```
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   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
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  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         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$
NT$
NT$
NT$

NT$

NT$
NT$
NT$
NT$
NT$
NT$
NT$


NT$
NT$
NT$
NT$
NT$
NT$
PI
```

```
                                                                      N 9
**FILE**ID**NTOBLKIO
```

```
NN      NN  TTTTTTTTTT    OOOOOO    BBBBBBBB    LL          KK      KK    IIIIII      OOOOOO
NN      NN  TTTTTTTTTT    OOOOOO    BBBBBBBB    LL          KK      KK    IIIIII      OOOOOO
NN      NN      TT      OO      OO  BB      BB  LL          KK      KK      II      OO      OO
NN      NN      TT      OO      OO  BB      BB  LL          KK      KK      II      OO      OO
NNNN    NN      TT      OO    OOOO  BB      BB  LL          KK    KK        II      OO      OO
NNNN    NN      TT      OO    OOOO  BB      BB  LL          KK    KK        II      OO      OO
NN  NN  NN      TT      OO  OO  OO  BBBBBBBB    LL          KKKKKK          II      OO      OO
NN  NN  NN      TT      OO  OO  OO  BBBBBBBB    LL          KKKKKK          II      OO      OO
NN    NNNN      TT      OOOO    OO  BB      BB  LL          KK    KK        II      OO      OO
NN    NNNN      TT      OOOO    OO  BB      BB  LL          KK    KK        II      OO      OO
NN      NN      TT      OO      OO  BB      BB  LL          KK      KK      II      OO      OO      ....
NN      NN      TT      OO      OO  BB      BB  LL          KK      KK      II      OO      OO      ....
NN      NN      TT        OOOOOO    BBBBBBBB    LLLLLLLLLL  KK      KK    IIIIII      OOOOOO        ....
NN      NN      TT        OOOOOO    BBBBBBBB    LLLLLLLLLL  KK      KK    IIIIII      OOOOOO        ....

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
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
NT
Sy

$$
$$
$$
$$
$$
BD
BD
BD
BD
BD
CH
CH
CH
CH
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
DA
```

```
0000          $BEGIN  NTOBLKIO,000,NF$NETWORK,<NETWORK BLOCK I/O>
0000
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000
0000    ;++
0000    ; Facility: RMS
0000    ;
0000    ; Abstract:
0000    ;
0000    ;       This module communicates with the File Access Listener (FAL) at the
0000    ;       remote node to perform read, write, and space block I/O operations.
0000    ;
0000    ; Environment: VAX/VMS, executive mode
0000    ;
0000    ; Author: James A. Krycka,      Creation Date:  18-APR-1978
0000    ;
0000    ; Modified By:
0000    ;
0000    ;       V03-004 JAK0145         J A Krycka      12-APR-1984
0000    ;               Track changes in DAP message building algorithm.
0000    ;
0000    ;       V03-003 JAK0122         J A Krycka      22-AUG-1983
0000    ;               On $WRITE failure, verify that FAL has sent a Status message
0000    ;               before sending a Continue Transfer message to unlock FAL.
0000    ;
0000    ;       V03-002 JAK0116         J A Krycka      29-JUN-1983
0000    ;               Cleanup--remove unused code path.
0000    ;
0000    ;       V03-001 JAK0104         J A Krycka      22-APR-1983
0000    ;               Allow several DATA messages to be blocked in one transmit QIO
0000    ;               for $WRITE in file transfer mode.
0000    ;
0000    ;--
```

```
0000            .SBTTL  DECLARATIONS
0000
0000    ;
0000    ; Include Files:
0000    ;
0000
0000            $BDBDEF                                 ; Define BDB symbols
0000            $DAPPLGDEF                              ; Define DAP prologue symbols
0000            $DAPHDRDEF                              ; Define DAP message header
0000            $DAPCNFDEF                              ; Define DAP Configuration message
0000            $DAPCTLDEF                              ; Define DAP Control message
0000            $DAPDATDEF                              ; Define DAP Data message
0000            $DAPSTSDEF                              ; Define DAP Status message
0000            $IFBDEF                                 ; Define IFAB symbols
0000            $IRBDEF                                 ; Define IRAB symbols
0000            $NWADEF                                 ; Define Network Work Area symbols
0000            $RABDEF                                 ; Define Record Access Block symbols
0000            $RMSDEF                                 ; Define RMS completion codes
0000
0000    ;
0000    ; Macros:
0000    ;
0000            None
0000    ;
0000    ; Equated Symbols:
0000    ;
0000
0000            ASSUME  DAP$Q_DCODE_FLG EQ 0
0000            ASSUME  NWA$Q_FLG EQ 0
0000
0000    ;
0000    ; Own Storage:
0000    ;
0000            None
0000    ;
```

```
NT
Ps

PS
--
.
NF
$A

Ph
--
In
Co
Pa
Sy
Pa
Sy
Ps
Cr
As

Th
68
Th
56
27

Ma
--
_$
_$
TO

14

Th

MA
```

```
0000            .SBTTL  NT$READ - PERFORM NETWORK READ BLOCK FUNCTION
0000
0000    ;++
0000    ; NT$READ - engages in a DAP dialogue with the remote FAL to read the
0000    ;       specified blocks.
