A/S REGNECENTRALEN

SCANDINAVIAN INFORMATION PROCESSING SYSTEMS

RCSL no: 53-D2
Edition: January 1970
Author: Ellen Randløv

Title: Magnetic Tape System

Keywords: RC 4000, Basic Software, MTS, File Administration, Magnetic Tape, User's Manual
Abstract: MTS is a system of procedures to administer labelled files on magnetic tapes, structured according to ISO Standard. The MTS is an extension to the input/output administration standard procedures: open close and setposition in algol 5, for application of magnetic tapes. 27 pages.

SYSTEM LIBRARY

DK-2500 VALBY · BJERREGAARDSVEJ 5 · TELEPHONE: (01) 46 08 88 · TELEX: 64 64 rcinf dk · CABLES: INFOCENTRALEN

Contents

1. Introduction

MTS is a system of procedures to administer labelled files on magnetic tapes, structured according to

ISO Standard (ISO/TC 97, 181 E) 'Magnetic Tape Labelling and File Structure for Information Interchange', June 1967.

This proposed standard is completely identical with

'Proposed USA Standard for Magnetic Tape Labels for Information Interchange', COMM. ACM vol. 10, No. 11, Nov. 1967

and

ECMA Standard: 'Standard for Magnetic Tape Labelling'. Nov. 1967.

The MTS is an extension to the input/output administration standard procedures: open, close and setposition in algol 5, for application of magnetic tapes.

The user is supposed to be familiar with the 'Algol 5 Users Manual' by Søren Lauesen.

Files

MTS administers the following file types:

Single - Volume Files
Multi - Volume Files
Multi - File Volume

but not Multi - File Multi - Volumes.
The structure of the files is described in appendix 1.

Labels

A main function of the MTS is the label writing and checking:

Before any operation is executed on a volume the volume label is checked.

When writing a file MTS writes a header label before the file, forms the checksum (app. 1) if desired, and counts the number of blocks during the writing and writes an end-of-file label after the file. If a file is spread over more than one volume, an end-of-volume label is written after every completed volume, and on the next volume the writing of a header label prefaces continuation.

When reading a file MTS counts the blocks and forms, if desired, the checksum and checks the required labels, i.e. for every file, the header label and the end-of-file label, and for every change of volume the end-of-volume label and the header label.

The structure and contents of the labels are described in appendix 1.

Usual Operations

The operations in the user program are described below. Briefly the usual operations are:
Declare a zone and a blockproc (stderror may be used),
describe relevant volumes and files by an array,
open the zone by openadp,
position the tapes by positionadp,
operate with I/O operations,
add new tapes by tapeadp,
close the zone by closeadp.

Tapes

Tapes used by MTS must be named, i.e. the first block on the tape must be a VOL label (app. 1). Ex. 7.6 (page 16) gives an example of tape naming.

2. Survey of I/O Operations and General Rules for Use of MTS.

Input/Output Operations:

The I/O activities of a user of MTS are classified in 3 sorts of operations:

1) input as input operations you may use the following algol 5 standard procedures:
   - a) for record handling: inrec.
   - b) for character handling: read, readchar, readstring, readall.

2) output as output operations you may use the following algol 5 standard procedures:
   - a) for record handling: outrec.
   - b) for character handling: write.

3) output continued being rather special this form of output is described further in a later appendix.

File Structuring Information to the MTS.

The user must describe the relevant volumes and files to the MTS. The description is stored in an real array, here called ident, as follows. This information is fetched by openadp (see below).

