

Sym
NT
NT
NT
NT
NT
NT

NT
NT
NT
NT
NT
NT
NT
NT
NT
NT
NT
NT
NT
NT
NT
NT
NT
NT
NT
NT
NT

NT
NT
NT
NT
NT
NT
PI

```
RRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSSSS
RRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSSSS
RRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSSSS
RRR      RRR      MMMMMM      MMMMMM      SSS
RRR      RRR      MMMMMM      MMMMMM      SSS
RRR      RRR      MMMMMM      MMMMMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSS
RRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSS
RRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSS
RRR      RRR      MMM      MMM      SSS
RRR      RRR      MMM      MMM      SSS
RRR      RRR      MMM      MMM      SSS
RRR      RRR      MMM      MMM      SSS
RRR      RRR      MMM      MMM      SSS
RRR      RRR      MMM      MMM      SSS
RRR      RRR      MMM      MMM      SSS
RRR      RRR      MMM      MMM      SSSSSSSSSSSSS
RRR      RRR      MMM      MMM      SSSSSSSSSSSSS
RRR      RRR      MMM      MMM      SSSSSSSSSSSSS
```

```

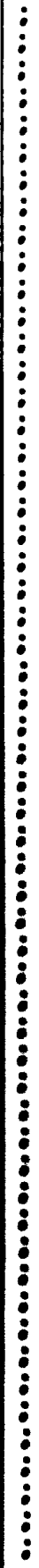
RRRRRRRR      MM      MM      333333      PPPPPPPP      000000      SSSSSSSS      KK      KK      EEEEEEEEEEE      YY      YY
RRRRRRRR      MM      MM      333333      PPPPPPPP      000000      SSSSSSSS      KK      KK      EEEEEEEEEEE      YY      YY
RR      RR      MMMM      MMMM      33      33      PP      PP      00      00      SS      KK      KK      EE      YY      YY
RR      RR      MMMM      MMMM      33      33      PP      PP      00      00      SS      KK      KK      EE      YY      YY
RR      RR      MM      MM      33      33      PP      PP      00      00      SS      KK      KK      EE      YY      YY
RRRRRRRR      MM      MM      33      PPPPPPPP      00      00      SSSSSS      KKKKKK      EEEEEEEEE      YY
RRRRRRRR      MM      MM      33      PPPPPPPP      00      00      SSSSSS      KKKKKK      EEEEEEEEE      YY
RR      RR      MM      MM      33      33      PP      PP      00      00      SS      KK      KK      EE      YY
RR      RR      MM      MM      33      33      PP      PP      00      00      SS      KK      KK      EE      YY
RR      RR      MM      MM      33      33      PP      PP      00      00      SS      KK      KK      EE      YY
RR      RR      MM      MM      33      33      PP      PP      00      00      SS      KK      KK      EE      YY
RR      RR      MM      MM      333333      PP      000000      SSSSSSSS      KK      KK      EEEEEEEEEEE      YY
RR      RR      MM      MM      333333      PP      000000      SSSSSSSS      KK      KK      EEEEEEEEEEE      YY

```

```

LL      IIIIII      SSSSSSSS
LL      IIIIII      SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLLLL      IIIIII      SSSSSSSS
LLLLLLLLLLLL      IIIIII      SSSSSSSS

