CHD

Title:

TELEDATA UTILITY PROGRAMS

A/S REGNECENTRALEN

RC SYSTEM LIBRARY: FALKONERALLE 1 DK-2000 COPENHAGEN F

RCSL No: 21-T008

Edition: March 1978

Author: Carl Henrik Dreyer
Benny Hammarkvist
Ivan Larsen
Jørgen Lindballe

Keywords: TELEDATA, RC8000, utility programs, administration, safety.

Abstract: The Teledata utility programs are used for routine administration of a Teledata system. They cover the functions: control of runs including automatic recovery, tape administration and safety dumping, safety checking of disc files, fast copying of disc files, extraction of log.

English edition, pages: 110.

CONTENTS: Page

0. Introduction

The present release 2 includes the programs basic to a routine TD system. More will be added later.
A few general notes should be stated here, valid for all or most of the utility programs.

Program description subsections

Every program description is devided in the following subsections (some of which may be missing):

N	purpose
N.1	example
N.2	Call
N.3	Explanation of the parameters
N.4	Function
N.5	Requirements
N.6	Messages from program
N.7	Listings from program
N.8	Further examples
N.9	Error messages.

Program call and FP parameters

The rules mentioned below are valid for all utility programs. Exceptions (if any) are mentioned explicitly in the program descriptions.

A program call is followed by a number of parameters. These parameters are devided in paramter groups.

parameter group

A parameter group must be be started by a space. The group consists of one or several parameters separated by points. The first parameter is a keyword for the group. The groups can be stated in any order, and any number of times.

0. Introduction

If a group defines a run parameter, the last mentioned group will be valid. If the group is an activation of a program function, all the mentioned groups will be active in the stated order.

in.<infile>	Special keyword: in.<infile> When this group appears, the reading of parameters will proceed from <infile>, until an EM character is met. The syntax in <infile> is the same as in fp. After <infile>, the parameter reading proceeds from fp.

Catalog files used by utility programs

statusfile	The statusfile is read and updated by several utility programs. A description is found in section 6 (Telestatus).
tapecat	The tape catalog is read and updated by Teletape and Teleop. A description is found in section 4 (Teletape).
adm admcontrol	Teleadm is using the two textfiles adm and admcontrol. All the run files and command files are too used by Teleadm.

1. Teleadm

Teleadm controls the execution of a run; this means that it controls the proper calls of the (normal- and rescue -) command files specified in the current run file.

Once the teleadm program is started by the operator for a run, this control is performed by alternating call of a command file and call of the teleadm program; these calls are generated by the teleadm program itself.

1.1 Example

A run is started when the operator types:

```
i adm
```

The file "adm" contains:

```
(end
teleadm stat.sf
i admcontrol)
```

When the program asks, the operator types in the name of the run file:

```
teleadm, name of run file = rf
```

For example rf contains:

```
a, b, c
b, o
c, a
```

1. Teleadm

During the run the names of the command files are printed:

```
teleadm, a
teleadm, c
teleadm, a
teleadm, b
```

In this example the command file "a" is first called; the result is 2, so "c" is called and then "a" again. This time the result is 1; therefore "b" is called, and the run is finished.

A command file is a file containing fp-commands, f.ex. the command file "a" may contain:

```
telestatus filename.sf comres.2
teleop status.sf input.process
telestatus filename.sf comres.1
```

In this example the status file was "sf". The file named "admcontrol" is a small work file used only by teleadm for the calls generated, so after each call of a command file, teleadm is automatically called; data which should survive from call to call is stored in the status file.

1.2 Call

Teleadm is called by the fp-command:

```
i <name of adm.file>
```

Teleadm does not care about the name of the adm.file.

1. Teleadm

The adm. file must contain:

```
(end
teleadm stat. <name of status file>
i admcontrol)
```

When the program writes:

```
teleadm, name of run file =
```

the operator should type the name of the run file terminated by the new line character.

Please note that the first run after "close" must be "open". In this manual the names "open" and "close" are used for short; in fact their names must be"open <1 to 7 characters>" and "close <1 to 6 characters>", respectively.

If the run file is "open" teleadm also caculates the weekday from the machineclock and writes:

```
teleadm, weekday = <day of the week>
```

which should be checked by the operator; if the day calculated is correct, the operator must type the newline character, otherwise he must type any letters terminated by new line.

1.3 Explanation of the parameters

Usually the name of the adm.file will be "adm".

1. Teleadm

The file "admcontrol" is a working file used by the teleadm program, only. It must exist before the call of teleadm. This is of course valid also for the status - and the run file.
Teleadm uses the following fields in the status file.

Field	Length	Explanation
State	2	0: after "close"; 1: after "open".
Runfile	8	Name of current run file.
Comres	2	Result from previous command file.
Comfile	8	Name of current command file.

Except for the result (comres), which is written by the command files, they are all written and read by teleadm.

The run file is explained in details in chapter 1.4.

1.4 Function

1.4.1 Runs during the day

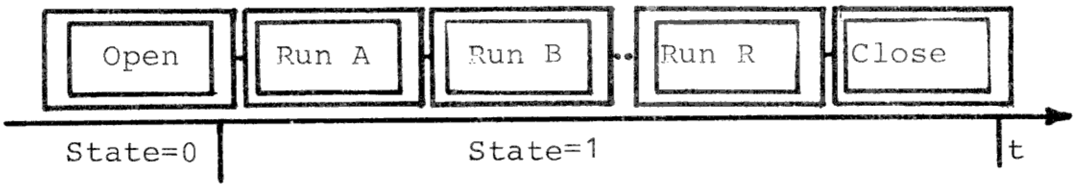


fig. 1.1

In figure 1.1 is shown that the runs of the day are initiated by "open" and finished by "close". By means of the internal variable "state" it is checked by teleadm that "open" is called after "close".

1.4.2 Execution of a run

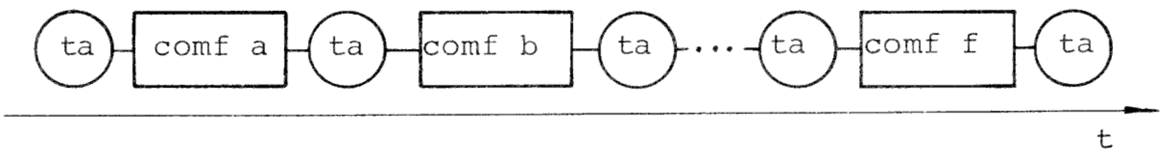


fig. 1.2

As shown in fig. 1.2 a run (open, close or any other run) consists of alternating call of teleadm (ta) and one of the command files (comf, specified in the current run file). Once called by the operator the remaining calls are automatically generated.
If a run has been terminated f.ex. due to power failure, it may be restarted by typing:

```
i adm
```

The contents of the status file determines whether this is a restart or a new start of a run.

1.4.3 The run file

The run file contains the names of the command files, normal as well as rescue, and the structure between them. For example a run file may contain:

```
a, b, c
b, 0, e
c, a, d
d, a,
e, b
```

which represents the command files a,b,..., e and their connections as shown in fig. 1.3 (the numbers represents the possible results, when executing the command file):

1. Teleadm

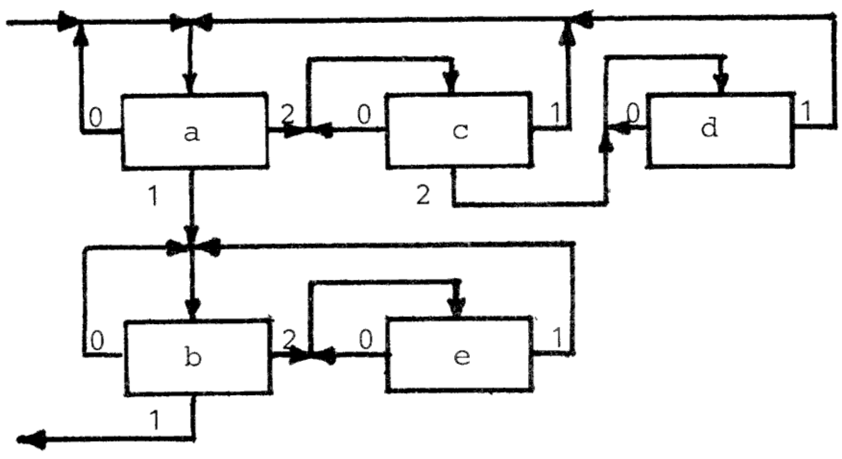


Fig. 1.3

The result of the execution of a command file is stored in the status file: status.syscat.comres. The value is reset to 0 by teleadm just before calling a command file. Usually the first and the last call in a command file will be calls of tele-status assigning the values 2 and 1, respectively, to comres. When returning to teleadm with the value of comres=2, this program will compare the counters transcur and transfin in the status file; if they are not equal, teleadm will add 1 to comres.

In this way the meaning of comres will be:

comres = 0:	The command file was not called due to some failure; it will be called automatically when teleadm is re-started.
comres = 1:	The command file was executed without any failure.

1. Teleadm

comres = 2: Due to some failure the command file was only partly executed; the counters status.safecat.transcur and status. safecat.transfin are equal.

comres = 3: Due to some failure the command file was only partly executed; the counters status.safecat.transcur and status. safecat.transfin are not equal.

The command files in the run file may be arranged according to this scheme.

The exact syntax of the contents of a run file is:

$$\langle\text{run file}\rangle ::= \left\{\langle\text{command line}\rangle\right\}^{\text{no of com.files}} \langle\text{EM}\rangle$$

$$\langle\text{command line}\rangle ::= \langle\text{com.file}\rangle\left\{,\ \langle\text{com.file}\rangle \middle| 0\right\}_{1}^{9} \langle\text{NL}\rangle$$

At present the upper limit of the number of different command files in a run file is 25, and the possible results of the execution of a command file are 0,1, ..., 9.

A command line consists of a name of a command file followed by the names of the command files, which should be executed depending on the result: 1,..,9 of the execution of the first mentioned command file.

The first line of the run file must contain the name of the first command file to be executed (in the example: "a"). The order of the following lines is arbitrary. A zero indicates "end of run", f.ex. if the result of the command file "b" is 0 the command file will be executed again; if the result is 1 the run will be finished, if the result is 2, the command file "e" will be executed.

1. Teleadm

1.5 Requirements

Size: minimum:	halfword
optimal:	-

Entries: 0

The 4 files: status file, run file, adm and admcontrol must exist before a run is started.

1.6 Messages

```
teleadm, <name of command file>.
```

Just before the execution of a command file, its name is printed by teleadm.

```
teleadm, next command file will be (type new
line if ok): <name of rescue file>
```

The last result was <>1, and the next rescue file is printed. The operator may type the below mentioned followed by the newline character:

the
(ter-

<empty>	The rescue file will be extecuted.
0	The run will be finished.
wait	The run will be temporarily terminated; by typing "i adm" teleadm will continue. (Note that this facility means that the name "wait" should not be used for any command file).

new

by tele-
the new-
etter
e status

<anything else>
If this is the name of a command file, it will be executed.

1. Teleadm

```
teleadm, next command file =
```

The operator is asked to type in the name of the next command file to be executed, because no command file was specified in the run file, or the command file does not exist.

```
teleadm, result = <result>.
```

After the execution of a command file, the result is printed.

```
teleadm, weekday = { monday | tuesday | . . . | sunday }
```

When initializing the run "open" teleadm calculates the day of the week from the machine clock. The day should be checked by the operator; if the day is correct, he must type the new line character, otherwise he must type any letters terminated by new line.

```
teleadm, end run:  <run name>
```

The run is finished.

