

RRRRRRRRRRRR		MMM		MMM	SSSSSSSSSSSS
RRRRRRRRRRRR		MMM		MMM	SSSSSSSSSSSS
RRRRRRRRRRRR		MMM		MMM	SSSSSSSSSSSS
RRR	RRR	MMMMMM	MMMMMM	SSS	
RRR	RRR	MMMMMM	MMMMMM	SSS	
RRR	RRR	MMMMMM	MMMMMM	SSS	
RRR	RRR	MMM	MMM	SSS	
RRR	RRR	MMM	MMM	SSS	
RRR	RRR	MMM	MMM	SSS	
RRRRRRRRRRRR		MMM		SSSSSSSSSS	
RRRRRRRRRRRR		MMM		SSSSSSSSSS	
RRRRRRRRRRRR		MMM		SSSSSSSSSS	
RRR	RRR	MMM			SSS
RRR	RRR	MMM			SSS
RRR	RRR	MMM			SSS
RRR	RRR	MMM			SSS
RRR	RRR	MMM			SSS
RRR	RRR	MMM			SSS
RRR	RRR	MMM			SSS
RRR	RRR	MMM		SSSSSSSSSSSS	
RRR	RRR	MMM		SSSSSSSSSSSS	
RRR	RRR	MMM		SSSSSSSSSSSS	

SY

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

```

RRRRRRRR      MM      MM      333333      XX      XX      KK      KK      EEEEEEEEEEE      YY      YY      000000
RRRRRRRR      MM      MM      333333      XX      XX      KK      KK      EEEEEEEEEEE      YY      YY      000000
RR      RR      MMMM      MMMM      33      33      XX      XX      KK      KK      EE      YY      YY      00      00
RR      RR      MMMM      MMMM      33      33      XX      XX      KK      KK      EE      YY      YY      00      00
RR      RR      MM      MM      MM      33      33      XX      XX      KK      KK      EE      YY      YY      00      00
RRRRRRRR      MM      MM      33      33      XX      XX      KKKKKK      YY      YY      00      00
RRRRRRRR      MM      MM      33      33      XX      XX      KKKKKK      YY      YY      00      00
RR      RR      MM      MM      MM      33      33      XX      XX      KK      KK      EE      YY      YY      00      00
RR      RR      MM      MM      MM      33      33      XX      XX      KK      KK      EE      YY      YY      00      00
RR      RR      MM      MM      MM      33      33      XX      XX      KK      KK      EE      YY      YY      00      00
RR      RR      MM      MM      MM      33      33      XX      XX      KK      KK      EE      YY      YY      00      00
RR      RR      MM      MM      MM      33      33      XX      XX      KK      KK      EE      YY      YY      00      00
RR      RR      MM      MM      MM      333333      XX      XX      KK      KK      EEEEEEEEEEE      YY      000000
RR      RR      MM      MM      MM      333333      XX      XX      KK      KK      EEEEEEEEEEE      YY      000000

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 RM3XKEYO (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 routine fills in the KEY XAB from the disk
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: D. M. BOUSQUET
45 0045 1 CREATION DATE: 18-AUG-78 14:19
46 0046 1
47 0047 1 MODIFIED BY:
48 0048 1
49 0049 1 V03-008 DAS0001 David Solomon 25-Mar-1984
50 0050 1 Fix broken branches.
51 0051 1
52 0052 1 V03-007 LJA0099 Laurie J. Anderson 26-Sep-1983
53 0053 1 Fix bugcheck, where someone will do a $DISPLAY on a
54 0054 1 file opened with BRO. In this case, the index descriptors
55 0055 1 are not allocated.
56 0056 1
57 0057 1 V03-006 MCN0002 Maria del C. Nasr 31-Mar-1983
```

More linkages reorganization.

V03-005	MCN0001	Maria del C. Nasr	01-Mar-1983
	Reorganize linkages		
V03-004	KBT0289	Keith B. Thompson	23-Aug-1982
	Reorganize psects		
V03-003	KBT0074	Keith B. Thompson	29-Jun-1982
	Enable the processing of no-contiguous key descriptors		
V03-002	KBT0017	Keith Thompson	19-Mar-1982
	Ignore compression bits when filling in xab of a prologue 1,2 file		
V03-001	JWH0001	Jeffrey W. Horn	16-March-1982
	Fix writing into space beyond the old end of XABKEY.		
V02-006	DJD0001	Darrell Duffy	1-March-1982
	Fix probing in RMSXKEY03.		
V02-005	KBT0004	K B Thompson	8-Feb-1982
	Correct compression bits when filling in key xab from prologue and stuff the prologue version if long key=0 xab		
V02-004	CDS0001	C Saether	9-Aug-1981
	Use alternate linkage declaration for RELEASE.		
V02-003	REFORMAT	D M WALP	24-JUL-1980

