RCSL No: 43-GL10639

Edition: October 1980

Author: Jørgen Hansen

Title:

DOMUS Utility Program
Programming Guide

A/S REGNECENTRALEN
af 1979

Keywords:

RC3600, DOMUS, MUSIL, Manual.

Abstract:

This manual gives information to the MUSIL programmer about programming DOMUS utility programs.

(30 printed pages)

Printed by A/S Regnecentralen af 1979, Copenhagen

42-i 1341

TABLE OF CONTENTS PAGE

1. INTRODUCTION

1.

This manual gives some information to the MUSIL programmes about the conventions for DOMUS utility programs.

The reader must be familiar to the MUSIL language.

2. DOMUS UTILITY PROGRAMS 2.

A DOMUS utility program is an ordinary MUSIL program, which fulfils certain conditions:

1) Runtime parameters are transferred to the program in connection with the call of the program.

2) Use of peripheral devices is done by means of device descriptors, which are catalog entries containing information about device name, giveupmask, kind, mode, file, and block.

3) Whenever output to a printer is produced, the program will start with a 'top of form' command. A 'top of form' is not output, when the printout is finished.

4) When a program terminates its execution, all resources used (i.e. workfiles, area processes) must be released.

5) When the program has finished its execution, it sends an internal request to the father process in order to be removed from memory.

6) Any runtime error during program will result in fetch of an error message from the DOMUS message file. This text is output to the operator console, and the program execution is terminated. This means that interactive operator communication is not used.

In order to ease the programming, a number of codeprocedures have been developed. They are described in chapter 3 of this manual.

3. CODEPROCEDURES 3.

3.1 Getparams 3.1

Declaration:

```
procedure GETPARAMS (    descr:   string (1);
                         init:    string (1);
                         par:     string (1);
                         catalog: string (6);
                     var key:     integer);
codebody p0260;
```

Place the parameters to the utility program in the string 'par'. The parameters are syntax checked according to the string 'descr'. If a parameter is not specified, a default value is fetched from the string 'init'. Furthermore the name of the current catalog and the key that has been used in the CONNECT-command are delivered in the string 'catalog' and the integer 'key'.

The constant string 'descr' describes the parameters and must meet the following syntax:

```
DESCR= '<5 bytes parameter name><type byte>
       .
       .
       .
       .
       <5 bytes parameter name><type byte>
       <255>',
```

In case a parameter name is less than 5 characters, zero bytes must be inserted in the end of the parameter name. The 'type byte' describes the type of the actual parameter described by the name placed in front of the line.

The last byte in the description string is 255, terminating the parameter list.

Five values of parameter types exist:

<128 text (STRING(value))
129 Boolean (STRING (1))
The string equals <255> if parameter is yes, and <0>
if parameter is no.
130 Integer (INTEGER)
134 Name (STRING (6))
140 Filename (STRING (12))

or as an example:

```
CONST

DESCR= 'TX<0><0><10>
       COUNT<130>
       DEV<0><0><134>
       IN<0><0><0><140>
       OP<0><0><0><129>
       <255>',
```

Here five parameters are defined.

The first parameter has the name TX and is defined as a string of length 10 bytes.
The second parameter has the name COUNT and is defined as an integer.
The third parameter has the name DEV and is defined as a name, which is contained in a string of length 6 bytes.
The fourth parameter has the name IN and is defined as a filename, which is contained in a string of length 12 bytes.
The fifth parameter has the name OP and is defined as a boolean.

Parameter type 'filename' has three possible formats, which will be stored in the string in the MUSIL program as described below. Not used positions in the string are set to binary zero.

1) <name>
   The name is stored from the seventh byte and on.
2) <name>:<number>
   The name is stored from the seventh byte and on. The value of the number, which must not exceed 255 is inserted in the twelfth byte.
3) <name>/<name>
   The first name is stored from the first byte and on, and the second name is stored from the seventh name and on.

This parameter type is normally used to point out a certain catalog entry and then has the following interpretation:

```
Byte 1-5  : Catalog name (if not specified, the main catalog)
Byte 6    : Not used
Byte 7-11 : Entry name
Byte 12   : Unit number (if not specified, zero)
```

It is not possible to specify both a catalog name and a unit number.

