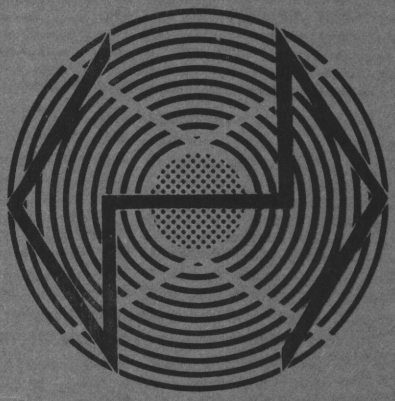


A/S REGNECENTRALEN

SCANDINAVIAN INFORMATION PROCESSING SYSTEMS

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

RCSL No: 31-D198

Edition: August 1972

Author: Chr. Gram, T. Aris

Title: MAGNETIC TAPE SYSTEM MTS 2

Keywords: RC 4000, Software, Magnetic Tape, File Handling, File Labels

Abstract: The report describes the set of procedures for handling labelled files on magnetic tape. The handling comprises label creation and check, positioning of tapes, use of variable record length. 32 pages.

CONTENTS

Section Page

1. INTRODUCTION 3
2. COMPATIBILITY AND CHANGES, MTS - MTS 2 3
   2.1. Compatibility between MTS and MTS 2 3
   2.2. What is New in MTS 2 4
3. SURVEY OF PROCEDURES AND OPERATIONS 4
   3.1. I/O Operations 4
   3.2. File Administration- Zone Handling 5
   3.3. Zone Declaration 5
4. I/O PROCEDURES INVAR, OUTVAR, CHANGEVAR, CHECKVAR 6
   4.1. Integer Procedure Invar 6
   4.2. Integer Procedure Outvar 7
   4.3. Integer Procedure Changevar 7
   4.4. Integer Procedure Checkvar 8
5. MTS 2 PROCEDURES 8
   5.1. Procedure Openadp 10
   5.2. Procedure Positionadp 10
   5.3. Procedure Openandpos 11
   5.4. Procedure Closeadp 12
   5.5. Procedure Tapeadp 12
6. STANDARD ERROR ACTIONS 13
7. FILE AND LABEL STRUCTURE 17
8. EXAMPLES 22
   8.1. Simple Use of Variable Length Records 22
   8.2. Several Files copied to one File 22
   8.3. Updating of register with creating and cancelling 23
   8.4. Use of Tapeadp 27
9. REFERENCES 27
10. Procedure stderroradp 28

MAGNETIC TAPE SYSTEM 2

1. INTRODUCTION

The Magnetic Tape System 2 MTS 2, is a set of procedures for administering labelled files on magnetic tapes in Algol and Fortran programs.
The MTS 2 procedures create and check the file labels and substitute thus the algol standard procedures open, close and setposition. The data transfer itself is performed by means of the standard procedures inrec and outrec, or by means of two extensions of these, invar and outvar better suited for handling files with variable record length.
MTS 2 administers three file types:

Single-volume and Multi-volume files,

Multi-file volume,

but not multi-file multi-volume. (sec.7)
The labels follow the ISO standard [ref.1] which is completely identical with the USA standard [2] and the ECMA standard [3].
The user is supposed to be familiar with the algol system as described in the algol manual [4].

2. COMPATIBILITY AND CHANGES, MTS - MTS 2

The MTS 2 is designed to be used in programs under Boss 2 but may also be used in programs under Boss 1 or s (see 2.2.D)

2.1. Compatibility between MTS and MTS 2.

Magnetic tape files created by means of the old MTS can be handled under MTS 2 by the standard in/out procedures and vice versa, but if the new in/out procedures invar, outvar are used for generating a file, this file cannot be handled by an old program because of a change in the record structure.

2.2. What is New in MTS 2.

New procedures: openandpos for handling of tape files, i.e. openandpos combines openadp and positionadp (see 5.4.)

stderroradp to be used (straight or changed slightly) as block procedure for zones used by MTS.

Changes in existing procedures of MTS. The procedures are also described in this manual (see 5):

A. Call of block procedure, if user mask says so, also in following cases (see 5):
   1. end of tape - single volume
   2. max. no. of reels too small - multi volume
   3. error in record checksum
   4. error in label
   5. block cunt error
   6. output to operator's log

B. Creation date in labels (see 7).

C. Preparation of zones for invar and outvar (see 4).

D. Under system 2 the VOL labels will be checked by MTS2 in case the tapes are not mounted before called for. Under system 3 no check of VOL label, Boss 2 cares of identification of labels (see 7). For programs not run under Boss 2 it is recommended to check VOL labels by the FP-Utility program check label.

