A/S REGNECENTRALEN

RC SYSTEM LIBRARY: FALKONERALLE 1 · DK-2000 COPENHAGEN F

RCSL No: 43-Ri0363

Edition: 76.06.10

Author: Per Nielsen

Title: Data Entry Supervisor Programming Guide

Keywords: Data Entry, Supervisor Programming Guide.

Abstract: Programming Rules, Description of Standard Code Procedures.

43-Ri0363
76.06.10
Per Nielsen

Data Entry Supervisor Programming Guide

Data Entry, Supervisor Programming Guide.

Programming Rules, Description of Standard Code Procedures.

Data Entry Supervisor Programming Guide

1. Introduction

All supervisor programs must be coded in the MUSIL programming language and compiled using the MUSIL Compiler. Output from the MUSIL Compiler is a relocatable binary object code which can be added to the Data Entry System by loading it to disc (see section 4). When the Data Entry System receives a supervisor command, it looks up the corresponding entry in the RC3600 file system catalog, loads the program to the supervisor area in memory, starts the program as a process, and sends a message containing the command line typed by the keying operator to the process.

The standard code procedures described in section 2 has been implemented in order to make it possible for the Data Entry supervisor to transfer command lines and parameters to the supervisor programs, and to get receipts from the programs before removal of the area processes after program termination.

These procedures must be declared and called in the MUSIL program, and copied into the program at compilation time. This is done when the compiler message "load nnnnn" appears on the RC3600 console device, where "nnnnn" is the name of the code procedure to be loaded.

2. Description of standard code procedures

2.1 Code procedure get command

This procedure is used to receive a command line from the supervisor in the Data Entry System.

The call of this procedure must be the first statement executed in the supervisor program.

The procedure must be declared as follow:

```
procedure cmmd (var comline: string(112);
                var return:  integer);
codebody;
```

Parameters:

Comline:	Return parameter of type "string". The length of the parameter must be at least 112 bytes. The parameter contains a command line, typed in by the keying operator, and must not be changed by the program.
Return:	Return parameter of type "integer". The parameter contains the message address used by the supervisor. This parameter must be used when returning to the supervisor (see section 2.3), and must therefore not be changed by the program.

After a syntax check, made by the supervisor, each parameter to the program will be packed in "comline" as groups of information as follow:

1. <u>Type of parameter</u>

   2 bytes; where 0 = integer parameter
   1 = text parameter
   2 = termination

2. <u>The parameter</u>

   2 bytes for integer parameters, and
   6 bytes for text parameters, each parameter terminated by at least one null character

3. <u>Terminator for parameter</u>

   2 bytes, where 0 = space
   1 = period
   2 = termination

After a call of the procedure return (see section 2.3) it is possible to get an empty command line (i.e. only ENTER has been pressed on the supervisor keystation). The contents of comline will then be: <0><255> if no syntax check has been made, or <0> <2> after a syntax check (i.e. termination).

2.2 Code procedure get parameter

This procedure is used to get a single parameter from the command line. The first call of this procedure in the program will return the program name, and the subsequent calls will return the parameters in the order in which they were typed on the supervisor keystation.

The procedure must be declared as follow:

```
    procedure gtpm (var comline: string(112);
                    var item:    string(6);
                    var value:   integer;
                    var kind:    integer;
                    var sep:     integer);
    codebody;
```

Parameters:

Comline:	Call parameter of type "string". See section 2.1.
Item:	Return parameter of type "string".

The length of the parameter must be at least 6 bytes.

The parameter contains a text from the command line if kind = 1.

The text is terminated by at least one null character.

The first character in a text parameter must be a letter followed by letters or digits.

Value: Return parameter of type "integer".

The parameter contains an integer value if kind = 0.

An integer parameter may be typed on the supervisor keystation either in signed/unsigned decimal representation, or in octal representation (identified by a preceeding apostrophe).

Kind: Return parameter of type "integer".

The parameter contains the type of the returned parameter, where:

Kind = 0 means integer parameter.

Kind = 1 means text parameter.

