


```

NN      NN      TTTTTTTTTT      000000      SSSSSSSS      CCCCCCCC      NN      NN      XX      XX      AAAAAA      BBBB8888
NN      NN      TTTTTTTTTT      000000      SSSSSSSS      CCCCCCCC      NN      NN      XX      XX      AAAAAA      BBBB8888
NN      NN      TT              00          00      SS              CC              NN      NN      XX      XX      AA          AA      BB          BB
NN      NN      TT              00          00      SS              CC              NN      NN      XX      XX      AA          AA      BB          BB
NNNN    NN      TT              00          0000    SS              CC              NNNN    NN      XX      XX      AA          AA      BB          BB
NNNN    NN      TT              00          0000    SS              CC              NNNN    NN      XX      XX      AA          AA      BB          BB
NN      NN      NN      TT      00      00      00      SSSSSS      CC              NN      NN      NN      XX      XX      AA          AA      BBBB8888
NN      NN      NN      TT      00      00      00      SSSSSS      CC              NN      NN      NN      XX      XX      AA          AA      BBBB8888
NN      NN      NN      TT      0000      00      SS              CC              NN      NNNN    XX      XX      AAAAAAAAAA      BB          BB
NN      NN      NN      TT      0000      00      SS              CC              NN      NNNN    XX      XX      AAAAAAAAAA      BB          BB
NN      NN      NN      TT      00          00      SS              CC              NN      NN      XX      XX      AA          AA      BB          BB
NN      NN      NN      TT      00          00      SS              CC              NN      NN      XX      XX      AA          AA      BB          BB
NN      NN      NN      TT      000000      SSSSSSSS      CCCCCCCC      NN      NN      XX      XX      AA          AA      BBBB8888
NN      NN      NN      TT      000000      SSSSSSSS      CCCCCCCC      NN      NN      XX      XX      AA          AA      BBBB8888

```

```

LL      IIIIIII      SSSSSSSS
LL      IIIIIII      SSSSSSSS
LL      II          SS
LL      II          SS
LL      II          SS
LL      II          SS
LL      II          SSSSSS
LL      II          SSSSSS
LL      II          SS
LL      II          SS
LL      II          SS
LL      II          SS
LLLLLLLLLLLL      IIIIIII      SSSSSSSS
LLLLLLLLLLLL      IIIIIII      SSSSSSSS

```

(2)	66
(3)	107
(4)	396
(4)	397
(5)	460

DECLARATIONS
NT\$SCAN_XABCHN - SCAN XAB CHAIN
NT\$SCAN_KEYXAB - SCAN KEY DEFINITION XAB
NT\$SCAN_ALLXAB - SCAN ALLOCATION XAB
NT\$SCAN_NAMBLK - SCAN NAME BLOCK


```

0000 66          .SBTTL  DECLARATIONS
0000 67
0000 68 :
0000 69 : Include Files:
0000 70 :
0000 71 :
0000 72          $DAPCNFDEF      : Define DAP Configuration message
0000 73          $DAPACCDEF      : Define DAP Access message
0000 74          $FABDEF         : Define File Access Block symbols
0000 75          $IFBDEF         : Define IFAB symbols
0000 76          $NAMDEF         : Define Name Block symbols
0000 77          $NWADEF         : Define Network Work Area symbols
0000 78          $XABDEF         : Define symbols common to all XABs
0000 79          $XABALLDEF      : Define Allocation XAB symbols
0000 80          $XABDATDEF      : Define Date and Time XAB symbols
0000 81          $XABFHCDEF      : Define File Header Char symbols
0000 82          $XABKEYDEF      : Define Key Definition XAB symbols
0000 83          $XABPRODEF      : Define Protection XAB symbols
0000 84          $XABRDTDEF      : Define Revision Date/Time XAB symbols
0000 85          $XABSUMDEF      : Define Summary XAB symbols
0000 86 :          $XABTRMDEF      : Define Terminal XAB symbols
0000 87 :          $XABCXFDEF         : Define FAB Context XAB symbols
0000 88 :          $XABCXRDEF         : Define RAB Context XAB symbols
0000 89          $XABJNLDEF      : Define Journal XAB symbols
0000 90
0000 91 :
0000 92 : Macros:
0000 93 :
0000 94          None
0000 95 :
0000 96 : Equated Symbols:
0000 97 :
0000 98 :
0000 99          ASSUME  NWA$Q_FLG EQ 0
0000 100
0000 101 :
0000 102 : Own Storage:
0000 103 :
0000 104          None
0000 105 :

