```
RRRRRRRRRRR      MMM          MMM      SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSSSSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRRRRRRRRRR      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  SSSSSSSSSSS
RRR       RRR    MMM          MMM               SSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
```

```
NN       NN  TTTTTTTTTT    000000    DDDDDDDD    EEEEEEEEEE    CCCCCCCC    000000    DDDDDDDD    EEEEEEEEEE
NN       NN  TTTTTTTTTT    000000    DDDDDDDD    EEEEEEEEEE    CCCCCCCC    000000    DDDDDDDD    EEEEEEEEEE
NN       NN      TT       00    00  DD      DD  EE          CC          00    00  DD      DD  EE
NN       NN      TT       00    00  DD      DD  EE          CC          00    00  DD      DD  EE
NNNN     NN      TT       00    00  DD      DD  EE          CC          00    00  DD      DD  EE
NNNN     NN      TT       00    00  DD      DD  EE          CC          00    00  DD      DD  EE
NN  NN   NN      TT       00 00 00  DD      DD  EEEEEEEE    CC          00    00  DD      DD  EEEEEEEE
NN  NN   NN      TT       00 00 00  DD      DD  EEEEEEEE    CC          00    00  DD      DD  EEEEEEEE
NN    NNNN       TT       0000  00  DD      DD  EE          CC          00    00  DD      DD  EE
NN    NNNN       TT       0000  00  DD      DD  EE          CC          00    00  DD      DD  EE
NN       NN      TT       00    00  DD      DD  EE          CC          00    00  DD      DD  EE                ....
NN       NN      TT       00    00  DD      DD  EE          CC          00    00  DD      DD  EE                ....
NN       NN      TT         000000    DDDDDDDD    EEEEEEEEEE    CCCCCCCC    000000    DDDDDDDD    EEEEEEEEEE  ....
NN       NN      TT         000000    DDDDDDDD    EEEEEEEEEE    CCCCCCCC    000000    DDDDDDDD    EEEEEEEEEE  ....

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

```
0000     1         $BEGIN  NTODECODE,000,NH$NETWORK,<DECODE DAP MESSAGE>
0000     2
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28
0000    29 ;++
0000    30 ; Facility: RMS
0000    31 ;
0000    32 ; Abstract:
0000    33 ;
0000    34 ;       This module decodes (parses) the next DAP message and stores the
0000    35 ;       validated fields in the DAP control block.
0000    36 ;
0000    37 ; Environment: VAX/VMS, executive mode
0000    38 ;
0000    39 ; Author: James A. Krycka,      Creation Date:  16-JUN-1977
0000    40 ;
0000    41 ; Modified By:
0000    42 ;
0000    43 ;       V03-007 JEJ0046         J E Johnson     05-Jul-1984
0000    44 ;               Since resultant name strings can be up to 255 bytes long
0000    45 ;               make sure to use that as a valid length limit check.
0000    46 ;
0000    47 ;       V03-006 JEJ0019         J E Johnson     27-Mar-1984
0000    48 ;               Alter CECK_OPERATING_SYSTEM to use DAP$V_P_OS as the
0000    49 ;               P/OS flag due to naming conflict with DAP$V_POS magtape
0000    50 ;               positioning flag. Also use DAP$K_P_OS.
0000    51 ;
0000    52 ;       V03-005 JAK0124         J A Krycka      06-SEP-1983
0000    53 ;               Update new DAP$Q_DCODE_FLG status bits during parse of
0000    54 ;               Configuration message.
0000    55 ;
0000    56 ;       V03-004 JAK0113         J A Krycka      22-JUN-1983
0000    57 ;               Continuation of support for DAP V7 0 specification.
```

```
0000     58 ;                  Add support for 64-bit binary keys.
0000     59 ;                  Also, set DAP$V_VMS_XPFn flags as appropriate.
0000     60 ;
0000     61 ;          V03-003 KRM0108         K Malik         12-May-1983
0000     62 ;                  Update to support DAP V7.0 specification.
0000     63 ;
0000     64 ;          V03-002 KRM0089         K Malik         18-Mar-1983
0000     65 ;                  Add support for STMLF and  STMCR file formats.
0000     66 ;                  Also, set DAP$V_GEQ_V70 bit as appropriate.
0000     67 ;
0000     68 ;--
```

```
                     0000     70          .SBTTL  DECLARATIONS
                     0000     71
                     0000     72 ;
                     0000     73 ; Include Files:
                     0000     74 ;
                     0000     75
                     0000     76          $DAPPLGDEF                      ; Define DAP prologue symbols
                     0000     77          $DAPHDRDEF                      ; Define DAP message header
                     0000     78          $DAPSSPDEF                      ; Define DAP system specific field
                     0000     79          $DAPCNFDEF                      ; Define DAP Configuration message
                     0000     80          $DAPATTDEF                      ; Define DAP Attributes message
                     0000     81          $DAPACCDEF                      ; Define DAP Access message
                     0000     82          $DAPCTLDEF                      ; Define DAP Control message
                     0000     83          $DAPCONDEF                      ; Define DAP Continue Transfer message
                     0000     84          $DAPACKDEF                      ; Define DAP Acknowledge message
                     0000     85          $DAPCMPDEF                      ; Define DAP Access Complete message
                     0000     86          $DAPDATDEF                      ; Define DAP Data message
                     0000     87          $DAPSTSDEF                      ; Define DAP Status message
                     0000     88          $DAPKEYDEF                      ; Define DAP Key Definition message
                     0000     89          $DAPALLDEF                      ; Define DAP Allocation message
                     0000     90          $DAPSUMDEF                      ; Define DAP Summary message
                     0000     91          $DAPTIMDEF                      ; Define DAP Date and Time message
                     0000     92          $DAPPRODEF                      ; Define DAP Protection message
                     0000     93          $DAPNAMDEF                      ; Define DAP Name message
                     0000     94          $DAPFIDDEF                      ; Define DAP field ID symbols
                     0000     95          $FWADEF                         ; Define FWA symbols
                     0000     96
                     0000     97 ;
                     0000     98 ; Macros:
                     0000     99 ;
                     0000    100 ;       See next page for local macro definitions.
                     0000    101 ;
                     0000    102 ; Equated Symbols:
                     0000    103 ;
                     0000    104
00000000             0000    105 K_EXT=0                                 ; Extensible field format
00000001             0000    106 K_FIX=1                                 ; Fixed length field format
00000002             0000    107 K_IMG=2                                 ; Image field format
00000003             0000    108 K_ROM=3                                 ; Rest-of-message field format
                     0000    109
00000004             0000    110 V_DESC=4                                ; Store descriptor of source field
00000005             0000    111 V_TRUNC=5                               ; Truncate source field if necessary
00000006             0000    112 V_SRCR3=6                               ; Source field size in R3
                     0000    113                                         ;  (applicable only if K_FIX specified)
                     0000    114
00000010             0000    115 M_DESC=<1@V_DESC>                       ; Mask for V_DESC
00000020             0000    116 M_TRUNC=<1@V_TRUNC>                     ; Mask for V_TRUNC
00000040             0000    117 M_SRCR3=<1@V_SRCR3>                     ; Mask for V_SRCR3
                     0000    118
                     0000    119          ASSUME  DAP$Q_DCODE_FLG EQ 0
                     0000    120
                     0000    121 ;
                     0000    122 ; Own Storage:
                     0000    123 ;
                     0000    124 ;       None
                     0000    125 ;
```

```
0000   127          .SBTTL  LOCAL MACRO DEFINITIONS
0000   128
0000   129 ;++
0000   130 ; STORE_FIELD obtains the next field (if any) from the DAP message being parsed,
0000   131 ; converts it to an appropriate format, and stores the result in the designated
0000   132 ; field of the DAP control block. The arguments (coded in-line) are:
0000   133 ;
0000   134 ;       NAME   = the symbolic name of DAP field used to generate symbolic DAP
0000   135 ;                control block offset and field ID values.
0000   136 ;       SIZE   = the size in bytes of designated field in DAP control block.
0000   137 ;       FORMAT= the format or structure of the source field. Choices are:
0000   138 ;                K_EXT = extensible field (bit7 of each byte is used to signify
0000   139 ;                        termination/continuation (0/1) of the field).
0000   140 ;                K_FIX = fixed length field.
0000   141 ;                K_IMG = image field (counted string).
0000   142 ;                K_ROM = rest-of-message is taken as the next field.
0000   143 ;       MASK   = the flags to control field processing:
0000   144 ;                M_DESC= store only descriptor of the source field.
0000   145 ;                M_TRUNC=truncate extra bytes if SRC field size is larger than
0000   146 ;                        DST field size (instead of declaring an error).
0000   147 ;                M_SRCR3=size of source field is given in R3 (applicable only if
0000   148 ;                        K_FIX is also specified).
0000   149 ;--
0000   150
0000   151          .MACRO  STORE_FIELD NAME,SIZE=1,FORMAT=1,MASK=0
0000   152          BSBW    STORE_FIELD
0000   153          .BYTE   SIZE
0000   154 TMP1..=.
0000   155          .IIF EQ <SIZE-1>,       .BYTE   DAP$B_'NAME
0000   156          .IIF EQ <SIZE-2>,       .BYTE   DAP$W_'NAME
0000   157          .IIF EQ <SIZE-4>,       .BYTE   DAP$L_'NAME
0000   158          .IIF EQ <SIZE-6>,       .BYTE   DAP$W_'NAME
0000   159          .IIF EQ <SIZE-8>,       .BYTE   DAP$Q_'NAME
0000   160 TMP2..=.
0000   161          .IIF EQ <TMP2..-TMP1..>,.ERROR  ;***** invalid field size *****;
0000   162          .BYTE   DAP$_'NAME
0000   163          .BYTE   FORMAT!MASK
0000   164          .ENDM   STORE_FIELD
0000   165
0000   166 ;++
0000   167 ; CHECK_MASKS examines the designated field of the DAP control block for
0000   168 ; invalid and unsupported bits se.. The arguments (coded in-line) are:
0000   169 ;
0000   170 ;       NAME   = the symbolic name of the DAP field used to generate symbolic
0000   171 ;                invalid and unsupported mask values.
0000   172 ;       SIZE   = the size in bytes of designated field in the DAP control block.
0000   173 ;--
0000   174
0000   175          .MACRO  CHECK_MASKS NAME,SIZE=1
0000   176          BSBW    CHECK_MASKS
0000   177          .BYTE   SIZE
0000   178 TMP1..=.
0000   179          .IIF EQ <SIZE-1>,       .BYTE   DAP$K_'NAME'_I,DAP$K_'NAME'_U
0000   180          .IIF EQ <SIZE-2>,       .WORD   DAP$K_'NAME'_I,DAP$K_'NAME'_U
0000   181          .IIF EQ <SIZE-4>,       .LONG   DAP$K_'NAME'_I,DAP$K_'NAME'_U
0000   182 TMP2..=.
0000   183          .IIF EQ <TMP2..-TMP1..>,.ERROR  ;***** invalid field size *****;
```

```
0000   184          .ENDM   CHECK_MASKS
```

```
                        .SBTTL  NT$DECODE_MSG - DECODE DAP MESSAGE

                ;++
                ; NT$DECODE_MSG - is responsible for parsing a DAP message into its constituent
                ;       fields, storing these field values into corresponding fields in the
                ;       DAP control block, and finally performing validity checks on the
                ;       contents of the converted fields to screen out invalid and unsupported
                ;       bit options or field values.
                ;
                ;       Each DAP message logically consists of two parts:
                ;       (1) a message header (called the operator field in DAP).
                ;       (2) a message body (called the operand field in DAP).
                ;       In addition, the message header may optionally contain a system
                ;       specific field for use by homogeneous systems which is treated as a
                ;       mini-message with discrete fields.
                ;
                ; Calling Sequence:
                ;
                ;       CALLS   #1,NT$DECODE_MSG
                ;
                ; Input Parameters:
                ;
                ;       4(AP)   Address of DAP control block
                ;
                ; Implicit Inputs:
                ;
                ;       None
                ;
                ; Output Parameters:
                ;
                ;       R0      Status code
                ;       R1      Destroyed
                ;
                ; Implicit Outputs:
                ;
                ;       Various fields of the DAP control block are updated.
                ;
                ; Completion Codes:
                ;
                ;       DAP$L_DCODE_STS is returned in R0 where bit 0 indicates success/failure.
                ;
                ; Side Effects:
                ;
                ;       None
                ;
                ;--

                        .ENTRY  NT$DECODE_MSG,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                                                ; Entry point

                ; Perform initialization.
                ;

59   04 AC      D0          MOVL    4(AP),R9                    ; Get address of DAP control block
18 A9   01      D0          MOVL    #1,DAP$L_DCODE_STS(R9)      ; Assume successful parse
5A   08 A9      7D          MOVQ    DAP$Q_MSG_BUF1(R9),R10      ; R10 = size of message
```

```
                     000E    243                                                ; R11 = address of start-of-message
          10 A9  5A  7D  000E    244          MOVQ    R10,DAP$Q_MSG_BUF2(R9)    ; Store in result descriptor
             5A  5B  C0  0012    245          ADDL2   R11,R10                   ; R10 = address of end-of-message + 1
                         0015    246          $ZERO_FILL-                       ; Zero current message work area
                         0015    247                  DST=DAP$L_CMWA(R9)-       ;  in DAP control block
                         0015    248                  SIZE=#DAP$K_CMWA          ;
                24 A9  94  001F    249          CLRB    DAP$B_X_FIELD(R9)         ; Clear 'explicitly specified' flags
                         0022    250
                         0022    251 ;+
                         0022    252 ; Note the current status of registers R8-R11:
                         0022    253 ;
                         0022    254 ;       R8      Currently undefined; later it will be used to contain the
                         0022    255 ;               address of the routine to execute on reaching end-of-message
                         0022    256 ;       R9      Address of DAP control block
                         0022    257 ;       R10     Address of end-of-message-buffer + 1; later it will contain
                         0022    258 ;               the address of end-of-message + 1
                         0022    259 ;       R11     Address of start-of-message; it will be continually updated
                         0022    260 ;               to contain the address of the next byte to parse
                         0022    261 ;-
```

```
                     0022   263          .SBTTL  HEADER - DECODE MESSAGE HEADER
                     0022   264
                     0022   265 ;++
                     0022   266 ; Decode the header of the DAP message (operator portion of the message).
                     0022   267 ; Then dispatch on message type to parse the body of the DAP message (operand
                     0022   268 ; portion of the message).
                     0022   269 ;--
                     0022   270
                     0022   271 HEADER:                                  ; Continuation of mainline
                     0022   272
                     0022   273          ASSUME  DAP$K_CNF_MSG EQ 1
                     0022   274          ASSUME  DAP$K_ATT_MSG EQ 2
                     0022   275          ASSUME  DAP$K_ACC_MSG EQ 3
                     0022   276          ASSUME  DAP$K_CTL_MSG EQ 4
                     0022   277          ASSUME  DAP$K_CON_MSG EQ 5
                     0022   278          ASSUME  DAP$K_ACK_MSG EQ 6
                     0022   279          ASSUME  DAP$K_CMP_MSG EQ 7
                     0022   280          ASSUME  DAP$K_DAT_MSG EQ 8
                     0022   281          ASSUME  DAP$K_STS_MSG EQ 9
                     0022   282          ASSUME  DAP$K_KEY_MSG EQ 10
                     0022   283          ASSUME  DAP$K_ALL_MSG EQ 11
                     0022   284          ASSUME  DAP$K_SUM_MSG EQ 12
                     0022   285          ASSUME  DAP$K_TIM_MSG EQ 13
                     0022   286          ASSUME  DAP$K_PRO_MSG EQ 14
                     0022   287          ASSUME  DAP$K_NAM_MSG EQ 15
                     0022   288
                     0022   289 ;
                     0022   290 ; For optional fields, apply default values as appropriate.
                     0022   291 ;
                     0022   292
3C A9   69   7E      0022   293          MOVAQ   (R9),DAP$Q_SYSPEC+4(R9) ; Initialize descriptor
                     0026   294
                     0026   295 ;
                     0026   296 ; Process the DAP message type field (required).
                     0026   297 ;
                     0026   298
58   0846'CF   9E    0026   299          MOVAB   W^ERROR_FORMAT,R8       ; Specify transfer address on EOM
                     002B   300          STORE_FIELD     TYPE,1,K_FIX    ; Save type field
          66   95    0032   301          TSTB    (R6)                    ; Test for valid value
          0B   13    0034   302          BEQL    10$                     ; Branch if out-of-range
     0F   66   91    0036   303          CMPB    (R6),#DAP$K_NAM_MSG     ; Test for valid value
          06   1A    0039   304          BGTRU   10$                     ; Branch if out-of-range
1A A9     66   90    003B   305          MOVB    (R6),DAP$B_DCODE_MSG(R9); Return message type in status code
          03   11    003F   306          BRB     20$                     ; Continue
        0811   31    0041   307 10$:     BRW     ERROR_INVALID           ; Branch aid
                     0044   308
                     0044   309 ;+
                     0044   310 ; Process the DAP message flags field (required for most messages).
                     0044   311 ; This is a combination menu and bit option field whereby each bit set denotes
                     0044   312 ; that either an associated field is included in the message or a message
                     0044   313 ; option is specified.
                     0044   314 ;
                     0044   315 ; Note: If no flags field is found (i.e., its a one-byte message), the
                     0044   316 ;       associated operand parse routine for the message will still be entered
                     0044   317 ;       (via DISPATCH_TABLE) to determine if the message is valid and to apply
                     0044   318 ;       operand field default values.
                     0044   319 ;-
```

