

FILEID**RM3CMPKEY

I 15

RRRRRRRR	MM	MM	333333	CCCCCCCC	MM	MM	PPPPPPPP	KK	KK	EEEEEEEEE	YY	YY
RRRRRRRR	MM	MM	333333	CCCCCCCC	MM	MM	PPPPPPPP	KK	KK	EEEEEEEEE	YY	YY
RR RR	RR	MMMM	MM	33	33	CC	MM	MM	PP	PP	KK	EE
RR RR	RR	MM	MM	33	33	CC	MM	MM	PP	PP	KK	EE
RR RR	RR	MM	MM	33	33	CC	MM	MM	PP	PP	KK	EE
RR RR	RR	MM	MM	33	33	CC	MM	MM	PP	PP	KK	EE
RRRRRRRR	MM	MM	33	CC	MM	MM	PPPPPPPP	KKKKKK	EE	EEEEEEE	YY	YY
RRRRRRRR	MM	MM	33	CC	MM	MM	PPPPPPPP	KKKKKK	EE	EEEEEEE	YY	YY
RR RR	RR	MM	MM	33	CC	MM	MM	PP	KK	KK	EE	YY
RR RR	RR	MM	MM	33	CC	MM	MM	PP	KK	KK	EE	YY
RR RR	RR	MM	MM	33	33	CC	MM	MM	PP	KK	KK	EE
RR RR	RR	MM	MM	33	33	CC	MM	MM	PP	KK	KK	EE
RR RR	RR	MM	MM	333333	CCCCCCCC	MM	MM	PP	KK	KK	EEEEEEEEE	YY
RR RR	RR	MM	MM	333333	CCCCCCCC	MM	MM	PP	KK	KK	EEEEEEEEE	YY
....

LL		SSSSSSS
LL		SSSSSSS
LL		SS
LLLLLLLLL		SSSSSSS
LLLLLLLLL		SSSSSSS

(2)	108	DECLARATIONS
(3)	128	RM\$NULLKEY - COMPARE KEY TO NULL KEY VALUE
(4)	184	RM\$COMPARE_KEY - COMPARE TWO KEYS

```
0000 1      $BEGIN RM3CMPKEY,000,RM$RMS3,<>,<PIC,NOWRT,QUAD>
0000 2
0000 3
0000 4 :*****
0000 5 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 6 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 7 :* ALL RIGHTS RESERVED.
0000 8
0000 9
0000 10 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 :* TRANSFERRED.
0000 16 :*
0000 17 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 :* CORPORATION.
0000 20 :*
0000 21 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 :*
0000 24 :*
0000 25 :*****
0000 26 :*
0000 27 :*
0000 28 :++
0000 29 : Facility:    rms32 index sequentail file organization
0000 30 :
0000 31 : Abstract:
0000 32 :     this module compares a contiguous or a noncontiguous search key
0000 33 :     to either a data record or another contiguous key
0000 34 :
0000 35 :
0000 36 :
0000 37 : Environment:
0000 38 :     vax/vms operating system
0000 39 :     internal exec routines.
0000 40 :
0000 41 : Author: d. h. gillespie, creation date: 25-MAR-1978
0000 42 :
0000 43 : Modified By:
0000 44 :
0000 45 :     V03-006 MCN0002      Maria del C. Nasr      04-Apr-1983
0000 46 :     Preserve register 3 in call to RM$NULLKEY, so that we can
0000 47 :     use general linkages. Also, move input parameter to R1
0000 48 :     to be used by RM$COMPARE_KEY.
0000 49 :
0000 50 :     V03-005 DAS0001      David Solomon      26-Jan-1983
0000 51 :     Add comparison of 64-bit binary keys (IN8 and BN8).
0000 52 :
0000 53 :     V03-004 TMK0003      Todd M. Katz      16-Dec-1982
0000 54 :     I made another mistake in TMK0002. I used a BITB instead of a
0000 55 :     CMPZV and this basically caused some comparisons of segmented
0000 56 :     keys to fail. If at first you don't succeed....
0000 57 :
```