Kind = 2 means that the procedure has been called too many times (i.e. more than the number of parameters in the command line), or that an empty command line has been received (i.e. only ENTER has been pressed on the supervisor keystation) after a call of the procedure return (see section 2.3).

Sep: Return parameter of type "integer".

The parameter contains an integer value which indicates the terminator of the returned parameter, where:

sep = 0 means space

sep = 1 means period

sep = 2 means termination (i.e. ENTER on the supervisor keystation).

2.3 Code procedure return

This procedure is used to return to the supervisor in the Data Entry System, either to get a new command line from the supervisor, or to indicate that the program execution has been terminated and that the process may be removed from the supervisor area in memory. In case of termination the call of this procedure must be the last statement executed in the supervisor program.

The procedure must be declared as follow:

```
procedure retur (var return: integer;
                 var result: integer;
                 var textno: integer~~)~~;  (≡ action)
codebody:        var textmode: integer;
                 var comline: string(80));
```

Parameters:

Return:	Call parameter of type "integer". See section 2.1.
Result:	Call parameter of typr "integer". This parameter may contain information, which may be output on the supervisor keystation display as an octal value (see textno).

Textno: Call parameter of type "integer".
The parameter contains information about a possible text to be output on the supervisor keystation display, special actions to be taken by the supervisor, and about continuation of program execution. The information must be packed as follow:

(text1 shift 10) + (text2 shift 4) + (special shift 1) + continue.

Text 1: Number of the first text to be output. (see section 5).
= 0 means no text.

Text 2: Number of the second text to be output. (see section 5).
= 0 means no text.

Special:= 0 means no special action.
= 1 means output results as octal value after first and second text.

= 4 means no syntax check of the next command line in connexion with continue = 1

= 5 means both actions described by special = 1 and 4.

Continue: = 0 means program execution has been terminated. Remove process from supervisor area in memory.

= 1 means get a new command line from the supervisor after output of the specified text on the supervisor keystation display.

When a new command line is wanted (i.e. continue = 1) the following should be noted:

1) After call of the return procedure, the procedure get command must be called again to get the new command line.

2) If no syntax check of the new command line is wanted (i.e. special = 4 or 5), the contens of comline must be used because this will be the command, exactly as it is typed on the supervisor keystation, followed by a null character.

3) If the new command line has been syntax checked by the supervisor (i.e. special = 0 or 1), the procedure get parameter must be called the number of times necessary to get the parameters.

textmode: 0 = normal retur (= returnering af hvad der er angivet i action)

1 = returnering af string

2 = kald af nyt supervisorprog.

3. Programming hints

3.1 Operator communication

When a supervisor program has been started from the supervisor keystation, the communication with the program may be done either from the supervisor key-station using the code procedures "get parameter" and "get command", or from the RC3600 console device using the standard operator communication procedures described in the MUSIL manual, RCSL: 44-RT 740.

The latter communication is recommended for greater supervisor programs such as line image print programs, paper tape conversion programs, etc., because it gives the possibility for temporary halts of the program execution caused by the operator.

3.2 Use of disc

Because of security reasons for the Data Entry System, writing on a disc must never be done from supervisor programs, as this may imply a risk for destruction of catalog files and other disc files. However it is always permitted to read from a disc file.

3.3 Use of line printer

It should be noted that all line printer drivers, used by the Data Entry System, has the process names "LPT", "LPT1", etc. regardless of which type of printer they are communicating with, and that they all contains a conversion table for conversion from ASCII characters to the actual print drum on each RC3600 system.

This means that no conversion table must be stated in the line printer zone descriptor in a supervisor program, and that all print data must be converted to ASCII characters by the supervisor program <u>before</u> it is passed on to the line printer driver. However, on application to A/S Regnecentralen, it is possible to get a special line printer driver which allows use of a conversion table in the line printer zone descriptor in the supervisor program.

3.4 Program termination

Before returning to the supervisor by means of the code procedure "return" with continue = 0, all processes (i.e. driver-, and area processes), which have been used by the program, must be released. This is done by calling the procedure "close" with non-zero release (i.e. close (zone, 1);).