```
                                0044    320
                                0044    321          ASSUME  DAP$V_STREAMID+1 EQ DAP$V_LENGTH
                                0044    322          ASSUME  DAP$V_LENGTH+1 EQ DAP$V_LEN256
                                0044    323          ASSUME  DAP$V_LEN256+1 EQ DAP$V_BITCNT
                                0044    324          ASSUME  DAP$V_BITCNT+2 EQ DAP$V_SYSPEC
                                0044    325          ASSUME  DAP$V_SYSPEC+1 EQ DAP$V_SEGMENT
                                0044    326
        58   00C8'CF   9E       0044    327 20$:     MOVAB   W^DISPATCH_TABLE,R8     ; Specify transfer address on EOM
                                0049    328          STORE_FIELD     FLAGS,1,K_EXT   ; Save flags field
                                0050    329          CHECK_MASKS     FLAGS,1         ; Validate bit options
        58   0846'CF   9E       0056    330          MOVAB   W^ERROR_FORMAT,R8       ; Specify transfer address on EOM
             5C   66   9A       005B    331          MOVZBL  (R6),AP                 ; Copy menu to scratch register
                                005E    332 HDR_LOOP:                               ;
  50    5C   07   00   EA       005E    333          FFS     #0,#DAP$V_SEGMENT+1,AP,R0; Get position of next bit set
                                0063    334          $CLRBIT R0,AP                   ; Clear menu bit just found
                F4   AF   9F    0067    335          PUSHAB  B^HDR_LOOP              ; Push return address on stack
                                006A    336          $CASEB  SELECTOR=R0-            ; Next field/option:
                                006A    337                  DISPL=<-               ;
                                006A    338                      10$-               ;  STREAMID
                                006A    339                      20$-               ;  LENGTH
                                006A    340                      30$-               ;  LEN256
                                006A    341                      ERROR_UNSUPPORT-   ;  BITCNT
                                006A    342                      ERROR_FORMAT-      ;  Reserved
                                006A    343                      60$-               ;  SYSPEC
                                006A    344                      ERROR_UNSUPPORT-   ;  SEGMENT
                                006A    345                  >                      ;
                   4A   11      007C    346          BRB     DISPATCH_TABLE          ; Message header syntax is correct
                                007E    347
                                007E    348 ;
                                007E    349 ; Process each field/option specified in the menu (optional).
                                007E    350 ;
                                007E    351
                                007E    352 10$:     STORE_FIELD     STREAMID,1,K_FIX; Save data stream identification field
              66   95           0085    353          TSTB    (R6)                    ; Currently, multi-streams are
                                0087    354                                          ;  not supported, so check value
              3C   12           0087    355          BNEQ    HDR_UNSUPPORT           ; Branch on error
                   05           0089    356          RSB                             ;
                                008A    357 20$:     STORE_FIELD     LENGTH,1,K_FIX  ; Save length field
        08    5C   02   E1       0091    358          BBC     #DAP$V_LEN256,AP,35$    ; Branch if length value in header is
                                0095    359                                          ;  expressed in one byte (i.e., there
                   05           0095    360          RSB                             ;  is no LEN256 field present)
                                0096    361 30$:     STORE_FIELD     LEN256,1,K_FIX  ; Save length extension field
                                009D    362
                                009D    363 ;
                                009D    364 ; Determine end-of-message based on operand length value in message header.
                                009D    365 ;
                                009D    366
                                009D    367          ASSUME  DAP$B_LENGTH+1 EQ DAP$B_LEN256
                                009D    368
          50    33   A9   3C     009D    369 35$:     MOVZWL  DAP$B_LENGTH(R9),R0     ; Get operand length value
    51    5B    50   C1           00A1    370          ADDL3   R0,R11,R1               ; Compute new end-of-message + 1 address
          5A    51   D1           00A5    371          CMPL    R1,R10                  ; Error if not enough bytes in buffer
                18   1A           00A8    372          BGTRU   HDR_INVALID             ;  to contain message
          5A    51   D0           00AA    373          MOVL    R1,R10                  ; Update end-of-message address
                   05           00AD    374          RSB                             ;
                                00AE    375
                                00AE    376 ;+
```

```
                     00AE    377 ; Suggested code to support the BITCNT field is shown below.
                     00AE    378 ;
                     00AE    379 ;40$:    STORE_FIELD     BITCNT,1,K_FIX  ; Save bit count field
                     00AE    380 ;        CMPB    DAP$B_TYPE(R9),-        ; BITCNT field allowed only in
                     00AE    381 ;                #DAP$K_DAT_MSG          ;  Data message
                     00AE    382 ;        BNEQ    80$                     ; Branch on error
                     00AE    383 ;        CMPB    (R6),#7                 ; Check for value in the range 0-7
                     00AE    384 ;        BGTRU   HDR_INVALID             ; Branch on error
                     00AE    385 ;        RSB                             ;
                     00AE    386 ;-
                     00AE    387
                     00AE    388 60$:     STORE_FIELD     SYSPEC,8,K_IMG,<M_DESC>
                     00B5    389                                          ; Save descriptor of system specific
                     00B5    390                                          ;  field
          30 A9      91   00B5    391          CMPB    DAP$B_TYPE(R9),-        ; SYSPEC field not allowed in
             01           00B8    392                  #DAP$K_CNF_MSG          ;  Configuration message
             05      13   00B9    393          BEQL    80$                     ; Branch on error
01 69        34      E1   00BB    394          BBC     #DAP$V_VAXVMS,(R9),80$  ; SYSPEC field allowed only if
                     00BF    395                                          ;  systems are homogeneous
             05      00   00BF    396          RSB                             ;
             68      17   00C0    397 80$:     JMP     (R8)                    ; Branch to error_format routine
                     00C2    398
                     00C2    399 ;
                     00C2    400 ; Branch here on exception condition.
                     00C2    401 ;
                     00C2    402
                     00C2    403 HDR_INVALID:                             ;
           0790      31   00C2    404          BRW     ERROR_INVALID           ; Branch aid
                     00C5    405 HDR_UNSUPPORT:                           ;
           0799      31   00C5    406          BRW     ERROR_UNSUPPORT         ; Branch aid
```

```
                     00C8   408          .SBTTL  DISPATCH_TABLE - CASE ON MESSAGE TYPE
                     00C8   409
                     00C8   410 ;+
                     00C8   411 ; The DAP messasge header has been successfully parsed. Now dispatch on message
                     00C8   412 ; type to the appropriate code segment to process the body of the message.
                     00C8   413 ;
                     00C8   414 ; Note: The case table entries below should match the DAP$K_VALID_F2R message
                     00C8   415 ;       mask!
                     00C8   416 ;-
                     00C8   417
                     00C8   418 DISPATCH_TABLE:                              ; Continuation of mainline
57   00   9A        00C8   419          MOVZBL  #DAP$_UNKNOWN,R7            ; Set field ID to 'unknown'
                     00CB   420          $CASEB  SELECTOR=DAP$B_TYPE(R9)-    ; Dispatch to message specific decode
                     00CB   421                  BASE=#DAP$K_CNF_MSG-        ;  routine to process:
                     00CB   422                  DISPL=<-                    ;
                     00CB   423                      CNF_MSG-                ;  Configuration message
                     00CB   424                      ATT_MSG-                ;  Attributes message
                     00CB   425                      ERROR_SYNC-             ;  Access message
                     00CB   426                      ERROR_SYNC-             ;  Control message
                     00CB   427                      ERROR_SYNC-             ;  Continue Transfer message
                     00CB   428                      ACK_MSG-                ;  Acknowledge message
                     00CB   429                      CMP_MSG-                ;  Access Complete message
                     00CB   430                      DAT_MSG-                ;  Data message
                     00CB   431                      STS_MSG-                ;  Status message
                     00CB   432                      KEY_MSG-                ;  Key Definition message
                     00CB   433                      ALL_MSG-                ;  Allocation message
                     00CB   434                      SUM_MSG-                ;  Summary message
                     00CB   435                      TIM_MSG-                ;  Date and Time message
                     00CB   436                      PRO_MSG-                ;  Protection message
                     00CB   437                      NAM_MSG-                ;  Name message
                     00CB   438                  >                           ;
                     00EE   439
                     00EE   440 ;
                     00EE   441 ; The message type value has been validated (bounds checked), so the type value
                     00EE   442 ; will not be outside the range of the case table above.
                     00EE   443 ;
```

```
                     00EE    445          .SBTTL  CNF_MSG - DECODE CONFIGURATION MESSAGE
                     00EE    446
                     00EE    447 ;++
                     00EE    448 ; Decode the operand fields of the Configuration message.
                     00EE    449 ;--
                     00EE    450
                     00EE    451 CNF_MSG:                                    ; Code segment of mainline
 58   0846'CF   9E   00EE    452          MOVAB   W^ERROR_FORMAT,R8          ; Specify transfer address on EOM
                     00F3    453
                     00F3    454 ;
                     00F3    455 ; Process the buffer size field (required).
                     00F3    456 ;
                     00F3    457
                     00F3    458          STORE_FIELD     BUFSIZ,2,K_FIX  ; Save buffer size field
                     00FA    459
                     00FA    460 ;
                     00FA    461 ; Process system software and DAP protocol version number fields (required).
                     00FA    462 ; These fields are for information purposes only; hence no bounds checking
                     00FA    463 ; on their values is performed.
                     00FA    464 ;
                     00FA    465
                     00FA    466          STORE_FIELD     OSTYPE,1,K_FIX  ; Save operating system type field
                     0101    467          STORE_FIELD     FILESYS,1,K_FIX ; Save file system type field
                     0108    468          STORE_FIELD     VERNUM,1,K_FIX  ; Save DAP version # field
                     010F    469          STORE_FIELD     ECONUM,1,K_FIX  ; Save ECO version # field
                     0116    470          STORE_FIELD     USRNUM,1,K_FIX  ; Save user protocol version # field
                     011D    471          STORE_FIELD     DECVER,1,K_FIX  ; Save DEC software version # field
                     0124    472          STORE_FIELD     USRVER,1,K_FIX  ; Save user software version # field
                     012B    473
                     012B    474 ;
                     012B    475 ; Process the system capabilities field (required).
                     012B    476 ; Bits set that are not defined in the DAP spec are ignored (not flagged as
                     012B    477 ; an error) to facilitate compatibility with earlier implementations of DAP.
                     012B    478 ;
                     012B    479
                     012B    480          STORE_FIELD     SYSCAP,8,K_EXT,<M_TRUNC>
                     0132    481                                             ; Save system capabilities field
                     0132    482
                     0132    483 CHECK_PROTOCOL_VERSION:                     ; Set appropriate DAP$Q_DCODE_FLG bits
   50   44 A9   9A  0132    484          MOVZBL  DAP$B_VERNUM(R9),R0        ; Combine version number and ECO
50   50   08   78  0136    485          ASHL    #8,R0,R0                   ;  number fields into one 16-bit
   50   45 A9   80  013A    486          ADDB2   DAP$B_ECONUM(R9),R0        ;  value for easy comparison
                     013E    487
                     013E    488 ;
                     013E    489 ; Set status flag if partner implemented to DAP spec since V4.1.
                     013E    490 ;
                     013E    491
0401 8F    50   B1  013E    492          CMPW    R0,#^X0401                 ; Did partner implement since DAP V4.1?
          51   1F  0143    493          BLSSU   10$                        ; Branch if not
                     0145    494          $SETBIT #DAP$V_GEQ_V41,(R9)        ; Set flag
                     0149    495
                     0149    496 ;
                     0149    497 ; Set status flag if partner implemented to DAP spec since V4.2.
                     0149    498 ;
                     0149    499
0402 8F    50   B1  0149    500          CMPW    R0,#^X0402                 ; Did partner implement since DAP V4.2?
          46   1F  014E    501          BLSSU   10$                        ; Branch if not
```

```
                     0150   502          $SETBIT #DAP$V_GEQ_V42,(R9)     ; Set flag
                     0154   503
                     0154   504 ;
                     0154   505 ; Set status flag if partner implemented to DAP spec since V5.2.
                     0154   506 ;
                     0154   507
0502 8F   50   B1    0154   508          CMPW    R0,#^X0502              ; Did partner implement since DAP V5.2?
          3B   1F    0159   509          BLSSU   10$                     ; Branch if not
                     015B   510          $SETBIT #DAP$V_GEQ_V52,(R9)     ; Set flag
                     015F   511
                     015F   512 ;
                     015F   513 ; Set status flag if partner implemented to DAP spec since V5.4.
                     015F   514 ;
                     015F   515
0504 8F   50   B1    015F   516          CMPW    R0,#^X0504              ; Did partner implement since DAP V5.4?
          30   1F    0164   517          BLSSU   10$                     ; Branch if not
                     0166   518          $SETBIT #DAP$V_GEQ_V54,(R9)     ; Set flag
                     016A   519
                     016A   520 ;
                     016A   521 ; Set status flag if partner implemented to DAP spec since V5.6.
                     016A   522 ;
                     016A   523
0506 8F   50   B1    016A   524          CMPW    R0,#^X0506              ; Did partner implement since DAP V5.6?
          25   1F    016F   525          BLSSU   10$                     ; Branch if not
                     0171   526          $SETBIT #DAP$V_GEQ_V56,(R9)     ; Set flag
                     0175   527
                     0175   528 ;
                     0175   529 ; Set status flag if partner implemented to DAP spec since V6.0.
                     0175   530 ;
                     0175   531
0600 8F   50   B1    0175   532          CMPW    R0,#^X0600              ; Did partner implement since DAP V6.0?
          1A   1F    017A   533          BLSSU   10$                     ; Branch if not
                     017C   534          $SETBIT #DAP$V_GEQ_V60,(R9)     ; Set flag
                     0180   535
                     0180   536 ;
                     0180   537 ; Set status flag if partner implemented to DAP spec since V7.0.
                     0180   538 ;
                     0180   539
0700 8F   50   B1    0180   540          CMPW    R0,#^X0700              ; Did partner implement since DAP V7.0?
          0F   1F    0185   541          BLSSU   10$                     ; Branch if not
                     0187   542          $SETBIT #DAP$V_GEQ_V70,(R9)     ; Set flag
                     018B   543
                     018B   544 ;
                     018B   545 ; Set status flag if partner implemented to DAP spec since V7.1.
                     018B   546 ;
                     018B   547
0701 8F   50   B1    018B   548          CMPW    R0,#^X0701              ; Did partner implement since DAP V7.1?
          04   1F    0190   549          BLSSU   10$                     ; Branch if not
                     0192   550          $SETBIT #DAP$V_GEQ_V71,(R9)     ; Set flag
                     0196   551
                     0196   552 ;
                     0196   553 ; Set experimental protocol flags from the low order four bits of the USRNUM
                     0196   554 ; field only if partner is VAX/VMS.
                     0196   555 ;
                     0196   556
     42   A9   91    0196   557 10$:     CMPB    DAP$B_OSTYPE(R9),-      ; Branch if partner is not VAX/VMS
          07         0199   558                  #DAP$K_VAXVMS           ;
```

```
                     0B   12  019A   559          BNEQ    CHECK_FILE_SYSTEM       ;
          50   46 A9   04   00   EF  019C   560          EXTZV   #0,#4,DAP$B_USRNUM(R9),R0  ; Get low order four bits of USRNUM
             69   04   2C   50   F0  01A2   561          INSV    R0,#DAP$V_VMS_XPF1,#4,(R9) ; Set experimental protocol flags
                                   01A7   562
                                   01A7   563 CHECK_FILE_SYSTEM:                       ; Set appropriate DAP$Q_DCODE_FLG bit.
                                   01A7   564
                                   01A7   565          ASSUME  DAP$K_RMS11 EQ 1
                                   01A7   566          ASSUME  DAP$K_RMS20 EQ 2
                                   01A7   567          ASSUME  DAP$K_RMS32 EQ 3
                                   01A7   568          ASSUME  DAP$K_FCS11 EQ 4
                                   01A7   569          ASSUME  DAP$K_RT11FS EQ 5
                                   01A7   570          ASSUME  DAP$K_NO_FS EQ 6
                                   01A7   571          ASSUME  DAP$K_TOPS20FS EQ 7
                                   01A7   572          ASSUME  DAP$K_TOPS10FS EQ 8
                                   01A7   573          ASSUME  DAP$K_RMS32S EQ 10
                                   01A7   574
                                   01A7   575 ;
                                   01A7   576 ; Set status flag pertaining to the type of file system used by the remote node.
                                   01A7   577 ;
                                   01A7   578
                                   01A7   579          $CASEB  SELECTOR=DAP$B_FILESYS(R9)-
                                   01A7   580                  BASE=#DAP$K_RMS11-      ; Type of remote file system:
                                   01A7   581                  DISPL=<-                ;
                                   01A7   582                      10$-                ;  RMS-11
                                   01A7   583                      10$-                ;  RMS-20
                                   01A7   584                      10$-                ;  RMS-32
                                   01A7   585                      20$-                ;  FCS-11
                                   01A7   586                      30$-                ;  RT-11
                                   01A7   587                      40$-                ;  No file system present
                                   01A7   588                      30$-                ;  TOPS-20
                                   01A7   589                      30$-                ;  TOPS-10
                                   01A7   590                      40$-                ;  Undefined
                                   01A7   591                      10$-                ;  RMS-32 subset
                                   01A7   592                  >                       ;
     10    11                    01C0   593          BRB     40$                     ;
                                   01C2   594 10$:     $SETBIT #DAP$V_RMS,(R9)         ; Set RMS based file system flag
     0A    11                    01C6   595          BRB     40$                     ;
                                   01C8   596 20$:     $SETBIT #DAP$V_FCS,(R9)         ; Set FCS based file system flag
     04    11                    01CC   597          BRB     40$                     ;
                                   01CE   598 30$:     $SETBIT #DAP$V_STM_ONLY,(R9)    ; Set stream ASCII file system flag
                                   01D2   599 40$:                                     ;
                                   01D2   600
                                   01D2   601 CHECK_OPERATING_SYSTEM:                  ; Set appropriate DAP$Q_DCODE_FLG bit
                                   01D2   602
                                   01D2   603          ASSUME  DAP$K_RT11 EQ 1
                                   01D2   604          ASSUME  DAP$K_RSTS EQ 2
                                   01D2   605          ASSUME  DAP$K_RSX11S EQ 3
                                   01D2   606          ASSUME  DAP$K_RSX11M EQ 4
                                   01D2   607          ASSUME  DAP$K_RSX11D EQ 5
                                   01D2   608          ASSUME  DAP$K_IAS EQ 6
                                   01D2   609          ASSUME  DAP$K_VAXVMS EQ 7
                                   01D2   610          ASSUME  DAP$K_TOPS20 EQ 8
                                   01D2   611          ASSUME  DAP$K_TOPS10 EQ 9
                                   01D2   612          ASSUME  DAP$K_RSX11MP EQ 12
                                   01D2   613          ASSUME  DAP$K_COPOS11 EQ 13
                                   01D2   614          ASSUME  DAP$K_P_OS EQ 14
                                   01D2   615          ASSUME  DAP$K_VAXELAN EQ 15
```

