE NAME. <file> SIZE. <size> ATT.
<attributes>

FUNCTION:

The command activates the utility program CREATE which creates a file with the specified name, size and attributes.

PARAMETERS:

The ident <file> is the name of the file to be created.

The integer <size> is the number of sectors in the file. 1 is default size.

The ident <attributes> is composed of file specification characters. Allowed specifications are: B (big slice extension of file), P (permanent file), W (write protected file) F (fixed size). V (extendable) is default.

EXAMPLE:

CREATE WORK 100 PF
FINIS CREAT

The file work is created with size 100 sectors and with file specifications permanent file and fixed size.

SPECIAL MESSAGES:

0200 *** NOT ENOUGH ARGUMENTS
0202 *** ILLEGAL FILE SPECIFICATION

INTERNAL EXECUTION:

As described above.

CORE REQUIREMENT:

1520 bytes.

OTHER REQUIREMENTS:

None.

DOMUS-UTILITY:	DELETE
FORMAT:	DELETE NAME. <file>
FUNCTION:	The command activates the utility program delete which deletes the file with the specified name.
PARAMETERS:	The ident <file> is the name of the file to be deleted.
EXAMPLE:	DELETE WORK FINIS DELET The file work is deleted.
SPECIAL MESSAGES:	0200 *** NOT ENOUGH ARGUMENTS.
INTERNAL EXECUTION:	As described above.
CORE REQUIREMENT:	1336 bytes.
OTHER REQUIREMENTS:	None.

DOMUS-UTILITY: DISK

FORMAT: DISK UNIT. <unit no>

FUNCTION: Number of free segments and number of used segments on catalog unit <unit no> is output on the operator console.

PARAMETERS: Only parameter is the integer <unitno>, which must not exceed 255. Default value is unit = 0.

EXAMPLE: DISK 2
>DISK
USED: 2500 LEFT: 1000
>S
FINIS DISK

SPECIAL MESSAGES: 0206 *** UNITNO GREATER THAN 255
0207 *** UNIT NOT MOUNTED
0208 *** UNIT DOES NOT EXIST

INTERNAL EXECUTION: As above.

CORE REQUIREMENTS: 1378 Bytes dec

OTHER REQUIREMENTS: None.

DOMUS-UTILITY: EDIT

FORMAT: EDIT <filename>

FUNCTION: This utility is the system text editor.

All functions are described in the Text Editor manual (version two). [4]
Using the editor command H will finish editing as described in [4] , and finish the editor function.

PARAMETERS: If <filename> is typed the editor will perform UY command on <filename> automatically, and the first page is ready in edit-buffer.

SPECIAL MESSAGES: See [4] .

INTERNAL EXECUTION: As above.

CORE REQUIREMENTS: 13714 bytes dec.

DEVICE HANDLING: Device descriptors are not supported by the Text Editor, for further details see [4] .

DOMUS-UTILITY:

FCOPY

FORMAT:

FCOPY FUNC. <direction> MASK. <name mask> LIST. <file> FIL. <filenumber> MT. <magtape no> UPDATE. <boolean>

FUNCTION:

Dumps or loads all or selected disc-files, except system files and files with names equal to loaded processes, to/from magnetic tape. A log of the disc filenames and sizes is output on the file selected by LIST parameter.

PARAMETERS:

The direction of the transfer is selected by <direction>, which must be the names DUMP or LOAD, or else the call is rejected with parameter error. <namemask> is the name(s) of the discfile to dump or load. The character \$ means all characters, and only filenames matching the mask is transferred. If an unit number is appended on parameter MASK, transfer is done to/from this catalog unit.

Parameter FIL is the filenumber on magtape station <magtape no> on which files are read or written.

UPDATE is only used if LOAD is specified in which case the value YES means that already existing files with the same name can be overwritten, else the disc transfer is skipped.

Default values are:

FCOPY FUNC.DUMP MASK.\$\$\$\$\$ LIST.\$LPT
FIL.1 MT.0 UPDATE.NO

SPECIAL MESSAGES:

0003 *** PARAM

CORE REQUIREMENTS:

6900 bytes dec

OTHER REQUIREMENTS:

Each file on the tape contains for each disc file one block of 32 bytes containing the entry and a number of 512 bytes blocks matching the file length of the file.