E. Under system 3 MTS 2 will place <:mt:> in front of the text in Ident(16) in the open calls, e.g. if Ident(16) = 285024, the zone will be opened to mt285024.

3. SURVEY OF PROCEDURES AND OPERATIONS

3.1. I/O perations.

I/O is performed by means of the standard algol procedures:

inrec, outrec, changerec, inrec6, outrec6, changerec6, invar, outvar, changevar (for record handling)

read, readchar, readstring, readall (for character input)

write (for character output)

The procedures invar, outvar, changevar, and checkvar are described in detail in section 4 below, the others are described in [4].

3.2. File Administration - Zone Handling.

The zone and the document are handled by means of the procedures described in 5.

openadp: connects reels to the zone,
sets information into the zone about number of
reels, parity mode, error mask etc.

positionadp: prepares the zone for input or output,
takes care of positioning of the tape to the described file, and
the label handling.

openandpos: combines openadp and positionadp
it connects the volume file to the zone and prepares the zone
for in/out handling by giving it information.

closeadp: terminates the use of the zone and the volume file handling.

tapeadp: add new tapes (multi volume)

Errors, tape marks, end of tape etc. are handled (as usual) by cll of the block procedure given in the zone declaration. It is recommended to use:

stderroradp: print a message to the perator and continue the job in most cases.
Only fatal errors cause stopping of the job.
May also produce a logging of all label handling.

3.3. Zone Declaration.

The zone must be declared as usual in algol or fortran, but with a change in the interpretation of the first parameter 'buf': If the user wants a zone buffer area of length N, the value of the first parameter must be

buf = N + length_of_ident

where length_of_ident is the length (in reals) of an array with file ructure information used in the procedure openandpos (openadp). If only one reel is to be used on the zone, we have length_of_ident = 16 , and in general we have

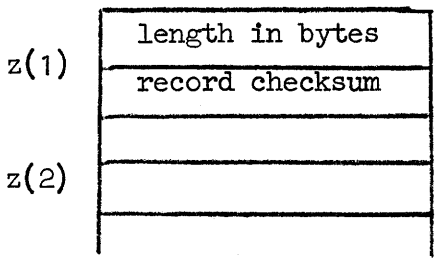
length_of_ident = max. no. of reels + 15.

The contents of this array is described in section 5.

It is recommended to declare the zone with 2 shares for efficient use of the tapes and in many programs the standard block procedure stderroradp will suffice. But it is recommended to study the specifications of stderroradp carefully and modify it if it does not fulfill the wanted requirements. The procedure is designed as a reasonable simple and general tool, but is very easily extendable.

4. I/O PROCEDURES INVAR, OUTVAR, CHANGEVAR, CHECKVAR

The procedures invar, outvar are designed for easy handling of variable length records under MTS 2.
Each record produced by outvar contains the length and the checksum in its first (real) element:



Whether the record checksum will be used for checking or not is defined through a parameter in the call of openandpos (and the information is stored in the zone descriptor).
The procedures also allow record counting in the following sense: A counter in the zone descriptor is initialized to zero by positionadp, and each call of invar and outvar increases it by one. After the call getzone(z,ia) or getzone6(z,ia), record count = ia(11) extract 23.
The procedure specifications are in detail as follows:

4.1. Integer Procedure Invar.

Call: invar(z)

invar (return value, integer): The number of bytes left in the present block for further calls of invar exactly as for inrec6.

z (call and return value, zone): The zone used for input.

Function: Makes available the next record, the length of which is found in its first word. If the checksum flag in the zone descriptor is set, the check is carried out. If an impossible record length or a checksum error is found, the blockprocedure for the zone is called. Impossible recordlength is: odd, less than 4 or greater than left in block. Finally the record counter in the zone descriptor is increased by one.

Requirements: The zone must be opened and ready for inrec/invar.

4.2. Integer Procedure Outvar.

Call: outvar (z, A)

outvar (return value, integer): The number of bytes available for further calls of outvar before change of block takes place exactly as for outrec6.

z (call and return value, zone): The zone used for output.