```



```

1 0001 0 MODULE RM3PCSKEY (LANGUAGE (BLISS32) ,
2 0002 0 IDENT = 'V04-000'
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
10 0010 1 * ALL RIGHTS RESERVED. *
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
17 0017 1 * TRANSFERRED. *
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
21 0021 1 * CORPORATION. *
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1 **
30 0030 1
31 0031 1 FACILITY: RMS32 index sequential file organization
32 0032 1
33 0033 1 ABSTRACT:
34 0034 1 This module positions to a record by key value.
35 0035 1
36 0036 1
37 0037 1 ENVIRONMENT:
38 0038 1
39 0039 1 VAX/VMS Operating System
40 0040 1
41 0041 1 --
42 0042 1
43 0043 1
44 0044 1 AUTHOR: Todd M. Katz RE-CREATION DATE: 17-Jan-83
45 0045 1
46 0046 1
47 0047 1 MODIFIED BY:
48 0048 1
49 0049 1 V03-007 TSK0001 Tamar Krichevsky 15-Jun-1983
50 0050 1 Change addressing mode for RMSRU_RECLAIM to long relative.
51 0051 1
52 0052 1 V03-006 MCN0002 Maria del C. Nasr 22-Mar-1983
53 0053 1 More changes in the linkages
54 0054 1
55 0055 1 V03-005 MCN0001 Maria del C. Nasr 24-Feb-1983
56 0056 1 Reorganize linkages
57 0057 1

```

```
.. 58      0058 1  |      V03-004 TMK0003      Todd M. Katz      17-Jan-1983
.. 59      0059 1  |      Re-write the routines within this module adding support for
.. 60      0060 1  |      Recovery Unit Journaling and RU ROLLBACK Recovery of ISAM
.. 61      0061 1  |      files.
.. 62      0062 1  |      |
.. 63      0063 1  |      |*****
.. 64      0064 1  |      |
.. 65      0065 1  |      LIBRARY 'RMSLIB:RMS';
.. 66      0066 1  |      |
.. 67      0067 1  |      REQUIRE 'RMSSRC:RMSIDXDEF';
.. 68      0132 1  |      |
.. 69      0133 1  |      | Define default PSECTS for code.
.. 70      0134 1  |      |
.. 71      0135 1  |      PSECT
.. 72      0136 1  |      CODE = RMSRMS3(PSECT_ATTR),
.. 73      0137 1  |      PLIT = RMSRMS3(PSECT_ATTR);
.. 74      0138 1  |      |
.. 75      0139 1  |      | Linkages.
.. 76      0140 1  |      |
.. 77      0141 1  |      LINKAGE
.. 78      0142 1  |      L_RABREG_67,
.. 79      0143 1  |      L_PRESERVE1;
.. 80      0144 1  |      |
.. 81      0145 1  |      | External Routines.
.. 82      0146 1  |      |
.. 83      0147 1  |      EXTERNAL ROUTINE
.. 84      0148 1  |      RMSSEARCH_TREE      : RLSRABREG_67,
.. 85      0149 1  |      RMSGETNEXT_REC      : RLSRABREG_67,
.. 86      0150 1  |      RMSRLSBKT           : RLSPRESERVE1,
.. 87      0151 1  |      RMSRU_RECLAIM       : RLSRABREG_67 ADDRESSING_MODE( LONG_RELATIVE ),
.. 88      0152 1  |      RMSSEARCH_SIDR      : RLSRABREG_67;
```

```

90      0153 1 GLOBAL ROUTINE RMSPOS_KEY : RLSRABREG_67 =
91      0154 1
92      0155 1  ++
93      0156 1
94      0157 1  FUNCTIONAL DESCRIPTION:
95      0158 1
96      0159 1      This routine is responsible for positioning to the first non-deleted
97      0160 1      primary data record that matches the key in keybuffer 2 according to
98      0161 1      a well-defined set of search characteristics.
99      0162 1
100     0163 1      If RMS encounters a record that is marked RU_DELETE and the Recovery
101     0164 1      Unit in which the record was deleted is still active, then RMS returns
102     0165 1      this record as the non-deleted primary data record and lets a higher
103     0166 1      level routine decide what to do. If the Recovery Unit in which the
104     0167 1      record was deleted has successfully terminated, then RMS will continue
105     0168 1      its search for a non-deleted primary data record after deleting this
106     0169 1      RU_DELETED record (the latter if it has write access to the file).
107     0170 1
108     0171 1      If RMS encounters a record that is marked RU_UPDATE and is in a special
109     0172 1      format then RMS will return this record as the non-deleted primary data
110     0173 1      record after reformatting it. The reformatting is done if RMS has write
111     0174 1      access to the file, and the Recovery Unit in which it was updated has
112     0175 1      successfully terminated.
113     0176 1
114     0177 1  CALLING SEQUENCE:
115     0178 1
116     0179 1      RMSPOS_KEY()
117     0180 1
118     0181 1  INPUT PARAMETERS:
119     0182 1      NONE
120     0183 1
121     0184 1  IMPLICIT INPUTS:
122     0185 1
123     0186 1      IDX_DFN                - address of current index descriptor
124     0187 1      IDX$B_KEYREF             - key of reference
125     0188 1
126     0189 1      IFAR                - address of IFAB
127     0190 1      IFBSV_WRTACC           - if set, file is open for write access
128     0191 1
129     0192 1  OUTPUT PARAMETERS:
130     0193 1      NONE
131     0194 1
132     0195 1  IMPLICIT OUTPUTS:
133     0196 1
134     0197 1      IRAB
135     0198 1          IRBSL_CURBDB        - address of BDB for current primary data bucket
136     0199 1          IRBSW_RFA_ID         - ID of current record (primary only)
137     0200 1          IRBSL_RFA_VBN       - VBN of current record
138     0201 1          IRBSB_STOPLEVEL      - 0
139     0202 1
140     0203 1      REC_ADDR                - address of primary data record
141     0204 1
142     0205 1  ROUTINE VALUE:
143     0206 1
144     0207 1      RNF                - unable to position to a primary data record by key value.
145     0208 1      SUC                - successfully positioned to a primary data record by key value.
146     0209 1