REVISION HISTORY:

Wendy Koenig, 24-OCT-78 14:03
X0002 - make changes caused by sharing conventions

LIBRARY 'RMSLIB:RMS';
REQUIRE 'RMSSRC:RMSIDXDEF';

! Define default psects for code

PSECT

CODE = RMSRMS3(PSECT_ATTR),
PLIT = RMSRMS3(PSECT_ATTR);

LINKAGE

L_CACHE,
L_CHKSUM,
L_FABREG_7,
L_LINK_7-10_11,
L_RELEASE_FAB;

! External Routines

RM3XKEYO
V04-000

N 1
16-Sep-1984 02:12:32 VAX-11 Bliss-32 V4.0-742 Page 3
14-Sep-1984 13:01:45 DISK\$VMSMASTER:[RMS.SRC]RM3XKEYO.B32;1 (1)

:	115	0179	1	EXTERNAL ROUTINE	:	RL\$CACHE ADDRESSING_MODE(LONG_RELATIVE),
:	116	0180	1	RM\$CACHE	:	RL\$CHKSUM,
:	117	0181	1	RM\$CHKSUM	:	RL\$LINK 7 10 11,
:	118	0182	1	RM\$GET_NEXT_KEY	:	RL\$RELEASE_FAB ADDRESSING_MODE(LONG_RELATIVE);
:	119	0183	1	RM\$RELEASE	:	
:	120	0184	1		:	

```
122 0185 1 %SBTTL 'RM$XKEYO3'
123 0186 1 GLOBAL ROUTINE RM$XKEYO3 ( XAB : REF BBLOCK ) : RL$FABREG_7 =
124 0187 1
125 0188 1 ++
126 0189 1
127 0190 1 FUNCTIONAL DESCRIPTION:
128 0191 1
129 0192 1 The KEY XAB is filled in from the KEY descriptor
130 0193 1
131 0194 1 CALLING SEQUENCE:
132 0195 1
133 0196 1 RM$XKEYO3 ( XAB )
134 0197 1
135 0198 1 INPUT PARAMETERS:
136 0199 1
137 0200 1 XAB - Pointer to the KEY XAB we are filling in
138 0201 1
139 0202 1 IMPLICIT INPUTS:
140 0203 1
141 0204 1 $XAB
142 0205 1 [REF] - Key of reference
143 0206 1 [KNM] - Pointer to key name buffer
144 0207 1
145 0208 1 $KEYDEF
146 0209 1 [KEY$$_KEYNAM] - Size of the keyname buffer, 32
147 0210 1 [KEY$_KEYNAM] - Buffer of the keyname (table)
148 0211 1
149 0212 1 $IFAB
150 0213 1 [IFB$$_ORGCASE] - File organization
151 0214 1 [IFB$$_IDX] - Indexed file organization constant
152 0215 1 [IFB$$_MODE] - Mode of the key name buffer
153 0216 1 [IFB$$_PLG_VER] - Index file prologue version
154 0217 1
155 0218 1 $IDX_DFN
156 0219 1 [all fields in the index descriptor are input,
157 0220 1 but specifically reference a few]
158 0221 1 [IANUM] - Index area number
159 0222 1 [SEGMENTS] - Number of segments
160 0223 1 [FIXED_BLN] - Fixed block length analogous to the XAB
161 0224 1 [POSITION] - Position of first segment
162 0225 1 [SIZE] - Size of first segment
163 0226 1
164 0227 1 OUTPUT PARAMETERS:
165 0228 1
166 0229 1 XAB - Untouched by this routine
167 0230 1
168 0231 1 IMPLICIT OUTPUTS:
169 0232 1
170 0233 1 $XAB
171 0234 1 [all fields are filled in from the index descriptor]
172 0235 1
173 0236 1 ROUTINE VALUE:
174 0237 1
175 0238 1 RMSERR
176 0239 1 (SUC) - Success code
177 0240 1
178 0241 1 SIDE EFFECTS:
```

```
179 0242 1 | none
180 0243 1 |
181 0244 1 | --
182 0245 1 |
183 0246 2 | BEGIN
184 0247 2 |
185 0248 2 | EXTERNAL REGISTER
186 0249 2 |   R_IDX_DFN_STR,
187 0250 2 |   COMMON_FAB_STR;
188 0251 2 |
189 0252 2 | LOCAL
190 0253 2 |   SAV_BDB,
191 0254 2 |   KEY_DESC      : REF BBLOCK;
192 0255 2 |
193 0256 2 | ! If BIO was set in the FAB, then we only want to make sure user knows that
194 0257 2 | ! the NUM_KEYS is not filled in
195 0258 2 |
196 0259 2 | IF .IFAB [ IFB$B_NUM_KEYS ] EQL 0
197 0260 2 | THEN
198 0261 2 |   RETURN RMSSUC( OK_NOP );
199 0262 2 |
200 0263 2 | ! Let's first check to see if this is a indexed file