A (call value, real array): A one-dimensional array in which first word of A(1) contains the number of bytes to be copied into the zone record.

Function: Creates a new zone record with the byte length stated in first part of A(1) and copies into it the elements A(2)... A(round (bytelength/2)/2). If the length is an odd number, the record will be one byte larger. The check sum of z(2)..., is calculated. z(1) will contain the record length in bytes and the checksum. If length<>0 and length<4 the procedure will break with the alarm message rec len <length>. The record counter in the zone descriptor is increased by one if record length > 0.

Outvar (z,z) will output a record identical to the last one.

Requirements: The zone z must be opened and ready for outrec/outvar.

4.3. Integer Procedure Changevar.

Call: changevar (z,A)

changevar (return value, integer). The number of bytes left in the present block for further calls of outvar exactly as for outrec6

z (call and return value, zone), the zone used for output.

Function: cancels the last record output on z and replaces it by A as for outvar (z,A).

The record counter in the zone descriptor is decreased by 1 if record length = 0.

The call changevar (z,z) will change the length of the record to the new length in z(1), all values of z(i) will be intact.

Requirement: The zone must be open and used for outvar.

4.4. Integer Procedure Checkvar.

Call: checkvar (z)

checkvar (return value, integer) The checksum which was stored in the zone before call of checkvar.

z (call and return value, zone) the zone used for input/output.

Function: calculates the checksum and stores it in z(1) exactly as outvar The procedure should only be used if it is known that the checksum is destroyed and it will be demanded later on.

Requirements: The record must be placed by either outvar or invar.

If the length in z(1) differs from the record length in the zone description (ia(16)), the procedure will break with the alarm message checklen <length>.

The value of the checksum is

$$-\left(3 + \text{reclength} + \sum_{i=2}^{\text{reclength}//4} z(i,\text{first}) + z(i,\text{second})\right)$$

i.e. the total sum of the record is -3.

An example of the use of these procedures is found in section 8.

5. MTS 2 PROCEDURES

Before using a magnetic tape file, the user must describe the structure of the relevant volumes and files to the MTS 2. For each document the description must be available in an real array, here called Ident, as follows. This information is fetched by openandpos (openadp). The array must be declared as

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

Contents of Ident:

At the time of calling openandpos (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. If ident(2)>1 then fileno must be 1.

Ident(3) = actual number of reels.

Ident(4) = mask for call of the users block procedure. Ident(4) substitutes the give-up-mask, used by the algol standard procedure open. In this mask 12 bits are used to define when a call of the block-procedure is wanted:

1 shift 0 for: tapemark after datafile is read.

1 shift 1 for: parity error was detected during block transfer.

1 shift 2 for: number of actual reels of a file is too small (relevant only for multi volume file).

1 shift 3 for: maximal number of reels of a file is too small (relevant only for multi volume file).

1 shift 4 for: end of tape is read for a single volume file.

1 shift 5 for: error in record checksum.

1 shift 12 for: checksum error in datafile or volume.

1 shift 13 for: user input from labels is wanted.

1 shift 14 for: user output to labels is wanted.

1 shift 15 for: error in label, i.e. hard error (parity or block length error), error in label name, file number, etc.

1 shift 16 for: block counting error in data file or volume.

1 shift 23 for: output to operators log is wanted for each label writing and each label checking.

Ident(5) - Ident(15): irrelevant.

Ident(16) = name of first reel ($\leq$ 6 characters). Preferable this name should be the hardware number of the mag.tap. e.g. 285024.

Ident(17) = name of second reel.

.

.

.

Ident(15 + actual number of reels) = name of last reel.

possible remainder of Ident: irrelevant.

5.1. Procedure Openadp.

Call: openadp(z, Ident, file_checksum);

z (call and return value, zone): in/out zone.

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

file_checksum (call value, boolean): true if checksum calculation and checking is required; the checksum is calculated file section wise (see 7).

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.

Requirements: 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).

5.2. Procedure Positionadp.

Call: positionadp(z, fileno, op);

z (call and return value, zone): in/out 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.

Function: Positionadp completes latest operation, and if latest operation was output an EOF label is written on the tape. 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.

Requirements: 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.

5.3. Procedure Openandpos.

Call: openandpos(z, Ident, filechecksum, fileno, op);

z (call and return value, zoen): in/out zone.

