



```

FFFFFFFFF  AAAAAA  LL  DDDDDDD  EEEEEEEEE  CCCCCCCC  000000  DDDDDDD  EEEEEEEEE
FFFFFFFFF  AAAAAA  LL  DDDDDDD  EEEEEEEEE  CCCCCCCC  000000  DDDDDDD  EEEEEEEEE
FF          AA      AA  LL  DD          CC          00      00  DD          DD  EE
FF          AA      AA  LL  DD          CC          00      00  DD          DD  EE
FF          AA      AA  LL  DD          CC          00      00  DD          DD  EE
FF          AA      AA  LL  DD          CC          00      00  DD          DD  EE
FFFFFFFFF  AA      AA  LL  DD          CC          00      00  DD          DD  EEEEEEE
FFFFFFFFF  AA      AA  LL  DD          CC          00      00  DD          DD  EEEEEEE
FF          AAAAAAAAAA LL  DD          CC          00      00  DD          DD  EE
FF          AAAAAAAAAA LL  DD          CC          00      00  DD          DD  EE
FF          AA      AA  LL  DD          CC          00      00  DD          DD  EE
FF          AA      AA  LL  DD          CC          00      00  DD          DD  EE
FF          AA      AA  LL  DD          CC          00      00  DD          DD  EE
FF          AA      AA  LLLLLLLLLL DDDDDDD  EEEEEEEEE  CCCCCCCC  000000  DDDDDDD  EEEEEEEEE
FF          AA      AA  LLLLLLLLLL DDDDDDD  EEEEEEEEE  CCCCCCCC  000000  DDDDDDD  EEEEEEEEE

```

```

LL          IIIII  SSSSSSS
LL          IIIII  SSSSSSS
LL          II     SS
LL          II     SS
LL          II     SS
LL          II     SS
LL          II     SSSSS
LL          II     SSSSS
LL          II     SS
LL          II     SS
LL          II     SS
LL          II     SS
LLLLLLLLLL IIIII  SSSSSSS
LLLLLLLLLL IIIII  SSSSSSS

```

(2)	73	DECLARATIONS
(3)	129	LOCAL_MACRO_DEFINITIONS
(4)	188	FAL\$DECODE_MSG - DECODE DAP MESSAGE
(5)	268	HEADER - DECODE MESSAGE HEADER
(6)	413	DISPATCH_TABLE - CASE ON MESSAGE TYPE
(7)	450	CNF_MSG - DECODE CONFIGURATION MESSAGE
(8)	667	ATT_MSG - DECODE ATTRIBUTES MESSAGE
(9)	827	ACC_MSG - DECODE ACCESS MESSAGE
(10)	915	CTL_MSG - DECODE CONTROL MESSAGE
(11)	1048	CON_MSG - DECODE CONTINUE TRANSFER MESSAGE
(12)	1082	CMP_MSG - DECODE ACCESS COMPLETE MESSAGE
(13)	1142	DAT_MSG - DECODE DATA MESSAGE
(14)	1187	KEY_MSG - DECODE KEY DEFINITION MESSAGE
(15)	1340	ALL_MSG - DECODE ALLOCATION MESSAGE
(16)	1428	TIM_MSG - DECODE DATE AND TIME MESSAGE
(17)	1548	PRO_MSG - DECODE PROTECTION MESSAGE
(18)	1623	NAM_MSG - DECODE NAME MESSAGE
(19)	1664	STORE_FIELD - STORE NEXT FIELD ROUTINES
(20)	1779	STORE_EXT - STORE EXTENSIBLE FIELD
(21)	1809	STORE_FIX - STORE FIXED LENGTH FIELD
(22)	1825	STORE_IMG - STORE IMAGE FIELD
(23)	1842	STORE_ROM - STORE REST OF MESSAGE
(24)	1881	ERROR AND SUCCESS EXIT ROUTINES
(25)	1947	CHECK_MASKS - VALIDATE FIELD BIT OPTIONS
(26)	2024	SSP_MINI_MSG - DECODE SYSTEM SPECIFIC FIELD

```

0000 1      .TITLE FALDECODE - DECODE DAP MESSAGE
0000 2      .IDENT 'V04-000'
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: FAL (DECnet File Access Listener)
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, user mode
0000 38 :
0000 39 : Author: James A. Krycka,      Creation Date: 16-JUN-1977
0000 40 :
0000 41 : Modified By:
0000 42 :
0000 43 :     V03-008 JEJ0048      J E Johnson      13-Jul-1984
0000 44 :     Eliminate the check for a file name of 128 characters or
0000 45 :     more.
0000 46 :
0000 47 :     V03-007 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$K_P_OS.
0000 51 :
0000 52 :     V03-006 JAK0124      J A Krycka      06-SEP-1983
0000 53 :     Update new DAP$Q_DCODE_FLG status bits during parse of
0000 54 :     Configuration message.
0000 55 :
0000 56 :     V03-005 JAK0113      J A Krycka      22-JUN-1983
0000 57 :     Continuation of support for DAP V7.0 spec.

```

0000 58 :  
0000 59 :  
0000 60 :  
0000 61 :  
0000 62 :  
0000 63 :  
0000 64 :  
0000 65 :  
0000 66 :  
0000 67 :  
0000 68 :  
0000 69 :  
0000 70 :  
0000 71 :--

Add support for 64-bit binary keys.  
Also, set DAP\$V\_VMS\_XPFn flags as appropriate.

V03-004 KRM0105 K Malik 10-May-1983  
Update to support DAP V7.0 specification.

V03-003 KRM0085 K Malik 23-Mar-1983  
Add support for STMLF and STMCR file formats.  
Also, set DAP\$V\_GEQ\_V70 bit as appropriate.

V03-002 KRM0069 K Malik 23-Nov-1982  
Add support for \$RENAME service.

```

0000 73      .SBTTL  DECLARATIONS
0000 74
0000 75      :
0000 76      : Include Files:
0000 77      :
0000 78
0000 79      $DAPPLGDEF      : Define DAP prologue symbols
0000 80      $DAPHDRDEF     : Define DAP message header
0000 81      $DAPSSPDEF     : Define DAP system specific field
0000 82      $DAPCNFDEF     : Define DAP Configuration message
0000 83      $DAPATTDEF     : Define DAP Attributes message
0000 84      $DAPACCDEF     : Define DAP Access message
0000 85      $DAPCTLDEF     : Define DAP Control message
0000 86      $DAPCONDEF     : Define DAP Continue Transfer message
0000 87      $DAPACKDEF     : Define DAP Acknowledge message
0000 88      $DAPCMPDEF     : Define DAP Access Complete message
0000 89      $DAPDATDEF     : Define DAP Data message
0000 90      $DAPSTSDEF     : Define DAP Status message
0000 91      $DAPKEYDEF     : Define DAP Key Definition message
0000 92      $DAPALLDEF     : Define DAP Allocation message
0000 93      $DAPSUMDEF     : Define DAP Summary message
0000 94      $DAPTIMDEF     : Define DAP Date and Time message
0000 95      $DAPPRODEF     : Define DAP Protection message
0000 96      $DAPNAMDEF     : Define DAP Name message
0000 97      $DAPFIDDEF     : Define DAP field ID symbols
0000 98
0000 99      :
0000 100     : Macros:
0000 101     :
0000 102     : See next page for local macro definitions.
0000 103     :
0000 104     : Equated Symbols:
0000 105     :
0000 106
00000000 0000 107 K_EXT=0      : Extensible field format
00000001 0000 108 K_FIX=1      : Fixed length field format
00000002 0000 109 K_IMG=2      : Image field format
00000003 0000 110 K_ROM=3      : Rest-of-message field format
0000 111
00000004 0000 112 V_DESC=4      : Store descriptor of source field
00000005 0000 113 V_TRUNC=5     : Truncate source field if necessary
00000006 0000 114 V_SRCR3=6     : Source field size in R3
0000 115      : (applicable only if K_FIX specified)
0000 116
00000010 0000 117 M_DESC=<1@V_DESC>      : Mask for V_DESC
00000020 0000 118 M_TRUNC=<1@V_TRUNC>    : Mask for V_TRUNC
00000040 0000 119 M_SRCR3=<1@V_SRCR3>    : Mask for V_SRCR3
0000 120
0000 121      ASSUME DAP$Q_DCODE_FLG EQ 0
0000 122
0000 123      :
0000 124      : Own Storage:
0000 125      :
0000 126      : None
0000 127      :

```

```

0000 129      .SBTTL  LOCAL MACRO DEFINITIONS
0000 130
0000 131      :++
0000 132      : STORE_FIELD obtains the next field (if any) from the DAP message being parsed,
0000 133      : converts it to an appropriate format, and stores the result in the designated
0000 134      : field of the DAP control block. The arguments (coded in-line) are:
0000 135      :
0000 136      :     NAME = the symbolic name of DAP field used to generate symbolic DAP
0000 137      :           control block offset and field ID values.
0000 138      :     SIZE = the size in bytes of designated field in DAP control block.
0000 139      :     FORMAT= the format or structure of the source field. Choices are:
0000 140      :           K_EXT = extensible field (bit7 of each byte is used to signify
0000 141      :                   termination/continuation (0/1) of the field).
0000 142      :           K_FIX = fixed length field.
0000 143      :           K_IMG = image field (counted string).
0000 144      :           K_ROM = rest-of-message is taken as the next field.
0000 145      :     MASK = the flags to control field processing:
0000 146      :           M_DESC= store only descriptor of the source field.
0000 147      :           M_TRUNC=truncate extra bytes if SRC field size is larger than
0000 148      :                   DST field size (instead of declaring an error).
0000 149      :           M_SRCR3=size of source field is given in R3 (applicable only if
0000 150      :                   K_FIX is also specified).
0000 151      :--
0000 152
0000 153      .MACRO  STORE_FIELD NAME,SIZE=1,FORMAT=1,MASK=0
0000 154      BSBW  STORE_FIELD
0000 155      .BYTE  SIZE
0000 156      TMP1..=.
0000 157      .IIF EQ <SIZE-1>,      .BYTE  DAP$B_'NAME
0000 158      .IIF EQ <SIZE-2>,      .BYTE  DAP$W_'NAME
0000 159      .IIF EQ <SIZE-4>,      .BYTE  DAP$L_'NAME
0000 160      .IIF EQ <SIZE-6>,      .BYTE  DAP$W_'NAME
0000 161      .IIF EQ <SIZE-8>,      .BYTE  DAP$Q_'NAME
0000 162      TMP2..=.
0000 163      .IIF EQ <TMP2..-TMP1..>,.ERROR ;***** invalid field size *****;
0000 164      .BYTE  DAP$_'NAME
0000 165      .BYTE  FORMAT!MASK
0000 166      .ENDM  STORE_FIELD
0000 167
0000 168      :++
0000 169      : CHECK_MASKS examines the designated field of the DAP control block for
0000 170      : invalid and unsupported bits set. The arguments (coded in-line) are:
0000 171      :
0000 172      :     NAME = the symbolic name of the DAP field used to generate symbolic
0000 173      :           invalid and unsupported mask values.
0000 174      :     SIZE = the size in bytes of designated field in the DAP control block.
0000 175      :--
0000 176
0000 177      .MACRO  CHECK_MASKS NAME,SIZE=1
0000 178      BSBW  CHECK_MASKS
0000 179      .BYTE  SIZE
0000 180      TMP1..=.
0000 181      .IIF EQ <SIZE-1>,      .BYTE  DAP$K_'NAME'_I,DAP$K_'NAME'_U
0000 182      .IIF EQ <SIZE-2>,      .WORD  DAP$K_'NAME'_I,DAP$K_'NAME'_U
0000 183      .IIF EQ <SIZE-4>,      .LONG  DAP$K_'NAME'_I,DAP$K_'NAME'_U
0000 184      TMP2..=.
0000 185      .IIF EQ <TMP2..-TMP1..>,.ERROR ;***** invalid field size *****;

```

FALDECODE  
V04-000

- DECODE DAP MESSAGE  
LOCAL MACRO DEFINITIONS

H 3

16-SEP-1984 01:42:32 VAX/VMS Macro V04-00  
5-SEP-1984 01:16:49 [FAL.SRC]FALDECODE.MAR;1

Page 5  
(3)

FA  
VO

0000 186 .ENDM CHECK\_MASKS



```

0000 188      .SBTTL FALS$DECODE_MSG - DECODE DAP MESSAGE
00000000 189      .PSECT FALS$CODE      NOSHR,EXE,RD,NOWRT,BYTE
0000 190
0000 191      :++
0000 192      : Functional Description:
0000 193      :
0000 194      : FALS$DECODE_MSG is responsible for parsing a DAP message into its
0000 195      : constituent fields, storing these field values into corresponding fields
0000 196      : in the DAP control block, and finally performing validity checks on the
0000 197      : contents of the converted fields to screen out invalid and unsupported
0000 198      : bit options or field values.
0000 199      :
0000 200      : Each DAP message logically consists of two parts:
0000 201      : (1) a message header (called the operator field in DAP).
0000 202      : (2) a message body (called the operand field in DAP).
0000 203      : In addition, the message header may optionally contain a system
0000 204      : specific field for use by homogeneous systems which is treated as a
0000 205      : mini-message with discrete fields.
0000 206      :
0000 207      : Calling Sequence:
0000 208      :
0000 209      : CALLS #1,FALS$DECODE_MSG
0000 210      :
0000 211      : Input Parameters:
0000 212      :
0000 213      : 4(AP) Address of DAP control block
0000 214      :
0000 215      : Implicit Inputs:
0000 216      :
0000 217      : None
0000 218      :
0000 219      : Output Parameters:
0000 220      :
0000 221      : R0      Status code
0000 222      : R1      Destroyed
0000 223      :
0000 224      : Implicit Outputs:
0000 225      :
0000 226      : Various fields of the DAP control block are updated.
0000 227      :
0000 228      : Completion Codes:
0000 229      :
0000 230      : DAP$L_DCODE_STS is returned in R0 where bit 0 indicates success/failure.
0000 231      :
0000 232      : Side Effects:
0000 233      :
0000 234      : None
0000 235      :
0000 236      :--
OFFC 0000 237      :
OFFC 0000 238      : .ENTRY FALS$DECODE_MSG,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
OFFC 0002 239      : Entry point
OFFC 0002 240
OFFC 0002 241 :
OFFC 0002 242 : Perform initialization.
OFFC 0002 243 :
OFFC 0002 244

```

59	04	AC	D0	0002	245	MOVL	4(AP),R9	:	Get address of DAP control block
18	A9	01	D0	0006	246	MOVL	#1,DAP\$L_DCODE_STS(R9)	:	Assume successful parse
5A	08	A9	7D	000A	247	MOVQ	DAP\$Q_MSG_BUF1(R9),R10	:	R10 = size of message
				000E	248			:	R11 = address of start-of-message
10	A9	5A	7D	000E	249	MOVQ	R10,DAP\$Q_MSG_BUF2(R9)	:	Store in result descriptor
	5A	5B	C0	0012	250	ADDL2	R11,R10	:	R10 = address of end-of-message + 1
				0015	251	\$ZERO_FILL-		:	Zero current message work area
				0015	252		DST=DAP\$K_CMWA(R9)-	:	in DAP control block
				0015	253		SIZE=#DAP\$K_CMWA	:	
	24	A9	94	001F	254	CLRB	DAP\$B_X_FIELD(R9)	:	Clear 'explicitly specified' flags
				0022	255			:	
				0022	256			:	
				0022	257			:	
				0022	258			:	
				0022	259			:	
				0022	260			:	
				0022	261			:	
				0022	262			:	
				0022	263			:	
				0022	264			:	
				0022	265			:	
				0022	266			:	