1.8 Further examples

On the next page you will find an example of the stack of current input, the contents of " ˉ ınd ısed a ! !.

The operator may type what is mentioned below followed by the newline character:

0	The run will be finished
wait	The run will be temporarily terminated; by typing "i adm" teleadm will continue.
<anything else>	If this is the name of a command file, it will be executed.

i admcontrol)

1. Teleadm

```
STACK OF CURRENT        ADMCONTROL                  FP-COMMAND AREA
INPUT

c
                                                    i adm
adm
c
                                                    (end
                                                    teleadm stat.<status file>
                                                    i admcontrol)
c                       i <command file1>
                        (end
                        teleadm stat.<status file>
                        i admcontrol)
admcontrol
c
                                                    i <command file1>
<command file1>
admcontrol
c
                                                    <commands from file1>
admcontrol
c
                                                    (end
                                                    teleadm stat.<status file>
                                                    i admcontrol)
c                       i <command file2>
                        (end
                        teleadm stat.<status file>
                        i admcontrol)
admcontrol
c
                                                    i <command file2>
<command file2>
admcontrol
c
```

STACK OF CURRENT INPUT	ADMCONTROL	FP-COMMAND AREA
		<commands from file2>
admcontrol		
c		
		(end
		teleadm stat.<status file>
		i admcontrol)
c		
	<empty>	
admcontrol		
c		
c		

1.9 Errormessages

```
***teleadm, file: <name> does not exist
```

The file specified (the status file, run file, command file, etc) does not exist. For the command file the operator is asked to type in the name of the next command file to be executed. For the status file or run file the run is terminated.

```
***teleadm, no new comfile,
   previous comfile = <name>, result = <result>
```

The command file mentioned as previous command file has just been executed with the result specified, but the run file does not contain a new command file corresponding to this result. The operator is asked to type in the name of the next command file to be executed.

```
***teleadm, open should be run.
```

Open should be run after close. The run is terminated.

```
***teleadm, open should not be run.
```

The previous run was not close, so open should not be run. The run is terminated

EMNE TELEADM version 3.1			Side: 1 af 1
Udarbejdet af: JL/BIN	Afd.nr.: 4150	Dato: 18.5.1978	

Manualens afsnit 1.9. Der tilføjes mellem 4. og 5. fejludskrift:

```
***teleadm, run file: <run file> does not
   contain com file: <com file>
```

A discrepancy exists in the status file: the run file specified in:

statusfile.syscat.runfile

does not contain the command file specified in

statusfile.syscat.comfile.

This may be true if you have changed the contents of the abovementioned fields, or if you have changed the contents of the run file after an unnormal termination of a run.

Med venlig hilsen

[illegible]

Jørgen Lindballe

4. Teletape

This program, developed for maintaining the tape-catalog of the TELEDATA system, gives possibilities for creating, updating and printing of records in the catalog, furthermore it is possible to create new tapecatalogs by means of teletape.

4.1 Examples

4.1.1 ex. 1

```
teletape tapecat.catfile.init pool.1,
create.25.mtxxxxx1.mtxxxxx2------.mtxxxx25
```

The file catfile, which must exist, is initiated by teletape as a tapecatalog, i.e. a filehead is generated, cf sect. 4.5.1, the filehead is used only by teletape for administrative purposes, pool number one is selected and a poolhead, cf. sect. 4.5.2, is created, further 25 taperecords, cf. sect. 4.5.3, is generated, one for each tape created. These tapes can only be accessed through the key -pool.1-.

4.1.2 ex. 2

```
teletape tapecat.catfile pool.1,
extend.3.mtxxxx26.mtxxxx27.mtxxxx28 1st.pool
```

The pool from ex. 1 with 25 tapes is extende with 3 more tapes and a list of the content of the ntire pool is produced - list.pool.

4. Teletape

4.2 Call

The program is activated by means of the fp. command:

$$\texttt{teletape}\left\{\texttt{tapecat.<catname>}\left\{\texttt{.init}\right\}_{0}^{1}\right\}_{0}^{1}\ \texttt{pool.<poolno>,}$$

$$\texttt{create.<no\_of\_vers>.}\left\{\texttt{<tapename>}\right\}_{\texttt{no\_of\_vers}}^{\texttt{no\_of\_vers}}$$

$$\texttt{extend.<no\_of\_vers>.}\left\{\texttt{<tapename>}\right\}_{\texttt{no\_of\_vers}}^{\texttt{no\_of\_vers}}$$

$$\texttt{delete.<poolno>}$$

$$\texttt{insert.<vers>.}\left\{\texttt{<tapename>}\right\}_{1}^{*}$$

$$\texttt{change.<vers>.<tapename>}\left\{\begin{array}{l}\texttt{.name.<new name>}\\ \texttt{.version.<new vers>}\\ \texttt{.par1.<new par1>}\\ \texttt{.par2.<new par2>}\\ \texttt{.state.<new state>}\end{array}\right\}_{1}^{*}$$

$$\texttt{remove.<vers>}\left\{\texttt{.<tapename>}\right\}_{0}^{1}$$

$$\texttt{reserve.<no of tapes>}$$

$$\texttt{listout.<filename>}$$

$$\texttt{list.}\left\{\begin{array}{l}\texttt{all}\\ \texttt{pool}\\ \texttt{version.<vers>}\\ \texttt{tape.<tapename>}\end{array}\right\}_{1}^{1}$$

<catname>	::=	name - the name of the tapecatalogfile
<poolno>	::=	number - the number of the wanted pool
<no_of_vers>	::=	number - the number of steps in the cycle
<tapename>	::=	name - the name of an mt.
<vers>	::=	number - the version, as shortclock
<no_of_tapes>	::=	number - the number of tapes to be reserved
<filename>	::=	name - the name of the file in which output is to be printed.

4.3 Description of parameters

The parameters tapecat *) and pool must be present at every call of teletape, in the mentioned order. Tapecat determins the file and pool the pool on which teletape will operate; all other parameters assume that a catalogfile and a pool has ben selected.

4.3.1 Create is the keyword telling that a pool is to be generated, to carry out this command some more parameters are needed, namely: no_of_vers and a sequence of tapenames. No_of_vers indicates how many steps this pool shall contain, and at creation time, there must exist one tape per step -version- in the cycle, therefore no_of_vers tape-names must follow.

4.3.2 Extend is the keyword telling that the selected pool is to be extended by <no_of_vers> steps in the cycle. <no_of_vers> tapenames must be supplied.

*) The parameter tapecat may be omitted, if this is done, the tapecatalog is assumed to exist in a file with the name: tapecat.

4. Teletape

4.3.3 Delete is the keyword that causes deletion of the pool with the number <poolno> if the selected pool also have the number <poolno>, else the run is terminated with an alarm.

4.3.4 Insert, keyword indicating that a version, in the selected pool, and identified by <vers> is to be extended by one or more tapes. A version only consists of one tape just after creation.

4.3.5 Change indicates that the tape specified by <vers> and <tapename> and which must exist in the selected pool is to be changed. The change concerns one or more of the following fields:

```
tapename - 8 halfwords
version  - 4 halfwords
optional - 2x4 halfwords
state    - 2 halfwords
```

tapename is indicated by the subkeyword: name.
Optional is split into two parts, each of type text, indicated by the subkeywords:

```
par1
par2.
```

Version is informed as a shortclock yymmdd.hhmm.

4.3.6 Remove, gives possibility for removal of a single tape or a whole version in the selected pool. A version must be supplied as parameter, and if the version stands alone, all tapes of the version specified is removed, if the versionparameter is followed by a tapename only the specified tape is removed, from the specified version in the selected pool.

4.3.7 Reserve, reserves <no_of_tapes> in the next version in the cycle of the selected pool if there are tapes available, the run is terminated with an alarm otherwise; the names of the reserved tapes are listed.

tape1 , tape2 , tape3

4.3.8 Listout, keyword which must be followed by the name of an existing file; this file is then made current output, the run is terminated with an alarm if no such file exists.

4.3.9 List, gives various possibilities for listing the content of the tapecatalog, cf. sect. 4.2 - call.

4.4 Function of Teletape

Teletape is a maintenance program used for:

a. creating of taperecords
b. changing of taperecords
c. listing of taperecords
d. removing of taperecords

in a tapecatalog

The program also gives possibility for creation of new tapecatalogs.
All functions are controlled by means of fp. parameters, whithin which a certain order must take place, i.e. the first parameter MUST be the name of the tape-catalogfile and parameter number two MUST be the identification of the pool on which the program shall operate, until a new poolselecting parameter is met.

4. Teletape

4.9 Error messages

All error messages start with three asterisks followed by the name of the program, teletape, and a few words to classify the type of error, finally the run is terminated with an alarm containing the keyword, pointing out the action from where the error was detected.

4.9.1 List of error messages

```
***teletape, illegal param.group
```

```
***teletape, illegal function
```

```
***teletape, catfile: <filename> unknown
```

```
***teletape, no pool selected
```

```
teletape, too many extensions      ; this error does NOT
last included: <tapename>          ; terminate the run
```

```
***teletape, illegal delete
```

```
***teletape, illegal insert
```

```
***teletape, illegal subkeyword
```

```
***teletape, version not found
```

4. Teletape

```
***teletape, tapename not found
```

```
***teletape, subkeyword unknown
```

```
***teletape, too few tapes
```

```
***teletape, outfile: <filename> unknown
```

4.9.2 Example of an error message after the call

```
teletape tapecat.telecat
***teletape, catfile: telecat unknown
tapecat  1  line  xxxx - xxxx
```

The run is terminated and the ok-bit set to false.

4.10 Catalog structure

The tapecatalog is split into three sections namely: the fileheadsection, the poolhead section, and the taperecordsection; the following is a closer description of the content of each section. Do not read this section unless you intend to access the tapecatalog with your own programs.

4. Teletape

4.10.1 The filehead contains some information of importance for teletape to minimize the searching in the catalog, the fields are:

first poolhead	::=	segm. addr.
first taperecord	::=	segm. addr.
free poolhead	::=	segm. addr ; i.e. room for creation ; of next poolhead
free taperecord	::=	segm. addr ; as for free poolhead
length of poolhead	::=	number of halfwords
highest poolnumber	::=	max 760
no of taperecords pr. block	::=	i.e. halfword addr. of last record in a block.

The filehead also contains a socalled pooltable which supplies the address of a pool, when the pooltable is addressed (indexed) with the poolnumber.