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

Example of definition of default values:

```
INIT= 'HEADLINE<0><0>',
      <2><128>
      MT0<0><0><0>
      <0><0><0><0><0><0><$PTRN<0>
      <255>',
```

The parameters are the same as defined above.

The default values are:

```
TX:     'HEADLINE<0><0>'
COUNT:  640         (= 2*256+128)
DEV:    MT0
IN:     $PTRN
OP:     YES
```

Note that the default value of the parameter COUNT must be computed from two bytes, the first byte giving the 8 leftmost bits, and the second giving the 8 rightmost bits.

The 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 built as a record in MUSIL, and must as the first element contain a string (6), which is set to the loadname of the utility program.

Example:

```
VAR
PAR:        RECORD
                LNAME:      STRING(6);
                TX:         STRING(10);
                COUNT:      INTEGER;
                DEV:        STRING(6);
                INCAT:      STRING(6);
                INNAME:     STRING(5);
                INUNIT:     STRING(1);
                OP:         STRING(1);
            END;
```

Notice, that the parameter IN has been split into three smaller pieces to ease the access to the fields.

The fourth parameter is a variable string of six bytes, in which the name of the current catalog at call time is delivered. If no CONNECT-command has been executed, the name will be '<0><0><0>.

The fifth parameter is an integer, which after call contains the value of the protection key given in the last CONNECT-command. If no key was given or if current catalog is the main catalog, the value is zero.

3.2 Connectfile 3.2

Declaration:

```
Procedure CONNECTFILE (file f;
                       const mode: integer;
                             ident: string(2);
                       var  key:   integer);
codebody p0261;
```

This codeprocedure looks up the catalog entry given by the call parameter 'ident', which consists of 12 bytes, where the first 6 bytes is the name of a catalog, the next 5 bytes is an entry name, and the last byte is a unit number. If the catalog name is empty (consists of 6 binary zeroes), the main catalog is used. If a catalog name is specified, the unit number must be zero.

Depending of the entry and the parameter 'mode', different things will happen.

1) The entry is an ordinary disc file, and the mode is input:
   The disc file will be opened with the given mode, kind=8'36 and positioned at first segment.

2) The entry does not exist, and the mode is output:
   A disc file with the given name will be created in the given catalog. The file will be opened in the given mode, kind=8'36 and positioned at first segment.

   In this case the parameter 'key' is used. It must be set to the key of the used catalog (key:=0 if main catalog is used).

In case of incorrect key, the MUSIL giveup procedure will be called with 1b3+1b6.

3) The entry does not exist, and the mode is input:
The giveup procedure in the MUSIL program is called.

4) The entry is an ordinary disc file, and the mode is output:
The MUSIL giveup procedure is called with 1b3+1b11 in order to prevent overwrite of existing data.

5) The entry is a devicedescriptor (attribute bit 13 and 14 set):
The information of device name, mode, kind, file, block, and giveupmask is transferred from the entry to the zone variables. The mode of the zone is set to the or'ed value of the mode specified in the call and the mode from the entry.

Then the zone is opened in this mode and positioned at the given file and block.
The format of a devicedescriptor is:

Byte 0-5	Entry name
6-11	Optional 1 (Reserved for future use)
12-13	Attribute = 2'110 (+ evt. permanent, writeprotect)
14-15	Giveupmask
16-17	Segment number (set by CAT)
18-19	Reserved length (set by CAT)
20-25	Device name
26-27	Device kind
28-29	Device mode
30	Device file
31	Device block

An error in one of the catalog operations performed will result in a call of the giveup procedure in the MUSIL program. If at this time zone.zname is empty, the error had occurred at the attempt to make a call of NEWCAT. Otherwise the error comes from LOOKUPENTRY, CREATEENTRY, OPEN, or SETPOSITION.

If the giveup procedure is called, the zone is left in a neutral state.

Note: The parameter 'key' is always changed at return from the procedure.

3.3 Splitshare 3.3

Declaration:

```
procedure SPLITSHARE (file f);
codebody P0262;
```

This procedure divides the share of a zone into four to get multibuffering.

The zone must be declared with only one share with sharelength 512 bytes. A call of this procedure will change the share into four new shares with sharelength 90 bytes. Four extra message buffers will be allocated too.