```

: R

: S
: R
: F
: L
: L

```

147 0210 1 : SIDE EFFECTS:
148 0211 1
149 0212 1     On success, REC_ADDR points to the non-deleted primary data record
150 0213 1     and the BDB of the primary data bucket maybe found in IRB$$_CURBDB.
151 0214 1     On failures, all accessed buckets are released.
152 0215 1     If RU_DELETED records are encountered, they might have been deleted.
153 0216 1     If RU_UPDATED records are encountered, they might have been reformatted.
154 0217 1
155 0218 1 --
156 0219 1
157 0220 2 BEGIN
158 0221 2
159 0222 2 BUILTIN
160 0223 2 AP;
161 0224 2
162 0225 2 EXTERNAL REGISTER
163 0226 2 COMMON RAB STR,
164 0227 2 R_IDX_DFN STR,
165 0228 2 R_REC_ADDR_STR;
166 0229 2
167 0230 2 ! Initialize several variables, and then position to the (primary or
168 0231 2 ! secondary) data record by key value.
169 0232 2
170 0233 2 IRAB[IRB$$_STOPLEVEL] = 0;
171 0234 2 IRAB[IRB$$_CURBDB] = 0;
172 0235 2 IRAB[IRB$$_SAVE_POS] = 0;
173 0236 2
174 0237 2 RETURN_ON_ERROR (RMSSEARCH_TREE());
175 0238 2
176 0239 2 ! If RMS is to position by primary key, then position to the first
177 0240 2 ! non-deleted primary data record whose primary key matches the search key
178 0241 2 ! according to the characteristics of the search.
179 0242 2
180 0243 2 IF .IDX_DFN[IDX$$_KEYREF] EQLU 0
181 0244 2 THEN
182 0245 2 BEGIN
183 0246 2
184 0247 2 WHILE (.REC_ADDR[IRCSV_DELETED]
185 0248 2 OR
186 0249 2 .REC_ADDR[IRCSV_RU_DELETE])
187 0250 2 DO
188 0251 2 BEGIN
189 0252 2
190 0253 2 ! If RMS finds that the current record has been deleted within a
191 0254 2 ! Recovery Unit, then it subjects this record to further processing
192 0255 2 ! before deciding whether to return this record as the non-deleted
193 0256 2 ! primary data record, or to continue with the search.
194 0257 2
195 0258 2 IF .REC_ADDR[IRCSV_RU_DELETE]
196 0259 2 THEN
197 0260 2 BEGIN
198 0261 2
199 0262 2 LOCAL
200 0263 2 RECORD_ID : WORD,
201 0264 2 STATUS;
202 0265 2
203 0266 2 RECORD_ID = .REC_ADDR[IRCSW_ID];

