



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

	Side
1. Introduction	1
2. Description of standard code procedures	2
2.1 Code procedure get command	2
2.2 Code procedure get parameter	3
2.3 Code procedure return	6
3. Programming hints	9
3.1 Operator communication	9
3.2 Use of disc	9
3.3 Use of line printer	9
3.4 Program termination	10
3.5 Error procedures	10
4. Installation of new supervisor programs	12
5. Standard texts within the Data Entry System	13
6. Supervisor program examples	15

## 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. Type of parameter

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

2. The parameter

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

3. Terminator for parameter

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 combine: string(80));
```

#### Parameters:

Return: Call parameter of type "integer".  
See section 2.1.

Result: Call parameter of type "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.

Conti- = 0 means program execution has been  
nue: 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 contents 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.

Footnote: 0 = normal return (returning w/ hand dir w/ any value)  
1 = returning w/ string  
2 = hand w/ just supervisor pg

### 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 keystation 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 before 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:

<u>Textno.</u>	<u>Text</u>
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

000  
001  
002  
003  
004  
005  
006  
007  
008  
009  
0010  
0011  
0012  
0013  
0014  
0015  
0016  
0017  
0018  
0019  
0020  
0021  
0022  
0023  
0024  
0025  
0026  
0027  
0028  
0029  
0030  
0031  
0032  
0033  
0034  
0035  
0036  
0037  
0038  
0039  
0040  
0041  
0042  
0043  
0044  
0045  
0046  
0047  
0048  
0049  
0050  
0051  
0052  
0053  
0054  
0055  
0056  
0057  
0058  
0059  
0060  
0061  
0062  
0063

RCSL: 43-R10209  
AUTHOR: JH  
EDITED: 76.03.22

PROGRAM: RC36-90025-02  
SKRTV

KEYWORDS: KUSTL, 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.

0064 !  
0065  
0066 TITLE: SKRIV  
0067  
0068 ABSTRACT: THIS PROGRAM PRINTS A DISCFILE BY PRINTING THE DECIMAL  
0069 VALUE OF EACH CHARACTER.  
0070 THIS PROGRAM IS A DATA ENTRY SUPERVISOR PROGRAM.  
0071  
0072  
0073 SIZE: 4840 BYTES, INCLUDING ONE 512-BYTE INPUT BUFFER  
0074 AND ONE 132-BYTE OUTPUT BUFFER.  
0075  
0076 DATE: MARCH 22ND 1976.  
0077  
0078 CALL: SKRIV <DISC FILE NAME>  
0079  
0080 OUTPUT MESSAGES:  
0081  
0082 SYNTAX SYNTAX ERROR IN THE CALL LINE.  
0083 OK END OF FILE. THE PROGRAM EXECUTION IS TERMINATED  
0084 SUCCESSFULLY.  
0085 DISC ERROR <CODE>  