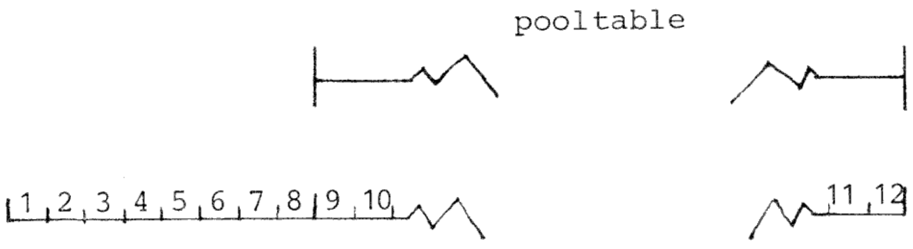
4.10.2 The poolheadsection consists of a number of poolheads each containing information about where to find the taperecords belonging to this pool, the number of versions this pool can handle and some information of administrative importance. The number of poolheads is limited to 760, because of the convenience by having the filehead, included the pooltable, within one block. The fields of a poolhead are as follows:

poolno	::=	the logic identication of the pool
vers. cycle	::=	number of steps in the versioncycle
used vers	::=	addr. of last written tape(s)
next vers	::=	addr. of tape ready for next writing
first taperecord	::=	addr. of first taperecord in the cycle
last taperecord	::=	addr. of last taperecord in the cycle

4.10.3 The taperecordsection contains the list of tapes in this tapecatalog. Physically the taperecordsection is placed at last in the file, to avoid limitation of the number of tapes a tapecatalog can hold; the taperecords are described below:

taperecord ref.	::=	if the fieldcontent is greater than -1 it is the addr. of the next taperecord belonging to this version.
pool ref.	::=	the logic identification of the pool to which this taperecord belongs, expressed as an integral number.
creation date	::=	shortclock for creation of this taperecord.
version	::=	shortclock for last writing on the tape described in this record.
tapename	::=	the name of the tape described in this taperecord.
optional	::=	field for the users own purpose
state	::=	0 = ununsed, >0 = used state times

4.10.4 Filehead



4. Teletape

1. addr of first poolhead
2. addr of first taperecord
3. addr of free poolhead
4. addr of free taperecord
5. length of poolhead
6. length of taperecord
7. highest poolno created
8. no. of taperecords pr. block
9. addr of first poolhead
10. addr of next poolhead
11. addr of poolhead no 759
12 addr of poolhead no 760 = last poolhead

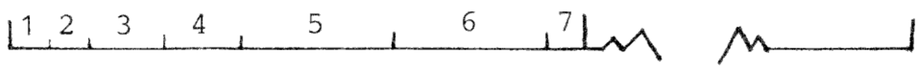
4.10.5 Poolhead

1
2
3
4
5
6
7
8
9
10
11
12

1. pool number (identification)
2. length of cycle (no. of versions)
3. used version (last written)
4. next version (for writing)
5. first taperecord in pool
6. last taperecord in pool

4. Teletape

4.10.6 Taperecord



1. next tape reference
2. poolhead reference
3. creation date
4. version
5. tapename
6. optional
7. state

6. Telestatus

The program is intended for off-line handling of a status file in the teledata system. Telestatus provides facilities for setting and/or inspection of any entry (field) in the status catalogs.

6.1 Examples

6.1.1

Ex. 1

```
telestatus filename.status clear.safecat,
                 print.syscat
```

Suppose a status file named 'status' exists on backing storage. The record constituting the safety catalog (safecat, see also section 6.3.2) is initialized to zero. The contents of the record constituting the system catalog (syscat) is printed on current output.

6.1.2

Ex. 2

```
telestatus filename.status tapelog.1.2,
                 print.safecat.logmode
```

In the status file named 'status' file# and block# for the tape log file are set to 1 and 2 respectively. The contents of the safety catalog is printed out together with a single entry (logmode) from the system catalog.

An example of a complete updating of a status file is shown in section 6.8. Examples on output from the program are shown in section 6.4.5.

6.2 Call

The program is called by the fp-command:

6. Telestatus

$$
\text{telestatus} \left\{ \begin{array}{l}
\text{filename.<name>} \\[2ex]
\text{<filekey>.} \left\{ \begin{array}{l} \text{<name>} \\ \text{<name>.<unsigned integer>.<unsigned integer>} \\ \text{<unsigned integer>.<unsigned integer>} \end{array} \right\} \\[2ex]
\text{<elementkey>.<unsigned integer>} \\
\text{<pointkey>.<unsigned integer>.<unsigned integer>} \\[2ex]
\text{checkpoint.} \left\{ \begin{array}{l} \text{yes} \\ \text{no} \end{array} \right\} \\[2ex]
\text{transac.} \left\{ \begin{array}{l} \text{yes} \\ \text{no} \end{array} \right\} \\[2ex]
\text{runfile.<name>} \\
\text{comfile.<name>} \\[2ex]
\text{clear} \left\{ \text{.<parameterkey>} \right\}_{1}^{*} \\
\text{print} \left\{ \begin{array}{l} \text{.<parameterkey>} \\ \text{.filename} \end{array} \right\}_{1}^{*}
\end{array} \right\}_{1}^{*}
$$

where:

$$
\text{<filekey>::=} \left\{ \begin{array}{l}
\text{secinp} \\
\text{replog} \\
\text{disclog} \\
\text{tapelog} \\
\text{cenout}
\end{array} \right\}
$$

6. Telestatus

$$\langle\text{elementkey}\rangle ::= \left\{\begin{array}{l}\text{transcur}\\ \text{transfin}\\ \text{logrecs}\\ \text{termcur}\\ \text{func}\\ \text{safemode}\\ \text{logmode}\\ \text{date}\\ \text{state}\\ \text{comres}\end{array}\right\}$$

$$\langle\text{pointkey}\rangle ::= \left\{\begin{array}{l}\text{discpoint}\\ \text{tapepoint}\end{array}\right\}$$

$$\langle\text{parameterkey}\rangle ::= \left\{\begin{array}{l}\text{all}\\ \text{checkpoint}\\ \text{safecat}\\ \text{syscat}\\ \text{transac}\\ \text{runfile}\\ \text{comfile}\\ \langle\text{filekey}\rangle\\ \langle\text{elementkey}\rangle\\ \langle\text{pointkey}\rangle\end{array}\right\}$$

6. Telestatus

Notice:

a. all parameter groups have the format:

```
<keyword>.<parameters separated by .'s>
```

b. if transac.yes is given a transaction text line must follow the fp-call.

6.3 Description of the parameters

This section summarizes the information held in the status file and describes how the information is referred to by the parameters.

The filename specifies for telestatus the name of the actual status file. If no filename command is given the default name 'statusfile' is used.

Any other group directly or indirectly describes an action to be performed on a particular field or on a collection of fields in the status file.

The status file consist of 3 records (segments) where:

record 1 is called the safety catalog (safecat)
record 2 is called the system catalog (syscat).

Record 0 is not used at present.

6.3.1 Survey of information in the status file

The status file holds information about a number of constituents (files, states and modes) of the running teledata system. The abbreviations shown in the present survey are the keywords used in the fp-parameters.

The use of the information is not considered here. The user is referred to the manuals for Teleop, Teleadm etc.

6. Telestatus

teledata notion	keyword
system files.	
secondary input	secinp
reprocessing log	replog
reestablishing log disc	disclog
reestablishing log tape	tapelog
central output	cenout
on-line state.	
current transaction no.	transcur
current transaction text	transac
last (finished) transaction	transfin
number of records in reest.log	logrecs
current terminal	termcur
system modes and state.	
program function (teleop)	func
safety mode (degree of safety in on-line system)	safemode
log mode (degree of logging)	logmode
state of system	state
result from execution of command file	comres
run file	runfile
command file	comfile
restart points (checkpoint).	checkpoint
restart point disclog	discpoint
restart point tapelog	tapepoint

6.3.2 Safety catalog and system catalog

The information in the status file is split up into a safety catalog and a system catalog. The safety catalog is currently updated by the on-line control system. The system catalog is of a more static nature and mainly updated off-line.

The present surveys shows the information actually available about the entities listed in the previous section.

Segment- halfword- file- and block numbers are 'last used'.

Contents of safety catalog	data type
Segment# and halfword# for secondary input-, reprocessing log-, disclog- and central output file	integers
File# and block# for tapelog file	integers
Current transaction# (counted)	integer
Last transaction#	integer
Number of logged records	integer
Current terminal#	integer
Current transaction text	characters (max 150)

6. Telestatus

Contents of system catalog	data type
Program function *)	integer
Safety mode *)	integer
Logmode *)	integer
Date	integer
State *)	integer
Result from command file exec. *)	integer
Restart point for disc log segment# and halfword#	integers
Restart point for tape log file# and block#	integers
Name	characters (max 11)
of secondary input file	
reprocessing log file	
disc log file	
tape log	
central output file	
run file	
command file	

*) the values are outlined in what follows.

Program function:

1:	teledata
2 and 3:	not used for the moment

6. Telestatus

Safety mode:

1: the status file is updated when a transaction is finished, (reprocessing required after system breakdown).

2: data transports started to status -, disc log - and tape log file are completed when a transaction is finished, (reestablishing possible).

3: the status file is updated when the processing of a transaction is initiated. Data transports started to the database are completed when a transaction is finished, (immediate (warm) restart may be possible)

Notice that safemode 1 is included in 2 and 3, and safemode 2 is included in 3.

Log mode:

0:	no log (Teleop must be started with log.no)
1:	tape log only
2:	disc log only
3:	tape- and disc log

State:

0:	after close
1:	after open

Comres:

0: error after teleadm
1: OK
2: error after command file

(see Teleadm manual)

6.4 Function of telestatus

The functions of the program are:

1. Setting of individual fields in the status file to specific values.

2. Creation of checkpoint

3. Reading a complete transaction text.

4. Clearing a field or groups of fields.

5. Printing the contents of a particular field or group of fields.

The present section outlines by the aid of a number of examples how these functions are used.

6.4.0 General

All parameters in the call of telestatus are read and checked before any updating of the file is performed.

6. Telestatus

The parameters are interpreted ('executed') from left to right, i.e. in the order given in the call. The only exception is the filename command which is not executable but simply specifies for telestatus the name of the actual status file. This command may be given anywhere in the call, but if more than 1 is given the last (rightmost) one will be valid for the entire call.

6.4.1 Setting (updating)

Ex. 1.1, file group:

```
disclog.logfile1
disclog.100.1
disclog.logfile1.100.1
```

The first command causes insertion of the name 'logfile1' into the name entry for disc log file in the system catalog. In the second case segment# and halfword# for disc log file in the safety catalog are set to 100 and 1 respectively. The third case combines the actions of the former two.

Ex. 1.2, element group:

```
safemode.2
```

The safety mode entry in the system catalog in set to 2 (see also section 6.3.2).

Ex. 1.3, point group:

```
tapepoint.7.23
```

In the system catalog the file# and block# for the restart point on the tape log are set to 7 and 23 respectively. See also checkpoint below.

Ex. 1.4, date:

```
date.780203
```

For further examples see section 6.8.

6.4.2 Checkpoint

```
checkpoint.yes
```

A checkpoint is created, i.e the segment# and halfword# for the disc log file and the file# and block# for the tape log file are moved from the safety catalog to the discpoint and tapepoint entries respectively in the system catalog.

The command checkpoint.no has no effect. Default is checkpoint.no.

6.4.3 Transaction

```
transac.yes
```

This command causes telestatus to read the line, following the last line of FP-parameters in the call of telestatus in the current input, into the transaction entry of the safety catalog. (max. 150 chars. including LF). Default is transac.no. Notice that the last line of FP-parameters must not be terminated with a ','.

6.4.4 Clear

The clear command is used to initialize fields, regardless of type, to binary zero.

Ex. 4.1   `clear.all`

The complete file is cleared.

Ex. 4.2   `clear.safecat.checkpoint`

The safety catalog and the 2 restart points in the system catalog are cleared.

