

```

RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSSSSS
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSSSSS
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSSSSS
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
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSS
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSS
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      SSSSSSSSSSSSSS
RRR      RRR      MMM      MMM      SSSSSSSSSSSSSS
RRR      RRR      MMM      MMM      SSSSSSSSSSSSSS

```

— 52

Syn

NTS

NTS
NTS

NTS

NTS
NTS

1994

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS
NTS

NTS

NTS
NTS

1000

NTS

NTS

NTS
NTS

NTS

NTS
NTS

NTS

10

NTS

NTS

NT
NT

NT

NI
PI

10

10

1

10

12

1

1

[illegible]

(2)	116	DECLARATIONS
(3)	148	RM\$EXTRMS - EXIT RMS ROUTINE
(5)	417	RMS\$ECHO - Echo SYS\$INPUT to SYS\$OUTPUT

RMO
Pse

PSE

RMS
\$AB

Pha

Ini
Com
Pas
Sym
Pas
Sym
Pse
Cro
Ass

The
552
The
487
31

Mac

\$2
- \$2
- \$2
TOT
119
The
MAC


```
0000 1          $BEGIN RMOEXTRMS,000,RM$RMS0,<CLEANUP AND RETURN ROUTINE>,<NOWRT,QUAD>
0000 2
0000 3
0000 4 *****
0000 5 *
0000 6 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 *  ALL RIGHTS RESERVED.
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
0000 30
0000 31 Abstract:
0000 32 this routine performs final cleanup and sets
0000 33 status codes appropriately, generates user-level
0000 34 asts as required, and exits rms returning to
0000 35 the user via the exec.
0000 36
0000 37 Environment:
0000 38 star processor running starlet exec.
0000 39
0000 40 Author: L F Laverdure, creation date: 5-JAN-1977
0000 41
0000 42 Modified By:
0000 43
0000 44 V03-015 RAS0307 Ron Schaefer 14-May-1984
0000 45 Save/restore R2, R3, R4 and R5 in RMS$ECHO subroutine
0000 46 as required by the VMS Calling Standard.
0000 47
0000 48 V03-014 JEJ0032 J E Johnson 30-Apr-1984
0000 49 Use the correct form of the bit equates for the async/10
0000 50 rundown test.
0000 51
0000 52 V03-013 RAS0269 Ron Schaefer 19-Mar-1984
0000 53 A little performance boost by re-arranging some code
0000 54 and branches and some instruction optimization.
0000 55 Cope with -1 addr from RM$STALL and
0000 56 give the user a CDA error.
0000 57
```



```
0000 58 : V03-012 DAS0001 David Solomon 24-Jan-1984
0000 59 : Remove code to deallocate EFN. Implement new mechanism for
0000 60 : echoing reads from SYSS$INPUT to SYSS$OUTPUT (declare an AST
0000 61 : in caller's mode here to do the $PUT, instead of returning a
0000 62 : special status to CMODSSDSP and doing the $PUT there).
0000 63 :
0000 64 : V03-011 DGB0002 Donald G. Blair 04-Jan-1984
0000 65 : This routine is executed during EVERY (?) RMS operation and
0000 66 : was found to be critical to performance. Therefore, I've
0000 67 : streamlined the most common execution paths.
0000 68 :
0000 69 : V03-010 KPL0002 Peter Lieberwirth 13-Oct-1983
0000 70 : Use IMP$V_RUH_SYNCH to determine whether to set the asynch
0000 71 : EFN. RUH_SYNCH is set when the recovery unit handler needs
0000 72 : to be notified that an RMS operation it is waiting on has
0000 73 : completed. The handler waits for all user RMS requests to
0000 74 : finish at markpoint, cancel, and end, so that it can then