```
                     01D2   616
                     01D2   617 ;
                     01D2   618 ; Set status flag pertaining to the type of operating system being used at the
                     01D2   619 ; remote node.
                     01D2   620 ;
                     01D2   621
                     01D2   622          $CASEB  SELECTOR=DAP$B_OSTYPE(R9)-
                     01D2   623                  BASE=#DAP$K_RT11-       ; Type of remote operating system:
                     01D2   624                  DISPL=<-                ;
                     01D2   625                      50$-                ; RT-11
                     01D2   626                      60$-                ; RSTS/E
                     01D2   627                      70$-                ; RSX-11S
                     01D2   628                      70$-                ; RSX-11M
                     01D2   629                      80$-                ; RSX-11D (classified as IAS)
                     01D2   630                      80$-                ; IAS
                     01D2   631                      10$-                ; VAX/VMS
                     01D2   632                      40$-                ; TOPS-20
                     01D2   633                      30$-                ; TOPS-10
                     01D2   634                      100$-               ; Undefined
                     01D2   635                      100$-               ; Undefined
                     01D2   636                      70$-                ; RSX-11M-PLUS
                     01D2   637                      40$-                ; TOPS-20 (using 2050/2060 front end)
                     01D2   638                      90$-                ; P/OS
                     01D2   639                      20$-                ; VAXELAN
                     01D2   640                  >                       ;
               34 11 01F5   641          BRB     100$                    ;
                     01F7   642 10$:     $SETBIT #DAP$V_VAXVMS,(R9)      ; Set VAX/VMS system flag
               2E 11 01FB   643          BRB     100$                    ;
                     01FD   644 20$:     $SETBIT #DAP$V_VAXELAN,(R9)     ; Set VAXELAN system flag
               28 11 0201   645          BRB     100$                    ;
                     0203   646 30$:     $SETBIT #DAP$V_TOPS10,(R9)      ; Set TOPS-10 system flag
               22 11 0207   647          BRB     100$                    ;
                     0209   648 40$:     $SETBIT #DAP$V_TOPS20,(R9)      ; Set TOPS-20 and COPOS11 system flag
               1C 11 020D   649          BRB     100$                    ;
                     020F   650 50$:     $SETBIT #DAP$V_RT11,(R9)        ; Set RT-11 system flag
               16 11 0213   651          BRB     100$                    ;
                     0215   652 60$:     $SETBIT #DAP$V_RSTS,(R9)        ; Set RSTS/E system flag
               10 11 0219   653          BRB     100$                    ;
                     021B   654 70$:     $SETBIT #DAP$V_RSX,(R9)         ; Set RSX-11S, RSX-11M, and RSX-11M-PLUS
               0A 11 021F   655          BRB     100$                    ;  system flag
                     0221   656 80$:     $SETBIT #DAP$V_IAS,(R9)         ; Set IAS and RSX-11D system flag
               04 11 0225   657          BRB     100$                    ;
                     0227   658 90$:     $SETBIT #DAP$V_P_OS,(R9)        ; Set PO/S system flag
                     022B   659
             0640 31 022B   660 100$:    BRW     EXIT_SUCCESS            ; Message syntax is correct
```

```
                                022E   662          .SBTTL  ATT_MSG - DECODE ATTRIBUTES MESSAGE
                                022E   663
                                022E   664 ;++
                                022E   665 ; Decode the operand fields of the Attributes message.
                                022E   666 ;--
                                022E   667
                                022E   668 ATT_MSG:                                 ; Code segment of mainline
                                022E   669
                                022E   670 ;
                                022E   671 ; For optional fields, apply default values as appropriate.
                                022E   672 ;
                                022E   673
      44 A9    02    90         022E   674          MOVB    #DAP$K_DATATYP_D,DAP$B_DATATYPE(R9)
      45 A9    00    90         0232   675          MOVB    #DAP$K_ORG_D,DAP$B_ORG(R9)
      46 A9    01    90         0236   676          MOVB    #DAP$K_RFM_D,DAP$B_RFM(R9)
48 A9   0200  8F    B0         023A   677          MOVW    #DAP$K_BLS_D,DAP$W_BLS(R9)
      52 A9    08    90         0240   678          MOVB    #DAP$K_BSZ_D,DAP$B_BSZ(R9)
      60 A9    69    7E         0244   679          MOVAQ   (R9),DAP$Q_RUNSYS+4(R9) ; Initialize descriptor
                                0248   680
                                0248   681 ;
                                0248   682 ; Process the attributes menu field (optional).
                                0248   683 ; Each bit set denotes that its associated field follows in the message.
                                0248   684 ;
                                0248   685
                                0248   686          ASSUME  DAP$V_DATATYPE+1 EQ DAP$V_ORG
                                0248   687          ASSUME  DAP$V_ORG+1 EQ DAP$V_RFM
                                0248   688          ASSUME  DAP$V_RFM+1 EQ DAP$V_RAT
                                0248   689          ASSUME  DAP$V_RAT+1 EQ DAP$V_BLS
                                0248   690          ASSUME  DAP$V_BLS+1 EQ DAP$V_MRS
                                0248   691          ASSUME  DAP$V_MRS+1 EQ DAP$V_ALQ1
                                0248   692          ASSUME  DAP$V_ALQ1+1 EQ DAP$V_BKS
                                0248   693          ASSUME  DAP$V_BKS+1 EQ DAP$V_FSZ
                                0248   694          ASSUME  DAP$V_FSZ+1 EQ DAP$V_MRN
                                0248   695          ASSUME  DAP$V_MRN+1 EQ DAP$V_RUNSYS
                                0248   696          ASSUME  DAP$V_RUNSYS+1 EQ DAP$V_DEQ1
                                0248   697          ASSUME  DAP$V_DEQ1+1 EQ DAP$V_FOP1
                                0248   698          ASSUME  DAP$V_FOP1+1 EQ DAP$V_BSZ
                                0248   699          ASSUME  DAP$V_BSZ+1 EQ DAP$V_DEV
                                0248   700          ASSUME  DAP$V_DEV+2 EQ DAP$V_LRL
                                0248   701          ASSUME  DAP$V_LRL+1 EQ DAP$V_HBK
                                0248   702          ASSUME  DAP$V_HBK+1 EQ DAP$V_EBK
                                0248   703          ASSUME  DAP$V_EBK+1 EQ DAP$V_FFB
                                0248   704          ASSUME  DAP$V_FFB+1 EQ DAP$V_SBN
                                0248   705
   58    086E'CF    9E         0248   706          MOVAB   W^EXIT_SUCCESS,R8       ; All done if end-of-message
                                024D   707          STORE_FIELD     ATTMENU,4,K_EXT ; Save attributes menu field
                                0254   708          CHECK_MASKS     ATTMENU,4       ; Validate bit options
   58    0846'CF    9E         0260   709          MOVAB   W^ERROR_FORMAT,R8       ; Specify transfer address on EOM
      5C    66    D0         0265   710          MOVL    (R6),AP                 ; Copy menu to scratch register
                                0268   711 ATT_LOOP:                             ;
50    5C    15    00    EA     0268   712          FFS     #0,#DAP$V_SBN+1,AP,R0   ; Get position of next bit set
                                026D   713          $CLRBIT R0,AP                   ; Clear menu bit just found
         F4 AF    9F         0271   714          PUSHAB  B^ATT_LOOP              ; Push return address on stack
                                0274   715          $CASEB  SELECTOR=R0-            ; Next field:
                                0274   716                  DISPL=<-               ;
                                0274   717                      10$-               ;  DATATYPE
                                0274   718                      20$-               ;  ORG
```

```
                0274    719                     30$-                    ;  RFM
                0274    720                     40$-                    ;  RAT
                0274    721                     50$-                    ;  BLS
                0274    722                     60$-                    ;  MRS
                0274    723                     70$-                    ;  ALQ1
                0274    724                     80$-                    ;  BKS
                0274    725                     90$-                    ;  FSZ
                0274    726                     100$-                   ;  MRN
                0274    727                     110$-                   ;  RUNSYS
                0274    728                     120$-                   ;  DEQ1
                0274    729                     130$-                   ;  FOP1
                0274    730                     140$-                   ;  BSZ
                0274    731                     150$-                   ;  DEV
                0274    732                     ERROR_FORMAT-           ;  Reserved
                0274    733                     170$-                   ;  LRL
                0274    734                     180$-                   ;  HBK
                0274    735                     190$-                   ;  EBK
                0274    736                     200$-                   ;  FFB
                0274    737                     210$-                   ;  SBN
                0274    738                 >                           ;
        05C9 31 02A2    739             BRW     EXIT_SUCCESS            ; Message syntax is correct
                02A5    740
                02A5    741 ;
                02A5    742 ; Process each field specified in the menu (optional).
                02A5    743 ;
                02A5    744
                02A5    745 40$:        STORE_FIELD     RAT,1,K_EXT     ; Save record attributes field
                02AC    746             CHECK_MASKS     RAT,1           ; Validate bit options
             05 02B2    747             RSB                             ;
                02B3    748 50$:        STORE_FIELD     BLS,2,K_FIX     ; Save block size field
             05 02BA    749             RSB                             ;
                02BB    750 60$:        STORE_FIELD     MRS,2,K_FIX     ; Save maximum record size field
             05 02C2    751             RSB                             ;
                02C3    752 70$:        STORE_FIELD     ALQ1,4,K_IMG    ; Save allocation quantity field
             05 02CA    753             RSB                             ;
                02CB    754 80$:        STORE_FIELD     BKS,1,K_FIX     ; Save bucket size field
             05 02D2    755             RSB                             ;
                02D3    756 90$:        STORE_FIELD     FSZ,1,K_FIX     ; Save fixed control area size field
             05 02DA    757             RSB                             ;
                02DB    758 100$:       STORE_FIELD     MRN,4,K_IMG     ; Save maximum record number field
             05 02E2    759             RSB                             ;
                02E3    760 110$:       STORE_FIELD     RUNSYS,8,K_IMG,<M_DESC>
                02EA    761                                             ; Save descriptor of run-time
                02EA    762                                             ;  system string
             05 02EA    763             RSB                             ;
                02EB    764 120$:       STORE_FIELD     DEQ1,2,K_FIX    ; Save default extension quantity field
             05 02F2    765             RSB                             ;
                02F3    766 130$:       STORE_FIELD     FOP1,4,K_EXT    ; Save file options field
                02FA    767             CHECK_MASKS     FOP,4           ; Validate bit options
             05 0306    768             RSB                             ;
                0307    769 10$:        STORE_FIELD     DATATYPE,1,K_EXT; Save data type field
                030E    770             CHECK_MASKS     DATATYP,1       ; Validate bit options
             05 0314    771             RSB                             ;
                0315    772 20$:        STORE_FIELD     ORG,1,K_FIX     ; Save file organization field
                031C    773
                031C    774             ASSUME  DAP$K_SEQ EQ 0
                031C    775             ASSUME  DAP$K_REL EQ 16
```

```
                     031C   776          ASSUME  DAP$K_IDX EQ 32
                     031C   777
     66    95        031C   778          TSTB    (R6)                    ; Check for valid value
     0C    13        031E   779          BEQL    25$                     ; Branch if ok
10   66    91        0320   780          CMPB    (R6),#DAP$K_REL         ; Check for valid value
     07    13        0323   781          BEQL    25$                     ; Branch if ok
20   66    91        0325   782          CMPB    (R6),#DAP$K_IDX         ; Check for valid value
     02    13        0328   783          BEQL    25$                     ; Branch if ok
     52    11        032A   784          BRB     ATT_INVALID             ; Branch on error
     05              032C   785 25$:      RSB                             ;
                     032D   786 30$:      STORE_FIELD     RFM,1,K_FIX     ; Save record format field
                     0334   787
                     0334   788          ASSUME  DAP$K_UDF EQ 0
                     0334   789          ASSUME  DAP$K_FIX EQ 1
                     0334   790          ASSUME  DAP$K_VAR EQ 2
                     0334   791          ASSUME  DAP$K_VFC EQ 3
                     0334   792          ASSUME  DAP$K_STM EQ 4
                     0334   793          ASSUME  DAP$K_STMLF EQ 5
                     0334   794          ASSUME  DAP$K_STMCR EQ 6
                     0334   795
06   66    91        0334   796          CMPB    (R6),#DAP$K_STMCR       ; Check for valid value
     45    1A        0337   797          BGTRU   ATT_INVALID             ; Branch if out-of-range
     05              0339   798          RSB                             ;
                     033A   799 140$:     STORE_FIELD     BSZ,1,K_FIX     ; Save byte size field
     05              0341   800          RSB                             ;
                     0342   801 150$:     STORE_FIELD     DEV,4,K_EXT     ; Save device characteristics field
                     0349   802          CHECK_MASKS     DEV,4           ; Validate bit options
     05              0355   803          RSB                             ;
                     0356   804 170$:     STORE_FIELD     LRL,2,K_FIX     ; Save longest record length field
     05              035D   805          RSB                             ;
                     035E   806 180$:     STORE_FIELD     HBK,4,K_IMG     ; Save highest virtual block number
     05              0365   807          RSB                             ;  field
                     0366   808 190$:     STORE_FIELD     EBK,4,K_IMG     ; Save end-of-file block number field
     05              036D   809          RSB                             ;
                     036E   810 200$:     STORE_FIELD     FFB,2,K_FIX     ; Save first free byte in EOF block
     05              0375   811          RSB                             ;  field
                     0376   812 210$:     STORE_FIELD     SBN,4,K_IMG     ; Save starting logical block number
     05              037D   813          RSB                             ;  field
                     037E   814
                     037E   815 ;
                     037E   816 ; Branch here on exception condition.
                     037E   817 ;
                     037E   818
                     037E   819 ATT_INVALID:                              ;
04D4 31              037E   820          BRW     ERROR_INVALID           ; Branch aid
```

```
                     0381    822          .SBTTL  ACK_MSG - DECODE ACKNOWLEDGE MESSAGE
                     0381    823
                     0381    824 ;++
                     0381    825 ; There are no operand fields in the Acknowledge message.
                     0381    826 ;--
                     0381    827
                     0381    828 ACK_MSG:                                    ; Code segment of mainline
04EA    31           0381    829          BRW     EXIT_SUCCESS               ; Message syntax is correct
```

```
                     0384    831          .SBTTL  CMP_MSG - DECODE ACCESS COMPLETE MESSAGE
                     0384    832
                     0384    833 ;++
                     0384    834 ; Decode the operand fields of the Access Complete message.
                     0384    835 ;--
                     0384    836
                     0384    837 CMP_MSG:                                ; Code segment of mainline
                     0384    838
                     0384    839 ;
                     0384    840 ; For optional fields, apply default values as appropriate.
                     0384    841 ;
                     0384    842
                     0384    843 ;       <there are no defaults to apply>
                     0384    844
                     0384    845 ;
                     0384    846 ; Process the access complete function field (required).
                     0384    847 ;
                     0384    848
        58  0846'CF    9E   0384    849          MOVAB   W^ERROR_FORMAT,R8       ; Specify transfer address on EOM
                     0389    850          STORE_FIELD     CMPFUNC,1,K_FIX ; Save access complete function field
                     0390    851
                     0390    852          ASSUME  DAP$K_CLOSE EQ 1
                     0390    853          ASSUME  DAP$K_RESPONSE EQ 2
                     0390    854          ASSUME  DAP$K_RESET EQ 3
                     0390    855          ASSUME  DAP$K_DISCONN EQ 4
                     0390    856          ASSUME  DAP$K_SKIP_FILE EQ 5
                     0390    857          ASSUME  DAP$K_CHANGE_B EQ 6
                     0390    858          ASSUME  DAP$K_CHANGE_E EQ 7
                     0390    859          ASSUME  DAP$K_TERMINATE EQ 8
                     0390    860
        66  95   0390    861          TSTB    (R6)                    ; Branch if value is
        2B  13   0392    862          BEQL    CMP_INVALID             ;  too low
    08  66  91   0394    863          CMPB    (R6),#DAP$K_TERMINATE   ;  or
        26  1A   0397    864          BGTRU   CMP_INVALID             ;  too high
                     0399    865
                     0399    866 ;
                     0399    867 ; Process the file options field (optional).
                     0399    868 ;
                     0399    869
        58  086E'CF    9E   0399    870          MOVAB   W^EXIT_SUCCESS,R8       ; All done if end-of-message
                     039E    871          STORE_FIELD     FOP2,4,K_EXT    ; Save file options field
                     03A5    872          CHECK_MASKS     FOP,4           ; Validate bit options
                     03B1    873
                     03B1    874 ;
                     03B1    875 ; Process the CRC checksum field (optional).
                     03B1    876 ;
                     03B1    877
                     03B1    878          STORE_FIELD     CHECK,2,K_FIX   ; Save CRC checksum field
                     03B8    879          $SETBIT #DAP$V_X_CHECK,-        ; Denote field explicitly specified
                     03B8    880                  DAP$B_X_FIELD(R9)       ;  (to distinguish between CRC value
                     03BD    881                                          ;  of zero and none specified)
        68  17   03BD    882          JMP     (R8)                    ; Message syntax is correct
                     03BF    883
                     03BF    884 ;
                     03BF    885 ; Branch here on exception condition.
                     03BF    886 ;
                     03BF    887
```

```
                03BF    888 CMP_INVALID:                        ;
0493     31     03BF    889         BRW     ERROR_INVALID       ; Branch aid
```

