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

FILEID**NTOENCODE

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

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

```
0000     1          $BEGIN  NTOENCODE,000,NH$NETWORK,<DAP MESSAGE ENCODE ROUTINES>
0000     2
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28
0000    29 ;++
0000    30 ; Facility: RMS
0000    31 ;
0000    32 ; Abstract:
0000    33 ;
0000    34 ;       This module contains support routines that encode (build) portions of
0000    35 ;       A DAP message. Included are routines to build a message header and to
0000    36 ;       convert binary data to extensible or image format.
0000    37 ;
0000    38 ; Environment: VAX/VMS, executive mode
0000    39 ;
0000    40 ; Author: James A. Krycka,      Creation Date: 16-JUN-1977
0000    41 ;
0000    42 ; Modified By:
0000    43 ;
0000    44 ;       V03-001 JAK0145         J A Krycka      12-APR-1984
0000    45 ;                Track changes in DAP message building algorithm.
0000    46 ;
0000    47 ;--
```

```
0000     49          .SBTTL  DECLARATIONS
0000     50
0000     51 ;
0000     52 ; I:  ude Files:
0000     53 ;
0000     54
0000     55          $DAPHDRDEF                      ; Define DAP message header
0000     56          $DAPCNFDEF                      ; Define DAP Configuration message
0000     57          $NWADEF                         ; Define Network Work Area symbols
0000     58
0000     59 ;
0000     60 ; Macros:
0000     61 ;
0000     62 ;        None
0000     63 ;
0000     64 ; Equated Symbols:
0000     65 ;
0000     66
0000     67          ASSUME  NWA$Q_FLG EQ 0
0000     68
0000     69 ;
0000     70 ; Own Storage:
0000     71 ;
0000     72 ;        None
0000     73 ;
```

```
                        .SBTTL  NT$BUILD_HEAD - BUILD DAP MESSAGE HEADER

                ;++
                ; NT$BUILD_HEAD - obtains a buffer and constructs a DAP message header in it.
                ;       NT$BUILD_TAIL is a companion routine that is called after the message
                ;       body has been formed to update the length value in the header. Both
                ;       1-byte and 2-byte length fields are supported for DAP message blocking.
                ;
                ;       Note that the algorithm for building the header does not support the
                ;       use of optional fields such as STREAMID or SYSPEC.
                ;
                ; Calling Sequence:
                ;
                ;       BSBW    NT$BUILD_HEAD
                ;
                ; Input Parameters:
                ;
                ;       R0      DAP message type value
                ;       R7      Address of NWA (=DAP)
                ;
                ; Implicit Inputs:
                ;
                ;       DAP$V_BIGBLK
                ;       DAP$V_MSGBLK
                ;       NWA$Q_XMT
                ;       NWA$V_LAST_MSG
                ;
                ; Output Parameters:
                ;
                ;       R5      Address of next byte available for message body
                ;
                ; Implicit Outputs:
                ;
                ;       NWA$Q_BLD
                ;       DAP message header is placed in the message buffer.
                ;
                ; Completion Codes:
                ;
                ;       None
                ;
                ; Side Effects:
                ;
                ;       None
                ;
                ;--
```

```
                     0000   121 ;++
                     0000   122 ; On exit from NT$BUILD_HEAD the incomplete message header will be either
                     0000   123 ; 2, 3, or 4 bytes long in one of the following three formats:
                     0000   124 ;
                     0000   125 ;       TYPE=msg#       TYPE=msg#       TYPE=8
                     0000   126 ;       FLAGS=0         FLAGS=2         FLAGS=6
                     0000   127 ;                       LENGTH=0        LENGTH=0
                     0000   128 ;                                       LEN256=0
                     0000   129 ;
                     0000   130 ; Note: The four byte format containing the extended length field is used only
                     0000   131 ;       for a DAP Data message.