201 0264 2 |
202 0265 2 | IF .IFAB [ IFB$B_ORGCASE ] EQL IFB$C_IDX
203 0266 2 | THEN
204 0267 2 |   BEGIN
205 0268 2 |
206 0269 2 |   ! Now to make sure that the reference input is valid find the internal
207 0270 2 |   ! index descriptor
208 0271 2 |
209 0272 2 |   ! Start with the primary descriptor. NOTE: We cannot call rm$key_desc
210 0273 2 |   ! here because of register problems but we do know that the first
211 0274 2 |   ! index descriptor off the ifab is the one for key 0 so get it that way
212 0275 2 |
213 0276 2 |   IDX_DFN = .IFAB [ IFB$L_IDX_PTR ];
214 0277 2 |
215 0278 2 |   ! Make sure that the index descriptors have been allocated before
216 0279 2 |   ! you use it. Return success if it's not there, that's okay.
217 0280 2 |   IF .IDX_DFN EQLU 0
218 0281 2 |   THEN
219 0282 2 |     RETURN RMSSUC( SUC );
220 0283 2 |
221 0284 2 |   ! Loop until we find it
222 0285 2 |
223 0286 2 |   WHILE .XAB [ XAB$B_REF ] NEQ .IDX_DFN [ IDX$B_KEYREF ]
224 0287 2 |   DO
225 0288 2 |
226 0289 2 |     ! If we ran out of keys there is a problem
227 0290 2 |
228 0291 2 |     IF NOT RM$GET_NEXT_KEY()
229 0292 2 |     THEN
230 0293 2 |       RETURN RMSERR( REF );
231 0294 2 |
232 0295 2 |   ! Now to read in prologue descriptor
233 0296 2 |
234 0297 4 | BEGIN
235 0298 4 | GLOBAL REGISTER
```

```

236      COMMON_IO_STR;
237
238      LOCAL
239      STATUS;
240
241      STATUS = RM$CACHE( .IDX_DFN [ IDX$L_VBN ], 512, 0);
242
243      ! If error then return with error code in status
244      !
245      IF NOT .STATUS
246      THEN
247          RETURN .STATUS;
248
249      ! Now to release the bucket and check it
250      !
251      RETURN_ON_ERROR( RM$CHKSUM(
252                      RM$RELEASE(0)
253                      );
254
255      ! Now point to the key descriptor in the prologue
256      !
257      KEY_DESC = .BKT_ADDR + .IDX_DFN [ IDX$W_OFFSET ];
258
259      ! Now to save the BDB before CH$MOVE clobbers it
260      !
261      SAV_BDB = .BDB
262
263      END;
264
265      ! Now to do a straight move from the key descriptor to the XAB
266      !
267      CH$MOVE( $BYTEOFFSET( KEY$T_KEYNAM ) - $BYTEOFFSET ( KEY$B_IANUM ),
268              KEY_DESC [ KEY$B_IANUM ],
269              XAB [ XAB$B_IAN ] );
270
271      ! If this is a prologue 3 file correct the compression bits
272      !
273      IF .IFAB [ IFB$B_PLG_VER ] GEQU 3
274      THEN
275          ! SET in the prologue = CLEAR in the xab
276          !
277          XAB [ XAB$B_FLG ] = .XAB [ XAB$B_FLG ] XOR ( XAB$M_IDX_NCMPR OR
278                                                         XAB$M_KEY_NCMPR OR
279                                                         XAB$M_DAT_NCMPR );
280
281      ! If this is a long key xab and it is key-0 (primary)
282      ! then fill in the prologue version number
283      !
284      IF ( .XAB [ XAB$B_BLN ] EQLU XAB$K_KEYLEN ) AND
285          ( .XAB [ XAB$B_REF ] EQLU 0 )
286      THEN
287          XAB [ XAB$B_PROLOG ] = .IFAB [ IFB$B_PLG_VER ];
288
289      ! If the user has a key name buffer fill it in
290      !
291      IF .XAB [ XAB$L_KNM ] NEQ 0
292
```