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

filechecksum (call value, boolean): true if checksum calculation and checking is required; the checksum is calculated file section wise (see 7).

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.

Function: Openandpos 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 openandpos writes on the console:

mount <name of reel>,

advising the operator without waiting for mounting.
The first reel is connected to the zone.
If the first volume is not mounted the console message:

mount <name of reel>

is written, and mounting is waited for.
The procedure starts the positioning to the specified datafile. The state of the zone is set to after_positionadp = 0.
Openandpos is identical to openadp + positionadp.

Requirements: 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).
if multi-volume file then fileno must be 1 else fileno > 0,
1 <= op <= 3.

5.4. Procedure Closeadp.

Call: closeadp(z, action);

z (call and return value, zone): in/out 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.

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.

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

0 <= action <= 3

5.5. Procedure Tapeadp.

Call: tapeadp(z, name);

z (call and return value, zone): the in/out zone.
name (call value, real): contains the name of a reel - 6 characters.

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

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

actual number of reels <= max number of reels.

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 con-

nected to the zone.
An example 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 2 will call blockproc (see 3.3) 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.

6. STANDARD ERROR ACTIONS

The block procedure called stderroradp handles all error cases and the operator log in a way satisfactory for most programs. The procedure is designed to continue the job whenever possible. The activation of the procedure depends on the user mask given in Ident(4) (see 5) and on the result of a tape activation: The procedure is called when one of the events occur for which the user has set the corresponding bit in Ident(4), see above.
The procedure stderroradp acts as described in the table below. If more than one event occur simltaneously they are handled in the order of the table.
Bits No. 13 and 14 concerning input/output of private user information in the labels are ignored by stderroradp.

csition in dent (4) from ight	Event	Action	Message, always preceded by: MTS, reel: <document name> (On Current Output)
0	Tapemark after datafile	All other events are skipped, if any.	
1	Parity error	Continue with next block	Parity error. File<No.> , block<No.> , bytes skipped =<No>
2	No. of actual reels too small	For output: Call tapeadp to ask for new tape. For input: Stop job	For input: Number of input reels too small = <No.> Job stopped.
3	Max. No. of reels too small	close, stop job.	Max. No. of reels too small = <No.> Job stopped.
5	Record checksum error or record length error	Skip whole block and continue	record length error or checksum in record= <value>, calculated=<value> File <No.> , block<No.> , record <No.>, record length=<No> No of bytes skipped = <No.>
4	End of tape, single volume	close, stop job	End of tape. File <No.> , block<No.> , recordno <No.> Job stopped.
12	File checksum error	Continue	File checksum = <value>, error = <value>. File <No.>
15	Label error	Try to reposition the tape up to 5 times. After that stop the job.	First 5 times: Label error. File <No.> . Tape repositioned. 6th time. Repeated label error on file <No.> Job stopped.
16	Block count error	Continue	Block count = <value> , error = <value>. File <No.>
23	Label checking.	Continue	Label <input or output>, record count = <value> all fields of label as characters, checksum as integer For all messages: Something = <value>, error = <value> Something is the value stored in the zone (label) and error is the calculated value.

When debugging a program it may be worth while to set all the bits of the mask, i.e. to initialize with

```
Ident(4)  := -1.
```

For running programs without operator log, without user labels on the magnetic tapes, but with complete error checking and handling, it is recommended to set all bits except No.s 13, 14, and 23, i.e. to initialize with

```
Ident(4)  := -1 - (1 shift 23 + 1 shift 14 + 1 shift 13).
```

The events corresponding to positions No. 1, 2, 3, 4, 12, 15, 16 are considered serious in the following sense: If the user has not set the corresponding bit in the mask and thus specified no action, the system will stop the job with an alarm message. Below is a list of messages from MTS2, the two first are caused by programming errors.

```
MTS, <procedure>
z.state <zone state>

MTS, <procedure>
<name of error parameter> = <value>

MTS,
reel <name>
file No <No>
record count = <No>
status:
   <cause 1>
   <cause 2>
   ....
bytes <No>

MTS
reel <name>
file No <No>
label error:
status:
   <cause 1>
   <cause 2>
   ....
bytes <No>

MTS,
reel <name>
label error: length
name: <name>, reel No <No>, file <No>
bytes <No>
```

```
MTS,
reel <name>
label eroror: name in label <name>
name: <name>, reel No <No>, file <No>
bytes <No>

MTS,
reel <name>
label error: reel No in label <No>
name: <name>, reel No <No>, file <No>
bytes <No>

MTS,
reel <name>
label error: file No in label <No>
name: <name>, reel No <No>, file <No>
bytes <No>

MTS,
reel <name>
label error: blockcount <No>, in label <No>
name: <name>, reel No <No>, file <No>
bytes <No>

MTS
reel <name>
label error: fielchecksum
name: <name>, reel No <No>, file <No>
bytes <No>

MTS, <procedure>
reel <name>
end of tape
volume <No>
```