0086 CONSULT THE APPENDIX TO THE RC3600 DATA ENTRY USER'S  
0087 MANUAL.  
0088 REENTER ERROR <CODE>  
0089 CONSULT THE APPENDIX TO THE RC3600 DATA ENTRY USER'S  
0090 MANUAL.  
0091  
0092  
0093 SPECIAL REQUIREMENTS:  
0094 CMRD (R0001: R0SL: 43-RT0111)  
0095 GTRM (R0003: R0SL: 43-RT0117)  
0096 RETUR (R0004: R0SL: 43-RT0120)  
0097 !  
0098  
0099

0100			
0101			
0102	CONST		
0103			
0104	SPACE#	36,	
0105	REFLTYPE#	'<13><10>' ;	
0106			
0107	MAX#		
0108			
0109	DEVSTR#	STRING(5);	
0110	CHAR#	INTEGER;	
0111	COMINT#	INTEGER;	
0112	COMMTYPE#	STRING(112);	
0113	REFNUM#	INTEGER;	
0114	CFILE#	STRING(6);	
0115	VALDEF#	INTEGER;	
0116	RJLDT#	INTEGER;	
0117	SEED#	INTEGER;	
0118	RESULT#	INTEGER;	
0119	TEXTAD#	INTEGER;	
0120	TEXT#	INTEGER;	
0121	TEXT#	INTEGER;	
0122	SPECIAL#	INTEGER;	
0123	COMINT#	INTEGER;	
0124			
0125	LPT#	FILE	! OUTPUT FILE DESCRIPTION
0126		'LPT1'	! NAME OF OUTPUT DRIVER
0127		1	! KIND = CHARACTER
0128		1	! NO OF BUFFERS
0129		132	! BUFFER SIZE
0130		007	! FORMAT = UNFORMATTED BLOCKED
0131		GIVEUP LPTERROR#	! GIVEUP PROCEDURE
0132		2'1110001111111110	! GIVEUP MASK
0133		OF STRING(132);	! RECORD STRUCTURE
0134			
0135	BATCH#	FILE	! IN - PUT, FTLE DESCRIPTION
0136		'DUMMY'	! NAME OF DISC FILE, CHANGED
0137			! BY THE PROGRAM AT RUNTIME
0138		2'1111002	! KIND = SEQUENTIAL DISC FILE,
0139			! ACCESSED BY AREAPROCESS
0140			! REPEATABLE
0141			! POSITIONABLE
0142		1	! NO OF BUFFERS
0143		512	! BUFFER SIZE
0144		07	! FORMAT = UNFORMATTED
0145		GIVEUP BATERROR#	! GIVEUP PROCEDURE
0146		2'1111111111111111	! GIVEUP MASK
0147		OF STRING(512);	! RECORD STRUCTURE
0148			
0149			
0150			

0151  
0152  
0153 PROCEDURE CMD ! GET COMMAND ! (VAR COMLINE: STRING(112);  
0154 VAR RETURN: INTEGER);  
0155 COMMENT;  
0156  
0157 PROCEDURE GTPM ! GET PARAMETER ! (VAR COMLINE: STRING(112);  
0158 VAR ITEM: STRING(6);  
0159 VAR VALUE: INTEGER;  
0160 VAR KIND: INTEGER;  
0161 VAR SEE: INTEGER);  
0162 COMMENT;  
0163  
0164 PROCEDURE RETUR ! RETURN ! (VAR RETURN: INTEGER;  
0165 VAR RESULT: INTEGER;  
0166 VAR TEXTNO: INTEGER);  
0167 COMMENT;  
0168  
0169 PROCEDURE LPTERROR;  
0170 REDEF;  
0171 TEXT1:= 2; ! TEXT1 = PRINTER !  
0172 TEXT2:= 7; ! TEXT2 = ERROR !  
0173 RESULT:= LPT\_Z0; ! RESULT = LPT STATUSWORD !  
0174 SPECIAL:= 12; ! OUTPUT RESULT AS OCTAL VALUE !  
0175 CONTINUE:= 0; ! TERMINATE PROGRAM EXECUTION !  
0176 GOTO 40;  
0177 END;  
0178  
0179 PROCEDURE BATERROR;  
0180 REDEF;  
0181 IF BATCH\_Z0 AND 2^10000 <> 0 THEN GOTO 30; ! END MEDIUM !  
0182 TEXT1:= 15; ! TEXT1 = DISC !  
0183 TEXT2:= 7; ! TEXT2 = ERROR !  
0184 RESULT:= BATCH\_Z0; ! RESULT = BATCH STATUS WORD !  
0185 SPECIAL:= 13; ! OUTPUT RESULT AS OCTAL VALUE !  
0186 CONTINUE:= 0; ! TERMINATE PROGRAM EXECUTION !  
0187 GOTO 40;  
0188 END;  
0189  
0190  
01