Declaration of Ident:

```
(real) array ident (1:<max. number of reels> + 15);
```

Contents of Ident:

At the time of calling openadp the contents of ident must be as follows:

ident(1) = mode of datafiles and labels, where mode is 0 for odd parity and 2 for even parity.

ident(2) = max number of reels

ident(3) = actual number of reels

ident(4) = mask for call of the users block procedure. ident(4) corresponds to the give up mask, used by the algol 5 standard procedure open. In error cases, which are not taken charge of by the user (by ident (4) and blockproc) an alarm is given and execution is terminated (app. 2).

```
            ident(4) = sum of numbers below according to desired call of
            blockproc:
            1          for: tapemark after datafile is read.
            1 shift 1  for: parity error was detected during block trans-
                            fer.
            1 shift 2 for: number of actuel reels of a file is too small.
            1 shift 12 for: checksum error in datafile or volume.
            1 shift 13 for: user input from labels.
            1 shift 14 for: user output to labels.
            (see blockproc p.5 and examples p.12)
   ident(5) - ident(15):       irrelevant.
   ident(16) = name of first reel (≤ 6 characters)
   ident(17) =  -     - second reel
      .
      .
      .
   ident(15 + actual number of reels) = name of last reel
   possible remainder of ident:  irrelevant.
```

<u>Ex 2.1</u>: For declaration and evaluation of ident, the following values are read from current input (without any testing): mode, max number of reels, actual number of reels, mask, names of volumes.

```
begin integer i; real r, s;
  read(in, r, s);
  begin array ident(1:15+s); name(1:2);
    ident(1):= r; ident(2):= s;
    read(in, ident(3), ident(4));
    for i:= 1 step 1 until s do
    begin readstring(in, name, 1);
          ident(15+i):= name(1);
    end i;
      .
      .
      .
(here follows call of openadp, inrec, etc.)
```

Declaration of the Zone

The zone must be declared as usual in Algol 5, but with a change in interpretation of the parameter buf:

```
zone z(buf, sh, blockproc);
     buf    -   (integer): length of user buffer area +
                           length of the identification array ident, in
                           double words.
     sh     -   (integer): number of shares
     blockproc - (procedure with 3 parameters).
```

Length of Buffers

The procedure openadp distributes the entire buffer area to

1) MTS administration
2) sh shares, each with length in double words:
   (buf - length of ident) // sh.

Number of Shares

The usual number of shares is 1, 2 or 3.
Choice of number of shares is discussed in the 'Algol 5 Users Manual' (6.3.1).

Blockproc

The block procedure of the zone must be declared as usual with 3 parameters or it may be the algol 5 standard procedure stderror.

Procedure blockproc will be called during input/output in situations specified by ident(4) (page 3), thus allowing desired user actions.

```
procedure blockproc(z, s, b);
     z - (call and return value, zone): the input/output zone
     b - (call and return value, integer): number of bytes transferred,
         i.e. the latest block transferred is available in z(1:b/4)
     s - (call and return value, integer): specifies the actual situation
         in the following way:
         if s            extract 1 = 1: tapemark after datafile is read,
         if s shift (-1) extract 1 = 1: parity error in datablock,
if s shift (-2) extract 1 = 1: actual number of reels of a
                file is too small;
                you can add a name to the list of names of reels
                by tapeadp (see below) and continue,
if s shift (-12) extract 1 = 1: checksum error in datafile or vo-
                lume;
if s shift (-13) extract 1 = 1: a required label is just read,
if s shift (-14) extract 1 = 1: a required label is to be written,

in the 3 last cases the actual label is available in z(1:b/4).
```

Ex 2.2   A block procedure accepting n parity errors and ignoring connected checksum errors.

At the time of calling openadp, ident(4) must be initiated ident(4):= 1 shift 1 + 1 shift 12:

```
procedure blproc(z, s, b);
zone z; integer s, b;
begin
  own integer i; comment counts parity errors;
  own boolean ch; comment ch = true for checksum error tolerated;
  comment i and ch are initiated 0 and false;
  if s shift (-1) extract 1 = 1 then
  begin i:= i + 1; if i > n then goto error 1;
        ch:= true;
  end;
  if s shift (-12) extract 1 = 1 then
  begin if ch then ch:= false
        else goto error 2;
  end;
end
```

3. Openadp

Call: openadp(z, ident, checksum);

z - (call and return value, zone): I/O zone

ident - (call value, real array): contains relevant information for file structuring.

checksum - (call value, boolean): true if checksum calculation and checking is required, checksum is calculated file section wise (app. 1).