7. FILE AND LABEL STRUCTURE

File and label contents are structured according to 'Proposed USA Standard for Magnetic Tape Labels for Information Interchange', [ref. 2].
The following 4 cases of file structures are to be handled, where

-> denotes start of tape,

* means tape mark,

The contents of the various labels are specified in detail below.

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 ×

data blocks of first file

× EOF ×

.

.

.

HDR ×

data blocks of mth and last file

× EOF ××.

4. Volume without Files

-> VOL ×

Tapes used by the MTS 2 must be named, i.e. must start with a VOL label.

FORMAT AND CONTENTS OF LABELS: Each label written by MTS 2 is a 84 character block, label block lengths tolerated by MTS 2 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 Boss 2
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 2.

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 2
1	HDR	3	yes
2	1	1	yes
3, 4	spaces	23	no
5	reel No.	4	yes
6	file No.	4	yes
7, 8	000000	6	no
9	creation date, consisting of:		no
	space	1	
	year	2	
	day of year	3	
10	space 00000	6	no
11	space	1	
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	Reel 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 2.

In reference 1, reel is called file section.

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 2
1	EOF	3	yes
2	1	1	yes
3- 11	as the preceeding standard HDR label		
12	No. of data blocks after preceeding standard HDR	6	yes
13, first part	checksum of data blocks after the preceeding standard HDR	3	yes
13, last part and 14	spaces	17	no
15	circumflexes	4	no

Between this EOF label and the following x 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 2.

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 2
1	EOV	3	yes
2	1	1	yes
3-11	as the preceeding standard HDR label		
12	No. of data blocks after preceeding standard HDR	6	yes
13, first part	checksum of data blocks after preceeding standard HDR	3	yes
13, last part and 14	spaces	17	no
15	circumflexes	4	no

Between this EOV label and the following x 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 2.

8. EXAMPLES

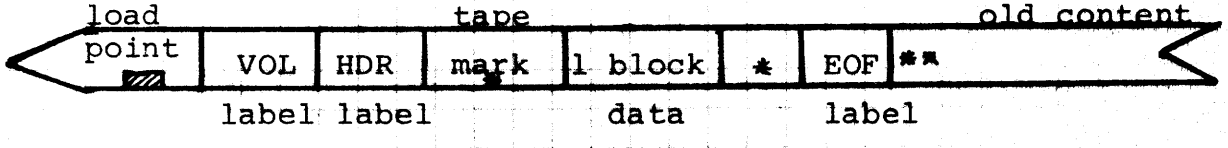
8.1. Simple Use of Variable Length Records.

A file may be copied from one tape to another via two zones OLDFILE and NEWFILE by an algorithm approximately as follows:

```
begin
   array Ident(1:16); name(1:2);
   zone OLDFILE, NEWFILE (116, 1, stderroradp);
   Ident(1):= 0; Ident(2):= Ident(3):= 1; Ident(4):=
   -1-(1 shift 23+1 shift 14+1 shift 13);
   readstring(in, name, 1);
   Ident(16):= name(1);
   openandpos (OLDFILE, Ident, true, 1,1);
   readstring(in, name, 1);
   Ident(16):= name(1);
   openandpos (NEWFILE, Ident, true, 1,2);
   Next_rec: invar (OLDFILE);
             comment update in the OLDFILE record;
             outvar (NEWFILE, OLDFILE);
             goto Next _rec;
END:
```

The updating could also be done after the call of outvar, in the NEWFILE record, but that would spoil the record checksum generated by outvar.

Picture of the tape:



8.2.

