Title:

DOMUS - Utility EXEC

User's Guide

A/S REGNECENTRALEN

RC SYSTEM LIBRARY: FALKONERALLE 1 DK-2000 COPENHAGEN F

RCSL No: 43-GL 6251 (P1)

Edition: February 1978

Author: Dan Holmer Andersen

Keywords:

RC3600, DOMUS, Utility, Batch Processor.

Abstract:

This manual is a user's guide for the DOMUS Batch Processor EXEC.

Printed by A/S Regnecentralen, Copenhagen

CONTENTS PAGE

This page is intentionally left blank.

1. INTRODUCTION. 1.

The EXEC program is the DOMUS batch processor module, which executes utility-program calls and S-functions in a sequential user defined sequence. Commands are supplied by the user as ASCII characters on a discfile or external device.

In contrary to the DOMUS INT(ERPRET) function each termination of a utility-program is awaited and the success of the call is checked. In case of error the whole job is aborted if specified so by the user in the EXEC call.

The command file is read to the internal workfile .XEC before interpretation, hereby releasing the input device for use in the job.

2. PARAMETER FORMAT.

The syntax of the call follows the general rules given in [2], and the parameters are:

EXEC IN.<comfile> LIST.<listfile> STOP.<boolean>

<comfile> is the file containing a number of lines with normal S-functions, internal commands or utility calls in the same format as used when entering commands to the S-process from the operator console.

The syntax check is left to the S-process, and the function of the EXEC program is mainly to split the file into single commands send for interpretation and execution at the S-process.

<listfile> a log of the commands is produced on this file. The file can not be used by programs loaded during the job.

<boolean> if equals YES a check is performed on each utility termination, and if the program reports a non-successful termination the whole job is terminated with an error message.

Default values of call are:

EXEC IN.$PTR STOP.NO

i.e. no log is produced.

3. COMMAND FILE SYNTAX. 3.

The only check performed by the EXEC program is a check on the identifier found as the first item on each line, and a final check is left to the operating system and the programs loaded.

The syntax is then the one described in [1] and [2], and the only operation done by the EXEC module is the line separation, as the lines are handed over to the S-process for execution line by line.

However, following exceptions exists:

- BEGIN and END commands are blind and the rest of the line is skipped, hereby left for comments.

- A DRIVE function implies that following program-loads are performed form the catalog unit given as parameter, until a new DRIVE funtion is found, and the END function will not reset the drive number to zero.

- Comments can be given anywhere in the command file as !<any number of characters except!>!. The whole comment is skipped and replaced by a single space, but transferred to the listfile unchanged. Newline characters in a comment string have no effect.

- LIST functions are ignorred by the S-process.

- Each line must not exceed 500 characters in length, including comments which are counted as one character.

- Characters which are interpreted as blind or illegal by the S-process are skipped.

- Tabulation characters are converted to spaces on the listfile.

The action taken by the EXEC module depends on the identifier found as the first item on each line, and is given in the below table. Internal functions can be viewed on as utility functions, although no load is performed.

BEGIN — The line is ignored by the EXEC program.

BOOT — The line is send to the S-process for execution. If the command is accepted, the DOMUS system, including the EXEC-program, is overwritten and properly termination of log output or job can not be expected.

CLEAN — The line is send to the S-process.

CLEAR — The line is send to the S-process.

DRIVE — The drive-number specified as parameter will be the catalog unit from where the following program loads are performed. Note that the drive-number is not changed even if an END command is found in contrary to normal use of console entered commands.

Before a DRIVE command or a load from a catalog unit different from zero is used, an INIT command should be given. Only decimal drive numbers are allowed.

END — The command is ignored by the EXEC program.

FINISH — This is an internal command. Job execution is terminated, and the EXEC program is removed from core, and all processes loaded during run is removed too by the operating system.

This command must be the last in the command file.

FREE The line is send to the S-process.

GET The line is send to the S-process.

INIT The line is send to the S-process.

INT The line is send to the S-process.

KILL The line is send to the S-process.
The process to kill must not have messages pending at the EXEC process, else this is breaked with breakcause 1.

LIST Ignored by the operating system.

LOAD The line is send to the S-process.
LOAD function can be used for utility program loads like

LOAD (DELETE PIP)

but the termination of the utility program is not awaited. The termination messages will, however, be granted and the utility program is removed when the message is send. If the utility function has not terminated when the EXEC program finds a FINISH command, it is removed unconditionally.

PAUSE This is an internal command. A number sign (#) is output on the operator console, and the EXEC process waits for operator intervention. If operator inputs STOP the EXEC process will stop execution as if a FINISH command was met, and outputs the message 'EXECUTION STOPPED BY OPERATOR', else the execution is continued.

START The line is send to the S-process.

STOP The line is send to the S-process.

<any other name> The line is send to the S-process as a utility program load function. A termination message from the utility module is awaited, and if this indicates an unsuccessful run the execution is aborted if STOP.YES is specified, else job execution is continued. If the job is aborted the text 'UTILITY PROGRAM ERROR, PROGRAM <name> is output on the operator console.

Note:
Do not load other modules than utility programs in this way, as a missing termination message will hang up EXEC for ever.

4. OPERATOR COMMUNICATION.

The EXEC program accepts input from operator during run.

If operator inputs STOP the execution of the command-file is aborted and the program outputs

EXECUTION STOPPED BY OPERATOR.

If a PAUSE command is met a numbersign (#) is output, and the program waits for input. If operator inputs STOP the execution is aborted as above explained, else the execution is continued.

Example (input marked with _):

```
@G
>EXEC↵
STOP↵
EXECUTION STOPPED BY OPERATOR
>S
FINIS EXEC
```

5. CHAINED PROGRAMS.

When utility programs sends termination messages, this messages can hold a string of commands, which is requested to be interpreted and executed when removal of the program from core has taken place.

The EXEC module interprets such commands as if they were present in the operator specified command file after the previous executed line, and the commands must therefore obey the rules given in section 3. The list file output is ignored during execution of these commands, and they are invisible to operator, as if the requesting program was still running, and can be repeated any number of times, only limited by disc space.

Example:

Command file	PROG1 request	PROG2 request	PROG3 request
PROG1			
	PROG2		
		PROG4	
		PROG5	
	PROG3		
			PROG6

Resulting sequence of program calls:

PROG1
PROG2
PROG4
PROG5
PROG3
PROG6

} Not listed on log.

If the file TESTC contains the text:

```
CHART ASC1
EDIT  ASC1                      ! UNPROTECT, EDIT AND !
CHATR ASC1 PW                   ! PROTECT SOURCE 1    !
PAUSE                           ! WAITING FOR READER  !
COPY $PTR ASC2                  ! READ   ,EDIT AND    !
EDIT  ASC2                      ! PROTECT SOURCE 2
CHART ASC2 PW
DOMAC LIST.$LPT  BIN.R1 ASC1
DELETE XXX
DOMAC LIST.XXX   BIN.R2 ASC2
PRINT XXX
PRINT XXX                       ! PRODUCE TWO LISTING OF ASC2 !
DELETE TEST
LINK TEST  LOG.$LPT FORM.R  IN.R1 R2
TEST                            ! LOAD THE PROGRAM     !
FINISH
```

the call EXEC TESTC

will call the Editor twice, assemble the two sources and load the resulting program. Listing of the code is produced for both modules, with two copies of the source read from paper tape.

This page is intentionally left blank.

APPENDIX A - REFERENCES.

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

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