                     0000   132 ;--
                     0000   133
                     0000   134         ASSUME  DAP$V_STREAMID+1 EQ DAP$V_LENGTH
                     0000   135         ASSUME  DAP$V_LENGTH+1 EQ DAP$V_LEN256
                     0000   136
                     0000   137 NT$BUILD_HEAD::                          ; Entry point
     00E8 C7    C1   0000   138         ADDL3   NWA$Q_XMT(R7),-             ; Compute address of first unused byte
55   00EC C7         0004   139                 NWA$Q_XMT+4(R7),R5           ;  in transmit buffer and return in R5
00F4 C7    55   D0   0008   140         MOVL    R5,NWA$Q_BLD+4(R7)          ; Update build message descriptor
       85   50   90   000D   141         MOVB    R0,(R5)+                    ; Store DAP message type value
     18 67    00   E0   0010   142         BBS     #NWA$V_LAST_MSG,(R7),20$; Branch if this will be last message
                     0014   143                                             ;  to block, thus requiring no length
                     0014   144                                             ;  field in header
          08   50   91   0014   145         CMPB    R0,#DAP$K_DAT_MSG           ; Branch if this is not a Data message
               0B   12   0017   146         BNEQ    10$                         ;  (because only a Data message is
                     0019   147                                             ;  potentially longer than 255 bytes)
               14   E1   0019   148         BBC     #DAP$V_BIGBLK,-             ; Branch if 2-byte length field is
         06 28 A7         001B   149                 DAP$Q_SYSCAP(R7),10$        ;  not supported by partner
          85   06   90   001E   150         MOVB    #<<DAP$M_LENGTH>!-          ; Store FLAGS field indicating that
                     0021   151                   <DAP$M_LEN256>!-          ;  an extended length field will be
                     0021   152                   0>,(R5)+                  ;  included
               85   B4   0021   153         CLRW    (R5)+                       ; Reserve space for 2-byte length value
                     0023   154                                             ;  (to be filled in by NT$BUILD_TAIL)
               05   0023   155         RSB                                 ; Exit
               12   E1   0024   156 10$:   BBC     #DAP$V_MSGBLK,-             ; Branch if 1-byte length field is
         03 28 A7         0026   157                 DAP$Q_SYSCAP(R7),20$        ;  not supported by partner
          85   02   90   0029   158         MOVB    #DAP$M_LENGTH,(R5)+         ; Store FLAGS field
               85   94   002C   159 20$:   CLRB    (R5)+                       ; Reserve space for 1-byte length value
                     002E   160                                             ;  (to be filled in by NT$BUILD_TAIL)
                     002E   161                                             ;  OR overwrite FLAGS field with zero
                     002E   162                                             ;  if message blocking is not being used
               05   002E   163         RSB                                 ; Exit
```

```
002F    165          .SBTTL  NT$BUILD_TAIL - COMPLETE DAP MESSAGE HEADER
002F    166
002F    167 ;++
002F    168 ; NT$BUILD_TAIL - completes construction of the DAP message header that
002F    169 ;       NT$BUILD_HEAD initiated by updating the length value in the header.
002F    170 ;       Both 1-byte and 2-byte length fields are supported for DAP message
002F    171 ;       blocking.
002F    172 ;
002F    173 ;       Note that the algorithm for building the header does not support the
002F    174 ;       use of optional fields such as STREAMID or SYSPEC.
002F    175 ;
002F    176 ;       Note also that the LENGTH field will be converted to a STREAMID field
002F    177 ;       with a value of zero if a 1-byte length field was allocated by
002F    178 ;       NT$BUILD_HEAD, but the message body turned out to be more than 255
002F    179 ;       bytes long which cannot be represented in a single byte field.
002F    180 ;
002F    181 ; Calling Sequence:
002F    182 ;
002F    183 ;       BSBW    NT$BUILD_TAIL
002F    184 ;
002F    185 ; Input Parameters:
002F    186 ;
002F    187 ;       R5      Address of last byte of message + 1
002F    188 ;       R7      Address of NWA (=DAP)
002F    189 ;
002F    190 ; Implicit Inputs:
002F    191 ;
002F    192 ;       NWA$Q_BLD
002F    193 ;       DAP message header
002F    194 ;
002F    195 ; Output Parameters:
002F    196 ;
002F    197 ;       R0-R1   Destroyed
002F    198 ;
002F    199 ; Implicit Outputs:
002F    200 ;
002F    201 ;       NWA$Q_BLD
002F    202 ;       LENGTH and LEN256 fields (if present) in the header are updated.
002F    203 ;
002F    204 ; Completion Codes:
002F    205 ;
002F    206 ;       None
002F    207 ;
002F    208 ; Side Effects:
002F    209 ;
002F    210 ;       LENGTH field may be converted to a STREAMID field with a value of zero.