0000 75 : flush buffers and journal entries. It uses the asynch EFN
0000 76 : to synchronize with the completion of user-inspired RMS
0000 77 : requests.
0000 78 :
0000 79 : V03-009 SHZ00011 Stephen H. Zalewski 1-Sep-1983
0000 80 : Do not call RM$LOWER_LOCK for sequential gets and puts.
0000 81 : (This is a performance enhancement).
0000 82 :
0000 83 : V03-008 KPL0001 Peter Lieberwirth 16-Jun-1983
0000 84 : Add support to write the AT journal record describing a
0000 85 : record operation, if required.
0000 86 :
0000 87 : V03-007 RAS0120 Ron Schaefer 25-Jan-1983
0000 88 : Add support to echo SYSS$INPUT to SYSS$OUTPUT.
0000 89 :
0000 90 : V03-006 KBT0365 Keith B. Thompson 6-Oct-1982
0000 91 : Rewrite this garbage and deallocate the ifab asb with
0000 92 : retblk since it is now a "real" rms structure
0000 93 :
0000 94 : V03-005 KBT0330 Keith B. Thompson 10-Sep-1982
0000 95 : Remove all SO sharing code
0000 96 :
0000 97 : V03-004 LJA0011 Laurie Anderson 02-Sep-1982
0000 98 : Add support for New Context XABs. If XABCXF or XABCXR is
0000 99 : attached to the FAB or RAB (respectively), then fill in
0000 100 : XAB with appropriate information. Uses XAB_SCAN.
0000 101 :
0000 102 : V03-003 JWH0001 Jeffrey W. Horn 01-Sep-1982
0000 103 : Set asynch event flag always if in recovery unit
0000 104 : handler.
0000 105 :
0000 106 : V03-002 KBT0303 Keith B. Thompson 28-Aug-1982
0000 107 : Reorganize psects
0000 108 :
0000 109 : V03-001 RAS0089 Ron Schaefer 7-Jun-1982
0000 110 : Delete reference to incorrect $SQUAD_ALIGN macro;
0000 111 : re-arrange code to accomplish alignment.
0000 112 :
0000 113 :--
0000 114 :
```

```
0000 116      .SBTTL  DECLARATIONS
0000 117
0000 118      :
0000 119      : Include Files:
0000 120      :
0000 121      :
0000 122      :
0000 123      : Macros:
0000 124      :
0000 125
0000 126      $IFBDEF
0000 127      $ASBDEF
0000 128      $IRBDEF
0000 129      $RABDEF
0000 130      $FABDEF
0000 131      $IMPDEF
0000 132      $PIODEF
0000 133      $PSLDEF
0000 134      $RMSDEF
0000 135
0000 136      :
0000 137      : Equated Symbols:
0000 138      :
0000 139
00000100 0000 140 M_RUH SYNCH      =      1@IMP$V_RUH SYNCH
00000010 0000 141 M_IORUNDOWN    =      1@IMP$V_IORUNDOWN
0000 142
0000 143      :
0000 144      : Own Storage:
0000 145      :
0000 146
```



```
0000 148          .SBTTL RM$EXTRMS - EXIT RMS ROUTINE
0000 149
0000 150      :++
0000 151      :
0000 152      : RM$EX_NOSTR
0000 153      : RM$EX_NIRAB_SHR
0000 154      : RM$EX$SUC
0000 155      : RM$EXRMS
0000 156
0000 157      : this routine cleans up rms structures and returns
0000 158      : to the user via the exec. there are three entry points as follows:
0000 159      :
0000 160      : 1. rm$exsuc      exit with success
0000 161      : 2. rm$exrms    exit with status code in r0
0000 162      : 3. rm$ex_nostr exit with status code in r0
0000 163      :                  but without a valid ifab or irab
0000 164      :                  (in this case ap must point to user's argument
0000 165      :                  list and r7 must contain user's mode)
0000 166
0000 167      : the following functions are performed:
0000 168      :
0000 169      : 1. clear busy and restore the argument list pointer