DOMU-UTILITY: GENER

FORMAT: GENER IN. <source> LIST. <file> OUT. <file2>

FUNCTION: The program reads an ASCII file <source> containing all system messages, produces a listing on file <file1> of these messages and of any errors detected in the text file, and creates a disc file <file2> containing these messages in a form accessible by DOMUS. Number of errors is output on console, and line and page number of each error can be found in the listing.

PARAMETERS: Default values are
 GENER IN.TRE LIST.LP OUT.SSYSE
 Please note that standard device names are not used.

EXAMPLE: GENER ERMES \$LPT MSYSE
 This call causes gener to read and check the ASCII text file ERMES and to create the file MSYSE with listing on \$LPT.

SPECIAL MESSAGES: Standard error messages is not used. In case of I/O trouble the text
 *** DEVICE STATUS <file> <status>
 is output.
 <file> is the device or file causing troubles and
 <status> is the octal status word.

INTERNAL EXECUTION: As above.

CORE REQUIREMENTS:

8400 bytes dec.

OTHER REQUIREMENTS:

The ASCII input file contains the system messages in ascending order each in the form:

<number> <identification> '<message>' <NL>

The first line should contain a zero and the identification of the file.

This line contains no message. The following lines contains the messages.

<number> is a sequence of decimal digits.

<identification> is any sequence of characters not containing a ' <quote>. This sequence is not checked.

<message> is a mixture of printable ASCII characters and constructs of the form

<< number >>. In general NL and FF are skipped outside numbers.

Characters with value > 127 are converted into ?.

Maximum message length is 502.

DOMUS-UTILITY:

LIBE

FORMAT:

LIBE LIB. <library> OUT. <file> FUNC.
<function> PROC. <procedure>

FUNCTION:

With this utility MUSIL-code procedure libraries can be edited by extract, delete, addition and list of procedures in the library.

A code procedure is defined as a number of relocatable binary blocks with a leading tittleblock and a trailing startblock.

Except for the listfile only discfiles are supported by the Library Editor.

PARAMETERS:

<library> is the discfile containing the library to work on.

<file> is the logfile

<function> is a name defining the operation:

LIST: List code procedure names and sizes on logfile.

ADD: Insert the code procedure contained in file <procedure> in the library. Library file is created if non existing.

DEL: Remove the code procedure with title <procedure> from the library.

EXT: Place the codeprocedure with title

<procedure> on a discfile with same name. The procedure remains untouched in the library.

Default values of OUT and FUNC are:

OUT. \$TTY FUNC.LIST

SPECIAL MESSAGES:	None
INTERNAL EXECUTION:	As above
CORE REQUIREMENTS:	7828 bytes dec.
OTHER REQUIREMENTS:	A workfile is created the file with the same name as the library, except for the first character which is replaced by '.' (dot).

DOMUS-UTILITY: MUSIL

FORMAT: None

FUNCTION: MUSIL high-level language compiler.

PARAMETERS: Runtime parameters can be entered after loading, when MUSIL READY is output on console.

For parameter format and entering please find the description in MUSIL COMPILER, Operating Guide [5] .

SPECIAL MESSAGES: See [5] .

CORE REQUIREMENTS: 13336 bytes dec.

DEVICE HANDLING: Device-descriptors are not supported by the MUSIL-compiler, for further details see [5] .

Process name of compiler: COMP.

DOMUS-UTILITY:

NEWCAT

FORMAT:

NEWCAT UNIT. <unitno> BIG. <size1>
SMALL. <size2> SEG. <number>

FUNCTION:

A new and empty catalog is created on
unit = <unit no> with big slice size =
<size2>, small size = <size1> and <number>
segments.

PARAMETERS:

Unit-number must not be 0 as unit0 is the
system disc at running time, and maximal
unitnumber accepted is 255.
Slices sizes must fulfil the condition 1
< size2 < size1 < 255.

EXAMPLE:

NEWCAT 1 24 6 4872

All files on catalog unit1 is removed, and a
new and empty catalog is created.
Slice sizes are selected to be 6 and 24.
Total number of segments in unit1 is 4872,
i.e. a 2.4 mb disc is used.