3.5 Error procedures

After an error which cannot be corrected by the operator, or no correction is wanted by the program, the program execution must be terminated as described in section 3.4, but before that, the mode in the erroneous zone should be set to zero to avoid a program loop when the procedure "close" is called.

This program loop may occur when a zone is closed after an output error (e.g. printer off-line in the sample programs in section 6), because closing the output zone may result in a new output message, which again results in an error return to the zone giveup procedure, after which a new call of the procedure "close" must be done, etc. etc.

Changing the mode to zero before closing the zone will prevent further output messages, and thus only release the driver process, but some output data may be lost.

4. Installation of new supervisor programs

A new supervisor program may be put into the Data Entry System on a disc by means of the standard supervisor program "PUT". In this way it is not necessary to generate a new system tape every time a new supervisor program has been made.

The binary supervisor program (i.e. output from the MUSIL compiler) may be read from either paper tape or magnetic tape.

The commands for "PUT" is as follow:

1. Key: PUT	
2. Key: PTR	for paper tape input,
or: MTC. <fileno>	for magnetic tape input, where <fileno> is the fileno on the magnetic tape from which the new supervisor program must be read. <fileno> must be greater than zero.
3. Key: <name>	(1 to 5 characters), where <name> is the name which must be used later on in calls of the new supervisor program.
4. Press the ENTER key.	

```
Examples: PUT PTR SKRIV
          PUT MTC.1 WRITE
```

In case <name> already exists as a disc file, the contents of this disc file will be replaced by the new supervisor program. So be very careful to choose a <name> which is not used for anything else in order not to destroy data batches etc.. On the other hand it also gives a possibility to replace old versions of a supervisor program without changing the name.

5. Standard texts within the Data Entry System

At the present moment the following standard texts is available in the Data Entry System:

Textno.	Text
1	stop
2	printer
3	break
4	syntax
5	batch
6	state
7	error
8	magtape
9	load err
10	**supv
11	not name
12	ok
13	cf list
14	cf list;
15	disc
16	file
17	ident
18	unknown
19	exist
20	no room
21	name
22	chars
23	copied
24	not
25	record
26	length
27	next