```

```

: 204 0267 5
: 205 0268 5
: 206 0269 5
: 207 0270 5
: 208 0271 5
: 209 0272 5
: 210 0273 5
: 211 0274 5
: 212 0275 5
: 213 0276 5
: 214 0277 5
: 215 0278 5
: 216 0279 5
: 217 0280 6
: 218 0281 5
: 219 0282 6
: 220 0283 5
: 221 0284 5
: 222 0285 5
: 223 0286 5
: 224 0287 5
: 225 0288 5
: 226 0289 5
: 227 0290 5
: 228 0291 5
: 229 0292 5
: 230 0293 5
: 231 0294 5
: 232 0295 5
: 233 0296 5
: 234 0297 5
: 235 0298 5
: 236 0299 5
: 237 0300 5
: 238 0301 5
: 239 0302 5
: 240 0303 4
: 241 0304 4
: 242 0305 4
: 243 0306 4
: 244 0307 4
: 245 0308 4
: 246 0309 4
: 247 0310 4
: 248 0311 4
: 249 0312 4
: 250 0313 3
: 251 0314 3
: 252 0315 3
: 253 0316 3
: 254 0317 3
: 255 0318 3
: 256 0319 3
: 257 0320 3
: 258 0321 3
: 259 0322 3
: 260 0323 3

```

```

: If RMS finds that Recovery Unit in which this record was
: locked is still active or the file has not been opened for
: write access, then RMS can not delete this record. If another
: stream has the current record locked, RMS returns the record
: as the non-deleted primary data record, and lets a higher
: level routine decide what to do with it. However, if it is
: the current stream that has the record locked, or if the
: current stream is able to lock the record but does not have
: write access to the file, then RMS considers the current
: record to be deleted, and positions to the next record in
: order to continue the search.
IF NOT (STATUS = RMSRU_RECLAIM())
THEN
  IF .STATUS<0,16> EQLU RMSERR(RLK)
  THEN
    EXITLOOP
  ELSE
    RMSGETNEXT_REC()

: If RMS is able to reclaim only some of the space occupied
: by the current primary data record it proceeds to
: position to the next record; otherwise, it is already
: positioned there.
ELSE
  IF .RECORD_ID EQLU .REC_ADDR[IRCSW_ID]
  THEN
    RMSGETNEXT_REC();
END

: If the current record is marked deleted, then continue the search
: for a non-deleted primary data record with the next record in the
: bucket.
ELSE
  RMSGETNEXT_REC();

: The previous records RMS has looked at were all deleted. If the
: record RMS has positioned to matches the key in keybuffer 2
: according to the search characteristics then continue with the
: search to see if it too is marked deleted, or whether it maybe
: returned as the non-deleted primary data record.
RETURN_ON_ERROR (RMSSEARCH_TREE());
END;

: RMS has found a record to return as the non-deleted primary data
: record. If this record was updated within a Recovery Unit, then
: re-format the record provided the Recovery Unit has completed and the
: stream has write access to the file.
IF .REC_ADDR[IRCSV_RU_UPDATE]
AND
.IFAB[IFBSV_WRTACC]
AND