0000    ;
0000    ; Calling Sequence:
0000    ;
0000    ;       BSBW    NT$READ
0000    ;
0000    ; Input Parameters:
0000    ;
0000    ;       R4      BDB address
0000    ;       R5      VBN of 1st block for transfer
0000    ;       R8      RAB address
0000    ;       R9      IRAB address
0000    ;       R10     IFAB address
0000    ;       R11     Impure Area address
0000    ;
0000    ; Implicit Inputs:
0000    ;
0000    ;       BDB$L_ADDR
0000    ;       BDB$W_NUMB
0000    ;       BDB$W_SIZE
0000    ;       BDB$L_VBN
0000    ;       DAP$L_CRC_RSLT
0000    ;       DAP$V_DAPCRC
0000    ;       DAP$V_GEQ_V56
0000    ;       IFB$V_SQO
0000    ;       NWA$V_FTM_EOF
0000    ;       NWA$V_FTM_INIT
0000    ;       NWA$V_FTM_STORE
0000    ;
0000    ; Output Parameters:
0000    ;
0000    ;       R0      Status code (RMS)
0000    ;       R1-R3   Destroyed
0000    ;       AP      Destroyed
0000    ;
0000    ; Implicit Outputs:
0000    ;
0000    ;       BDB buffer contents
0000    ;       BDB$W_NUMB
0000    ;       BDB$B_REL_VBN destroyed
0000    ;       DAP$L_CRC_RSLT
0000    ;       NWA$V_FTM_EOF
0000    ;       NWA$V_FTM_INIT cleared
0000    ;       NWA$V_FTM_RETRV
0000    ;       RAB$W_RFA
0000    ;
0000    ; Completion Codes:
0000    ;
0000    ;       Standard RMS completion codes
0000    ;
0000    ; Side Effects:
0000    ;
0000    ;       None
```

```
                          0000      ;
                          0000      ;--
                          0000
                          0000      NT$READ::                                          ; Entry point
                          0000              $TSTPT  NTREAD                             ;
         00F0 8F     BB   0006              PUSHR   #^M<R4,R5,R6,R7>                   ; Save registers
           56   54   D0   000A              MOVL    R4,R6                              ; Copy address of BDB
      57   3C AA     D0   000D              MOVL    IFB$L_NWA_PTR(R10),R7              ; Get address of NWA (and DAP)
           14 A6     B4   0011              CLRW    BDB$W_NUMB(R6)                     ; Zero # bytes in BDB buffer count
                          0014                                                         ; Note: BDB$W_NUMB = BDB$W_SIZE on input
           48 A6     94   0014              CLRB    BDB$B_REL_VBN(R6)                  ; Zero relative VBN to start of buffer
      07 67   1B     E0   0017              BBS     #NWA$V_FTM_STORE,(R7),10$;$READ after $WRITE illegal in FTM
         67   1D     E1   001B              BBC     #NWA$V_FTM_EOF,(R7),-              ; Check for EOF received while in FTM
              06          001E                      READ_LOOP                          ;  from a previous $READ
            00CB     31   001F              BRW     ERREOF                             ; Branch aid
            00C1     31   0022      10$:    BRW     ERRFTM                             ; Branch aid
                          0025
                          0025      ;+
                          0025      ; Start of loop to read next block and append it to the user buffer.
                          0025      ;
                          0025      ; Note: The data access protocol allows only one block to be transferred per
                          0025      ;       block I/O request. Therefore, a multi-block user request is performed
                          0025      ;       via several one-block DAP requests.
                          0025      ;-
                          0025
                          0025      READ_LOOP:                                         ;
      05 6A   2D     E0   0025              BBS     #IFB$V_SQO,(R10),10$               ; Branch if sequential-only specified
         51   04     9A   0029              MOVZBL  #DAP$K_BLK_VBN,R1                  ; Set RAC for DAP message
              0B     11   002C              BRB     READ_SEND_CTL                      ; Join common code
         67   19     E5   002E      10$:    BBCC    #NWA$V_FTM_INIT,(R7),-             ; Branch if no Control message required
              32          0031                      READ_BLOCK                         ;  and turn off single-shot flag
                          0032              $SETBIT #NWA$V_FTM_RETRV,(R7)              ; Set file transfer mode retrieval flag
         51   05     9A   0036              MOVZBL  #DAP$K_BLK_FILE,R1                 ; Set RAC for DAP message
                          0039
                          0039      ;+
                          0039      ; Build and send DAP Control message to partner.
                          0039      ;-
                          0039
                          0039      READ_SEND_CTL:                                     ;
                          0039              $SETBIT #NWA$V_LAST_MSG,(R7)               ; Declare this last message to block
         50   04     D0   003D              MOVL    #DAP$K_CTL_MSG,R0                  ; Get message type value
            FFBD'    30   0040              BSBW    NT$BUILD_HEAD                      ; Construct message header
         85   01     90   0043              MOVB    #DAP$K_GET_READ,(R5)+              ; Store CTLFUNC field
         85   03     90   0046              MOVB    #<<DAP$M_RAC>!-                    ; Store CTLMENU field
                          0049                        <DAP$M_KEY>!-                    ;
                          0049                      0>,(R5)+                           ;
         85   51     90   0049              MOVB    R1,(R5)+                           ; Store RAC field
      50   48 A6     9A   004C              MOVZBL  BDB$B_REL_VBN(R6),R0               ; Get relative VBN to start of buffer
51    1C A6   50     C1   0050              ADDL3   R0,BDB$L_VBN(R6),R1                ; Compute next VBN to request
            FFA8'    30   0055              BSBW    NT$CVT_BN4_IMG                     ; Store KEY as an image field
            FFA5'    30   0058              BSBW    NT$BUILD_TAIL                      ; Finish building message
            FFA2'    30   005B              BSBW    NT$TRANSMIT                        ; Send Control message to FAL
         03 50       E8   005E              BLBS    R0,READ_BLOCK                      ; Branch on success
            008E     31   0061              BRW     EXIT                               ; Branch aid
                          0064
                          0064      ;+
                          0064      ; Receive DAP Data message from partner containing the requested block.