Textno.	Text
28	punch
29	transmit
30	erase
31	date
32	tape ser
33	number
34	file gen
35	informa
36	real seq
37	block
38	factor
39	load

```
!
                                        RCSL:    43-R10209

                                        AUTHOR: JH

                                        EDITED: 76.03.22




                    PROGRAM RC36-90025.02

                    SKRIV




KEYWORDS:          BOSIL, DPD, LPT, DATA ENTRY, LISTING

ABSTRACT:          THIS PROGRAM PRINTS A DISCFILE BY PRINTING THE DECIMAL
                   VALUE OF EACH CHARACTER.
                   THIS PROGRAM IS A DATA ENTRY SUPERVISOR PROGRAM.

RCSL: 43-R10210: ASCII SOURCE TAPE.
RCSL: 43-R10211: REL. BIN. TAPE.
!
```

```
!

TITLE:          SKRIV

ABSTRACT:       THIS PROGRAM PRINTS A DISCFILE BY PRINTING THE DECIMAL
                VALUE OF EACH CHARACTER.
                THIS PROGRAM IS A DATA ENTRY SUPERVISOR PROGRAM.


SIZE:           1840 BYTES, INCLUDING ONE 512-BYTE INPUT BUFFER
                AND ONE 132-BYTE OUTPUT BUFFER.

DATE:           MARCH 22ND 1976.

CALL:           SKRIV <DISC FILE NAME>

OUTPUT MESSAGES:

  SYNTAX        SYNTAX ERROR IN THE CALL LINE.
  OK            END OF FILE. THE PROGRAM EXECUTION IS TERMINATED
                SUCCESFULLY.
  DISC ERROR <CODE>
                CONSULT THE APPENDIX TO THE RC3600 DATA ENTRY USER'S
                MANUAL.
  PRINTER ERROR <CODE>
                CONSULT THE APPENDIX TO THE RC3600 DATA ENTRY USER'S
                MANUAL.


SPECIAL REQUIREMENTS:
                CMMD (R0001: RCSL: 43-RI0111)
                GTPM (R0003: RCSL: 43-RI0117)
                RETUR (R0004: RCSL: 43-RI0120)
!
```

```


CONST

SPACE=              32,
NEWLINE=            '<13><10>';

VAR

OPCSTRING:          STRING(5);
CHAR:               INTEGER;
COUNT:              INTEGER;
COMLINE:            STRING(112);
RETURN:             INTEGER;
ITEM:               STRING(6);
VALUE:              INTEGER;
KIND:               INTEGER;
SEP:                INTEGER;
RESULT:             INTEGER;
TEXTNO:             INTEGER;
TEXT1:              INTEGER;
TEXT2:              INTEGER;
SPECIAL:            INTEGER;
CONTINUE:           INTEGER;

LPT:      FILE                         ! OUTPUT FILE DESCRIPTION          !
          'LPT',                       ! NAME OF OUTPUT DRIVER            !
          1,                           ! KIND = CHARACTER                 !
          1,                           ! NO OF BUFFERS                    !
          132,                         ! BUFFER SIZE                      !
          0U;                          ! FORMAT = UNFORMATTED BLOCKED     !
          GIVEUP LPTERROR,             ! GIVEUP PROCEDURE                 !
          2'1110001111111110           ! GIVEUP MASK                      !
          OF STRING(132);              ! RECORD STRUCTURE                 !

BATCH:    FILE                         ! IN  PUT FILE DESCRIPTION         !
          'DUMMY',                     ! NAME OF DISC FILE, CHANGED       !
                                       ! BY THE PROGRAM AT RUNTIME        !
          2'111100,                    ! KIND = SEQUENTIAL DISC FILE,     !
                                       !        ACCESSED BY AREAPROCESS   !
                                       !        REPEATABLE                !
                                       !        POSITIONABLE              !
          1,                           ! NO OF BUFFERS                    !
          512,                         ! BUFFER SIZE                      !
          0;                           ! FORMAT = UNFORMATTED             !
          GIVEUP BATERROR,             ! GIVEUP PROCEDURE                 !
          2'1111111111111111           ! GIVEUP MASK                      !
          OF STRING(512);              ! RECORD STRUCTURE                 !