```

```

: 261 0324 3
: 262 0325 3
: 263 0326 3
: 264 0327 3
: 265 0328 3
: 266 0329 3
: 267 0330 3
: 268 0331 3
: 269 0332 3
: 270 0333 3
: 271 0334 3
: 272 0335 3
: 273 0336 2
: 274 0337 2
: 275 0338 2
: 276 0339 2
: 277 0340 2
: 278 0341 3
: 279 0342 3
: 280 0343 3
: 281 0344 3
: 282 0345 3
: 283 0346 3
: 284 0347 3
: 285 0348 3
: 286 0349 3
: 287 0350 3
: 288 0351 3
: 289 0352 4
: 290 0353 3
: 291 0354 4
: 292 0355 4
: 293 0356 4
: 294 0357 4
: 295 0358 4
: 296 0359 4
: 297 0360 4
: 298 0361 4
: 299 0362 3
: 300 0363 3
: 301 0364 3
: 302 0365 3
: 303 0366 2
: 304 0367 2
: 305 0368 1

```

```

NOT .REC_ADDR[IRCSV_RU_DELETE]
THEN
  RMSPU_RECLAIM();

! RMS has found a record to return. Extract the record's ID and the
! VBN of the bucket it is in for use in updating the NRP, and then
! return success.
IRAB[IRBSW_RFA_ID] = IRC$ ID(REC_ADDR);
IRAB[IRBSL_RFA_VBN] = .BBOCK[.IRAB[IRBSL_CURBDB], BDB$V_VBN];
RETURN RMSSUC();
END;

! RMS is to position to a non-deleted primary data record by an alternate
! key value.
BEGIN
LOCAL
  STATUS;

! Search all the SIDR arrays whose keys match the key in keybuffer 2
! according to the characteristics of the search until either a non-deleted
! primary data record is found, or all SIDRs with appropriate keys are
! exhausted.
STATUS = RMSSEARCH_SIDR();

IF .STATUS<0, 16> EQL RMSERR(RNF)
THEN
  BEGIN
    GLOBAL REGISTER
      R_BDB_STR;

    IF .IRAB[IRBSL_CURBDB] NEQ 0
    THEN
      RELEASE(IRAB[IRBSL_CURBDB]);
    END;
  RETURN .STATUS
END;
END;
END;

```

```

.TITLE RM3POSKEY
.IDENT \V04-000\

.EXTRN RMSSEARCH_TREE
.EXTRN RMSGETNEXT_REC, RMSRLSBKT
.EXTRN RMSRU_RECLAIM, RMSSEARCH_SIDR

.PSECT RMSRMS3.NOWRT, GBL, PIC,2