0000 58 : V03-003 TMK0002 Todd M. Katz 09-Dec-1982
0000 59 : I made a mistake in TMK0001. I used a BNEQ instead of a BEQL
0000 60 : and this basically caused all comparisons of segmented keys
0000 61 : to fail.
0000 62 :
0000 63 : V03-002 TMK0001 Todd M. Katz 02-Dec-1982
0000 64 : If the routine RMS\$COMPARE KEY is performing a contiguous key -
0000 65 : contiguous key compare, then there is a need only to perform
0000 66 : one comparison regardless of the number of segments. As the
0000 67 : routine currently is constructed, the comparisons are done
0000 68 : segment-by-segment even though both keys are contiguous and
0000 69 : only one comparison need be done. The routine was written like
0000 70 : this with heterogeneous segmented keys in mind where there would
0000 71 : be a need to always compare keys segment-by-segment, regardless
0000 72 : of whether they were or weren't contiguous, because the segments
0000 73 : could be of different types. Because RMS is no longer going to
0000 74 : implement such a key, the need to compare keys segment-by-
0000 75 : segment, if both are contiguous has disappeared, and only one
0000 76 : comparison need be done resulting in a significant performance
0000 77 : enhancement.
0000 78 :
0000 79 : V03-001 KBT0158 Keith B. Thompson 21-Aug-1982
0000 80 : Reorganize psects
0000 81 :
0000 82 : V02-008 MCN0001 Maria del C. Nasr 11-May-1981
0000 83 : Loop to compare all key segments independent of type.
0000 84 : Also, eliminate RMS\$COMP_ASCII routine, not needed anymore.
0000 85 :
0000 86 : V02-007 PSK0003 Paulina Knibbe 30-Apr-1981
0000 87 : Change branch in RMS\$COMP_ASCII to make it compare
0000 88 : correctly against high key value
0000 89 :
0000 90 : V02-006 PSK0002 Paulina Knibbe 10-Apr-1981
0000 91 : Add RMS\$COMP_ASCII routine to compare ascii-equivalent keys
0000 92 :
0000 93 : V02-005 PSK0001 Paulina Knibbe 11-Mar-1981
0000 94 : Change size field for key segments from words to bytes
0000 95 : Get datatype from the segment array
0000 96 :
0000 97 : V02-004 REFORMAT Ron Schaefer 30-Jul-1980 09:18
0000 98 : Reformat the source.
0000 99 :
0000 100 : Revision History:
0000 101 :
0000 102 : w. koenig 11-DEC-1978 fix bug in nullkey -- set bit indicating that
0000 103 : r3 points to a contiguous key and when comparing a
0000 104 : packed decimal zero, use a zero length
0000 105 :
0000 106 :--

DECLARATIONS

16-SEP-1984 01:07:09 VAX/VMS Macro V04-00
5-SEP-1984 16:24:18 [RMS.SRC]RM3CMPKEY.MAR;1Page 3
(2)

```
0000 108      .SBTTL DECLARATIONS
0000 109      ;
0000 110      ; Include Files:
0000 111      ;
0000 112      ;
0000 113      ;
0000 114      ; Macros:
0000 115      ;
0000 116      ;
0000 117      $IDXDEF
0000 118      ;
0000 119      ;
0000 120      ; Equated Symbols:
0000 121      ;
0000 122      ;
0000 123      ;
0000 124      ; Own Storage:
0000 125      ;
0000 126      ;
```