```

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

Th
64
Th
52
34

Ma
--
-\$
-\$
-\$
TO

13

Th

MA

```

0000 107      .SBTTL  NT$SCAN_XABCHN - SCAN XAB CHAIN
0000 108
0000 109 :++
0000 110 : NT$SCAN_XABCHN - scans the user XAB chain and examines FAL's capabilities
0000 111 : to determine which DAP Attributes and Extended Attributes messages to
0000 112 : request the remote FAL to return. It also verifies that all Allocation
0000 113 : XABs found are chained sequentially and that all Key Definition XABs
0000 114 : found are chained sequentially (i.e., they form sub-chains).
0000 115
0000 116 : The message request mask is returned in R2.
0000 117
0000 118 : Calling Sequence:
0000 119
0000 120 :     BSBW  NT$SCAN_XABCHN
0000 121
0000 122 : Input Parameters:
0000 123
0000 124 :     R6      Close operation flag
0000 125 :     R7      NWA (=DAP) address
0000 126 :     R8      FAB address
0000 127 :     R9      IFAB address
0000 128 :     R10     IFAB/FWA address
0000 129 :     R11     Impure Area address
0000 130
0000 131 : Implicit Inputs:
0000 132
0000 133 :     User ALL, DAT, FHC, JNL, KEY, PRO, RDT, and SUM XABs
0000 134 :     DAP$Q_SY$CAP bits KEYXAB, ALLXAB, SUMXAB, TIMXAB, PROXAB,
0000 135 :     CHGTIMCLS, CHGPROCLS
0000 136
0000 137 : Output Parameters:
0000 138
0000 139 :     R0      Status code (RMS)
0000 140 :     R1      Destroyed
0000 141 :     R2      Message request mask
0000 142 :     R3-R5   Destroyed
0000 143
0000 144 : Implicit Outputs:
0000 145
0000 146 :     N$ASB_ALLXABCNT
0000 147 :     N$ASB_KEYXABCNT
0000 148 :     N$ASW_JNLXABJOP
0000 149 :     N$ASL_ALLXABADR
0000 150 :     N$ASL_DATXABADR
0000 151 :     N$ASL_FHCXABADR
0000 152 :     N$ASL_KEYXABADR
0000 153 :     N$ASL_PROXABADR
0000 154 :     N$ASL_RDTXABADR
0000 155 :     N$ASL_SUMXABADR
0000 156
0000 157 : Completion Codes:
0000 158
0000 159 :     Standard RMS completion codes
0000 160
0000 161 : Side Effects:
0000 162
0000 163 :     User XABs are probed for writeability.

```

```

0000 164 :
0000 165 :--
0000 166 :
0000 167 NT$SCAN_XABCHN:: : Entry point
0000 168 $ZERO_FILL- : Zero XAB scan results block
0000 169 DST=NWAST_SCAN(R7)- : Address of block
0000 170 SIZE=#NWASC_SCAN : Length of block
000A 171 : Zero R2 (request mask) as side effect
000A 172 : of executing a MOVCS instruction
53 24 AB D0 000A 173 MOVL FAB$L_XAB(R8),R3 : Get first XAB address in chain
07 11 000E 174 BRB CHKXAB :
0010 175 :
0010 176 :+
0010 177 : Process next XAB in the chain.
0010 178 :
0010 179 : Note: XABs not supported by DECnet (if found in the XAB chain) will be
0010 180 : ignored.
0010 181 :-
0010 182 :
0010 183 ASSUME XAB$C_DAT EQ 18
0010 184 ASSUME XAB$C_PRO EQ 19
0010 185 ASSUME XAB$C_ALL EQ 20
0010 186 ASSUME XAB$C_KEY EQ 21
0010 187 ASSUME XAB$C_SUM EQ 22
0010 188 ASSUME XAB$C_FHC EQ 29
0010 189 ASSUME XAB$C_RDT EQ 30
0010 190 : ASSUME XAB$C_TRM EQ 31 : Not supported for network use
0010 191 : ASSUME XAB$C_CXF EQ 32 : Not supported for network use
0010 192 : ASSUME XAB$C_CXR EQ 33 : Not supported for network use
0010 193 : ASSUME XAB$C_JNL EQ 34 : Not supported for network use
0010 194 : BUT will be processed to obtain
0010 195 : JOP field for use by NT$CREATE
0010 196 :
53 54 53 D0 0010 197 NXTXAB: MOVL R3,R4 : Save address of current XAB in chain
04 A3 D0 0013 198 MOVL XAB$L_NXT(R3),R3 : Get address of next XAB in chain
30 13 0017 199 CHKXAB: BEQL EXIT : Branch if none
0019 200 IFNORD #XAB$L_NXT+4,(R3),- : Probe for readability thru NXT field
0019 201 ERRXAB,IFBSB_MODE(R9) : of XAB and branch on failure
ED AF 9F 0020 202 PUSHAB B^NXTXAB : Push return address on stack
0023 203 $CASEB SELECTOR=XAB$B_COD(R3)- : Dispatch to routine to process:
0023 204 BASE=#XAB$C_DAT-
0023 205 DISPL=<-
0023 206 DATE TIME- : XABDAT
0023 207 PROTECTION- : XABPRO
0023 208 ALLOCATION- : XABALL
0023 209 KEY DEFINITION- : XABKEY
0023 210 SUMMARY- : XABSUM
0023 211 ERRCOD- : Invalid XAB type
0023 212 ERRCOD- : Invalid XAB type
0023 213 ERRCOD- : Invalid XAB type
0023 214 ERRCOD- : Invalid XAB type
0023 215 ERRCOD- : Invalid XAB type
0023 216 ERRCOD- : Invalid XAB type
0023 217 FILE HEADER- : XABFHC
0023 218 REV DATE_TIME- : XABRDT
0023 219 EXIT- : Ignore XABTRM
0023 220 EXIT- : Ignore XABCXF