Ex. 4.3   `clear.replog`

Segment# and halfword# and name for the reprocessing log are cleared. If only the safety catalog part is to be touched then use the command:

```
replog.0.0
```

Ex. 4.4   `clear.transcur.transfin.logrecs`

Equivalent to transcur.0 transfin.0 logrecs.0

6.4.5 Print

The print command having the same parameters as the clear command is used to print out the contents of individual fields or groups of fields. Numerical fields are interpreted as integers, character fields as text strings.

6. Telestatus

The result of a print command is one or several lines of output (on current out) for which the following format is typical:

```
,telestatus print, <explanatory text>
   <keyword>.<field value(s)>
```

Ex. 5.1   print.disclog.tapelog

Will cause the following output (names and numbers are arbitrarily chosen).

```
,telestatus print, disclog: name.segmentno.halfwordno
   disclog.discfile1.123.400

, telestatus print, tapelog: name.filno.blockno
   tapelog.mt0037.9.12
```

Notice that the format of keywords and actual values in the second print line is exactly fp-parameters for the fp-parameters for setting the same fields.

If the print parameters 'all', 'safecat' or 'syscat' are used then the heading 'telestatus print' is only printed once.

Ex. 5.2   print.safemode safemode.2 print.safemode

Both the old and the new value of safemode will be printed.

6. Telestatus

Cleared fields

Ex. 5.3 clear.replog print.replog

The print output will be:

```
, telestatus print, reprocessing log: name.segmentno.halfwordno
    replog..0.0
```

The empty string between the two points indicates that the name is cleared.

6.5 Requirements

size:	minimum:	halfwords
	optimal:	halfwords

6.6 Messages

If safety catalog or system catalog is modified (cleared or set) telestatus writes on current out:

```
telestatus safety catalog updated
```

```
telestatus system catalog updated
```

6.8 Further examples

The present example shows a complete updating of a status file. Notice that names and numbers are arbitrarily chosen.

6. Telestatus

```
telestatus     filename.status    clear.all      ,
               secinp.secfile1.10.102            ,
               replog.repfile1.111.102           ,
               disclog.discfile1.12.104          ,
               tapelog.tapefile1.0.48            ,
               cenout.cenfile1.13.104            ,
               ,
               transcur.64                       ,
               transfin.64                       ,
               logrecs.280                       ,
               termcur.7                         ,
               ,
               func.1                            ,
               logmode.3                         ,
               safemode.2                        ,
               date.780206                       ,
               tapepoint.0.0                     ,
               discpoint.0.0                     ,
               state.1                           ,
               comres.1                          ,
               runfile.run1                      ,
               comfile.command1                  ,
               ,
               transac.yes
this is a transaction
```

6.9 Error messages

According to the format of parameter groups outlined in section 6.2 (<keyword>.<parameterkey(s)>) two types of errors may be detected during parameter check:

6. Telestatus

```
***telestatus, illegal keyword: <keyword>
```

```
***telestatus, unknown parameterkey: <parameterkey>
```

```
***telestatus, : statusfile does
                 not exist: <filename>
```

The specified file name can not be looked up in the catalog.

8. Telemove

The program performs moving of files on disc (backing storage). Any kind of files may be moved.

8.1 Examples

Ex.1 telemove file1.to.file2

The contents of a file named 'file1' is moved to a file named 'file2'.

Ex.2 telemove in.movespec

Say 'movespec' is a text file containing the text 'file1.to.file2' the call will have the same effect as the call in ex.1.

8.2 Call

The program is called by the fp-command

$$\text{telemove}\left\{\begin{array}{l}\langle\text{from file}\rangle\text{.to.}\langle\text{to file}\rangle\\ \langle\text{from file}\rangle\text{.}\langle\text{to file}\rangle\\ \text{in.}\langle\text{spec file}\rangle\end{array}\right\}_{1}^{*}\quad\left\{\text{checksum.}\left\{\begin{array}{l}\text{yes}\\ \text{no}\end{array}\right\}\right\}_{0}^{*}$$

8.3 Description of the parameters

<from file> and <to file> must exist, i.e. be names of catalog entries describing disc files.

<spec file> must be the name of a text file containing a parameter list as specified above. Note that this parameter list must not contain a new in.<spec file> command.

If checksum.yes is specified a checksum generated during the copying is inserted in the tail of the catalog entry of the <to file>.

Notice that the checksum command is valid for all file pairs in the parameter list regardless of the actual sequence of parameters. If more than one checksum command is given the last (rightmost) one will be valid. Default is checksum.no.

8.4 Function

Telemove moves the contents of files on discs regardless of the actual contents of the files.

Names of the files to be moved are given as fp-parameters in the call or in a text file referenced.

The tail of the catalog entry of <from file> is moved to catalog entry tail of <to file>. If file sizes are different the size of <to file> is set equal to the size of <from file>.

A checksum generated during the copying is optionally inserted in the catalog entry of the <to file>. The checksum is a long generated by simple 48-bit addition of moved double words. Word 8 of the entry tail receives the leftmost 24 bit, word 10 the rightmost 24 bit. Warning: the checksum must only be inserted for cf files.

Errors detected during the execution of telemove does not cause a termination of the run. An error message will be output and processing will continue with the next file pair.

8.5 Requirements

size:	minimum:	halfwords
	optimal:	"
mess.buffers:	min. 2	
area proc.s.:	min. 4	

Notice that telemove utilizes as buffer the core allocated to the process but not occupied by code. So, in order to make the turnaround time as short as possible, it is essential that the program is run in a 'size' as large as possible.

8.6 Messages

For every file succesfully moved telemove writes on current out a line with this format:

```
<from file> to <to file> segments = <integer> {checksum = <long>} 1
                                                                   0
```

where:

<integer>	indicates the number of segments moved
<long>	is the checksum inserted (only if checksum.yes specified in the call).

Ex. 6.1

```
discfile to savefile  segments = 210
```

8.9 Error messages

After reading of the parameters errors may be detected during the actual copying of a file or at a final checks of the catalog entry of the written file. The latter type of error will occur very infrequently.

8. Telemove

Error messages generated while attempting to copy a file has the following format:

```
<from file> to <to file>  move not ok
*** <explanation of cause>
```

As indicated in the following survey the catalog entry of the <to file> may have been changed when the error is detected.

8. Telemove

Explanation of cause	catalog entry <to_file> changed	<to_file> changed
lookup $\left\{\begin{array}{l}\text{to-file}\\ \text{from\_file}\end{array}\right\}$	no	no
file cannot be looked up in catalog.		
create, reserve $\left\{\begin{array}{l}\text{from\_area}\\ \text{to\_area}\end{array}\right\}$	no	no
area process cannot be created or reserved. Allocate more areas to the job.		
too few message buffers	no	no
catalog entry of to_file not changed	no	no
probably because <from file> is larger than <to file> and the claims on permanent disc segments are exceeded.		
write message not send	yes	probably not
message buffer not available during copying.		
writing unsuccessfull	yes	?
output operation failed.		

8. Telemove

Explanation of cause	catalog entry <to_file> changed	<to_file> changed
no checksum inserted copying ok but catalog could not be changed for insertion of checksum.	yes	yes
segments moved <> segments in from_file number of moved segments was different from number expected.	yes	yes

When the actual copying between two files is reported ok a final check of the catalog entry of the <to_file> is performed. If inconsistencies are detected, which will be very exceptional, the following messages are used.

```
*** no lookup on to_file after move
```

```
*** to_size in cat. <> from_size  <segments in from_file>
```

```
*** checksum in cat. <> actual     <actual checksum>
```

9. Telelogex

Extraction of log

The program performs extractions from the LOG (re-establishinglog), generated by the program Teleop. The LOG is read, and relevant records are generated on several output files.

DUET

SODA

Telelogex is a CONTROL program for the DUET-SODA system (like duetframe), including facilities relevant in a log extraction. This means that all the processing is controlled by a DUET program, and all the input/output on record level is controlled by a SODA LD-description.

9. Telelogex

9.3 Explanation of the parameters

```
stdout.<name>
```

States the name of the file, on which the DUET/ SODA system normally writes error messages. The file must exist.
Default: stdout.duetlog

```
read.<name>
```

States the name of the file, used for reading character input by means of duet operations read and getline. If used, the file must exist.
Default: read.inputfile.

```
stdassign.<no>
```

Controls the assignment of norm value to a variable when a standard mark is read (duet operation read).
<no> = 0 means that the variable is not altered.
<no> <>0 means that the norm value is assigned to the variable.
Default: stdassign.1

$$\boxed{\texttt{print.<no>}\left\{\texttt{.<name>}\right\}_{\texttt{<no>}}^{\texttt{<no>}}}$$

Defines the number of print channels (for the duet operation print), and the corresponding names of disc files. <no> is the number of print

9.3 Explanation of the parameters

channels. They will be numbered 1, 2, ... <no>.
<no> must not exceed 5.

Exactly <no> number of different file names must be stated. The files must exist (if used).
Default: print.1.printout

descrip.<name>

States the name of the description file, containing the compiled SODA LD description.
Default: descrip.descripfile

ldsection.<no>

States the section no of the compiled SODA LD description, as identification of the LD description in question (among others in the description file).
Default: ldsection.0.

duetfile.<name>

States the name of the file containing the compiled DUET program.
Default: duetfile.duetfile

9. Telelogex

9.3 Explanation of the parameters

$\boxed{\texttt{user.<no>}}$

States the user number for this program call. Used by the DUET interpreter for dynamic checking of legal block references. User.0 means all users. Default: user.0.

$\boxed{\texttt{duetarea.<no>}}$

States the size (in number of words) of the core buffer used by the DUET interpreter for dynamic storage of compiled duet program blocks. Default: duetarea.3000.

9.4 Function

control program

Telelogex is a control program for the DUET/SODA system. It is the standard duetframe extended with 2 facilitites: Multi output files and moving of transaction text (see below). All the processing is controlled by a DUET program (and the corresponding SODA LD description). In order to show a solution to the log extraction, a model solution will be supplied, consisting of db-, ld-description and duet program. The model solution does not include a skeleton part, and is in all respect suitable for revisions. Section 9.4.2 gives a survey.

model

9. Telelogex

9.4 Function

i/o files

multi output

All files accessed must exist with the name, type (fx blocklength) as described in the DB description (and SODA LD). There is one exception to this: multi output files. The purpose of this facility is to generate output on user specific files by using the user number as index among the multi output files.

multiset

index 0

algol 1

One of the sets defined in SODA LD is selected as multiset (FP parameter). To this set there is connected a file, described in the DB description and referred to in the SODA LD. This file gets index 0. By means of algol operation 1 this file can be exchanged with another file, indicated by another index. This may be repeated any number of times (the file names - except index 0 - are stated as FP parameter).

program start

select file index

program end

1 segm/block

At program start, file index 0 is selected. When reading log records originating from a specific user, the file index <user no> can be selected, and all records created and put using the multiset will end up on the selected file. The selection can be done every time a new user number appears in the log records. At program end, telelogex will terminate the use of all files connected to multiset. All files on multiset must have blocklength 1 segm.

log file

The log file is in most regards treated as a normal input file. It is described in the DB description, using the "copyrecord" facility, as a great deal of the contents is a copy of records from the (rest of the) data base.