```

```
                                                          ! RC36-90025 PAGE 03 !

PROCEDURE CMMD ! GET COMMAND ! (VAR COMLINE:  STRING(112);
                                VAR RETURN:   INTEGER);
CODEBODY;

PROCEDURE GTPM ! GET PARAMETER ! (VAR COMLINE:  STRING(112);
                                  VAR ITEM:     STRING(6);
                                  VAR VALUE:    INTEGER;
                                  VAR KIND:     INTEGER;
                                  VAR SEP:      INTEGER);
CODEBODY;

PROCEDURE RETUR ! RETURN ! (VAR RETURN:  INTEGER;
                            VAR RESULT:  INTEGER;
                            VAR TEXTNO:  INTEGER);
CODEBODY;

PROCEDURE LPTERROR;
BEGIN
     TEXT1:= 2;          ! TEXT1 = PRINTER !
     TEXT2:= 7;          ! TEXT2 = ERROR !
     RESULT:= LPT.Z0;    ! RESULT = LPT STATUSWORD !
     SPECIAL:= 1;        ! OUTPUT RESULT AS OCTAL VALUE !
     CONTINUE:= 0;       ! TERMINATE PROGRAM EXECUTION !
     GOTO 40;
END;

PROCEDURE BATERROR;
BEGIN
     IF BATCH.Z0 AND 2'10000 <> 0 THEN GOTO 30;      ! END MEDIUM !
     TEXT1:= 15;               ! TEXT1 = DISC !
     TEXT2:= 7;                ! TEXT2 = ERROR !
     RESULT:= BATCH.Z0;        ! RESULT = BATCH STATUS WORD !
     SPECIAL:= 1;              ! OUTPUT RESULT AS OCTAL VALUE !
     CONTINUE:= 0;             ! TERMINATE PROGRAM EXECUTION !
     GOTO 40;
END;
```

```
                                           ! RC36-90025 PAGE 04 !

    BEGIN
         [illegible](COMLINE, RETURN);      ! GET COMMAND !
         GIFM(COMLINE, ITEM, VALUE, KIND, SEP);       ! GET PROGRAM NAME !
         GIFM(COMLINE, ITEM, VALUE, KIND, SEP);       ! GET DISC FILE NAME !
         IF KIND <> 1 THEN
         BEGIN  !  PARAMETER NOT NAME !
              TEXT1:= 4;            ! TEXT1 = SYNTAX !
              TEXT2:= 0,            ! NO SECOND TEXT !
              SPECIAL:= 0;          ! NO SPECIAL ACTION !
              CONTINUE:= 0;         ! TERMINATE PROGRAM EXECUTION !
              GOTO 50;
         END;
         BATCH.ZNAME:= ITEM;    ! SET DISCFILE NAME IN INPUT FILE DESCRIPTION !
         OPEN(LPT, 5);
         OPEN(BATCH, 1);
         SETPOSITION(BATCH, 0, 0);
         PUTREC(LPT, 9);
         MOVE(NEWLINE, 0, LPT↑, 0, 2);
         MOVE(ITEM, 0, LPT↑, 2, 5);
         MOVE(NEWLINE, 0, LPT↑, 7, 2);
     10: COUNT:= 0;
     20: INCHAR(BATCH, CHAR);
         BINDEC(CHAR, DECSTRING);
         PUTREC(LPT, 4);
         MOVE(DECSTRING, 2, LPT↑, 0, 3);
         INSERT(SPACE, LPT↑, 3);
         COUNT:= COUNT + 1;
         IF BATCH.ZREM = 0 THEN COUNT:= 30;     ! BLOCK CHANGE !
         IF COUNT < 30 THEN GOTO 20;
         PUTREC(LPT, 2);
         MOVE(NEWLINE, 0, LPT↑, 0, 2);
         GOTO 10;

     30: ! NORMAL RETURN !
         CLOSE(LPT, 1);
         CLOSE(BATCH, 1);
         TEXT1:= 12;           ! TEXT1 = OK !
         TEXT2:= 0;            ! NO SECOND TEXT !
         SPECIAL:= 0;          ! NO SPECIAL ACTION !
         CONTINUE:= 0;         ! TERMINATE PROGRAM EXECUTION !
         GOTO 50;

     40: ! ERROR RETURN !
         IF BATCH.ZMODE <> 0 THEN
         BEGIN
              BATCH.ZMODE:= 0;
              CLOSE(BATCH, 1);     ! RELEASE DISCFILE AREA PROCESS !
         END;
         IF LPT.ZMODE <> 0 THEN
         BEGIN
              LPT.ZMODE:= 0;
              CLOSE(LPT, 1);     ! RELEASE LINE PRINTER DRIVER !
         END;

     50: ! RETURN TO SUPERVISOR !
         TEXTNO:= TEXT1 SHIFT 10 + TEXT2 SHIFT 4 + SPECIAL SHIFT 1 + CONTINUE;
         RETURN(RETURN, RESULT, TEXTNO);     ! RETURN !
    END,
SIZE: 00920
```

```
!                                                       RCSL:    43-RI0212

                                                        AUTHOR: PEN

                                                        EDITED: 76.03.22




                            PROGRAM RC36-90026.02

                            WRITE




KEYWORDS:            MUSIL, MTA, LPT, DATA ENTRY, LISTING

ABSTRACT:            THIS PROGRAM PRINTS A MAGTAPE FILE BY PRINTING THE
                     DECIMAL VALUE OF EACH CHARACTER AND AFTER EACH BLOCK
                     THE BLOCK LENGTH.
                     THIS PROGRAM IS A DATA ENTRY SUPERVISOR PROGRAM.

RCSL: 43-RI0213: ASCII SOURCE TAPE
RCSL: 43-RI0214: REL. BIN. TAPE
!
```

```
!                                                 RC36-90026 PAGE 01

TITLE:                  WRITE.

ABSTRACT:               THIS PROGRAM PRINTS A MAGTAPE FILE BY PRINTING THE
                        DECIMAL VALUE OF EACH CHARACTER AND AFTER EACH BLOCK
                        THE BLOCK LENGTH.
                        THIS PROGRAM IS A DATA ENTRY SUPERVISOR PROGRAM.

SIZE:                   3744 BYTES, INCLUDING ONE 2400-BYTE INPUT BUFFER
                        AND ONE 132-BYTE OUTPUT BUFFER.

DATE:                   MARCH 22ND 1976.

CALL:                   WRITE <FILE NUMBER>
                        WHERE <FILE NUMBER> >= 1.

OUTPUT MESSAGES:
  SYNTAX                SYNTAX ERROR IN THE CALL LINE.
  OK                    END OF FILE. THE PROGRAM EXECUTION IS TERMINATED
                        SUCCESFULLY.
  MAGTAPE ERROR <CODE>
                        CONSULT THE APPENDIX TO THE RC3600 DATA ENTRY USER'S
                        MANUAL.
  PRINTER ERROR <CODE>
                        CONSULT THE APPENDIX TO THE RC3600 DATA ENTRY USER'S
                        MANUAL.

SPECIAL REQUIREMENTS:
                        CMMD (R0001: RCSL: 43-RI0111)
                        GTPM (R0003: RCSL: 43-RI0117)
                        RETUR (R0004: RCSL: 43-RI0120)
!