002F    211 ;
002F    212 ;--
```

```
                                 002F   214 ;++
                                 002F   215 ; On exit from NT$BUILD_TAIL the completed message header will be either
                                 002F   216 ; 2, 3, or 4 bytes long in one of the following four formats:
                                 002F   217 ;
                                 002F   218 ;       TYPE=msg#       TYPE=msg#       TYPE=8          TYPE=8
                                 002F   219 ;       FLAGS=0         FLAGS=2         FLAGS=6         FLAGS=1
                                 002F   220 ;                       LENGTH=size     LENGTH=size0    STREAMID-0
                                 002F   221 ;                                       LEN256=size1
                                 002F   222 ;
                                 002F   223 ; Note: The four byte format containing the extended length field is used only
                                 002F   224 ;       for a DAP Data message.
                                 002F   225 ;--
                                 002F   226
                                 002F   227          ASSUME  DAP$V_STREAMID+1 EQ DAP$V_LENGTH
                                 002F   228          ASSUME  DAP$V_LENGTH+1 EQ DAP$V_LEN256
                                 002F   229
                                 002F   230 NT$BUILD_TAIL::                          ; Entry point
00F0 C7   55   00F4 C7   C3      002F   231          SUBL3   NWA$Q_BLD+4(R7),R5,-    ; Compute size of DAP message and
                                 0037   232                  NWA$Q_BLD(R7)          ;  update build descriptor
          50   00F0 C7   7D      0037   233          MOVQ    NWA$Q_BLD(R7),R0       ; Put message descriptor in <R0,R1>
                        0A   E1  003C   234          BBC     #<DAP$V_LEN256+8>,-    ; Branch if 2-byte length field was
                     06 61       003E   235                  (R1),10$               ;  not allocated in message header
             02 A1   50   04   A3 0040   236          SUBW3   #4,R0,2(R1)            ; Compute size of message body and
                                 0045   237                                         ;  store value in <LEN256,LENGTH> field
                           05    0045   238          RSB                            ; Exit
                        09   E1  0046   239 10$:     BBC     #<DAP$V_LENGTH+8>,-    ; Branch if 1-byte length field was
                     13 61       0048   240                  (R1),30$               ;  not allocated in message header
                50   03   A2     004A   241          SUBW2   #3,R0                  ; Compute size of message body
             00FF 8F   50   B1   004D   242          CMPW    R0,#255                ; Branch if length of message body
                        05   1A  0052   243          BGTRU   20$                    ;  will not fit in LENGTH field
             02 A1   50   90     0054   244          MOVB    R0,2(R1)               ; Update LENGTH field
                           05    0058   245          RSB                            ; Exit
             01 A1   01   90     0059   246 20$:     MOVB    #DAP$M_STREAMID,1(R1)  ; Rewrite FLAGS field (converting
                                 005D   247                                         ;  LENGTH field into STREAMID field)
                           05    005D   248 30$:     RSB                            ; Exit
```

```
                     005E   250          .SBTTL  NT$CVT_BN4_EXT - CONVERT BINARY TO EXTENSIBLE
                     005E   251
                     005E   252 ;++
                     005E   253 ; NT$CVT_BN4_EXT - converts an unsigned longword value to an extensible field
                     005E   254 ;       format and stores the result in a minimal number of bytes.
                     005E   255 ;
                     005E   256 ; Calling Sequence:
                     005E   257 ;
                     005E   258 ;       BSBW    NT$CVT_BN4_EXT
                     005E   259 ;
                     005E   260 ; Input Parameters:
                     005E   261 ;
                     005E   262 ;       R1      Binary value to convert and store
                     005E   263 ;       R5      Address of next byte in buffer to store result
                     005E   264 ;
                     005E   265 ; Implicit Inputs:
                     005E   266 ;
                     005E   267 ;       None
                     005E   268 ;
                     005E   269 ; Output Parameters:
                     005E   270 ;
                     005E   271 ;       R1      Zeroed
                     005E   272 ;       R5      Address of last byte of result + 1
                     005E   273 ;
                     005E   274 ; Implicit Outputs:
                     005E   275 ;
                     005E   276 ;       None
                     005E   277 ;
                     005E   278 ; Completion Codes:
                     005E   279 ;
                     005E   280 ;       None
                     005E   281 ;
                     005E   282 ; Side Effects:
                     005E   283 ;
                     005E   284 ;       None
                     005E   285 ;
                     005E   286 ;--
                     005E   287
                     005E   288 NT$CVT_BN4_EXT::                     ; Entry point
        85    51     90  005E   289          MOVB    R1,(R5)+        ; Copy 7 bits to DST byte--the high
                     0061   290                                      ;  bit will be corrected later
     51    7F  8F    8A  0061   291          BICB2   #^X7F,R1        ; Discard SRC bits just copied
51   51    F9  8F    9C  0065   292          ROTL    #-7,R1,R1       ; Move next 7 bits into place
                 07    13  006A   293          BEQL    10$             ; All done if remaining SRC bits
                     006C   294                                      ;  are zero
  FF A5    80  8F    88  006C   295          BISB2   #^X80,-1(R5)    ; Set extensible bit in DST byte
                 EB    11  0071   296          BRB     NT$CVT_BN4_EXT  ;  and process next byte
                       05  0073   297 10$:     RSB                     ; Exit
```

```
                     0074   299          .SBTTL  NT$CVT_BN8_EXT - CONVERT BINARY TO EXTENSIBLE
                     0074   300
                     0074   301 ;++
                     0074   302 ; NT$CVT_BN8_EXT - converts an unsigned quadword value to an extensible field
                     0074   303 ;       format and stores the result in a minimal number of bytes.