:+ Note the current status of registers R8-R11:  
 R8 Currently undefined; later it will be used to contain the  
 address of the routine to execute on reaching end-of-message  
 R9 Address of DAP control block  
 R10 Address of end-of-message-buffer + 1; later it will contain  
 the address of end-of-message + 1  
 R11 Address of start-of-message; it will be continually updated  
 to contain the address of the next byte to parse  
 :-

```

0022 268      .SBTTL  HEADER - DECODE MESSAGE HEADER
0022 269
0022 270      ;++
0022 271      ; Decode the header of the DAP message (operator portion of the message).
0022 272      ; Then dispatch on message type to parse the body of the DAP message (operand
0022 273      ; portion of the message).
0022 274      ;--
0022 275
0022 276      HEADER:                                ; Continuation of mainline
0022 277
0022 278      ASSUME  DAP$K_CNF_MSG EQ 1
0022 279      ASSUME  DAP$K_ATT_MSG EQ 2
0022 280      ASSUME  DAP$K_ACC_MSG EQ 3
0022 281      ASSUME  DAP$K_CTL_MSG EQ 4
0022 282      ASSUME  DAP$K_CON_MSG EQ 5
0022 283      ASSUME  DAP$K_ACK_MSG EQ 6
0022 284      ASSUME  DAP$K_CMP_MSG EQ 7
0022 285      ASSUME  DAP$K_DAT_MSG EQ 8
0022 286      ASSUME  DAP$K_STS_MSG EQ 9
0022 287      ASSUME  DAP$K_KEY_MSG EQ 10
0022 288      ASSUME  DAP$K_ALL_MSG EQ 11
0022 289      ASSUME  DAP$K_SUM_MSG EQ 12
0022 290      ASSUME  DAP$K_TIM_MSG EQ 13
0022 291      ASSUME  DAP$K_PRO_MSG EQ 14
0022 292      ASSUME  DAP$K_NAM_MSG EQ 15
0022 293
0022 294      ;
0022 295      ; For optional fields, apply default values as appropriate.
0022 296      ;
0022 297
3C A9 69 7E 0022 298      MOVAQ  (R9),DAP$Q_SYSPEC+4(R9) ; Initialize descriptor
0026 299
0026 300      ;
0026 301      ; Process the DAP message type field (required).
0026 302      ;
0026 303
58 091D'CF 9E 0026 304      MOVAB  W^ERROR_FORMAT,R8      ; Specify transfer address on EOM
002B 305      STORE_FIELD  TYPE,1,K_FIX      ; Save type field
0032 306      TSTB   (R6)                      ; Test for valid value
0034 307      BEQL  10$                        ; Branch if out-of-range
OF 66 91 0036 308      CMPB   (R6),#DAP$K_NAM_MSG      ; Test for valid value
0039 309      BGTRU  10$                        ; Branch if out-of-range
1A A9 66 90 003B 310      MOVB   (R6),DAP$B_DCODE_MSG(R9); Return message type in status code
003F 311      BRB   20$                        ; Continue
08E8 31 0041 312 10$: BRW   ERROR_INVALID      ; Branch aid
0044 313
0044 314      ;+
0044 315      ; Process the DAP message flags field (required for most messages).
0044 316      ; This is a combination menu and bit option field whereby each bit set denotes
0044 317      ; that either an associated field is included in the message or a message
0044 318      ; option is specified.
0044 319      ;
0044 320      ; Note: If no flags field is found (i.e., its a one-byte message), the
0044 321      ; associated operand parse routine for the message will still be entered
0044 322      ; (via DISPATCH_TABLE) to determine if the message is valid and to apply
0044 323      ; operand field default values.
0044 324      ;-

```

```

0044 325
0044 326 ASSUME DAP$V_STREAMID+1 EQ DAP$V_LENGTH
0044 327 ASSUME DAP$V_LENGTH+1 EQ DAP$V_LEN256
0044 328 ASSUME DAP$V_LEN256+1 EQ DAP$V_BITCNT
0044 329 ASSUME DAP$V_BITCNT+2 EQ DAP$V_SYSPEC
0044 330 ASSUME DAP$V_SYSPEC+1 EQ DAP$V_SEGMENT
0044 331
58 00C8'CF 9E 0044 332 20$: MOVAB W*DISPATCH_TABLE,R8 ; Specify transfer address on EOM
0049 333 STORE_FIELD FLAGS,1,K_EXT ; Save flags field
0050 334 CHECK_MASKS FLAGS,1 ; Validate bit options
58 091D'CF 9E 0056 335 MOVAB W*ERROR_FORMAT,R8 ; Specify transfer address on EOM
5C 66 9A 005B 336 MOVZBL (R6),AP ; Copy menu to scratch register
005E 337 HDR_LOOP:
50 5C 07 00 EA 005E 338 FFS #0,#DAP$V_SEGMENT+1,AP,R0 ; Get position of next bit set
0063 339 $CLRBIT R0,AP ; Clear menu bit just found
F4 AF 9F 0067 340 PUSHAB B*HDR_LOOP ; Push return address on stack
006A 341 $CASEB SELECTOR=R0- ; Next field/option:
006A 342 DISPL=<-
006A 343 10$- ; STREAMID
006A 344 20$- ; LENGTH
006A 345 30$- ; LEN256
006A 346 ERROR_UNSupport- ; BITCNT
006A 347 ERROR_FORMAT- ; Reserved
006A 348 60$- ; SYSPEC
006A 349 ERROR_UNSupport- ; SEGMENT
006A 350 >
4A 11 007C 351 BRB DISPATCH_TABLE ; Message header syntax is correct
007E 352
007E 353 ;
007E 354 ; Process each field/option specified in the menu (optional).
007E 355 ;
007E 356
66 95 007E 357 10$: STORE_FIELD STREAMID,1,K_FIX ; Save data stream identification field
0085 358 TSTB (R6) ; Currently, multi-streams are
0087 359 ; not supported, so check value
3C 12 0087 360 BNEQ HDR_UNSupport ; Branch on error
05 0089 361 RSB
008A 362 20$: STORE_FIELD LENGTH,1,K_FIX ; Save length field
08 5C 02 E1 0091 363 BBC #DAP$V_LEN256,AP,35$ ; Branch if length value in header is
0095 364 ; expressed in one byte (i.e., there
05 0095 365 RSB ; is no LEN256 field present)
0096 366 30$: STORE_FIELD LEN256,1,K_FIX ; Save length extension field
009D 367
009D 368 ;
009D 369 ; Determine end-of-message based on operand length value in message header.
009D 370 ;
009D 371 ;
009D 372 ASSUME DAP$B_LENGTH+1 EQ DAP$B_LEN256
009D 373
50 33 A9 3C 009D 374 35$: MOVZWL DAP$B_LENGTH(R9),R0 ; Get operand length value
51 5B 50 C1 00A1 375 ADDL3 R0,R1T,R1 ; Compute new end-of-message + 1 address
5A 51 D1 00A5 376 CMPL R1,R10 ; Error if not enough bytes in buffer
18 1A 00AB 377 BGTRU HDR_INVALID ; to contain message
5A 51 D0 00AA 378 MOVL R1,R10 ; Update end-of-message address
05 00AD 379 RSB
00AE 380
00AE 381 ;+

```

```

00AE 382 : Suggested code to support the BITCNT field is shown below.
00AE 383 :
00AE 384 :40$: STORE_FIELD BITCNT,1,K_FIX : Save bit count field
00AE 385 : CMPB DAP$B_TYPE(R9),- : BITCNT field allowed only in
00AE 386 : : #DAP$R_DAT_MSG : Data message
00AE 387 : : BNEQ 80$ : Branch on error
00AE 388 : : CMPB (R6),#7 : Check for value in the range 0-7
00AE 389 : : BGTRU HDR_INVALID : Branch on error
00AE 390 : : RSB :
00AE 391 : :-
00AE 392 :
00AE 393 :60$: STORE_FIELD SYSPEC,8,K_IMG,<M_DESC>
00B5 394 : : Save descriptor of system specific
00B5 395 : : field
01 A9 91 00B5 396 : CMPB DAP$B_TYPE(R9),- : SYSPEC field not allowed in
01 01 00B8 397 : : #DAP$R_CNF_MSG : Configuration message
01 05 13 00B9 398 : : BEQL 80$ : Branch on error
01 69 34 E1 00BB 399 : : BBC #DAP$V_VAXVMS,(R9),80$ : SYSPEC field allowed only if
00BF 400 : : systems are homogeneous
00BF 401 : : RSB :
01 68 17 00C0 402 :80$: JMP (R8) : Branch to error_format routine
00C2 403 :
00C2 404 :
00C2 405 : Branch here on exception condition.
00C2 406 :
00C2 407 :
00C2 408 HDR_INVALID: :
0867 31 00C2 409 : BRW ERROR_INVALID : Branch aid
00C5 410 HDR_UN SUPPORT: :
0870 31 00C5 411 : BRW ERROR_UN SUPPORT : Branch aid

```

```

OOCB 413          .SBTTL DISPATCH_TABLE - CASE ON MESSAGE TYPE
OOCB 414
OOCB 415 :+
OOCB 416 : The DAP message header has been successfully parsed. Now dispatch on message
OOCB 417 : type to the appropriate code segment to process the body of the message.
OOCB 418 :
OOCB 419 : Note: The case table entries below should match the DAP$K_VALID_R2F message
OOCB 420 : mask!
OOCB 421 :-
OOCB 422
OOCB 423 DISPATCH TABLE:
57 00 9A OOCB 424   MOVZBL #DAP$ UNKNOWN,R7          ; Continuation of mainline
OOCB 425   $CASEB  SELECTOR=DAP$B TYPE(R9)-; Set field ID to 'unknown'
OOCB 426   BASE=#DAP$K_CNF_MSG-          ; Dispatch to message specific decode
OOCB 427   DISPL=<-                      ; routine to process:
OOCB 428   CNF_MSG-                      ; Configuration message
OOCB 429   ATT_MSG-                      ; Attributes message
OOCB 430   ACC_MSG-                      ; Access message
OOCB 431   CTL_MSG-                      ; Control message
OOCB 432   CON_MSG-                      ; Continue Transfer message
OOCB 433   ERROR_SYNC-                  ; Acknowledge message
OOCB 434   CMP_MSG-                      ; Access Complete message
OOCB 435   DAT_MSG-                      ; Data message
OOCB 436   ERROR_SYNC-                  ; Status message
OOCB 437   KEY_MSG-                      ; Key Definition message
OOCB 438   ALL_MSG-                      ; Allocation message
OOCB 439   ERROR_SYNC-                  ; Summary message
OOCB 440   TIM_MSG-                      ; Date and Time message
OOCB 441   PRO_MSG-                      ; Protection message
OOCB 442   NAM_MSG-                      ; Name message
OOCB 443   >
OOEE 444
OOEE 445 :
OOEE 446 : The message type value has been validated (bounds checked), so the type value
OOEE 447 : will not be outside the range of the case table above.
OOEE 448 :

```

```

OOEE 450      .SBTTL CNF_MSG - DECODE CONFIGURATION MESSAGE
OOEE 451
OOEE 452      :++
OOEE 453      : Decode the operand fields of the Configuration message.
OOEE 454      :--
OOEE 455
58 091D'CF 9E OOEE 456 CNF_MSG:                ; Code segment of mainline
OOEE 457      MOVAB  W^ERROR_FORMAT,R8          ; Specify transfer address on EOM
OOE3 458
OOE3 459      :
OOE3 460      : Process the buffer size field (required).
OOE3 461      :
OOE3 462
OOE3 463      STORE_FIELD  BUFSIZ,2,K_FIX      ; Save buffer size field
OOFA 464
OOFA 465      :
OOFA 466      : Process system software and DAP protocol version number fields (required).
OOFA 467      : These fields are for information purposes only; hence no bounds checking
OOFA 468      : on their values is performed.
OOFA 469      :
OOFA 470
OOFA 471      STORE_FIELD  OSTYPE,1,K_FIX      ; Save operating system type field
0101 472      STORE_FIELD  FILESYS,1,K_FIX     ; Save file system type field
0108 473      STORE_FIELD  VERNUM,1,K_FIX     ; Save DAP version # field
010F 474      STORE_FIELD  ECONUM,1,K_FIX     ; Save ECO version # field
0116 475      STORE_FIELD  USRNUM,1,K_FIX     ; Save user protocol version # field
011D 476      STORE_FIELD  DECVER,1,K_FIX     ; Save DEC software version # field
0124 477      STORE_FIELD  USRVER,1,K_FIX     ; Save user software version # field
012B 478
012B 479      :
012B 480      : Process the system capabilities field (required).
012B 481      : Bits set that are not defined in the DAP spec are ignored (not flagged as
012B 482      : an error) to facilitate compatibility with earlier implementations of DAP.
012B 483      :
012B 484
012B 485      STORE_FIELD  SYSCAP,8,K_EXT,<M_TRUNC>
0132 486      : Save system capabilities field
0132 487
50 44 A9 9A 0132 488 CHECK_PROTOCOL_VERSION:    ; Set appropriate DAP$Q_DCODE_FLG bits
50 50 08 78 0132 489 MOVZBL  DAP$B_VERNUM(R9),R0          ; Combine version number and ECO
50 45 A9 80 0136 490 ASHL   #8,R0,R0           ; number fields into one 16-bit
013A 491      ADBB2  DAP$B_ECONUM(R9),R0      ; value for easy comparison
013E 492
013E 493      :
013E 494      : Set status flag if partner implemented to DAP spec since V4.1.
013E 495      :
0401 8F 50 B1 013E 496
51 1F 0143 497      CMPW   R0,#^X0401          ; Did partner implement since DAP V4.1?
0145 498      BLSSU  10$                       ; Branch if not
0149 499      $SETBIT #DAP$V_GEQ_V41,(R9)    ; Set flag
0149 500
0149 501      :
0149 502      : Set status flag if partner implemented to DAP spec since V4.2.
0149 503      :
0402 8F 50 B1 0149 504
46 1F 014E 505      CMPW   R0,#^X0402          ; Did partner implement since DAP V4.2?
014E 506      BLSSU  10$                       ; Branch if not

```

```

0150 507      $SETBIT #DAP$V_GEQ_V42,(R9)      ; Set flag
0154 508
0154 509
0154 510      ; Set status flag if partner implemented to DAP spec since V5.2.
0154 511
0154 512
0502 8F 50   B1 0154 513      CMPW   R0,#^X0502      ; Did partner implement since DAP V5.2?
      3B 1F 0159 514      BLSSU  10$           ; Branch if not
      015B 515      $SETBIT #DAP$V_GEQ_V52,(R9)      ; Set flag
      015F 516
      015F 517
      015F 518      ; Set status flag if partner implemented to DAP spec since V5.4.
      015F 519
      015F 520
0504 8F 50   B1 015F 521      CMPW   R0,#^X0504      ; Did partner implement since DAP V5.4?
      30 1F 0164 522      BLSSU  10$           ; Branch if not
      0166 523      $SETBIT #DAP$V_GEQ_V54,(R9)      ; Set flag
      016A 524
      016A 525
      016A 526      ; Set status flag if partner implemented to DAP spec since V5.6.
      016A 527
      016A 528
0506 8F 50   B1 016A 529      CMPW   R0,#^X0506      ; Did partner implement since DAP V5.6?
      25 1F 016F 530      BLSSU  10$           ; Branch if not
      0171 531      $SETBIT #DAP$V_GEQ_V56,(R9)      ; Set flag
      0175 532
      0175 533
      0175 534      ; Set status flag if partner implemented to DAP spec since V6.0.
      0175 535
      0175 536
0600 8F 50   B1 0175 537      CMPW   R0,#^X0600      ; Did partner implement since DAP V6.0?
      1A 1F 017A 538      BLSSU  10$           ; Branch if not
      017C 539      $SETBIT #DAP$V_GEQ_V60,(R9)      ; Set flag
      0180 540
      0180 541
      0180 542      ; Set status flag if partner implemented to DAP spec since V7.0.
      0180 543
      0180 544
0700 8F 50   B1 0180 545      CMPW   R0,#^X0700      ; Did partner implement since DAP V7.0?
      0F 1F 0185 546      BLSSU  10$           ; Branch if not
      0187 547      $SETBIT #DAP$V_GEQ_V70,(R9)      ; Set flag
      018B 548
      018B 549
      018B 550      ; Set status flag if partner implemented to DAP spec since V7.1.
      018B 551
      018B 552
0701 8F 50   B1 018B 553      CMPW   R0,#^X0701      ; Did partner implement since DAP V7.1?
      04 1F 0190 554      BLSSU  10$           ; Branch if not
      0192 555      $SETBIT #DAP$V_GEQ_V71,(R9)      ; Set flag
      0196 556
      0196 557
      0196 558      ; Set experimental protocol flags from the low order four bits of the USRNUM
      0196 559      ; field only if partner is VAX/VMS.
      0196 560
      0196 561
      42 A9 91 0196 562 10$:  CMPB   DAP$B_OSTYPE(R9),-      ; Branch if partner is not VAX/VMS
      07 0199 563      #DAP$R_VAXVMS