```

```
                                                ! RC36-90026 PAGE 02 !


CONST

SPACE=          32,
NEWLINE=        '<13><10>';

VAR

DECSTRING:      STRING(5);
CHAR:           INTEGER;
COUNT:          INTEGER;
LENGTH:         INTEGER;
COMLINE:        STRING(112);
RETURN:         INTEGER;
ITEM:           STRING(6);
VALUE:          INTEGER;
KIND:           INTEGER;
SEP:            INTEGER;
RESULT:         INTEGER;
TEXTNO:         INTEGER;
TEXT1:          INTEGER;
TEXT2:          INTEGER;
SPECIAL:        INTEGER;
CONTINUE:       INTEGER;

LPT:    FILE                       ! OUTPUT FILE DESCRIPTION      !
        'LPT',                     ! NAME OF OUTPUT DRIVER        !
        1,                         ! KIND = CHARACTER             !
        1,                         ! NO OF BUFFERS                !
        132,                       ! BUFFER SIZE                  !
        UB;                        ! FORMAT = UNFORMATTED BLOCKED !
        GIVEUP LPTERROR,           ! GIVEUP PROCEDURE             !
        2'1110001111111110         ! GIVEUP MASK                  !
        OF STRING(132);            ! RECORD STRUCTURE             !

MT:     FILE                       ! INPUT FILE DESCRIPTION       !
        'MTO',                     ! NAME OF INPUT DRIVER         !
        2'1110,                    ! KIND = REPEATABLE            !
                                   !        POSITIONABLE          !
                                   !        BLOCKED               !
        1,                         ! NO OF BUFFERS                !
        2400,                      ! BUFFER SIZE                  !
        U;                         ! FORMAT = UNFORMATTED         !
        GIVEUP MTERROR,            ! GIVEUP PROCEDURE             !
        2'1111001110011111         ! GIVEUP MASK                  !
        OF STRING(2400);           ! RECORD STRUCTURE             !