```



```

0023 221 EXIT- : Ignore XABCXR
0023 222 JOURNALING- : XABJNL
0023 223 > :
0049 224 :
0049 225 :+
0049 226 : Exit paths.
0049 227 :-
0049 228 :
0049 229 EXIT: RMSSUC : Return success
05 004C 230 RSB : Exit with RMS code in R0
01 BA 004D 231 ERRCOD: POPR #^M<R0> : Discard return address
10 004F 232 RMSERR COD : Invalid XAB type code
01 BA 0054 233 BRB SETSTV :
01 BA 0056 234 ERRIMX: POPR #^M<R0> : Discard return address
07 0058 235 RMSERR IMX : Duplicate XAB or XABs are not dense
03 BA 005D 236 BRB SETSTV :
03 BA 005F 237 ERRXAB2: POPR #^M<R0,R1> : Discard return addresses
OC A8 53 D0 0061 238 ERRXAB: RMSERR XAB : XAB too short or not accessible
05 0066 239 SETSTV: MOVL R3,FAB$L_STV(R8) : Return XAB address in STV field
006A 240 RSB : Exit with RMS code in R0
006B 241 :
006B 242 :+
006B 243 : This routine checks the control block for correct length and writeability.
006B 244 :-
006B 245 :
006B 246 VALIDATE XAB: : Entry point
50 01 A3 9A 006B 247 MOVZBL XAB$B_BLN(R3),R0 : Get stated length of block
51 50 D1 006F 248 CML R0,R1 : Compare against expected length
EB 1F 0072 249 BLSSU ERRXAB2 : Branch if too small
0074 250 IFNOWRT R0,(R3),ERRXAB2,- : Probe for writeability and branch on
0074 251 IFB$B_MODE(R9) : failure
05 007B 252 RSB : Exit
007C 253 :
007C 254 :+
007C 255 : This routine handles the Date and Time XAB.
007C 256 :-
007C 257 :
007C 258 DATE_TIME: : Entry point
51 24 9A 007C 259 MOVZBL #XAB$C_DATLEN_V2,R1 : Get minimum (i.e., V2) length of XAB
EA 10 007F 260 BSBB VALIDATE XAB : Check length and accessibility
0104 C7 D5 0081 261 TSTL NWA$L_DATXABADR(R7) : Declare error as this is a duplicate
CF 12 0085 262 BNEQ ERRIMX : XAB
0104 C7 53 D0 0087 263 MOVL R3,NWA$L_DATXABADR(R7) : Save address of Date and Time XAB
1A E1 008C 264 ABC #DAP$V_TIMXAB,- : Branch if Date and Time message is
07 28 A7 008E 265 DAP$Q_SYSCAP(R7),10$ : not supported by partner
04 56 EB 0091 266 BLBS R6,10$ : This XAB is not an input on close
0094 267 $SETBIT #DAP$V_DSP_TIM,R2 : Update request mask
05 0098 268 10$: RSB : Exit
0099 269 :
0099 270 :+
0099 271 : This routine handles the Protection XAB.
0099 272 :-
0099 273 :
0099 274 PROTECTION: : Entry point
51 10 9A 0099 275 MOVZBL #XAB$C_PROLEN_V3,R1 : Get minimum (i.e., V3) length of XAB
CD 10 009C 276 BSBB VALIDATE XAB : Check length and accessibility
0110 C7 D5 009E 277 TSTL NWA$L_PROXABADR(R7) : Declare error as this is a duplicate