>

0000 128 .SBTTL RMSNULLKEY - COMPARE KEY TO NULL KEY VALUE
 0000 129
 0000 130 :++
 0000 131
 0000 132 RMSNULLKEY: compare key to null key value
 0000 133 this routine compares the current key in a data record
 0000 134 to the null value depending on the key data type.
 0000 135
 0000 136 Calling sequence:
 0000 137 bsbw rm\$nullkey
 0000 138
 0000 139 Input Parameters:
 0000 140 Address of data record
 0000 141
 0000 142 Implicit Inputs:
 0000 143 r7 = address of index descriptor for current key of reference
 0000 144 ap = same as for rm\$comparekey
 0000 145
 0000 146 Output Parameters:
 0000 147 r0 = 0 key is null
 0000 148 = 1 key not null
 0000 149
 0000 150 Implicit Outputs:
 0000 151 none
 0000 152
 0000 153 Routine Value:
 0000 154 none
 0000 155
 0000 156 Side Effects:
 0000 157 register r0,r1 are clobbered
 0000 158
 0000 159 :--
 0000 160
 0000 161 .ENABL LSB
 0000 162 RMSNULLKEY:
 51 08 AE BB 0000 163 PUSHR #^M<R3>
 0C DD D0 0002 164 MOVL 8(SP),R1 ; move input param for RMSCOMPARE KEY
 7E D4 0006 165 PUSHL #12 ; put packed decimal zero on stack
 21 A7 95 0008 166 CLRL -(SP) ; put zero on stack for binary compares
 OE 1C A7 02 E1 000F 167 TSTB IDX\$B_KEYREF(R7) ; is this the primary key?
 13 13 000D 168 BEQL 90\$; primary keys can not be null
 0014 169 BBC #IDX\$V_NULKEYS,IDX\$B_FLAGS(R7),90\$; if null keys not allowed,
 5C 06 88 0014 170 ; return not null key
 0017 171 BISB2 #6,AP ; signal this is a null key compare
 50 53 5E D0 0017 172 ; and that r3 points to a ctg key
 20 A7 9A 001A 173 MOVL SP,R3 ; setup key address to stack
 0B 10 001E 174 MOVZBL IDX\$B_KEYSZ(R7),R0 ; compare total size of key
 03 13 0020 175 BSBBL RMSCOMPARE_KEY ; call key compare
 0022 176 BEQL 100\$; r0 already equals zero
 50 01 9A 0022 178 90\$: MOVZBL #1,R0 ; set return value
 5E 08 C0 0025 179 100\$: ADDL2 #8,SP ; restore stack pointer
 08 BA 0028 180 POPR #^M<R3>
 05 002A 181 RSB ; return to caller
 002B 182 .DSABL LSB

```

002B 184 .SBTTL RMS$COMPARE_KEY - COMPARE TWO KEYS
002B 185
002B 186 ++
002B 187
002B 188 RM$COMPARE_KEY:
002B 189 given the index descriptor for the current key of reference, a contiguous
002B 190 or a non-contiguous search key is compared with the second character string
002B 191 which may be either another contiguous key or a data record. the compare
002B 192 is done according to the data type. each segment of the key is
002B 193 compared until the requested number of characters has been examined.
002B 194
002B 195 Calling sequence:
002B 196 bsbw rms$compare_key
002B 197
002B 198 Input Parameters:
002B 199 ap - bit 0 0 string pointed to by r1 is a data record
002B 200 ap - bit 0 1 string pointed to by r1 is contiguous key
002B 201 ap - bit 1 0 string pointed to by r3 is non-contiguous key
002B 202 ap - bit 1 1 string pointed to by r3 is contiguous key
002B 203 ap - bit 2 0 not null key compare
002B 204 ap - bit 2 1 null key compare
002B 205 r1 - address of data record/index
002B 206 r3 - address of search key
002B 207 r0 - number of characters to compare
002B 208 r7 - address of index descriptor for current key of refer
002B 209
002B 210 Implicit Inputs:
002B 211 none
002B 212
002B 213 Output Parameters:
002B 214 r0 - 0 index/data record = search key
002B 215 r0 - -1 index/data record > search key
002B 216 r0 - +1 index/data record < search key
002B 217
002B 218 Implicit Outputs:
002B 219 none
002B 220
002B 221 Routine Value:
002B 222 none
002B 223
002B 224 Side Effects:
002B 225 register r0,r1,r3 are clobbered
002B 226
002B 227 --
002B 228
002B 229 .ENABL LSB ; local symbols
002B 230
002B 231 RM$COMPARE_KEY::
002B 232 PUSHR #^M<R2,R4,R5,R6> ; save these registers
002B 233 MOVL R0,R4 ; setup size
002B 234 MOVAW IDX$W_POSITION(R7),R6 ; setup pter to pos/size/type array
002B 235 MOVZBL IDX$B_SEGMENTS(R7),R5 ; pickup # of segments in this key
003A 236
003A 237
003A 238 ; if it is necessary to position into data record, do so. positioning into
003A 239 data record on the first segment is handled differently than the others.
003A 240

