RCSL No: 31-D635

Edition: February 1981

Author: Edith Rosenberg

Title:

Corrections to RCSL No 31-D601
Extensions to the RC8000
Indexed Sequential Files System (ISQ).

RCSL 42-11592

A/S REGNECENTRALEN
af 1979

Keywords:

RC8000, Backing Storage Package,
Indexed Sequential Files, corrections.

Abstract:

This paper describes the manual opdates due to changes in the software package and misprintings in the original manual (RCSL No 31-D601).

(24 printed pages)

Printed by A/S Regnecentralen af 1979, Copenhagen

RCSL 42-i1592

FOREWORD

First edition: RCSL No 31-D635
This set of corrections can be used as single-leaf cancels in RCSL No 31-D601: Extensions to the RC8000 Indexed Sequential Files System (ISQ).

The changes are indicated by correction lines in the left margin.

The new pages contain the description of the new procedure "removeupdi", and a changed description of "extendi" whereas the descriptions of "blocki", "priorreci", "putblocki", and "putdirecti" have been removed in order to bring the manual in agreement with the present software package.

Edith Rosenberg
A/S Regnecentralen af 1979, February 1981

CONTENTS	PAGE	CHANGED IN
		edition 1
	3	x
	4	-
	7	x
	8	cancelled
	9	cancelled
	10	x
	11	x
	12	x
	13	x
	14	x
	15	x
	16	x
	17	x
	18	-
	19	x
	20	x
	21	x
	22	-

2. EXTENSIONS AND MAJOR CHANGES 2.

2.1 Update Mark 2.1

The update mark is actually an integer stored in connection with the buckettable. Bits are used as flags to indicate some file states:

- the file is during intialization or recovering
- the file has entered an updating mode

The check and recover program (see section 3.7) cannot handle a file, which has not got through the initialization, i.e. passed a call of one of the mode procedures after the init-procedures, or a former recovering.

If a file has been in updating mode, it must return to reading before it is closed. If it does not, the case is signalled as a result from startfilei (ref. 1). The file may then be read, but if it enters updating, the system will cause a run time alarm, and the file must be reestablished by a backup, or by the check and recover program.

(Section 2.2 has been removed).

2.3 Recovery Program

2.3.1 Errors in isq-files

Inconsistencies in isq-files may origin from breaks between the writing of a record block and the corresponding table block, or between writings of two record blocks in a complicated insertion. Both cases will be signalled by the update mark. The check and recover program, recoveri, is able to recover the file without loss of records in the first case, which is the most probable. In the second one a number of halfwords (approx. an average record) may be lost or will exist in duplicate in the broken file, and recoveri cannot compensate the loss, while it will select a winner among the duplicates of records.

It is hard to predict, which other types of errors that may be reflected in a file, as some could occur when it is not reserved by the isq-system and are thus not signalled by the update mark. It is recommended to check the file regularly by recoveri.

2.3.2 Program Functions

recoveri is designed to check that a file is correct and to repair an incorrect file, so that usage can be resumed as soon as possible.

It consists of two parts, which may be executed separately or as a whole. Fig. 1 shows the two parts and the files they have in use.

In the recover part the records of 'block file' are interpreted sequentially and 'isq file' repaired blockwise. The tables are adjusted and record blocks cleared, and the 'isq file' is ready for isq-processing. Then the 'insert file' is read and the records inserted in 'isq file' by the isq-procedure insertreci. Duplicates of records are dismissed with a diagnostic at 'doc file'.

'recoveri' uses current output for run time alarm messages and warning messages, see app. B, while error diagnostics are written at 'doc file', which may be handled as a normal text file. An error diagnostic consists of an explanatory text and various fields, which identify the error. A survey is given in app. C.

2.3.3 Program Requirements

The core requirements for 'recoveri' may be as low as 23000 hw for files with minimal block lengths, but as the program reads 'isq file' with super-buffering and have internal sortings the processing time will decrease much with a greater core area. A sensible lower limit will be about 50000 hw for small files to which may be added the size of 'isq-file' to get the upper limit for profitable core utilization. If more than one error is found 'recoveri' may need some working area at the backing storage as mentioned in ref. 3.

Files which have been broken during initialization or recovering cannot be handled by recoveri.

(Section 2.4 has been removed).

2.5 New Service Procedures 2.5

Two new file handling procedures are introduced, headparamsi, which reads the file definition parameters from a file head, and extendi, which may extend a file with more buckets or cut unused buckets. See the procedure definitions in section 3.3 and 3.2 respectively.

headparamsi may be perceived as the reverse procedure of headfilei (ref.1), which creates an isq-file head from the user's parameters. headparamsi reads a file head and supplies the user with the original file head parameters.