```
                        03C2    891          .SBTTL  DAT_MSG - DECODE DATA MESSAGE
                        03C2    892
                        03C2    893 ;++
                        03C2    894 ; Decode the operand fields of the Data message.
                        03C2    895 ;--
                        03C2    896
                        03C2    897 DAT_MSG:                                ; Code segment of mainline
                        03C2    898
                        03C2    899 ;
                        03C2    900 ; For optional fields, apply default values as appropriate.
                        03C2    901 ;
                        03C2    902
      48 A9   69   7E   03C2    903          MOVAQ   (R9),DAP$Q_FILEDATA+4(R9) ; Initialize descriptor
                        03C6    904
                        03C6    905 ;
                        03C6    906 ; Process the record number field (required).
                        03C6    907 ;
                        03C6    908 ; Note: Since there is no menu for the Data message (an unfortunate oversight
                        03C6    909 ;       in the DAP spec), this field must be present. However, it is necessary
                        03C6    910 ;       to distinguish between receiving a null value and a zero value, so that
                        03C6    911 ;       it can be determined whether the RECNUM field overrides the KEY field.
                        03C6    912 ;       To solve this problem, the DAP spec states that a byte count of zero
                        03C6    913 ;       for this image field means that no value has been specified. I.e.,
                        03C6    914 ;       <byte0 = 0> means ignore field, whereas <byte0 = 1 and byte1 = 0> means
                        03C6    915 ;       a value of zero overrides the KEY field value.
                        03C6    916 ;
                        03C6    917
   58   0846'CF   9E    03C6    918          MOVAB   W^ERROR_FORMAT,R8           ; Specify transfer address on EOM
   5C   5B   01   C1    03CB    919          ADDL3   #1,R11,AP                   ; Mark address of next byte + 1
                        03CF    920          STORE_FIELD     RECNUM1,4,K_IMG ; Save record number field
        5C   5B   D1    03D6    921          CMPL    R11,AP                      ; Branch if this image field was
             05   13    03D9    922          BEQLU   10$                         ;  exactly one byte long
                        03DB    923          $SETBIT #DAP$V_X_RECNUM,-           ; Denote field explicitly specified
                        03DB    924                  DAP$B_X_FIELD(R9)          ;  in message
                        03E0    925
                        03E0    926 ;
                        03E0    927 ; Process the file data field (optional for zero length record).
                        03E0    928 ;
                        03E0    929
   58   086E'CF   9E    03E0    930 10$:     MOVAB   W^EXIT_SUCCESS,R8           ; All done if end-of-message
                        03E5    931          STORE_FIELD     FILEDATA,8,K_ROM,<M_DESC>
                        03EC    932                                             ; Save descriptor of user data string
                        03EC    933                                             ;  (the record/block just received)
             68   17    03EC    934          JMP     (R8)                        ; Message syntax is correct
```

```
                        03EE    936          .SBTTL  STS_MSG - DECODE STATUS MESSAGE
                        03EE    937
                        03EE    938 ;++
                        03EE    939 ; Decode the operand fields of the Status message.
                        03EE    940 ;--
                        03EE    941
                        03EE    942 STS_MSG:                                ; Code segment of mainline
                        03EE    943
                        03EE    944 ;
                        03EE    945 ; For optional fields, apply default values as appropriate.
                        03EE    946 ;
                        03EE    947
      50 A9   69   7E   03EE    948          MOVAQ   (R9),DAP$Q_STX(R9)      ; Initialize descriptor
                        03F2    949
                        03F2    950 ;
                        03F2    951 ; Process the status code field (required).
                        03F2    952 ;
                        03F2    953
   58   0846'CF   9E   03F2    954          MOVAB   W^ERROR_FORMAT,R8       ; Specify transfer address on EOM
                        03F7    955          STORE_FIELD     STSCODE,2,K_FIX ; Save status code field
                        03FE    956
                        03FE    957 ;
                        03FE    958 ; Process the RFA, RECNUM, STV, and STX fields (optional).
                        03FE    959 ;
                        03FE    960
   58   086E'CF   9E   03FE    961          MOVAB   W^EXIT_SUCCESS,R8       ; All done if end-of-message
                        0403    962          STORE_FIELD     RFA,6,K_IMG     ; Save record file address field
                        040A    963          STORE_FIELD     RECNUM2,4,K_IMG ; Save record number field
                        0411    964          STORE_FIELD     STV,4,K_IMG     ; Save secondary status value field
                        0418    965          STORE_FIELD     STX,8,K_IMG,<M_DESC>
                        041F    966                                          ; Save descriptor of secondary status
                        041F    967                                          ;  text field
         68   17   041F    968          JMP     (R8)                        ; Message syntax is correct
```

```
                     0421   970          .SBTTL  KEY_MSG - DECODE KEY DEFINITION MESSAGE
                     0421   971
                     0421   972 ;++
                     0421   973 ; Decode the operand fields of the Key Definition message.
                     0421   974 ;--
                     0421   975
                     0421   976 KEY_MSG:                                ; Code segment of mainline
                     0421   977
                     0421   978 ;
                     0421   979 ; For optional fields, apply default values as appropriate.
                     0421   980 ;
                     0421   981
 68 A9   69   7E     0421   982          MOVAQ   (R9),DAP$Q_KNM+4(R9)    ; Initialize descriptor
                     0425   983
                     0425   984 ;
                     0425   985 ; Process the attributes menu field (optional).
                     0425   986 ; Each bit set denotes that its associated field follows in the message.
                     0425   987 ;
                     0425   988
                     0425   989          ASSUME  DAP$V_FLG+1 EQ DAP$V_DFL
                     0425   990          ASSUME  DAP$V_DFL+1 EQ DAP$V_IFL
                     0425   991          ASSUME  DAP$V_IFL+1 EQ DAP$V_NSG
                     0425   992          ASSUME  DAP$V_NSG+1 EQ DAP$V_REF
                     0425   993          ASSUME  DAP$V_REF+1 EQ DAP$V_KNM
                     0425   994          ASSUME  DAP$V_KNM+1 EQ DAP$V_NUL
                     0425   995          ASSUME  DAP$V_NUL+1 EQ DAP$V_IAN
                     0425   996          ASSUME  DAP$V_IAN+1 EQ DAP$V_LAN
                     0425   997          ASSUME  DAP$V_LAN+1 EQ DAP$V_DAN
                     0425   998          ASSUME  DAP$V_DAN+1 EQ DAP$V_DTP
                     0425   999          ASSUME  DAP$V_DTP+1 EQ DAP$V_RVB
                     0425  1000          ASSUME  DAP$V_RVB+2 EQ DAP$V_DVB
                     0425  1001          ASSUME  DAP$V_DVB+1 EQ DAP$V_DBS
                     0425  1002          ASSUME  DAP$V_DBS+1 EQ DAP$V_IBS
                     0425  1003          ASSUME  DAP$V_IBS+1 EQ DAP$V_LVL
                     0425  1004          ASSUME  DAP$V_LVL+1 EQ DAP$V_TKS
                     0425  1005          ASSUME  DAP$V_TKS+1 EQ DAP$V_MRL
                     0425  1006
    58   086E'CF   9E   0425  1007          MOVAB   W^EXIT_SUCCESS,R8       ; All done if end-of-message
                     042A  1008          STORE_FIELD     KEYMENU,4,K_EXT ; Save key definition menu field
                     0431  1009          CHECK_MASKS     KEYMENU,4       ; Validate bit options
    58   0846'CF   9E   043D  1010          MOVAB   W^ERROR_FORMAT,R8       ; Specify transfer address on EOM
         5C   66   D0   0442  1011          MOVL    (R6),AP                 ; Copy menu to scratch register
                     0445  1012 KEY_LOOP:                                ;
50   5C   13   00   EA   0445  1013          FFS     #0,#DAP$V_MRL+1,AP,R0   ; Get position of next bit set
                     044A  1014          $CLRBIT R0,AP                   ; Clear menu bit just found
            F4 AF   9F   044E  1015          PUSHAB  B^KEY_LOOP              ; Push return address on stack
                     0451  1016          $CASEB  SELECTOR=R0-            ; Next field:
                     0451  1017                  DISPL=<-                ;
                     0451  1018                      10$-                ;  FLG
                     0451  1019                      20$-                ;  DFL
                     0451  1020                      30$-                ;  IFL
                     0451  1021                      40$-                ;  NSG, POS, SIZ
                     0451  1022                      50$-                ;  REF
                     0451  1023                      60$-                ;  KNM
                     0451  1024                      70$-                ;  NUL
                     0451  1025                      80$-                ;  IAN
                     0451  1026                      90$-                ;  LAN
```

```
                         0451  1027                        100$-                   ;  DAN
                         0451  1028                        110$-                   ;  DTP
                         0451  1029                        120$-                   ;  RVB
                         0451  1030                        ERROR_FORMAT-           ;  Reserved
                         0451  1031                        140$-                   ;  DVB
                         0451  1032                        150$-                   ;  DBS
                         0451  1033                        160$-                   ;  IBS
                         0451  1034                        170$-                   ;  LVL
                         0451  1035                        180$-                   ;  TKS
                         0451  1036                        190$-                   ;  MRL
                         0451  1037              >                                 ;
                  03F0   31      047B  1038              BRW     EXIT_SUCCESS      ; Message syntax is correct
                         047E  1039
                         047E  1040 ;
                         047E  1041 ; Process each field specified in the menu (optional).
                         047E  1042 ;
                         047E  1043
                         047E  1044 10$:  STORE_FIELD     FLG,1,K_EXT             ; Save key options field
                         0485  1045       CHECK_MASKS     FLG,1                   ; Validate bit options
                      05 048B  1046       RSB                                     ;
                         048C  1047 20$:  STORE_FIELD     DFL,2,K_FIX             ; Save data bucket fill quantity field
                      05 0493  1048       RSB                                     ;
                         0494  1049 30$:  STORE_FIELD     IFL,2,K_FIX             ; Save index bucket fill quantity field
                      05 049B  1050       RSB                                     ;
                         049C  1051 40$:  STORE_FIELD     NSG,1,K_FIX             ; Save number of key segments field
                   52 66 9A 04A3  1052       MOVZBL  (R6),R2                     ; Use number of segments as loop count
                      2C 13 04A6  1053       BEQL    47$                         ; Branch if zero
                   08 52 D1 04A8  1054       CMPL    R2,#8                       ; Check for value too high
                      28 1A 04AB  1055       BGTRU   49$                         ; Branch on error
              50   4C A9 3E 04AD  1056       MOVAW   DAP$W_POS(R9),R0            ; Get address of POS array
              51   5C A9 9E 04B1  1057       MOVAB   DAP$B_SIZ(R9),R1            ; Get address of SIZ array
                      07 BB 04B5  1058 45$:  PUSHR   #^M<R0,R1,R2>               ;
                         04B7  1059       STORE_FIELD     POS_TMP,2,K_FIX         ; Find next key segment size field
                      01 BA 04BE  1060       POPR    #^M<R0>                     ;
                   80 66 B0 04C0  1061       MOVW    (R6),(R0)+                  ; Save it in array
                      01 BB 04C3  1062       PUSHR   #^M<R0>                     ;
                         04C5  1063       STORE_FIELD     SIZ_TMP,1,K_FIX         ; Find next key segment size field
                      07 BA 04CC  1064       POPR    #^M<R0,R1,R2>               ;
                   81 66 90 04CE  1065       MOVB    (R6),(R1)+                  ; Save it in array
                     E1 52 F5 04D1  1066       SOBGTR  R2,45$                      ; Branch if more segments to go
                      05 04D4  1067 47$:  RSB                                    ;
                      7F 11 04D5  1068 49$:  BRB     KEY_INVALID                 ; Branch aid
                         04D7  1069 50$:  STORE_FIELD     REF,1,K_FIX             ; Save key of reference field
                      05 04DE  1070       RSB                                     ;
                         04DF  1071 60$:  STORE_FIELD     KNM,8,K_IMG,<M_DESC>
                         04E6  1072                                               ; Save descriptor of key name string
                   28 66 91 04E6  1073       CMPB    (R6),#40                    ; Check for string too long
                      6B 1A 04E9  1074       BGTRU   KEY_INVALID                 ; Branch on error
                   20 66 91 04EB  1075       CMPB    (R6),#32                    ; Check for string too long
                      66 1A 04EE  1076       BGTRU   KEY_INVALID                 ; Branch on error
                      05 04F0  1077       RSB                                     ;
                         04F1  1078 70$:  STORE_FIELD     NUL,1,K_FIX             ; Save null key character field
                      05 04F8  1079       RSB                                     ;
                         04F9  1080 80$:  STORE_FIELD     IAN,1,K_FIX             ; Save index area number field
                      05 0500  1081       RSB                                     ;
                         0501  1082 90$:  STORE_FIELD     LAN,1,K_FIX             ; Save lowest level index area number
                         0508  1083                                               ;  field
```

```
                05  0508  1084          RSB                                 ;
                    0509  1085 100$:     STORE_FIELD     DAN,1,K_FIX         ; Save data area number field
                05  0510  1086          RSB                                 ;
                    0511  1087 110$:     STORE_FIELD     DTP,1,K_FIX         ; Save key data type field
                    0518  1088
                    0518  1089          ASSUME  DAP$K_STG EQ 0
                    0518  1090          ASSUME  DAP$K_IN2 EQ 1
                    0518  1091          ASSUME  DAP$K_BN2 EQ 2
                    0518  1092          ASSUME  DAP$K_IN4 EQ 3
                    0518  1093          ASSUME  DAP$K_BN4 EQ 4
                    0518  1094          ASSUME  DAP$K_PAC EQ 5
                    0518  1095          ASSUME  DAP$K_IN8 EQ 6
                    0518  1096          ASSUME  DAP$K_BN8 EQ 7
                    0518  1097
07   66   91       0518  1098          CMPB    (R6),#DAP$K_BN8             ; Check for value too high
     39   1A       051B  1099          BGTRU   KEY_INVALID                 ; Branch on error
                05  051D  1100          RSB                                 ;
                    051E  1101 120$:     STORE_FIELD     RVB,4,K_IMG         ; Save root bucket start VBN field
                05  0525  1102          RSB                                 ;
                    0526  1103 140$:     STORE_FIELD     DVB,4,K_IMG         ; Save first data bucket start VBN field
                05  052D  1104          RSB                                 ;
                    052E  1105 150$:     STORE_FIELD     DBS,1,K_FIX         ; Save data bucket fill size field
                05  0535  1106          RSB                                 ;
                    0536  1107 160$:     STORE_FIELD     IBS,1,K_FIX         ; Save index bucket fill size field
                05  053D  1108          RSB                                 ;
                    053E  1109 170$:     STORE_FIELD     LVL,1,K_FIX         ; Save level of root buckets field
                05  0545  1110          RSB                                 ;
                    0546  1111 180$:     STORE_FIELD     TKS,1,K_FIX         ; Save total key size field
                05  054D  1112          RSB                                 ;
                    054E  1113 190$:     STORE_FIELD     MRL,2,K_FIX         ; Save minimum record length to contain
                05  0555  1114          RSB                                 ;  key field
                    0556  1115
                    0556  1116 ;
                    0556  1117 ; Branch here on exception condition.
                    0556  1118 ;
                    0556  1119
                    0556  1120 KEY_INVALID:                                 ;
02FC    31  0556  1121          BRW     ERROR_INVALID                       ; Branch aid
```

```
                     0559  1123          .SBTTL  ALL_MSG - DECODE ALLOCATION MESSAGE
                     0559  1124
                     0559  1125 ;++
                     0559  1126 ; Decode the operand fields of the Allocation message.
                     0559  1127 ;--
                     0559  1128
                     0559  1129 ALL_MSG:                                ; Code segment of mainline
                     0559  1130
                     0559  1131 ;
                     0559  1132 ; For optional fields, apply default values as appropriate.
                     0559  1133 ;
                     0559  1134
                     0559  1135 ;       <there are no defaults to apply>
                     0559  1136
                     0559  1137 ;
                     0559  1138 ; Process the allocation menu field (optional).
                     0559  1139 ; Each bit set denotes that its associated field follows in the message.
                     0559  1140 ;
                     0559  1141
                     0559  1142          ASSUME  DAP$V_VOL+1 EQ DAP$V_ALN
                     0559  1143          ASSUME  DAP$V_ALN+1 EQ DAP$V_AOP
                     0559  1144          ASSUME  DAP$V_AOP+1 EQ DAP$V_LOC
                     0559  1145          ASSUME  DAP$V_LOC+2 EQ DAP$V_ALQ2
                     0559  1146          ASSUME  DAP$V_ALQ2+1 EQ DAP$V_AID
                     0559  1147          ASSUME  DAP$V_AID+1 EQ DAP$V_BKZ
                     0559  1148          ASSUME  DAP$V_BKZ+1 EQ DAP$V_DEQ2
                     0559  1149
     58  086E'CF  9E 0559  1150          MOVAB   W^EXIT_SUCCESS,R8       ; All done if end-of-message
                     055E  1151          STORE_FIELD     ALLMENU,2,K_EXT ; Save allocation menu field
                     0565  1152          CHECK_MASKS     ALLMENU,2       ; Validate bit options
     58  0846'CF  9E 056D  1153          MOVAB   W^ERROR_FORMAT,R8       ; Specify transfer address on EOM
         5C   66  3C 0572  1154          MOVZWL  (R6),AP                 ; Copy menu to scratch register
                     0575  1155 ALL_LOOP:                               ;
50   5C  09   00  EA 0575  1156          FFS     #0,#DAP$V_DEQ2+1,AP,R0  ; Get position of next bit set
                     057A  1157          $CLRBIT R0,AP                   ; Clear menu bit just found
           F4 AF  9F 057E  1158          PUSHAB  B^ALL_LOOP              ; Push return address on stack
                     0581  1159          $CASEB  SELECTOR=R0-            ; Next field:
                     0581  1160                  DISPL=<-                ;
                     0581  1161                      10$-                ;  VOL
                     0581  1162                      20$-                ;  ALN
                     0581  1163                      30$-                ;  AOP
                     0581  1164                      40$-                ;  LOC
                     0581  1165                      ERROR_FORMAT-       ;  Reserved
                     0581  1166                      60$-                ;  ALQ2
                     0581  1167                      70$-                ;  AID
                     0581  1168                      80$-                ;  BKZ
                     0581  1169                      90$-                ;  DEQ2
                     0581  1170                  >                       ;
        02D4  31     0597  1171          BRW     EXIT_SUCCESS            ; Message syntax is correct
                     059A  1172
                     059A  1173 ;
                     059A  1174 ; Process each field specified in the menu (optional).
                     059A  1175 ;
                     059A  1176
                     059A  1177 10$:     STORE_FIELD     VOL,2,K_FIX     ; Save volume number field
                  05 05A1  1178          RSB                             ;
                     05A2  1179
```