```

0074 8F BB 54 50 D0 56 2C A7 3E 55 1E A7 9A	002B 0032 0036 003A 003A 003A 003A	002B 002F 0032 0034 0035 0036 0037 0038 0039 0040	#^M<R2,R4,R5,R6> MOVL R0,R4 MOVAW IDX\$W_POSITION(R7),R6 MOVZBL IDX\$B_SEGMENTS(R7),R5 ; save these registers ; setup size ; setup pter to pos/size/type array ; pickup # of segments in this key ; if it is necessary to position into data record, do so. positioning into ; data record on the first segment is handled differently than the others. ;
--	---	---	---

```

      50 86 3C 003A 241
      03 5C E8 003D 242      MOVZWL (R6)+,R0      ; disp in record of first key segment
      51 50 C0 0040 243      BLBS AP,5$      ; is this a data record?
      23 5C 01 E0 0043 244      ADDL2 R0,R1      ; yes, set record pter to addr of first key
      53 50 C0 0047 245 5$:      BBS #1,AP,20$      ; is key contiguous?
      1E 11 004A 246      ADDL2 R0,R3      ; no, set key pter to addr of first key char
      004C 247      BRB 20$      ; skip position for subsequent key
      004C 248
      004C 249      ; this logic calculates the start character of subsequent segments when r1
      004C 250      ; points to a data record.
      004C 251
      004C 252      ;
      004C 253
      52 50 86 3C 004C 254 10$:      MOVZWL (R6)+,R0      ; pickup disp of this key segment
      52 FA A6 3C 004F 255      MOVZWL -6(R6),R2      ; pickup disp of previous key
      50 52 C2 0053 256      SUBL2 R2,R0      ; subtract it from this segments disp
      52 FC A6 9A 0056 257      MOVZBL -4(R6),R2      ; pickup len of previous segment
      50 52 C2 005A 258      SUBL2 R2,R0      ; subtract it from this segment disp
      03 5C E8 005D 259      BLBS AP,15$      ; is this a data record?
      0060 260
      0060 261      ; add difference of first char this segment minus last char +1 of previous segment
      0060 262
      0060 263      ;
      0060 264
      03 51 50 C0 0060 265      ADDL2 R0,R1      ; position to first char of this key segment
      03 5C 01 E0 0063 266 15$:      BBS #1,AP,20$      ; is key contiguous?
      53 50 C0 0067 267      ADDL2 R0,R3      ; no, position to first char of this key seg
      006A 268
      006A 269      ; now calculate length of current compare
      006A 270
      006A 271      ;
      006A 272
      52 86 9A 006A 273 20$:      MOVZBL (R6)+,R2      ; get length of this segment
      50 86 9A 006D 274      MOVZBL (R6)+,R0      ; get datatype of this segments
      0070 275
      03 5C 02 00 ED 0070 276      CMPZV #0,#2,AP,#3      ; if both keys are contiguous then only
      05 13 0075 277      BEQLU 25$      ; one comparison will be necessary
      0077 278
      54 52 D1 0077 279      CMPL R2,R4      ; compare to remaining compare length
      03 1B 007A 280      BLEQU 30$      ; if it is gtr remaining
      52 54 D0 007C 281 25$:      MOVL R4,R2      ; compare remaining length only
      007F 282
      54 52 C2 007F 283 30$:      SUBL2 R2,R4      ; update # of char remaining in compare
      0082 284
      0082 285      ; compare is done according to data type.
      0082 286
      0082 287      ;
      0082 288
      0082 289      CASE TYPE=B,SRC=R0,-      ; DISPLIST=<STR,IN2,BN2,IN4,BN4,PAC,IN8,BN8>
      0082 290
      0096 291
      0096 292      ; data string compare
      0096 293
      0096 294      ;
      0096 295
      63 0D 5C 02 E0 0096 296 STR: BBS #2,AP,NULSTR      ; is this a null string compare?
      61 52 29 009A 297 CMPC3 R2,(R1),(R3)      ; compare characters