```

0192
0193
0194 68650:
0195  GET(COMLINES, RETURN);      ! GET COMMAND !
0196  GET(COMLINES, ITEM, VALUE, KIND, SEP);   ! GET PROGRAM NAME !
0197  GET(COMLINES, ITEM, VALUE, KIND, SEP);   ! GET DISC FILE NAME !
0198  IF KIND <> 1 THEN
0199    BEGIN ! PARAMETER NOT NAME !
0200      TEXT1:= 4;           ! TEXT1 = SYNTAX !
0201      TEXT2:= 0;           ! NO SECOND TEXT !
0202      SPECIAL:= 0;         ! NO SPECIAL ACTION !
0203      CONTINUE:= 0;        ! TERMINATE PROGRAM EXECUTION !
0204      GOTO 50;
0205  ENDIF
0206  SETCH,ZNAME:= ITEM;   ! SET DISCFILE NAME IN INPUT FILE DESCRIPTION !
0207  OPENCLPT, 502
0208  OPENBATCH, 102
0209  SETPOSITION(BATCH, 0, 0);
0210  PUTREC(CLPT, 90);
0211  MOVE(CNLINES, 0, LPT1, 0, 20);
0212  MOVE(ITEM, 0, LPT1, 2, 50);
0213  MOVE(CNLINES, 0, LPT1, 7, 20);
0214  COUNT:= 0;
0215  DO:  INCHAR(BATCH, CHAR);
0216  READREC(CHAR, DECSTRING);
0217  PUTREC(CLPT, 40);
0218  MOVE(DECSTRING, 2, LPT1, 0, 50);
0219  USEP(SPACE, LPT1, 30);
0220  COUNT:= COUNT + 1;
0221  IF BATCH,ZREM = 0 THEN COUNT:= 30;   ! BLOCK CHANGE !
0222  IF COUNT < 50 THEN GOTO 20;
0223  PUTREC(CLPT, 20);
0224  MOVE(CNLINES, 0, LPT1, 0, 20);
0225  GOTO 10;
0226
0227  BUT ! NORMAL RETURN !
0228  CLOSECLPT, 102
0229  CLOSEBATCH, 102
0230  TEXT1:= 127;          ! TEXT1 = OK !
0231  TEXT2:= 0;             ! NO SECOND TEXT !
0232  SPECIAL:= 0;           ! NO SPECIAL ACTION !
0233  CONTINUE:= 0;          ! TERMINATE PROGRAM EXECUTION !
0234  GOTO 50;
0235
0236  40: ! ERROR RETURN !
0237  IF BATCH,ZMODE <> 0 THEN
0238    BEGIN
0239      BATCH,ZMODE:= 0;
0240      CLOSEBATCH, 10;     ! RELEASE DISCFILE AREA PROCESS !
0241    END;
0242  IF LPT,ZMODE <> 0 THEN
0243    BEGIN
0244      LPT,ZMODE:= 0;
0245      CLOSECLPT, 10;     ! RELEASE LINE PRINTER DRIVER !
0246    END;
0247
0248  50: ! RETURN TO SUPERVISOR !
0249  TEXTNO:= TEXT1 SHIFT 10 + TEXT2 SHIFT 4 + SPECIAL SHIFT 1 + CONTINUE;
0250  RETUR(GRETURN, RESULT, TEXTNO);   ! RETURN !
0251  ERN,
0252  SIZE:= 0;

```

RCSL: 43-R10212

AUTHOR: PEN

EDITED: 76.03.22

## PROGRAM RC36-90026.02

WRITE

0055 KEYWORDS: MUSTER, MTA, LPT, DATA ENTRY, LISTING

0055 ABSTRACT: THIS PROGRAM PRINTS A MAGTAPE FILE BY PRINTING THE DECIMAL VALUE OF EACH CHARACTER AND AFTER EACH BLOCK THE BLOCK LENGTH.

0055 THIS PROGRAM IS A DATA ENTRY SUPERVISOR PROGRAM.

0060 RCSL: 43-R10213: ASCII SOURCE TAPE

0061 RCSL: 43-R10214: REL. BIN. TAPE

0062 !

0063 !

RC36-90026 PAGE 01

0064 !

0065

0066 TITLE:

WRITE.

0067

0068 ABSTRACT:

THIS PROGRAM PRINTS A MAGTAPE FILE BY PRINTING THE DECIMAL VALUE OF EACH CHARACTER AND AFTER EACH BLOCK THE BLOCK LENGTH.

0069

THIS PROGRAM IS A DATA ENTRY SUPERVISOR PROGRAM.

0070

0071 SIZE:

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

0072

0073 DATE:

MARCH 22ND 1976.

0074

0075 CALL:

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

0076

0077 OUTPUT MESSAGES:

0078

0079 SYNTAX

SYNTAX ERROR IN THE CALL LINE.

0080

OK END OF FILE. THE PROGRAM EXECUTION IS TERMINATED  
SUCCESFULLY.

0081

0082 MAGTAPE ERROR &lt;CODE&gt;

CONSULT THE APPENDIX TO THE RC3600 DATA ENTRY USER'S  
MANUAL.

0083

0084 PRINTER ERROR &lt;CODE&gt;

CONSULT THE APPENDIX TO THE RC3600 DATA ENTRY USER'S  
MANUAL.

0085

0086 SPECIAL REQUIREMENTS:

0087

CMMD (R0001: RCSL: 43-R10111)

0088

GTPM (R0003: RCSL: 43-PI0117)

0089

RETUR (R0004: RCSL: 43-R10120)

0090

0091

0092

0093

0094

0095

0096 !

0097

0098