The division will only take place if the zone is declared with kind bit 15 (character oriented) set, otherwise a call of the procedure has no effect.

The procedure is to be used in DOMUS Utility Programs, just after a call of CONNECTFILE.

3.4 Getname 3.4

Declaration:

```
procedure GETNAME (catalog: string(6);
                   name:    string(6);
                   text:    string(18);
codebody p0263;
```

The procedure builds a text string in parameter 'text' from the two call parameters 'catalog' and 'name'. The procedure can be used in connection with procedure Geterror as an easy way to write error messages on the operator console.

If parameter 'catalog' contains anything different from binary zero in the first byte, the catalog name is inserted in the string 'text' followed by a slash (/). After this the first five bytes of parameter 'name' are inserted. If the sixth byte of 'name' is nonzero, a colon (:) is inserted followed by the value of this byte as three decimal digits.

The unused part of string 'text' is filled with binary zeroes.

Examples:

catalog	name	text
<0><0><0><0><0><0>	PIP<0><0><0>	PIP
<0><0><0><0><0><0>	PIP<0><0><1>	PIP:001
SUB<0><0><0>	PIP<0><0><0>	SUB/PIP
SUB<0><0><0>	PIP<0><0><1>	SUB/PIP:001 not used

3.5 Geterror 3.5

Declaration:

```
procedure GETERROR (file f;
                    const error: integer
                          text:  string(1));
codebody p0264;
```

This procedure will fetch a message from the DOMUS message file and output it on the operator console followed by the text string given in parameter 'text'.

The zone must be declared with sharelength 512 bytes. The zone variables will be destroyed.
The parameter 'error' is a call parameter defining the DOMUS message number.
The string 'text' must be terminated by a binary zero byte. It might be produced by codeprocedure Getname.

3.6 Finis 3.6

Declaration:

```
procedure FINIS (const result: integer);
codebody p0084;
```

A call of this procedure will result in a removal of the calling process, and thereby a release of the core area reserved. The calling process is defined as the one which executes the call of FINIS. The procedure must be called, when the utility program has terminated its execution and released all its ressources. The call parameter 'result' is used to give information to the father process about the job execution. A simple convention exists:

result <> 0 : job execution OK
result = 0 : job execution not OK

This information is not used by the DOMUS operating system, but it is used by other programs using internal commands (e.g. EXEC).