```

```
                              0064  ;-
                              0064
                              0064  READ_BLOCK:                                           ;
                              0064          $SETBIT #DAP$K_DAT_MSG,DAP$L_MSG_MASK(R7)
                              0069                                                        ; Expect response of Data message
             FF94'   30       0069          BSBW    NT$RECEIVE                            ; Read block
             5C 50   E9       006C          BLBC    R0,CHKEOF                             ; Branch on failure
                15   E1       006F          BBC     #DAP$V_DAPCRC,-                       ; Branch if partner does not support
          10 28 A7            0071                  DAP$Q_SYSCAP(R7),10$                  ;  file level CRC checksum
      52     44 A7   7D       0074          MOVQ    DAP$Q_FILEDATA(R7),R2                 ; Put descriptor of block in <R2,R3>
           0000'CF   0B       0078          CRC     W^NT$CRC_TABLE,-                      ; Compute CRC (destroying R0-R3)
             20 A7            007C                  DAP$L_CRC_RSLT(R7),-                  ;  using result of previous CRC
          63    52            007E                  R2,(R3)                               ;  calculation as initial CRC value
          20 A7 50   D0       0080          MOVL    R0,DAP$L_CRC_RSLT(R7)                 ; Store CRC resultant value
      52     44 A7   7D       0084  10$:    MOVQ    DAP$Q_FILEDATA(R7),R2                 ; Put descriptor of block in <R2,R3>
      50     14 A6   3C       0088          MOVZWL  BDB$W_NUMB(R6),R0                     ; Get # bytes already in BDB buffer
    51    52    50   A1       008C          ADDW3   R0,R2,R1                              ; Compute projected total
          16 A6 51   B1       0090          CMPW    R1,BDB$W_SIZE(R6)                     ; Will this overflow BDB buffer?
                05   1B       0094          BLEQU   20$                                   ; Branch if not
  52      16 A6 50   A3       0096          SUBW3   R0,BDB$W_SIZE(R6),R2                  ; Compute # free bytes in BDB buffer
          14 A6 52   A0       009B  20$:    ADDW2   R2,BDB$W_NUMB(R6)                     ; Update byte count in BDB
          63    52   28       009F          MOVC3   R2,(R3),-                             ; Append new block to BDB buffer
          18 B640             00A2                  @BDB$L_ADDR(R6)[R0]                   ;
                              00A5
                              00A5  ;+
                              00A5  ; Receive DAP Status message from partner if we are not in file transfer mode
                              00A5  ; and return record file address of the first block accessed.
                              00A5  ;-
                              00A5
                              00A5  READ_RECV_STS:                                        ;
                              00A5          RMSSUC                                        ; Anticipate success
          12 6A 2D   E0       00A8          BBS     #IFB$V_SQO,(R10),CHK1                 ; Branch if in file transfer mode
          0E 67 24   E1       00AC          BBC     #DAP$V_GEQ_V56,(R7),CHK1; Branch if partner uses DAP before V5.6
                              00B0  ; ***** $SETBIT #DAP$K_STS_MSG,DAP$L_MSG_MASK(R7); Implied for receive
             FF4D'   30       00B0          BSBW    NT$RECEIVE                            ; Obtain status of read request
             3C 50   E9       00B3          BLBC    R0,EXIT                               ; Branch on failure
             48 A6   95       00B6          TSTB    BDB$B_REL_VBN(R6)                     ; Return RFA value to user RAB on
                03   12       00B9          BNEQ    CHK1                                  ;  first pass thru loop as RFA refers
             FF42'   30       00BB          BSBW    NT$RET_RFA                            ;  to the first block read
                              00BE
                              00BE  ;
                              00BE  ; Determine whether or not user block I/O request has been completed.
                              00BE  ;
                              00BE
             14 A6   B1       00BE  CHK1:   CMPW    BDB$W_NUMB(R6),-                      ; Check # bytes received against
             16 A6            00C1                  BDB$W_SIZE(R6)                        ;  # bytes requested
                2D   1E       00C3          BGEQU   EXIT                                  ; Branch if user request satisfied
             48 A6   96       00C5          INCB    BDB$B_REL_VBN(R6)                     ; Update relative VBN for next time thru
              FF5A   31       00C8          BRW     READ_LOOP                             ; Branch to read next block
                              00CB
                              00CB  ;
                              00CB  ; Check for end-of-file.
                              00CB  ;
                              00CB
       827A 8F  50   B1       00CB  CHKEOF: CMPW    R0,#<RMS$_EOF&^XFFFF>                 ; Is it an end-of-file?
                20   12       00D0          BNEQ    EXIT                                  ; Branch if not
          06 6A 2D   E1       00D2          BBC     #IFB$V_SQO,(R10),10$                  ; Branch if not file transfer mode
```

```
                    00D6          $SETBIT #NWA$V_FTM_EOF,(R7)   ; Denote that end-of-file has been
         16    11   00DA          BRB     EXIT                  ;  reached so that EOF status will be
                    00DC                                        ;  returned on next read attempt;
                    00DC                                        ;  also it's an input to NT$CLOSE
      14 A6    B5   00DC  10$:    TSTW    BDB$W_NUMB(R6)        ; If no data was received from FAL
         11    13   00DF          BEQL    EXIT                  ;  then return an EOF condition,
                    00E1          RMSSUC                        ;  else return success with the data
         0C    11   00E4          BRB     EXIT                  ;  (which will cause BDB$L_VBN to be
                    00E6                                        ;  updated on next entry to NT$READ)
                    00E6
                    00E6  ;+
                    00E6  ; Error processing and exit paths for read operation.
                    00E6  ;-
                    00E6
                    00E6  ERRFTM: RMSERR  FTM                   ; Declare file transfer mode error
         05    11   00EB          BRB     EXIT                  ;
                    00ED  ERREOF: RMSERR  EOF                   ; Declare end-of-file
    00F0 8F    BA   00F2  EXIT:   POPR    #^M<R4,R5,R6,R7>      ; Restore registers
               05   00F6          RSB                           ; Exit with RMS code in R0
```

```
00F7          .SBTTL  NT$WRITE - PERFORM NETWORK WRITE BLOCK FUNCTION
00F7
00F7    ;++
00F7    ; NT$WRITE - engages in a DAP dialogue with the remote FAL to write the
00F7    ;       specified blocks.