```


		41	A9	94	00002	PUSHL	R4	: 0153	
		20	A9	D4	00005	CLRB	65(IRAB)	: 0233	
		76	A9	B4	00008	CLRL	32(IRAB)	: 0234	
				0000G	30	0000B	CLRW	118(IRAB)	: 0235
	6B			50	E9	0000E	BSBW	RM\$SEARCH TREE	: 0237
		21	A7	95	00011	BLBC	STATUS, 10\$		
			68	12	00014	TSTB	33(IDX_DFN)	: 0243	
			02	E0	00016	BNEQ	11\$		
04	66		05	E1	0001A	BBS	#2, (REC_ADDR), 2\$: 0247	
2B	66		05	E1	0001E	BBC	#5, (REC_ADDR), 6\$: 0249	
1C	54	01	A6	B0	00022	BBC	#5, (REC_ADDR), 4\$: 0258	
			00000000G	EF	16	00026	MOVW	1(REC_ADDR), RECORD_ID	: 0266
	09			50	E8	0002C	JSB	RM\$RU RECLAIM	: 0280
	82AA	8F		50	B1	0002F	BLBS	STATUS, 3\$	
				13	13	00034	CMPW	STATUS, #33450	: 0282
				06	11	00036	BEQL	6\$	
	01	A6		54	B1	00038	BRB	4\$: 0286
				03	12	0003C	CMPW	RECORD_ID, 1(REC_ADDR)	: 0294
			0000G	30	0003E	4\$:	BNEQ	5\$	
			0000G	30	00041	5\$:	BSBW	RM\$GETNEXT_REC	: 0304
		CF		50	E8	00044	BSBW	RM\$SEARCH_TREE	: 0312
				59	11	00047	BLBS	STATUS, 1\$	
0E	66			06	E1	00049	BRB	13\$	
	0A	06	AA	E9	0004D	BBC	#6, (REC_ADDR), 7\$: 0320	
06	66		05	E0	00051	BLBC	6(IFAB), 7\$: 0322	
			00000000G	EF	16	00055	BBS	#5, (REC_ADDR), 7\$: 0324
	03	00B7	CA	91	0005B	7\$:	JSB	RM\$RU RECLAIM	: 0326
				06	1E	00060	CMPB	183(IFAB), #3	: 0332
	50	01	A6	9A	00062	BGEQU	8\$		
				04	11	00066	MOVZBL	1(REC_ADDR), R0	
	74	50	01	A6	3C	00068	BRB	9\$	
		A9		50	B0	0006C	MOVZWL	1(REC_ADDR), R0	
	70	50	20	A9	D0	00070	MOVW	R0, 1T6(IRAB)	
		50	1C	A0	D0	00074	MOVL	32(IRAB), R0	: 0333
				01	D0	00079	MOVL	28(R0), 112(IRAB)	
				24	11	0007C	MOVL	#1, R0	: 0334
			0000G	30	0007E	10\$:	BRB	13\$	
				50	D0	00081	BSBW	RM\$SEARCH SIDR	: 0350
	82B2	8F		51	B1	00084	MOVL	R0, STATUS	
				14	12	00089	CMPW	STATUS, #33458	: 0352
				20	A9	D5	BNEQ	12\$	
				0F	13	0008E	TSTL	32(IRAB)	: 0359
	54	20	A9	D0	00090	BEQL	12\$		
		20	A9	D4	00094	MOVL	32(IRAB), BDB	: 0361	
				7E	D4	00097	CLRL	32(IRAB)	
			0000G	30	00099	CLRL	-(SP)		
	5E			04	C0	0009C	BSBW	RM\$RLSBKT	
	50			51	D0	0009F	ADDL2	#', SP	
				10	BA	000A2	MOVL	STATUS, R0	: 0364
				05	000A4	13\$:	POPR	#^M<R4>	: 0368
							RSB		

; Routine Size: 165 bytes, Routine Base: RM\$RMS3 + 0000

; 306 0369 1
; 307 0370 1 END

: 308 0371 1
: 309 0372 0 ELUDOM

PSECT SUMMARY

Name Bytes Attributes
RMSRMS3 165 NOVEC,NOWRT, RD , EXE,NOSHR, GBL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	Symbols		Pages Mapped	Processing Time
	Total	Loaded Percent		
_\$255\$DUA28:[RMS.OBJ]RMS.L32;1	3109	42 1	154	00:00.4

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:RM3POSKEY/OBJ=OBJ\$:RM3POSKEY MSRC\$:RM3POSKEY/UPDATE=(ENH\$:RM3POSKEY)

: Size: 165 code + 0 data bytes
: Run Time: 00:06.2
: Elapsed Time: 00:20.1
: Lines/CPU Min: 3617
: Lexemes/CPU-Min: 14508
: Memory Used: 90 pages
: Compilation Complete

This image displays a grid of 140 terminal windows, arranged in 10 rows and 14 columns. Each window contains text-based output from a VAX/VMS system. The windows are organized into several distinct sections, each with a title:

- RM3NEXTRE LIS** (Row 1, Column 6)
- RM3OPEN LIS** (Row 2, Column 8)
- RM3POSRA LIS** (Row 3, Column 12)
- RM3PCKUNP LIS** (Row 5, Column 8)
- RM3POSKEY LIS** (Row 6, Column 10)
- RM3POSSEQ LIS** (Row 10, Column 12)

The text within the windows includes system status reports, error messages, and diagnostic data. Some windows show lists of files or directory structures, while others display numerical data or error codes. The overall layout is a dense, structured array of system information.