0000 170      : 2. prefix the rms32 facility code to the status
0000 171      :    code (16 bits) in r0 and store in sts
0000 172      : 3. generate a user level completion ast if appropriate
0000 173      : 4. re-enable rms internal asts
0000 174      : 5. return to the user via the exec (ret)
0000 175
0000 176      : Calling sequence:
0000 177      :
0000 178      : entered via jump to the appropriate entry point
0000 179
0000 180      : Input Parameters:
0000 181      :
0000 182      : r0      status code (unless entry at rm$exsuc)
0000 183      :         note: stv must be set already.
0000 184      : r7      caller's mode (only if entry at rm$ex_nostr)
0000 185      : r8      user structure addr (fab or rab)
0000 186      : r9      internal structure addr (ifab or irab)
0000 187      :         (not an input for entry at rm$ex_nostr)
0000 188      : r10     ifab address (if r9 is irab address)
0000 189      : r11     impure area addr
0000 190      : ap      arglist addr (only if entry at rm$ex_nostr)
0000 191
0000 192      : Implicit Inputs:
0000 193      :
0000 194      : none
0000 195
0000 196      : Output Parameters:
0000 197      :
0000 198      : r0      full 32-bit rms status code
0000 199
0000 200      : Implicit Outputs:
0000 201      :
0000 202      : the busy bit for the internal structure is cleared,
0000 203      : sts is set to the status code.
0000 204      :
```

```
0000 205 : Completion Codes:
0000 206 :
0000 207 :         standard rms
0000 208 :
0000 209 : Side Effects:
0000 210 :
0000 211 :         internal rms asts are re-enabled.
0000 212 :
0000 213 :--
0000 214 :
0000 215 :++
0000 216 :
0000 217 :         although the following offsets in the code
0000 218 :         refer to the fab or ifab, the code works equally well
0000 219 :         for the rab and irab as the definitions are equivalent
0000 220 :
0000 221 :--
0000 222 :
0000 223 :         ASSUME  IFB$V_BUSY      EQ  IRB$V_BUSY
0000 224 :         ASSUME  IFB$L_ARGLST    EQ  IRB$L_ARGLST
0000 225 :         ASSUME  IFB$B_MODE      EQ  IRB$B_MODE
0000 226 :         ASSUME  FAB$L_STS       EQ  RAB$L_STS
0000 227 :         ASSUME  FAB$L_STV       EQ  RAB$L_STV
```



```
0000 229 :++
0000 230 :
0000 231 : entry point for rm2conn and rm3conn on errors instead of
0000 232 : to ex_nostr. the irab has been deallocated. Also from disconnect.
0000 233 :
0000 234 : ap = arglist addr
0000 235 : r7 = mode
0000 236 : r10 = ifab
0000 237 :
0000 238 :--
0000 239 :
0000 240 RMSEX_NIRAB_SHR::
59 5A D0 0000 241 MOVE R10,R9 ; put ifab into r9
0003 242 SSB #IFB$V_BUSY,(R10) ; note ifab busy in case of stall
50 DD 0007 243 PUSHL R0 ; save the status
FFF4' 30 0009 244 BSBW RM$LOWER_LOCK ; release file lock
50 8ED0 000C 245 POPL R0 ; restore status
000F 246 CSB #IFB$V_BUSY,(R10) ; clear busy
0013 247 :
0013 248 :++
0013 249 :
0013 250 : entry point if exiting without r9 pointing to a valid internal structure
0013 251 :
0013 252 : ap = arglist addr
0013 253 : r7 = caller's mode
0013 254 :
0013 255 :--
0013 256 :
0013 257 RMSEX_NOSTR::
59 D4 0013 258 CLRL R9 ; make sure R9 is cleared
6B 11 0015 259 BRB SETSTS ; and finish up
0017 260 :
0017 261 :++
0017 262 :
0017 263 : entry point to return a status code of success
0017 264 :
0017 265 :--
0017 266 :
0017 267 RMSEXSUC::