```

```

0110 C7 B2 12 00A2 278 BNEQ ERRIMX ; XAB
          53 D0 00A4 279 MOVL R3,NWASL PROXABADR(R7) ; Save address of Protection XAB
          1B E1 00A9 280 BBC #DAPSV PROXAB,- ; Branch if Protection message is
0C 28 A7 00AB 281 DAPSQ SYSCAP(R7),20$ ; not supported by partner
          05 56 E9 00AE 282 BLBC R6,10$ ; An additional system capabilities
          2C E1 00B1 283 BBC #DAPSV CHGPROCLS,- ; check is required if this is a
04 28 A7 00B3 284 DAPSQ SYSCAP(R7),20$ ; change operation
          00B6 285 10$: $SETBIT #DAPSV_DSP_PRO,R2 ; Update request mask
          05 00BA 286 20$: RSB ; Exit
          00BB 287
          00BB 288 ;+
          00BB 289 ; This routine handles the Allocation XAB.
          00BB 290 ; -
          00BB 291
          00BB 292 ALLOCATION: ; Entry point
          51 20 9A 00BB 293 MOVZBL #XABSC_ALLLEN,R1 ; Get length of XAB
          AB 10 00BE 294 BSBW VALIDATE XAB ; Check length and accessibility
01 011C C7 96 00C0 295 INCB NWASB_AL[XABCNT(R7) ; Increment XAB counter
01 011C C7 91 00C4 296 CMPB NWASB_ALLXABCNT(R7),#1 ; Branch if this is first
          08 13 00C9 297 BEQL 10$ ; Allocation XAB in chain
          14 64 91 00CB 298 CMPB XABSB_COD(R4),#XABSC_ALL ; Check previous XAB in chain;
          11 13 00CE 299 BEQL 20$ ; it must also be an Allocation XAB
          FF83 31 00D0 300 BRW ERRIMX ; else this XAB is out of order
0100 C7 53 D0 00D3 301 10$: MOVL R3,NWASL ALLXABADR(R7) ; Save address of first Allocation XAB
          17 E1 00D8 302 BBC #DAPSV_AL[XAB,- ; Branch if Allocation message is
          04 28 A7 00DA 303 DAPSQ SYSCAP(R7),20$ ; not supported by partner
          00DD 304 $SETBIT #DAPSV_DSP_ALL,R2 ; Update request mask
          05 00E1 305 20$: RSB ; Exit
          00E2 306
          00E2 307 ;+
          00E2 308 ; This routine handles the Key Definition XAB.
          00E2 309 ; -
          00E2 310
          00E2 311 KEY_DEFINITION: ; Entry point
          51 40 8F 9A 00E2 312 MOVZBL #XABSC_KEYLEN_V2,R1 ; Get minimum (i.e., V2) length of XAB
          FF82 30 00E6 313 BSBW VALIDATE XAB ; Check length and accessibility
01 011D C7 96 00E9 314 INCB NWASB_KEYXABCNT(R7) ; Increment XAB counter
01 011D C7 91 00ED 315 CMPB NWASB_KEYXABCNT(R7),#1 ; Branch if this is first
          08 13 00F2 316 BEQL 10$ ; Key Definition XAB in chain
          15 64 91 00F4 317 CMPB XABSB_COD(R4),#XABSC_KEY ; Check previous XAB in chain;
          11 13 00F7 318 BEQL 20$ ; it must also be a Key Definition XAB
          FF5A 31 00F9 319 BRW ERRIMX ; else this XAB is out of order
010C C7 53 D0 00FC 320 10$: MOVL R3,NWASL KEYXABADR(R7) ; Save address of first Key Def XAB
          16 E1 0101 321 BBC #DAPSV_KEYXAB,- ; Branch if Key Definition message is
          04 28 A7 0103 322 DAPSQ SYSCAP(R7),20$ ; not supported by partner
          0106 323 $SETBIT #DAPSV_DSP_KEY,R2 ; Update request mask
          05 010A 324 20$: RSB ; Exit
          010B 325
          010B 326 ;+
          010B 327 ; This routine handles the Summary XAB.
          010B 328 ; -
          010B 329
          010B 330 SUMMARY: ; Entry point
          51 0C 9A 010B 331 MOVZBL #XABSC_SUMLLEN,R1 ; Get length of XAB
          FF5A 30 010E 332 BSBW VALIDATE XAB ; Check length and accessibility
0118 C7 D5 0111 333 TSTL NWASL_SUMXABADR(R7) ; Declare error as this is a duplicate
          55 12 0115 334 BNEQ ERRIMX1 ; XAB