A. REFERENCES

1. RCSL No
DOMUS User's Guide, Part 1.

2. RCSL No
DOMUS System, Programmer's Guide.

B. EXAMPLE OF A DOMUS UTILITY PROGRAM

B

```
MUSIL COMPILER 11
PRINT


0001
0002 !                                                  RCSL:    43-GL9837
0003                                                    AUTHOR:  DHA/JHA
0004                                                    EDITED:  80.01.28
0005
0006
0007
0008
0009
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
0020
0021
0022
0023
0024                            PROGRAM RC36-00560.02
0025                            DOMUS UTILITY PRINT
0026
0027
0028
0029
0030
0031
0032
0033
0034
0035
0036
0037
0038
0039
0040
0041
0042
0043
0044
0045
0046 KEYWORDS:          MUSIL,DOMUS,UTILITY,FILE PRINT
0047
0048 ABSTRACT:          THIS DOMUS UTILITY PROGRAM PRINTS AN ASCII FILE
0049                    ON ANY DEVICE. LINENUMBERS EQUIVALENT TO NUMBERS
0050                    PRINTED BY MUSIL COMPILER CAN BE OUTPUT IN FRONT
0051                    OF EACH LINE.
0052
0053 RCSL: 43-GL9836: ACSII SOURCE
0054 RCSL: 43-GL9838: REL.BINARY
0055 !
0056
```

```
0057
0058 !
0059
0060 DOMUS UTILITY:          PRINT
0061
0062 FORMAT:                 PRINT IN.<FILE1> LINE.<BOOLEAN> OUT.<FILE2>
0063
0064 FUNCTION:               FILE <FILE1> IS OUTPUT ON FILE <FILE2>
0065                         (NORMALLY THE LINEPRINTER). THE CHARACTER
0066                         TAb (ASCII VALUE 9) IS CONVERTED TO SPACES.
0067                         IF LINE = YES, LINENUMBERS ARE OUTPUT IN
0068                         FRONT OF EACH LINE AS A FOUR DIGIT NUMBER.
0069                         LINENUMBERS ARE EQUIVALENT TO NUMBERS
0070                         PRINTED BY MUSIL COMPILER.
0071
0072 PARAMETERS:             FIRST PARAMETER IS THE NAME OF THE FILE
0073                         TO BE PRINTED. SECOND PARAMETER IS A
0074                         BOOLEAN WITH VALUE YES OR NO SPECIFYING
0075                         IF LINENUMBERS SHOULD BE PRINTED. THIRD
0076                         PARAMETER IS THE OUTPUT FILE.
0077
0078                         DEFAULT VALUES ARE:
0079
0080                             PRINT IN.$PTR LINE.NO OUT.$LPT
0081
0082 EXAMPLES:               PRINT PIP OUT.$SP
0083
0084                             FILE PIP IS PRINTED ON DEVICE $SP.
0085
0086                         PRINT PAP:1 YES
0087
0088                             FILE PAP ON CATALOG UNIT 1 IS PRINTED ON
0089                             DEVICE $LPT WITH LINENUMBERS.
0090
0091                         PRINT POP/PIP
0092
0093                             FILE PIP IN SUBCATALOG POP IS PRINTED ON
0094                             DEVICE $LPT.
0095
0096 CORE REQUIREMENTS:      4600 BYTES DECIMAL.
0097
0098 OTHER REQUIREMENTS:     DEPENDING ON PRINTER-DRUM STANDARD CONVERSION
0099                         CAN BE USED BY CONNECTING A CONVERSION TABLE
0100                         TO THE PRINTER DRIVER WITH PROGRAM STACO.
0101                         NOTE:
0102                         IT IS RECOMMENDED TO USE PROGRAM COPY FOR
0103                         DATA TRANSFER TO NON PRINTER FILES.
0104 !
0105
```

```
0106
0107                                                    ! RC36-00560 PAGE 02 !
0108
0109 CONST
0110
0111 PROGID=          '800128 S60.02',
0112
0113 DESCR=           'IN<0><0><0><140>
0114                   LINE<0><129>
0115                   OUT<0><0><140>
0116                   <255>',
0117
0118 INIT=            '<0><0><0><0><0><0>
0119                   $PTR<0><0>
0120                   <0>
0121                   <0><0><0><0><0><0>
0122                   $LPT<0><0>',
0123
0124 EMPTY=           #
0125                   0 0 0 0 0 0
0126                   #,
0127
0128 HEADTXT=         '*** PRINT OF FILE ',
0129
0130 SYSER=           '*** SYSTEM ERROR ',
0131
0132 NL=              10,
0133 FF=              12,
0134 CR=              13,
0135 TAB=             9,
0136 SP=              32,
0137 EM=              25,