00F7    ;
00F7    ; Calling Sequence:
00F7    ;
00F7    ;       BSBW    NT$WRITE
00F7    ;
00F7    ; Input Parameters:
00F7    ;
00F7    ;       R4      BDB address
00F7    ;       R5      VBN of 1st block for transfer
00F7    ;       R8      RAB address
00F7    ;       R9      IRAB address
00F7    ;       R10     IFAB address
00F7    ;       R11     Impure Area address
00F7    ;
00F7    ; Implicit Inputs:
00F7    ;
00F7    ;       BDB buffer contents
00F7    ;       BDB$L_ADDR
00F7    ;       BDB$W_NUMB
00F7    ;       BDB$W_SIZE
00F7    ;       BDB$L_VBN
00F7    ;       DAP$L_CRC_RSLT
00F7    ;       DAP$V_DAPCRC
00F7    ;       DAP$V_GEQ_V56
00F7    ;       IFB$V_SQO
00F7    ;       NWA$V_FTM_INIT
00F7    ;       NWA$V_FTM_RETRV
00F7    ;       NWA$Q_BLD
00F7    ;
00F7    ; Output Parameters:
00F7    ;
00F7    ;       R0      Status code (RMS)
00F7    ;       R1-R3   Destroyed
00F7    ;       AP      Destroyed
00F7    ;
00F7    ; Implicit Outputs:
00F7    ;
00F7    ;       BDB$W_NUMB
00F7    ;       BDB$B_REL_VBN destroyed
00F7    ;       DAP$L_CRC_RSLT
00F7    ;       NWA$V_FTM_INTI cleared
00F7    ;       NWA$V_FTM_STORE
00F7    ;       RAB$W_RFA
00F7    ;
00F7    ; Completion Codes:
00F7    ;
00F7    ;       Standard RMS completion codes
00F7    ;
00F7    ; Side Effects:
00F7    ;
00F7    ;       None
00F7    ;
```

```
                          00F7   ;--
                          00F7
                          00F7   NT$WRITE::                                        ; Entry point
                          00F7           $TSTPT  NTWRITE                           ;
          00F0 8F     BB  00FD           PUSHR   #^M<R4,R5,R6,R7>                  ; Save registers
            56   54   D0  0101           MOVL    R4,R6                             ; Copy address of BDB
      57    3C   AA   D0  0104           MOVL    IFB$L_NWA_PTR(R10),R7             ; Get address of NWA (and DAP)
      67    1A        E0  0108           BBS     #NWA$V_FTM_RETRV,(R7),-           ; $WRITE after $READ illegal in FTM
                 DA       010B           ERRFTM                                    ;
            14   A6   B4  010C           CLRW    BDB$W_NUMB(R6)                    ; Zero # bytes in BDB buffer count
                          010F                                                     ; Note: BDB$W_NUMB = BDB$W_SIZE on input
            48   A6   94  010F           CLRB    BDB$B_REL_VBN(R6)                 ; Zero relative VBN to start of buffer
                          0112
                          0112   ;+
                          0112   ; Start of loop to write next block and append it to the user buffer.
                          0112   ;
                          0112   ; Note: The data access protocol allows only one block to be transferred per
                          0112   ;       block I/O request. Therefore, a multi-block user request is performed
                          0112   ;       via several one-block DAP requests.
                          0112   ;-
                          0112
                          0112   WRITE_LOOP:                                       ;
       05 6A     2D   E0  0112           BBS     #IFB$V_SQO,(R10),10$              ; Branch if sequential-only specified
          51     04   9A  0116           MOVZBL  #DAP$K_BLK_VBN,R1                 ; Set RAC for DAP message
                 0B   11  0119           BRB     WRITE_SEND_CTL                    ; Join common code
          67     19   E5  011B   10$:    BBCC    #NWA$V_FTM_INIT,(R7),-            ; Branch if no Control message required
                 2E       011E           WRITE_BLOCK                               ;  and turn off single-shot flag
                          011F           $SETBIT #NWA$V_FTM_STORE,(R7)             ; Set file transfer mode storage flag
          51     05   9A  0123           MOVZBL  #DAP$K_BLK_FILE,R1                ; Set RAC for DAP message
                          0126
                          0126   ;+
                          0126   ; Build and send DAP Control message to partner.
                          0126   ;-
                          0126
                          0126   WRITE_SEND_CTL:                                   ;
          50     04   D0  0126           MOVL    #DAP$K_CTL_MSG,R0                 ; Get message type value
               FED4'  30  0129           BSBW    NT$BUILD_HEAD                     ; Construct message header
          85     04   90  012C           MOVB    #DAP$K_PUT_WRITE,(R5)+            ; Store CTLFUNC field
          85     03   90  012F           MOVB    #<<DAP$M_RAC>!-                   ; Store CTLMENU field
                          0132             <DAP$M_KEY>!-                           ;
                          0132             0>,(R5)+                                ;
          85     51   90  0132           MOVB    R1,(R5)+                          ; Store RAC field
       50 48     A6   9A  0135           MOVZBL  BDB$B_REL_VBN(R6),R0              ; Get relative VBN to start of buffer
  51    1C A6    50   C1  0139           ADDL3   R0,BDB$L_VBN(R6),R1               ; Compute next VBN to request
               FEBF'  30  013E           BSBW    NT$CVT_BN4_IMG                    ; Store KEY as an image field
               FEBC'  30  0141           BSBW    NT$BUILD_TAIL                     ; Finish building message
               FEB9'  30  0144           BSBW    NT$TRANSMIT                       ; Send Control message to FAL
              03 50   E8  0147           BLBS    R0,WRITE_BLOCK                    ; Branch on success
               00BB   31  014A           BRW     EXIT1                             ; Branch on failure
                          014D
                          014D   ;+
                          014D   ; Build and send DAP Data message to partner containing the next block.