Restrictions:

The zone must be in one of the following states: after declaration, after close or after closeadp,

length of zone buffer > length of ident,
ident(1) must be 0 or 2,
ident(2) >= ident(3) >= 1,
ident must be declared from 1 to 15 + value of ident(2).

Function: Openadp fetches relevant information from ident and reserves the first part of the zone buffer for MTS administration and splits the remaining part of the buffer into the wanted number of shares. For each name in ident of a reel which is not mounted openadp writes on the console: mount <name of reel>, advising the operator without waiting for mounting. The first reel is connected to the zone.
The state of the zone is set to after openadp = 8.

Use: Openadp is used to initiate the MTS administration of the I/O operations.

Examples: see 7.1, 7.3 and 7.4.

4. Closeadp

Call: Closeadp(z, action);

z - (call and return value, zone): I/O zone

action - (call value, integer): action for actual volume connected to z,

action = 0 for stand by
       = 1 for stand by, release
       = 2 for rewind, release
       = 3 for unload, release.

Restrictions: The zone must be in a state different from: after declaration, after close and after closeadp.

0 <= action <= 3

Function: latest operation is completed, if latest operation was output an EOF label (app. 1) is written on the tape.

The action specified by 2nd parameter is executed.

The state of the zone is set to after closeadp = 4.

Use: Closeadp is used for completing the MTS administration.

Examples: see 7.1, 7.2, 7.3 and 7.4.

5. Positionadp

Call: positionadp(z, fileno, op);

z - (call and return value, zone): I/O zone.

fileno - (call value, integer):

specifies the number of a datafile, datafiles are numbered: 1, 2, 3, ...

op - (call value, integer):

specifies the operation:

op = 1 for input

2 for output

3 for output continued.

Restrictions: The zone must be in one of the following states:

after open and positioned, after input, after output or after open not positioned,

if multi-volume file then fileno must be 1 else fileno > 0,

1 <= op <= 3,

if op = 3 zone state must be after open not positioned (op = 3 will be described in a later appendix).

Function and Effect: Positionadp completes latest operation, and if latest operation was output an EOF label is written on the tape.
The first positionadp after openadp checks the VOL label of the first volume, if the first volume is not mounted the console message:

mount <name of reel>

is written, and mounting is waited for.

Positionadp starts the positioning to the specified datafile and provides for completion of the positioning and for the label checking and writing. The state of the zone is set to after positionadp = 0.

Use: Positionadp must be used before any I/O operation on a datafile and before change to another class or level of I/O operation (see page 3)

Examples: see 7.1, 7.2, 7.3 and 7.4.

8. Tapeadp

Call: tapeadp(z, name);

z - (call and return value, zone): the I/O zone.

name - (call value, real):

contains the name of a reel - 6 characters.

Restrictions: The zone must be in a state different from:

after declaration, after close and after closeadp.

actual number of reels < max number of reels.

Function: Appends name to the list of names of volumes connectable to z, and increases number of actual reels with one.

Note that the list of the names of the reels is initiated by openadp from ident. At that state the actual number of reels = ident(3), and it is not changed by tapeadp.
The user may update ident himself if he wants to keep account of volumes connected to the zone.

Ex of updating of ident in connection with a call of tapeadp.

```
begin
array ident(     );
zone z(     );
     .
     .
     .
openadp(z, ident, checksum);
     .
     .
     .
```

```
tapeadp(z, name);
ident(3) := ident(3) + 1;
ident(15 + ident(3)):= name;
     .
     .
     .
```

Tapeadp makes it possible to deal with a pool of volumes common for several multivolume files, assigning them only when necessary.

If ident(4) shift (-2) extract 1 = 1, MTS will call blockproc (page 5) when the reels already assigned (at least one) to the file are used up; when called the block procedure may assign a new reel to that file by means of tapeadp - and the execution can continue.

<u>Examples</u> see 7.5.