SPECIAL MESSAGES:

0209 *** ILLEGAL UNITNUMBER
0210 *** ILLEGAL DISC SIZE

INTERNAL EXECUTION:

As above

CORE REQUIREMENTS:

1382 bytes dec

OTHER REQUIREMENTS:

Catalog initialization process CATI must be
loaded, or else error

2013 *** DRIVER MISSING, FILE CATI
is output on console.

WARNING:

All files on selected unit are deleted
independent of attribute protection.

DOMUS-UTILITY:

PRINT

FORMAT:

PRINT IN. <file1> LINE. <boolean> OUT.
<file2>

FUNCTION:

File <file1> is output on the file <file2> (normally the lineprinter). Character tab with ASCII value 9 is converted to spaces. If LINE=YES, linenumbers are output in front of each line as a four digit number. Linenumbers are equivalent to linenumbers printed by MUSIL compiler.

PARAMETERS:

First parameter is the name of the file to be printed.
Third parameter is the output file, and second is a boolean with value YES or NO specifying if linenumbers should be printed.

Default values are:

PRINT IN.\$PTR LINE.NO OUT.\$LPT

EXAMPLE:

PRINT QQ LINE.YES

File QQ is printed on device \$LPT with linenumbers.

PRINT QW:1 YES \$SP

File QW on catalog unit 1 is printed on device \$SP with linenumbers.

SPECIAL MESSAGES:

Only standard messages are used.

INTERNAL EXECUTION:

As above.

CORE REQUIREMENTS:

3116 bytes dec

OTHER REQUIREMENTS:

Depending on printer-drum standard conversion can be used, by connecting a conversiontable to the printerdriver with program STACO.

It is recommended to use program COPY for datatransfer to non printer files.

DOMUS-UTILITY:

PUNCH

FORMAT:

PUNCH IN. <file> MODE. <modename>
 PNO. <punch num>

FUNCTION:

File <file> is output on punch number
 <punch num> with the type defined
 in <modename>.

Modename = NO : no parity
 Modename = ASCII : even parity
 Modename = EVEN : even parity
 Modename = ODD : odd parity

PARAMETERS:

Default value of mode is EVEN and for
 punchnumber 0.

Only first character in modename is
 checked, and if any other character than
 A, E, N, or O is found no parity punch
 is performed.

EXAMPLE:

PUNCH PIP A 1

File PIP is transferred to PTP1 in even
 parity.

PUNCH PAP: 2 NO

File PAP is punched on PTP without parity.

SPECIAL MESSAGES:

2026 *** PUNCH RESERVED
 2031 *** PAPER LOW ON PUNCH
 2033 *** PUNCH DRIVER NOT LOADED
 2034 *** PUNCH ERROR OR TIMEOUT

INTERNAL EXECUTION:

As above

CORE REQUIREMENT:

2440 bytes dec

OTHER REQUIREMENTS:

None

DOMUS-UTILITY:

RENAME

FORMAT:

RENAME OLD. <file> NEW. <file>

FUNCTION:

The command activates the utility program
RENAME which changes the name OLD.
<file> to NEW. <file>

PARAMETERS:

<file> is an ident and indicates current name
and new name of the file.

EXAMPLE:

RENAME WORK SAVE
FINIS RENAM

The file WORK is renamed SAVE and the
name WORK is removed from the catalog.

SPECIAL MESSAGES:

0200 *** NOT ENOUGH ARGUMENTS
0201 *** UNIT NUMBER CONFLICT

INTERNAL EXECUTION:

As described above.

CORE REQUIREMENT:

1518 bytes

OTHER REQUIREMENTS:

None

DOMUS-UTILITY:

SET

FORMAT:

SET NAME. ^{at. u.s. character name} <entry> DEVICE <docname>
 FILE. <filno> BLOCK. <blockno> MODE.
 <modevalue> KIND. <kindvalue> MASK.
 <givup>

FUNCTION:

This utility creates a new device descriptor or change an existing devicedescriptor according to the values given at call.

PARAMETERS:

The ident <entry> is the name of the device-descriptor to change or create.