                          014D   ;-
                          014D
                          014D   WRITE_BLOCK:                                      ;
       09 6A     2D   E1  014D           BBC     #IFB$V_SQO,(R10),5$               ; Branch if not in file transfer mode
          00CA   C7   B1  0151           CMPW    NWA$W_DAPBUFSIZ(R7),-             ; Allow blocking of DATA messages in
          0410 8F         0155   400                         #<1024+16>              ; transmit QIO if at least two will
            04     1E     0158   401              BGEQU      10$                     ;  fit in the DAP buffer
                          015A   402 5$:          $SETBIT    #NWA$V_LAST_MSG,(R7)    ; Declare this last message to block
         50  08    D0     015E   403 10$:         MOVL       #DAP$K_DAT_MSG,R0       ; Get message type value
            FE9C'  30     0161   404              BSBW       NT$BUILD_HEAD           ; Construct message header
      54   00F4 C7 D0     0164   405              MOVL       NWA$Q_BLD+4(R7),R4      ; Get address of build message buffer
         50  48 A6 9A     0169   406              MOVZBL     BDB$B_REL_VBN(R6),R0    ; Get relative VBN to start of buffer
   51    1C A6  50 C1     016D   407              ADDL3      R0,BDB$L_VBN(R6),R1     ; Compute next VBN to request
            FE8B'  30     0172   408              BSBW       NT$CVT_BN4_IMG          ; Store RECNUM as an image field
            53  55 D0     0175   409              MOVL       R5,R3                   ; Save next byte pointer
         50  14 A6 3C     0178   410              MOVZWL     BDB$W_NUMB(R6),R0       ; Get # bytes already sent from BDB buf
      52  16 A6 50 A3     017C   411              SUBW3      R0,BDB$W_SIZE(R6),R2    ; Compute # bytes remaining to send
          0200 8F 52 B1   0181   412              CMPW       R2,#512                 ; Is it more than one block?
                05 1B     0186   413              BLEQU      20$                     ; Branch if not
      52  0200 8F B0      0188   414              MOVW       #512,R2                 ; Send exactly one block
         14 A6  52 A0     018D   415 20$:         ADDW2      R2,BDB$W_NUMB(R6)       ; Update byte count in BDB for next time
      51   55   54 C3     0191   416              SUBL3      R4,R5,R1                ; Compute # DAP overhead bytes in msg
      55   51   52 C1     0195   417              ADDL3      R2,R1,R5                ; Compute projected size of DAP message
         00CA C7 55 B1    0199   418              CMPW       R5,NWA$W_DAPBUFSIZ(R7)  ; Make sure message will fit in buffer
                63 1A     019E   419              BGTRU      ERRRSZ                  ; Branch if record is too big
         0120 C7 52 7D    01A0   420              MOVQ       R2,NWA$Q_SAVE_DESC(R7)  ; Save descriptor of user block
                24 BB     01A5   421              PUSHR      #^M<R2,R5>              ; Save registers
63     18 B640  52 28     01A7   422              MOVC3      R2,@BDB$L_ADDR(R6)[R0],(R3); Move block into DAP message
                24 BA     01AD   423              POPR       #^M<R2,R5>              ; Restore registers
            55  53 D0     01AF   424              MOVL       R3,R5                   ; Save next byte pointer
            FE4B'  30     01B2   425              BSBW       NT$BUILD_TAIL           ; Finish building message
                15 E1     01B5   426              BBC        #DAP$V_DAPCRC,-         ; Branch if partner does not support
         11 28 A7         01B7   427                         DAP$Q_SYSCAP(R7),30$    ;  file level CRC checksum
      52   0120 C7 7D     01BA   428              MOVQ       NWA$Q_SAVE_DESC(R7),R2  ; Put descriptor of block in <R2,R3>
          0000'CF  0B     01BF   429              CRC        W^NT$CRC_TABLE,-        ; Compute CRC (destroying R0-R3)
            20 A7         01C3   430                         DAP$L_CRC_RSLT(R7),-    ;  using result of previous CRC
            63  52        01C5   431                         R2,(R3)                 ;  calculation as initial CRC value
         20 A7  50 D0     01C7   432              MOVL       R0,DAP$L_CRC_RSLT(R7)   ; Store CRC resultant value
            FE32'  30     01CB   433 30$:         BSBW       NT$TRANSMIT             ; Write block
            23 50  E9     01CE   434              BLBC       R0,CHKSTS               ; Branch on failure
                          01D1   435
                          01D1   436 ;+
                          01D1   437 ; Receive DAP Status message from partner if we are not in file transfer mode
                          01D1   438 ; and return record file address of the first block accessed.
                          01D1   439 ;-
                          01D1   440
                          01D1   441 WRITE_RECV_STS:                                 ;
         12 6A  2D E0     01D1   442              BBS        #IFB$V_SQO,(R10),CHK2   ; Branch if in file transfer mode
         0E 67  24 E1     01D5   443              BBC        #DAP$V_GEQ_V56,(R7),CHK2; Branch if partner uses DAP before V5.6
                          01D9   444 ; *****      $SETBIT    #DAP$K_STS_MSG,DAP$L_MSG_MASK(R7); Implied for receive
            FE24'  30     01D9   445              BSBW       NT$RECEIVE              ; Obtain status of write request
            15 50  E9     01DC   446              BLBC       R0,CHKSTS               ; Branch on failure
            48 A6  95     01DF   447              TSTB       BDB$B_REL_VBN(R6)       ; Return RFA value to user RAB on
               03  12     01E2   448              BNEQ       CHK2                    ;  first pass thru loop as RFA refers
            FE19'  30     01E4   449              BSBW       NT$RET_RFA              ;  to the first block written
                          01E7   450
                          01E7   451 ;
                          01E7   452 ; Determine whether or not user block I/O request has been completed.