0138 NO=              '<0>',
0139 TRUE=            1,
0140 FALSE=           0,
0141 REMOVE=          2,
0142 REMOVEERR=       3;
0143
```

```
0144
0145                                                    ! RC36-00560 PAGE 03 !
0146
0147 VAR
0148
0149 PAR:               RECORD
0150                        PROG:       STRING(6);
0151                        INCAT:      STRING(6);
0152                        INNAME:     STRING(6);
0153                        LINE:       STRING(1);
0154                        OUTCAT:     STRING(6);
0155                        OUTNAME:    STRING(6)
0156                    END;
0157
0158 ACTION:            INTEGER;
0159 CHAR:              INTEGER;
0160 LASTCHAR:         INTEGER;
0161 TABVAL:           INTEGER;
0162 LINENO:           INTEGER;
0163 FLAG:             INTEGER;
0164 LREC:             RECORD
0165                        FIRST:      STRING(1);
0166                        LAST:       STRING(6)
0167                    END;
0168 NAME:             STRING(18);
0169 HELPSTRING:       STRING(10);
0170
0171 CATALOG:          STRING(6);
0172 KEY:              INTEGER;
0173 INID:             RECORD
0174                        CAT:        STRING(6);
0175                        NAME:       STRING(6);
0176                        UNIT:       STRING(1) FROM 12
0177                    END;
0178 OUTID:            RECORD
0179                        CAT:        STRING(6);
0180                        NAME:       STRING(6);
0181                        UNIT:       STRING(1) FROM 12
0182                    END;
0183 OUTKEY:           INTEGER;
0184
0185 CREATEFLAG:       INTEGER;
0186 ERROR:            INTEGER;
0187 ERRORCAT:         STRING(6);
0188 ERRORNAME:       STRING(6);
0189 STATUS:           INTEGER;
0190
0191 IN:               FILE '<0><0>',30,1,512,U;
0192                    GIVEUP INERROR, 8'177777
0193                    OF STRING(512);
0194
0195 OUT:              FILE '<0><0>',4,1,512,U;
0196                    GIVEUP OUTERROR, 8'177777
0197                    OF STRING(512);
0198
```

```
0199
0200                                                        ! RC36-00500 PAGE 04 !
0201
0202 PROCEDURE GETPARAMS      (CONST DESCR:          STRING(1);
0203                           CONST INIT:           STRING(1);
0204                           VAR   PAR:            STRING(1);
0205                           VAR   CATALOG:        STRING(6);
0206                           VAR   KEY:            INTEGER);
0207 CODEBODY P0260;
0208
0209
0210 PROCEDURE CONNECTFILE    (FILE  F;
0211                           CONST MODE:           INTEGER;
0212                           CONST NAME:           STRING(12);
0213                           VAR   KEY:            INTEGER);
0214 CODEBODY P0261;
0215
0216
0217 PROCEDURE TAKEADDRESS    (CONST NAME:          STRING(8);
0218                           VAR   ADDR:           INTEGER);
0219 CODEBODY P0159;
0220
0221
0222 PROCEDURE SPLITSHARE     (FILE  F);
0223 CODEBODY P0262;
0224
0225
0226 PROCEDURE GETTIME.       (VAR   TIME:          STRING(8));
0227 CODEBODY P0149;
0228
0229
0230 PROCEDURE GETDATE        (VAR   DATE:          STRING(8));
0231 CODEBODY P0150;
0232
0233
0234 PROCEDURE GETNAME        (CONST CATALOG:       STRING(6);
0235                           CONST NAME:           STRING(6);
0236                           VAR   TEXT:           STRING(18));
0237 CODEBODY P0263;
0238
0239
0240 PROCEDURE GETERROR       (FILE  F;
0241                           CONST ERROR:          INTEGER;
0242                           CONST TEXT:           STRING(18));
0243 CODEBODY P0264;
0244
0245
0246 PROCEDURE FINIS          (CONST RESULT:        INTEGER);
0247 CODEBODY P0084;
0248
0249
0250 PROCEDURE SETERROR;
0251     BEGIN
0252         IF STATUS AND 8'010000 <> 0 THEN ERROR := 100
0253             ELSE IF STATUS AND 8'004000 <> 0 THEN ERROR := 120
0254                 ELSE ERROR := 2000;
0255         STATUS := STATUS AND 8'163777;
0256         WHILE STATUS > 0 DO
0257             BEGIN