```

```

0118 C7 53 D0 0117 335      MOVL   R3,NWASL_SUMXABADR(R7) ; Save address of Summary XAB
          18 E1 011C 336      BBC     #DAPSV_SOMXAB,-         ; Branch if Summary message is
04 28 A7   011E 337      DAP$Q SYSCAP(R7),10$         ; not supported by partner
          05 0121 338      $SETBIT #DAP$V_DSP_SUM,R2    ; Update request mask
          0125 339 10$:   RSB                                     ; Exit
          0126 340
          0126 341 :+
          0126 342 : This routine handles the File Header Characteristics XAB.
          0126 343 :
          0126 344 : Note: The File Header Characteristics XAB is supported in DAP through the
          0126 345 : DAP Attributes message. Thus there is no system capabilities check
          0126 346 : associated with it.
          0126 347 :-
          0126 348
          0126 349 FILE_HEADER:                                     ; Entry point
51 2C 9A 0126 350      MOVZBL #XAB$C_FHCLLEN,R1          ; Get length of XAB
          FF3F 30 0129 351      BSBW   VALIDATE_XAB              ; Check length and accessibility
0108 C7 D5 012C 352      TSTL   NWASL_FHCXABADR(R7)         ; Declare error as this is a duplicate
          3A 12 0130 353      BNEQ   ERRIMX1                  ; XAB
0108 C7 53 D0 0132 354      MOVL   R3,NWASL_FHCXABADR(R7)         ; Save address of File Header Char XAB
          0137 355      $SETBIT #DAP$V_DSP_ATT,R2            ; Update request mask
          05 0138 356      RSB                                     ; Exit
          013C 357
          013C 358 :+
          013C 359 : This routine handles the Revision Date and Time XAB.
          013C 360 :
          013C 361 : Note: Both the Date and Time XAB and the Revision Date and Time XAB are
          013C 362 : supported in DAP through the DAP Date and Time message.
          013C 363 :-
          013C 364
          013C 365 REV_DATE TIME:                                     ; Entry point
51 14 9A 013C 366      MOVZBL #XAB$C_RDTLEN,R1          ; Get length of XAB
          FF29 30 013F 367      BSBW   VALIDATE_XAB              ; Check length and accessibility
0114 C7 D5 0142 368      TSTL   NWASL_RDTXABADR(R7)         ; Declare error as this is a duplicate
          24 12 0146 369      BNEQ   ERRIMX1                  ; XAB
0114 C7 53 D0 0148 370      MOVL   R3,NWASL_RDTXABADR(R7)         ; Save address of Rev Date and Time XAB
          1A E1 014D 371      BBC     #DAPSV_TIMXAB,-         ; Branch if Date and Time message
0C 28 A7   014F 372      DAP$Q S/SCAP(R7),20$             ; not supported by partner
          05 56 E9 0152 373      BLBC   R6,10$                ; An additional system capabilities
          28 E1 0155 374      BBC     #DAP$V_CHGTIMCLS,-       ; check is required if this is a
04 28 A7   0157 375      DAP$Q SYSCAP(R7),20$             ; change operation
          05 015A 376 10$:   $SETBIT #DAP$V_DSP_TIM,R2        ; Update request mask
          015E 377 20$:   RSB                                     ; Exit
          015F 378
          015F 379 :+
          015F 380 : This routine handles the Journaling XAB.
          015F 381 :
          015F 382 : Note: This XAB is not supported for network use--it will be ignored unless
          015F 383 : the journaling options field is non-zero on create. Consequently, this
          015F 384 : routine saves the JOP field in the NWA for use by NT$CREATE.
          015F 385 :-
          015F 386
          015F 387 JOURNALING:                                     ; Entry point
51 3C 9A 015F 388      MOVZBL #XAB$C_JNLLEN,R1          ; Get length of XAB
          FF06 30 0162 389      BSBW   VALIDATE_XAB              ; Check length and accessibility
          08 A3 B0 0165 390      MOVL   XAB$W_JOP(R3),-         ; Save journaling options value in NWA
011E C7   0168 391      NWASW_JNLXABJOP(R7)                   ;

```

NTOSCNXAB
V04-000

SCAN XAB CHAIN
NT\$SCAN_XABCHN - SCAN XAB CHAIN

M 6

16-SEP-1984 00:07:06 VAX/VMS Macro V04-00
5-SEP-1984 16:21:07 [RMS.SRC]NTOSCNXAB.MAR;1

Page 9
(3)