```





```

01D2 621
01D2 622
01D2 623 : Set status flag pertaining to the type of operating system being used at the
01D2 624 : remote node.
01D2 625 :
01D2 626
01D2 627 $CASEB SELECTOR=DAP$B_OSTYPE(R9)-
01D2 628 BASE=#DAP$K_RTT1- : Type of remote operating system:
01D2 629 DISPL=<-
01D2 630 50$- : RT-11
01D2 631 60$- : RSTS/E
01D2 632 70$- : RSX-11S
01D2 633 70$- : RSX-11M
01D2 634 80$- : RSX-11D (classified as IAS)
01D2 635 80$- : IAS
01D2 636 10$- : VAX/VMS
01D2 637 40$- : TOPS-20
01D2 638 30$- : TOPS-10
01D2 639 100$- : Undefined
01D2 640 100$- : Undefined
01D2 641 70$- : RSX-11M-PLUS
01D2 642 40$- : TOPS-20 (using 2050/2060 front end)
01D2 643 90$- : P/OS
01D2 644 20$- : VAXELAN
01D2 645 >
34 i1 01F5 646 BRB 100$ BRB 100$ : Set VAX/VMS system flag
2E 11 01F7 647 10$: $SETBIT #DAP$V_VAXVMS,(R9) : Set VAX/VMS system flag
01FB 648 BRB 100$
28 11 01FD 649 20$: $SETBIT #DAP$V_VAXELAN,(R9) : Set VAXELAN system flag
0201 650 BRB 100$
22 11 0203 651 30$: $SETBIT #DAP$V_TOPS10,(R9) : Set TOPS-10 system flag
0207 652 BRB 100$
1C 11 0209 653 40$: $SETBIT #DAP$V_TOPS20,(R9) : Set TOPS-20 and COPOS11 system flag
020D 654 BRB 100$
16 11 020F 655 50$: $SETBIT #DAP$V_RT11,(R9) : Set RT-11 system flag
0213 656 BRB 100$
10 11 0215 657 60$: $SETBIT #DAP$V_RSTS,(R9) : Set RSTS/E system flag
0219 658 BRB 100$
0A 11 021B 659 70$: $SETBIT #DAP$V_RSX,(R9) : Set RSX-11S, RSX-11M, and RSX-11M-PLUS
021F 660 BRB 100$ : system flag
04 11 0221 661 80$: $SETBIT #DAP$V_IAS,(R9) : Set IAS and RSX-11D system flag
0225 662 BRB 100$
0227 663 90$: $SETBIT #DAP$V_P_OS,(R9) : Set PO/S system flag
022B 664
0717 31 022B 665 100$: BRW EXIT_SUCCESS : Message syntax is correct

```

```

022E 667      .SBTTL ATT_MSG - DECODE ATTRIBUTES MESSAGE
022E 668
022E 669      :++
022E 670      : Decode the operand fields of the Attributes message.
022E 671      :--
022E 672
022E 673      ATT_MSG:                               ; Code segment of mainline
022E 674
022E 675      :
022E 676      : For optional fields, apply default values as appropriate.
022E 677      :
022E 678
44 A9 02 90 022E 679      MOVB      #DAP$K_DATATYP_D,DAP$B_DATATYPE(R9)
45 A9 00 90 0232 680      MOVB      #DAP$K_ORG_D,DAP$B_ORG(R9)
46 A9 01 90 0236 681      MOVB      #DAP$K_RFM_D,DAP$B_RFM(R9)
48 A9 0200 8F 80 023A 682      MOVW     #DAP$K_BLS_D,DAP$W_BLS(R9)
52 A9 08 90 0240 683      MOVB      #DAP$K_BSZ_D,DAP$B_BSZ(R9)
60 A9 69 7E 0244 684      MOVAQ     (R9),DAP$Q_RUNSYS+4(R9) ; Initialize descriptor
0248 685
0248 686      :
0248 687      : Process the attributes menu field (optional).
0248 688      : Each bit set denotes that its associated field follows in the message.
0248 689      :
0248 690
0248 691      ASSUME   DAP$V_DATATYPE+1 EQ DAP$V_ORG
0248 692      ASSUME   DAP$V_ORG+1 EQ DAP$V_RFM
0248 693      ASSUME   DAP$V_RFM+1 EQ DAP$V_RAT
0248 694      ASSUME   DAP$V_RAT+1 EQ DAP$V_BLS
0248 695      ASSUME   DAP$V_BLS+1 EQ DAP$V_MRS
0248 696      ASSUME   DAP$V_MRS+1 EQ DAP$V_ALQ1
0248 697      ASSUME   DAP$V_ALQ1+1 EQ DAP$V_BKS
0248 698      ASSUME   DAP$V_BKS+1 EQ DAP$V_FSZ
0248 699      ASSUME   DAP$V_FSZ+1 EQ DAP$V_MRN
0248 700      ASSUME   DAP$V_MRN+1 EQ DAP$V_RUNSYS
0248 701      ASSUME   DAP$V_RUNSYS+1 EQ DAP$V_DEQ1
0248 702      ASSUME   DAP$V_DEQ1+1 EQ DAP$V_FOP1
0248 703      ASSUME   DAP$V_FOP1+1 EQ DAP$V_BSZ
0248 704      ASSUME   DAP$V_BSZ+1 EQ DAP$V_DEV
0248 705      ASSUME   DAP$V_DEV+2 EQ DAP$V_LRL
0248 706      ASSUME   DAP$V_LRL+1 EQ DAP$V_HBK
0248 707      ASSUME   DAP$V_HBK+1 EQ DAP$V_EBK
0248 708      ASSUME   DAP$V_EBK+1 EQ DAP$V_FFB
0248 709      ASSUME   DAP$V_FFB+1 EQ DAP$V_SBN
0248 710
58 0945'CF 9E 0248 711      MOVAB     W^EXIT_SUCCESS,R8      ; All done if end-of-message
024D 712      STORE_FIELD  ATTMENU,4,K_EXT ; Save attributes menu field
0254 713      CHECK_MASKS  ATTMENU,4      ; Validate bit options
58 091D'CF 9E 0260 714      MOVAB     W^ERROR_FORMAT,R8      ; Specify transfer address on EOM
5C 66 80 0265 715      MOVL      (R6),AP              ; Copy menu to scratch register
0268 716      ATT_LOOP:
50 5C 15 00 EA 0268 717      FFS      #0,#DAP$V_SBN+1,AP,R0 ; Get position of next bit set
026D 718      $CLRBIT  R0,AP              ; Clear menu bit just found
F4 AF 9F 0271 719      PUSHAB   B^ATT_LOOP          ; Push return address on stack
0274 720      $CASEB   SELECTOR=R0-      ; Next field:
0274 721      DISPL=<-
0274 722      10$-
0274 723      20$-
: DATATYPE
: ORG

```

```

0274 724 30$- : RFM
0274 725 40$- : RAT
0274 726 50$- : BLS
0274 727 60$- : MRS
0274 728 70$- : ALQ1
0274 729 80$- : BKS
0274 730 90$- : FSZ
0274 731 100$- : MRN
0274 732 110$- : RUNSYS
0274 733 120$- : DEQ1
0274 734 130$- : FOP1
0274 735 140$- : BSZ
0274 736 150$- : DEV
0274 737 ERROR_FORMAT- : Reserved
0274 738 170$- : LRL
0274 739 180$- : HBK
0274 740 190$- : EBK
0274 741 200$- : FFB
0274 742 210$- : SBN
06A0 31 02A2 744 BRW > EXIT_SUCCESS : Message syntax is correct
02A5 745 :
02A5 746 :
02A5 747 : Process each field specified in the menu (optional).
02A5 748 :
02A5 749 :
02A5 750 40$: STORE_FIELD RAT,1,K_EXT : Save record attributes field
02AC 751 CHECK_MASKS RAT,1 : Validate bit options
05 02B2 752 RSB :
02B3 753 50$: STORE_FIELD BLS,2,K_FIX : Save block size field
05 02BA 754 RSB :
02BB 755 60$: STORE_FIELD MRS,2,K_FIX : Save maximum record size field
05 02C2 756 RSB :
02C3 757 70$: STORE_FIELD ALQ1,4,K_IMG : Save allocation quantity field
05 02CA 758 RSB :
02CB 759 80$: STORE_FIELD BKS,1,K_FIX : Save bucket size field
05 02D2 760 RSB :
02D3 761 90$: STORE_FIELD FSZ,1,K_FIX : Save fixed control area size field
05 02DA 762 RSB :
02DB 763 100$: STORE_FIELD MRN,4,K_IMG : Save maximum record number field
05 02E2 764 RSB :
02E3 765 110$: STORE_FIELD RUNSYS,8,K_IMG,<M_DESC> : Save descriptor of run-time
02EA 766 : system string
05 02EA 767 RSB :
02EB 768 120$: STORE_FIELD DEQ1,2,K_FIX : Save default extension quantity field
05 02F2 770 RSB :
02F3 771 130$: STORE_FIELD FOP1,4,K_EXT : Save file options field
02FA 772 CHECK_MASKS FOP,4 : Validate bit options
05 0306 773 RSB :
0307 774 10$: STORE_FIELD DATATYPE,1,K_EXT : Save data type field
030E 775 CHECK_MASKS DATATYP,1 : Validate bit options
05 0314 776 RSB :
0315 777 20$: STORE_FIELD ORG,1,K_FIX : Save file organization field
031C 778 :
031C 779 ASSUME DAP$K_SEQ EQ 0
031C 780 ASSUME DAP$K_REL EQ 16

```

```

031C 781 ASSUME DAP$K_IDX EQ 32
031C 782
66 95 031C 783 TSTB (R6) ; Check for valid value
0C 13 031E 784 BEQL 25$ ; Branch if ok
10 66 91 0320 785 CMPB (R6),#DAP$K_REL ; Check for valid value
07 13 0323 786 BEQL 25$ ; Branch if ok
20 66 91 0325 787 CMPB (R6),#DAP$K_IDX ; Check for valid value
02 13 0328 788 BEQL 25$ ; Branch if ok
52 11 032A 789 BRB ATT_INVALID ; Branch on error
05 032C 790 25$: RSB ;
032D 791 30$: STORE_FIELD RFM,1,K_FIX ; Save record format field
0334 792
0334 793 ASSUME DAP$K_UDF EQ 0
0334 794 ASSUME DAP$K_FIX EQ 1
0334 795 ASSUME DAP$K_VAR EQ 2
0334 796 ASSUME DAP$K_VFC EQ 3
0334 797 ASSUME DAP$K_STM EQ 4
0334 798 ASSUME DAP$K_STMLF EQ 5
0334 799 ASSUME DAP$K_STMCR EQ 6
06 66 91 0334 800
45 1A 0337 801 CMPB (R6),#DAP$K_STMCR ; Check for valid value
05 0339 802 BGTRU ATT_INVALID ; Branch if out-of-range
033A 803 RSB ;
05 0341 804 140$: STORE_FIELD BSZ,1,K_FIX ; Save byte size field
0342 805 RSB ;
0349 806 150$: STORE_FIELD DEV,4,K_EXT ; Save device characteristics field
05 0355 807 CHECK_MASKS DEV,4 ; Validate bit options
0356 808 RSB ;
05 035D 809 170$: STORE_FIELD LRL,2,K_FIX ; Save longest record length field
035E 810 RSB ;
05 0365 811 180$: STORE_FIELD HBK,4,K_IMG ; Save highest virtual block number
0366 812 RSB ; field
05 036D 813 190$: STORE_FIELD EBK,4,K_IMG ; Save end-of-file block number field
036E 814 RSB ;
05 0375 815 200$: STORE_FIELD FFB,2,K_FIX ; Save first free byte in EOF block
0376 816 RSB ; field
05 037D 817 210$: STORE_FIELD SBN,4,K_IMG ; Save starting logical block number
037E 818 RSB ; field
037E 819 ;
037E 820 ;
037E 821 ; Branch here on exception condition.
037E 822 ;
037E 823 ;
05AB 31 037E 824 ATT_INVALID: ;
037E 825 BRW ERROR_INVALID ; Branch aid

```

```

0381 827 .SBTTL ACC_MSG - DECODE ACCESS MESSAGE
0381 828
0381 829 :++
0381 830 : Decode the operand fields of the Access message.
0381 831 :--
0381 832
0381 833 ACC_MSG: ; Code segment of mainline
0381 834
0381 835 :
0381 836 : For optional fields, apply default values as appropriate.
0381 837 :
0381 838 : Note: The default value for the DISPLAY field is applied after the ACCFUNC
0381 839 : field is processed.
0381 840 :
0381 841 :
42 A9 02 90 0381 842 MOVB #DAP$K_FAC_D,DAP$B_FAC(R9)
43 A9 00 90 0385 843 MOVB #DAP$K_SHR_D,DAP$B_SHR(R9)
48 A9 69 7E 0389 844 MOVAQ (R9),DAP$Q_FILESPEC+4(R9) ; Initialize descriptor
54 A9 69 7E 038D 845 MOVAQ (R9),DAP$Q_PASSWORD+4(R9) ; Initialize descriptor
0391 846
0391 847 :
0391 848 : Process the access function field (required).
0391 849 :
58 091D'CF 9E 0391 850
0391 851 MOVAB W*ERROR_FORMAT,R8 ; Specify transfer address on EOM
0396 852 STORE_FIELD ACCFUNC,1,K_FIX ; Save access function field
039D 853
039D 854 ASSUME DAP$K_OPEN EQ 1
039D 855 ASSUME DAP$K_CREATE EQ 2
039D 856 ASSUME DAP$K_RENAME EQ 3
039D 857 ASSUME DAP$K_ERASE EQ 4
039D 858 ASSUME DAP$K_DIR_LIST EQ 6
039D 859 ASSUME DAP$K_SUBMIT EQ 7
039D 860 ASSUME DAP$K_EXECUTE EQ 8
039D 861
039D 862 $CASEB SELECTOR=(R6),- ; Check for valid value
039D 863 BASE=#DAP$K_OPEN- ;
039D 864 DISPL=<- ; Function:
039D 865 10$- ; $OPEN
039D 866 10$- ; $CREATE
039D 867 20$- ; $RENAME
039D 868 20$- ; $ERASE
039D 869 ERROR_INVALID- ; Reserved
039D 870 20$- ; Directory list
039D 871 10$- ; Submit command file
039D 872 20$- ; Execute command file
039D 873 >
0578 31 0381 874 BRW ERROR_INVALID ; Value ou -of-range
01 80 0384 875 10$: MOVW #DAP$M_DSP_ATT,- ; Apply default DISPLAY value
4C A9 0386 876 DAP$W_DISPLAY1(R9) ; per ACCFUNC value
0388 877
0388 878 :
0388 879 : Process the access options and filespec fields (required).
0388 880 :
0388 881
0388 882 20$: STORE_FIELD ACCOPT,1,K_EXT ; Save access options field
038F 883 CHECK_MASKS ACCOPT,1 ; Validate bit options

```

```

03C5 884      STORE_FIELD  FILESPEC,8,K_IMG,<M_DESC>
03CC 885      ; Save descriptor of file
03CC 886      ; specification string
03CC 887
03CC 888 :
03CC 889 : Process the file access and file sharing fields (optional).
03CC 890 :
03CC 891 :
58 0945'CF 9E 03CC 892      MOVAB  W^EXIT_SUCCESS,R8      ; All done if end-of-message
03D1 893      STORE_FIELD  FAC,1,K_EXT      ; Save file access field
03D8 894      CHECK_MASKS  FAC,1          ; Validate bit options
03DE 895      STORE_FIELD  SHR,1,K_EXT      ; Save file sharing field
03E5 896      CHECK_MASKS  SHR,1          ; Validate bit options
03EB 897
03EB 898 :
03EB 899 : Process the display and password fields (optional).
03EB 900 :
03EB 901 :
03EB 902      STORE_FIELD  DISPLAY1,2,K_EXT; Save display attributes field
03F2 903      CHECK_MASKS  DISPLAY,2      ; Validate bit options
03FA 904      STORE_FIELD  PASSWORD,8,K_IMG,<M_DESC>
0401 905      ; Save descriptor of password string
68 17 0401 906      JMP      (R8)          ; Message syntax is correct
0403 907
0403 908 :
0403 909 : Branch here on exception condition.
0403 910 :
0403 911 :
0526 31 0403 912 ACC_INVALID:
0403 913      BRW      ERROR_INVALID      ; Branch aid