                     0074   304 ;
                     0074   305 ;       Note that only source bits 00-62 are used; bit 63 is ignored.
                     0074   306 ;
                     0074   307 ; Calling Sequence:
                     0074   308 ;
                     0074   309 ;       BSBW    NT$CVT_BN8_EXT
                     0074   310 ;
                     0074   311 ; Input Parameters:
                     0074   312 ;
                     0074   313 ;       R1      Binary value to convert and store (low order bits)
                     0074   314 ;       R2      Binary value to convert and store (high order bits)
                     0074   315 ;       R5      Address of next byte in buffer to store result
                     0074   316 ;
                     0074   317 ; Implicit Inputs:
                     0074   318 ;
                     0074   319 ;       None
                     0074   320 ;
                     0074   321 ; Output Parameters:
                     0074   322 ;
                     0074   323 ;       R1-R2   Zeroed
                     0074   324 ;       R5      Address of last byte of result + 1
                     0074   325 ;
                     0074   326 ; Implicit Outputs:
                     0074   327 ;
                     0074   328 ;       None
                     0074   329 ;
                     0074   330 ; Completion Codes:
                     0074   331 ;
                     0074   332 ;       None
                     0074   333 ;
                     0074   334 ; Side Effects:
                     0074   335 ;
                     0074   336 ;       None
                     0074   337 ;
                     0074   338 ;--
                     0074   339
                     0074   340 NT$CVT_BN8_EXT::                     ; Entry point
                     0074   341          $CLRBIT #31,R2              ; Clear high order bit so that zero
                     0078   342                                      ;  will always propogate on shift
          85    51   90   0078   343 10$:     MOVB    R1,(R5)+        ; Copy 7 bits to DST byte--the high
                     007B   344                                      ;  bit will be corrected later
     51    7F  8F   8A   007B   345          BICB2   #^X7F,R1        ; Discard SRC bits just copied
51   51    F9  8F   79   007F   346          ASHQ    #-7,R1,R1       ; Move next 7 bits into place
                07   13   0084   347          BEQL    20$             ; All done if remaining SRC bits
                     0086   348                                      ;  are zero
  FF A5    80  8F   88   0086   349          BISB2   #^X80,-1(R5)    ; Set extensible bit in DST byte
                EB   11   008B   350          BRB     10$             ;  and process next byte
                     008D   351 20$:     RSB                         ; Exit
```

```
                     008E    353          .SBTTL  NT$CVT_BN4_IMG - CONVERT BINARY TO IMAGE
                     008E    354
                     008E    355 ;++
                     008E    356 ; NT$CVT_BN4_IMG - converts an unsigned longwork value to an image field format
                     008E    357 ;       (counted binary string) and stores the result in a minimal number of
                     008E    358 ;       bytes.