The ident <docname> is the name of the equivalent document name i.e. the driver name.

Integer <filno> and <blockno> are the wanted position of the document when the descriptor is used and referenced.

The integers <kindvalue> and <givup> are the values used in the filedescriptor when the user utility inputs from/ outputs to the document.

Default values are when devicedescriptor is created:

FILE.1 BLOCK.1 MODE.1 KIND.1
 MASK.8'163777

If descriptor exists default values are the values found in this entry.

EXAMPLE:

```
SET $PTP PTP MODE.11
FINIS SET
```

Devicedescriptor \$PTP describes the document punched tape with ASCII even parity.

```
SET $MT0 MT0 4 1 1 2'1110
FINIS SET
```

Devicedescriptor \$MT0 describes file 4 on MT0. Kind is set to repeateable, positionable and blocked.

Define the task to copy from MT1 file # 2 to file 10 on MT0:

This can be done by:

```
SET MTOUT 10 1 3 2'1110
FINIS SET
SET MTIN MT1 2 1 1 2'1110
FINIS SET
COPY MTIN MTOUT
FINIS COPY
```

SPECIAL MESSAGES:

```
0211 *** NOT DEVICE DESCRIPTOR, ENTRY
                                     <entry>
0212 *** FILE OR BLOCK TOO LARGE
```

INTERNAL EXECUTION:

As above

CORE REQUIREMENTS:

1670 bytes dec

OTHER REQUIREMENTS:

None

DOMUS-UTILITY:

STACO

FORMAT:

STACO <drivename>. <item name>

FUNCTION:

The program searches for the driver <driver name> and inserts the table loaded in the utility core item <item name> as standard conversion.

The owner of the core item containing the table is changed to be the driver. Thus the table may only be removed from core by killing the driver.

PARAMETERS:

No default values.

EXAMPLE:

```
LOAD LPT TAB7
STACO LPT. TAB7
FINIS STACO
```

Table found in item TAB7 is taken as standard conversion in LPT driver.

SPECIAL MESSAGES:

```
*** SYNTAX
*** DRIVER DOES NOT EXIST
*** NOT DRIVER PROCESS
*** DRIVER RESERVED
*** STANDARD CONVERSION EXISTS
*** TABLE DOES NOT EXIST
*** TABLE NOT UTILITY COREITEM
```

INTERNAL EXECUTION:

As above

CORE REQUIREMENTS:

460 bytes dec

OTHER REQUIREMENTS:

For table format and coding see appendix D of this guide.

DOMUS-UTILITY:	TYPE
FORMAT:	TYPE IN. <file> LINE. <boolean>
FUNCTION:	File <file> is output on the operator console. Tab character (ASCII value 9) is converted to spaces. If LINE=YES linenumbers are output in front of each line as a four digit number, and linenumbers are equivalent to linenumbers printed by MUSIL-compiler.
PARAMETERS:	First parameter is the file to be output. Second parameter is a boolean with value YES or NO, specifying if line numbers are wanted in front of each line. Default values are: TYPE IN. \$PTR LINE.NO
EXAMPLE:	TYPE QQ YES File QQ is typed with linenumbers. TYPE LINE. YES Paper tape in reader is output with linenumbers.
SPECIAL MESSAGES:	Only standard messages.
INTERNAL EXECUTION:	As above.
CORE REQUIREMENTS:	2452 bytes dec
OTHER REQUIREMENTS:	None

DOMUS-UTILITY: XREF

FORMAT: XREF IN. <source> OUT. <file>

FUNCTION: This utility makes a cross-reference list of all constants, types, variables, procedures and labels on file <file> in a MUSIL source on file <source>. A sorted list of all types declared is produced, with linenumbers in which they appear.

PARAMETERS: Default values are:
XREF IN. \$PTR OUT.\$LPT

EXAMPLE: XREF SOURCE \$SP
Cross-reference list of file source is output on device \$SP.

SPECIAL MESSAGES: 0203 *** COMMUNICATION ERROR WITH S
0204 *** SHORT OF CORE STORAGE
0205 *** NON-ASCII CHARACTER IN INPUT

INTERNAL EXECUTION: As above

CORE REQUIREMENTS: 5136 bytes dec and coreitem XREFC for internal sort.