```

```

0406 915      .SBTTL CTL_MSG - DECODE CONTROL MESSAGE
0406 916
0406 917      :++
0406 918      : Decode the operand fields of the Control message.
0406 919      :--
0406 920
0406 921      CTL_MSG:                ; Code segment of mainline
0406 922
0406 923      :
0406 924      : For optional fields, apply default values as appropriate.
0406 925      :
0406 926
4C A9 69 7E 0406 927      MOVAQ   (R9),DAP$Q_KEY+4(R9)    ; Initialize descriptor
040A 928
040A 929      :
040A 930      : Process the control function field (required).
040A 931      :
040A 932
58 091D'CF 9E 040A 933      MOVAB   W^ERROR_FORMAT,R8      ; Specify transfer address on EOM
040F 934      STORE_FIELD CTLFUNC,1,K_FIX ; Save control function field
0416 935
0416 936      ASSUME   DAP$K_GET_READ EQ 1
0416 937      ASSUME   DAP$K_CONNECT EQ 2
0416 938      ASSUME   DAP$K_UPDATE EQ 3
0416 939      ASSUME   DAP$K_PUT_WRITE EQ 4
0416 940      ASSUME   DAP$K_DELETE EQ 5
0416 941      ASSUME   DAP$K_REWIND EQ 6
0416 942      ASSUME   DAP$K_TRUNCATE EQ 7
0416 943      ASSUME   DAP$K_RELEASE EQ 9
0416 944      ASSUME   DAP$K_FREE EQ 10
0416 945      ASSUME   DAP$K_EXTEND B EQ 11
0416 946      ASSUME   DAP$K_FLUSH EQ 12
0416 947      ASSUME   DAP$K_FIND EQ 14
0416 948      ASSUME   DAP$K_EXTEND E EQ 15
0416 949      ASSUME   DAP$K_DISPLAY EQ 16
0416 950      ASSUME   DAP$K_SPACE_FW EQ 17
0416 951      ASSUME   DAP$K_SPACE_BW EQ 18
0416 952
0416 953      $CASEB  SELECTOR=(R6)-                ; Check for valid value
0416 954      BASE=#DAP$K_GET_READ-
0416 955      DISPL=<-
0416 956      10$-
0416 957      10$-
0416 958      10$-
0416 959      10$-
0416 960      10$-
0416 961      10$-
0416 962      10$-
0416 963      ERROR_INVALID-
0416 964      10$-
0416 965      10$-
0416 966      10$-
0416 967      10$-
0416 968      ERROR_UNSUPPORTED-
0416 969      10$-
0416 970      10$-
0416 971      10$-
                                ; Function:
                                ; $GET or $READ
                                ; $CONNECT
                                ; $UPDATE
                                ; $PUT or $WRITE
                                ; $DELETE
                                ; $REWIND
                                ; $TRUNCATE
                                ; Reserved for $MODIFY
                                ; $RELEASE
                                ; $FREE
                                ; $EXTEND (beginning message of seq)
                                ; $FLUSH
                                ; Reserved for $NXTVOL--was defined
                                ; $FIND
                                ; $EXTEND (ending message of seq)
                                ; $DISPLAY

```



```

0416 972          10$-          : $SPACE (forward)
0416 973          10$-          : $SPACE (backward)
0416 974          >           : Reserved for checkpoint-file function
043E 975          :           : Reserved for recovery-get function
043E 976          :           : Reserved for recovery-put function
04EB 31 043E 977      BRW      ERROR_INVALID : Value is out-of-range
0441 978          :
0441 979          :
0441 980          : Process the control menu field (optional).
0441 981          : Each bit set denotes that its associated field follows in the message.
0441 982          :
0441 983          :
0441 984          ASSUME  DAP$V_RAC+1 EQ DAP$V_KEY
0441 985          ASSUME  DAP$V_KEY+1 EQ DAP$V_KRF
0441 986          ASSUME  DAP$V_KRF+1 EQ DAP$V_ROP
0441 987          ASSUME  DAP$V_ROP+2 EQ DAP$V_DISPLAY2
0441 988          ASSUME  DAP$V_DISPLAY2+1 EQ DAP$V_BLKCNT
0441 989          :
58 0945'CF 9E 0441 990 10$:  MOVAB  W^EXIT_SUCCESS,R8      : All done if end-of-message
                                STORE_FIELD CTLMENU,2,K_EXT : Save control menu field
044D 991          CHECK_MASKS CTLMENU,2          : Validate bit options
58 091D'CF 9E 0455 993  MOVAB  W^ERROR_FORMAT,R8      : Specify transfer address on EOM
5C 66 3C 045A 994  MOVZWL  (R6),AP          : Copy menu to scratch register
045D 995  CTL_LOOP:
50 5C 07 00 EA 045D 996  FFS      #0,#DAP$V_BLKCNT+1,AP,R0 : Get position of next bit set
                                $CLRBIT R0,AP          : Clear menu bit just found
                                F4 AF 9F 0466 998  PUSHAB  B^CTL_LOOP : Push return address on stack
                                0469 999  $CASEB  SELECTOR=R0- : Next field:
                                0469 1000  DISPL=<-
                                0469 1001  10$-          : RAC
                                0469 1002  20$-          : KEY
                                0469 1003  30$-          : KRF
                                0469 1004  40$-          : ROP
                                0469 1005  ERROR_FORMAT- : Reserved
                                0469 1006  60$-          : DISPLAY2
                                0469 1007  70$-          : BLKCNT
04C7 31 0469 1008      >           :
047B 1009      BRW      EXIT_SUCCESS          : Message syntax is correct
047E 1010          :
047E 1011          :
047E 1012          : Process the fields specified in the menu (optional).
047E 1013          :
047E 1014          :
047E 1015 10$:  STORE_FIELD  RAC,1,K_FIX      : Save record access field
0485 1016          :
0485 1017          ASSUME  DAP$K_SEQ_ACC EQ 0
0485 1018          ASSUME  DAP$K_KEY_ACC EQ 1
0485 1019          ASSUME  DAP$K_RFA_ACC EQ 2
0485 1020          ASSUME  DAP$K_SEQ_FILE EQ 3
0485 1021          ASSUME  DAP$K_BLK_VBN EQ 4
0485 1022          ASSUME  DAP$K_BLK_FILE EQ 5
0485 1023          :
05 66 91 0485 1024  CMPB  (R6),#DAP$K_BLK_FILE : Check for value too high
3D 1A 0488 1025  BGTRU  CTL_INVALID          : Branch on error
05 048A 1026  RSB
048B 1027 20$:  STORE_FIELD  KEY,8,K_IMG,<M_DESC>
0492 1028          : Save descriptor of key string

```

```

05 0492 1029 RSB
0493 1030 30$: STORE_F, D KRF,1,K_FIX : Save key of reference field
05 049A 1031 RSB
049B 1032 40$: STORE_FIELD ROP,4,K_EXT : Save record options field
04A2 1033 CHECK_MASKS ROP,4 : Validate bit options
05 04AE 1034 RSB
04AF 1035 60$: STORE_FIELD DISPLAY2,2,K_EXT : Save display attributes field
04B6 1036 CHECK_MASKS DISPLAY,2 : Validate bit options
05 04BE 1037 RSB
04BF 1038 70$: STORE_FIELD BLKCNT,1,K_FIX : Save block count field
05 04C6 1039 RSB
04C7 1040
04C7 1041 :
04C7 1042 : Branch here on exception condition.
04C7 1043 :
04C7 1044
04C7 1045 CTL_INVALID:
0462 31 04C7 1046 BRW ERROR_INVALID : Branch aid

```

```

04CA 1048      .SBTTL CON_MSG - DECODE CONTINUE TRANSFER MESSAGE
04CA 1049
04CA 1050      :++
04CA 1051      : Decode the operand fields of the Continue Transfer message.
04CA 1052      :--
04CA 1053
04CA 1054      CON_MSG:                                : Code segment of mainline
04CA 1055
04CA 1056      :
04CA 1057      : Process the continue transfer function field (required).
04CA 1058      :
04CA 1059
58  091D'CF  9E 04CA 1060      MOVAB  W^ERROR_FORMAT,R8      : Specify transfer address on EOM
04CF 1061      STORE_FIELD  CONFUNC,1,K_FIX : Save continue transfer function field
04D6 1062
04D6 1063      ASSUME  DAP$K_RETRY EQ 1
04D6 1064      ASSUME  DAP$K_SKIP_REC EQ 2
04D6 1065      ASSUME  DAP$K_ABORT EQ 3
04D6 1066      ASSUME  DAP$K_RESUME EQ 4
04D6 1067      ASSUME  DAP$K_QUIT EQ 5
04D6 1068
    66  95 04D6 1069      TSTB   (R6)                                : Branch if value is
    08  13 04D8 1070      BEQL   CON_INVALID                          : too low
05  66  91 04DA 1071      CMPB   (R6),#DAP$K_QUIT                          : or
    03  1A 04DD 1072      BGTRU  CON_INVALID                          : too high
0463  31 04DF 1073      BRW    EXIT_SUCCESS                          : Message syntax is correct
    04E2 1074
    04E2 1075      :
    04E2 1076      : Branch here on exception condition.
    04E2 1077      :
    04E2 1078
0447  31 04E2 1079      CON_INVALID:
    04E2 1080      BRW    ERROR_INVALID                          : Branch aid
  
```

```

04E5 1082      .SBTTL  CMP_MSG - DECODE ACCESS COMPLETE MESSAGE
04E5 1083
04E5 1084      :++
04E5 1085      : Decode the operand fields of the Access Complete message.
04E5 1086      :--
04E5 1087
04E5 1088      CMP_MSG:      ; Code segment of mainline
04E5 1089
04E5 1090      :
04E5 1091      : For optional fields, apply default values as appropriate.
04E5 1092      :
04E5 1093      :
04E5 1094      :      <there are no defaults to apply>
04E5 1095      :
04E5 1096      :
04E5 1097      : Process the access complete function field (required).
04E5 1098      :
04E5 1099
58  091D'CF  9E  04E5 1100      MOVAB  W^ERROR_FORMAT,R8      ; Specify transfer address on EOM
04EA 1101      STORE_FIELD  CMPFUNC,1,K_FIX ; Save access complete function field
04F1 1102
04F1 1103      ASSUME  DAPSK_CLOSE EQ 1
04F1 1104      ASSUME  DAPSK_RESPONSE EQ 2
04F1 1105      ASSUME  DAPSK_RESET EQ 3
04F1 1106      ASSUME  DAPSK_DISCONN EQ 4
04F1 1107      ASSUME  DAPSK_SKIP_FILE EQ 5
04F1 1108      ASSUME  DAPSK_CHANGE_B EQ 6
04F1 1109      ASSUME  DAPSK_CHANGE_E EQ 7
04F1 1110      ASSUME  DAPSK_TERMINATE EQ 8
04F1 1111
    66  95  04F1 1112      TSTB   (R6)      ; Branch if value is
    28  13  04F3 1113      BEQL   CMP_INVALID ; too low
08  66  91  04F5 1114      CMPB   (R6),#DAPSK_TERMINATE ; or
    26  1A  04F8 1115      BGTRU  CMP_INVALID ; too high
04FA 1116
04FA 1117      :
04FA 1118      : Process the file options field (optional).
04FA 1119      :
04FA 1120
58  0945'CF  9E  04FA 1121      MOVAB  W^EXIT_SUCCESS,R8      ; All done if end-of-message
04FF 1122      STORE_FIELD  FOP2,4,K_EXT ; Save file options field
0506 1123      CHECK_MASKS  FOP,4      ; Validate bit options
0512 1124
0512 1125      :
0512 1126      : Process the CRC checksum field (optional).
0512 1127      :
0512 1128
0512 1129      STORE_FIELD  CHECK,2,K_FIX ; Save CRC checksum field
0519 1130      $SETBIT #DAPSV_X_CHECK,- ; Denote field explicitly specified
0519 1131      DAPSB_X_FIELD(R9) ; (to distinguish between CRC value
051E 1132      : of zero and none specified)
    68  17  051E 1133      JMP   (R8)      ; Message syntax is correct
0520 1134
0520 1135      :
0520 1136      : Branch here on exception condition.
0520 1137      :
0520 1138

```

0409 31 0520 1139 CMP\_INVALID:  
0520 1140 BRW ERROR\_INVALID ; Branch aid

```

0523 1142      .SBTTL  DAT_MSG - DECODE DATA MESSAGE
0523 1143
0523 1144      :++
0523 1145      : Decode the operand fields of the Data message.
0523 1146      :--
0523 1147
0523 1148 DAT_MSG:      ; Code segment of mainline
0523 1149
0523 1150      :
0523 1151      : For optional fields, apply default values as appropriate.
0523 1152      :
0523 1153
48 A9 69 7E 0523 1154      MOVAQ   (R9),DAP$Q_FILEDATA+4(R9) ; Initialize descriptor
0527 1155
0527 1156      :
0527 1157      : Process the record number field (required).
0527 1158      :
0527 1159      : Note: Since there is no menu for the Data message (an unfortunate oversight
0527 1160      : in the DAP spec), this field must be present. However, it is necessary
0527 1161      : to distinguish between receiving a null value and a zero value, so that
0527 1162      : it can be determined whether the RECNUM field overrides the KEY field.
0527 1163      : To solve this problem, the DAP spec states that a byte count of zero
0527 1164      : for this image field means that no value has been specified. I.e.,
0527 1165      : <byte0 = 0> means ignore field, whereas <byte0 = 1 and byte1 = 0> means
0527 1166      : a value of zero overrides the KEY field value.
0527 1167      :
0527 1168
58 091D'CF 9E 0527 1169      MOVAB   W^ERROR_FORMAT,R8      ; Specify transfer address on EOM
5C 5B 01 C1 052C 1170      ADDL3   #1,R11,AP              ; Mark address of next byte + 1
0530 1171      STORE_FIELD RECNUM1,4,K_IMG ; Save record number field
5C 5B D1 0537 1172      CMPL   R11,AP                ; Branch if this image field was
05 05 13 053A 1173      BEQLU  10$                    ; exactly one byte long
053C 1174      $SETBIT #DAP$V_X_RECNUM,-   ; Denote field explicitly specified
053C 1175      DAP$B_X_FIELD(R9)         ; in message
0541 1176
0541 1177      :
0541 1178      : Process the file data field (optional for zero length record).
0541 1179      :
0541 1180
58 0945'CF 9E 0541 1181 10$: MOVAB   W^EXIT_SUCCESS,R8      ; All done if end-of-message
0546 1182      STORE_FIELD FILEDATA,8,K_ROM,<M_DESC> ; Save descriptor of user data string
054D 1183      ; (the record/block just received)
054D 1184      ;
68 17 054D 1185      JMP    (R8)                  ; Message syntax is correct

```

```

054F 1187 .SBTTL KEY_MSG - DECODE KEY DEFINITION MESSAGE
054F 1188
054F 1189 :++
054F 1190 : Decode the operand fields of the Key Definition message.
054F 1191 :--
054F 1192
054F 1193 KEY_MSG: ; Code segment of mainline
054F 1194
054F 1195 :
054F 1196 : For optional fields, apply default values as appropriate.
054F 1197 :
68 A9 69 7E 054F 1198
054F 1199 MOVAQ (R9),DAP$Q_KNM+4(R9) ; Initialize descriptor
0553 1200
0553 1201 :
0553 1202 : Process the key menu field (optional).
0553 1203 : Each bit set denotes that its associated field follows in the message.
0553 1204 :
0553 1205
0553 1206 ASSUME DAP$V_FLG+1 EQ DAP$V_DFL
0553 1207 ASSUME DAP$V_DFL+1 EQ DAP$V_IFL
0553 1208 ASSUME DAP$V_IFL+1 EQ DAP$V_NSQ
0553 1209 ASSUME DAP$V_NSQ+1 EQ DAP$V_REF
0553 1210 ASSUME DAP$V_REF+1 EQ DAP$V_KNM
0553 1211 ASSUME DAP$V_KNM+1 EQ DAP$V_NUL
0553 1212 ASSUME DAP$V_NUL+1 EQ DAP$V_IAN
0553 1213 ASSUME DAP$V_IAN+1 EQ DAP$V_LAN
0553 1214 ASSUME DAP$V_LAN+1 EQ DAP$V_DAN
0553 1215 ASSUME DAP$V_DAN+1 EQ DAP$V_DTP
0553 1216 ASSUME DAP$V_DTP+1 EQ DAP$V_RVB
0553 1217 ASSUME DAP$V_RVB+2 EQ DAP$V_DVB
0553 1218 ASSUME DAP$V_DVB+1 EQ DAP$V_DBS
0553 1219 ASSUME DAP$V_DBS+1 EQ DAP$V_IBS
0553 1220 ASSUME DAP$V_IBS+1 EQ DAP$V_LVL
0553 1221 ASSUME DAP$V_LVL+1 EQ DAP$V_TKS
0553 1222 ASSUME DAP$V_TKS+1 EQ DAP$V_MRL
0553 1223
58 0945'CF 9E 0553 1224 MOVAB W^EXIT_SUCCESS,R8 ; All done if end-of-message
0558 1225 STORE_FIELD KEYMENU,4,K_EXT ; Save key definition menu field
055F 1226 CHECK_MASKS KEYMENU,4 ; Validate bit options
58 091D'CF 9E 056B 1227 MOVAB W^ERROR_FORMAT,R8 ; Specify transfer address on EOM
5C 66 D0 0570 1228 MOVL (R6),AP ; Copy menu to scratch register
50 5C 13 00 EA 0573 1229 KEY_LOOP:
0573 1230 FFS #0,#DAP$V_MRL+1,AP,R0 ; Get position of next bit set
0578 1231 $CLRBIT R0,AP ; Clear menu bit just found
057C 1232 PUSHAB B^KEY_LOOP ; Push return address on stack
057F 1233 $CASEB SELECTOR=R0- ; Next field:
057F 1234 DISPL=<-
057F 1235 10$- ; FLG
057F 1236 20$- ; DFL
057F 1237 30$- ; IFL
057F 1238 40$- ; NSQ, POS, SIZ
057F 1239 50$- ; REF
057F 1240 60$- ; KNM
057F 1241 70$- ; NUL
057F 1242 80$- ; IAN
057F 1243 90$- ; LAN