```
                05A2  1180          ASSUME  DAP$K_ANY EQ 0
                05A2  1181          ASSUME  DAP$K_CYL EQ 1
                05A2  1182          ASSUME  DAP$K_LBN EQ 2
                05A2  1183          ASSUME  DAP$K_VBN EQ 3
                05A2  1184          ASSUME  DAP$K_RFI EQ 4
                05A2  1185
                05A2  1186 20$:     STORE_FIELD     ALN,1,K_FIX     ; Save alignment options field
03   66   91    05A9  1187          CMPB    (R6),#DAP$K_VBN         ; Check for value too high
     37   1A    05AC  1188          BGTRU   ALL_INVALID             ; Branch on error
          05    05AE  1189          RSB                             ;
                05AF  1190 30$:     STORE_FIELD     AOP,1,K_EXT     ; Save allocation options field
                05B6  1191          CHECK_MASKS     AOP,1           ; Validate bit options
          05    05BC  1192          RSB                             ;
                05BD  1193 40$:     STORE_FIELD     LOC,4,K_IMG     ; Save starting location field
          05    05C4  1194          RSB                             ;
                05C5  1195 60$:     STORE_FIELD     ALQ2,4,K_IMG    ; Save allocation quantity field
          05    05CC  1196          RSB                             ;
                05CD  1197 70$:     STORE_FIELD     AID,1,K_FIX     ; Save area identification field
          05    05D4  1198          RSB                             ;
                05D5  1199 80$:     STORE_FIELD     BKZ,1,K_FIX     ; Save bucket size field
          05    05DC  1200          RSB                             ;
                05DD  1201 90$:     STORE_FIELD     DEQ2,2,K_FIX    ; Save default extension quantity field
          05    05E4  1202          RSB                             ;
                05E5  1203
                05E5  1204 ;
                05E5  1205 ; Branch here on exception condition.
                05E5  1206 ;
                05E5  1207
                05E5  1208 ALL_INVALID:                             ;
   026D   31    05E5  1209          BRW     ERROR_INVALID           ; Branch aid
```

```
                        05E8  1211          .SBTTL  SUM_MSG - DECODE SUMMARY MESSAGE
                        05E8  1212
                        05E8  1213 ;++
                        05E8  1214 ; Decode the operand fields of the Summary message.
                        05E8  1215 ;--
                        05E8  1216
                        05E8  1217 SUM_MSG:                                 ; Code segment of mainline
                        05E8  1218
                        05E8  1219 ;
                        05E8  1220 ; For optional fields, apply default values as appropriate.
                        05E8  1221 ;
                        05E8  1222
                        05E8  1223 ;       <there are no defaults to apply>
                        05E8  1224
                        05E8  1225 ;
                        05E8  1226 ; Process the summary menu field (optional).
                        05E8  1227 ; Each bit set denotes that its associated field follows in the message.
                        05E8  1228 ;
                        05E8  1229
                        05E8  1230          ASSUME  DAP$V_NOK+1 EQ DAP$V_NOA
                        05E8  1231          ASSUME  DAP$V_NOA+1 EQ DAP$V_NOR
                        05E8  1232          ASSUME  DAP$V_NOR+1 EQ DAP$V_PVN
                        05E8  1233
     58  086E'CF   9E   05E8  1234          MOVAB   W^EXIT_SUCCESS,R8        ; All done if end-of-message
                        05ED  1235          STORE_FIELD     SUMENU,2,K_EXT   ; Save summary menu field
                        05F4  1236          CHECK_MASKS     SUMENU,2         ; Validate bit options
     58  0846'CF   9E   05FC  1237          MOVAB   W^ERROR_FORMAT,R8        ; Specify transfer address on EOM
         5C    66   3C   0601  1238          MOVZWL  (R6),AP                  ; Copy menu to scratch register
                        0604  1239 SUM_LOOP:                                ;
 50  5C  04    00   EA   0604  1240          FFS     #0,#DAP$V_PVN+1,AP,R0    ; Get position of next bit set
                        0609  1241          $CLRBIT R0,AP                    ; Clear menu bit just found
               F4 AF   9F   060D  1242          PUSHAB  B^SUM_LOOP               ; Push return address on stack
                        0610  1243          $CASEB  SELECTOR=R0-             ; Next field:
                        0610  1244                  DISPL=<-                 ;
                        0610  1245                      10$-                 ;  NOK
                        0610  1246                      20$-                 ;  NOA
                        0610  1247                      30$-                 ;  NOR (not used by RMS-32)
                        0610  1248                      40$-                 ;  PVN
                        0610  1249                  >                        ;
           024F  31     0614  1250          BRW     EXIT_SUCCESS             ; Message syntax is correct
                        0617  1251
                        0617  1252 ;
                        0617  1253 ; Process each field specified in the menu (optional).
                        0617  1254 ;
                        0617  1255
                        0617  1256 10$:     STORE_FIELD     NOK,1,K_FIX      ; Save number of keys field
                  05    061E  1257          RSB                              ;
                        061F  1258 20$:     STORE_FIELD     NOA,1,K_FIX      ; Save number of allocation areas field
                  05    0626  1259          RSB                              ;
                        0627  1260 30$:     STORE_FIELD     NOR,1,K_FIX      ; Save number of record descriptors field
                  05    062E  1261          RSB                              ;
                        062F  1262 40$:     STORE_FIELD     PVN,2,K_FIX      ; Save prologue version number field
                  05    0636  1263          RSB                              ;
```

```
                     063F  1265          .SBTTL  TIM_MSG - DECODE DATE AND TIME MESSAGE
                     063F  1266
                     063F  1267 ;++
                     063F  1268 ; Decode the operand fields of the Date and Time message.
                     063F  1269 ;--
                     063F  1270
                     063F  1271 TIM_MSG:                                ; Code segment of mainline
                     063F  1272
                     063F  1273 ;
                     063F  1274 ; For optional fields, apply default values as appropriate.
                     063F  1275 ;
                     063F  1276
                     063F  1277 ;       <there are no defaults to apply>
                     063F  1278
                     063F  1279 ;
                     063F  1280 ; Process the date and time menu field (optional).
                     063F  1281 ; Each bit set denotes that its associated field follows in the message.
                     063F  1282 ;
                     063F  1283
                     063F  1284          ASSUME  DAP$V_CDT+1 EQ DAP$V_RDT
                     063F  1285          ASSUME  DAP$V_RDT+1 EQ DAP$V_EDT
                     063F  1286          ASSUME  DAP$V_EDT+1 EQ DAP$V_RVN
                     063F  1287          ASSUME  DAP$V_RVN+1 EQ DAP$V_BDT
                     063F  1288          ASSUME  DAP$V_BDT+1 EQ DAP$V_PDT
                     063F  1289          ASSUME  DAP$V_PDT+1 EQ DAP$V_ADT
                     063F  1290
        58  086E'CF  9E  063F  1291          MOVAB   W^EXIT_SUCCESS,R8      ; All done if end-of-message
                     0644  1292          STORE_FIELD     TIMENU,2,K_EXT  ; Save date and time menu field
                     064B  1293          CHECK_MASKS     TIMENU,2        ; Validate bit options
        58  0846'CF  9E  0653  1294          MOVAB   W^ERROR_FORMAT,R8      ; Specify transfer address on EOM
            5C   66  3C  0658  1295          MOVZWL  (R6),AP                ; Copy menu to scratch register
                     065B  1296 TIM_LOOP:                               ;
            53   12  D0  065B  1297          MOVL    #18,R3                 ; Declare size of time (CDT, RDT, etc.)
                     065E  1298                                         ;  fields containing ASCII strings
    50  5C   07  00  EA  065E  1299          FFS     #0,#DAP$V_ADT+1,AP,R0  ; Get position of next bit set
                     0663  1300          $CLRBIT R0,AP                  ; Clear menu bit just found
              F1 AF  9F  0667  1301          PUSHAB  B^TIM_LOOP             ; Push return address on stack
                     066A  1302          $CASEB  SELECTOR=R0-           ; Next field:
                     066A  1303                  DISPL=<-               ;
                     066A  1304                      10$-               ;  CDT
                     066A  1305                      20$-               ;  RDT
                     066A  1306                      30$-               ;  EDT
                     066A  1307                      40$-               ;  RVN
                     066A  1308                      50$-               ;  BDT
                     066A  1309                      60$-               ;  PDT
                     066A  1310                      70$-               ;  ADT
                     066A  1311                  >                      ;
              01EF   31  067C  1312          BRW     EXIT_SUCCESS           ; Message syntax is correct
                     067F  1313
                     067F  1314 ;
                     067F  1315 ; Process each field specified in the menu (optional).
                     067F  1316 ;
                     067F  1317
                     067F  1318 10$:     STORE_FIELD     CDT,8,K_FIX,<M_SRCR3!M_DESC>
                     0686  1319                                         ; Save descriptor of creation
                     0686  1320                                         ;  date and time string
                35   11  0686  1321          BRB     100$               ;
```

```
                     0688  1322 20$:    STORE_FIELD     RDT,8,K_FIX,<M_SRCR3!M_DESC>
                     068F  1323                                         ; Save descriptor of revision
                     068F  1324                                         ;  date and time string
                2C   11   068F  1325          BRB     100$             ;
                     0691  1326 30$:    STORE_FIELD     EDT,8,K_FIX,<M_SRCR3!M_DESC>
                     0698  1327                                         ; Save descriptor of expiration
                     0698  1328                                         ;  date and time string
                23   11   0698  1329          BRB     100$             ;
                     069A  1330 40$:    STORE_FIELD     RVN,2,K_FIX     ; Save revision number field
                     05   06A1  1331          RSB                      ;
                     06A2  1332 50$:    STORE_FIELD     BDT,8,K_FIX,<M_SRCR3!M_DESC>
                     06A9  1333                                         ; Save descriptor of backup
                     06A9  1334                                         ;  date and time string
                12   11   06A9  1335          BRB     100$             ;
                     06AB  1336 60$:    STORE_FIELD     PDT,8,K_FIX,<M_SRCR3!M_DESC>
                     06B2  1337                                         ; Save descriptor of physical creation
                     06B2  1338                                         ;  date and time string
                09   11   06B2  1339          BRB     100$             ;
                     06B4  1340 70$:    STORE_FIELD     ADT,8,K_FIX,<M_SRCR3!M_DESC>
                     06BB  1341                                         ; Save descriptor of accessed
                     06BB  1342                                         ;  date and time string
                00   11   06BB  1343          BRB     100$             ;
                     06BD  1344
                     06BD  1345 ;+
                     06BD  1346 ; This sequence converts an ASCII time string into a 64-bit VMS binary time
                     06BD  1347 ; value. Note that the 64-bit result is stored in the quadword descriptor
                     06BD  1348 ; pointing to the time string on input.
                     06BD  1349 ;
                     06BD  1350 ; DAP defines network standard time as an 18 byte counted ASCII string in the
                     06BD  1351 ; format "dd-mmm-yybhh:mm:ss", whereas VMS uses a 23-byte ASCII string in the
                     06BD  1352 ; format "dd-mmm-yyyybhh:mm:ss.cc".
                     06BD  1353 ;-
                     06BD  1354
             5E  20  C2  06BD  1355 100$:   SUBL2   #<24+8>,SP          ; Allocate space from stack
          18 AE  17  D0  06C0  1356         MOVL    #<18+2+3>,24(SP)    ; Form descriptor of buffer to receive
          1C AE  5E  D0  06C4  1357         MOVL    SP,28(SP)           ;  altered ASCII string
     6E   04 B6  07  28  06C8  1358         MOVC3   #7,@4(R6),(SP)      ; Copy bytes 1-7 of input string
             36  61  91  06CD  1359         CMPB    (R1),#^A\6\         ; Compare decade against base decade
                 07  1A  06D0  1360         BGTRU   110$                ; Branch if '70 - '99
                     06D2  1361                                         ; Else it's '00 - '69
       83    3032  8F  B0  06D2  1362         MOVW    #^A\20\,(R3)+       ; Insert missing century digits
                 05  11  06D7  1363         BRB     120$                ; Continue
       83    3931  8F  B0  06D9  1364 110$:  MOVW    #^A\19\,(R3)+       ; Insert missing century digits
       63    61    0B  28  06DE  1365 120$:  MOVC3   #11,(R1),(R3)       ; Copy bytes 8-18 of input string
  83    2030302E  8F  D0  06E2  1366         MOVL    #^A\.00 \,(R3)+     ; Add hundredths of second digits
                     06E9  1367                                         ; Note R3 now points to time descriptor
          03 AE  20  8A  06E9  1368         BICB2   #^X20,3(SP)         ; Upcase 3-digit month string
          04 AE  20  8A  06ED  1369         BICB2   #^X20,4(SP)         ;  because $BINTIM objects to
          05 AE  20  8A  06F1  1370         BICB2   #^X20,5(SP)         ;  lowercase month
                     06F5  1371         $BINTIM_S-                      ; Convert ASCII time to binary time
                     06F5  1372                 TIMBUF=(R3)-            ;  Address of descriptor of ASCII string
                     06F5  1373                 TIMADR=(R6)             ;  Address of 64-bit result value
             5E  20  C0  0700  1374         ADDL2   #<24+8>,SP          ; Deallocate space from the stack
               01 50  E9  0703  1375         BLBC    R0,TIM_INVALID      ; Branch on conversion error
                     05  0706  1376         RSB                         ; Exit
                     0707  1377
                     0707  1378 ;
```

```
                     0707  1379 ; Branch here on exception condition.
                     0707  1380 ;
                     0707  1381
                     0707  1382 TIM_INVALID:                        ;
014B    31           0707  1383          BRW     ERROR_INVALID      ; Branch aid
```

```
                     070A  1385          .SBTTL  PRO_MSG - DECODE PROTECTION MESSAGE
                     070A  1386
                     070A  1387 ;++
                     070A  1388 ; Decode the operand fields of the Protection message.
                     070A  1389 ;--
                     070A  1390
                     070A  1391 PRO_MSG:                                ; Code segment of mainline
                     070A  1392
                     070A  1393 ;
                     070A  1394 ; For optional fields, apply default values as appropriate.
                     070A  1395 ;
                     070A  1396
          4C A9  69  7E 070A  1397          MOVAQ   (R9),DAP$Q_OWNER+4(R9)  ; Initialize descriptor
                     070E  1398
                     070E  1399 ;
                     070E  1400 ; Process the protection menu field (optional).
                     070E  1401 ; Each bit set denotes that its associated field follows in the message.
                     070E  1402 ;
                     070E  1403
                     070E  1404          ASSUME  DAP$V_OWNER+1 EQ DAP$V_PROSYS
                     070E  1405          ASSUME  DAP$V_PROSYS+1 EQ DAP$V_PROOWN
                     070E  1406          ASSUME  DAP$V_PROOWN+1 EQ DAP$V_PROGRP
                     070E  1407          ASSUME  DAP$V_PROGRP+1 EQ DAP$V_PROWLD
                     070E  1408
       58  086E'CF  9E 070E  1409          MOVAB   W^EXIT_SUCCESS,R8       ; All done if end-of-message
                     0713  1410          STORE_FIELD     PROMENU,2,K_EXT ; Save proection menu field
                     071A  1411          CHECK_MASKS     PROMENU,2       ; Validate bit options
       58  0846'CF  9E 0722  1412          MOVAB   W^ERROR_FORMAT,R8       ; Specify transfer address on EOM
           5C   66  3C 0727  1413          MOVZWL  (R6),AP                 ; Copy menu to scratch register
                     072A  1414 PRO_LOOP:                               ;
    50  5C  05  00  EA 072A  1415          FFS     #0,#DAP$V_PROWLD+1,AP,R0; Get position of next bit set
                     072F  1416          $CLRBIT R0,AP                   ; Clear menu bit just found
              F4 AF  9F 0733  1417          PUSHAB  B^PRO_LOOP              ; Push return address on stack
                     0736  1418          $CASEB  SELECTOR=R0-            ; Next field:
                     0736  1419                  DISPL=<-                ;
                     0736  1420                      10$-                ;  OWNER
                     0736  1421                      20$-                ;  PROSYS
                     0736  1422                      30$-                ;  PROOWN
                     0736  1423                      40$-                ;  PROGRP
                     0736  1424                      50$-                ;  PROWLD
                     0736  1425                  >                       ;
        0127  31     0744  1426          BRW     EXIT_SUCCESS            ; Message syntax is correct
                     0747  1427
                     0747  1428 ;
                     0747  1429 ; Process each field specified in the menu (optional).
                     0747  1430 ;
                     0747  1431
                     0747  1432 10$:     STORE_FIELD     OWNER,8,K_IMG,<M_DESC>
                     074E  1433                                         ; Save descriptor of file owner string
    28  66  91        074E  1434          CMPB    (R6),#40                ; Declare an error if owner string
        2C  1A       0751  1435          BGTRU   PRO_INVALID             ;  is too long
            05       0753  1436          RSB                             ;
                     0754  1437 20$:     STORE_FIELD     PROSYS,2,K_EXT  ; Save system protection field
        19  11       075B  1438          BRB     100$                    ;
                     075D  1439 30$:     STORE_FIELD     PROOWN,2,K_EXT  ; Save owner protection field
        10  11       0764  1440          BRB     100$                    ;
                     0766  1441 40$:     STORE_FIELD     PROGRP,2,K_EXT  ; Save group protection field
```