Teleop log extraction

The log is written by Teleop, <u>not</u> using this description. The description will only be used by the log extraction. Therefore only log records relevant to the log extraction may be described. The DB description of the log file is part of dbsource (Model Solution).

logset

rectype 7402

algol 2

In the SODA LD for log extraction, a record set is described on the log file. This gives access to all the relevant log records, except one: record type 7402, containing a text field of variable length. The text is the transaction text keyed in (supplied with terminal ident). As the tools do not support this kind of fields, it can be moved to a duet variable by means of the algol operation 2. The access to rectype 7402, as well as moving of all the other fields of the record type, are normal SODA facilities.

9. Telelogex

9.4.1 Description of algol operations

1. Select multiindex file

call

```
algol 1 (index, result)
```

parameters

index	(word, long or constant). Call parameter. Points out the file to be selected.
result	(word). Return parameter. 0 = OK. 1 = index not defined in FP parameters (but the index is selected anyway: file name is 'dummy'). 2 = index outside range (0 - 200) (no selection is performed)

function

Terminates use of current (last) file, and stores the description of this. Makes the new file available for normal processing by SODA, setting up the relevant description.

2. Move transaction text

call

```
algol 2 (text, size, chars, missing)
```

parameters

text	(text). Return parameter. The text is moved to this variable.
size	(word, long or constant). Call parameter. Number of characters in the declaration of the variable text. The number must be a number like 5, 11, 17, 23, -----.
chars	(word). Return parameter. Number of characters moved to text
missing	(word). Return parameter. Number of characters, that could not be moved to text (no room in text).

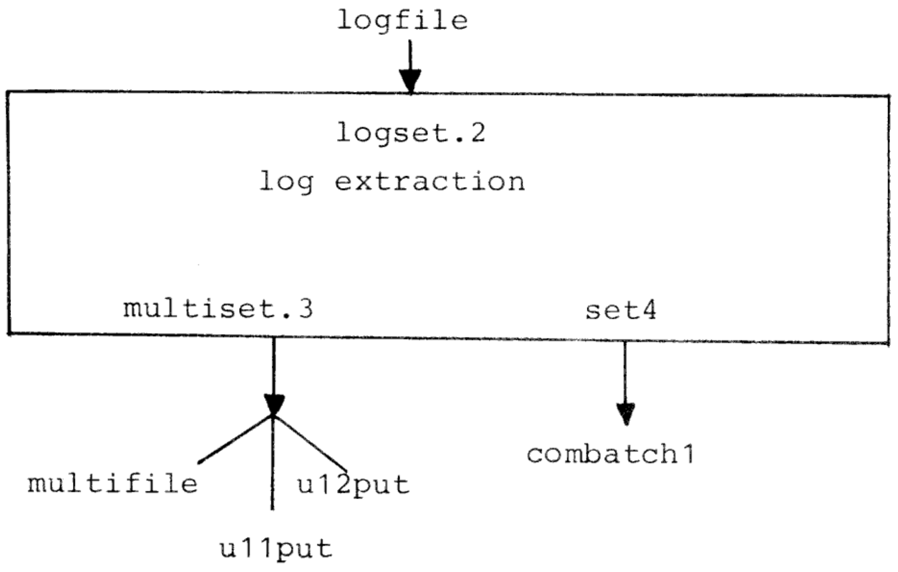
9. Telelogex

9.4.1 Description of algol operations

function

If the current log record is not rectype 7402, chars and missing will be -1, and text is not touched. Otherwise the text in the log record is moved to the variable text. The rest of text is reset (NULL-characters). Chars and missing are assigned. If the text field terminates with a new line (iso: 10), this character is not moved to text.

9.4.2 Survey of log extraction model



The figure shows input and output files in a log extraction run. Below are comments on all the files, plus the source texts for the model.

9.4.2 Survey of log extraction model

logfile	Described as part of the Model Solution DB description (dbsource). Declared as set 2 in SODA LD (reference to 'descripfile' of the Model Solution). Declared as 'record input set'. The SODA LD compiler then generates variables of field specifications for all fields in all the relevant record types, thus ensuring that all fields of the log records are moved to variables.
multifile	Described as part of the post DB description (lxdbsource). Declared as set 3 in SODA LD (reference to 'postdescr' of the post processing). The file is not used, as all output on this set is directed to u11put and u12put.
u11put u12put	These files are declared in the call of Telelogex (index.11.u11put index.12.u12put). The selection of the files is done in the duet program (algol 1), when log records from user 11 and 12, respectively, are met (only these users are included in the current data base of the Model Solution).
combatch1	Described as part of the post DB description (lxdbsource). Declared as set 3 in SODA LD (reference to 'postdescr' has been done). The file contains records to be 'sent' to a batch system.
lxldsource	Contains the SODA LD description for the log extraction model. It utilizes two facilities of the SODA LD: Mixing of more DB descriptions in on LD, and record input set declaration.

9.4.2 Survey of log extraction model

lxdusource — Contains the duet program (source text) for the log extraction model. Besides the above mentioned output generation, it might produce a survey of the log records read (log head information mostly). This is activated by means of a parameter read from the 'read' file (FP parameter).

9.5 Requirements

Depends very much on the volume of SODA LD and DUET program. The following estimates are for the log extraction model:

size:	60.000
buf:	10
area:	10
perm/temp disc:	depends on log size
time:	approx. (cpu) 100 log record per second.

9.6 Messages from program

Every program call generates the standard log of the DUET system. Telelogex writes (example from the model):

```
Telelogex: no of records on log: 66 eof(segm.word):22.236

Telelogex: start of multi output:
  index  file         segm  records    eof(block.byte)
      0  multifile       1        0          0.0
     11  u11put          1        0          0.0
     12  u12put          1        0          0.0

Telelogex: end of multi output:
  index  file         segm  records    eof(block.byte)
      0  multifile       1        0          0.0
     11  u11put          1        8          0.276
     12  u12put         21       17          1.350
```

9. Telelogex

9.8 Example

In the following example the following files are present:

logfile	contains the log
postdescr	contains compiled LD descr.
lxduetfile	contains compiled DUET progr.

```
lookup  logfile
param = copy 1
1
multifile = set 1 disc 0 0 0 20.1
u11put    = set 1 disc 0 0 0 20.1
u12put    = set 1 disc 0 0 0 20.1
logprint  = copy 0
duetlog   = copy 0

telelogex logset.2 multiset.3,
  index.11.u11put index.12.u12put,
  descrip.postdescr ldsection.110,
  duetfile.lxduetfile,
  read.param,
  print.1.logprint

c = copy logprint
c = copy duetlog
```

9.9 Error messages

```
Duet program error 28, 29 or 30
```

Will be activated from the algol operations, if some errors in the parameters are detected (wrong type ect).

```
xxxfile: <name> does not exist
```

```
xxxfile: <name> not 1 segm per block
```

These errors will only occur on multi output files defined in FP parameters.

```
xxx.no resources for: <name> <res>
```

Ressource troubles in algol 1 (select multi index file), when the file mentioned should be selected. <res> is a two digit number (XY), where X is result of 'create area process', and Y is result of 'reserve process'. Too few area's (in job line) will give X<>0. If some other process (job) is using the file, Y<>0 will occur.

10. Teleclock

The program calculates the Shortclock value from the machine clock, or it loads the Shortclock from a file entry, and it stores the Shortclock in one or more file entries; it also compares the Shortclock in several file entries, or it stores the Shortclock in the status file.

10.1 Examples

10.1.1 Ex. 1 `teleclock func.set clock.e files.a.b`

The Shortclock from the tail (6) of the entry of the file e is stored in tail (6) of the files a and b. On current output is printed:

```
teleclock, a: 19780104.0900
teleclock, b: 19780104.0900
```

10.1.2 Ex. 2 `teleclock func.compare files.in.d`

The Shortclock values from tail (6) of the files specified in d (each name begins with the period: .e.f.g) are compared and printed on current output:

```
teleclock, e: 19780104.1035
teleclock, f: 19780104.0900
teleclock, g: 19780104.1035
teleclock, values not equal.
```

10.1.3 Ex. 3 `teleclock fun.setstat clock.present files.h`

The present Shortclock value is calculated from the machine clock and stored in the status file h, field date. The program prints on current output:

```
teleclock, h: 19780104.1235
```

10. Teleclock

10.2 Call

The program is called by the fp-command:

$$\text{teleclock}\left[\begin{array}{l} \text{func. <function>} \\ \text{clock.}\left\{\begin{array}{l} \text{present} \\ \text{<filename>} \end{array}\right\} \\ \text{files}\left\{\begin{array}{l} \left\{\text{.<filename>}\right\}_{1}^{*} \\ \text{.in.<filename>} \end{array}\right\} \end{array}\right]_{1}^{*}$$

$$\text{<function>:=}\left\{\begin{array}{l} \text{set} \\ \text{compare} \\ \text{setstat} \end{array}\right\}$$

10.3 Explanation of the parameters

The present clock is the machine clock at the run time converted to the Shortclock representation.

It should be noticed that if this term is used:

files.in.<filename>

the file mentioned must contain one or more file-names each begining with the period, ex.:

.e.f.g

For the function "compare" the clock parameter group should not be specified.

Default-values are: func.<empty>
files.<empty>
clock.present

10. Teleclock

10.4 Function

10.4.1 func.set

The Shortclock is stored in word# 6 of the tail of the entry of the file(-s) specified.

If the present clock is wanted, the Shortclock value is calculated from the machine clock, otherwise it is taken from tail (6) of the entry of the clock file.

If one "set" is unsuccesful, for example because the file does not exist, the program proceeds with the next set after the message:

```
  ***teleclock, lookup-entry, <filename>, <result>
```

For each file is printed a line on current output:

```
  teleclock, <filename>:<shortclock>
```

10.4.2 func.compare

The Shortclock in word# 6 of the tail of the entry of the files specified are all compared and printed on current output:

```
  teleclock, <filename>:<shortclock>
```

terminated by (if they are all equal):

```
  teleclock, values are equal.
```

otherwise by

```
  teleclock, values not equal.
```

If one "compare" is unsuccesful, for example because the file does not exist, the program proceeds with the next compare after the message:

```
  ***teleclock, lookup-entry, <filename>, <result>
```

The clock is not used, so it need not to be written in the call.

10.4.3 func.setstat

The Shortclock is stored in the field date of the status file specified. On current output is printed:

```
  teleclock <name of statusfile><shortclock>
```

If the present value of the clock is wanted, the Shortclock is calculated from the actual machine clock, otherwise it is taken from tail (6) of the entry of the file specified.

If more than one file is specified in the files' parameter group, the first one is assumed to be the status file.

10.5 Requirements

size:	minimum:		halfwords
	optimal:		-
entries:		0	

10.6 Messages

```
teleclock, <filename>:<shortclock>
```

is written by teleclock after each "set", "compare" and "setstat".

```
teleclock, values are equal.
```

All Shortclock values are equal after a "compare".

```
teleclock, values not equal.
```

The Shortclock values are not all equal after a "compare".

10. Teleclock

10.9 Error messages

```
***teleclock, change-entry, filename : <filename>, result=<result>
```

A result <>0 is received from change-entry.

```
***teleclock, lookup-entry, filename : <filename>, result=<result>
```

A result <>0 is received from lookup-entry.

```
***teleclock, param: files.
```

No files specified.

```
***teleclock, param: function.
```

Function illegal (i.e. not set, compare or setstatus) or missing.

```
***teleclock, statusfile does not exist.
```

Statusfile does not exist.