```

			057F	1244		100\$-	:	DAN
			057F	1245		110\$-	:	DTP
			057F	1246		120\$-	:	RVB
			057F	1247		ERROR_FORMAT-	:	Reserved
			057F	1248		140\$-	:	DVB
			057F	1249		150\$-	:	DBS
			057F	1250		160\$-	:	IBS
			057F	1251		170\$-	:	LVL
			057F	1252		180\$-	:	TKS
			057F	1253		190\$-	:	MRL
			057F	1254	>		:	
0399		31	05A9	1255	BRW	EXIT_SUCCESS	:	Message syntax is correct
			05AC	1256			:	
			05AC	1257			:	
			05AC	1258	:		:	Process each field specified in the menu (optional).
			05AC	1259	:		:	
			05AC	1260	:		:	
			05AC	1261	10\$:	STORE_FIELD	FLG,1,K_EXT	Save key options field
			05B3	1262		CHECK_MASKS	FLG,1	Validate bit options
		05	05B9	1263		RSB		
			05BA	1264	20\$:	STORE_FIELD	DFL,2,K_FIX	Save data bucket fill quantity field
		05	05C1	1265		RSB		
			05C2	1266	30\$:	STORE_FIELD	IFL,2,K_FIX	Save index bucket fill quantity field
		05	05C9	1267		RSB		
			05CA	1268	40\$:	STORE_FIELD	NSG,1,K_FIX	Save number of key segments field
52	66	9A	05D1	1269		MOVZBC	(R6),R2	Use number of segments as loop count
	2C	13	05D4	1270		BEQL	47\$	Branch if zero
08	52	D1	05D6	1271		CMPL	R2,#8	Check for value too high
	28	1A	05D9	1272		BGTRU	49\$	Branch on error
50	4C	A9	3E	05DB	1273	MOVAV	DAP\$W_POS(R9),R0	Get address of POS array
51	5C	A9	9E	05DF	1274	MOVAB	DAP\$B_SIZ(R9),R1	Get address of SIZ array
		07	BB	05E3	1275	PUSHR	#*M<R0,R1,R2>	
				05E5	1276	STORE_FIELD	POS_TMP,2,K_FIX	Find next key segment size field
	01	BA	05EC	1277		POPR	#*M<R0>	
80	66	B0	05EE	1278		MOVW	(R6),(R0)+	Save it in array
	01	BB	05F1	1279		PUSHR	#*M<R0>	
				05F3	1280	STORE_FIELD	SIZ_TMP,1,K_FIX	Find next key segment size field
	07	BA	05FA	1281		POPR	#*M<R0,R1,R2>	
81	66	90	05FC	1282		MOVB	(R6),(R1)+	Save it in array
	E1	52	F5	05FF	1283	SOBGTR	R2,45\$	Branch if more segments to go
		05	0602	1284	47\$:	RSB		
		7F	11	0603	1285	BRB	KEY_INVALID	Branch aid
				0605	1286	STORE_FIELD	REF,1,K_FIX	Save key of reference field
		05	060C	1287		RSB		
			060D	1288	60\$:	STORE_FIELD	KNM,8,K_IMG,<M_DESC>	Save descriptor of key name string
			0614	1289				
28	66	91	0614	1290		CMPB	(R6),#40	Check for string too long
	68	1A	0617	1291		BGTRU	KEY_INVALID	Branch on error
20	66	91	0619	1292		CMPB	(R6),#32	Check for string too long
	66	1A	061C	1293		BGTRU	KEY_INVALID	Branch on error
		05	061E	1294		RSB		
			061F	1295	70\$:	STORE_FIELD	NUL,1,K_FIX	Save null key character field
		05	0626	1296		RSB		
			0627	1297	80\$:	STORE_FIELD	IAN,1,K_FIX	Save index area number field
		05	062E	1298		RSB		
			062F	1299	90\$:	STORE_FIELD	LAN,1,K_FIX	Save lowest level index area number field
			0636	1300				field



```

05 0636 1301 RSB ;
0637 1302 100$: STORE_FIELD DAN,1,K_FIX ; Save data area number field
05 063E 1303 RSB ;
063F 1304 110$: STORE_FIELD DTP,1,K_FIX ; Save key data type field
0646 1305 ;
0646 1306 ASSUME DAP$K_STG EQ 0
0646 1307 ASSUME DAP$K_IN2 EQ 1
0646 1308 ASSUME DAP$K_BN2 EQ 2
0646 1309 ASSUME DAP$K_IN4 EQ 3
0646 1310 ASSUME DAP$K_BN4 EQ 4
0646 1311 ASSUME DAP$K_PAC EQ 5
0646 1312 ASSUME DAP$K_IN8 EQ 6
0646 1313 ASSUME DAP$K_BN8 EQ 7
0646 1314 ;
07 66 91 0646 1315 CMPB (R6),#DAP$K_BN8 ; Check for value too high
39 1A 0649 1316 BGTRU KEY_INVALID ; Branch on error
05 064B 1317 RSB ;
064C 1318 120$: STORE_FIELD RVB,4,K_IMG ; Save root bucket start VBN field
05 0653 1319 RSB ;
0654 1320 140$: STORE_FIELD DVB,4,K_IMG ; Save first data bucket start VBN field
05 065B 1321 RSB ;
065C 1322 150$: STORE_FIELD DBS,1,K_FIX ; Save data bucket fill size field
05 0663 1323 RSB ;
0664 1324 160$: STORE_FIELD IBS,1,K_FIX ; Save index bucket fill size field
05 066B 1325 RSB ;
066C 1326 170$: STORE_FIELD LVL,1,K_FIX ; Save level of root buckets field
05 0673 1327 RSB ;
0674 1328 180$: STORE_FIELD TKS,1,K_FIX ; Save total key size field
05 067B 1329 RSB ;
067C 1330 190$: STORE_FIELD MRL,2,K_FIX ; Save minimum record length to contain
05 0683 1331 RSB ; key field
0684 1332 ;
0684 1333 ;
0684 1334 ; Branch here on exception condition.
0684 1335 ;
0684 1336 ;
02A5 31 0684 1337 KEY_INVALID: ;
0684 1338 BRW ERROR_INVALID ; Branch aid

```

```

0687 1340 .SBTTL ALL_MSG - DECODE ALLOCATION MESSAGE
0687 1341
0687 1342 :++
0687 1343 : Decode the operand fields of the Allocation message.
0687 1344 :--
0687 1345
0687 1346 ALL_MSG: ; Code segment of mainline
0687 1347
0687 1348 :
0687 1349 : For optional fields, apply default values as appropriate.
0687 1350 :
0687 1351 :
0687 1352 : <there are no defaults to apply>
0687 1353 :
0687 1354 :
0687 1355 : Process the allocation menu field (optional).
0687 1356 : Each bit set denotes that its associated field follows in the message.
0687 1357 :
0687 1358
0687 1359 ASSUME DAP$V_VOL+1 EQ DAP$V_ALN
0687 1360 ASSUME DAP$V_ALN+1 EQ DAP$V_AOP
0687 1361 ASSUME DAP$V_AOP+1 EQ DAP$V_LOC
0687 1362 ASSUME DAP$V_LOC+2 EQ DAP$V_ALQ2
0687 1363 ASSUME DAP$V_ALQ2+1 EQ DAP$V_AID
0687 1364 ASSUME DAP$V_AID+1 EQ DAP$V_BKZ
0687 1365 ASSUME DAP$V_BKZ+1 EQ DAP$V_DEQ2
0687 1366
58 0945'CF 9E 0687 1367 MOVAB W^EXIT_SUCCESS,R8 ; All done if end-of-message
0687 1368 STORE_FIELD ALLMENU,2,K_EXT ; Save allocation menu field
0693 1369 CHECK_MASKS ALLMENU,2 ; Validate bit options
58 091D'CF 9E 069B 1370 MOVAB W^ERROR_FORMAT,R8 ; Specify transfer address on EOM
SC 66 3C 06A0 1371 MOVZWL (R6),AP ; Copy menu to scratch register
50 5C 09 00 EA 06A3 1372 ALL_LOOP:
06A3 1373 FFS #0,#DAP$V_DEQ2+1,AP,R0 ; Get position of next bit set
06A8 1374 $CLRBIT R0,AP ; Clear menu bit just found
F4 AF 9F 06AC 1375 PUSHAB B^ALL_LOOP ; Push return address ... tack
06AF 1376 $CASEB SELECTOR=R0- ; Next field:
06AF 1377 DISPL=<-
06AF 1378 10%- ; VOL
06AF 1379 20%- ; ALN
06AF 1380 30%- ; AOP
06AF 1381 40%- ; LOC
06AF 1382 ERROR_FORMAT- ; Reserved
06AF 1383 60%- ; ALQ2
06AF 1384 70%- ; AID
06AF 1385 80%- ; BKZ
06AF 1386 90%- ; DEQ2
06AF 1387 >
027D 31 06C5 1388 BRW EXIT_SUCCESS ; Message syntax is correct
06C8 1389
06C8 1390 :
06C8 1391 : Process each field specified in the menu (optional).
06C8 1392 :
06C8 1393 :
06C8 1394 10$: STORE_FIELD VOL,2,K_FIX ; Save volume number field
05 06CF 1395 RSB ;
06D0 1396

```

```

06D0 1397 ASSUME DAP$K_ANY EQ 0
06D0 1398 ASSUME DAP$K_CYL EQ 1
06D0 1399 ASSUME DAP$K_LBN EQ 2
06D0 1400 ASSUME DAP$K_VBN EQ 3
06D0 1401 ASSUME DAP$K_RFI EQ 4
06D0 1402
03 66 91 06D0 1403 20$: STORE_FIELD ALN,1,K_FIX ; Save alignment options field
37 1A 06D7 1404 CMPB (R6),#DAP$K_VBN ; Check for value too high
05 06DA 1405 BGTRU ALL_INVALID ; Branch on error
05 06DC 1406 RSB ;
06DD 1407 30$: STORE_FIELD AOP,1,K_EXT ; Save allocation options field
06E4 1408 CHECK_MASKS AOP,1 ; Validate bit options
05 06EA 1409 RSB ;
06EB 1410 40$: STORE_FIELD LOC,4,K_IMG ; Save starting location field
05 06F2 1411 RSB ;
06F3 1412 60$: STORE_FIELD ALQ2,4,K_IMG ; Save allocation quantity field
05 06FA 1413 RSB ;
06FB 1414 70$: STORE_FIELD AID,1,K_FIX ; Save area identification field
05 0702 1415 RSB ;
0703 1416 80$: STORE_FIELD BKZ,1,K_FIX ; Save bucket size field
05 070A 1417 RSB ;
070B 1418 90$: STORE_FIELD DEQ2,2,K_FIX ; Save default extension quantity field
05 0712 1419 RSB ;
0713 1420 ;
0713 1421 ;
0713 1422 ; Branch here on exception condition.
0713 1423 ;
0713 1424 ;
0216 31 0713 1425 ALL_INVALID: ;
0713 1426 BRW ERROR_INVALID ; Branch aid

```

```

0716 1428 .SBTTL TIM_MSG - DECODE DATE AND TIME MESSAGE
0716 1429
0716 1430 :++
0716 1431 : Decode the operand fields of the Date and Time message.
0716 1432 :--
0716 1433
0716 1434 TIM_MSG: ; Code segment of mainline
0716 1435
0716 1436 :
0716 1437 : For optional fields, apply default values as appropriate.
0716 1438 :
0716 1439 :
0716 1440 : <there are no defaults to apply>
0716 1441 :
0716 1442 :
0716 1443 : Process the date and time menu field (optional).
0716 1444 : Each bit set denotes that its associated field follows in the message.
0716 1445 :
0716 1446
0716 1447 ASSUME DAPSV_CDT+1 EQ DAPSV_RDT
0716 1448 ASSUME DAPSV_RDT+1 EQ DAPSV_EDT
0716 1449 ASSUME DAPSV_EDT+1 EQ DAPSV_RVN
0716 1450 ASSUME DAPSV_RVN+1 EQ DAPSV_BDT
0716 1451 ASSUME DAPSV_BDT+1 EQ DAPSV_PDT
0716 1452 ASSUME DAPSV_PDT+1 EQ DAPSV_ADT
0716 1453
58 0945'CF 9E 0716 1454 MOVAB W^EXIT_SUCCESS,R8 ; All done if end-of-message
0716 1455 STORE_FIELD TIMENU,2,K_EXT ; Save date and time menu field
0722 1456 CHECK_MASKS TIMENU,2 ; Validate bit options
58 091D'CF 9E 072A 1457 MOVAB W^ERROR_FORMAT,R8 ; Specify transfer address on EOM
5C 66 3C 072F 1458 MOVZWL (R6),AP ; Copy menu to scratch register
0732 1459 TIM_LOOP:
53 12 D0 0732 1460 MOVL #18,R3 ; Declare size of time (CDT, RDT, etc.)
50 5C 07 00 EA 0735 1461 ; fields containing ASCII strings
0735 1462 FFS #0,#DAPSV_ADT+1,AP,R0 ; Get position of next bit set
073A 1463 $CLRBIT R0,AP ; Clear menu bit just found
F1 AF 9F 073E 1464 PUSHAB B^TIM_LOOP ; Push return address on stack
0741 1465 $CASEB SELECTOR=R0- ; Next field:
0741 1466 DISPL=<-
0741 1467 10%- ; CDT
0741 1468 20%- ; RDT
0741 1469 30%- ; EDT
0741 1470 40%- ; RVN
0741 1471 50%- ; BDT
0741 1472 60%- ; PDT
0741 1473 70%- ; ADT
0741 1474 >
01EF 31 0753 1475 BRW EXIT_SUCCESS ; Message syntax is correct
0756 1476
0756 1477 :
0756 1478 : Process each field specified in the menu (optional).
0756 1479 :
0756 1480
0756 1481 10$: STORE_FIELD CDT,8,K_FIX,<M_SRCR3!M_DESC>
075D 1482 ; Save descriptor of creation
075D 1483 ; date and time string
35 11 075D 1484 BRB 100$

```

```

075F 1485 20$: STORE_FIELD RDT,8,K_FIX,<M_SRCR3!M_DESC>
0766 1486 : Save descriptor of revision
0766 1487 : date and time string
2C 11 0766 1488 BRB 100$
0768 1489 30$: STORE_FIELD EDT,8,K_FIX,<M_SRCR3!M_DESC>
076F 1490 : Save descriptor of expiration
076F 1491 : date and time string
23 11 076F 1492 BRB 100$
0771 1493 40$: STORE_FIELD RVN,2,K_FIX : Save revision number field
05 0778 1494 RSB
0779 1495 50$: STORE_FIELD BDT,8,K_FIX,<M_SRCR3!M_DESC>
0780 1496 : Save descriptor of backup
0780 1497 : date and time string
12 11 0780 1498 BRB 100$
0782 1499 60$: STORE_FIELD PDT,8,K_FIX,<M_SRCR3!M_DESC>
0789 1500 : Save descriptor of physical creation
0789 1501 : date and time string
09 11 0789 1502 BRB 100$
078B 1503 70$: STORE_FIELD ADT,8,K_FIX,<M_SRCR3!M_DESC>
0792 1504 : Save descriptor of accessed
0792 1505 : date and time string
00 11 0792 1506 BRB 100$
0794 1507 :
0794 1508 :+
0794 1509 : This sequence converts an ASCII time string into a 64-bit VMS binary time
0794 1510 : value. Note that the 64-bit result is stored in the quadword descriptor
0794 1511 : pointing to the time string on input.
0794 1512 :
0794 1513 : DAP defines network standard time as an 18 byte counted ASCII string in the
0794 1514 : format 'dd-mmm-yybh:mm:ss', whereas VMS uses a 23-byte ASCII string in the
0794 1515 : format 'dd-mmm-yyybh:mm:ss.cc'.
0794 1516 :-
0794 1517 :-
6E 18 5E 20 C2 0794 1518 100$: SUBL2 #<24+8>,SP : Allocate space from stack
1C AE 17 D0 0797 1519 MOVL #<18+2+3>,24(SP) : Form descriptor of buffer to receive
04 B6 07 28 079B 1520 MOVL SP,28(SP) : altered ASCII string
36 61 91 079F 1521 MOVC3 #7,@4(R6),(SP) : Copy bytes 1-7 of input string
07 1A 07A4 1522 CMPB (R1),#^A\6\ : Compare decade against base decade
07A7 1523 BGTRU 110$ : Branch if '70 - '99
07A9 1524 : Else it's '00 - '69
83 3032 8F B0 07A9 1525 MOVW #^A\20\,(R3)+ : Insert missing century digits
05 11 07AE 1526 BRB 120$ : Continue
83 3931 8F 60 07B0 1527 110$: MOVW #^A\19\,(R3)+ : Insert missing century digits
63 61 0B 28 07B5 1528 120$: MOVC3 #11,(R1),(R3) : Copy bytes 8-18 of input string
83 2030302E 8F D0 07B9 1529 MOVL #^A\00 \,(R3)+ : Add hundredths of second digits
07C0 1530 : Note R3 now points to time descriptor
03 AE 20 8A 07C0 1531 BICB2 #^X20,3(SP) : Upcase 3-digit month string
04 AE 20 8A 07C4 1532 BICB2 #^X20,4(SP) : because $BINTIM objects to
05 AE 20 8A 07C8 1533 BICB2 #^X20,5(SP) : lowercase month
07CC 1534 $BINTIM_S- : Convert ASCII time to binary time
07CC 1535 -TIMBUF=(R3)- : Address of descriptor of ASCII string
07CC 1536 TIMADR=(R6) : Address of 64-bit result value
5E 01 20 C0 07D7 1537 ADDL2 #<24+8>,SP : Deallocate space from the stack
07DA 1538 BLBC R0,TIM_INVALID : Branch on conversion error
07DD 1539 RSB : Exit
07DE 1540
07DE 1541 ;