File number 1, 2, ..., 5, with max. blocklength = 100 double words on mt285010 are copied to one file on mt285020:

```
begin
   boolean eof;
   integer fileno, p, i;
   array Ident(1:16); name(1:2);
   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;
   readstring(in, name, 1);
   Ident(16):= name(1); openadp(zi, Ident, true);
   readstring(in, name, 1); name(1);
   Ident(16):= name(1); openandpos(zo, Ident, true, 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);
endinput:
285010
285020
```

8.3.

mt285104 is a one file register tape, with records as follows, sorted after account number

byte	1-2	length of record in bytes
	3-4	checksum
	5-6	account number
	7-10	amount
	11-	name and address

mt285612 is a one file transaction tape, with records as follows, sorted after account and transaction type number.

byte	1-2	length of record in bytes
	3-4	checksum
	5-6	account number
	7-8	transaction type
	9-12	amount
	13-	name and address

type =	1	create
	2	change name (byte 9-12 = 0)
	3	update amount (record length = 12)
	0	cancel (record length = 8)

The following program updates the register on tape mt285103.

Input:

285104

285612

285103

```
begin
integer action, waiting, actual, endnumber,k,i;
integer field key, reclength, type;
real field regamount, transamount;
real array field regname, transname;
boolean checkerror;
zone new,old,trans(216,2,endadp);
array ident(1:16), name(1:2);

procedure endadp(z,s,b); zone z; integer s,b;
begin integer array ia(1:20);
  if s extract 1=0 then stderroradp(z,s,b) else
  begin
    getzone6(z,ia); ia(16):=key; setzone6(z,ia);
    b:=key;
    z.reclength:=key; z.key:=endnumber;
    checkvar(z);
  end
end endadp;
```

```
  ident(1):=0;
  ident(2):=ident(3):=1;
  ident(4):= 1;
  readstring(in,name,1);
  ident(16):=name(1);
  openandpos(old,ident,true,1,1);
  readstring(in,name,1);
  ident(16):=name(1);
  openandpos(trans,ident,true,1,1);

  readstring(in,name,1);
  ident(16):=name(1);
  openandpos(new,ident,true,1,2);

  reclength:=2; regamount:=10; transamount:=12;
  regname:=10; transname:=12; type:=8; key:=6;

  invar(old);
  invar(trans);
  action:=1; checkerror:=false; endnumber:=8000000;

repeat:
  waiting:=if action=1 then endnumber else new.key;
  actual:=old.key; action:=4;
  if trans.key=actual then action:=6 else
  if trans.key<actual then
    begin actual:=trans.key; action:=2 end;
  if waiting=actual then action:=action+1 else
  if waiting<actual then action:=1;

  case action of
  begin

    begin comment 1, not more for this one;
      if checkerror then checkvar(new);
      checkerror:=false;
    end;

    begin comment 2, the transaction is a creation or something is wrong;
      if trans.type<>1
      then write(out,<:<10>transaction for nonexisting account:>)
      else
      begin
        trans.reclength:=trans.reclength-2;
        trans.regamount:=trans.transamount;
        k:=(trans.reclength-10)//4;
        for i:=1 step 1 until k do trans.regname(i):=trans.transname(i);
        outvar(new,trans);
      end;
      invar(trans);
    end;
```

```
    begin comment 3, trans=new;
      if trans.type=0 then
      begin comment cancel;
        action:=1; checkerror:=false;
        new(1):=0; changevar(new,new);
      end
      else
      if trans.type>3 then write(out,<:<10>unknown transaction type:>)
      else
      begin comment update or change name;
        checkerror:=true;
        if trans.type=3 then new.regamount:=new.regamount+trans.transamount
        else
        begin
          if new.reclength<>trans.reclength-2 then
          begin comment the new name has another length than the old name;
            new.reclength:=trans.reclength-2; changevar(new,new);
          end;
          k:=(trans.reclength-10)//4;
          for i:=1 step 1 until k do new.regname(i):=trans.transname(i);
        end
      end;
      invar(trans);
    end;

    begin comment 4, trans>old;
action4:
      outvar(new,old);
      invar(old);
    end;

    begin comment 5, two old with same number, should be impossible;
      write(out,<:<10>error in oldfile:>);
    end;

    begin comment 6, old=trans, first old is transferred to new and
      next action will be 3;
      goto action4
    end;

    begin comment 7, new=old=trans, either an error or endnumber;
      if  old.key<>endnumber
        then write(out,<:<10>action 7 error:>);
      goto finis
    end;

  end case;

  goto repeat;

finis:

  closeadp(old,2);
  closeadp(trans,2);
  closeadp(new,2);

end
```

8.4.