                          01E7   453 ;
                          01E7   454
            14 A6  B1     01E7   455 CHK2:        CMPW       BDB$W_NUMB(R6),-        ; Check # bytes transmitted against
            16 A6         01EA   456                         BDB$W_SIZE(R6)          ;  # bytes requested
```

```
        1A   1E  01EC          BGEQU   EXIT1                   ; Branch if user request satisfied
     48 A6   96  01EE          INCB    BDB$B_REL_VBN(R6)       ; Update relative VBN for next time thru
      FF:E   31  01F1          BRW     WRITE_LOOP              ; Branch to write next block
                 01F4
                 01F4  ;+
                 01F4  ; Error processing and exit paths for write operation.
                 01F4  ;-
                 01F4
     30 A7   91  01F4  CHKSTS: CMPB    DAP$B_TYPE(R7),-        ; Branch if failure was not the result
        09       01F7                  #DAP$K_STS_MSG          ;  of Status message returned by FAL
        0E   12  01F8          BNEQ    EXIT1                   ;
        01   BB  01FA          PUSHR   #^M<R0>                 ; Save primary error code
     FE01'   30  01FC          BSBW    NT$RESUME_FAL           ; Tell FAL what to do on write error via
                 01FF                                          ;  interrupt Continue Transfer message
        01   BA  01FF          POPR    #^M<R0>                 ; Restore primary error code
        05   11  0201          BRB     EXIT1                   ;
                 0203  ERRRSZ: RMSERR  RSZ                     ; Invalid record size
   00F0 8F   BA  0208  EXIT1:  POPR    #^M<R4,R5,R6,R7>        ; Restore registers
             05  020C          RSB                             ; Exit with RMS code in R0
```

```
                  020D        .SBTTL  NT$SPACE - PERFORM NETWORK SPACE BLOCK FUNCTION
                  020D
                  020D  ;++
                  020D  ; NT$SPACE - engages in a DAP dialogue with the remote FAL to space the
                  020D  ;         file forward or backward the specified number of blocks.
                  020D  ;
                  020D  ; Calling Sequence:
                  020D  ;
                  020D  ;         BSBW    NT$SPACE
                  020D  ;
                  020D  ; Input Parameters:
                  020D  ;
                  020D  ;         R1      # blocks to space as a signed number
                  020D  ;         R8      RAB address
                  020D  ;         R9      IRAB address
                  020D  ;         R10     IFAB address
                  020D  ;         R11     Impure Area address
                  020D  ;
                  020D  ; Implicit Inputs:
                  020D  ;
                  020D  ;         None
                  020D  ;
                  020D  ; Output Parameters:
                  020D  ;
                  020D  ;         R0      Status code (RMS)
                  020D  ;         R1-R5   Destroyed
                  020D  ;         R6      Actual # blocks spaced as an unsigned number
                  020D  ;         R7      Destroyed
                  020D  ;         AP      Destroyed
                  020D  ;
                  020D  ; Implicit Outputs:
                  020D  ;
                  020D  ;         None
                  020D  ;
                  020D  ; Completion Codes:
                  020D  ;
                  020D  ;         Standard RMS completion codes
                  020D  ;
                  020D  ; Side Effects:
                  020D  ;
                  020D  ;         None
                  020D  ;
                  020D  ;--
                  020D
                  020D  NT$SPACE::                                      ; Entry point
                  020D          $TSTPT  NTSPACE                         ;
           56  D4 0213          CLRL    R6                              ; Zero # blocks spaced
  34 6A    2D  E0 0215          BBS     #IFB$V_SQO,(R10),ERRFTM2; Network space function not allowed
                  0219                                                  ;  if file transfer mode selected
  57    3C AA  D0 0219          MOVL    IFB$L_NWA_PTR(R10),R7   ; Get address of NWA (and DAP)
                  021D
                  021D  ;+
                  021D  ; Build and send DAP Control message to partner.
                  021D  ;-
                  021D
                  021D  SPACE_SEND_CTL:                                 ;
                  021D          $SETBIT #NWA$V_LAST_MSG,(R7)    ; Declare this last message to block
```

```
          50    04   D0 0221          MOVL    #DAP$K_CTL_MSG,R0           ; Get message type value
              FDD9'  30 0224          BSBW    NT$BUILD_HEAD               ; Construct message header
                51   D5 0227          TSTL    R1                          ; Space foreward request?
                05   19 0229          BLSS    10$                         ; Branch if not
          85    11   90 022B          MOVB    #DAP$K_SPACE_FW,(R5)+       ; Set CTLFUNC field for forward space
                06   11 022E          BRB     20$                         ;
          85    12   90 0230  10$:    MOVB    #DAP$K_SPACE_BW,(R5)+       ; Set CTLFUNC field for backward space
          51    51   CE 0233          MNEGL   R1,R1                       ; Make value positive
          85    02   90 0236  20$:    MOVB    #DAP$M_KEY,(R5)+            ; Store CTLMENU field
              FDC4'  30 0239          BSBW    NT$CVT_BN4_IMG              ; Store KEY as an image field
              FDC1'  30 023C          BSBW    NT$BUILD_TAIL               ; Finish building message
              FDBE'  30 023F          BSBW    NT$TRANSMIT                 ; Send Control message to FAL
             0D 50   E9 0242          BLBC    R0,EXIT2                    ; Branch on failure
                        0245
                        0245  ;+
                        0245  ; Receive DAP Status message from partner to obtain actual number of blocks
                        0245  ; spaced.