```
                     076D  1442                BRB     100$                    ;
                     076F  1443 50$:     STORE_FIELD     PROWLD,2,K_EXT  ; Save world protection field
                     0776  1444
                     0776  1445 ;
                     0776  1446 ; Perform common validity checks on data in the protection field being
                     0776  1447 ; processed.
                     0776  1448 ;
                     0776  1449
                     0776  1450 100$:    CHECK_MASKS     PROTECT,2       ; Validate bit options
                  05 077E  1451                RSB                     ;
                     077F  1452
                     077F  1453 ;
                     077F  1454 ; Branch here on exception condition.
                     077F  1455 ;
                     077F  1456
                     077F  1457 PRO_INVALID:                             ;
              00D3   31 077F  1458                BRW     ERROR_INVALID           ; Branch aid
```

```
                     0782  1460          .SBTTL  NAM_MSG - DECODE NAME MESSAGE
                     0782  1461
                     0782  1462 ;++
                     0782  1463 ; Decode the operand fields of the Name message.
                     0782  1464 ;--
                     0782  1465
                     0782  1466 NAM_MSG:                                ; Code segment of mainline
                     0782  1467
                     0782  1468 ;
                     0782  1469 ; For optional fields, apply default values as appropriate.
                     0782  1470 ;
                     0782  1471
  48 A9    69   7E   0782  1472          MOVAQ   (R9),DAP$Q_NAMESPEC+4(R9) ; Initialize descriptor
                     0786  1473
                     0786  1474 ;
                     0786  1475 ; Process the name type field (required).
                     0786  1476 ;
                     0786  1477
58   0846'CF    9E   0786  1478          MOVAB   W^ERROR_FORMAT,R8         ; Specify transfer address on EOM
                     078B  1479          STORE_FIELD     NAMETYPE,1,K_FIX; Save the name type field
                     0792  1480
                     0792  1481 ;
                     0792  1482 ; Process the name field (optional).
                     0792  1483 ;
                     0792  1484
58   086E'CF    9E   0792  1485          MOVAB   W^EXIT_SUCCESS,R8         ; All done if end-of-message
                     0797  1486          STORE_FIELD     NAMESPEC,8,K_IMG,<M_DESC>
                     079E  1487                                          ; Save descriptor of name
                     079E  1488                                          ;  specification string
  FF 8F    66   91   079E  1489          CMPB    (R6),#FWA$C_MAXQUOTED     ; Check for string too long
           02   1A   07A2  1490          BGTRU   NAM_INVALID               ; Branch on error
           68   17   07A4  1491          JMP     (R8)                      ; Message syntax is correct
                     07A6  1492
                     07A6  1493 ;
                     07A6  1494 ; Branch here on exception condition.
                     07A6  1495 ;
                     07A6  1496
                     07A6  1497 NAM_INVALID:                             ;
00AC    31   07A6  1498          BRW     ERROR_INVALID                  ; Branch aid
```

```
07A9  1500          .SBTTL  STORE_FIELD - STORE NEXT FIELD ROUTINES
07A9  1501
07A9  1502 ;++
07A9  1503 ; STORE_FIELD - invoked from the STORE_FIELD macro results in the execution
07A9  1504 ;       of one of the following routines:
07A9  1505 ;
07A9  1506 ;       STORE_EXT interprets the next field of the DAP message as an
07A9  1507 ;       extensible field of 1 to 16 bytes and stores the data portion of
07A9  1508 ;       the field in the designated field of the DAP control block.
07A9  1509 ;
07A9  1510 ;       STORE_FIX interprets the next field of the DAP message as a
07A9  1511 ;       fixed length field of 1 to 255 bytes and stores the string in the
07A9  1512 ;       designated field of the DAP control block.
07A9  1513 ;
07A9  1514 ;       STORE_IMG interprets the next field of the DAP message as an
07A9  1515 ;       image field of 1 to 256 bytes and stores the data portion of the
07A9  1516 ;       field in the designated field of the DAP control block.
07A9  1517 ;
07A9  1518 ;       STORE_ROM interprets the next field of the DAP message as a
07A9  1519 ;       binary field of 1 to 65535 bytes consisting of the rest of the
07A9  1520 ;       message and stores the string in the designated field of the DAP
07A9  1521 ;       control block.
07A9  1522 ;
07A9  1523 ; Calling Sequence:
07A9  1524 ;
07A9  1525 ;       BSBW    STORE_FIELD
07A9  1526 ;
07A9  1527 ; Input Parameters:
07A9  1528 ;
07A9  1529 ;       R3      Size in bytes of source field iff V_SRCR3 set and field is in
07A9  1530 ;               fixed length format
07A9  1531 ;       R8      Address of routine to execute if end-of-message encountered
07A9  1532 ;       R9      Address of DAP control block
07A9  1533 ;       R10     Address of last byte + 1 of DAP message being parsed
07A9  1534 ;       R11     Address of next byte of DAP message being parsed
07A9  1535 ;
07A9  1536 ;       In-line coded arguments:
07A9  1537 ;
07A9  1538 ;       Byte0   Size in bytes of the destination field in DAP control block
07A9  1539 ;       Byte1   Offset of destination field in DAP control block
07A9  1540 ;       Byte2   DAP field identifier (used to build DAP status code on error)
07A9  1541 ;       Byte3   Control byte to direct processing of DAP field:
07A9  1542 ;               Bits 0-3:                   ; Format of source field:
07A9  1543 ;                       K_EXT=  ^X00        ;  Extensible field format
07A9  1544 ;                       K_FIX=  ^X01        ;  Fixed length field format
07A9  1545 ;                       K_IMG=  ^X02        ;  Image field format
07A9  1546 ;                       K_ROM=  ^X03        ;  Rest-of-message field format
07A9  1547 ;               Bits 4-7:                   ; Field processing flags:
07A9  1548 ;                       M_DESC= ^X10        ;  Store only descriptor of SRC field
07A9  1549 ;                       M_TRUNC=^X20        ;  Truncate extra bytes if SRC field
07A9  1550 ;                                           ;  size is larger than DST field size
07A9  1551 ;                       M_SRCR3=^X40        ;  Size of SRC field is in R3
07A9  1552 ;                                           ;  (applicable only if K_FIX specified)
07A9  1553 ;
07A9  1554 ; Implicit Inputs:
07A9  1555 ;
07A9  1556 ;       None
```

```
                                07A9  1557 ;
                                07A9  1558 ; Output Parameters:
                                07A9  1559 ;
                                07A9  1560 ;       R0-R5   Destroyed
                                07A9  1561 ;       R6      Address of destination field in DAP control block
                                07A9  1562 ;       R7      Field ID value
                                07A9  1563 ;       R8-R10  Unchanged
                                07A9  1564 ;       R11     Updated next byte pointer
                                07A9  1565 ;
                                07A9  1566 ; Implicit Outputs:
                                07A9  1567 ;
                                07A9  1568 ;       The specified field of the DAP control block is updated.
                                07A9  1569 ;
                                07A9  1570 ; Completion Codes:
                                07A9  1571 ;
                                07A9  1572 ;       None
                                07A9  1573 ;
                                07A9  1574 ; Side Effects:
                                07A9  1575 ;
                                07A9  1576 ;       If end-of-message is encountered, control is given to the specified
                                07A9  1577 ;       action routine.
                                07A9  1578 ;
                                07A9  1579 ;       If a parse error is detected, control is given to an appropriate
                                07A9  1580 ;       error routine.
                                07A9  1581 ;
                                07A9  1582 ;       An exception exit described above, leaves the return address on the
                                07A9  1583 ;       stack.
                                07A9  1584 ;
                                07A9  1585 ;--
                                07A9  1586
                                07A9  1587 STORE_FIELD:                         ; Entry point
                                07A9  1588
                                07A9  1589 ;
                                07A9  1590 ; Obtain the in-line coded arguments, check for end-of-message, and transfer
                                07A9  1591 ; control to the appropriate routine.
                                07A9  1592 ;
                                07A9  1593
                          50    6E    D0  07A9  1594          MOVL    (SP),R0             ; Get address of in-line arguments
                          55    80    9A  07AC  1595          MOVZBL  (R0)+,R5            ; Get DST field size
                          56    80    9A  07AF  1596          MOVZBL  (R0)+,R6            ; Get DST field offset
                          56    59    C0  07B2  1597          ADDL2   R9,R6               ; Compute DST field address
                          57    80    9A  07B5  1598          MOVZBL  (R0)+,R7            ; Get DAP field ID value
                          52    80    9A  07B8  1599          MOVZBL  (R0)+,R2            ; Get control byte value
                          6E    50    D0  07BB  1600          MOVL    R0,(SP)             ; Bump return address past argument list
                          5A    5B    D1  07BE  1601          CMPL    R11,R10             ; Is there at least one byte left?
                                11    18  07C1  1602          BGEQ    10$                 ; Branch if end-of-message
                51    52    04    00    EF  07C3  1603          EXTZV   #0,#4,R2,R1         ; Get index of routine
                                07C8  1604          $CASEB  SELECTOR=R1-        ; Dispatch on field format:
                                07C8  1605                  DISPL=<-            ;
                                07C8  1606                      STORE_EXT-      ;  Extensible
                                07C8  1607                      STORE_FIX-      ;  Fixed length
                                07C8  1608                      STORE_IMG-      ;  Image
                                07C8  1609                      STORE_ROM-      ;  Rest-of-message
                                07C8  1610                  >                   ;
                                      68    17  07D4  1611 10$:     JMP     (R8)            ; Jump to designated EOM routine
```

```
                     07D6  1613          .SBTTL  STORE_EXT - STORE EXTENSIBLE FIELD
                     07D6  1614
                     07D6  1615 ;++
                     07D6  1616 ; This routine interprets the next field of the DAP message as an extensible
                     07D6  1617 ; field of 1 to 16 bytes where bit7 of each byte determines whether to
                     07D6  1618 ; continue (1) the field to the next byte or to terminate (0) the field.
                     07D6  1619 ; First, the source field is compressed in a work area (i.e., bit7 of each byte
                     07D6  1620 ; is discarded and the remaining bits are squeezed together). Then, the
                     07D6  1621 ; compressed string is copied to the specified destination field in the DAP
                     07D6  1622 ; control block.
                     07D6  1623 ;--
                     07D6  1624
                     07D6  1625          ASSUME  DAP$K_TEMP GE 16
                     07D6  1626
                     07D6  1627 STORE_EXT:                                   ; Code segment of STORE_FIELD
     54    0090 C9   9E  07D6  1628          MOVAB   DAP$L_TEMP(R9),R4       ; Get address of scratch work area
                50   D4  07DB  1629          CLRL    R0                      ; Initialize bit position index
                53   D4  07DD  1630          CLRL    R3                      ; Initialize byte count
           5A   5B   D1  07DF  1631 10$:      CMPL    R11,R10                 ; Error if end-of-message is reached
                62   18  07E2  1632          BGEQ    ERROR_FORMAT            ;  before end-of-field is reached
                53   D6  07E4  1633          INCL    R3                      ; Increment byte count
           10   53   D1  07E6  1634          CMPL    R3,#16                  ; Branch if SRC field is longer than
                5B   1A  07E9  1635          BGTRU   ERROR_FORMAT            ;  scratch work area
     64    07   50   6B  F0  07EB  1636          INSV    (R11),R0,#7,(R4)    ; Copy lower 7 bits of next byte
                50   07  C0  07F0  1637          ADDL2   #7,R0               ; Update bit position index
        E8 8B   07   E0  07F3  1638          BBS     #7,(R11)+,10$           ; Loop if field extends to next byte
     64    53   50   00  F0  07F7  1639          INSV    #0,R0,R3,(R4)       ; Zero fill rest of SRC field
                     07FC  1640                                              ;  (1 bit for each byte compressed)
                28   11  07FC  1641          BRB     MOVE_FIELD              ; Copy string to DST field
```

```
                     07FE  1643          .SBTTL  STORE_FIX - STORE FIXED LENGTH FIELD
                     07FE  1644
                     07FE  1645 ;++
                     07FE  1646 ; This routine interprets the next field of the DAP message as a fixed length
                     07FE  1647 ; field of 1 to 255 bytes and copies the string to the specified field in the
                     07FE  1648 ; DAP control block.
                     07FE  1649 ;--
                     07FE  1650
                     07FE  1651 STORE_FIX:                                  ; Code segment of STORE_FIELD
03 52    06    E0    07FE  1652          BBS     #V_SRCR3,R2,10$           ; Branch if SRC field size is in r3
   53    55    D0    0802  1653          MOVL    R5,R3                     ; DST field size = SRC field size
   01    53    D1    0805  1654 10$:     CMPL    R3,#1                     ; Branch if field is longer than
         07    1A    0808  1655          BGTRU   STORE_IMG1                ;  one byte
   66    8B    90    080A  1656          MOVB    (R11)+,(R6)               ; Store field in DAP control block
               05    080D  1657          RSB                               ; Exit
```

```
                     080E  1659          .SBTTL  STORE_IMG - STORE IMAGE FIELD
                     080E  1660
                     080E  1661 ;++
                     080E  1662 ; This routine interprets the next field of the DAP message as an image field
                     080E  1663 ; of 1 to 256 bytes where the first byte contains a count of the number of
                     080E  1664 ; data bytes to follow. The data portion of the field is copied to the
                     080E  1665 ; specified field of the DAP control block.
                     080E  1666 ;--
                     080E  1667
                     080E  1668 STORE_IMG:                                  ; Code segment of STORE_FIELD
          53   8B   9A  080E  1669          MOVZBL  (R11)+,R3                ; Get byte count of SRC field
                     0811  1670 STORE_IMG1:                                 ;
          54   5B   D0  0811  1671          MOVL    R11,R4                   ; Copy address of data string
    000C  5B   53   5A  F1  0814  1672          ACBL    R10,R3,R11,MOVE_FIELD    ; Ok if <R3+R11> LEQ <R10>
                  2A   11  081A  1673          BRB     ERROR_FORMAT             ; Error if not enough bytes in
                     081C  1674                                                 ;  message to contain field
```

```
                                 081C  1676          .SBTTL  STORE_ROM - STORE REST OF MESSAGE
                                 081C  1677
                                 081C  1678 ;++
                                 081C  1679 ; This routine interprets the next field of the DAP message as a binary field
                                 081C  1680 ; of 1 to 65535 bytes consisting of the rest of the message. The string is
                                 081C  1681 ; copied to the specified field of the DAP control block.
                                 081C  1682 ;--
                                 081C  1683
                                 081C  1684 STORE_ROM:                                  ; Code segment of STORE_FIELD
                      53   5A    5B   C3  081C  1685          SUBL3   R11,R10,R3        ; Compute SRC field size
                           54    5B   D0  0820  1686          MOVL    R11,R4            ; Copy SRC field address
                           5B    5A   D0  0823  1687          MOVL    R10,R11           ; Advance next byte pointer to EOM
                                 0826  1688
                                 0826  1689 ;
                                 0826  1690 ; <R3,R4> contains descriptor of SRC field, and
                                 0826  1691 ; <R5,R6> contains descriptor of DST field.
                                 0826  1692 ;
                                 0826  1693
                                 0826  1694 MOVE_FIELD:                                 ; Copy SRC field to DST field with
                                 0826  1695                                            ;  zero fill
                        18 52    04   E0  0826  1696          BBS     #V_DESC,R2,DESCRIPTOR ; Branch if only descriptor desired
                66   55   00    64   53   2C  082A  1697          MOVC5   R3,(R4),#0,R5,(R6) ; Move field to DAP control block
                                 0F   1B  0830  1698          BLEQU   20$               ; Done if all SRC bytes are copied
                                 0832  1699                                            ;  (i.e., SRC size LEQU DST size)
                           52    6E   D0  0832  1700          MOVL    (SP).R2           ; Get address of control flag
                                 0835  1701                                            ;  parameter + 1 (i.e., return address)
                      07 FF A2   05   E0  0835  1702          BBS     #V_TRUNC,-1(R2),20$ ; Done if extra bytes are to be
                                 083A  1703                                            ;  truncated; note:
                                 083A  1704                                            ;  R0 = # unmoved bytes
                                 083A  1705                                            ;  R1 = address of unmoved string
                           81    95   083A  1706 10$:      TSTB    (R1)+             ; Error if any unmoved bytes are
                           23    12   083C  1707          BNEQ    ERROR_UNSUPPORT   ;  non-zero
                        F9 50    F5   083E  1708          SOBGTR  R0,10$            ; Continue until all extra bytes
                                 0841  1709                                            ;  are checked
                                 05   0841  1710 20$:      RSB                       ; Exit
                                 0842  1711 DESCRIPTOR:                              ; DST field is a descriptor
                66   53    7D   0842  1712          MOVQ    R3,(R6)                   ; Store only quadword descriptor
                                 05   0845  1713          RSB                       ;  of SRC field and exit
```