0017 268 RMSSUC
001A 269 :
001A 270 :++
001A 271 :
001A 272 : entry point to clean up the internal structure
001A 273 : by clearing busy and restoring the user's ap
001A 274 :
001A 275 :--
001A 276 :
001A 277 RMSEXRMS::
001A 278 ASSUME <IFB$C_BID&1> EQ 1
001A 279 ASSUME <IRB$C_BID&1> EQ 0
001A 280 ASSUME IFB$B_BID EQ IRB$B_BID
03 08 A9 E9 001A 281 BLBC IFB$B_BID(R9),1$ ; Branch if irab -- R10 already => ifab
5A 59 D0 001E 282 MOVL R9,R10 ; Set R10 to ifab address
78 AA D5 0021 283 1$: TSTL IFB$L_SFBS_PTR(R10) ; Are we sharing this file?
08 13 0024 284 BEQL 3$ ; No, do not bother with rm$lower_lock
50 DD 0026 285 PUSHL R0 ; save the status
```

```
FFD5' 30 0028 286 BSBW RM$LOWER_LOCK ; release file lock
50 8ED0 002B 287 POPL R0 ; restore status
002E 288
002E 289 ASSUME IFB$B_BID EQ IRB$B_BID
002E 290
15 00A0 CA 04 E1 002E 291 3$: BBC #IFB$V_AT,IFB$B_JNLFLG(R10),10$ ; skip if not AT journaling
11 08 A9 E8 0034 292 BLBS IFB$B_BID(R9),10$ ; skip if this is an ifab
50 DD 0038 293 PUSHL R0 ; save status till now
00000000'EF 16 003A 294 JSB RM$AT_JNL_RECORD ; write the AT journal record
03 50 E8 0040 295 ; maintain status from AT_JNL_RECORD
6E 50 D0 0043 296 BLBS R0,5$ ; skip if success
50 8E D0 0046 297 MOVL R0,(SP) ; new status if failure
0049 298 5$: MOVL (SP)+,R0 ; restore status
5C 18 A9 D0 0049 300 10$: MOVL IFB$L_ARGLST(R9),AP ; restore user ap from ifab or irab
004D 301 CSB #IFB$V_BUSY,(R9) ; clear the busy flag in ifab or irab
57 0A A9 9A 0051 302 MOVZBL IFB$B_MODE(R9),R7 ; get user's access mode from ifab/irab
0055 303
0055 304 ;+
0055 305 ; If we are to echo SYS$INPUT to SYS$OUTPUT, then declare AST to do so.
0055 306 ;:-
0055 307
29 08 A9 E8 0055 308 BLBS IFB$B_BID(R9),SETSTS ; done if IFAB
69 22 E1 0059 309 BBC #IRB$V_PPF_IMAGE,(R9),- ; done if not SYS$INPUT
25 005C 310 SETSTS-
69 39 E5 005D 311 BBCC #IRB$V_PPF_ECHO,(R9),- ; done if not wanted
21 0060 312 SETSTS-
54 2A AA 3C 0061 313 MOVZWL IFB$W_ECHO_ISI(R10),R4 ; get basic ISI
1B 13 0065 314 BEQL SETSTS ; done if no ISI
A9 0067 315 BISW3 #FAB$M_PPF_IND+- ; make into indirect isi with CCL
0068 316 <FAB$M-CR&FAB$V_PPF_RAT>-
16 AB 54 4080 8F 0068 317 R4,RAB$W_RFA+6(R8) ; and store in RAB
50 DD 006E 318 PUSHL R0 ; save R0
0070 319 $DCLAST_S - ; declare AST to do the $PUT
0070 320 ASTADR=W^RMS$ECHO,- ; routine address
0070 321 ASTPRM=R8,- ; astprm is user's RAB
0070 322 ACMODE=R7 ; deliver in caller's mode
50 8ED0 007F 323 POPL R0 ; restore R0
0082 324
08 AB 50 D0 0082 325 SETSTS: SSB #16,R0 ; add rms facility code
FF73' 30 0086 326 MOVL R0,FAB$L_STS(R8) ; and store in fab/rab
008A 327 BSBW RM$SETEXTRMS ; Call routine to fill in context XABs
008D 328
008D 329 ;+
008D 330 ; Revalidate the user's argument list and generate a user-level completion
008D 331 ; ast if the user has included the appropriate "ERR=" or "SUC=" parameter.
008D 332 ;:-
008D 333
02 6C 91 0093 334 IFNORD #1,(AP),10$,R7 ; skip ast if no access to arg list
44 1F 0096 335 CMPB (AP),#2 ; completion routine specified?
OC 50 E8 0098 336 BLSSU 10$ ; branch if none
009B 337 BLBS R0,2$ ; branch if RMS operation successful
51 08 BC DE 00A1 338 IFNORD #12,(AP),7$,R7 ; branch if no access to ERR argument
OC 11 00A5 339 MOVAL @8(AP),R1 ; get error completion routine addr