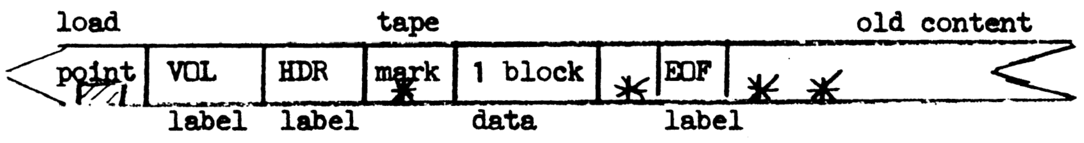
7. Examples

Ex 7.1

One file consisting of one block is written, as the first file on tape mt1234:

```
begin
  array ident(1:16);
  zone z(116, 1, stderror);
  ident(1):= 0; ident(2):= ident(3):= 1; ident(4):= 0;
  ident(16):= real<:mt123:> add 52;
  openadp(z, ident, true);
  positionadp(z, 1, 2);
  outrec(z, 70);
  z(1):= ... ; comment filling;
  closeadp(z, 2);
end
```

Picture of the tape:



Ex 7.2

An empty file is made by outrec(z, 0):

```
    .
    .
    .
positionadp(z,  , 2);
outrec(z, 0);
closeadp(z,  ) or positionadp(z,  ,  );
```

Ex 7.3

File number 1, 2, ..., 5, with max. blocklength = 200 double words, on tape mt10 are copied to one file on tape mt20:

```
begin
  boolean eof;
  integer fileno, p, i;
  array ident(1:16);
  zone zi, zo(216, 2, endproc);
  procedure endproc(z, s, b); zone z; integer s, b;
  eof:= true;

  ident(1):= 0; ident(2):= ident(3):= ident(4):= 1;
  ident(16):= real<:mt10:>; openadp(zi, ident, true);
  ident(16):= real<:mt20:>; openadp(zo, ident, true);
  positionadp(zo, 1, 2);
  for fileno:= 1 step 1 until 5 do
  begin
    positionadp(zi, fileno, 1); eof:= false;
    i:= inrec(zi, 0);
Next_block:
    inrec(zi, i); outrec(zo, i);
    for p:= 1 step 1 until i do zo(p):= zi(p);
    i:= inrec(zi, 0);
    if -, eof then goto Next_block;
  end;
  closeadp(zi, 0); closeadp(zo, 0);
end
```

Ex 7.4

mt104 is a one file register tape, with records as follows, sorted after account number:

1th	double w.	= length of record (between 4 and 20)	(real)
2nd	-	= account number	-
3rd	-	= amount	-
4th-20th		= name and address	

mt5612 is a one file transaction tape, with records as follows, sorted after account number:

1th double w.	=	length of record (between 3 and 20) (real)
2nd -	=	account number -
3rd -	=	transaction amount -
4th-20th -	=	name and address for new accounts just opened.

The following program updates the register on tape mt103. In the HDR label the date is written as char. number 7-12:

```
begin
  integer i, p;
  zone array zi(2, 416, 2, iproc); zone zo(416, 2, oproc);
  array ident(1:16);

  procedure iproc(z, s, b); zone z; integer s, b;
  comment iproc is called when tapemark is read after datafile;
  begin b:= 8; z(2):= greatest_no; end;

  procedure oproc(z, s, b); zone z; integer s, b;
  comment oproc is called just before a label is written, z(1:14) contents
  the label;
  if z(1) shift (-16) shift 16 = real<:HDR1:> then z(2):= date;

  real procedure date;
  comment fetches date by systime and brings it on character form; ... ;

  ident(1):= 0; ident(2):= ident(3):= 1;
  ident(4):= 1; ident(16):= real <:mt104:>; openadp(zi(1), ident, true);
  ident(16):= real <:mt561:> add 50; openadp(zi(2), ident, true);
  ident(4):= 1 shift 14; ident(16):= real<:mt103:>; openadp(zo, ident, true);
  positionadp(zi(1), 1, 1); positionadp(zi(2), 1, 1);
  positionadp(zo, 1, 2);
  inrec(zi(1), 2); inrec(zi(2), 2);
```

```
LOOP 1: if zi(1, 2) > zi(2,2) then p:= 2
        else if zi(1, 2) = greatest_no then goto FINISH
        else p:= 1;
        outrec(zo, zi(p, 1)); zo(1):= zi(p, 1); zo(2):= zi(p, 2);
        inrec(zi(p), zi(p, 1) -2);
        for i:= 3 step 1 until zo(1) do zo(i):= zi(1, i-2);
        inrec(zi(p), 2);
LOOP 2: if zo(2) = zi(2, 2) then
        begin inrec(zi(2), zi(2, 1) -2);
          zo(3):= zo(3) + zi(2, 1);
          inrec(zi(2), 2);
          goto LOOP 2;
        end;
        goto LOOP 1;
FINISH: closeadp(zi(1), 2); closeadp(zi(2), 2);
        closeadp(zo, 2);
end
```