0258                 ERROR := ERROR + 1;
0259                 STATUS := STATUS SHIFT 1;
0260             END;
0261     END;
0262
0263
0264
```

```
0265
0266                                                    ! RC36-00560 PAGE 05 !
0267
0268 PROCEDURE INERROR;
0269 BEGIN
0270     IF ERROR = 0 THEN
0271     BEGIN
0272         IF IN.Z0 AND 8'167357 <> 0 THEN
0273             IF IN.Z0 <> 8'4020 THEN
0274             BEGIN
0275                 STATUS := IN.Z0;
0276                 SETERROR;
0277                 IF IN.ZNAME <> EMPTY THEN
0278                 BEGIN
0279                     IF PAR.INCAT = INID.CAT THEN ERRORCAT := PAR.INCAT;
0280                     ERRORNAME := PAR.INNAME;
0281                 END
0282                 ELSE
0283                 BEGIN
0284                     ERRORNAME := INID.CAT;
0285                     INSERT (0,ERRORNAME,5);
0286                 END;
0287             END;
0288         IF IN.ZREM = 0 THEN
0289         BEGIN
0290             IN.ZFIRST := IN.ZTOP;
0291             IN.ZREM := 1;
0292             INSERT (EM,IN↑,0);
0293         END;
0294     END
0295     ELSE
0296     BEGIN
0297         OPMESS (SYSER);
0298         BINDEC (IN.ZFILE,CATALOG);
0299         OPMESS (CATALOG);
0300         FINIS (FALSE);
0301     END;
0302 END;
0303
0304
0305
0306 PROCEDURE OUTERROR;
0307 BEGIN
0308     IF ERROR = 0 THEN
0309     BEGIN
0310             STATUS := OUT.Z0;
0311             SETERROR;
0312             IF OUT.Z0 AND 8'11000 = 8'11000 THEN
0313                 IF OUTKEY = 0 THEN ERROR := 143;
0314             IF OUT.ZNAME <> EMPTY THEN
0315             BEGIN
0316                 ERRORNAME := PAR.OUTNAME;
0317                 IF PAR.OUTCAT = OUTID.CAT THEN ERRORCAT := OUTID.CAT;
0318             END
0319             ELSE
0320             BEGIN
0321                 ERRORNAME := OUTID.CAT;
0322                 INSERT (0,ERRORNAME,5);
0323             END;
0324     END
0325     ELSE
0326     ACTION := REMOVEERR;
0327     CHAP := EM;
0328 END;
0329
```

```
0330
0331                                                    ! VC36-00560 PAGE 06 !
0332
0333 PROCEDURE OUTHEAD;
0334     BEGIN
0335         OUTCHAR (OUT,FF);
0336         OUTTEXT (OUT,HEADTXT);
0337         GETNAME (PAR.INCAT,PAR.INNAME,NAME);
0338         OUTTEXT (OUT,NAME);
0339         OUTCHAR (OUT,SP);
0340         GETDATE (HELPSTRING);
0341         INSERT (0,HELPSTRING,8);
0342         OUTTEXT (OUT,HELPSTRING);
0343         OUTCHAR (OUT,SP);
0344         GETTIME (HELPSTRING);
0345         INSERT (0,HELPSTRING,8);
0346         OUTTEXT (OUT,HELPSTRING);
0347         OUTCHAR (OUT,NL);
0348         OUTCHAR (OUT,NL);
0349     END;
0350
```

```
0351
0352                                                    ! RC36-00560 PAGE 07 !
0353
0354
0355 PROCEDURE NEWLINE;
0356     BEGIN
0357         IF OUT.ZKIND EXTRACT 1 = 1 THEN OUTBLOCK (OUT);
0358         IF PAR.LINE <> NO THEN
0359             BEGIN
0360                 BINDEC (LINENO,LREC);
0361                 INSERT (SP,LREC,0);
0362                 OUTTEXT (OUT,LREC);
0363                 LINENO := LINENO + 1;
0364             END;
0365         OUTCHAR (OUT,SP);
0366         TABVAL := 0;
0367         FLAG := FALSE;
0368     END;
0369
0370
0371 PROCEDURE COPYPROC;
0372     BEGIN
0373         OUTHEAD;
0374         NEWLINE;
0375         LASTCHAR := 0;
0376         REPEAT
0377             INCHAR (IN,CHAR);
0378             IF CHAR >= 32 THEN
0379                 BEGIN
0380                     IF CHAR < 127 THEN
0381                         BEGIN
0382                             IF FLAG = TRUE THEN NEWLINE;
0383                             OUTCHAR (OUT,CHAR);
0384                             TABVAL := TABVAL + 1;
0385                         END;
0386                 END
0387                 ELSE
0388                 BEGIN
0389                     IF CHAR = TAB THEN
0390                         BEGIN
0391                             IF FLAG = TRUE THEN NEWLINE;
0392                             REPEAT