33 13 00A7 340 BRB 5$
00A9 341 2$: BEQL 10$ ; branch if just 2 params
342 ; (implies no success routine)
```



```

      51  0C BC  DE  00A9  343      IFNORD  #16,(AP),7$,R7      ; branch if not readable
      00AF  344      MOVAL  @12(AP),R1      ; get success completion routine addr
      00B3  345
      52  51  27  13  00B3  346  5$:  BEQL  10$      ; ast addr of 0 => 'don't generate ast'
      00B5  347      ADDL3  #1,R1,R2      ; bogus -1 addr from RM$STALL?
      10  13  00B9  348      BEQL  7$      ; give CDA error if so
      00BB  349      $DCLAST_S      ASTADR=(R1),-
      00BB  350      ASTPRM=R8,-
      00BB  351      ACMODE=R7
      0D  50  E8  00C8  352      BLBS  R0,9$      ; branch on error
      00CB  353
      0C A8  08 A8  D0  00CB  354  7$:  MOVL  FAB$L_STS(R8),FAB$L_STV(R8) ; register alternate error status
      08 A8  0001C0E4 8F D0  00D0  355      MOVL  #RMS$_CDA,FAB$L_STS(R8) ; ; store new error status
      50  08 A8  D0  00D8  356  9$:  MOVL  FAB$L_STS(R8),R0      ; restore status code
      00DC  357
      34  6B  01  E5  00DC  358  10$:  BBCC  #IMP$V_AST,(R11),ENBAST
      59  D5  00E0  359      TSTL  R9      ; zero ifab/irab addr?
      08  13  00E2  360      BEQL  SETIOR      ; branch if yes
      00E4  361
      00E4  362
      00E4  363 ; NOTE: The above indicates that we have just deallocated the ifab
      00E4  364 ; or irab in $close or $disconnect.
      00E4  365 ; Assumes async efn must be set.
      00E4  366 ;
      00E4  367 ;
      00E4  368 ;+
      00E4  369 ; Check for setting of async event flag.
      00E4  370 ;-
      00E4  371
      0E  69  23  E1  00E4  372      ASSUME  IRB$V_ASYNC  EQ  IFB$V_ASYNC
      00E4  373      BBC  #IRB$V_ASYNC,(R9),SYNCH      ; branch if not async i/o
      00E8  374
      20  69  24  E5  00E8  375      ASSUME  IRB$V_ASYNCWAIT EQ  IFB$V_ASYNCWAIT
      00E8  376  25$:  BBCC  #IRB$V_ASYNCWAIT,(R9),RETURN      ; branch if no wait issued
      00EC  377      ; else set flag
      00EC  378
      00EC  379      ASSUME  IMP$C_IOREFN  EQ  IMP$C_ASYEFN
      00EC  380
      00EC  381  SETIOR: $SETEF_S      #IMP$C_ASYEFN      ; kick off the stalled wait
      04  00F5  382      RET
      00F6  383
      00F6  384
      0B A9  94  00F6  385  SYNCH:  CLRB  IRB$B_EFN(R9)      ; show no efn allocated
      00F9  386
      00F9  387 ;+
      00F9  388 ; return the ifab's asb (if any) as operation is now complete
      00F9  389 ;-
      00F9  390
      00F9  391      ASSUME  <IFB$C_BID&1>  EQ  1
      00F9  392      ASSUME  <IRB$C_BID&1>  EQ  0
      00F9  393      ASSUME  IFB$B_BID  EQ  IRB$B_BID
      00F9  394
      0F  08 A9  E9  00F9  395      BLBC  IRB$B_BID(R9),RETURN      ; branch if not ifab
      54  14 A9  D0  00FD  396      MOVL  IFB$L_ASBADDR(R9),R4      ; get asb addr
      09  13  0101  397      BEQL  RETURN      ; dont deallocate if not present
      53  5B  D0  0103  398      MOVL  R11,R3      ; free space header at start
      0106  399      ; of this page
```

```

      FEF7' 30 0106 400      BSBW  RM$RETLK      ; free it up
      14 A9 D4 0109 401      CLRL  IFB$$_ASBADDR(R9) ; show no asb
6B  0110 8F B3 010C 402 RETURN: BITW #M_RUH_SYNCH!M_IORUNDOWN,(R11) ; branch if i/o rundown or