```

RMSCOMPARE_KEY - COMPARE TWO KEYS

16-SEP-1984 01:07:09 VAX/VMS Macro V04-00
5-SEP-1984 16:24:18 [RMS.SRC]RM3CMPKEY.MAR;1Page 7
(4)

```

44 12 009E 298      BNEQ   90$          ; if neq, terminate compare
54 D5 00A0 299 50$: TSTL   R4          ; check # of characters remaining
6F 12 00A2 300      BNEQ   140$         ; if more, loop for next segment
0073 31 00A4 301      BRW    145$         ; if no more, compare is done and eql
00A7 302
00A7 303
00A7 304 : compare key string segments to null value
00A7 305 :
00A7 306
61 52 1F A7 3B 00A7 307 NULSTR: SKPC  IDX$B_NULLCHAR(R7),R2,(R1) ; look for other than null char
F2 13 00AC 308 BEQL   50$          ; if none found, found null key
5E 11 00AE 309 BRB    130$         ; did not find null key
00B0 310
00B0 311
00B0 312 : packed decimal
00B0 313
00B0 314 :
00B0 315
52 02 C4 00B0 316 PAC:   MULL2  #2,R2          ; 4 bits to a nibble
52 D7 00B3 317 DECL   R2          ; sign does not count
06 02 E0 00B5 318 BBS    #2,AP,NULPAC       ; is this a null key compare?
83 81 5C 00B9 319 CMPP   R2,(R1)+,(R3)+     ; compare one packed decimal string
1A 11 00BD 320 BRB    60$          ; do signed branches
00BF 321
00BF 322 : compare packed decimal to zero, the null value
00BF 323
00BF 324 :
00BF 325
54 D4 00BF 326 NULPAC: CLRL   R4          ; length of packed decimal zero
83 D5 00C1 327 TSTL   (R3)+        ; skip binary zero
54 37 00C3 328 CMPP4  R4,(R3),R2,(R1)   ; the search null routine puts packed decimal
49 13 00C8 329 BEQL   140$          ; string is zero
42 11 00CA 330 BRB    130$          ; not zero
00CC 331
00CC 332 : signed word compare
00CC 333
00CC 334 :
00CC 335
83 81 B1 00CC 336 IN2:   CMPW   (R1)+,(R3)+    ; compare one word
08 11 00CF 337 BRB    60$          ; do signed branches
00D1 338
00D1 339 : unsigned word compare
00D1 340
00D1 341 :
00D1 342
83 81 B1 00D1 343 BN2:   CMPW   (R1)+,(R3)+    ; compare one word
0C 11 00D4 344 BRB    80$          ; do unsigned branches
00D6 345
00D6 346 : signed long word compare
00D6 347
00D6 348 :
00D6 349
83 81 D1 00D6 350 IN4:   CMPL   (R1)+,(R3)+    ; compare one long word
38 13 00D9 351 60$: BEQL   140$         ; if equal, check if done
31 19 C0DB 352 70$: BLSS   130$         ; key > data record
29 11 00DD 353 BRB    120$         ; key < data record
00DF 354

```

```

          00DF 355
          00DF 356 ; unsigned long word compare
          00DF 357
          00DF 358
83   81   D1   00DF 359 BN4: CMPL   (R1)+,(R3)+      ; compare one long word
2F   13   00E2 360 80$: BEQLU  140$      ; if equal, check if done
28   1F   00E4 361 90$: BLSSU   130$      ; key > data record
20   11   00E6 362 BRB     120$      ; key < data record
          00E8 363
          00E8 364 ; signed quadword compare
          00E8 365
          00E8 366
          00E8 367
04 A3  04 A1  D1   00E8 368 IN8: CMPL   4(R1),4(R3)    ; Compare upper longwords.
19   14   00ED 369 BGTR   120$      ; If gtr, Key < data.
1D   12   00EF 370 BNEQ   130$      ; Else if neq, key > data.
          00F1 371
63   61   D1   00F1 372 CMPL   (R1),(R3)    ; Compare lower longwords.
1D   13   00F4 373 BEQLU  140$      ; Both longwords match; key = data.
16   1F   00F6 374 BLSSU   130$      ; key > data.
OE   11   00F8 375 BRB     120$      ; key < data.
          00FA 376
          00FA 377 ; unsigned quadword compare
          00FA 378
          00FA 379
          00FA 380
04 A3  04 A1  D1   00FA 381 BN8: CMPL   4(R1),4(R3)    ; Compare upper longwords.
05   12   00FF 382 BNEQU  110$      ; If nequ, check if done.
63   61   D1   0101 383 CMPL   (R1),(R3)    ; Compare lower longwords.
0D   13   0104 384 BEQLU  140$      ; Both longwords match; key = data.
06   1F   0106 385 110$: BLSSU  130$      ; Key > data.
          0108 386
          0108 387
          0108 388 ; return search key < data record/index
          0108 389
          0108 390
          0108 391
50   FF   8F   98   0108 392 120$: CVTBL  #-1,R0
          0E   11   010C 393 BRB     150$      ; drop down to 120$ if key < data.
          010E 394
          010E 395
          010E 396 ; return search key > data record/index
          010E 397
          010E 398
50   01   09   9A   010E 399 130$: MOVZBL #1,R0
          09   11   0111 400 BRB     150$      ; search key segment = data record/index segment
          0113 401
          0113 402
          0113 403 ; search key segment = data record/index segment
          0113 404
          0113 405
55   D7   0113 406 140$: DECL   R5      ; decrement number of segments left
03   15   0115 407 BLEQ   145$      ; if none left, exit with match
FF32  31   0117 408 BRW     10$      ; otherwise, try next segment
          011A 409
          011A 410
          011A 411 ; return search key = data record/index key