Example 4, headparamsi:

```
comment create a file head for file b, which is equal
     to that of file a, except that it has an extra key
     field.
;
open(za, 4, <:a:>, 0);
open(zb, 4, <:b:>, 0);
headparamsi(za, recdescr, nkey, maxreclength, maxlength
          segsperbuck, segsperblock);
nkey:= nkey +1;
comment move the length definition:;
recdescr(nkey +1, 1):= recdescr(nkey, 1);
recdescr(nkey +1, 2):= recdescr(nkey, 2);
comment the extra field is of type integer and placed two
   hw after the previous field:
;
recdescr(nkey, 1):= 2;
recdescr(nkey, 2):= recdescr(nkey -1, 2) +2;
```

extendi may be used when the file is in any isq-mode except during initialization. It includes new buckets in the file or excludes unused buckets and segments in the end of the file by changing the catalog entry and the buckettable. The zone must be declared with some extra double words for addition of new bucket table entries. The size of an entry is given in ref. 1 section 1.3.

Example 5, extendi, include new buckets:

```
zone zi(buflengthi(<:i:>, true) +10, 3, stderror);
comment the zone is declared to hold five extra bucket
   table entries of the size 1 + keypartsize = 1 + 1 =
   2.
;
.
.
.
insertreci(zi, record);
case resulti of
begin
   ; <*insert ok*>
   result2;
   <*resulti = 3, too expensive*>
      goto res4;
   begin <* resulti = 4, file is full*>
   res4: extendi(zi, 1);
      if resulti <> 1 then
         system(9, resulti, <:<10> extend:>)
      else
      begin
         insertreci(zi, record);
         if resulti <> 1 then
            system(9, resulti, <:<10>2nd insr:>);
      end
    end;
    result 5;
    result 6;
  end;
```

Example 6, extendi, exclude unused segments:

```
comment the program reorganizes an isq-file by
sequential copying from one file to another.
;
initfilei(zo, 0.8, 0.8);
startfilei(zi);
while resulti = 1 do
begin
  initreci(zo, zi);
  if resulti <> 1 then
  begin
     .
     .
     .
     .
  end;
  nextreci(zi);
end;
extendi(zo, -1);
if resulti <> 1 then
  system(9, resulti, <:<10>no cut:>);
```

3. PROGRAMMING INTERFACES 3.

In this chapter the new entries to the isq-system are described in alphabetic order.

(Section 3.1 has been removed).

3.2 Procedure extendi 3.2

Call: extendi(z, segments)

z (call and return value, zone).
Specifies the file.

segments (call value, integer).
If segments > 0, the file will be extended with as many buckets as needed to include 'segments'.
If segments < 0, unused segments in the end of the file area are released.
If segments = 0, the size of the file is unchanged.

Function: The procedure changes the catalog entry of the file, so that it may either be extended or cut. If extension is wanted, the zone should be declared with some extra room for extension of the buck table. In all cases the shortclock in the catalog entry is updated.

Requirements: zonestate = readonlyi, readnexti, puti, or updatei.

Results: zonestate unchanged.

Available record: The first in the file, except when resulti = 5, in which case the available record is unchanged.

resulti:

1	Ok
2	Ok, but only room for simple insertion.
3	As 1, but update mark found.
4	As 2, but update mark found.
5	Illegal zonestate. The file, zonestate, and available record are unchanged.
6	The wanted extension exceeds maxbucks, the file is extended only to maxbucks * segsperbuck segments.
>10 000	Not done because of error at a call of a monitor function: resulti = monitor result * 10000 + monitor function no. Probable results: 40044 changeentry, protected 60044 changeentry, claims exceeded.

Alarms: The alarm "prep i 3" may occur from extendi if the zone is too small for the new entries in the bucket table. However, the file contains no update mark, so if the program is restarted buflengthi will compute the correct buffersize for the extended file.

3.3 Procedure headparamsi 3.3

Call: headparamsi(z, recdescr, nkey, maxreclength, maxbucks, segsperbuck, segsperblock)

The parameters are similar to those of headfilei (ref. 1), but they are all used for return values.

Function: Extracts from the head of an indexed sequential file connected to the zone z, the call values of the original call of headfilei. The zone should be able to hold at least nkey* 10 + 45 double words. Zonestate is 0 after the call.

Errors: The run may be terminated with an alarm, if the parameters z and recdescr cannot hold the return values or if one of the following rare causes coincides:

head i <i>
relative position in the filehead exceeds limits. <i> displays the position.

comp ins <i>
the compare code in the filehead is erroneous. <i> displays the value for an instruction.

gets ins <i>
the getsize code in the filehead is erroneous. <i> displays the value for an instruction.

The three error causes above will normally indicate that the filehead has been violated and it will not be possible to initialize, start or recover the file. Otherwise the cause should be reported as a basic program error.

(Section 3.4 has been removed).

(Section 3.5 has been removed).

(Section 3.6 has been removed).

3.7 Program recoveri

Call: recoveri <isq file> <recover files>[1/0] <doc>

<runtype>[1/0] <dupspec>[1/0] <maxerror>[1/0]

<isq file> (area entry). The name of the isq-file.

<recover files>:: = <insert file> <block file>
(area entries). If a pure checking is wanted the two file names may be omitted. If errors a the two files are used for the recovering information.