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:>;
   openandpos(z, Ident, true 1, 2);
   finis:= false;
   for p:= 1, p+1 while -, finis do
   begin outrec(z, 10); ...; comment filling;
   end;
   closeadp;
end
```

9. REFERENCES

1. ISO Standard (ISO/TC 97, 181E): 'Magnetic Tape Labelling and File Structure for Information Interchange'. June 1967.
2. 'Proposed USA Standard for Magnetic Tape Labels for Information Interchange.' Comm. ACM vol. 10 (No.11, Nov. 1967)
3. ECMA Standard: 'Standard for Magnetic Tape Labelling'' Geneve, Nov. 1967.
4. S. Lauesen: 'Algol 5, User's Manual.' RCSL No. 55-D42, Regnecentralen 1970.
5. Ellen Randløv: 'Exchange of Magnetic Tapes with other Computers.' RCSL: No. 31-D4, Regnecentralen July 1971, Danish edition.

10. Procedure stderroradp

```
external
procedure stderroradp(z,s,b);
zone z; integer s,b;
begin real q,r;
boolean output;
own boolean closecalled;
own integer rewind;
integer array ia(1:20);
integer i,f,g,j,file,block,record,reels,maxreels,checksum;

procedure writechar(r,from,to); real r; integer from,to;
begin integer i;
  for from:=from step 8 until to do
  begin
    i:=r shift from extract 8;
    if i>32 and i<126 and
      -,(i=35 or i=36 or i=39 or i=64 or i=94) then
      write(out,false add i,1) else
      write(out,<<d>,<:<60>:>,i,<:>:>);
  end;
end writechar;

procedure getfromadp(file,block,record,output,reels,maxreels,
                     checksum);
integer file,block,record,reels,maxreels,checksum;
boolean output;
begin integer j,j1,j2,j3,j4,j5,j6,j7,j8,j9,j10,j11,j12,
        j13,j14,j15,j16,j17,j18,j19,j20,j21,j22,j23,j24;
own real volname; real r;
  if s extract 6=0 then j:=4 else j:=1; adpcode(z,j);
  file:=j9;
  block:=j1;
  record:=j24 shift 1 shift (-1);
  output:=j6>1;
  reels:=j21;
  maxreels:=j10;
  checksum:=j2;
  r:=real<::> add j12 shift 24 add j13;
  if r<>volname then
    begin volname:=r; rewind:=0 end;
  if -,(s=1 shift 13 or s=1 shift 14) then
  begin
    write(out,<:<10>MTS<10>reel: :>);
    for j:=8,j-8 while r shift j extract 8=0 do;
    writechar(volname,-40,j);
  end;
end getfromadp;
```

```
getfromadp(file,block,record,output,reels,
           maxreels,checksum);

getzone6(z,ia);

if s shift ( 0) extract 1=1 then
begin comment tapemark read after datafile;
  b:=4;
  goto EXIT;
end;

if s shift ( -1) extract 1=1 then
begin comment parity error;
  write(out,<:<10>Parity error. File:>,file,
            <:, block:>,block,<:, bytes skipped=:>,b);
  b:=ia(16):=0;
end;

if s shift ( -2) extract 1=1 then
begin comment number of actual reels too small;
  if output then tapeadp(z,real<::>) else
  begin
    write(out,<:<10>Number of input reels too :>);
    system(9,reels,<:<10>small = :>);
  end
end;

if s shift ( -3) extract 1=1 then
begin comment max number of reels too small;
  write(out,<:<10>Max No. of reels too small = :>, maxreels);
if -,closecalled  then
begin comment closeadp is only called once;
  closecalled := true;
  closeadp(z,3);
  closecalled := false;
end close;
end;

if s shift( -5) extract 1=1 then
begin comment record checksum error;
  i:=ia(16); j:=ia(15)-ia(14);
  if i mod 2=1 or i<4 or i>j then
    write(out,<:<10>record length error:>)
  else
    write(out,<:<10>checksum in record=:>,z(1) extract 24,
              <:, calculated=:>,b);
  write(out,<:<10>file=:>,file,<:, block=:>,block,
            <:, record=:>,record+1,<:, record length=:>,i,
            <:<10>no. of bytes skipped=:>,j);
  b:=ia(16):=0;
end;
```