```

```
07DE 1542 ; Branch here on exception condition.  
07DE 1543 ;  
07DE 1544 ;  
014B 31 07DE 1545 TIM_INVALID: ;  
07DE 1546 BRW ERROR_INVALID ; Branch aid
```

```

07E1 1548      .SBTTL  PRO_MSG - DECODE PROTECTION MESSAGE
07E1 1549
07E1 1550      :++
07E1 1551      : Decode the operand fields of the Protection message.
07E1 1552      :--
07E1 1553
07E1 1554      PRO_MSG:          ; Code segment of mainline
07E1 1555
07E1 1556      :
07E1 1557      : For optional fields, apply default values as appropriate.
07E1 1558      :
07E1 1559
4C A9 69 7E 07E1 1560      MOVAQ   (R9),DAP$Q_OWNER+4(R9) ; Initialize descriptor
07E5 1561
07E5 1562      :
07E5 1563      : Process the protection menu field (optional).
07E5 1564      : Each bit set denotes that its associated field follows in the message.
07E5 1565      :
07E5 1566
07E5 1567      ASSUME  DAP$V_OWNER+1 EQ DAP$V_PROSYS
07E5 1568      ASSUME  DAP$V_PROSYS+1 EQ DAP$V_PROOWN
07E5 1569      ASSUME  DAP$V_PROOWN+1 EQ DAP$V_PROGRP
07E5 1570      ASSUME  DAP$V_PROGRP+1 EQ DAP$V_PROWLD
07E5 1571
58 0945'CF 9E 07E5 1572      MOVAB   W^EXIT_SUCCESS,R8          ; All done if end-of-message
07EA 1573      STORE_FIELD  PROMENU,2,K_EXT ; Save proection menu field
07F1 1574      CHECK_MASKS  PROMENU,2      ; Validate bit options
58 091D'CF 9E 07F9 1575      MOVAB   W^ERROR_FORMAT,R8          ; Specify transfer address on EOM
5C 66 3C 07FE 1576      MOVZWL  (R6),AP              ; Copy menu to scratch register
0801 1577      PRO_LOOP:
50 5C 05 00 EA 0801 1578      FFS     #0,#DAP$V_PROWLD+1,AP,R0 ; Get position of next bit set
0806 1579      $CLRBIT  R0,AP              ; Clear menu bit just found
F4 AF 9F 080A 1580      PUSHAB  B^PRO_LOOP              ; Push return address on stack
080D 1581      $CASEB  SELECTOR=R0-          ; Next field:
080D 1582      DISPL=<-
080D 1583      10$-                          ; OWNER
080D 1584      20$-                          ; PROSYS
080D 1585      30$-                          ; PROOWN
080D 1586      40$-                          ; PROGRP
080D 1587      50$-                          ; PROWLD
080D 1588      >
0127 31 081B 1589      BRW     EXIT_SUCCESS          ; Message syntax is correct
081E 1590
081E 1591      :
081E 1592      : Process each field specified in the menu (optional).
081E 1593      :
081E 1594
081E 1595      10$:  STORE_FIELD  OWNER,8,K_IMG,<M_DESC>
0825 1596      ; Save descriptor of file owner string
28 66 91 0825 1597      CMPB   (R6),#40              ; Declare an error if owner string
2C 1A 0828 1598      BGTRU  PRO_INVALID          ; is too long
05 082A 1599      RSB
082B 1600      20$:  STORE_FIELD  PROSYS,2,K_EXT ; Save system protection field
19 11 0832 1601      BRB     100$
0834 1602      30$:  STORE_FIELD  PROOWN,2,K_EXT ; Save owner protection field
10 11 083B 1603      BRB     100$
083D 1604      40$:  STORE_FIELD  PROGRP,2,K_EXT ; Save group protection field

```

```
07 11 0844 1605 BRB 100$  
0846 1606 50$: STORE_FIELD PROWLD,2,K_EXT ; Save world protection field  
084D 1607  
084D 1608 ;  
084D 1609 ; Perform common validity checks on data in the protection field being  
084D 1610 ; processed.  
084D 1611 ;  
084D 1612 ;  
084D 1613 100$: CHECK_MASKS PROTECT,2 ; Validate bit options  
05 0855 1614 RSB ;  
0856 1615 ;  
0856 1616 ;  
0856 1617 ; Branch here on exception condition.  
0856 1618 ;  
0856 1619 ;  
00D3 31 0856 1620 PRO_INVALID: ;  
0856 1621 BRW ERROR_INVALID ; Branch aid
```













```

                                .SBTTL STORE_IMG - STORE IMAGE FIELD
08E5 1825
08E5 1826
08E5 1827 :++
08E5 1828 : This routine interprets the next field of the DAP message as an image field
08E5 1829 : of 1 to 256 bytes where the first byte contains a count of the number of
08E5 1830 : data bytes to follow. The data portion of the field is copied to the
08E5 1831 : specified field of the DAP control block.
08E5 1832 :--
08E5 1833
08E5 1834 STORE_IMG:                                : Code segment of STORE_FIELD
53 8B 9A 08E5 1835     MOVZBL (R11)+,R3           : Get byte count of SRC field
08E8 1836 STORE_IMG1:
000C 5B 54 5B D0 08E8 1837     MOVL R11,R4           : Copy address of data string
53 5A F1 08E8 1838     ACBL R10,R3,R11,MOVE_FIELD : Ok if <R3+R11> LEQ <R10>
2A 11 08F1 1839     BRB ERROR_FORMAT           : Error if not enough bytes in
08F3 1840                                : message to contain field

```

FA  
SY  
SS  
SS  
ST  
ST  
ST  
ST  
ST  
ST  
SY  
TI  
TI  
TI  
TI  
TI  
V-  
V-  
V-  
PS  
--  
S  
FA  
PH  
--  
In  
Co  
Pa  
Sy  
Pa  
Sy  
Ps  
Cr  
As  
TH  
12  
TH  
20  
30

```

                                .SBTTL STORE_ROM - STORE REST OF MESSAGE
08F3 1842
08F3 1843
08F3 1844 :++
08F3 1845 : This routine interprets the next field of the DAP message as a binary field
08F3 1846 : of 1 to 65535 bytes consisting of the rest of the message. The string is
08F3 1847 : copied to the specified field of the DAP control block.
08F3 1848 :--
08F3 1849
08F3 1850 STORE_ROM: ; Code segment of STORE_FIELD
53 5A 5B C3 08F3 1851 SUBL3 R11,R10,R3 ; Compute SRC field size
54 5B DO 08F7 1852 MOVL R11,R4 ; Copy SRC field address
5B 5A DO 08FA 1853 MOVL R10,R11 ; Advance next byte pointer to EOM
08FD 1854
08FD 1855 :
08FD 1856 : <R3,R4> contains descriptor of SRC field, and
08FD 1857 : <R5,R6> contains descriptor of DST field.
08FD 1858 :
08FD 1859
08FD 1860 MOVE_FIELD: ; Copy SRC field to DST field with
08FD 1861 ; zero fill
66 55 00 18 52 04 E0 08FD 1862 BBS #V_DESC,R2,DESCRIPTOR ; Branch if only descriptor desired
53 2C 0901 1863 MOVCS R3,(R4),#0,R5,(R6) ; Move field to DAP control block
0F 1B 0907 1864 BLEQU 20$ ; Done if all SRC bytes are copied
; (i.e., SRC size LEQU DST size)
52 6E DO 0909 1865 MOVL (SP),R2 ; Get address of control flag
07 FF A2 05 E0 090C 1867 BBS #V_TRUNC,-1(R2),20$ ; Done if extra bytes are to be
; truncated; note:
; R0 = # unmoved bytes
; R1 = address of unmoved string
81 95 0911 1872 10$: TSTB (R1)+ ; Error if any unmoved bytes are
23 12 0913 1873 BNEQ ERROR_UNSUPPORT ; non-zero
F9 50 F5 0915 1874 SOBGTR R0,10$ ; Continue until all extra bytes
; are checked
05 0918 1875 ;
66 53 7D 0919 1876 20$: RSB ; Exit
05 0918 1876 20$: RSB ; Exit
DESCRIPTOR: ; DST field is a descriptor
66 53 7D 0919 1877 MOVB R3,(R6) ; Store only quadword descriptor
05 091C 1879 RSB ; of SRC field and exit

```

```

091D 1881      .SBTTL  ERROR AND SUCCESS EXIT ROUTINES
091D 1882
091D 1883      :++
091D 1884      : Message parse has failed.
091D 1885      : Build DAP Status message and exit to caller.
091D 1886      :--
091D 1887
091D 1888      ERROR_FORMAT:      ; Format of message in incorrect
091D 1889      MOVB      #DAPS_FORMAT,-      ; Return MACCODE value
091F 1890      DAPS$DCODE MAC(R9)
01 10 A9 D1 0921 1891      CMPL      DAPS$MSG BDF2(R9),#1      ; Check for one-byte message
15 12 0925 1892      BNEQ      ERROR_COMMON      ; Take common path if not
57 08 9A 0927 1893      MOVZBL   #DAPS_FLAGS,R7      ; Change to flags field ID code
092A 1894      ; because format error was caused
092A 1895      ; by no flags field in message
10 11 092A 1896      BRB      ERROR_COMMON      ; Take common path
092C 1897      ERROR_INVALID:    ; Field of message has invalid value
092C 1898      MOVB      #DAPS_INVALID,-      ; Return MACCODE value
1B A9 90 092E 1899      DAPS$DCODE MAC(R9)
0A 11 0930 1900      BRB      ERROR_COMMON      ; Take common path
0932 1901      ERROR_SYNC:      ; Message received is out-of-sequence
1B A9 90 0932 1902      MOVB      #DAPS_MSG_SYNC,-      ; Return MACCODE value
0934 1903      DAPS$DCODE MAC(R9)
04 11 0936 1904      BRB      ERROR_COMMON      ; Take common path
0938 1905      ERROR_UNSupport:  ; Field of message has unsupported value
1B A9 90 0938 1906      MOVB      #DAPS_UNSupport,-      ; Return MACCODE value
093A 1907      DAPS$DCODE MAC(R9)
19 A9 57 90 093C 1908      ERROR_COMMON:    ; Common error exit sequence
18 A9 94 093C 1909      MOVB      R7,DAPS$DCODE_FID(R9)      ; Return ID of field in error
16 11 0940 1910      CLRB      DAPS$DCODE_STS(R9)      ; Indicate failure
0943 1911      BRB      EXIT_COMMON      ; Join common exit code
0945 1912
0945 1913      :++
0945 1914      : Message parse has been successful so far, ...
0945 1915      : Make additional validity checks.
0945 1916      :--
0945 1917
0945 1918      EXIT_SUCCESS:      ; Enter here on successful parse
57 00 9A 0945 1919      MOVZBL   #DAPS_UNKNOWN,R7      ; Set field ID to 'unknown'
5A 5B D1 0948 1920      CMPL      R11,R10      ; Branch if there are any unparsed
D0 12 094B 1921      BNEQ      ERROR_FORMAT      ; bytes left in DAP message
50 1A A9 9A 094D 1922      MOVZBL   DAPS$DCODE MSG(R9),R0      ; Get DAP message type
1C A9 50 E1 0951 1923      BBC      R0,DAPS$MSG_MASK(R9),-      ; Branch if this is not a valid
DC      ; message to receive
0955 1924
0956 1925
0956 1926      ;
0956 1927      ; Check for system specific fields in message header.
0956 1928      ;
0956 1929
38 A9 D5 0956 1930      TSTL      DAPS$SYSPEC(R9)      ; Any system specific fields?
4A 12 0959 1931      BNEQ      SSP_MINI_MSG      ; If yes, process them
095B 1932
095B 1933      ;
095B 1934      ; Update message descriptors in DAP control block.
095B 1935      ;
095B 1936
095B 1937      EXIT_COMMON:      ; Common exit sequence

```



10	A9	14	A9	C3	095B	1938	SUBL3	DAP\$Q_MSG_BUF2+4(R9),-	; Compute size of message just parsed
			5A		095E	1939		R10,DAP\$Q_MSG_BUF2(R9)	; and store it in descriptor
0C	A9		5A	D0	0961	1940	MOVL	R10,DAP\$Q_MSG_BUF1+4(R9);	Store address of next (blocked)
					0965	1941			message in buffer to parse
		10	A9	C2	0965	1942	SUBL2	DAP\$Q_MSG_BUF2(R9),-	; Store size of next (blocked)
		08	A9		0968	1943		DAP\$Q_MSG_BUF1(R9)	; message in buffer to parse
50		18	A9	D0	096A	1944	MOVL	DAP\$L_DCODE_STS(R9),R0	; Get return status code
				04	096E	1945	RET		; Return to caller

```

096F 1947      .SBTTL CHECK_MASKS - VALIDATE FIELD BIT OPTIONS
096F 1948
096F 1949      :++
096F 1950      : Functional Description:
096F 1951      :
096F 1952      :     CHECK_MASKS invoked from the CHECK_MASKS macro examines the designated
096F 1953      :     field for invalid and unsupported bits set.
096F 1954
096F 1955      : Calling Sequence:
096F 1956      :
096F 1957      :     BSBW    CHECK_MASKS
096F 1958
096F 1959      : Input Parameters:
096F 1960      :
096F 1961      :     R6     Address of designated field in DAP control block
096F 1962      :     R7     Field ID value
096F 1963
096F 1964      : In-line coded arguments:
096F 1965      :
096F 1966      :     Byte0  Size in bytes of the designated field in DAP control block
096F 1967      :     Byten  Mask of invalid bits (1-4 bytes; size specified in byte0)
096F 1968      :     Bytem  Mask of unsupported bits (1-4 bytes; size specified in byte0)
096F 1969
096F 1970      : Implicit Inputs:
096F 1971      :
096F 1972      :     None
096F 1973
096F 1974      : Output Parameters:
096F 1975      :
096F 1976      :     R0-R1  Destroyed
096F 1977      :     R6-R7  Unchanged
096F 1978
096F 1979      : Implicit Outputs:
096F 1980      :
096F 1981      :     The specified field of the DAP control block is validated.
096F 1982
096F 1983      : Completion Codes:
096F 1984      :
096F 1985      :     None
096F 1986
096F 1987      : Side Effects:
096F 1988      :
096F 1989      :     If any invalid or unsupported bits are set, control is given to an
096F 1990      :     appropriate error routine.
096F 1991
096F 1992      :     An exception exit described above, leaves the return address on the
096F 1993      :     stack.
096F 1994
096F 1995      :--
096F 1996
096F 1997      CHECK_MASKS:
50   6E   DO 096F 1998      MOVL    (SP),R0      ; Entry point
51   80   9A 0972 1999      MOVZBL (R0)+,R1      ; Get address of in-line arguments
0975 2000      $CASEB  SELECTOR=R1- ; Get DST field size
0975 2001      BASE=#1-   ; Dispatch on field size:
0975 2002      DISPL=<-   ;
0975 2003      10%-      ; 1-byte