      D9 12 0111 403      BNEQ  SETIOR           ; if synching with RU handler
      04 0113 404 EXIT:  RET                    ; exit rms
      0114 405
      0114 406 ;+
      0114 407 ; re-enable asts by clearing the ast inhibit bit
      0114 408 ; and if bit already clear perform an enable asts directive
      0114 409 ;-
      0114 410
F7 00000000'9F 00 E4 0114 411 ENBAST: BBSC #PIO$V_INHAST,@#PIO$GW_STATUS,EXIT ; Enable rms asts
      50 08 A8 D0 011C 412 $SETAST_S #1 ; Enable exec asts
      04 0125 413 MOVL -FAB$_STS(R8),R0 ; restore status code
      0129 414 RET
      012A 415
```



```
012A 417      .SBTTL RMS$ECHO - Echo SYSS$INPUT to SYSS$OUTPUT
012A 418
012A 419 :++
012A 420 :
012A 421 :      This routine echos a $GET from SYSS$INPUT to SYSS$OUTPUT. It is invoked
012A 422 :      as an AST routine in callers mode, declared by RM$EXRMS.
012A 423 :
012A 424 :      A RAB is constructed on the stack, and an asynchronous $PUT of the
012A 425 :      record described in the user's RAB is issued.
012A 426 :
012A 427 :      Calling sequence:
012A 428 :
012A 429 :          CALLED as an AST routine
012A 430 :
012A 431 :      Input Parameters:
012A 432 :
012A 433 :          4(AP)  User RAB address
012A 434 :
012A 435 :      Implicit Inputs:
012A 436 :
012A 437 :          RAB$W_RFA+6      ISI to use for $PUT (w/ppf_ind set and ccl set up).
012A 438 :                          Note that this is currently a spare word.
012A 439 :
012A 440 :      Output Parameters:
012A 441 :
012A 442 :          none
012A 443 :
012A 444 :      Implicit Outputs:
012A 445 :
012A 446 :          The record read from SYSS$INPUT is echoed to SYSS$OUTPUT.
012A 447 :
012A 448 :      Completion Codes:
012A 449 :
012A 450 :          status of $PUT
012A 451 :
012A 452 :      Side Effects:
012A 453 :
012A 454 :          none
012A 455 :
012A 456 :--
003C 012A 457      .ENTRY RMS$ECHO,^M<R2,R3,R4,R5>
012C 459
012C 460 :+
012C 461 :      Allocate and zero a RAB on the stack.
012C 462 :-
012C 463
012C 464      MOVAB  -RAB$C_BLN(SP),SP      ; allocate a RAB on the stack
00 5E      BC AE      9E 0130 465      MOVCS  #0,(SP),#0,-      ; zero the RAB
6E      6E      00 2C      0134 466      #RAB$C_BLN,(SP)      ; (R1 now contains address of RAB)
0138 467
0138 468 :+
0138 469 :      Now set up RAB and do the $PUT.
0138 470 :-
0138 471
0138 472      ASSUME RAB$B_BID      EQ      0
0138 473      ASSUME RAB$B_BLN      EQ      RAB$B_BID+1
```

```
61 50 4401 8F B0 0138 474
      04 AC D0 0138 475
      22 A0 B0 013D 476
      22 A1 B0 0141 477
      28 A0 D0 0144 478
      28 A1 D0 0146 479
      16 A0 B0 0149 480
      02 A1 B0 014B 481
           014E 482
           0150 483
           0155 484
      04 015E 485
           015F 486
           015F 487

      MOVW #<RAB$C_BID+<RAB$C_BLN@8>>,(R1) ; fill in BID and BLN
      MOVL 4(AP),R0 ; get user RAB address
      MOVW RAB$W_RSZ(R0),- ; copy record size
           RAB$W_RSZ(R1) ;
      MOVL RAB$L_RBF(R0),- ; copy record address
           RAB$L_RBF(R1) ;
      MOVW RAB$W_RFA+6(R0),- ; store PPF ISI & ccl
           RAB$W_ISI(R1) ;
      SSB #RAB$V_ASY,RAB$L_ROP(R1) ; specify async operation
      $PUT RAB=R1 ; echo the record to SYS$OUTPUT
      RET ; all done - return

      .END
```