Ex 7.5. Writing on max. 4 reels, adding the names of the reels by tapeadp:

```
begin
  boolean finis; integer p;
  array ident(1:19);
  zone z(219, 2, blproc);
  procedure blproc(z, s, b); zone z; integer s, b;
  comment blproc is called when the actual number of reels is to small;
  begin tapeadp(z, case ident(3) of
              (<:mt2:>, <:mt3:>, <:mt4:>));
       ident(3):= ident(3) + 1;
  end blproc;
```

```
    ident(1):= ident(3):= 1;
    ident(2):= ident(4):= 4;
    ident(16):= real<:mt1:>;
    openadp(z, ident, true);
    finis:= false;
    positionadp(z, 1, 2);
    for p:= 1, p+1 while -,finis do
    begin outrec(z, 10); ... ; comment filling;
    end;
    closeadp;
  end
```

Ex. 7.6   Tape naming.
The name of the reel shall be written on current input, note that name must be a 1 to 6 character textstring of small letters and digits beginning with a letter.

```
begin
  integer val, p, i, j;
  real array name(1:2);
  zone z(84, 1, stderror);

  name(1):= name(2):= real<::>;
  write(out, <:<10> write name of reel: :>; setposition(out, 0, 0);
  readchar(in, val);
  for p:= 1, p+1 while val <> 10 do
  begin name(1):= name(1) shift 8 add val;
        readchar(in, val);
  end;
  i:= p - 1;
  for p:= p step 1 until 6 do name(1):= name(1) shift 8;
  j:= 1; open(z, 18, string name(increase(j)), 0);
  setposition(z, 0, 0); j:= 1;
  write(z, <:VOL1:>, string name(increase(j)), false, 6-i,
        false add 32, 70, false add 94, 4);
  close(z, false);
end
```

Appendix 1.

Magnetic Tape Labels

Structuring of files and contents of labels according to 'Proposed USA Standard for Magnetic Tape Labels for Information Interchange', COMM. ACM, Nov. 1967:

The following 4 cases of file structures are to be handled,
-> denotes start of tape,
× means tape mark,

The various labels are specified in detail below the list of cases.

1. Single - Volume File

```
-> VOL
   HDR ×
   data blocks
   × EOF ××
```

2. Multi - Volume File

```
-> VOL
   HDR  ×
   data blocks of first volume
   end of tape mark
   × EOV  ××
-> .
   .
   .
-> VOL
   HDR  ×
   data blocks of nth and last volume
   × EOF  ××
```

3. Multi - File Volume

```
-> VOL
   HDR  x
   data blocks of first file
   x EOF x
   .
   .
   .
   HDR x
   data blocks of mth and last file
   x EOF xx .
```

4. Volume without Files

```
-> VOL x
```

Tapes used by the MTS must be named, i.e. must start with a VOL label.
Ex of tape naming, see 7.6.

FORMAT AND CONTENTS OF LABELS:

Each label written by MTS is a 84 character block,
label block lengths tolerated by MTS are 80-84 characters.

VOL:

A VOL (volume header) label is required at the beginning of every volume.

field No.	contents	No. of characters	contents checked by MTS
1	VOL	3	yes
2	1	1	yes
3	name of volume	6	yes
4, 5, 6, 7, 8	spaces	69	no
9	1 or space	1	no
10	circumflexes	4	no

A VOL label may be directly followed by one or more label blocks, the first field of which must be UVL (user volume label). UVL's are permitted, but not checked by MTS.