                        0245  ;-
                        0245
                        0245  SPACE_RECV_STS:                             ;
                        0245  ; ***** $SETBIT #DAP$K_STS_MSG,DAP$L_MSG_MASK(R7); Implied for receive
                        0245                                              ; Expect response of Status message
              FDB8'  30 0245          BSBW    NT$RECEIVE                  ; Receive status of space request
     56     48 A7    D0 0248          MOVL    DAP$L_RECNUM2(R7),R6        ; Get # blocks actually spaced
                        024C                                              ;  as an unsigned number
                     05 024C          RSB                                 ; Exit with RMS code in R0
                        024D
                        024D  ;+
                        024D  ; Error processing and exit paths for space operation.
                        024D  ;-
                        024D
                        024D  ERRFTM2:RMSERR  FTM                         ; Declare file transfer mode error
                     05 0252  EXIT2:  RSB                                 ; Exit with RMS code in R0
                        0253
                        0253          .END                                ; End of module
```

```
$$.PSECT_EP                    = 00000000                 DAP$M_BITCNT                   = 00000008
$$RMSTEST                      = 0000001A                 DAP$M_BLKCNT                   = 00000040
$$RMS_PBUGCHK                  = 00000010                 DAP$M_KEY                      = 00000002
$$RMS_TBUGCHK                  = 00000008                 DAP$M_RAC                      = 00000001
$$RMS_UMODE                    = 00000004                 DAP$M_SEGMENT                  = 00000040
BDB$B_REL_VBN                  = 00000048                 DAP$M_TMP1$                    = 00000008
BDB$L_ADDR                     = 00000018                 DAP$M_TMP2$                    = FFF80000
BDB$L_VBN                      = 0000001C                 DAP$Q_DCODE_FLG                  00000000
BDB$W_NUMB                     = 00000014                 DAP$Q_FILEDATA                   00000044
BDB$W_SIZE                     = 00000016                 DAP$Q_KEY                        00000048
CHK1                             000000BE R      01       DAP$Q_MSG_BUF1                   00000008
CHK2                             000001E7 R      01       DAP$Q_MSG_BUF2                   00000010
CHKEOF                           000000CB R      01       DAP$Q_STX                        00000050
CHKSTS                           000001F4 R      01       DAP$Q_SYSCAP                     00000028
DAP$B_BITCNT                     00000035                 DAP$Q_SYSPEC                     00000038
DAP$B_BLKCNT                     00000056                 DAP$V_DAPCRC                   = 00000015
DAP$B_CTLFUNC                    00000040                 DAP$V_GEQ_V56                  = 00000024
DAP$B_DCODE_FID                  00000019                 DAP$W_BUFSIZ                     00000040
DAP$B_DCODE_MAC                  0000001B                 DAP$W_CTLMENU                    00000044
DAP$B_DCODE_MSG                  0000001A                 DAP$W_DISPLAY2                   00000054
DAP$B_DECVER                     00000047                 DAP$W_PARTNER                    00000006
DAP$B_ECONUM                     00000045                 DAP$W_RFA                        00000042
DAP$B_FILESYS                    00000043                 DAP$W_STSCODE                    00000040
DAP$B_FLAGS                      00000031                 DAP$W_VERSION                    00000004
DAP$B_KRF                        00000047                 ERREOF                           000000ED R      01
DAP$B_LEN256                     00000034                 ERRFTM                           000000E6 R      01
DAP$B_LENGTH                     00000033                 ERRFTM2                          0000024D R      01
DAP$B_OSTYPE                     00000042                 ERRRSZ                           00000203 R      01
DAP$B_RAC                        00000046                 EXIT                             000000F2 R      01
DAP$B_STREAMID                   00000032                 EXIT1                            00000208 R      01
DAP$B_TYPE                       00000030                 EXIT2                            00000252 R      01
DAP$B_USRNUM                     00000046                 IFB$L_NWA_PTR                  = 0000003C
DAP$B_USRVER                     00000048                 IFB$V_SQO                      = 0000002D
DAP$B_VERNUM                     00000044                 NT$BUILD_HEAD                    ********    X   01
DAP$B_X_FIELD                    00000024                 NT$BUILD_TAIL                    ********    X   01
DAP$C_BLN                        000000C0                 NT$CRC_TABLE                     ********    X   01
DAP$K_BLK_FILE                 = 00000005                 NT$CVT_BN4_IMG                   ********    X   01
DAP$K_BLK_VBN                  = 00000004                 NT$READ                          00000000 RG     01
DAP$K_BLN                        000000C0                 NT$RECEIVE                       ********    X   01
DAP$K_CTL_MSG                  = 00000004                 NT$RESUME_FAL                    ********    X   01
DAP$K_DAT_MSG                  = 00000008                 NT$RET_RFA                       ********    X   01
DAP$K_GET_READ                 = 00000001                 NT$SPACE                         0000020D RG     01
DAP$K_PUT_WRITE                = 00000004                 NT$TRANSMIT                      ********    X   01
DAP$K_SEQ_ACC                  = 00000000                 NT$WRITE                         000000F7 RG     01
DAP$K_SPACE_BW                 = 00000012                 NWA$B_ALLXABCNT                  0000011C
DAP$K_SPACE_FW                 = 00000011                 NWA$B_DAP_RAC                    000000C9
DAP$K_STS_MSG                  = 00000009                 NWA$B_FILESYS                    000000C5
DAP$L_CMWA                       00000030                 NWA$B_KEYXABCNT                  0000011D
DAP$L_CRC_RSLT                   00000020                 NWA$B_NETSTRSIZ                  0000016F
DAP$L_DCODE_STS                  00000018                 NWA$B_NODBUFSIZ                  00000168