N
V

	05	016B	392		RSB	
		016C	393			
FEE7	31	016C	394	ERRIMX1:BRW	ERRIMX	

; Exit

; Branch aid

```

016F 396 .SBTTL NT$SCAN_KEYXAB - SCAN KEY DEFINITION XAB
016F 397 .SBTTL NT$SCAN_ALLXAB - SCAN ALLOCATION XAB
016F 398 :++
016F 399 : NT$SCAN_KEYXAB - scans a specific Key Definition XAB without scanning the
016F 400 : entire XAB chain.
016F 401 : NT$SCAN_ALLXAB - scans a specific Allocation XAB without scanning the
016F 402 : entire XAB chain.
016F 403 :
016F 404 : Calling Sequence:
016F 405 :
016F 406 :     BSBW    NT$SCAN_KEYXAB
016F 407 :     BSBW    NT$SCAN_ALLXAB
016F 408 :
016F 409 : Input Parameters:
016F 410 :
016F 411 :     R6      Allocation or Key Definition XAB address
016F 412 :     R7      NWA (=DAP) address
016F 413 :     R8      FAB address
016F 414 :     R9      IFAB address
016F 415 :     R10     IFAB/FWA address
016F 416 :     R11     Impure Area address
016F 417 :
016F 418 : Implicit Inputs:
016F 419 :
016F 420 :     None
016F 421 :
016F 422 : Output Parameters:
016F 423 :
016F 424 :     R0      Status code (RMS)
016F 425 :     R1      Destroyed
016F 426 :     R3      Destroyed
016F 427 :
016F 428 : Implicit Outputs:
016F 429 :
016F 430 :     None
016F 431 :
016F 432 : Completion Codes:
016F 433 :
016F 434 :     Standard RMS Completion codes
016F 435 :
016F 436 : Side Effects:
016F 437 :
016F 438 :     User XAB is probed for writeability
016F 439 :
016F 440 : --
016F 441 :
016F 442 NT$SCAN_KEYXAB:: : Entry point
51 40 8F 9A 016F 443     MOVZBL #XAB$C_KEYLEN_V2,R1 : Get minimum (i.e., V2) length of XAB
      03 11 0173 444     BRB COMMON_SCAN : Join common code
      0175 445 NT$SCAN_ALLXAB:: : Entry point
51 20 9A 0175 446     MOVZBL #XAB$C_ALLLEN,R1 : Get length of XAB
      0178 447 COMMON_SCAN: : Common code
53 56 D0 0178 448     MOVL R6,R3 : Get address of XAB to probe
      017B 449     IFNORD #XAB$N_NXT+4,(R3),- : Probe for readability thru NXT field
      017B 450     10$,IFB$B_MODE(R9) : of XAB and branch on failure
50 01 A3 9A 0182 451     MOVZBL XAB$B_BLNTR3),R0 : Get stated length of block
51 50 D1 0186 452     CMPL R0,R1 : Compare against expected length
    
```

OB	1F	0189	453	BLSSU	10\$:	Branch if too small
		018B	454	IFNOWRT	RO,(R3),10\$,-	:	Probe for writeability and branch on
		018B	455		IFB\$B_MODE(R9)	:	failure
		0192	456	RMSSUC		:	Return success
	05	0195	457	RSB		:	Exit
FEC8	31	0196	458 10\$:	BRW	ERRXAB	:	Failure

```

0199 460 .SBTTL NT$SCAN_NAMBLK - SCAN NAME BLOCK
0199 461
0199 462 :++
0199 463 : NT$SCAN_NAMBLK - scans the user Name Block and checks FAL's capabilities
0199 464 : to determine if a DAP (resultant) Name message should be requested
0199 465 : to be returned by the remote FAL.
0199 466 :
0199 467 : An updated message request mask is returned in R2.
0199 468 :
0199 469 : Calling Sequence:
0199 470 :
0199 471 : BSBW NT$SCAN_NAMBLK
0199 472 :
0199 473 : Input Parameters:
0199 474 :
0199 475 : R2 Message request mask
0199 476 : R7 NWA (=DAP) address
0199 477 : R8 FAB address
0199 478 : R9 IFAB address
0199 479 : R10 IFAB/FWA address
0199 480 : R11 Impure Area address
0199 481 :
0199 482 : Implicit Inputs:
0199 483 :
0199 484 : User Name Block
0199 485 :
0199 486 : Output Parameters:
0199 487 :
0199 488 : R0 Status code (RMS)
0199 489 : R1 Destroyed
0199 490 : R2 Updated message request mask
0199 491 :
0199 492 : Implicit Outputs:
0199 493 :
0199 494 : None
0199 495 :
0199 496 : Completion Codes:
0199 497 :
0199 498 : Standard RMS Completion codes
0199 499 :
0199 500 : Side Effects:
0199 501 :
0199 502 : User Name Block is probed for writeability
0199 503 :
0199 504 :--
0199 505

```

```

00C0 8F BB
57 28 A8 D0
19 13 01A1
FE5A' 30 01A3
13 50 E9 01A6
04 A7 D5 01A9
0E '3 01AC
00C0 8F BA 01AE
28 E1 01B2
09 28 A7 01B4