12. Telereadcf

Telereadcf is intended for off-line checking of selected parts of a Teledata database.

The program checks that the files are readable by the database management system (the Connected Files System (cf)). In addition certain application dependent record counters may be checked.

The program itself is not dependent on a particular database configuration. The necessary information for a definition of the actual configuration is given as fp-parameters and extracted from the database description file (descripfile).

No updating is performed in the database.

12.1 Examples

Ex.1
```
telereadcf  master.6  chain.24.25
```

Master file 6 is read and all chain occurrences of chain 24 and 25 are traversed. File number and chain numbers corresponds to the database description (file_section and list_section respectively). The actual filenames and field addresses etc. are found in a database description file named 'descripfile', which is the default name of the binary database description.

Ex.2
```
telereadcf descrip.newdescrip                   ,
                     master.5 chain.17.18.43.nocount
```

In this case the database description is found in a file named 'newdescrip'. Master file 5 and chain types 17, 18 and 43 are read. For some reason say an error in the application program, the check of the software record counter for chain type 43 is suppressed.

An example of a check of a complete Teledata database is shown in section 12.8. Examples on output from the program are shown in section 12.6.

12. Telereadcf

12.2 Call

The program is called by the fp-command:

$$\text{telereadcf}\left\{\text{descrip.<filename>}\right\}_{0}^{*}$$

$$\left\{\text{master.<fileno>}\left\{\begin{array}{l}\text{chain}\left\{\text{.<chainno>}\left\{\text{.nocount}\right\}_{0}^{1}\right\}_{1}^{a}\\[2em]\left\{\text{chain.<chainno>}\left\{\text{.nocount}\right\}_{0}^{1}\right\}_{0}^{b}\end{array}\right\}_{0}^{c}\right\}_{1}^{m}$$

$$\left\{\text{maxerror.<unsigned integer>}\right\}_{0}^{*}$$

$$\left\{\text{<size parameters>}\right\}_{0}^{*}$$

where:

<filename>	is the name of the (binary) database description file. Default value is 'descripfile'.
<fileno>	is a master file number from the database description (file_section).
<chainno>	is a chain number from the database description (list_section).

12. Telereadcf

c = a + b	is the number of chains attached to the master file.
m	is the number of master files in the database. Notice that each master file is only to be specified once.
nocount	implies that the record counter belonging to the last specified chain is not checked.
maxerror.<i>	specifies the number of errors to be accepted for each master file. See section 12.4.2.

$$\text{<size parameters>} ::= \left\{ \begin{array}{l} \texttt{maxfiles.<unsigned integer>} \\ \texttt{maxtypes.<unsigned integer>} \\ \texttt{maxfields.<unsigned integer>} \\ \texttt{minchain.<unsigned integer>} \\ \texttt{maxchain.<unsigned integer>} \end{array} \right\}$$

The size parameters are used for dimensioning of the internal database defining tables. They all have default values probably covering most applications.

maxfiles.<i>	number of cf files in the database. Default value 11.
minchain<i> maxchain<j>	defines the range of chain numbers used in accordance with the database description list_section. Default 17 and 43 respectively.

12. Telereadcf

maxtypes.<i>	maximum number of record types per file. Default 20.
maxfields.<i>	maximum number of fields per record. Default 80.

If the database description is extended, maxfields and maxtypes may be candidates for increasing.
Too small values will cause the program to terminate with an index alarm.

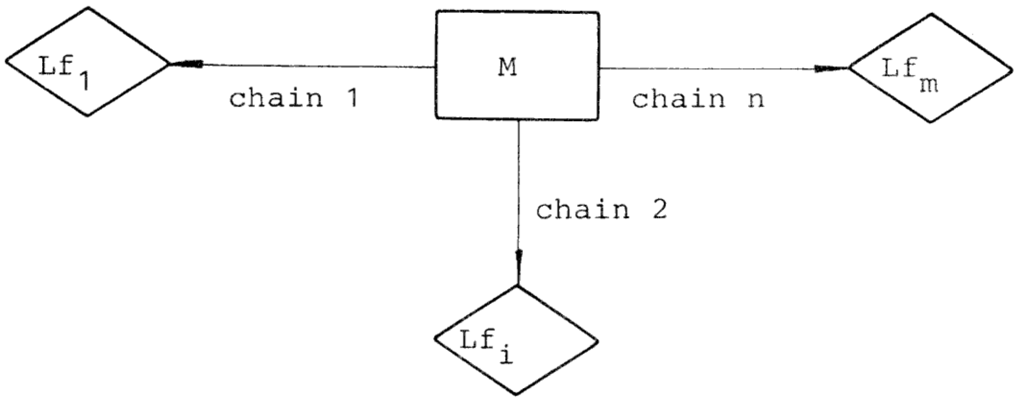
12.3 Telereadcf and the database

The purpose of this section is to clarify the relation between the parameters and the logical and physical database. Certain terms and conventions are introduced.

12.3.1 The database

The Telereadcf program considers the database to be a set of independent nodes, each consisting of a master file and its associated chains.

fig. 3.1 File level



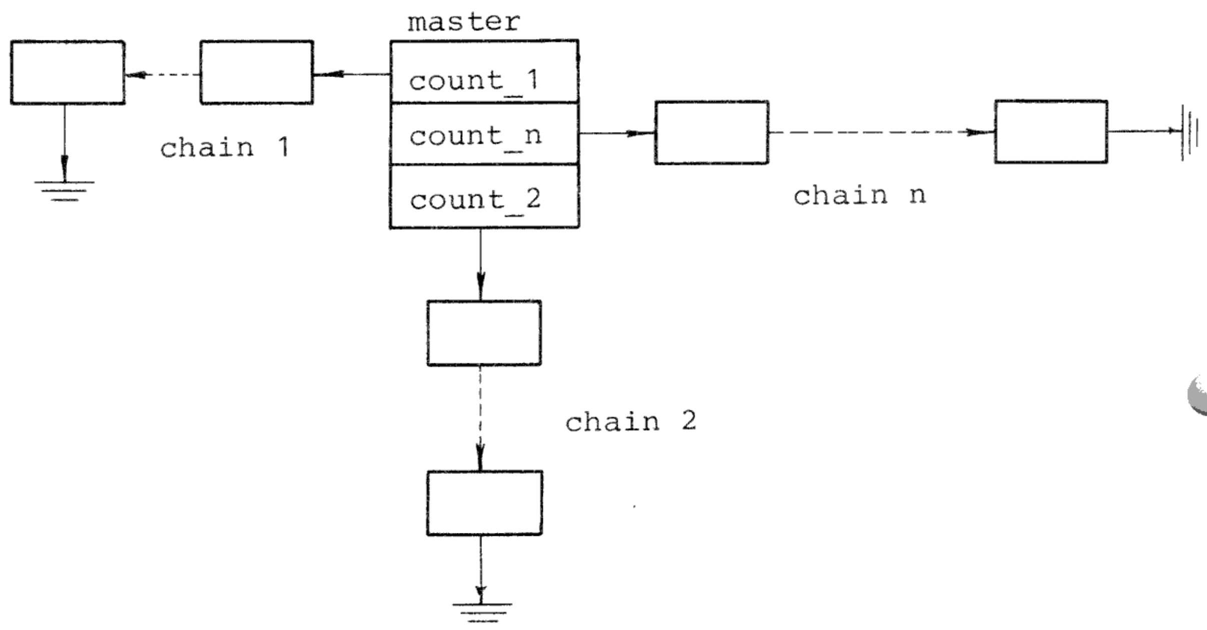
12. Telereadcf

In fig. 3.1 M denotes a master file consisting of a number of record occurrences. Chain 1 through chain n are the chain types attached to master M. Lf_1 through Lf_m denotes the list files holding the records in the chains.

File numbers and chain numbers are used instead of names because the numbers are considered unique, whereas the names may be changed.

For one master record occurrence the physical picture may look like fig. 3.2.

fig. 3.2 Record level



12. Telereadcf

Say the filenumber for M is 5, then the call:

```
telereadcf  master.5  chain.1.2
```

will cause the program to traverse master file 5; for each record occurrence cause the program to read chain 1 and 2 down to the nil point, count the list record occurrences in the chains and compare the result with the software counters count_1 and count_2 respectively. The last check may be suppressed by the nocount command.

12.3.2 Conventions

To make it possible for Telereadcf to look up the software counters the database description must obey the following rules.

The name of the count field must be of the form:

```
xx_count_<chainno>
```

where xx is some prefix and <chainno> is the chain number from the list_section of the description.
Example:

```
it_count_24.
```

The field must be of type half and file specific.

Notice that if no count field is declared the program automatically suppresses the counter check.

12.4 Function

The present section summarizes the functions of the program and outlines the error reactions and result.

12.4.1 Program function

The program checks the database, or selected parts of the database, as it is seen from the database management system.

The check, which is performed when the on-line system is closed down, should unveil possible damages in the database before the next start up of the on-line system. Thus reducing the probability for break down in the day hours.

Telereadcf reads the master files and the associated chains specified by the fp-parameters. In each chain occurrence the list record occurrences are counted. If a count field (software counter) is present in the master record the result is compared with this counter. No lookup on other master types connected with the list file is attempted.

For each master file the number of master records, the number of chain occurrences and record occurrences for each chain type are counted. The count results are displayed.

Section 12.6 shows an example of output from the program.

Notice that zones only are created and opened to files actually accessed in the run.

12.4.2 Error reaction

Two types of errors are detected during the reading of the files.

Softerrors denote inconsistencies between software counters and actual record occurrences in the chains.

Harderrors are caused by alarms from the cf-system, i.e unreadable files, destroyed chains etc.

In case of softerrors the results and the identification of the master record are displayed. The program continues to read the current master file. After 'maxerror' the counter check is suppressed.

In the case of harderror, identification of the current master record and if possible the current list record are displayed. The program does not terminate (break), but attempts to read the next master record. After 'maxerror', the reading of the current master file is terminated.

Notice that count results after harderror should not be considered valid.

Formats of error messages are shown in section 12.9.

12.4.3 Results

Any error detected by the program after the parameter reading will cause the ok-bit set false.

12. Telereadcf

12.5 Requirements

size:	Depends on number of files accessed. Should be large.
area:	Min. 2 + number of files accessed.

12.6 Output

For every master file successfully accessed Telereadcf will generate a survey as shown in fig. 6.1.

fig. 6.1 Output

```
start master file <m> time elapsed: cpu=<tcpu>s real=<treal>s
<filename1>
number of master records counted and expected=<record count 1>
   chainno    chainoccur.    recordoccur.      listfile
   <chainno>  <chain count>  <record count i>  <lf>  <filename i>
   .
   .
end master file <m> time elapsed: cpu=<tcpu>s real=<treal>s
```

where:

<m>	is the file number for the master file
<filename 1>	is the file name of the master file
<chainno>	is a chain number
<chain count>	is the total number of chain occurrences attached to master m from chain type <chainno>
<record count 1>	is the number of record occurrences in master <m>

12. Telereadcf

<record count i>	is the total number of record occurrences in the chain type <chainno>
<lf>	is the file number for the list file holding the chain type <chainno>
<filename i>	is the name of the list file <lf>
<tcpu>	is the cpu time elapsed since the start of the program
<treal>	is the real time elapsed since the start of the program

The chain- and list file information above is only displayed if chains are specified in the fp-parameters.