HDR:

A HDR (header) label is required at the beginning of every volume and at the beginning of every file:

field No.	contents	No. of characters	contents checked by MTS
1	HDR	3	yes
2	1	1	yes
3, 4	spaces	23	no
5	file section No.	4	yes
6	file No.	4	yes
7, 8, 9, 10, 11	spaces	19	no
12	000000	6	no
13, first part	000	3	no
13, last part and 14	spaces	17	no
15	circumflexes	4	no

Examples of file section numbers and file numbers may be as follows:

Case	File section No.	File No.
1	0001	0001
2, volume No. 1	0001	0001
2, volume No. n	n	0001
3, file No. 1	0001	0001
3, file No. m	0001	m

Between this HDR label and the following × one or more label blocks, the first field of which must be HDR or UHL (user header label), are permitted, but not checked by MTS.

EOF:

An EOF (end-of-file) label is required at the end of every file.

The format and contents must agree with the preceeding required HDR label.

field No.	contents	No. of characters	contents checked by MTS
1	EOF	3	yes
2	1	1	yes
3-11	as preceeding required HDR label		
12	No. of data blocks after preceeding required HDR	6	yes
13, first part	checksum of data blocks after preceeding required HDR	3	yes
13, last part and 14	spaces	17	no
15	circumflexes	4	no

Between this EOF label and the following × one or more label blocks, the first field of which must be EOF or UTL (user trailer label) are permitted, but not checked by MTS.

EOV:

An EOV (end-of-volume) label is required at the end of every volume. The format and contents must agree with the preceeding required HDR label.

field No.	contents	No. of characters	contents checked by MTS
1	EOV	3	yes
2	1	1	yes
3-11	as preceeding required HDR label		
12	No. of data blocks after preceeding required HDR	6	yes
13, first part	checksum of data blocks after preceeding required HDR	3	yes
13, last part and 14	spaces	17	no
15	circumflexes	4	no

Between this EOV label and the following × one or more label blocks, the first field of which must be EOV or UTL (user trailer label), are permitted, but not checked by MTS.

Appendix 2.

Error Reactions

In error cases an error message is written on current output and execution is terminated immediately.

However, in some special cases you may influence whether the execution shall stop or how to continue by the procedure blockproc (page 5), which is called on the basis of identification array (page 3), these cases are marked with x in the table below.

Format of Error Messages

Contents of an error message is as follows:

```
MTS , <name of procedure>:
<information>
<reason>
line - number - message from Running System.
```

Contents of Error Messages and Explanation

	<reason>	<information>	explanation
	z.state <integer>		zone state <integer> not allowed
	paramno <integer>		error associated with parameter No. <integer>
	label 2		length of required label is not permitted
	label 3		length of user label is not permitted
	label 4		name of required label is not permitted
	label 5	reel:<name of reel> <name of required label>	name of user label is not permitted
	label 6	<filesectionno.> <fileno.>	filesectionno. not corresponding to that in label
	label 7		fileno. not corresponding to that in label
	label 8		No. of blocks not corresponding to that in label
1 ×	label 9		checksum not corresponding to that in label
2 ×	error <status>	reel:<name of reel> <status> text	standard action in input/output has given up
	error 1 <status>		hard error in label input/output
3 ×	volume <actual No. of reels>	reel:<name of last reel>	actual No. of reels is too small
	volume <actual No. of reels>	reel:<name of last reel>	increasing actual No. of reels by tapeadp is impossible

<status> is the logical status word, the meaning of which is mentioned in 'Algol 5 Users Manual' (6.3.3)

<u>re. 1 x</u> If you use checksum writing and checking you may decide what to do in case of checksum error. Should be applied in case that you want to influence situations of parity error.

<u>re. 2 x</u> It is possible to influence on further execution after parity error (page 3 and 5)

<u>re. 3 x</u> It is possible to increase actual number of reels if max. number of reels > actual number of reels. (page 5).