```
                     0846  1715          .SBTTL  ERROR AND SUCCESS EXIT ROUTINES
                     0846  1716
                     0846  1717 ;++
                     0846  1718 ; Message parse has failed.
                     0846  1719 ; Build DAP Status message and exit to caller.
                     0846  1720 ;--
                     0846  1721
                     0846  1722 ERROR_FORMAT:                              ; Format of message in incorrect
          08    90   0846  1723          MOVB    #DAP$_FORMAT,-           ; Return MACCODE value
     1B A9           0848  1724                  DAP$B_DCODE_MAC(R9)       ;
01   10 A9    D1     084A  1725          CMPL    DAP$Q_MSG_BUF2(R9),#1    ; Check for one-byte message
          15    12   084E  1726          BNEQ    ERROR_COMMON             ; Take common path if not
     57   08    9A   0850  1727          MOVZBL  #DAP$_FLAGS,R7           ; Change to flags field ID code
                     0853  1728                                           ;  because format error was caused
                     0853  1729                                           ;  by no flags field in message
          10    11   0853  1730          BRB     ERROR_COMMON             ; Take common path
                     0855  1731 ERROR_INVALID:                            ; Field of message has invalid value
          09    90   0855  1732          MOVB    #DAP$_INVALID,-          ; Return MACCODE value
     1B A9           0857  1733                  DAP$B_DCODE_MAC(R9)       ;
          0A    11   0859  1734          BRB     ERROR_COMMON             ; Take common path
                     085B  1735 ERROR_SYNC:                               ; Message received is out-of-sequence
          0A    90   085B  1736          MOVB    #DAP$_MSG_SYNC,-         ; Return MACCODE value
     1B A9           085D  1737                  DAP$B_DCODE_MAC(R9)       ;
          04    11   085F  1738          BRB     ERROR_COMMON             ; Take common path
                     0861  1739 ERROR_UNSUPPORT:                          ; Field of message has unsupported value
          02    90   0861  1740          MOVB    #DAP$_UNSUPPORT,-        ; Return MACCODE value
     1B A9           0863  1741                  DAP$B_DCODE_MAC(R9)       ;
                     0865  1742 ERROR_COMMON:                             ; Common error exit sequence
19 A9     57    90   0865  1743          MOVB    R7,DAP$B_DCODE_FID(R9)   ; Return ID of field in error
     18 A9    94    0869  1744          CLRB    DAP$L_DCODE_STS(R9)      ; Indicate failure
          16    11   086C  1745          BRB     EXIT_COMMON              ; Join common exit code
                     086E  1746
                     086E  1747 ;++
                     086E  1748 ; Message parse has been successful so far, ...
                     086E  1749 ; Make additional validity checks.
                     086E  1750 ;--
                     086E  1751
                     086E  1752 EXIT_SUCCESS:                             ; Enter here on successful parse
     57   00    9A   086E  1753          MOVZBL  #DAP$_UNKNOWN,R7         ; Set field ID to 'unknown'
     5A   5B    D1   0871  1754          CMPL    R11,R10                  ; Branch if there are any unparsed
          D0    12   0874  1755          BNEQ    ERROR_FORMAT             ;  bytes left in DAP message
50     1A A9    9A   0876  1756          MOVZBL  DAP$B_DCODE_MSG(R9),R0   ; Get DAP message type
1C A9     50    E1   087A  1757          BBC     R0,DAP$L_MSG_MASK(R9),-  ; Branch if this is not a valid
          DC         087E  1758                  ERROR_SYNC               ;  message to receive
                     087F  1759
                     087F  1760 ;
                     087F  1761 ; Check for system specific fields in message header.
                     087F  1762 ;
                     087F  1763
     38 A9    D5   087F  1764          TSTL    DAP$Q_SYSPEC(R9)         ; Any system specific fields?
          4A    12   0882  1765          BNEQ    SSP_MINI_MSG             ; If yes, process them
                     0884  1766
                     0884  1767 ;
                     0884  1768 ; Update message descriptors in DAP control block.
                     0884  1769 ;
                     0884  1770
                     0884  1771 EXIT_COMMON:                              ; Common exit sequence
```

```
                14 A9   C3  0884  1772          SUBL3   DAP$Q_MSG_BUF2+4(R9),-  ; Compute size of message just parsed
          10 A9   5A        0887  1773                  R10,DAP$Q_MSG_BUF2(R9)  ;  and store it in descriptor
          0C A9   5A  D0    088A  1774          MOVL    R10,DAP$Q_MSG_BUF1+4(R9); Store address of next (blocked)
                            088E  1775                                          ;  message in buffer to parse
                10 A9   C2  088E  1776          SUBL2   DAP$Q_MSG_BUF2(R9),-    ; Store size of next (blocked)
                08 A9       0891  1777                  DAP$Q_MSG_BUF1(R9)      ;  message in buffer to parse
          50    18 A9   D0  0893  1778          MOVL    DAP$L_DCODE_STS(R9),R0  ; Get return status code
                            0897  1779          RET                             ; Return to caller
```

```
                     1781          .SBTTL  CHECK_MASKS - VALIDATE FIELD BIT OPTIONS
                     1782
                     1783 ;++
                     1784 ; CHECK_MASKS - invoked from the CHECK_MASKS macro examines the designated
                     1785 ;       field for invalid and unsupported bits set.
                     1786 ;
                     1787 ; Calling Sequence:
                     1788 ;
                     1789 ;       BSBW    CHECK_MASKS
                     1790 ;
                     1791 ; Input Parameters:
                     1792 ;
                     1793 ;       R6      Address of designated field in DAP control block
                     1794 ;       R7      Field ID value
                     1795 ;
                     1796 ;       In-line coded arguments:
                     1797 ;
                     1798 ;       Byte0   Size in bytes of the designated field in DAP control block
                     1799 ;       Byten   Mask of invalid bits (1-4 bytes; size specified in byte0)
                     1800 ;       Bytem   Mask of unsupported bits (1-4 bytes; size specified in byte0)
                     1801 ;
                     1802 ; Implicit Inputs:
                     1803 ;
                     1804 ;       None
                     1805 ;
                     1806 ; Output Parameters:
                     1807 ;
                     1808 ;       R0-R1   Destroyed
                     1809 ;       R6-R7   Unchanged
                     1810 ;
                     1811 ; Implicit Outputs:
                     1812 ;
                     1813 ;       The specified field of the DAP control block is validated.
                     1814 ;
                     1815 ; Completion Codes:
                     1816 ;
                     1817 ;       None
                     1818 ;
                     1819 ; Side Effects:
                     1820 ;
                     1821 ;       If any invalid or unsupported bits are set, control is given to an
                     1822 ;       appropriate error routine.
                     1823 ;
                     1824 ;       An exception exit described above, leaves the return address on the
                     1825 ;       stack.
                     1826 ;
                     1827 ;--
                     1828
                     1829 CHECK_MASKS:                                ; Entry point
50   6E   D0    0898 1830           MOVL    (SP),R0                 ; Get address of in-line arguments
51   80   9A    089B 1831           MOVZBL  (R0)+,R1                ; Get DST field size
                089E 1832           $CASEB  SELECTOR=R1-            ; Dispatch on field size:
                089E 1833                   BASE=#1-                ;
                089E 1834                   DISPL=<-                ;
                089E 1835                       10$-                ;  1-byte
                089E 1836                       20$-                ;  2-bytes
                089E 1837                       30$-                ;  Error
```

```
                089E  1838                      40$-                    ;  4-bytes
                089E  1839                    >                         ;
      9A  11    08AA  1840 30$:     BRB       ERROR_FORMAT            ; Value is out-of-range
80    66  93    08AC  1841 10$:     BITB      (R6),(R0)+              ; Check for invalid bits
      A4  12    08AF  1842          BNEQ      ERROR_INVALID           ; Branch on error
80    66  93    08B1  1843          BITB      (R6),(R0)+              ; Check for unsupported bits
      12  11    08B4  1844          BRB       50$                     ; Join common code
80    66  B3    08B6  1845 20$:     BITW      (R6),(R0)+              ; Check for invalid bits
      9A  12    08B9  1846          BNEQ      ERROR_INVALID           ; Branch on error
80    66  B3    08BB  1847          BITW      (R6),(R0)+              ; Check for unsupported bits
      08  11    08BE  1848          BRB       50$                     ; Join common code
80    66  D3    08C0  1849 40$:     BITL      (R6),(R0)+              ; Check for invalid bits
      90  12    08C3  1850          BNEQ      ERROR_INVALID           ; Branch on error
80    66  D3    08C5  1851          BITL      (R6),(R0)+              ; Check for unsupported bits
      97  12    08C8  1852 50$:     BNEQ      ERROR_UNSUPPORT         ; Branch on error
6E    50  D0    08CA  1853          MOVL      R0,(SP)                 ; Bump return address past argument list
          05    08CD  1854          RSB                               ; Exit
```

```
                     08CE  1856          .SBTTL  SSP_MINI_MSG - DECODE SYSTEM SPECIFIC FIELD
                     08CE  1857
                     08CE  1858 ;++
                     08CE  1859 ; Decode the system specific field found in the message header.
                     08CE  1860 ; Treat it as the operand portion of a mini-message that has a menu field
                     08CE  1861 ; and related fields.
                     08CE  1862 ;--
                     08CE  1863
                     08CE  1864 SSP_MINI_MSG:                            ; Code segment of mainline
           5A   DD   08CE  1865          PUSHL   R10                      ; Save end-of-message + 1 address
      5A   38 A9  7D 08D0  1866          MOVQ    DAP$Q_SYSPEC(R9),R10     ; R10 = size of syspec field
                     08D4  1867                                           ; R11 = address of start-of-field
         5A   5B  C0 08D4  1868          ADDL2   R11,R10                  ; R10 = address of end-of-field + 1
                     08D7  1869          $ZERO_FILL-                      ; Zero system specific work area
                     08D7  1870                  DST=DAP$L_SSPWA(R9)-     ;  in DAP control block
                     08D7  1871                  SIZE=#DAP$K_SSPWA        ;
                     08E1  1872
                     08E1  1873 ;
                     08E1  1874 ; Process the system specific menu field (optional).
                     08E1  1875 ; Each bit set denotes that its associated field follows in the message.
                     08E1  1876 ;
                     08E1  1877
                     08E1  1878          ASSUME  DAP$V_SSP_CAP+1 EQ DAP$V_SSP_FLG
                     08E1  1879
      58   092F'CF   9E 08E1  1880          MOVAB   W^SSP_SUCCESS,R8         ; Specify transfer address on EOM
                     08E6  1881          STORE_FIELD     SSP_MENU,2,K_EXT ; Save system specific menu field
                     08ED  1882          CHECK_MASKS     SSP_MEN,2        ; Validate bit options
      58   FF4D CF   9E 08F5  1883          MOVAB   W^ERROR_FORMAT,R8        ; Specify transfer address on EOM
           5C   66   3C 08FA  1884          MOVZWL  (R6),AP                  ; Copy menu to scratch register
                     08FD  1885 SSP_LOOP:                                ;
   50  5C   02   00   EA 08FD  1886          FFS     #0,#DAP$V_SSP_FLG+1,AP,R0; Get position of next bit set
                     0902  1887          $CLRBIT R0,AP                    ; Clear menu bit just found
              F4 AF   9F 0906  1888          PUSHAB  B^SSP_LOOP               ; Push return address on stack
                     0909  1889          $CASEB  SELECTOR=R0-             ; Next field:
                     0909  1890                  DISPL=<-                 ;
                     0909  1891                      10$-                 ;  SSP_CAP
                     0909  1892                      20$-                 ;  SSP_FLG
                     0909  1893                  >
   1C   11   0911  1894          BRB     SSP_SUCCESS              ; All fields parsed
             0913  1895
             0913  1896 ;
             0913  1897 ; Process each field specified in the menu (optional).
             0913  1898 ;
             0913  1899
             0913  1900 10$:     STORE_FIELD     SSP_CAP,4,K_EXT,<M_TRUNC>
             091A  1901                                           ; Save system specific capabilities
      05     091A  1902          RSB                              ;  field
             091B  1903 20$:     STORE_FIELD     SSP_FLG,4,K_EXT  ; Save system specific flags field
             0922  1904          CHECK_MASKS     SSP_FLG,4        ; Validate bit options
      05     092E  1905          RSB                              ;
             092F  1906
             092F  1907 ;
             092F  1908 ; System specific mini-message has been parsed successfully!
             092F  1909 ;
             092F  1910
             092F  1911 SSP_SUCCESS:                             ;
50 8ED0      092F  1912          POPL    R0                       ; Throw away return address on stack
```

```
                 0932  1913          POPL    R10                     ; Restore address of end-of-message + 1
FF4C    31       0935  1914          BRW     EXIT_COMMON             ; Exit here because this routine
                 0938  1915                                          ;  was entered from EXIT_SUCCESS
                 0938  1916
                 0938  1917          .END                            ; End of module
```

```
$$.PSECT_EP                     = 00000000
$$COUNT                         = 00000002
$$RMSTEST                       = 0000001A
$$RMS_PBUGCHK                   = 00000010
$$RMS_TBUGCHK                   = 00000008
$$RMS_UMODE                     = 00000004
ACK_MSG                           00000381 R      01
ALL_INVALID                       000005E5 R      01
ALL_LOOP                          00000575 R      01
ALL_MSG                           00000559 R      01
ATT_INVALID                       0000037E R      01
ATT_LOOP                          00000268 R      01
ATT_MSG                           0000022E R      01
CHECK_FILE_SYSTEM                 000001A7 R      01
CHECK_MASKS                       00000898 R      01
CHECK_OPERATING_SYSTEM            000001D2 R      01
CHECK_PROTOCOL_VERSION            00000132 R      01
CMP_INVALID                       000003BF R      01
CMP_MSG                           00000384 R      01
CNF_MSG                           000000EE R      01
DAP$B_ACCFUNC                     00000040
DAP$B_ACCOPT                      00000041
DAP$B_AID                         00000050
DAP$B_ALN                         00000044
DAP$B_AOP                         00000045
DAP$B_BITCNT                      00000035
DAP$B_BKS                         00000050
DAP$B_BKZ                         00000051
DAP$B_BLKCNT                      00000056
DAP$B_BSZ                         00000052
DAP$B_CMPFUNC                     00000040
DAP$B_CONFUNC                     00000040
DAP$B_CTLFUNC                     00000040
DAP$B_DAN                         00000070
DAP$B_DATATYPE                    00000044
DAP$B_DBS                         0000007C
DAP$B_DCODE_FID                   00000019
DAP$B_DCODE_MAC                   0000001B
DAP$B_DCODE_MSG                   0000001A
DAP$B_DECVER                      00000047
DAP$B_DTP                         00000071
DAP$B_ECONUM                      00000045
DAP$B_FAC                         00000042
DAP$B_FILESYS                     0000C043
DAP$B_FLAGS                       00000031
DAP$B_FLG                         00000048
DAP$B_FSZ                         00000051
DAP$B_IAN                         0000006E
DAP$B_IFS                         0000007D
DAP$B_KRF                         00000047
DAP$B_LAN                         0000006F
DAP$B_LEN256                      00000034
DAP$B_LENGTH                      00000033
DAP$B_LVL                         0000007E
DAP$B_NAMETYPE                    00000040
DAP$B_NOA                         00000045
DAP$B_NOK                         00000044
DAP$B_NOR                         00000046
DAP$B_NSG                         00000049
DAP$B_NUL                         0000006D
DAP$B_ORG                         00000045
DAP$B_OSTYPE                      00000042
DAP$B_RAC                         00000046
DAP$B_RAT                         00000047
DAP$B_REF                         0000006C
DAP$B_RFM                         00000046
DAP$B_SHR                         00000043
DAP$B_SIZ                         0000005C
DAP$B_SIZ_TMP                     0000004A
DAP$B_STREAMID                    00000032
DAP$B_TKS                         0000007F
DAP$B_TYPE                        00000030
DAP$B_USRNUM                      00000046
DAP$B_USRVER                      00000048
DAP$B_VERNUM                      00000044
DAP$B_X_FIELD                     00000024
DAP$C_BLN                         000000C0
DAP$K_ACC_MSG                   = 00000003
DAP$K_ACK_MSG                   = 00000006
DAP$K_ALLMENU_I                 = 0000FE10
DAP$K_ALLMENU_U                 = 00000000
DAP$K_ALL_MSG                   = 0000000B
DAP$K_ANY                       = 00000000
DAP$K_AOP_I                     = 000000F0
DAP$K_AOP_U                     = 00000000
DAP$K_ATTMENU_I                 = FFE08000
DAP$K_ATTMENU_U                 = 00000000
DAP$K_ATT_MSG                   = 00000002
DAP$K_BLN                         000000C0
DAP$K_BLS_D                     = 00000200
DAP$K_BN2                       = 00000002
DAP$K_BN4                       = 00000004
DAP$K_BN8                       = 00000007
DAP$K_BSZ_D                     = 00000008
DAP$K_CHANGE_B                  = 00000006
DAP$K_CHANGE_E                  = 00000007
DAP$K_CLOSE                     = 00000001
DAP$K_CMP_MSG                   = 00000007
DAP$K_CMWA                      = 000C0050
DAP$K_CNF_MSG                   = 00000001
DAP$K_CON_MSG                   = 00000005
DAP$K_COPOS11                   = 0000000D
DAP$K_CTL_MSG                   = 00000004
DAP$K_CYL                       = 00000001
DAP$K_DATATYP_D                 = 00000002
DAP$K_DATATYP_I                 = 00000004
DAP$K_DATATYP_U                 = 00000088
DAP$K_DAT_MSG                   = 00000008
DAP$K_DEV_I                     = FC000040
DAP$K_DEV_U                     = 00000000
DAP$K_DISCONN                   = 00000004
DAP$K_FCS11                     = 00000004
DAP$K_FIX                       = 00000001
DAP$K_FLAGS_I                   = 00000090
```