Finally if list files are accessed some information from the list file heads (maintained by the cf-system) is displayed as shown in fig. 6.2 below.

fig. 6.2 List file information

```
list file params.
 file   halfwords total   halfwords dead   fill limit
 <lf>      <used>              <dead>       <percentage>
  .
  .
```

where:

<lf>	is a list file number
<used>	is the number of halfwords used in file <lf>

12. Telereadcf

<dead>	is the number of inactive halfwords in file <lf>
<percentage>	is the max. fill percentage allowed in file <lf>.

For further interpretation of these figures the reader is referred to the manual of the cf-system (RCSL: 28-D5).
See also Telecleancf.

12. Telereadcf

12.8 Further examples

Below is shown the call of Telereadcf for check of a complete Teledata database. Assume the database description is found in 'descripfile'. Remember that the text after comma is a comment.

```
telereadcf,
           master.7                 , orders
                    chain.28        , od_odl
                    chain.29        , od_odr
           master.5                 , debcred
                    chain.19        , dc_ae
                    chain.17        , dc_dcs_part
                    chain.18        , dc_dcs_parent
                    chain.20        , dc_odl
                    chain.43        , dc_odr
           master.6                 , item
                    chain.24        , it_pl_parts
                    chain.25        , it_pl_used
                    chain.27        , it_odl
           master.8                 , tdadmin
                                    , userinf
           master.9
finis
```

or short:

```
telereadcf  master.7 chain.28.29 ,
            master.5 chain.19.17.18.20.43 ,
            master.6 chain.24.25.27 ,
            master.8  master.9
finis
```

12.9 Error messages

Telereadcf may generate error messages at 3 stages in one run.

1. During parameter reading.
2. During parameter check and generation of internal tables.
3. During actual reading of database.

In any case the first line of a message is:

```
*** telereadcf,    <explanatory text>
```

In the following survey only <explanatory text> is shown.

re 1. parameter text.

```
illegal keyword <keyword>
```

<keyword> is unknown or used in a wrong context.

re 2. parameter consistency.

```
illegal descripfile
```

The descripfile referred to is not a proper descripfile.

12. Telereadcf

```
master file specified twice <fileno>
```

Each master file must only occur once in each call.

```
file not found in descripfile <fileno>
```

File <fileno> is unknown.

```
use of file not in accordance with descripfile <fileno>
```

File <fileno> is not of the proper type.

```
chain not found in descripfile <fileno> <chainno>
```

Unknown chain number <chainno>.

```
use of chain not in accordance with descripfile <fileno> <chainno>
```

Chain <chainno> does not belong to master <fileno>.

12. Telereadcf

```
ident_field not found for register <logical fileno>
```

```
count_field not file specific <logical fileno> <chainno>
```

```
record type not found in descripfile <recordtype>
```

```
register type not found in descripfile <logical fileno>
```

Errors in database description.

```
stop caused by parameter inconsistency
```

One of the errors outlined above was detected in parameter check. The run is terminated.

re 3. database reading

```
master records, actual <> expected  <actual> <expected>
```

The number of records counted in the current master file does not agree with the value from the head of the file.

12. Telereadcf

```
list records , actual <> expected   <actual> <expected>
-------------------------------------------------------
the error was detected in master, chain  <fileno> <chainno>
-------------------------------------------------------
   master record identification.
   serno:   <word integer>
   itype:   <word integer>
   chdate:  <word integer>
   ident1:  <half integer>
   type:    <half integer>
   ident2:  <long or word integer>
```

Softerror (see section 12.3 and 12.4.2). The number of records counted <actual> in chain <chainno> does not agree with the value <expected> in the count-field (software counter) in master <fileno>.

The record identification is interpreted in accordance with the database description (version 3). Ident1 is userno.

```
 database error            <fileno>  <chainno>
-----------------------------------------------
   master record identification.
   .
   .
-----------------------------------------------
 list record identification.
 recno:    <word integer>
 rtype:    <word integer>
 chdate:   <word integer>
```

12. Telereadcf

Harderror (see also section 12.4.2). A cf-alarm occurred in master file <fileno> during attempt to read chain <chainno>. The master record identification is interpreted as for softerror above. The list record in accordance with the database description. Master- and list record identification are only displayed if there is a current record (zone record not empty).

```
stop after maxerror
```

Reading of the current master file is terminated after 'maxerror' harderrors detected.

```
count check suppressed after maxerror
```

'Maxerror' softerrors detected. Check and further messages suppressed. The reading of the master file is continued.

14. Telecleancf.

Telecleancf is intended for off-line cleaning of selected parts of a Teledata database.
Two types of cleaning are performed.

1. Records, erase-marked by the user, but well and alive in the database are deleted on demand, selected by user number.
2. List records deleted by the user, but still present in the database because the pointers are not completely updated by the system, are physically removed.

The program itself is not dependent on a particular database configuration.
The information nescessary for a definition of the actual configuration is given in form of fp-parameters and extracted from the database description file (descripfile).

The database is updated and since no logging is performed during the run a safety copy of the base must be created before the cleaning is started.

14.1 Examples.

Suppose a standard version 3 database is to be cleaned. The filenames and field addresses etc. are found in a database description file named 'descripfile', which is the default name of the binary database description.

14. Telecleancf.

Ex. 1

```
telecleancf master.7  user.1.clean.2        , orders
                chain.28.clean              , od_odl
                chain.29                    , od_odr
            master.5  user.3.4              , debcred
                chain.19.clean              , dc_ae
                chain.17.18.20.43
            master.6                        , items
                chain.24.25.27
```

Remember that the text following a comma is a comment.
The files are read and cleaned in the order specified in the call.
During the reading of master 7 and its associated chains 28 and 29,records belonging to user 1 and 2 are subject to special attention. For user 1 and 2 erase-marked records in chain 28 are deleted because of the clean command at chain level. User 1 also has the clean command at the master level. This implies that erase-marked master records for user 1 has to be deleted. This in turn implies (for user 1) that erase-marked records in chain 29 are deleted if the master record is erase-marked. The master records are deleted only if all the associated chains are empty. Being empty when the master was read or because all associated list records were erase-marked.

When master 5 is read deletions are only performed on erase-marked list records in chain 19 belonging to user 3 and 4. No master records are deleted.

14. Telecleancf.

During reading of master 6 with chain types 24, 25 and 27 no deletion is performed.

Since all master files and chains in the database are specified in the call cleaning type 2 is also accomplished. See also section 14.3.

14.2 Call.

The program is called by the fp-command:

$$\text{telecleancf} \left\{ \begin{array}{l} \text{descrip.<filename>} \\ \text{<master group>} \\ \text{maxerror.<unsigned integer>} \\ \text{<size parameters>} \end{array} \right\}_{1}^{*}$$

where:

<filename> is the name of the (binary) database description file. Default value is 'descripfile'.

<master group>::=master.<fileno>

$$\left\{ \text{user} \left\{ \left\{ \begin{array}{l} \text{.<userno>} \left\{ \text{.clean} \right\}_{0}^{1} \\ \text{. all} \left\{ \text{.clean} \right\}_{0}^{1} \end{array} \right\}_{1}^{*} \right\}_{1}^{1} \right\}_{0}^{1}$$

$$\left\{ \begin{array}{l} \text{chain} \left\{ \text{.<chainno>} \left\{ \text{.clean} \right\}_{0}^{1} \right\}_{1}^{a} \\ \left\{ \text{chain.<chainno>} \left\{ \text{.clean} \right\}_{0}^{1} \right\}_{0}^{b} \end{array} \right\}_{0}^{c}$$

14. Telecleancf.

<fileno>	is a master file number from the database description (file_section).
<userno>	is a Teledata user number. Only records belonging to users specified by <userno> (or all) are checked for an erase-mark.
clean	concerns cleaning of type 1 only. At the master level if implies that master records belonging to the last <userno>are deleted if erase-market. At the chain level the clean command implies that list records in the last chain in front of the command are deleted, if erase-marked and if the owner record (master) belongs to a user specified by <userno>
all	implies all users
<chainno>	is a chain number from the database description (list_section).
c=a+b	is the number of chains attached to the master file.
maxerror.<i>	defines the number of softerrors accepted. See section 14.3.

14. Telecleancf.

```
<size parameters>::= { maxfiles.<unsigned integer> }
                     { maxtypes.<      -"-      > }
                     { maxfields.<     -"-      > }
                     { minchain. <     -"-      > }
                     { maxchain. <     -"-      > }
```

The size parameters are used for dimensioning of the internal database defining tables. See Telereadcf description.

14.3 Telecleancf and the database.

The purpose of this section is to clarify the relation between the parameters and the logical and physical database. Certain terms and conventions are introduced.

14.3.1 The database

The Teleclean program considers the database to be a set of nodes each consisting of a master file and its associated chains. See fig. 3.1 below. Remember that chains (list record) only can be accessed via a master record.

14. Telecleancf.

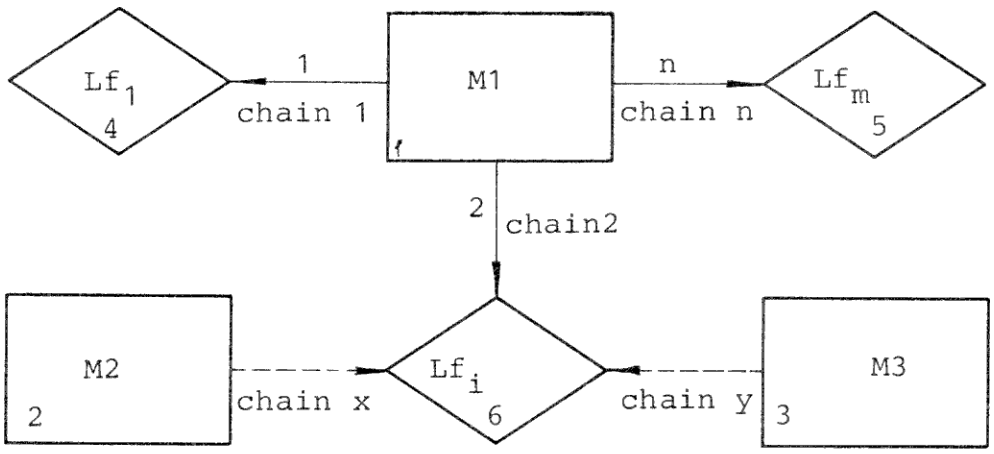


fig. 3.1 File level.