```
  if s shift ( -4) extract 1=1 then
  begin comment End of tape, single volume;
    write(out,<:<10>End of tape. File:>,file,<:, block:>,block,<:,<10>recordno:>,
  record);
  if -,closecalled then
  begin comment closeadp is only called once;
    closecalled := true;
    closeadp(z,3);
    closecalled := false;
  end close;
  end;

  if s extract 6>0 then goto EXIT;

  if b<>56 and b<>52 then
    write(out,<:<10>Label length should be 52/56 bytes, but is:>,b)
  else

  if b>23 and (s shift (-12) extract 1=1 or
     s shift (-15) extract 2 <>0 )  then
  begin
    write(out,<:<10>Name in label: :>);
    writechar(z(1),-40,-16);
    r:=z(5) shift 24 add
       (z(6) shift (-40) extract 8 shift 16);
    q:=48 shift 40 add 48 shift 32 add
       (48 +reels//10) shift 24 add (48+reels mod 10) shift 16;
    if r<>q then
    begin
      write(out,<:<10>Reel No.:>,reels,<:, reel No in label :>);
      writechar(r,-40,-16);
    end;
    r:=z(6) shift (-8) shift 16;
    q:=(48+file//1000) shift 40 add
       (48+(file//100) mod 10) shift 32 add
       (48+(file//10) mod 100) shift 24 add
       (48+file mod 10) shift 16;
    if r<>q then
    begin
      write(out,<:<10>File:>,file,<:, file No in label :>);
      writechar(r,-40,-16);
    end;
  end;

  if b>43 and s shift(-12) extract 1=1 then
  begin comment filechecksum error;
    write(out,<:<10>File checksum=:>,z(11) shift (-24) extract 24,
              <:, error=:>,checksum,<:. File:>,file);
  end;

comment reading and writing on label
```

```
if s shift (-13) extract 1=1 then
begin
  comment z(1) to z(14) is a HDR, EOF or EOV label
  r:=z(8)
  r is now creation date year+day add 32 e.g. <:71033:>add 32
    for February 2nd, 1971
  q:=z(9)
  q is now expiration date similar to r
  ...
  ...
end label read

if s shift (-14) extract 1=1 then
begin
  comment expiration date:=100 days from today
  r:=-100
  r:=rx24x60x60
  systime(1,r,r)
  j:=systime(2,r,r)
  i:=j//10000+(if j mod 4=0 and j//100 mod 100>2 then 1 else 0)
     +(case (j//100 mod 100) of
      (0,31,59,90,120,151,181,212,243,273,304,334))
  z(9):=real<::> add
     (j//10 mod 10 + 48) shift 8 add
     (j mod 10 + 48) shift 8 add
     (i//100 + 48) shift 8 add
     (i//10 mod 10 +48) shift 8 add
     (i mod 10 +48) shift 8 add 32
  ...
end label written;

if s shift (-15) extract 1=1 then
begin comment label error;
  if rewind=3 then
  begin
    write(out,<:<10>Repeated label error:>);
    system(9,file,<:<10>on file :>);
  end;
  getposition(z,f,g);
  rewind:=rewind+1;
  setposition(z,0,0);
  setposition(z,f,g);
  b:=ia(16):=0;
  write(out,<:<10>Label error. File:>,file,
            <:. Tape repositioned:>);
end;
```

```
  if b>39 and s shift (-16) extract 1=1 then
  begin comment blockcount error;
    write(out,<:<10>Block count=:>);
    writechar(z(10),-40,0);
    write(out,<:, error=:>,block,<:. File:>,file);
  end;

  if b>43 and s shift (-23) extract 1=1 then
  begin comment label checking;
    write(out,<:<10>Label :>,
              if output then <:output:> else <:input:>,
              <:,  record count=:>,record,<:<10>:>);
    for i:=1 step 1 until 10 do
    begin write(out,<:<10>:>);
      writechar(z(i),-40,0);
    end;
    write(out,<:<10>:>,z(11) shift (-24) extract 24,<: :>);
    writechar(z(11),-16,0);
    for i:=12,13,14 do if i×4<=b then
    begin write(out,<:<10>:>);
      writechar(z(i),-40,0);
    end
  end;

EXIT:
  write(out,<:<10>:>);  setzone6(z,ia);

end stderroradp;

message rc 1972.03.10, stderroradp text;

end
```