```

			0975	2004		20\$-	:	2-bytes
			0975	2005		30\$-	:	Error
			0975	2006		40\$-	:	4-bytes
			0975	2007		>	:	
80	9A	11	0981	2008	30\$:	BRB	:	Value is out-of-range
	66	93	0983	2009	10\$:	BITB	:	Check for invalid bits
	A4	12	0986	2010		BNEQ	:	Branch on error
80	66	93	0988	2011		BITB	:	Check for unsupported bits
	12	11	098B	2012		BRB	:	Join common code
80	66	B3	098D	2013	20\$:	BITW	:	Check for invalid bits
	9A	12	0990	2014		BNEQ	:	Branch on error
80	66	B3	0992	2015		BITW	:	Check for unsupported bits
	08	11	0995	2016		BRB	:	Join common code
80	66	D3	0997	2017	40\$:	BITL	:	Check for invalid bits
	90	12	099A	2018		BNEQ	:	Branch on error
80	66	D3	099C	2019		BITL	:	Check for unsupported bits
	97	12	099F	2020	50\$:	BNEQ	:	Branch on error
6E	50	D0	09A1	2021		MOVL	:	Bump return address past argument list
		05	09A4	2022		RSB	:	Exit

```

09A5 2024      .SBTTL  SSP_MINI_MSG - DECODE SYSTEM SPECIFIC FIELD
09A5 2025
09A5 2026      :++
09A5 2027      : Decode the system specific field found in the message header.
09A5 2028      : Treat it as the operand portion of a mini-message that has a menu field
09A5 2029      : and related fields.
09A5 2030      :--
09A5 2031
09A5 2032      SSP_MINI_MSG:      : Code segment of mainline
SA 38 A9 DD 09A5 2033      PUSHL  R10      : Save end-of-message + 1 address
SA 38 A9 7D 09A7 2034      MOVQ   DAP$Q_SYSPEC(R9),R10 : R10 = size of syspec field
09AB 2035      : R11 = address of start-of-field
SA 5B CO 09AB 2036      ADDL2  R11,R10 : R10 = address of end-of-field + 1
09AE 2037      $ZERO_FILL- : Zero system specific work area
09AE 2038      DST=DAP$L_SSPWA(R9)- : in DAP control block
09AE 2039      SIZE=#DAP$K_SSPWA :
09B8 2040
09B8 2041      :
09B8 2042      : Process the system specific menu field (optional).
09B8 2043      : Each bit set denotes that its associated field follows in the message.
09B8 2044      :
09B8 2045
09B8 2046      ASSUME  DAP$V_SSP_CAP+1 EQ DAP$V_SSP_FLG
09B8 2047
58 0A06'CF 9E 09B8 2048      MOVAB  W^SSP_SUCCESS,R8 : Specify transfer address on EOM
09BD 2049      STORE_FIELD  SSP_MENU,2,K_EXT : Save system specific menu field
09C4 2050      CHECK_MASKS  SSP_MEN,2 : Validate bit options
58 FF4D CF 9E 09CC 2051      MOVAB  W^ERROR_FORMAT,R8 : Specify transfer address on EOM
SC 66 3C 09D1 2052      MOVZWL (R6),AP : Copy menu to scratch register
50 5C 02 00 EA 09D4 2053      SSP_LOOP:
09D4 2054      FFS  #0,#DAP$V_SSP_FLG+1,AP,R0 : Get position of next bit set
09D9 2055      $CLRBIT R0,AP : Clear menu bit just found
09DD 2056      PUSHAB B^SSP_LOOP : Push return address on stack
09E0 2057      $CASEB  SELECTOR=R0- : Next field:
09E0 2058      DISPL=<- :
09E0 2059      10$- : SSP_CAP
09E0 2060      20$- : SSP_FLG
09E0 2061      > :
1C 11 09E8 2062      BRB  SSP_SUCCESS : All fields parsed
09EA 2063
09EA 2064      :
09EA 2065      : Process each field specified in the menu (optional).
09EA 2066      :
09EA 2067
09EA 2068      10$:  STORE_FIELD  SSP_CAP,4,K_EXT,<M TRUNC>
09F1 2069      : Save system specific capabilities
05 09F1 2070      RSB : field
09F2 2071      20$:  STORE_FIELD  SSP_FLG,4,K_EXT : Save system specific flags field
09F9 2072      CHECK_MASKS  SSP_FLG,4 : Validate bit options
05 0A05 2073      RSB :
0A06 2074
0A06 2075      :
0A06 2076      : System specific mini-message has been parsed successfully!
0A06 2077      :
0A06 2078
50 8ED0 0A06 2079      SSP_SUCCESS:
0A06 2080      POPL  R0 : Throw away return address on stack

```

```
5A 8ED0 0A09 2081      POPL  R10      ; Restore address of end-of-message + 1
FF4C 31 0A0C 2082      BRW  EXIT_COMMON ; Exit here because this routine
                   0A0F 2083      ; was entered from EXIT_SUCCESS
                   0A0F 2084
                   0A0F 2085      .END      ; End of module
```

FALDECODE  
Symbol table

- DECODE DAP MESSAGE

B 7

16-SEP-1984 01:42:32 VAX/VMS Macro V04-00  
5-SEP-1984 01:16:49 [FAL.SRC]FALDECODE.MAR;1

Page 51  
(26)

FAL  
VOI

\$\$COUNT	= 00000002			DAP\$B_NOK	00000044
ACC_INVALID	00000403	R	02	DAP\$B_NOR	00000046
ACC_MSG	00000381	R	02	DAP\$B_MSG	00000049
ALL_INVALID	00000713	R	02	DAP\$B_NUL	0000006D
ALL_LOOP	0C0006A3	R	02	DAP\$B_ORG	00000045
ALL_MSG	00000687	R	02	DAP\$B_QSTYPE	00000042
ATT_INVALID	0000037E	R	02	DAP\$B_RAC	00000046
ATT_LOOP	00000268	R	02	DAP\$B_RAT	00000047
ATT_MSG	0000022E	R	02	DAP\$B_REF	0000006C
CHECK_FILE_SYSTEM	000001A7	R	02	DAP\$B_RFM	00000046
CHECK_MASKS	0000096F	R	02	DAP\$B_SHR	00000043
CHECK_OPERATING_SYSTEM	000001D2	R	02	DAP\$B_SIZ	0000005C
CHECK_PROTOCOL_VERSION	00000132	R	02	DAP\$B_SIZ_TMP	0000004A
CMP_INVALID	00000520	R	02	DAP\$B_STREAMID	00000032
CMP_MSG	000004E5	R	02	DAP\$B_TKS	0000007F
CNF_MSG	000000EE	R	02	DAP\$B_TYPE	00000030
CON_INVALID	000004E2	R	02	DAP\$B_USRNUM	00000046
CON_MSG	000004CA	R	02	DAP\$B_USRVER	00000048
CTL_INVALID	000004C7	R	02	DAP\$B_VERNUM	00000044
CTL_LOOP	0000045D	R	02	DAP\$B_X_FIELD	00000024
CTL_MSG	00000406	R	02	DAP\$C_BCN	000000C0
DAP\$B_ACCFUNC	00000040			DAP\$K_ABORT	= 00000003
DAP\$B_ACCOPT	00000041			DAP\$K_ACCOPT_I	= 000000E0
DAP\$B_AID	00000050			DAP\$K_ACCOPT_U	= 00000016
DAP\$B_ALN	00000044			DAP\$K_ACC_MSG	= 00000003
DAP\$B_AOP	00000045			DAP\$K_ACK_MSG	= 00000006
DAP\$B_BITCNT	00000035			DAP\$K_ALLMENU_I	= 0000FE10
DAP\$B_BKS	00000050			DAP\$K_ALLMENU_U	= 00000000
DAP\$B_BKZ	00000051			DAP\$K_ALL_MSG	= 0000000B
DAP\$B_BLKCNT	00000056			DAP\$K_ANY	= 00000000
DAP\$B_BSZ	00000052			DAP\$K_AOP_I	= 000000F0
DAP\$B_CMPFUNC	00000040			DAP\$K_AOP_U	= 00000000
DAP\$B_CONFUNC	00000040			DAP\$K_ATTMENU_I	= FFE08000
DAP\$B_CTLFUNC	00000040			DAP\$K_ATTMENU_U	= 00000000
DAP\$B_DAN	00000070			DAP\$K_ATT_MSG	= 00000002
DAP\$B_DATATYPE	00000044			DAP\$K_BLK_FILE	= 00000005
DAP\$B_DBS	00C0007C			DAP\$K_BLK_VBN	= 00000004
DAP\$B_DCODE_FID	00000019			DAP\$K_BLN	= 000000C0
DAP\$B_DCODE_MAC	0000001B			DAP\$K_BLS_D	= 00000200
DAP\$B_DCODE_MSG	0000001A			DAP\$K_BN2	= 00000002
DAP\$B_DECVER	00000047			DAP\$K_BN4	= 00000004
DAP\$B_DTP	00000071			DAP\$K_BN8	= 00000007
DAP\$B_ECONUM	00000045			DAP\$K_BSZ_D	= 00000008
DAP\$B_FAC	00000042			DAP\$K_CHANGE_B	= 00000006
DAP\$B_FILESYS	00000043			DAP\$K_CHANGE_E	= 00000007
DAP\$B_FLAGS	00000031			DAP\$K_CLOSE	= 00000001
DAP\$B_FLG	00000048			DAP\$K_CMP_MSG	= 00000007
DAP\$B_FSZ	00000051			DAP\$K_CMWA	= 00000050
DAP\$B_IAN	0000006E			DAP\$K_CNF_MSG	= 00000001
DAP\$B_IBS	0000007D			DAP\$K_CONNECT	= 00000002
DAP\$B_KRF	00000047			DAP\$K_CON_MSG	= 00000005
DAP\$B_LAN	0000006F			DAP\$K_COPOS11	= 0000000D
DAP\$B_LEN256	00000034			DAP\$K_CREATE	= 00000002
DAP\$B_LENGTH	00000033			DAP\$K_CTMENU_I	= 0000FF90
DAP\$B_LVL	0000007E			DAP\$K_CTMENU_U	= 00000040
DAP\$B_NAME TYPE	00000040			DAP\$K_CTL_MSG	= 0000C004
DAP\$B_NOA	00000045			DAP\$K_CYL	= 00000001

FALDECODE  
Symbol table

- DECODE DAP MESSAGE

C 7

16-SEP-1984 01:42:32 VAX/VMS Macro V04-00  
5-SEP-1984 01:16:49 [FAL.SRC]FALDECODE.MAR;1

Page 52  
(26)

FA  
VO

DAPSK\_DATATYP\_D = 00000002  
DAPSK\_DATATYP\_I = 00000004  
DAPSK\_DATATYP\_U = 00000088  
DAPSK\_DAT\_MSG = 00000008  
DAPSK\_DELETE = 00000005  
DAPSK\_DEV\_I = FC000040  
DAPSK\_DEV\_U = 00000000  
DAPSK\_DIR\_LIST = 00000006  
DAPSK\_DISCONN = 00000004  
DAPSK\_DISPLAY = 00000010  
DAPSK\_DISPLAY\_I = 0000FCC0  
DAPSK\_DISPLAY\_U = 00000200  
DAPSK\_ERASE = 00000004  
DAPSK\_EXECUTE = 00000008  
DAPSK\_EXTEND\_B = 0000000B  
DAPSK\_EXTEND\_E = 0000000F  
DAPSK\_FAC\_D = 00000002  
DAPSK\_FAC\_I = 00000000  
DAPSK\_FAC\_U = 00000000  
DAPSK\_FCST1 = 00000004  
DAPSK\_FIND = 0000000E  
DAPSK\_FIX = 00000001  
DAPSK\_FLAGS\_I = 00000090  
DAPSK\_FLAGS\_U = 00000048  
DAPSK\_FLG\_I = 000000F8  
DAPSK\_FLG\_U = 00000000  
DAPSK\_FLUSH = 0000000C  
DAPSK\_FOP\_I = F1021004  
DAPSK\_FOP\_U = 00002000  
DAPSK\_FREE = 0000000A  
DAPSK\_GET\_READ = 00000001  
DAPSK\_IAS = 00000006  
DAPSK\_IDX = 00000020  
DAPSK\_IN2 = 00000001  
DAPSK\_IN4 = 00000003  
DAPSK\_IN8 = 00000006  
DAPSK\_KEYMENU\_I = FFF81000  
DAPSK\_KEYMENU\_U = 00000000  
DAPSK\_KEY\_ACC = 00000001  
DAPSK\_KEY\_MSG = 0000000A  
DAPSK\_LBN = 00000002  
DAPSK\_NAM\_MSG = 0000000F  
DAPSK\_NO\_FS = 00000006  
DAPSK\_OPEN = 00000001  
DAPSK\_ORG\_D = 00000000  
DAPSK\_PAC = 00000005  
DAPSK\_PROMENU\_I = 0000FF00  
DAPSK\_PROMENU\_U = 00000000  
DAPSK\_PROTECT\_I = 0000FE00  
DAPSK\_PROTECT\_U = 00000000  
DAPSK\_PRO\_MSG = 0000000E  
DAPSK\_PUT\_WRITE = 00000004  
DAPSK\_P\_OS = 0000000E  
DAPSK\_QUIT = 00000005  
DAPSK\_RAT\_I = 00000020  
DAPSK\_RAT\_U = 000000C0  
DAPSK\_REL = 00000010

DAPSK\_RELEASE = 00000009  
DAPSK\_RENAME = 00000003  
DAPSK\_RESET = 00000003  
DAPSK\_RESPONSE = 00000002  
DAPSK\_RESUME = 00000004  
DAPSK\_RETRY = 00000001  
DAPSK\_REWIND = 00000006  
DAPSK\_RFA\_ACC = 00000002  
DAPSK\_RFI = 00000004  
DAPSK\_RFM\_D = 00000001  
DAPSK\_RMST1 = 00000001  
DAPSK\_RMS20 = 00000002  
DAPSK\_RMS32 = 00000003  
DAPSK\_RMS32S = 0000000A  
DAPSK\_ROP\_I = FFF80008  
DAPSK\_ROP\_U = 00000000  
DAPSK\_RSTS = 00000002  
DAPSK\_RSX11D = 00000005  
DAPSK\_RSX11M = 00000004  
DAPSK\_RSX11MP = 0000000C  
DAPSK\_RSX11S = 00000003  
DAPSK\_RT11 = 00000001  
DAPSK\_RT11FS = 00000005  
DAPSK\_SEQ = 00000000  
DAPSK\_SEQ\_ACC = 00000000  
DAPSK\_SEQ\_FILE = 00000003  
DAPSK\_SHR\_D = 00000000  
DAPSK\_SHR\_I = 00000080  
DAPSK\_SHR\_U = 00000010  
DAPSK\_SKIP\_FILE = 00000005  
DAPSK\_SKIP\_REC = 00000002  
DAPSK\_SPACE\_BW = 00000012  
DAPSK\_SPACE\_FW = 00000011  
DAPSK\_SSPWA = 00000010  
DAPSK\_SSP\_FLG\_I = FFFFFFFE  
DAPSK\_SSP\_FLG\_U = 00000000  
DAPSK\_SSP\_MEN\_I = 0000FFFC  
DAPSK\_SSP\_MEN\_U = 00000000  
DAPSK\_STG = 00000000  
DAPSK\_STM = 00000004  
DAPSK\_STMCR = 00000006  
DAPSK\_STMLF = 00000005  
DAPSK\_STS\_MSG = 00000009  
DAPSK\_SUBMIT = 00000007  
DAPSK\_SUM\_MSG = 0000000C  
DAPSK\_TEMP = 00000010  
DAPSK\_TERMINATE = 00000008  
DAPSK\_TIMENU\_I = 0000FF80  
DAPSK\_TIMENU\_U = 00000000  
DAPSK\_TIM\_MSG = 0000000D  
DAPSK\_TOPS10 = 00000009  
DAPSK\_TOPS10FS = 00000008  
DAPSK\_TOPS20 = 00000008  
DAPSK\_TOPS20FS = 00000007  
DAPSK\_TRUNCATE = 00000007  
DAPSK\_UDF = 00000000  
DAPSK\_UPDATE = 00000003

FALDECODE  
Symbol table

- DECODE DAP MESSAGE

D 7

16-SEP-1984 01:42:32 VAX/VMS Macro V04-00  
5-SEP-1984 01:16:49 [FAL.SRC]FALDECODE.MAR;1

Page 53  
(26)