```

```

0199 506 NT$SCAN_NAMBLK:: : Entry point
0199 507 PUSHF #*M<R6,R7> : Save registers used
019D 508 MOVL FAB$$_NAM(R8),R7 : Get Name block address
01A1 509 BEQL 10$ : Branch if none
01A3 510 BSBW RMSCHKNAM : Check Name block validity
01A6 511 BLBC R0,10$ : Branch on error
01A9 512 TSTL NAM$$_RSA(R7) : Check for resultant string address
01AC 513 BEQL 10$ : Branch if none
01AE 514 POPR #*M<R6,R7> : Restore registers
01B2 515 BBC #DAP$V_NAMMSG,- : Branch if partner does not support
01B4 516 DAP$$_SYSCAP(R7),20$ : Name message

```

```
00C0 8F 05 01B7 517 $SETBIT #DAP$V_DSP_NAM,R2 ; Request Name message
05 01BB 518 RSB ; Exit
BA 01BC 519 10$: POPR #^M<R6,R7> ; Restore registers
05 01C0 520 20$: RSB ; Exit
01C1 521
01C1 522 .END ; End of module
```



```

$$PSECT_EP = 00000000
$$COUNT = 00000011
$$RMSTEST = 0000001A
$$RMS_PBUGCHK = 00000010
$$RMS_TBUGCHK = 00000008
$$RMS_UMODE = 00000004
ALLOCATION = 000000BB R 01
CHKXAB = 00000017 R 01
COMMON_SCAN = 00000178 R 01
DAPSB_ACCFJNC = 00000040
DAPSB_ACCOPT = 00000041
DAPSB_DECVAR = 00000047
DAPSB_ECONUM = 00000045
DAPSB_FAC = 00000042
DAPSB_FILESYS = 00000043
DAPSB_OSTYPE = 00000042
DAPSB_SHR = 00000043
DAPSB_USRNUM = 00000046
DAPSB_USRVER = 00000048
DAPSB_VERNUM = 00000044
DAPSM_DSP_3NAM = 00000200
DAPSM_GET = 00000002
DAPSM_GO_NOGO = 00000010
DAPSM_MSE = 00000010
DAPSM_TMP1$ = 000000C0
DAPSM_TMP2$ = 0000FC00
DAPSQ_FILESPEC = 00000044
DAPSQ_PASSWORD = 00000050
DAPSQ_SYSCAP = 00000028
DAPSV_ALLXAB = 00000017
DAPSV_CHGPROCLS = 0000002C
DAPSV_CHGTIMCLS = 0000002B
DAPSV_DSP_ALL = 00000002
DAPSV_DSP_ATT = 00000000
DAPSV_DSP_KEY = 00000001
DAPSV_DSP_NAM = 00000008
DAPSV_DSP_PRO = 00000005
DAPSV_DSP_SUM = 00000003
DAPSV_DSP_TIM = 00000004
DAPSV_KEYXAB = 00000016
DAPSV_NAMMSG = 00000028
DAPSV_PROXAB = 0000001B
DAPSV_SUMXAB = 00000018
DAPSV_TIMXAB = 0000001A
DAPSW_BUFSIZ = 00000040
DAPSW_DISPLAY1 = 0000004C
DATE_TIME = 0000007C R 01
ERRCOD = 0000004D R 01
ERRIMX = 00000056 R 01
ERRIMX1 = 0000016C R 01
ERRXAB = 00000061 R 01
ERRXAB2 = 0000005F R 01
EXIT = 00000049 R 01
FABSL_NAM = 00000028
FABSL_STV = 0000000C
FABSL_XAB = 00000024
FILE_READER = 00000126 R 01