RM3XKEYO
V04-000

RM\$XKEYO3

E 2
16-Sep-1984 02:12:32
14-Sep-1984 13:01:45

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[RMS.SRC]RM3XKEYO.B32;1 Page 7 (2)

```

: 293      0356 3      THEN
: 294      0357 4          BEGIN
: 295      0358 4          LOCAL
: 296      0359 4              KNM_ADDR;
: 297      0360 4
: 298      0361 4          KNM_ADDR = .XAB [ XAB$L_KNM ];
: 299      0362 4
: 300      0363 4          ! Probe it
: 301      0364 4          !
: 302      P 0365 4          IFNOWRT( %REF( KEY$$_KEYNAM ), .KNM_ADDR, IFAB [ IFB$B_MODE ],
: 303      P 0366 4              BEGIN
: 304      P 0367 4                  GLOBAL REGISTER
: 305      P 0368 4                      R_BDB_STR;
: 306      P 0369 4
: 307      P 0370 4                  BDB = .SAV_BDB;
: 308      P 0371 4                  RM$RELEASE(0);
: 309      P 0372 4                  RETURN RM$ERR(KNM)
: 310      0373 4                      END);
: 311      0374 4
: 312      0375 4
: 313      0376 4          ! Now to move the buffer
: 314      0377 4          !
: 315      0378 4          CH$MOVE( KEY$$_KEYNAM, KEY_DESC [ KEY$T_KEYNAM ], .KNM_ADDR )
: 316      0379 4
: 317      0380 3          END;
: 318      0381 3
: 319      0382 3          ! Now move last long word.
: 320      0383 3          !
: 321      0384 3          XAB [ XAB$L_DVB ] = .KEY_DESC [ KEY$L_LDVBN ];
: 322      0385 3
: 323      0386 4          BEGIN
: 324      0387 4              GLOBAL REGISTER
: 325      0388 4                  R_BDB_STR;
: 326      0389 4
: 327      0390 4              BDB = .SAV_BDB;
: 328      0391 4              RM$RELEASE(0)
: 329      0392 4              END
: 330      0393 4          END;
: 331      0394 2
: 332      0395 2          ! Now to return the success code if all went well
: 333      0396 2          !
: 334      0397 2          RETURN RM$SUC( SUC )
: 335      0398 3
: 336      0399 3          END;
: 337      0400 1
```

```

.TITLE  RM3XKEYO
.IDENT  \V04-000\

.EXTRN  RM$CACHE, RM$CHKSUM
.EXTRN  RM$GET_NEXT_KEY
.EXTRN  RM$RELEASE

.PSECT  RM$RMS3, NOWRT, GBL, PIC, 2
```

007C 8F BB 00000 RM\$XKEYO3::