FA  
VO

DAPSK\_VAR = 00000002  
DAPSK\_VAXELAN = 0000000F  
DAPSK\_VAXVMS = 00000007  
DAPSK\_VBN = 00000003  
DAPSK\_VFC = 00000003  
DAPSL\_ALQ1 = 0000004C  
DAPSL\_ALQ2 = 0000004C  
DAPSL\_ATTMENU = 00000040  
DAPSL\_CMWA = 00000030  
DAPSL\_CRC\_RSLT = 00000020  
DAPSL\_DCODE\_STS = 00000018  
DAPSL\_DEV = 00000068  
DAPSL\_DVB = 00000078  
DAPSL\_EBK = 00000078  
DAPSL\_FOP1 = 00000064  
DAPSL\_FOP2 = 00000044  
DAPSL\_HBK = 00000074  
DAPSL\_KEYMENU = 00000040  
DAPSL\_LOC = 00000048  
DAPSL\_MRN = 00000058  
DAPSL\_MSG\_MASK = 0000001C  
DAPSL\_RECNUM1 = 00000040  
DAPSL\_RECNUM2 = 00000048  
DAPSL\_ROP = 00000050  
DAPSL\_RVB = 00000074  
DAPSL\_SBN = 0000007C  
DAPSL\_SSPWA = 00000080  
DAPSL\_SSP\_CAP = 00000088  
DAPSL\_SSP\_FLG = 00000084  
DAPSL\_STV = 0000004C  
DAPSL\_TEMP = 00000090  
DAPSM\_BITCNT = 00000008  
DAPSM\_BLKCNT = 00000040  
DAPSM\_CMPFMT = 00000008  
DAPSM\_DFTSPEC = 00000010  
DAPSM\_DMO = 00002000  
DAPSM\_DSP\_3NAM = 00000200  
DAPSM\_DSP\_ATT = 00000001  
DAPSM\_EMBEDDED = 00000010  
DAPSM\_GET = 00000002  
DAPSM\_GO\_NOGO = 00000010  
DAPSM\_IMAGE = 00000002  
DAPSM\_LOADIM = 00000001  
DAPSM\_LSA = 00000040  
DAPSM\_MACY11 = 00000080  
DAPSM\_MSE = 00000010  
DAPSM\_SEGMENT = 00000040  
DAPSM\_TMP1\$ = 00000020  
DAPSM\_TMP2\$ = 000000C0  
DAPSM\_TMP3\$ = 00020000  
DAPSM\_TMP4\$ = 01000000  
DAPSM\_TMP5\$ = F0000000  
DAPSM\_ZERO = 00000080  
DAPSQ\_ADT = 00000070  
DAPSQ\_BDT = 00000060  
DAPSQ\_CDT = 00000048  
DAPSQ\_DCODE\_FLG = 00000000

DAPSQ\_EDT = 00000058  
DAPSQ\_FILEDATA = 00000044  
DAPSQ\_FILESPEC = 00000044  
DAPSQ\_KEY = 00000048  
DAPSQ\_KNM = 00000064  
DAPSQ\_MSG\_BUF1 = 00000008  
DAPSQ\_MSG\_BUF2 = 00000010  
DAPSQ\_NAMESPEC = 00000044  
DAPSQ\_OWNER = 00000048  
DAPSQ\_PASSWORD = 00000050  
DAPSQ\_PDT = 00000068  
DAPSQ\_RDT = 00000050  
DAPSQ\_RUNSYS = 0000005C  
DAPSQ\_STX = 00000050  
DAPSQ\_SYSCAP = 00000028  
DAPSQ\_SYSPEC = 00000038  
DAPSV\_ADT = 00000006  
DAPSV\_AID = 00000006  
DAPSV\_ALN = 00000001  
DAPSV\_ALQ1 = 00000006  
DAPSV\_ALQ2 = 00000005  
DAPSV\_AOP = 00000002  
DAPSV\_BDT = 00000004  
DAPSV\_BITCNT = 00000003  
DAPSV\_BKS = 00000007  
DAPSV\_BKZ = 00000007  
DAPSV\_BLKCNT = 00000006  
DAPSV\_BLS = 00000004  
DAPSV\_BSZ = 00000000  
DAPSV\_CDT = 00000000  
DAPSV\_DAN = 00000009  
DAPSV\_DATATYPE = 00000000  
DAPSV\_DBS = 0000000E  
DAPSV\_DEQ1 = 0000000B  
DAPSV\_DEQ2 = 00000008  
DAPSV\_DEV = 0000000E  
DAPSV\_DFL = 00000001  
DAPSV\_DISPLAY2 = 00000005  
DAPSV\_DTP = 0000000A  
DAPSV\_DVB = 0000000D  
DAPSV\_EBK = 00000012  
DAPSV\_EDT = 00000002  
DAPSV\_FCS = 00000031  
DAPSV\_FFB = 00000013  
DAPSV\_FLG = 00000000  
DAPSV\_FOP1 = 0000000C  
DAPSV\_FSZ = 00000008  
DAPSV\_GEQ\_V41 = 00000020  
DAPSV\_GEQ\_V42 = 00000021  
DAPSV\_GEQ\_V52 = 00000022  
DAPSV\_GEQ\_V54 = 00000023  
DAPSV\_GEQ\_V56 = 00000024  
DAPSV\_GEQ\_V60 = 00000025  
DAPSV\_GEQ\_V70 = 00000026  
DAPSV\_GEQ\_V71 = 00000027  
DAPSV\_HBK = 00000011  
DAPSV\_IAN = 00000007



FALDECODE  
Symbol table

- DECODE DAP MESSAGE

E 7

16-SEP-1984 01:42:32 VAX/VMS Macro V04-00  
5-SEP-1984 01:16:49 [FAL.SRC]FALDECODE.MAR;1

Page 54  
(26)

FA  
V0

DAPSV\_IAS = 0000003B  
DAPSV\_IBS = 0000000F  
DAPSV\_IFL = 00000002  
DAPSV\_KEY = 00000001  
DAPSV\_KNM = 00000005  
DAPSV\_KRF = 00000002  
DAPSV\_LAN = 00000008  
DAPSV\_LEN256 = 00000002  
DAPSV\_LENGTH = 00000001  
DAPSV\_LOC = 00000003  
DAPSV\_LRL = 00000010  
DAPSV\_LVL = 00000010  
DAPSV\_MRL = 00000012  
DAPSV\_MRN = 00000009  
DAPSV\_MRS = 00000005  
DAPSV\_NSQ = 00000003  
DAPSV\_NUL = 00000006  
DAPSV\_ORG = 00000001  
DAPSV\_OWNER = 00000000  
DAPSV\_PDT = 00000005  
DAPSV\_PROGRP = 00000003  
DAPSV\_PROOWN = 00000002  
DAPSV\_PROSYS = 00000001  
DAPSV\_PROWLD = 00000004  
DAPSV\_P OS = 0000003C  
DAPSV\_RAC = 00000000  
DAPSV\_RAT = 00000003  
DAPSV\_RDT = 00000001  
DAPSV\_REF = 00000004  
DAPSV\_RFM = 00000002  
DAPSV\_RMS = 00000030  
DAPSV\_ROP = 00000003  
DAPSV\_RSTS = 00000039  
DAPSV\_RSX = 0000003A  
DAPSV\_RT11 = 00000038  
DAPSV\_RUNSYS = 0000000A  
DAPSV\_RVB = 0000000B  
DAPSV\_RVN = 00000003  
DAPSV\_SBN = 00000014  
DAPSV\_SEGMENT = 00000006  
DAPSV\_SSP\_CAP = 00000000  
DAPSV\_SSP\_FLG = 00000001  
DAPSV\_STM\_ONLY = 00000032  
DAPSV\_STREAMID = 00000000  
DAPSV\_SYSPEC = 00000005  
DAPSV\_TKS = 00000011  
DAPSV\_TOPS10 = 00000036  
DAPSV\_TOPS20 = 00000037  
DAPSV\_VAXELAN = 00000035  
DAPSV\_VAXVMS = 00000034  
DAPSV\_VMS\_XPF1 = 0000002C  
DAPSV\_VOL = 00000000  
DAPSV\_X\_CHECK = 00000001  
DAPSV\_X\_RECNUM = 00000000  
DAPSW\_A[LMENU = 00000040  
DAPSW\_BLS = 00000048  
DAPSW\_BUFSIZ = 00000040

DAPSW\_CHECK = 00000042  
DAPSW\_CTLMENU = 00000044  
DAPSW\_DEQ1 = 00000054  
DAPSW\_DEQ2 = 00000052  
DAPSW\_DFL = 00000044  
DAPSW\_DISPLAY1 = 0000004C  
DAPSW\_DISPLAY2 = 00000054  
DAPSW\_FFB = 00000072  
DAPSW\_IFL = 00000046  
DAPSW\_LRL = 00000070  
DAPSW\_MRL = 00000072  
DAPSW\_MRS = 0000004A  
DAPSW\_PARTNER = 00000006  
DAPSW\_POS = 0000004C  
DAPSW\_POS\_TMP = 0000004A  
DAPSW\_PROGRP = 00000054  
DAPSW\_PROMENU = 00000040  
DAPSW\_PROOWN = 00000052  
DAPSW\_PROSYS = 00000050  
DAPSW\_PROWLD = 00000056  
DAPSW\_PVN = 00000042  
DAPSW\_RFA = 00000042  
DAPSW\_RVN = 00000042  
DAPSW\_SSP MENU = 00000080  
DAPSW\_STSCODE = 00000040  
DAPSW\_SUMENU = 00000040  
DAPSW\_TIMENU = 00000040  
DAPSW\_VERSION = 00000004  
DAPSW\_VOL = 00000042  
DAP\$\_ACCFUNC = 00000010  
DAP\$\_ACCOPT = 00000011  
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\$ BLKCNT = 00000018  
DAP\$ BLS = 00000015  
DAP\$ BSZ = 0000001E  
DAP\$ BUFSIZ = 00000010  
DAP\$ CDT = 00000011  
DAP\$ CHECK = 00000012  
DAP\$ CMPFUNC = 00000010  
DAP\$ CONFUNC = 00000010  
DAP\$ CTLFUNC = 00000010  
DAP\$ CTLMENU = 00000011  
DAP\$ DAN = 0000001C  
DAP\$ DATATYPE = 00000011  
DAP\$ DBS = 00000021  
DAP\$ DECVER = 00000016  
DAP\$ DEQ1 = 0000001C

FALDECODE  
Symbol table

- DECODE DAP MESSAGE

F 7

16-SEP-1984 01:42:32 VAX/VMS Macro V04-00  
5-SEP-1984 01:16:49 [FAL.SRC]FALDECODE.MAR;1

Page 55  
(26)

FA  
VO

DAPS\_DEQ2 = 00000019  
DAPS\_DEV = 0000001F  
DAPS\_DFL = 00000012  
DAPS\_DISPLAY1 = 00000015  
DAPS\_DISPLAY2 = 00000017  
DAPS\_DTP = 0000001D  
DAPS\_DVB = 00000020  
DAPS\_EBK = 00000023  
DAPS\_ECONUM = 00000014  
DAPS\_EDT = 00000013  
DAPS\_FAC = 00000013  
DAPS\_FFB = 00000024  
DAPS\_FILEDATA = 00000011  
DAPS\_FILESPEC = 00000012  
DAPS\_FILESYS = 00000012  
DAPS\_FLAGS = 00000008  
DAPS\_FLG = 00000011  
DAPS\_FOP1 = 0000001D  
DAPS\_FOP2 = 00000011  
DAPS\_FORMAT = 00000008  
DAPS\_FSZ = 00000019  
DAPS\_HBK = 00000022  
DAPS\_IAN = 0000001A  
DAPS\_IBS = 00000022  
DAPS\_IFL = 00000013  
DAPS\_INVALID = 00000009  
DAPS\_KEY = 00000013  
DAPS\_KEYMENU = 00000010  
DAPS\_KNM = 00000018  
DAPS\_KRF = 00000014  
DAPS\_LAN = 0000001B  
DAPS\_LEN256 = 0000000B  
DAPS\_LENGTH = 0000000A  
DAPS\_LOC = 00000014  
DAPS\_LRL = 00000021  
DAPS\_LVL = 00000023  
DAPS\_MRL = 00000025  
DAPS\_MRN = 0000001A  
DAPS\_MRS = 00000016  
DAPS\_MSG\_SYNC = 0000000A  
DAPS\_NAMESPEC = 00000011  
DAPS\_NAMETYPE = 00000010  
DAPS\_NSQ = 00000014  
DAPS\_NUL = 00000019  
DAPS\_ORG = 00000012  
DAPS\_OSTYPE = 00000011  
DAPS\_OWNER = 00000011  
DAPS\_PASSWORD = 00000016  
DAPS\_PDT = 00000016  
DAPS\_POS\_TMP = 00000015  
DAPS\_PROGRP = 00000014  
DAPS\_PROMENU = 00000010  
DAPS\_PROOWN = 00000013  
DAPS\_PROSYS = 00000012  
DAPS\_PROWLD = 00000015  
DAPS\_RAC = 00000012  
DAPS\_RAT = 00000014

DAPS\_RDT = 00000012  
DAPS\_RECNUM1 = 00000010  
DAPS\_REF = 00000017  
DAPS\_RFM = 00000013  
DAPS\_ROP = 00000015  
DAPS\_RUNSYS = 0000001B  
DAPS\_RVB = 0000001E  
DAPS\_RVN = 00000014  
DAPS\_SBN = 00000025  
DAPS\_SHR = 00000014  
DAPS\_SIZ\_TMP = 00000016  
DAPS\_SSP\_CAP = 0000000E  
DAPS\_SSP\_FLG = 0000000E  
DAPS\_SSP\_MENU = 0000000E  
DAPS\_STREAMID = 00000009  
DAPS\_SYSCAP = 00000018  
DAPS\_SYSPEC = 0000000E  
DAPS\_TIMENU = 00000010  
DAPS\_TKS = 00000024  
DAPS\_TYPE = 00000008  
DAPS\_UNKNOWN = 00000000  
DAPS\_UNSUPPORT = 00000002  
DAPS\_USRNUM = 00000015  
DAPS\_USRVER = 00000017  
DAPS\_VERNUM = 00000013  
DAPS\_VOL = 00000011  
DAT\_MSG = 00000523 R 02  
DESCRIPTOR = 00000919 R R 02  
DISPATCH\_TABLE = 000000C8 R R 02  
ERROR\_COMMON = 0000093C R R 02  
ERROR\_FORMAT = 0000091D R R 02  
ERROR\_INVALID = 0000092C R R 02  
ERROR\_SYNC = 00000932 R R 02  
ERROR\_UNSUPPORT = 00000938 R R 02  
EXIT\_COMMON = 0000095B R R 02  
EXIT\_SUCCESS = 00000945 R R 02  
FAL\$DECODE\_MSG = 00000000 R G 02  
HDR\_INVALID = 000000C2 R R 02  
HDR\_LOOP = 0000005E R R 02  
HDR\_UNSUPPORT = 000000C5 R R 02  
HEADER = 00000022 R R 02  
KEY\_INVALID = 00000684 R R 02  
KEY\_LOOP = 00000573 R R 02  
KEY\_MSG = 0000054F R R 02  
K\_EXT = 00000000  
K\_FIX = 00000001  
K\_IMG = 00000002  
K\_ROM = 00000003  
MOVE\_FIELD = 000008FD R 02  
M\_DESC = 00000010  
M\_SRCR3 = 00000040  
M\_TRUNC = 00000020  
NAM\_INVALID = 0000087D R 02  
NAM\_MSG = 00000859 R 02  
PRO\_INVALID = 00000856 R 02  
PRO\_LOOP = 00000801 R 02  
PRO\_MSG = 000007E1 R 02



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

Macro library name	Macros defined
-----	-----
_\$255\$DUA28:[FAL.OBJ]FAL.MLB;1	24
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	30

1460 GETS were required to define 30 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:FALDECODE/OBJ=OBJ\$:FALDECODE MSRC\$:FALDECODE/UPDATE=(ENH\$:FALDECODE)+LIB\$:FAL/LIB

FA  
PS

PS  
--  
: SA  
FA

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

Th  
23  
Th  
42  
12

Ma  
--  
- \$  
- \$  
TO  
49  
Th  
MA

0175 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

The image displays a grid of 140 terminal window screenshots, arranged in 10 rows and 14 columns. Each window shows a different system utility or data output. The windows are organized as follows:

- Row 1:** FALDECODE LIS, followed by 13 windows showing various data outputs.
- Row 2:** 14 windows showing various data outputs.
- Row 3:** 14 windows showing various data outputs.
- Row 4:** 14 windows showing various data outputs.
- Row 5:** 14 windows showing various data outputs.
- Row 6:** 14 windows showing various data outputs.
- Row 7:** 14 windows showing various data outputs.
- Row 8:** FALRMSDAP LIS, FALSTATE LIS, FDL, CREATEFDL MAP, followed by 10 windows showing various data outputs.
- Row 9:** FALENCODE LIS, FALLOGGER LIS, FALMAIN LIS, followed by 11 windows showing various data outputs.
- Row 10:** 14 windows showing various data outputs.