0393                                 OUTCHAR (OUT,SP);
0394                                 TABVAL := TABVAL + 1
0395                             UNTIL TABVAL AND 7 = 0;
0396                         END;
0397                     IF CHAR = FF THEN
0398                         BEGIN
0399                             IF FLAG = TRUE THEN NEWLINE;
0400                             OUTCHAR (OUT,FF);
0401                             FLAG := TRUE;
0402                         END;
0403                     IF CHAR = NL THEN
0404                       IF LASTCHAR <> CR THEN
0405                         BEGIN
0406                             IF FLAG = TRUE THEN NEWLINE;
0407                             OUTCHAR (OUT,NL);
0408                             FLAG := TRUE;
0409                         END;
0410                     IF CHAR = CR THEN
0411                         BEGIN
0412                             IF FLAG = TRUE THEN NEWLINE;
0413                             OUTCHAR (OUT,NL);
0414                             FLAG := TRUE;
0415                         END;
0416                 END;
0417                 LASTCHAR := CHAR
0418         UNTIL CHAR = EM;
0419         OUTCHAR (OUT,NL);
```

```
0422
0423                                                    ! RC36-00560 PAGE 08 !
0424 BEGIN
0425     ERROR := 0;
0426     ERRORCAT := EMPTY;
0427     ERRORNAME := EMPTY;
0428     LINENO := 1;
0429     CREATEFLAG := FALSE;
0430     ACTION := FALSE;
0431     GETPARAMS (DESCR,INIT,PAR,CATALOG,KEY);
0432     IF PAR.OUTCAT = EMPTY THEN
0433     BEGIN
0434         OUTID.CAT := CATALOG;
0435         OUTKEY := KEY;
0436     END
0437     ELSE
0438     BEGIN
0439         OUTID.CAT := PAR.OUTCAT;
0440         OUTKEY := 0;
0441     END;
0442     OUTID.NAME := PAR.OUTNAME;
0443     IF BYTE OUTID.UNIT <> 0 THEN OUTID.CAT := EMPTY;
0444     IF PAR.INCAT = EMPTY THEN INID.CAT := CATALOG
0445     ELSE INID.CAT := PAR.INCAT;
0446     INID.NAME := PAR.INNAME;
0447     IF BYTE INID.UNIT <> 0 THEN INID.CAT := EMPTY;
0448     IF ERROR = 0 THEN
0449     BEGIN
0450         KEY := OUTKEY;
0451         CONNECTFILE (OUT,3,OUTID,OUTKEY);
0452         IF ERROR = 0 THEN
0453         BEGIN
0454             CREATEFLAG := TRUE;
0455             IF OUT.ZBLOCK = 512 THEN SPLITSHARE (OUT);
0456             STATUS := 0;
0457             CONNECTFILE (IN,1,INID,STATUS);
0458             IF ERROR = 0 THEN
0459             BEGIN
0460                 SPLITSHARE (IN);
0461                 COPYPROC;
0462                 CLOSE (IN,1);
0463             END;
0464             CLOSE (OUT,1);
0465         END;
0466     END;
0467     IF ERROR = 0 THEN FINIS (TRUE);
0468
```

```
0469
0470                                                    ! RC36-00560 PAGE 09 !
0471
0472      IF CREATEFLAG = TRUE THEN
0473          IF OUT.ZKIND SHIFT 11 < 0 THEN
0474          BEGIN
0475              ACTION := REMOVE;
0476              TAKEADDRESS (OUTID.CAT,OUT.ZCONV);
0477              NEWCAT (OUT,KEY);
0478              IF ACTION <> REMOVEERR THEN
0479              BEGIN
0480                  REMOVEENTRY (OUT);
0481                  FREECAT (OUT);
0482              END;
0483          END;
0484      GETNAME (ERRORCAT,ERRORNAME,CATALOG);
0485      GETERROR (IN,ERROR,CATALOG);
0486      FINIS (FALSE);
0487  END;
```

RETURN LETTER

Title: DOMUS Utility Programs Programming Guide

RCSL No.: 43-GL10639

A/S Regnecentralen af 1979/RC Computer A/S maintains a continual effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback, your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability, and readability:

Do you find errors in this manual? If so, specify by page.

How can this manual be improved?

Other comments?

Name: _____ Title: _____

Company: _____

Address: _____

Date: _____

Thank you

42-i 1288

..... Fold here .....

..... Do not tear - Fold here and staple .....

Affix
postage
here

REGNECENTRALEN
af 1979
Information Department
Lautrupbjerg 1
DK-2750 Ballerup
Denmark