	5E		08	C2	00004	PUSHR	#^M<R2,R3,R4,R5,R6>	:	0186
		00B2	CA	95	00007	SUBL2	#8, SP	:	
			07	12	0000B	TSTB	178(IFAB)	:	0259
	50	8059	8F	3C	0000D	BNEQ	1\$	:	
			53	11	00012	MOVZWL	#32857, R0	:	0261
	02	23	AA	91	00014	BRB	6\$	:	
			03	13	00018	CMPB	35(IFAB), #2	:	0265
			00C2	31	0001A	BEQL	3\$	:	
	57	00AC	CA	D0	0001D	BRW	13\$	:	
			F6	13	00022	MOVL	172(IFAB), IDX_DFN	:	0276
	56	20	AE	D0	00024	BEQL	2\$	:	0280
21	A7	17	A6	91	00028	MOVL	XAB, R6	:	0286
			0D	13	0002D	CMPB	23(R6), 33(IDX_DFN)	:	
			0000G	30	0002F	BEQL	5\$	:	
F3			50	E8	00032	BSBW	RMSGET_NEXT_KEY	:	0291
50	875C		8F	3C	00035	BLBS	R0, 4\$	:	
			2B	11	0003A	MOVZWL	#34652, R0	:	0293
			53	D4	0003C	BRB	6\$	:	
52	0200		8F	3C	0003E	CLRL	R3	:	0304
51	0A		A7	D0	00043	MOVZWL	#512, R2	:	
	00000000G		EF	16	00047	MOVL	10(IDX_DFN), R1	:	
6D			50	E9	0004D	JSB	RMS\$CACHE	:	
			0000G	30	00050	BLBC	STATUS, 10\$	:	0308
04	AE		50	D0	00053	BSBW	RMSCHKSUM	:	0316
	0E	04	AE	E8	00057	MOVL	R0, STATUS	:	
			53	D4	0005B	BLBS	STATUS, 7\$	:	
	00000000G		EF	16	0005D	CLRL	R3	:	
50	04		AE	D0	00063	JSB	RMS\$RELEASE	:	
			79	11	00067	MOVL	STATUS, R0	:	
50	0E		A7	3C	00069	BRB	14\$	:	
6E			50	C1	0006D	MOVZWL	14(IDX_DFN), R0	:	0320
			54	D0	00071	ADDL3	R0, BKT_ADDR, KEY_DESC	:	
04	AE		06	C1	00075	MOVL	BDB, SAV_BDB	:	0324
	6E		2E	28	00079	ADDL3	#6, KEY_DESC, -(SP)	:	0332
08	A6		CA	91	0007E	MOVC3	#46, @ (SP)+, 8(R6)	:	
		00B7	05	1F	00083	CMPB	183(IFAB), #3	:	0336
			8F	8C	00085	BLSSU	8\$	:	
12	A6	C8	8F	91	0008A	XORB2	#200, 18(R6)	:	0341
4C	8F	01	0B	12	0008F	CMPB	1(R6), #76	:	0348
		17	A6	95	00091	BNEQ	9\$	:	
			06	12	00094	TSTB	23(R6)	:	0349
48	A6	00B7	CA	90	00096	BNEQ	9\$	:	
		38	A6	D5	0009C	MOVB	183(IFAB), 72(R6)	:	0351
			26	13	0009F	TSTL	56(R6)	:	0355
	55	38	A6	D0	000A1	BEQL	12\$	:	
65	20	0A	AA	0D	000A5	MOVL	56(R6), KNM_ADDR	:	0361
			13	12	000AA	PROBEW	10(IFAB), #32, (KNM_ADDR)	:	0373
	54	04	AE	D0	000AC	BNEQ	11\$	:	
			53	D4	000B0	MOVL	SAV_BDB, BDB	:	
	00000000G		EF	16	000B2	CLRL	R3	:	
50	8774		8F	3C	000B8	JSB	RMS\$RELEASE	:	
			23	11	000BD	MOVZWL	#34676, R0	:	
			34	C1	000BF	BRB	14\$	:	
7E	6E		20	28	000C3	ADDL3	#52, KEY_DESC, -(SP)	:	0378
65	9E		8F	C1	000C7	MOVC3	#32, @ (SP)+, (KNM_ADDR)	:	
50	6E	00000054	60	D0	000CF	ADDL3	#84, KEY_DESC, R0	:	0384
3C	A6					MOVL	(R0), 60(R6)	:	

RM3XKEYO
V04-000

RM\$XKEYO3

G 2
16-Sep-1984 02:12:32
14-Sep-1984 13:01:45

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[RMS.SRC]RM3XKEYO.B32;1
Page 9
(2)

54	04	AE	D0	000D3	MOVL	SAV_BDB, BDB	:	0390	
		53	D4	000D7	CLRL	R3	:	0391	
	00000000G	EF	16	000D9	JSB	RM\$RELEASE	:		
50		01	D0	000DF	13\$:	MOVL	#1, R0	:	0398
5E		08	C0	000E2	14\$:	ADDL2	#8, SP	:	0400
	007C	8F	BA	000E5	POPR	#^M<R2,R3,R4,R5,R6>	:		
		05	000E9	RSB			:		

; Routine Size: 234 bytes, Routine Base: RM\$RMS3 + 0000

: 338 0401 1
: 339 0402 1 END
: 340 0403 1
: 341 0404 0 ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
RM\$RMS3	234 NOVEC,NOWRT, RD ,	EXE,NOSHR, GBL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[RMS.OBJ]RMS.L32;1	3109	58	1	154	00:00.4

COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS\$:RM3XKEYO/OBJ=OBJ\$:RM3XKEYO MSRC\$:RM3XKEYO/UPDATE=(ENH\$:RM3XKEYO)

; Size: 234 code + 0 data bytes
; Run Time: 00:07.0
; Elapsed Time: 00:16.7
; Lines/CPU Min: 3472
; Lexemes/CPU-Min: 15292
; Memory Used: 98 pages
; Compilation Complete

0329 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