OTHER REQUIREMENTS: None

REFERENCES

- [1] RCSL: 43-R10165 DOMUS, User's Guide, Part I
- Keywords: DOMUS, MUS, Operating System, Loader, Disc.
- Abstract: This manual describes the disc operating system DOMUS for the RC 3600 line of computers.
- [2] RCSL: 43-R10164 DOMUS System Programmer's Guide
- Keywords: MUS, Operating System, Loader, Disc
- Abstract: This manual describes the interface between assembly programs and DOMUS
- [3] RCSL: 42-i0344 MUSIL Programming Guide
- Keywords: Programming, coding, MUSIL RC 3600, source language.
- Abstract: This manual shows how to program an RC 3600 in the MUSIL high-level language
- [4] RCSL: 42-i0337 TEXT EDITOR (version two)

[5] RCSL: 43-GL1349 MUSIL COMPILER
 Operators Guide

 Keywords: RC 3600 Musil Compiler

 Abstract: Compiler Guide, Operators Guide.

APPENDIX A, STANDARD UTILITY PROCEDURES.

Following three procedures have been made to ease the programming of utility programs when fetching parameters, making I/O with device descriptors and finish the utility function.

The procedures will when used give a standard interface to the operator and the operating system.

FETCH OF PARAMETERS.

Fetch of parameters can be done with the code procedure GETPARAMS (P0085), which is called with three parameters each of type string.

The first parameter is a constant string defining the expected format of the parameters, the parameter names, types and sequence.

The format of the string is:

```
'<5 bytes parameter name> <1 byte parameter type>  
.  
.  
.  
<5 bytes parameter name> <1 byte parameter type>  
<255>'
```

Last byte in the description string is 255, terminating the parameter list. In case the parameter name is less than 5 characters, zero bytes must be inserted in the end of the parameter name.

4 values of parameter types exist:

VALUE	TYPE OF PARAMETER
134	NAME (STRING(6))
130	INTEGER (INTEGER)
129	BOOLEAN (STRING(1)) The string equals <255> if parameter = 'YES' and <0> if parameter = 'NO'.

VALUE <128 TEXT (STRING (VALUE))

Or as an example:

CONST

.
.
.

DESC = '

```

IN <0> <0> <0> <134> : Parameter name = IN, Type = NAME!
COUNT <130>       : Parameter name =COUNT, Type = INTEGER!
OP <0> <0> <0> <129> : Parameter name = OP, Type = BOOLEAN!
TX <0> <0> <0> <10>  : Parameter name = TX, Type = STRING (10)!
<255>',           : Terminator!
```

The second parameter is a constant string defining the default of all the defined parameters. These values are taken if some of the parameters are not specified in the utility call:

INIT = '

```

$PTR <0> <0>       : FIRST PARAMETER DEFAULT!
      <0> <20>     : SECOND PARAMETER DEFAULT!
      <255>       : THIRD PARAMETER DEFAULT!
<0> <0> <0> <0> <0> <0> : FOURTH PARAMETER DEFAULT!
```

As seen the parameter OP has been assigned the default value 'YES'.

Third parameter is a variable string, which after call holds the parameters typed by the operator, or the assigned default value if the parameter is skipped by the operator.

It can be build as a record in MUSIL, and must as the first element contain a string (6), which is set to the loadname of the user program:

```
VAR
.
.
.
PAR:      RECORD
          LNAME:  STRING(6) ; !SPACE FOR LOADNAME!
          INV   :  STRING(6) ; !FIRST PARAMETER   !
          COUN  :  INTEGER ; !SECOND PARAMETER   !
          BOOL  :  STRING(1) ; !THIRD PARAMETER   !
          TEXT  :  STRING(10) !FOURTH PARAMETER   !
          END;
```

The content of the fields in PAR after call of GETPARAMS are either the default value specified by the DESC string, if the parameter is not typed by the operator, else the value assigned by the operator.