<doc> (key word parameter)
doc.<doc file>
<doc file> is the name of an area to be used for error listing and run summary.

<runtype> (key word parameter)

run. { check / recover / all }

If 'check' is specified only the file checking will be run.
If 'recover' is specified only the file recovering will be run, which requires that a 'check'-run has filled <recover files>.
If 'all' is specified both checking and recovering will be run.
run.all is default.

<dupspec> (key word parameter)

dup.<duptype>.<dupaddr> { . asc / . des }$_{0}^{1}$

This parameter is the specification of the winner record in the case of duplicates in the file. See section 2.3.

<duptype>:: = { half / integer / long / real }

<dupaddr>:: = <fieldaddr>

asc means ascending order (default).
des means descending order.
dup.integer.<last +2> is default, <last +2> meaning the word after the last defined key field.

<maxerror> (key word parameter)

max.<unsigned integer>
if the number of erroneous blocks exceeds the specified value, the run stops after the checking.
Default is no stop.

Function: The program holds two phases, the checking and the recovering, which may be called separately. During the checking errors are reported at <doc file> and records for recovering stored at <recover files>. The recovering part will read <recover files> and repair <isq file>. The user may modify <recover files>, see the file formats in app. D.

Requirements: Any trouble caused by parameters will cause a run time alarm. A list is given in app. B.1.

Results: After the recovering <isq file> may be accessed the normal way. If the program breaks during the recovering phase it cannot be guaranteed that <isq file> may be accessed again by this program or the isq-procedures. (Therefore it is recommended to take a backup copy before a complex recovering, see app. F).

After run.all or run.check with errors in <isq file> the fp-bit warning is true. After run.recover the same bit means that some of the records of <insert file> cannot be inserted.

3.8 Integer Procedure removeupdi

Call: remove_upd_i (z)

remove_upd_i (return value, integer). Result of call:
0: The file contains no update mark.
1: The file contained an update mark which is now removed.

z (call and return value, zone). Specifies the file.

Function: Removes a possible update mark from the file and re-initializes the zone by calling startfilei.

Requirements: zonestate = read_only_i (i.e. after call of startfilei).

Results: zonestate: unchanged

result_i:	Available record.
1 ok	The first in the file.
2 ok, but only room for simple insertion	The first in the file.

Errors: The procedure may terminate the run with one of the following runtime alarms:

z.state <i>	Zone state is not read_only_i. The actual zone state is shown.
upd.mark <i>	The update mark in the file is originating in an unfinished initialization (<i> = 4) or recovery run (<i> = 2), and cannot be removed.

Warning! Do not use this procedure unless you are sure that the file is consistent, i.e. the contents of the bucket table, the block table, and the block have been rewritten into the file after an insertion or deletion of records.

A. REFERENCES

1. RCSL No 31-D 600, June 1980, Inge Borch
RC8000 Indexed Sequential Files.

2. RCSL No 31-D 602, June 1980, Inge Borch
RC8000 SQ-SYSTEM.

3. RCSL No 31-D 562, April 1979, Jørgen Winther
RC8000 Backing Storage Area Sorting.

4. RCSL No 42-i 1278
ALGOL8, User's Manual, Part 2.

5. RCSL No 31-D 477, January 1978, Tove Ann Aaris
RC8000 MONITOR, PART 2, Reference Manual.

B. MESSAGES FROM THE RECOVERY PROGRAM

B.1 Run Time Alarm Messages

In the case that a requirement of an isq-procedure is violated, a message is printed at current output and a run time alarm is invoked. A list of the messages is given in ref. 1.

The recover-program uses the algol i/o-System, the SQ-System, mdsortproc, as well as the isq-system itself, and therefore some of the messages listed in ref. 1, 2, 3, and 4 may appear as a run time alarm from this program, but often the alarm is supplied with one of the messages listed below:

freecore <i>	the program needs more 'core' to handle the isq-file. <i> is the number of half-words left for a single share and should be increased to hold as many segments as possible to decrease processing time.
initmark <i>	an initmark was found in the isq-file, meaning that an initialization run was not completed and the file cannot be recovered. <i> is irrelevant.
lookup <i>	lookup of the isq-file without success. See ref. 4 and 5, monitor function 42. <i> is the monitor result. (<i> = 3 means file does not exist).
progcall <i>	error in the program call at parameter no. <i>.
sortdisc <i>	too few backing storage claims for sorting of recover files. <i> <0 means lacking entries, <i> > 0 means lacking segments.

RETURN LETTER

Title: Corrections to RCSL No 31-D601 RCSL No: RCSL No 31-D635
Extensions to the RC8000 Indexed Sequential Files System (ISQ).

A/S Regnecentralen af 1979/RC Computer A/S maintains a continual 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?

Name: _____ Title: _____

Company: _____

Address: _____

Date: _____

Thank you

42-i 1288

..... Fold here .....

..... Do not tear - Fold here and staple .....

Affix postage here

REGNECENTRALEN
af 1979

Information Department
Lautrupbjerg 1
DK-2750 Ballerup
Denmark