```

```

PROCEDURE CMMD ! GET COMMAND ! (VAR COMLINE:   STRING(112);
                                VAR RETURN:    INTEGER);
CODEBODY;

PROCEDURE GTPM ! GET PARAMETER ! (VAR COMLINE:  STRING(112);
                                  VAR ITEM:     STRING(6);
                                  VAR VALUE:    INTEGER;
                                  VAR KIND:     INTEGER;
                                  VAR SEP:      INTEGER);
CODEBODY;

PROCEDURE RETUR ! RETURN ! (VAR RETURN:  INTEGER;
                            VAR RESULT:  INTEGER;
                            VAR TEXTNO:  INTEGER);
CODEBODY;

PROCEDURE LPTERROR;
BEGIN
    TEXT1:= 2;              ! TEXT1 = PRINTER !
    TEXT2:= 7;              ! TEXT2 = ERROR !
    RESULT:= LPT.ZO;        ! RESULT = LPT STATUS WORD !
    SPECIAL:= 1;            ! OUTPUT RESULT AS OCTAL VALUE !
    CONTINUE:= 0;           ! TERMINATE PROGRAM EXECUTION !
    GOTO 50;
END;

PROCEDURE MTERROR;
BEGIN
    IF MT.ZO AND 8'400 <> 0 THEN GOTO 40;   ! END OF FILE !
    TEXT1:= 8;              ! TEXT1 = MAGTAPE !
    TEXT2:= 7;              ! TEXT2 = ERROR !
    RESULT:= MT.ZO;         ! RESULT = MT STATUS WORD !
    SPECIAL:= 1;            ! OUTPUT RESULT AS OCTAL VALUE !
    CONTINUE:= 0;           ! TERMINATE PROGRAM EXECUTION !
    GOTO 50;
END;

```

```
                                                    ! RC36-90026 PAGE 04 !

     BEGIN
         CMMD(COMLINE, RETURN);       ! GET COMMAND LINE !
         GTPM(COMLINE, ITEM, VALUE, KIND, SEP);     ! GET PROGRAM NAME !
         GTPM(COMLINE, ITEM, VALUE, KIND, SEP);     ! GET MT FILE NO !
         IF KIND <> 0 THEN
         BEGIN  ! PARAMETER NOT NUMMERIC !
             TEXT1:= 4;          ! TEXT1 = SYNTAX !
             TEXT2:= 0;          ! NO SECOND TEXT !
             SPECIAL:= 0;        ! NO SPECIAL ACTION !
             CONTINUE:= 0;       ! TERMINATE PROGRAM EXECUTION !
             GOTO 60;
         END;
         OPEN(LPT, 3);
         OPEN(MT, 5);
         SETPOSITION(MT, VALUE, 1);
     10: LENGTH:= 0;
     20: COUNT:= 0;
     30: INCHAR(MT, CHAR);
         BINDEC(CHAR, DECSTRING);
         PUTREC(LPT, 4);
         MOVE(DECSTRING, 2, LPT↑, 0, 3);
         INSERT(SPACE, LPT↑, 3);
         COUNT:= COUNT + 1;
         LENGTH:= LENGTH + 1;
         IF MT.ZREM = 0 THEN
         BEGIN  ! BLOCK CHANGE, OUTPUT BLOCK LENGTH !
             BINDEC(LENGTH, DECSTRING);
             PUTREC(LPT, 9);
             MOVE(NEWLINE, 0, LPT↑, 0, 2);
             MOVE(DECSTRING, 0, LPT↑, 2, 5);
             MOVE(NEWLINE, 0, LPT↑, 7, 2);
             GOTO 10;
         END;
         IF COUNT < 30 THEN GOTO 30;
         PUTREC(LPT, 2);
         MOVE(NEWLINE, 0, LPT↑, 0, 2);
         GOTO 20;

     40: ! NORMAL RETURN !
         CLOSE(LPT, 1);
         CLOSE(MT, 1);
         TEXT1:= 12;          ! TEXT1 = OK !
         TEXT2:= 0;           ! NO SECOND TEXT !
         SPECIAL:= 0;         ! NO SPECIAL ACTION !
         CONTINUE:= 0;        ! TERMINATE PROGRAM EXECUTION !
         GOTO 60;

     50: ! ERROR RETURN !
         IF LPT.ZMODE <> 0 THEN
         BEGIN
             LPT.ZMODE:= 0;
             CLOSE(LPT, 1);        ! RELEASE LINE PRINTER DRIVER !
         END;
         IF MT.ZMODE <> 0 THEN
         BEGIN
             MT.ZMODE:= 0;
             CLOSE(MT, 1);         ! RELEASE MT DRIVER !
         END;

     60: ! RETURN TO SUPERVISOR !
         TEXTNO:= TEXT1 SHIFT 10 + TEXT2 SHIFT 4 + SPECIAL SHIFT 1 + CONTINUE;
         RETUR(RETURN, RESULT, TEXTNO);
     END;
SIZE: 01872
```