1. To ex. the call

< program name >
will result in

```
PAR.LNAME   = ' < program name >'
PAR.INV     = ' $PTR <0> <0>'
PAR.COUN    = 20
PAR.BOOL    = ' < 255 >' ('YES')
PAR.TEXT    = ' <0> <0> <0> <0> <0> <0> <0> <0> <0> <0>'
```

as all default values are taken.

2. The call

< program name > MT 30 YES 'ABCD'.

will result in

```

PAR.LNAME      = ' < program name > '
PAR.INV        = ' MT <0> <0> <0> <0> '
PAR.COUN       = 30
PAR.BOOL       = ' < 255 > ' ( 'YES' )
PAR.TEXT       = ' ABCD <0> <0> <0> <0> <0> <0> '

```

3. and

< program name > MT OP.NO in

```

PAR.LNAME      = ' < program name > '
PAR.INV        = ' MT <0> <0> <0> <0> '
PAR.COUN       = 20                ! DEFAULT!
PAR.BOOL       = ' <0> '
PAR.TEXT       = ' <0> ..... <0> ' ! DEFAULT!

```

4. and

< program name > MT : 2 TX. 'QUERTY' in

```

PAR.LNAME      = ' < program name > '
PAR.INV        = ' MT <0> <0> <0> <2> ' ! UNIT IN BYTE 6!
PAR.COUN       = 20                ! DEFAULT !
PAR.BOOL       = ' < 255 > '      ! DEFAULT !
PAR.TEXT       = ' QUERTY <0> <0> <0> <0>'

```

If there is any conflict between the parameters typed by the operator and the names and types specified by the program, the procedure will not return, but output an error message on the console and finish the execution of the utility program.

The procedure declaration is

```
PROCEDURE GETPARAMS (DESC : STRING(1) ; INIT : STRING(1) ;  
                    VAR PARAMS : STRING(1)) ;
```

```
CODEBODY P0085;
```

UTILITY PROGRAM TERMINATION.

When the utility program has completed the job it is convenient to remove the process, and thereby free the core reserved.

This can be done by the procedure FINIS (P0084), which is called with a single integer parameter containing information about the result of the job.

This result is not used by the operating system, but can be used by other programs using internal commands for special jobs.

At present the simple convention exist:

```
RESULT < > 0 : job execution OK
```

and

```
RESULT = 0 : job execution not OK.
```

Declaration of procedure is:

```
PROCEDURE FINIS ( RESULT : INTEGER );  
CODEBODY P0084 ;
```

It is the programmers responsibility that all devices and files used during run are closed before use of this procedure.

FILE CONNECTION.

The use of utility programs is related to the disc system, and normally the data input or output by the program is placed on a discfile. However utility programs may be able to produce output or take input on non-disc devices (magtapes, paper tape, cards etc).

As these devices have different behaviour seen from the program all the characteristics connected to these devices can be placed in devicedescriptors, which are catalog entries created with utility program SET (consult the description for further information) and marked with a special attribute (entry only (1b13) and devicedescriptor (1b14)).

The zones used by the utility program can then be opened by use of the procedure CONNECTFILE (P0086). Three parameters are necessary in the call, they are:

The zone to open,
the mode in question and
the name identifying the filename or
the devicedescriptor name.

Declaration is then

```
PROCEDURE CONNECTFILE (FILE F; MODE: INTERGER;  
                        IDNAME: STRING(6) );
```

```
CODEBODY P0086;
```

The function of the procedure is:

1. The entry indentified by IDNAME is looked up in the catalog.
- 2a. If the entry found is a devicedescriptor the zone F is initialized with the kind, device name, giveup mask. The zone is opened with the or'ed value of parameter MODE and the mode found in the entry. Last the position is set according to filenumber and block-number found in the entry.
- 2b. If the entry is not a devicedescriptor, or the file is not existing the parameter MODE is examined. If mode is output an entry is created with the name = IDNAME, size = 1 and attribute = extendable.

Last the zonekind is set to disckind, and the zone is opened with the specified mode and positioned on block zero.

It should be mentioned that the buffersize must be 512 if discfiles are expected, and the giveup procedure can be called in connectfile (To example when the output disc file already exists).