```

0099
0100
0101 CONST
0102
0103 SPACE=      32,
0104 NEWLINE=    '<13><10>' ;
0105
0106 VAR
0107
0108 DECString:   STRING(5);
0109 CHAR:        INTEGER;
0110 COUNT:       INTEGER;
0111 LENGTH:      INTEGER;
0112 COMLINE:     STRING(112);
0113 RETURN:      INTEGER;
0114 ITEM:        STRING(6);
0115 VALUE:       INTEGER;
0116 KIND:        INTEGER;
0117 SEP:         INTEGER;
0118 RESULT:      INTEGER;
0119 TEXTNO:      INTEGER;
0120 TEXT1:        INTEGER;
0121 TEXT2:        INTEGER;
0122 SPECIAL:     INTEGER;
0123 CONTINUE:    INTEGER;
0124
0125 LPT:         FILE           ! OUTPUT FILE DESCRIPTION
0126   'LPT',          ! NAME OF OUTPUT DRIVER
0127   1,               ! KIND = CHARACTER
0128   1,               ! NO OF BUFFERS
0129   132,             ! BUFFER SIZE
0130   08;              ! FORMAT = UNFORMATTED BLOCKED
0131   GIVEUP LPTERROR, ! GIVEUP PROCEDURE
0132   2'111000111111110 ! GIVEUP MASK
0133   OF STRING(132);   ! RECORD STRUCTURE
0134
0135 MT:          FILE           ! INPUT FILE DESCRIPTION
0136   'MT0',          ! NAME OF INPUT DRIVER
0137   2'1110,          ! KIND = REPEATABLE
0138
0139
0140   1,               ! POSITIONABLE
0141   2400,            ! BLOCKED
0142   0;               ! NO OF BUFFERS
0143   GIVEUP MTERROR, ! BUFFER SIZE
0144   2'1111001110011111 ! FORMAT = UNFORMATTED
0145   OF STRING(2400); ! GIVEUP PROCEDURE
0146
0147   ! GIVEUP MASK
0148   ! RECORD STRUCTURE

```

```
0148
0149
0150 PROCEDURE CMMD ! GET COMMAND ! (VAR COMLINE: STRING(112);
0151                                     VAR RETURN: INTEGER);
0152 CODEBODY;
0153
0154 PROCEDURE GTPM ! GET PARAMETER ! (VAR COMLINE: STRING(112);
0155                                     VAR ITEM: STRING(6);
0156                                     VAR VALUE: INTEGER;
0157                                     VAR KIND: INTEGER;
0158                                     VAR SEP: INTEGER);
0159 CODEBODY;
0160
0161 PROCEDURE RETUR ! RETURN ! (VAR RETURN: INTEGER;
0162                                     VAR RESULT: INTEGER;
0163                                     VAR TEXTNO: INTEGER);
0164 CODEBODY;
0165
0166 PROCEDURE LPTERROR;
0167 BEGIN
0168     TEXT1:= 2;           ! TEXT1 = PRINTER !
0169     TEXT2:= 7;           ! TEXT2 = ERROR !
0170     RESULT:= LPT.Z0;    ! RESULT = LPT STATUS WORD !
0171     SPECIAL:= 1;        ! OUTPUT RESULT AS OCTAL VALUE !
0172     CONTINUE:= 0;        ! TERMINATE PROGRAM EXECUTION !
0173     GOTO 50;
0174 END;
0175
0176 PROCEDURE MTERROR;
0177 BEGIN
0178     IF MT.Z0 AND 8'400 <> 0 THEN GOTO 40; ! END OF FILE !
0179     TEXT1:= 8;           ! TEXT1 = MAGTAPE !
0180     TEXT2:= 7;           ! TEXT2 = ERROR !
0181     RESULT:= MT.Z0;    ! RESULT = MT STATUS WORD !
0182     SPECIAL:= 1;        ! OUTPUT RESULT AS OCTAL VALUE !
0183     CONTINUE:= 0;        ! TERMINATE PROGRAM EXECUTION !
0184     GOTO 50;
0185 END;
0186
0187
```