```

0074 50 D4 011A 412 :
 BA 011C 413 :
 50 D5 0120 414 145\$: CLRL R0
 05 0122 415 150\$: POPR #^M<R2,R4,R5,R6>
 416 TSTL R0
 417 RSB
 0123 418 .DSABL LSB
 0123 419 .END

; setup condition codes

RM3CMPKEY
Symbol table

\$\$PSECT_EP	=	00000000
\$\$RMSTEST	=	0000001A
\$\$RMS_PBUGCHK	=	00000010
\$\$RMS_TBUGCHK	=	00000008
\$\$RMS_UMODE	=	00000004
BN2		000000D1 R 01
BN4		000000DF R 01
BN8		000000FA R 01
IDX\$B_FLAGS	=	0000001C
IDX\$B_KEYREF	=	00000021
IDX\$B_KEYSZ	=	00000020
IDX\$B_NULLCHAR	=	0000001F
IDX\$B_SEGMENTS	=	0000001E
IDX\$V_NULKEYS	=	00000002
IDX\$W_POSITION	=	0000002C
IN2		000000CC R 01
IN4		000000D6 R 01
IN8		000000E8 R 01
NULPAC		000000BF R 01
NULSTR		000000A7 R 01
PAC		000000B0 R 01
RMSCOMPARE_KEY		0000002B RG 01
RMSNULLKEY		00000000 RG 01
STR		00000096 R 01

```
+-----+
! Psect synopsis !
+-----+
```

PSECT name	Allocation	PSECT No.	Attributes	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE
. ABS .	00000000	(0.) 00	(0.) NOPIC USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE
RMSRMS3	00000123	(291.) 01	(1.) PIC USR	CON	REL	GBI	NOSHR	EXE	RD	NOWRT	NOVEC	QUAD
\$ABSS	00000000	(0.) 02	(2.) NOPIC USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE

```
+-----+
! Performance indicators !
+-----+
```

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.08	00:00:00.75
Command processing	112	00:00:00.79	00:00:05.95
Pass 1	151	00:00:02.02	00:00:07.25
Symbol table sort	0	00:00:00.09	00:00:00.12
Pass 2	83	00:00:00.91	00:00:04.19
Symbol table output	3	00:00:00.03	00:00:00.03
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	383	00:00:03.96	00:00:18.32

The working set limit was 1200 pages.

9557 bytes (19 pages) of virtual memory were used to buffer the intermediate code.

There were 10 pages of symbol table space allocated to hold 76 non-local and 21 local symbols.

419 source lines were read in Pass 1, producing 13 object records in Pass 2.

11 pages of virtual memory were used to define 10 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name

\$255\$DUA28:[RMS.OBJ]RMS.MLB;1
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2
TOTALS (all libraries)

Macros defined

2
1
3
6

140 GETS were required to define 6 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LISS:RM3CMPKEY/OBJ=OBJ\$:RM3CMPKEY MSRC\$:RM3CMPKEY/UPDATE=(ENH\$:RM3CMPKEY)+EXECMLS\$/LIB+LIB\$:RMS/LIB

0323 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