The entry format of a devicedescriptor is:

	word no.
ENTRY NAME	+0
NOT USED	
ATTRIBUTE	+6
GIVEUPMASK	+7
0	
0	
DEVICE NAME	+10
DEVICE KIND	+13
DEVICE MODE	+14
FILENO* 256 + BLOCKNO	+15

FETCH OF SYSTEM ERROR TEXTS.

The DOMUS system has all standard error-texts placed on the file SSYSE, and the texts can be delivered on request from the operating system if specified by a text number.

Fetching of texts is done by sending the S-process a message with operation = 5, mess 3 = error number, and the rest of the message as a normal transput message with bytecount and byte address. [2]

Bytecount must be greater than 32.

In MUSIL this can be done by use of zones [3]. Imagine following declarations in the program:

VAR

```

ERR      :   FILE 'S', 1, 1, 100, U; ! PROCESS S !
          :   GIVEUP SERR, 8'177777 OF STRING(100);

I        :   INTEGER;
ERRNO    :   INTEGER;

```

Following procedure will then get the errortext defined by the integer ERRNO, and output the text on the operator device:

PROCEDURE SHOWERROR:

BEGIN

```

OPEN (ERR, 5);
ERR.Z BLOCK : = ERRNO; ! MESS3 SET TO ZBLOCK !
GETREC (ERR, I);
OPMESS (ERR†); ! DO NOT WORRY ON TEXT-!
CLOSE (ERR, 1); ! LENGTH AS LAST BYTE = Ø !

```

END;

The procedure can be used in case of any error if the error number is represented on the file SSYSE by a text, and the text-length is less than 100 bytes.

As seen from the comments the last byte delivered in the text is always zero, which makes it simple to use the OPMESS or OUTTEXT procedures.

INTERNAL COMMANDS.

If string

```
LOADCOM = 'LOAD MYCHILD <10>',
```

is added to the declarations in the last page, this command can be send to the S-process as if it was typed on the operator console, with the exception that the process will be the owner of the loaded process, and thereby protect the process from being removed by others.

The message send to the S-process must have operation = 3, and usual byte count and byteaddress as transput messages. In normal use MESS 3 must be zero. A further detailed description of the message and use of a non-zero mess 3 can be found in DOMUS, System Programmer's Guide [2] .

In case of error the message is returned, and the giveup procedure is called. Status is the S-error number *256. The numbers can be found in [1] , and the connected texts are placed in the error-text file identified by the same number.

The giveup procedure is simple if one use the procedure SHOWERR in previous section:

```
PROCEDURE SERR;
BEGIN
    ERRNO : = ERR.Z0 SHIFT (-8);
END;
```

Then the program sending the command LOAD MYCHILD can look like:

BEGIN

```
ERRNO := 0;
OPEN (ERR, 3);
ERR.ZBLOCK := 0;           ! NO ERROR NAME WANTED !
OUTTEXT (ERR, LOADCOM);
CLOSE (ERR, 1);           ! MESSAGE IS SEND !
IF ERRNO < > 0 THEN
    SHOWERR;             ! OUTPUT ERROR TEXT IN !
    .                   ! CASE OF ERROR !
    .
    .
```

The process MYCHILD is now loaded and running if ERRNO = 0 , and only removed if program sends a likewise command KILL <process>, or the owner program is removed by its owner.

APPENDIX B, STANDARD MESSAGES Rev. 01

- 0200 *** NOT ENOUGH ARGUMENTS
Non or too small a number of parameters has been assigned.
- 0201 *** UNIT NUMBER CONFLICT
The operation requested is only allowed within same catalog unit.
- 0202 *** ILLEGAL FILE SPECIFICATION
An illegal attribute type has been specified. Only B, P, W, F or V are allowed.
- 0203 *** COMMUNICATION ERROR WITH S
Due to errors the resources requested from S was not available.
- 0204 *** SHORT OF CORE STORAGE
There is not enough space in core to hold the work data.
- 0205 *** NON ASCII CHARACTER IN INPUT
Character with value > 127 found in input.
- 0206 *** UNITNUMBER GREATER THAN 255
Only unitnumbers 0 to 255 are defined in the system.
- 0207 *** UNIT NOT MOUNTED
The disc pack holding the requested catalog unit was not ready