DAP$L_MSG_MASK                   0000001C                 NWA$B_ORG                        000000C6
DAP$L_RECNUM1                    00000040                 NWA$B_OSTYPE                     000000C4
DAP$L_RECNUM2                    00000048                 NWA$B_RFM                        000000C7
DAP$L_ROP                        00000050                 NWA$B_RMS_RAC                    000000C8
DAP$L_SSPWA                      00000080                 NWA$C_BLN                        00000800
DAP$L_STV                        0000004C                 NWA$K_BLN                        00000800
DAP$L_TEMP                       00000090                 NWA$L_ALLXABADR                  00000100
```

```
                                                         C 11
NTOBLKIO                        NETWORK BLOCK I/O                          15-SEP-1984 23:50:03  VAX/VMS Macro V04-00        Page  14
Symbol table                                                                5-SEP-1984 16:20:16  [RMS.SRC]NTOBLKIO.MAR;1           (5)

NWA$L_DATXABADR                   00000104            PIO$A_TRACE                   ********    X    01
NWA$L_DEV                         000000C0            READ_BLOCK                    00000064 R       01
NWA$L_FHCXABADR                   00000108            READ_LOOP                     00000025 R       01
NWA$L_KEYXABADR                   0000010C            READ_RECV_STS                 000000A5 R       01
NWA$L_MSG_MASK                    000000D4            READ_SEND_CTL                 00000039 R       01
NWA$L_PROXABADR                   00000110            RMS$_EOF                    = 0001827A
NWA$L_RDTXABADR                   00000114            RMS$_FTM                    = 000187C4
NWA$L_SAVE_FLGS                   00000128            RMS$_RSZ                    = 000186A4
NWA$L_SUMXABADR                   00000118            SPACE_RECV_STS                00000245 R       01
NWA$L_THREAD                      000000FC            SPACE_SEND_CTL                0000021D R       01
NWA$L_XLTATTR                     00000238            TPT$L_NTREAD                  ********    X    01
NWA$L_XLTBUFFLG                   0000022C            TPT$L_NTSPACE                 ********    X    01
NWA$L_XLTCNT                      00000228            TPT$L_NTWRITE                 ********    X    01
NWA$L_XLTMAXINDX                  00000234            WRITE_BLOCK                   0000014D R       01
NWA$L_XLTSIZ                      00000230            WRITE_LOOP                    00000112 R       01
NWA$Q_ACS                         00000244            WRITE_RECV_STS                000001D1 R       01
NWA$Q_BIGBUF                      00000170            WRITE_SEND_CTL                00000126 R       01
NWA$Q_BLD                         000000F0
NWA$Q_FLG                         00000000
NWA$Q_INODE                       0000025C
NWA$Q_IOSB                        000000D8
NWA$Q_LNODE                       00000160
NWA$Q_LOGNAME                     0000023C
NWA$Q_NCB                         00000264
NWA$Q_RCV                         000000E0
NWA$Q_SAVE_DESC                   00000120
NWA$Q_XLTBUF1                     0000024C
NWA$Q_XLTBUF2                     00000254
NWA$Q_XMT                         000000E8
NWA$T_ACSBUF                      0000026C
NWA$T_AUXBUF                      000005E0
NWA$T_DAP                         00000000
NWA$T_INODEBUF                    000004AC
NWA$T_ITM_ATTR                    00000200
NWA$T_ITM_END                     00000224
NWA$T_ITM_LST                     00000200
NWA$T_ITM_MAXINDX                 00000218
NWA$T_ITM_STRING                  0000020C
NWA$T_NCBBUF                      0000052C
NWA$T_NODEBUF                     00000169
NWA$T_RCVBUF                      000001A0
NWA$T_SCAN                        00000100
NWA$T_TEMP                        00000120
NWA$T_XLTBUF1                     000002AC
NWA$T_XLTBUF2                     000003AC
NWA$T_XMTBUF                      000003C0
NWA$V_FTM_EOF                   = 0000001D
NWA$V_FTM_INIT                  = 00000019
NWA$V_FTM_RETRV                 = 0000001A
NWA$V_FTM_STORE                 = 0000001B
NWA$V_LAST_MSG                  = 00000000
NWA$W_BUILD                       000000D2
NWA$W_DAPBUFSIZ                   000000CA
NWA$W_DIR_OFF                     000000CC
NWA$W_DISPLAY                     000000D0
NWA$W_FIL_OFF                     000000CE
NWA$W_JNLXABJOP                   0000011E
```

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

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
NF$NETWORK                      00000253  (  595.)  01 (  1.)    PIC   USR   CON   REL   GBL NOSHR   EXE   RD   NOWRT NOVEC BYTE
$ABS$                           00000800  ( 2048.)  02 (  2.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE

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

Phase                      Page faults    CPU Time       Elapsed Time
-----                      -----------    --------       ------------
Initialization                      32    00:00:00.09    00:00:00.75
Command processing                 114    00:00:00.67    00:00:03.64
Pass 1                             342    00:00:12.96    00:00:29.71
Symbol table sort                    0    00:00:01.69    00:00:02.93
Pass 2                             111    00:00:02.51    00:00:06.19
Symbol table output                 23    00:00:00.17    00:00:00.81
Psect synopsis output                2    00:00:00.03    00:00:00.03
Cross-reference output               0    00:00:00.00    00:00:00.00
Assembler run totals               626    00:00:18.12    00:00:44.07

The working set limit was 1350 pages.
68178 bytes (134 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1202 non-local and 19 local symbols.
568 source lines were read in Pass 1, producing 15 object records in Pass 2.
27 pages of virtual memory were used to define 26 macros.

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

Macro library name                        Macros defined
------------------                        --------------
_$255$DUA28:[RMS.OBJ]RMS.MLB;1                  18
_$255$DUA28:[SYSLIB]STARLET.MLB;2                4
TOTALS (all libraries)                          22

1418 GETS were required to define 22 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:NTOBLKIO/OBJ=OBJ$:NTOBLKIO MSRC$:NTOBLKIO/UPDATE=(ENH$:NTOBLKIO)+LIB$:RMS/LIB
```

0315 AH-BT13A-SE DIGITAL EQUIPMENT CORPORATION
VAX/VMS V4.0 CONFIDENTIAL AND PROPRIETARY