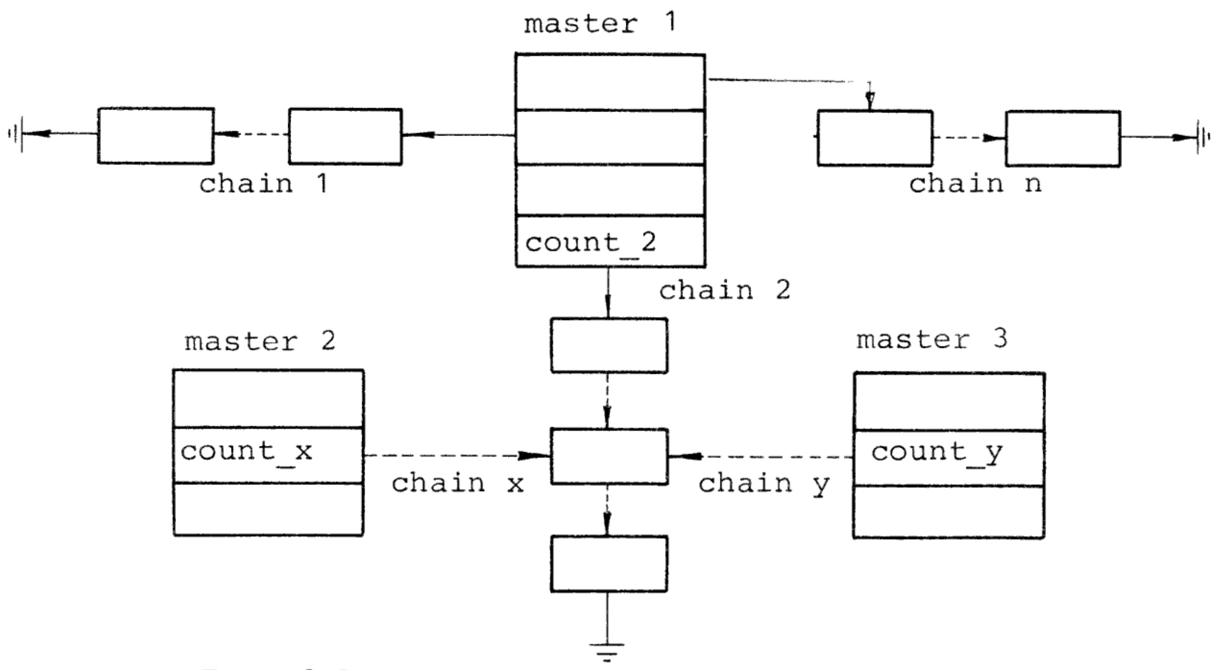
M1, M2 and M3 are master files each consisting of a number of record occurrences Chain 1 through chain n are the chain types attached to master M1. Lf_1 through Lf_m denotes the list files holding the records in the chains. Records in file Lf_i may also participate in chain types x and y belonging to M2 and M3 respectively.

In the call of the program, file numbers are used instead of names because the numbers are considered unique, whereas the names may be changed.

For one master record occurrence the physical picture may look like fig. 3.2 below.

14. Telecleancf.

fig. 3.2 Record level.



Erased list records.

Suppose M1 is the master file currently being read. Records in file Lf$_i$ are accessed via chain 2. If cleaning of type 1, deletion of erase-marked records, in chain 2 is to be performed then master files M2 and M3 must also be available because Telecleancf must update sofware record counters for chain x and y in the appropriate master records in master 2 and 3 respectively.
See also section 14.3.2 below.

Master files 2 and 3 are also required for the cleaning type 2 because the database management system does not use 'prior pointers'. This implies that a list record deleted in chain 2, though inaccessible for the user, is still physically present in the database as 'dead' record because chain x- and chain y pointers are not updated until the chains are traversed from master files 2 and 3. See also the manual of the cf-system (RCSL:28-05)

14. Telecleancf

Say the actual chain numbers for chains x and y are 20 and 21 respectively and the other file- and chain numbers are as indicated in fig. 3.1. Deletion of erase-marked records in chain 2 belonging to one user (say userno 11) and the cleaning type 2 in Lf_i as outlined above will be accomplished by the call.

```
telecleancf master.1 user.11        ,
                     chain.2.clean  ,
            master.2  chain.20      ,
            master.3  chain.21
```

Erased master records.

If a master record is to be deleted then all chains attached must be specified in the run. Say user 11 above also wants erase-marked records in master M1 deleted (fig.3.1) and the actual chain number for chain n is 22, then the call will be:

```
telecleancf master.1  user.11.clean      ,
                chain.2.clean.1.22       ,
            master.2  chain.20           ,
            master.3  chain.21
```

14.3.2 Conventions.

To make it possible for Telecleancf to look up fields in records, normally maintained by the application programs, the database description must obey certain rules.

14. Telecleancf

The fields concerned are protection-, count- and erase fields. If present they must be of type half and file specific. Notice that if a field is not declared in the database description of a file the program automatically suppresses the check of the field for that file.

Protection fields (master) indicates, if not zero, that some processing of the master record and/or its chains are not finished. No deletions are performed. The name of the protection field declared must be of the form:

x..x_protection

where x..x is some prefix ignored by Telecleancf.

Erase fields (master, list) indicates, if set to 1, that records are candidates for deletion by Telecleancf. The deletion is performed only if the call specifies the user, and a clean command is given for the particular master or chain. The name of the erase field declared must be of the form:

x..x_erased

where x..x is some prefix ignored by Telecleancf.

The application program must erase mark or delete all list records in chains attached to an erased master record. Otherwise the master record is not deleted by Telecleancf.

14. Telecleancf

Count fields (master) are the general record counters for chains as outlined for Telereadcf section 12.3.2, or counters for erase marked records. If the latter counter type is to be updated by Telecleancf when erase marked list records are deleted the field name declared must be of the form:

xx_killd_dd

where xx is some prefix ignored by Telecleancf and dd is the chain number.

The program checks that all files and chains required for the demanded clean action are specified in the call.

14.4 Function

The present section summarizes the functions of the program and outlines the error reactions and result.

14.4.1 Program function

The program reads the selected parts of the database as it is seen from the database management system. All records in the specified master files and chains are accessed.

Records erase-marked by the application program but for security reasons kept in the database for some time may be deleted for specified users. (cleaning type 1).

Records deleted by the application program but still present in the databsse (see section 14.3.1) are removed if the proper files and chains are specified, (cleaning type 2). If type 1 cleaning is performed in the same run of the program the order in which the master files are specified in the call are of importance because the files are read in that order. Generally the files and chains with clean commands should be specified first in order to achieve a total removal of 'dead records'. See example 1 in section 14.1. Recall that the amount of 'dead halfwords' is displayed when Telereadcf is run.

The number of master records and the number of list records in the chains are counted and checked as in Telereadcf.

If erase-marked list records are deleted by the program, then the software records counters (count fields) for all the chains, in which the records are members, are updated.

14.4.2 Error reaction

Two types of errors are detected during the processing of the files.

Softerrors denote inconsistencies between software counters and actual record occurences in the chains. Also attempts to clean out master records, erase-marked, but with non empty chains i.e. all records not erase-marked or deleted by the application program, are considered softerrors.

Harderrors are caused by alarms from the cf-system, i.e. unreadable files, destroyed chains etc.

In case of softerrors the identification of the current master record is displayed. The program continues to read the current master file, but after 'maxerror' count errors delected, the check of the count field is suppressed, and after 'maxerror' erase errors the message concerned is suppressed.

In case of harderror, identification of the current master record and if possible the current list record are displayed. The program does not terminate, but attempts to read the next master record in order to get the best possible error diagnosis. After 'maxerror' the reading of the current master file is terminated. After harderror the resulting database should not be used.
Formats of error messages are shown in section 12.9.

14.4.3 Results.

Errors detected by the program after the parameter reading will affect fp 'errorbits'. Harderrors or erase-mark errors will cause ok.no. Count error will cause warning.yes.

After harderrors the database should not be used.

14. Telecleancf

14.5 Requirements.

Size: depends on the number of files accessed. Should be large.

Area: min. 2 + number of files accessed.

14.6 Output.

For every master file and list file accessed the program will generate a survey of counted record- and chain occurrences as shown for Telereadcf section 12.6.

Further a survey of deletions performed for every file accessed in the run is displayed.

fig. 6.1 Deletions.

```
 file      recs.deleted
 <fileno>     <integer>
     .
     .
     .
```

14. Telecleancf

14.9 Error messages

Telecleancf may generate error messages at 3 stages in one run.

1) During parameter reading.
2) During parameter check and generation of internal tables.
3) During the actual processing of the database.

In any case the first line of an error message is:

```
*** telecleancf, <explanatory text>
```

In the following survey only the <explanatory text> part is shown.

re. 1 Parameter text.

```
illegal keyword   <keyword>
```

<keyword>is unknown or used in a wrong context.

re. 2 Parameter consistency

```
illegal descripfile
```

The descripfile referred to is not a proper descripfile.

```
file not found in descripfile <fileno>
```

File <fileno>is unknown.

14. Telecleancf

```
use of file not in accordance with descripfile<fileno>
```

File <filno> is not of the proper type.

```
chain not found in descripfile <fileno><chainno>
```

Unknown chain number <chainno>

```
use of chain not in accordance with descripfile<fileno><chainno>
```

Chain <chainno>does not belong to master <fileno>

```
ident field not found for register <logical fileno>
```

```
count field not file specific <logical fileno><chainno>
```

```
erase field not file specific <logical fileno>
```

```
record type not found in descripfile <recordtype>
```

```
register type not found in descripfile <logical fileno>
```

Errors in database description.

```
chains missing for file <fileno>
```

All chains required for processing of file <fileno>are not specified in the call.

```
stop caused by parameter inconsistency
```

One of the errors outlined above was detected in the paremeter check. The run is terminated.

14. Telecleancf

re. 3. Database reading

```
master records, actual<>expected  <actual><expected>
```

The number of records counted in the current master file does not agree with the value from the file head.

```
list records, actual<>expected  <actual><expected>
--------------------------------------------------------
the error was detected in master, chain <fileno><chainno>
--------------------------------------------------------

master record identification
serno:     <word integer>
type:           -"-
chdate:         -"-
ident :    <half integer>
type:           -"-
ident2:    <long or word integer>
```

The number of records counted <actual> in chain <chainno>does not agree with the value <expected>in the software counter in master <fileno> (softerror).
The record identification is interpreted in accordance with the database description (version 3). Ident1 is userno.

```
chain not empty    <integer>  <chainno>
----------------------------------------
master record identification
.
.
.
```

Deletion of erase-marked master record refused because chain <chainno>still contains <integer>records (softerror). The master identification is as for count error above.

14. Telecleancf

```
database error            <fileno> <chainno>
------------------------------------------------
 master record identification.
  .
  .
------------------------------------------------
list record identification.

recno:  <word integer>

rtype:          -"-

chdate:         -"-
```

A cf-alarm occurred in master <fileno> during attempt to read chain <chainno>(harderror). The master record identification is displayed as for softerrors above.
The list record identification is interpreted in accordance with the database description (version 3).

```
master not accessible from chain  <fileno> <chainno>
----------------------------------------------------
   <record identification>
```

A master record in file <fileno> could not be accessed from chain <chainno> for updating of record counter. <record identification> of current records is as for harderror above.

```
count check suppressed after maxerror
```

'maxerror' count errors detected. Check and messages suppressed.

14. Telecleancf

```
error message suppressed after maxerror
```

'maxerror' erase errors detected.

```
stop after maxerror
```

Processing of the current master file is terminated after 'maxerror' harderrors detected.

READER's COMMENTS

A/S Regnecentralen maintains a continuous effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback - your critical evaluation of this manual.

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

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

How can this manual be improved ?

Other comments ?

==

Please state your position: ______________________

Name:	______________	Organization:	______________
Address:	______________	Department:	______________
	______________		______________
	______________	Date:	______________

Thank you !

RETURN LETTER - CONTENTS AND LAYOUT

------------------------------------ Fold here ------------------------------------

==

Affix
postage
here

A/S REGNECENTRALEN
Marketing Department
Falkone Allé 1
2000 Copenhagen F
Denmark