```
DAP$K_FLAGS_U           =  00000048
DAP$K_FLG_I             =  000000F8
DAP$K_FLG_U             =  00000000
DAP$K_FOP_I             =  F1021004
DAP$K_FOP_U             =  00002000
DAP$K_IAS               =  00000006
DAP$K_IDX               =  00000020
DAP$K_IN2               =  00000001
DAP$K_IN4               =  00000003
DAP$K_IN8               =  00000006
DAP$K_KEYMENU_I         =  FFFB1000
DAP$K_KEYMENU_U         =  00000000
DAP$K_KEY_MSG           =  0000000A
DAP$K_LBN               =  00000002
DAP$K_NAM_MSG           =  0000000F
DAP$K_NO_FS             =  00000006
DAP$K_ORG_D             =  00000000
DAP$K_PAC               =  00000005
DAP$K_PROMENU_I         =  0000FFE0
DAP$K_PROMENU_U         =  00000000
DAP$K_PROTECT_I         =  0000FE00
DAP$K_PROTECT_U         =  00000000
DAP$K_PRO_MSG           =  0000000E
DAP$K_P_OS              =  0000000E
DAP$K_RAT_I             =  00000020
DAP$K_RAT_U             =  000000C0
DAP$K_REL               =  00000010
DAP$K_RESET             =  00000003
DAP$K_RESPONSE          =  00000002
DAP$K_RFI               =  00000004
DAP$K_RFM_D             =  00000001
DAP$K_RMST1             =  00000001
DAP$K_RMS20             =  00000002
DAP$K_RMS32             =  00000003
DAP$K_RMS32S            =  0000000A
DAP$K_RSTS              =  00000002
DAP$K_RSX11D            =  00000005
DAP$K_RSX11M            =  00000004
DAP$K_RSX11MP           =  0000000C
DAP$K_RSX11S            =  00000003
DAP$K_RT11              =  00000001
DAP$K_RT11FS            =  00000005
DAP$K_SEQ               =  00000000
DAP$K_SEQ_ACC           =  00000000
DAP$K_SKIP_FILE         =  00000005
DAP$K_SSPWA             =  00000010
DAP$K_SSP_FLG_I         =  FFFFFFFE
DAP$K_SSP_FLG_U         =  00000000
DAP$K_SSP_MEN_I         =  0000FFFC
DAP$K_SSP_MEN_U         =  00000000
DAP$K_STG               =  00000000
DAP$K_STM               =  00000004
DAP$K_STMCR             =  00000006
DAP$K_STMLF             =  00000005
DAP$K_STS_MSG           =  00000009
DAP$K_SUMENU_I          =  0000FFF0
DAP$K_SUMENU_U          =  00000000
DAP$K_SUM_MSG           =  0000000C
DAP$K_TEMP              =  00000010
DAP$K_TERMINATE         =  00000008
DAP$K_TIMENU_I          =  0000FF80
DAP$K_TIMENU_U          =  00000000
DAP$K_TIM_MSG           =  0000000D
DAP$K_TOPS10            =  00000009
DAP$K_TOPS10FS          =  00000008
DAP$K_TOPS20            =  00000008
DAP$K_TOPS20FS          =  00000007
DAP$K_UDF               =  00000000
DAP$K_VAR               =  00000002
DAP$K_VAXELAN           =  0000000F
DAP$K_VAXVMS            =  00000007
DAP$K_VBN               =  00000003
DAP$K_VFC               =  00000003
DAP$L_ALQ1                 0000004C
DAP$L_ALQ2                 0000004C
DAP$L_ATTMENU              00000040
DAP$L_CMWA                 00000030
DAP$L_CRC_RSLT             00000020
DAP$L_DCODE_STS            00000018
DAP$L_DEV                  00000068
DAP$L_DVB                  00000078
DAP$L_EBK                  00000078
DAP$L_FOP1                 00000064
DAP$L_FOP2                 00000044
DAP$L_HBK                  00000074
DAP$L_KEYMENU              00000040
DAP$L_LOC                  00000048
DAP$L_MRN                  00000058
DAP$L_MSG_MASK             0000001C
DAP$L_RECNUM1              00000040
DAP$L_RECNUM2              00000048
DAP$L_ROP                  00000050
DAP$L_RVB                  00000074
DAP$L_SBN                  0000007C
DAP$L_SSPWA                00000080
DAP$L_SSP_CAP              00000088
DAP$L_SSP_FLG              00000084
DAP$L_STV                  0000004C
DAP$L_TEMP                 00000090
DAP$M_BITCNT            =  00000008
DAP$M_BLKCNT            =  00000040
DAP$M_CMPFMT            =  00000008
DAP$M_DFTSPEC           =  00000010
DAP$M_DMO               =  00002000
DAP$M_DSP_3NAM          =  00000200
DAP$M_EMBEDDED          =  00000010
DAP$M_GET               =  00000002
DAP$M_GO_NOGO           =  00000010
DAP$M_IMAGE             =  00000002
DAP$M_LOADIM            =  00000001
DAP$M_LSA               =  00000040
DAP$M_MACY11            =  00000080
DAP$M_MSE               =  00000010
DAP$M_SEGMENT           =  00000040
```

```
DAP$M_TMP1$             =  00000020
DAP$M_TMP2$             =  000000C0
DAP$M_TMP3$             =  00020000
DAP$M_TMP4$             =  01000000
DAP$M_TMP5$             =  F0000000
DAP$M_ZERO              =  00000080
DAP$Q_ADT                  00000070
DAP$Q_BDT                  00000060
DAP$Q_CDT                  00000048
DAP$Q_DCODE_FLG            00000000
DAP$Q_EDT                  00000058
DAP$Q_FILEDATA             00000044
DAP$Q_FILESPEC             00000044
DAP$Q_KEY                  00000048
DAP$Q_KNM                  00000064
DAP$Q_MSG_BUF1             00000008
DAP$Q_MSG_BUF2             00000010
DAP$Q_NAMESPEC             00000044
DAP$Q_OWNER                00000048
DAP$Q_PASSWORD             00000050
DAP$Q_PDT                  00000068
DAP$Q_RDT                  00000050
DAP$Q_RUNSYS               0000005C
DAP$Q_STX                  00000050
DAP$Q_SYSCAP               00000028
DAP$Q_SYSPEC               00000038
DAP$V_ADT               =  00000006
DAP$V_AID               =  00000006
DAP$V_ALN               =  00000001
DAP$V_ALQ1              =  00000006
DAP$V_ALQ2              =  00000005
DAP$V_AOP               =  00000002
DAP$V_BDT               =  00000004
DAP$V_BITCNT            =  00000003
DAP$V_BKS               =  00000007
DAP$V_BKZ               =  00000007
DAP$V_BLS               =  00000004
DAP$V_BSZ               =  0000000D
DAP$V_CDT               =  00000000
DAP$V_DAN               =  00000009
DAP$V_DATATYPE          =  00000000
DAP$V_DBS               =  0000000E
DAP$V_DEQ1              =  0000000B
DAP$V_DEQ2              =  00000008
DAP$V_DEV               =  0000000E
DAP$V_DFL               =  00000001
DAP$V_DTP               =  0000000A
DAP$V_DVB               =  0000000D
DAP$V_EBK               =  00000012
DAP$V_EDT               =  00000002
DAP$V_FCS               =  00000031
DAP$V_FFB               =  00000013
DAP$V_FLG               =  00000000
DAP$V_FOP1              =  0000000C
DAP$V_FSZ               =  00000008
DAP$V_GEQ_V41           =  00000020
DAP$V_GEQ_V42           =  00000021
DAP$V_GEQ_V52           =  00000022
DAP$V_GEQ_V54           =  00000023
DAP$V_GEQ_V56           =  00000024
DAP$V_GEQ_V60           =  00000025
DAP$V_GEQ_V70           =  00000026
DAP$V_GEQ_V71           =  00000027
DAP$V_HBK               =  00000011
DAP$V_IAN               =  00000007
DAP$V_IAS               =  0C00003B
DAP$V_IBS               =  0000000F
DAP$V_IFL               =  00000002
DAP$V_KNM               =  00000005
DAP$V_LAN               =  00000008
DAP$V_LEN256            =  00000002
DAP$V_LENGTH            =  00000001
DAP$V_LOC               =  00000003
DAP$V_LRL               =  00000010
DAP$V_LVL               =  00000010
DAP$V_MRL               =  00000012
DAP$V_MRN               =  00000009
DAP$V_MRS               =  00000005
DAP$V_NOA               =  00000001
DAP$V_NOK               =  00000000
DAP$V_NOR               =  00000002
DAP$V_NSG               =  00000003
DAP$V_NUL               =  00000006
DAP$V_ORG               =  0000C001
DAP$V_OWNER             =  00000000
DAP$V_PDT               =  00000005
DAP$V_PROGRP            =  00000003
DAP$V_PROOWN            =  00000002
DAP$V_PROSYS            =  00000001
DAP$V_PROWLD            =  00000004
DAP$V_PVN               =  00000003
DAP$V_P_OS              =  0000003C
DAP$V_RAT               =  00000003
DAP$V_RDT               =  00000001
DAP$V_REF               =  00000004
DAP$V_RFM               =  00000002
DAP$V_RMS               =  00000030
DAP$V_RSTS              =  00000039
DAP$V_RSX               =  0000003A
DAP$V_RT11              =  00000038
DAP$V_RUNSYS            =  0000000A
DAP$V_RVB               =  0000000B
DAP$V_RVN               =  00000003
DAP$V_SBN               =  00000014
DAP$V_SEGMENT           =  00000006
DAP$V_SSP_CAP           =  00000C00
DAP$V_SSP_FLG           =  00000001
DAP$V_STM_ONLY          =  00000032
DAP$V_STREAMID          =  00000000
DAP$V_SYSPEC            =  00000005
DAP$V_TKS               =  00000011
DAP$V_TOPS10            =  00000036
DAP$V_TOPS20            =  00000037
DAP$V_VAXELAN           =  00000035
```

```
DAP$V_VAXVMS            =  00000034
DAP$V_VMS_XPF1          =  0000002C
DAP$V_VOL               =  00000000
DAP$V_X_CHECK           =  00000001
DAP$V_X_RECNUM          =  00000000
DAP$W_ACLMENU              00000040
DAP$W_BLS                  00000048
DAP$W_BUFSIZ               00000040
DAP$W_CHECK                00000042
DAP$W_CTLMENU              00000044
DAP$W_DEQ1                 00000054
DAP$W_DEQ2                 00000052
DAP$W_DFL                  00000044
DAP$W_DISPLAY1             0000004C
DAP$W_DISPLAY2             00000054
DAP$W_FFB                  00000072
DAP$W_IFL                  00000046
DAP$W_LRL                  00000070
DAP$W_MRL                  00000072
DAP$W_MRS                  0000004A
DAP$W_PARTNER              00000006
DAP$W_POS                  0000004C
DAP$W_POS_TMP              0000004A
DAP$W_PROGRP               00000054
DAP$W_PROMENU              00000040
DAP$W_PROOWN               00000052
DAP$W_PROSYS               00000050
DAP$W_PROWLD               00000056
DAP$W_PVN                  00000042
DAP$W_RFA                  00000042
DAP$W_RVN                  00000042
DAP$W_SSP_MENU             00000080
DAP$W_STSCODE              00000040
DAP$W_SUMENU               00000040
DAP$W_TIMENU               00000040
DAP$W_VERSION              00000004
DAP$W_VOL                  00000042
DAP$_ADT                =  00000017
DAP$_AID                =  00000017
DAP$_ALLMENU            =  00000010
DAP$_ALN                =  00000012
DAP$_ALQ1               =  00000017
DAP$_ALQ2               =  00000016
DAP$_AOP                =  00000013
DAP$_ATTMENU            =  00000010
DAP$_BDT                =  00000015
DAP$_BKS                =  00000018
DAP$_BKZ                =  00000018
DAP$_BLS                =  00000015
DAP$_BSZ                =  0000001E
DAP$_BUFSIZ             =  00000010
DAP$_CDT                =  00000011
DAP$_CHECK              =  00000012
DAP$_CMPFUNC            =  00000010
DAP$_DAN                =  0000001C
DAP$_DATATYPE           =  00000011
DAP$_DBS                =  00000021
DAP$_DECVER             =  00000016
DAP$_DEQ1               =  0000001C
DAP$_DEQ2               =  00000019
DAP$_DEV                =  0000001F
DAP$_DFL                =  00000012
DAP$_DTP                =  0000001D
DAP$_DVB                =  00000020
DAP$_EBK                =  00000023
DAP$_ECONUM             =  00000014
DAP$_EDT                =  00000013
DAP$_FFB                =  00000024
DAP$_FILEDATA           =  00000011
DAP$_FILESYS            =  00000012
DAP$_FLAGS              =  00000008
DAP$_FLG                =  00000011
DAP$_FOP1               =  0000001D
DAP$_FOP2               =  00000011
DAP$_FORMAT             =  00000008
DAP$_FSZ                =  00000019
DAP$_HBK                =  00000022
DAP$_IAN                =  0000001A
DAP$_IBS                =  00000022
DAP$_IFL                =  00000013
DAP$_INVALID            =  00000009
DAP$_KEYMENU            =  00000010
DAP$_KNM                =  00000018
DAP$_LAN                =  0000001B
DAP$_LEN256             =  0000000B
DAP$_LENGTH             =  0000000A
DAP$_LOC                =  00000014
DAP$_LRL                =  00000021
DAP$_LVL                =  00000023
DAP$_MRL                =  00000025
DAP$_MRN                =  0000001A
DAP$_MRS                =  00000016
DAP$_MSG_SYNC           =  0000000A
DAP$_NAMESPEC           =  00000011
DAP$_NAMETYPE           =  00000010
DAP$_NOA                =  00000012
DAP$_NOK                =  00000011
DAP$_NOR                =  00000013
DAP$_NSG                =  00000014
DAP$_NUL                =  00000019
DAP$_ORG                =  00000012
DAP$_OSTYPE             =  00000011
DAP$_OWNER              =  00000011
DAP$_PDT                =  00000016
DAP$_POS_TMP            =  00000015
DAP$_PROGRP             =  00000014
DAP$_PROMENU            =  00000010
DAP$_PROOWN             =  00000013
DAP$_PROSYS             =  00000012
DAP$_PROWLD             =  00000015
DAP$_PVN                =  00000014
DAP$_RAT                =  00000014
DAP$_RDT                =  00000012
DAP$_RECNUM1            =  00000010
```

```
DAP$_RECNUM2                    =  00000013
DAP$_REF                        =  00000017
DAP$_RFA                        =  00000012
DAP$_RFM                        =  00000013
DAP$_RUNSYS                     =  0000001B
DAP$_RVB                        =  0000001E
DAP$_RVN                        =  00000014
DAP$_SBN                        =  00000025
DAP$_SIZ_TMP                    =  0C000016
DAP$_SSP_CAP                    =  0000000E
DAP$_SSP_FLG                    =  0000000E
DAP$_SSP_MENU                   =  0000000E
DAP$_STREAMID                   =  00000009
DAP$_STSCODE                    =  00000010
DAP$_STV                        =  00000014
DAP$_STX                        =  00000015
DAP$_SUMENU                     =  0000C010
DAP$_SYSCAP                     =  00000018
DAP$_SYSPEC                     =  0000000E
DAP$_TIMENU                     =  00000010
DAP$_TKS                        =  00000024
DAP$_TYPE                       =  00000008
DAP$_UNKNOWN                    =  00000000
DAP$_UNSUPPORT                  =  00000002
DAP$_USRNUM                     =  00000015
DAP$_USRVER                     =  00000017
DAP$_VERNUM                     =  00000013
DAP$_VOL                        =  00000011
DAT_MSG                           000003C2 R      01
DESCRIPTOR                        00000842 R      01
DISPATCH_TABLE                    000000C8 R      01
ERROR_COMMON                      00000865 R      01
ERROR_FORMAT                      00000846 R      01
ERROR_INVALID                     00000855 R      01
ERROR_SYNC                        0000085B R      01
ERROR_UNSUPPORT                   00000861 R      01
EXIT_COMMON                       00000884 R      01
EXIT_SUCCESS                      0000086E R      01
FWA$C_MAXQUOTED                 =  000000FF
HDR_INVALID                       000000C2 R      01
HDR_LOOP                          0000005E R      01
HDR_UNSUPPORT                     000000C5 R      01
HEADER                            00000022 R      01
KEY_INVALID                       00000556 R      01
KEY_LOOP                          00000445 R      01
KEY_MSG                           00000421 R      01
K_EXT                           =  00000000
K_FIX                           =  00000001
K_IMG                           =  00000002
K_ROM                           =  00000003
MOVE_FIELD                        00000826 R      01
M_DESC                          =  00000010
M_SRCR3                         =  00000040
M_TRUNC                         =  00000020
NAM_INVALID                       000007A6 R      01
NAM_MSG                           00000782 R      01
NTSDECODE_MSG                     00000000 RG     01
PRO_INVALID                       0000077F R      01
PRO_LOOP                          0000072A R      01
PRO_MSG                           0000070A R      01
SSP_LOOP                          000008FD R      01
SSP_MINI_MSG                      000008CE R      01
SSP_SUCCESS                       0000092F R      01
STORE_EXT                         000007D6 R      01
STORE_FIELD                       000007A9 R      01
STORE_FIX                         000007FE R      01
STORE_IMG                         0000080E R      01
STORE_IMG1                        00000811 R      01
STORE_ROM                         0000081C R      01
STS_MSG                           000003EE R      01
SUM_LOOP                          00000604 R      01
SUM_MSG                           000005E8 R      01
SYS$BINTIM                        ********  GX    01
TIM_INVALID                       00000707 R      01
TIM_LOOP                          0000065B R      01
TIM_MSG                           0000063F R      01
TMPT..                          =  00000926 R      01
TMP2..                          =  0000092E R      01
V_DESC                          =  00000004
V_SRCR3                         =  00000006
V_TRUNC                         =  00000005
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
NH$NETWORK	00000938 (2360.)	01 (1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE
ABS	000000C0 (192.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.06	00:00:00.71
Command processing	125	00:00:00.59	00:00:04.45
Pass 1	497	00:00:26.55	00:00:57.51
Symbol table sort	0	00:00:02.15	00:00:03.56
Pass 2	329	00:00:06.27	00:00:15.19
Symbol table output	63	00:00:00.40	00:00:01.04
Psect synopsis output	2	00:00:00.02	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1050	00:00:36.06	00:01:22.51

The working set limit was 2100 pages.
131920 bytes (258 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1334 non-local and 149 local symbols.
1917 source lines were read in Pass 1, producing 23 object records in Pass 2.
39 pages of virtual memory were used to define 38 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	26
_255DUA28:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	32

1801 GETS were required to define 32 macros.

There were no errors, warnings or information messages.

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

NT0DAPRMS
LIS

NT0GET
LIS

NT0NWASET
LIS

NT0EXTEND
LIS

NT0DECODE
LIS

NT0ENCODE
LIS

NT0ERASE
LIS

NT0DISCON
LIS

NT0DISPLY
LIS

NT0MISC
LIS