- 0208 *** UNIT DOES NOT EXIST
The catalog unit to work on is not defined in the system.
- 0209 *** ILLEGAL UNITNUMBER
It is not allowed to overwrite the specified catalog unit or the unit is not defined in the system.
- 0210 *** ILLEGAL DISC SIZE
The disc size specified is too small to hold a catalog.
- 0211 *** NOT DEVICE DESCRIPTOR, ENTRY <name>
The entry <name> is not a device descriptor as attribute entry only and devicedescriptor is not set (bit 13 and bit 14)
- 0212 *** FILE OR BLOCK TOO LARGE
Maximum file or blocknumber in a devicedescriptor is 255.
- 2026 *** PUNCH RESERVED
The punch driver is used by another process.
- 2031 *** PAPER LOW ON PUNCH
Almost the whole paper roll has been used.
- 2033 *** PUNCH DRIVER NOT LOADED
The driver has not been loaded by the operator.
- 2034 *** PUNCH ERROR OR TIMEOUT
The character to output was not punched within a maximum time. Properly the paper has run out.

- 2041 *** STATION OFF-LINE, STATION <name>
The magtape station is or has been set local by
the operator.
- 2042 *** TAPE REWINDING, STATION <name>
The station is unable to operate as the tape is
rewinding.
- 2043 *** NOISE RECORD, STATION <name>
A noise record was met before the block read.
- 2045 *** WRITE LOCK, STATION <name>
The write enable ring is not mounted on the tape.
- 2046 *** ILLEGAL OPERATION, STATION <name>
In attempt to write on the tape has been rejected
as the writeenable-ring was not mounted.
- 2047 *** END OF FILE, STATION <name>
End-of-file mark read before logical end of data.
- 2048 *** BLOCK LENGTH ERROR, STATION <name>
The buffer size used was too small to hold the block
read. Data format error.
- 2049 *** DATA LATE, STATION <name>
Due to overload one or more characters was lost
during last transfer.
- 2050 *** PARITY ERROR, STATION <name>
The last block read/written has a parity error.
- 2051 *** END OF TAPE, STATION <name>
The read/write head is positioned after the EOT mark

- 2052 *** POSITION ERROR, STATION <name>
The position requested is not defined.
- 2053 *** DRIVER MISSING, STATION <name>
The driver is not loaded by the operator.
- 2054 *** TIMEOUT ERROR, STATION <name>
The station did not respond on the operation within
a maximum specified time.

APPENDIX C

Standard device descriptors in DOMUS system

* : Don't care.

NAME	DRIVER	FILE DEC.	BLOCK DEC.	MODE DEC.	KIND BIN.	MASK OCT.	COMMENT
\$ LPT	LPT	*	*	3	1	161777	unformatted
\$ SP	SP	*	*	3	1	161777	unformatted
\$ CPT	CPT	*	*	3	1	161777	unformatted
\$ PTR	PTR	*	*	9	1	1067	even parity
\$ PTRN	PTR	*	*	1	1	1067	no parity
\$ PTP	PTP	*	*	11	1	1027	even parity
\$ PTPN	PTP	*	*	3	1	1027	no parity
\$ CDR	CDR	*	*	9	10	161777	decimal with termination
\$ CDRN	CDR	*	*	1	10	161777	binary bytes
\$ TTY	TTY	*	*	1	1	0	TTY input and output
\$ MT0	MT0	1	1	1	1110	161777	magtape input/output
\$ CT0	CT0	1	1	1	1110	161777	cassette tape input/output

APPENDIX D, CONVERSION TABLE FORMAT.

The conversion tables used for standard conversion on lineprinters and initiated by program STACO must start in the sixth word after the program item head.

Tables can be made either in MUSIL code or assembler code.

In MUSIL-code the format is:

```
CONST
TABLE = # <table values> # ;
BEGIN
END;
```

and in assembler coding:

```
<program item head>
.BLK 5
TABLE START: <table values two bytes per word>
.END
```

Tables coded in this format may also be accessed from MUSIL programs using code-procedure CHANGETABLE, however an assembly coded table must then be provided with a process descriptor and some code stopping the process when started or broken.