```

```

IFBSB_MODE = 0007000A
JOURNALING = 0000015F R 01
KEY_DEFINITION = 000000E2 R 01
NAMSL_RSA = 00000004
NTSSCAN_ALLXAB = 00000175 RG 01
NTSSCAN_KEYXAB = 0000016F RG 01
NTSSCAN_NAMBLK = 00000199 RG 01
NTSSCAN_XABCHN = 00000000 RG 01
NWSB_ACLXABCNT = 0000011C
NWSB_DAP_RAC = 000000C9
NWSB_FILESYS = 000000C5
NWSB_KEYXABCNT = 0000011D
NWSB_NETSTRSIZ = 0000016F
NWSB_NODBUFSIZ = 00000168
NWSB_ORG = 000000C6
NWSB_OSTYPE = 000000C4
NWSB_RFM = 000000C7
NWSB_RMS_RAC = 000000C8
NWSB_BLN = 00000800
NWSB_SCAN = 00000020
NWSB_BLN = 00000800
NWSL_ALLXABADR = 00000100
NWSL_DATXABADR = 00000104
NWSL_DEV = 000000C0
NWSL_FHCXABADR = 00000108
NWSL_KEYXABADR = 0000010C
NWSL_MSG_MASK = 000000D4
NWSL_PROXABADR = 00000110
NWSL_RDXABADR = 00000114
NWSL_SAVE_FLGS = 00000128
NWSL_SUMXABADR = 00000118
NWSL_THREAD = 000000FC
NWSL_XLTATTR = 00000238
NWSL_XLTBUFLG = 0000022C
NWSL_XLTCNT = 00000228
NWSL_XLTMAXIDX = 00000234
NWSL_XLTSIZ = 00000230
NWSQ_ACS = 00000244
NWSQ_BIGBUF = 00000170
NWSQ_BLD = 000000F0
NWSQ_FLG = 00000000
NWSQ_INODE = 0000025C
NWSQ_IOSB = 000000D8
NWSQ_LNODE = 00000160
NWSQ_LOGNAME = 0000023C
NWSQ_NCB = 00000264
NWSQ_RCV = 000000E0
NWSQ_SAVE_DESC = 00000120
NWSQ_XLTBUF1 = 0000024C
NWSQ_XLTBUF2 = 00000254
NWSQ_XMT = 000000E8
NWSQ_ACSBUF = 0000026C
NWSQ_AUXBUF = 000005E0
NWSQ_DAP = 00000000
NWSQ_INODEBUF = 000004AC
NWSQ_ITM_ATTR = 00000200
NWSQ_ITM_END = 00000224

```

NWAST_ITM_LST	00000200		
NWAST_ITM_MAXINDX	00000218		
NWAST_ITM_STRING	0000020C		
NWAST_NCBBUF	0000052C		
NWAST_NODEBUF	00000169		
NWAST_RCVBUF	000001A0		
NWAST_SCAN	00000100		
NWAST_TEMP	00000120		
NWAST_XLTBUF1	000002AC		
NWAST_XLTBUF2	000003AC		
NWAST_XMTBUF	000003C0		
NWASW_BUILD	000000D2		
NWASW_DAPBUFSIZ	000000CA		
NWASW_DIR_OFF	000000CC		
NWASW_DISPLAY	000000D0		
NWASW_FIL_OFF	000000CE		
NWASW_JNLXABJOP	0000011E		
NXTXAB	00000010	R	01
PROTECTION	00000099	R	01
REV DATE TIME	0000013C	R	01
RMSCHKNAM	*****	X	01
RMS\$COD	*****	X	01
RMS\$IMX	*****	X	01
RMS\$XAB	*****	X	01
SETSTV	00000066	R	01
SUMMARY	0000010B	R	01
VALIDATE_XAB	0000006B	R	01
XAB\$B_BLN	= 00000001		
XAB\$B_COD	= 00000000		
XAB\$C_ALL	= 00000014		
XAB\$C_ALLLEN	= 00000020		
XAB\$C_DAT	= 00000012		
XAB\$C_DATLEN_V2	= 00000024		
XAB\$C_FHC	= 00000010		
XAB\$C_FHCLEN	= 0000002C		
XAB\$C_JNL	= 00000022		
XAB\$C_JNLLEN	= 0000003C		
XAB\$C_KEY	= 00000015		
XAB\$C_KEYLEN_V2	= 00000040		
XAB\$C_PRO	= 00000013		
XAB\$C_PROLEN_V3	= 00000010		
XAB\$C_RDT	= 0000001E		
XAB\$C_RDTLEN	= 00000014		
XAB\$C_SUM	= 00000016		
XAB\$C_SUMLLEN	= 0000000C		
XAB\$L_NXT	= 00000004		
XAB\$W_JOP	= 00000008		

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
NFSNETWORK	000001C1 (449.)	01 (1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE
\$AB\$\$	00000800 (2048.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.08	00:00:01.05
Command processing	143	00:00:00.80	00:00:04.52
Pass 1	334	00:00:12.16	00:00:35.27
Symbol table sort	0	00:00:01.49	00:00:03.11
Pass 2	103	00:00:02.39	00:00:09.14
Symbol table output	20	00:00:00.15	00:00:00.38
Psect synopsis output	1	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	635	00:00:17.10	00:00:53.51

The working set limit was 1500 pages.
64416 bytes (126 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1101 non-local and 22 local symbols.
522 source lines were read in Pass 1, producing 14 object records in Pass 2.
34 pages of virtual memory were used to define 33 macros.

! Macro library statistics !

Macro library name	Macros defined
-\$255\$DUA28:[RMS.OBJ]RMS.MLB;1	23
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	2
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	29

1326 GETS were required to define 29 macros.

There were no errors, warnings or information messages.

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