RMOEXTRMS
Symbol table

CLEANUP AND RETURN ROUTINE

B 10

16-SEP-1984 00:20:07 VAX/VMS Macro V04-00
5-SEP-1984 16:21:42 [RMS.SRC]RMOEXTRMS.MAR;1

Page 12
(5)

```

$$PSECT_EP      = 00000000
$$TMP1          = 00000001
$$TMP2          = 00000051
$$RMSTEST       = 0000001A
$$RMS_PBUGCHK   = 00000010
$$RMS_TBUGCHK   = 00000008
$$RMS_UMODE     = 00000004
$$T1            = 00000000
ENBAST          = 00000114 R      01
EXIT            = 00000113 R      01
FAB$$_STS       = 00000008
FAB$$_STV       = 0000000C
FAB$$_CR        = 00000002
FAB$$_PPF_IND   = 00004000
FAB$$_PPF_RAT   = 00000006
IFB$$_BID       = 00000008
IFB$$_JNLFLG    = 000000A0
IFB$$_MODE      = 0000000A
IFB$$_BID       = 0000000B
IFB$$_ARGLST    = 00000018
IFB$$_ASBADDR   = 00000014
IFB$$_SFSB_PTR  = 00000078
IFB$$_ASYNC     = 00000023
IFB$$_ASYNCWAIT = 00000024
IFB$$_AT        = 00000004
IFB$$_BUSY      = 00000020
IFB$$_ECHO_ISI  = 0000002A
IMP$$_ASYEFN    = 0000001E
IMP$$_IOREFN    = 0000001E
IMP$$_AST       = 00000001
IMP$$_IORUNDOWN = 00000004
IMP$$_RUH_SYNCH = 00000008
IRB$$_BID       = 00000008
IRB$$_EFN       = 0000000B
IRB$$_MODE      = 0000000A
IRB$$_BID       = 0000000A
IRB$$_ARGLST    = 00000018
IRB$$_ASYNC     = 00000023
IRB$$_ASYNCWAIT = 00000024
IRB$$_BUSY      = 00000020
IRB$$_PPF_ECHO  = 00000039
IRB$$_PPF_IMAGE = 00000022
M_IORUNDOWN     = 00000010
M_RUH_SYNCH     = 00000100
PIO$$_STATUS    = ***** X      01
PIO$$_INHAST    = 00000000
RAB$$_BID       = 00000000
RAB$$_BLN       = 00000001
RAB$$_BID       = 00000001
RAB$$_BLN       = 00000044
RAB$$_RBF       = 00000028
RAB$$_ROP       = 00000004
RAB$$_STS       = 00000008
RAB$$_STV       = 0000000C
RAB$$_ASY       = 00000000
RAB$$_ISI       = 00000002
RAB$$_RFA       = 00000010

```

```

RAB$$_RSZ      = 00000022
RETURN         = 0000010C R      01
RMSAT_JNL_RECORD ***** X      01
RMSEX_RMS      = 0000001A RG     01
RMSEX_SUC      = 00000017 RG     01
RMSEX_NIRAB_SHR = 00000000 RG     01
RMSEX_NOSTR     = 00000013 RG     01
RMSLOWER_LOCK  ***** X      01
RMSRETBLK      ***** X      01
RMSSETEXTRMS   ***** X      01
RMS$$_ECHO     = 0000012A RG     01
RMS$$_CDA      = 0001C0E4
SETIOR         = 000000EC R      01
SETSTS         = 00000082 R      01
SYNCH          = 000000F6 R      01
SYSDCLAST      ***** GX     01
SYSPUT         ***** GX     01
SYSSSETAST     ***** GX     01
SYSSSETEF      ***** GX     01

```

RMO
Tab

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

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
RMSRMSO	0000015F (351.)	01 (1.)	PIC USR CON REL GBL 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	36	00:00:00.06	00:00:01.79
Command processing	129	00:00:00.70	00:00:06.89
Pass 1	310	00:00:10.03	00:00:30.61
Symbol table sort	0	00:00:01.23	00:00:02.41
Pass 2	94	00:00:02.05	00:00:05.89
Symbol table output	10	00:00:00.12	00:00:00.60
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	583	00:00:14.21	00:00:48.22

The working set limit was 1500 pages.
55233 bytes (108 pages) of virtual memory were used to buffer the intermediate code.
There were 50 pages of symbol table space allocated to hold 1004 non-local and 15 local symbols.
487 source lines were read in Pass 1, producing 17 object records in Pass 2.
31 pages of virtual memory were used to define 29 macros.

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

Macro library name	Macros defined
_\$255\$DUA28:[RMS.OBJ]RMS.MLB;1	12
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	2
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	11
TOTALS (all libraries)	25

1192 GETs were required to define 25 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:RMOEXTRMS/OBJ=OBJ\$:RMOEXTRMS MSRC\$:RMOEXTRMS/UPDATE=(ENH\$:RMOEXTRMS)+EXECML\$/LIB+LIB\$:RMS/LIB

0318 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