```

0188
0189
0190 BEGIN
0191   CMD(COMLINE, RETURN);      ! GET COMMAND LINE !
0192   GTPM(COMLINE, ITEM, VALUE, KIND, SEP);    ! GET PROGRAM NAME !
0193   GTPM(COMLINE, ITEM, VALUE, KIND, SEP);    ! GET MT FILE NO !
0194   IF KIND <> 0 THEN
0195     BEGIN ! PARAMETER NOT NUMERIC !
0196       TEXT1:= 4;          ! TEXT1 = SYNTAX !
0197       TEXT2:= 0;          ! NO SECOND TEXT !
0198       SPECIAL:= 0;        ! NO SPECIAL ACTION !
0199       CONTINUE:= 0;        ! TERMINATE PROGRAM EXECUTION !
0200       GOTO 60;
0201   END;
0202   OPEN(LPT, 3);
0203   OPEN(MT, 5);
0204   SETPOSITION(MT, VALUE, 1);
0205   10: LENGTH:= 0;
0206   20: COUNT:= 0;
0207   30: INCHAR(MT, CHAR);
0208   BINDEC(CHAR, DECSTRING);
0209   PUTREC(LPT, 4);
0210   MOVE(DECSTRING, 2, LPT↑, 0, 30);
0211   INSERT(SPACE, LPT↑, 3);
0212   COUNT:= COUNT + 1;
0213   LENGTH:= LENGTH + 1;
0214   IF MT.ZREM = 0 THEN
0215     BEGIN ! BLOCK CHANGE, OUTPUT BLOCK LENGTH !
0216       BINDEC(LENGTH, DECSTRING);
0217       PUTREC(LPT, 9);
0218       MOVE(NEWLINE, 0, LPT↑, 0, 2);
0219       MOVE(DECSTRING, 0, LPT↑, 2, 5);
0220       MOVE(NEWLINE, 0, LPT↑, 7, 2);
0221       GOTO 10;
0222   END;
0223   IF COUNT < 30 THEN GOTO 30;
0224   PUTREC(LPT, 2);
0225   MOVE(NEWLINE, 0, LPT↑, 0, 2);
0226   GOTO 20;
0227
0228 40: ! NORMAL RETURN !
0229   CLOSE(LPT, 1);
0230   CLOSE(MT, 1);
0231   TEXT1:= 12;          ! TEXT1 = OK !
0232   TEXT2:= 0;          ! NO SECOND TEXT !
0233   SPECIAL:= 0;        ! NO SPECIAL ACTION !
0234   CONTINUE:= 0;        ! TERMINATE PROGRAM EXECUTION !
0235   GOTO 60;
0236
0237 50: ! ERROR RETURN !
0238   IF LPT.ZMODE <> 0 THEN
0239     BEGIN
0240       LPT.ZMODE:= 0;
0241       CLOSE(LPT, 1);      ! RELEASE LINE PRINTER DRIVER !
0242     END;
0243   IF MT.ZMODE <> 0 THEN
0244     BEGIN
0245       MT.ZMODE:= 0;
0246       CLOSE(MT, 1);      ! RELEASE MT DRIVER !
0247     END;
0248
0249 60: ! RETURN TO SUPERVISOR !
0250   TEXTNO:= TEXT1 SHIFT 10 + TEXT2 SHIFT 4 + SPECIAL SHIFT 1 + CONTINUE;
0251   RETUR(RETURN, RESULT, TEXTNO);
0252 END;
SIZE: 01872

```