                     008E    359 ;
                     008E    360 ; Calling Sequence:
                     008E    361 ;
                     008E    362 ;       BSBW    NT$CVT_BN4_IMG
                     008E    363 ;
                     008E    364 ; Input Parameters:
                     008E    365 ;
                     008E    366 ;       R1      Binary value to convert and store
                     008E    367 ;       R5      Address of next byte in buffer to store result
                     008E    368 ;
                     008E    369 ; Implicit Inputs:
                     008E    370 ;
                     008E    371 ;       None
                     008E    372 ;
                     008E    373 ; Output Parameters:
                     008E    374 ;
                     008E    375 ;       R1      Zeroed
                     008E    376 ;       R2      Destroyed
                     008E    377 ;       R5      Address of last byte of result + 1
                     008E    378 ;
                     008E    379 ; Implicit Outputs:
                     008E    380 ;
                     008E    381 ;       None
                     008E    382 ;
                     008E    383 ; Completion Codes:
                     008E    384 ;
                     008E    385 ;       None
                     008E    386 ;
                     008E    387 ; Side Effects:
                     008E    388 ;
                     008E    389 ;       None
                     008E    390 ;
                     008E    391 ;--
                     008E    392
                     008E    393 NT$CVT_BN4_IMG::                    ; Entry point
          52   55    D0  008E    394          MOVL    R5,R2          ; Save address of DST count byte
               85    94  0091    395          CLRB    (R5)+          ; Zero DST count byte
               51    D5  0093    396          TSTL    R1             ; Test value to convert
               0E    13  0095    397          BEQL    20$            ; All done if value is zero
          85   51    90  0097    398 10$:     MOVB    R1,(R5)+       ; Copy next byte to DST
               62    96  009A    399          INCB    (R2)           ; Increment byte counter
               51    94  009C    400          CLRB    R1             ; Discard byte just copied
51   51   F8 8F    9C  009E    401          ROTL    #-8,R1,R1      ; Move next byte into place
               F2    12  00A3    402          BNEQ    10$            ; There is more to do if any
                         00A5    403                                 ;  remaining bits are non-zero
                     05  00A5    404 20$:     RSB                    ; Exit
                         00A6    405
                         00A6    406          .END                   ; End of module
```

```
$$.PSECT_EP                     = 00000000
$$RMSTEST                       = 0000001A
$$RMS_PBUGCHK                   = 00000010
$$RMS_TBUGCHK                   = 00000008
$$RMS_UMODE                     = 00000004
DAP$B_BITCNT                      00000035
DAP$B_DECVER                      00000047
DAP$B_ECONUM                      00000045
DAP$B_FILESYS                     00000043
DAP$B_FLAGS                       00000031
DAP$B_LEN256                      00000034
DAP$B_LENGTH                      00000033
DAP$B_OSTYPE                      00000042
DAP$B_STREAMID                    00000032
DAP$B_TYPE                        00000030
DAP$B_USRNUM                      00000046
DAP$B_USRVER                      00000048
DAP$B_VERNUM                      00000044
DAP$K_DAT_MSG                   = 00000003
DAP$M_BITCNT                    = 00000008
DAP$M_LEN256                    = 00000004
DAP$M_LENGTH                    = 00000002
DAP$M_SEGMENT                   = 00000040
DAP$M_STREAMID                  = 00000001
DAP$M_TMP1$                     = 00000010
DAP$M_TMP2$                     = 00000080
DAP$Q_SYSCAP                      00000028
DAP$Q_SYSPEC                      00000038
DAP$V_BIGBLK                    = 00000014
DAP$V_LEN256                    = 00000002
DAP$V_LENGTH                    = 00000001
DAP$V_MSGBLK                    = 00000012
DAP$V_STREAMID                  = 00000000
DAP$W_BUFSIZ                      00000040
NT$BUILD_HEAD                     00000000 RG     01
NT$BUILD_TAIL                     0000002F RG     01
NT$CVT_BN4_EXT                    0000005E RG     01
NT$CVT_BN4_IMG                    0000008E RG     01
NT$CVT_BN8_EXT                    00000074 RG     01
NWA$B_ALLXABCNT                   0000011C
NWA$B_DAP_RAC                     000000C9
NWA$B_FILESYS                     000000C5
NWA$B_KEYXABCNT                   0000011D
NWA$B_NETSTRSIZ                   0000016F
NWA$B_NODBUFSIZ                   00000168
NWA$B_ORG                         000000C6
NWA$B_OSTYPE                      000000C4
NWA$B_RFM                         000000C7
NWA$B_RMS_RAC                     000000C8
NWA$C_BLN                         00000800
NWA$K_BLN                         00000800
NWA$L_ALLXABADR                   00000100
NWA$L_DATXABADR                   00000104
NWA$L_DEV                         000000C0
NWA$L_FHCXABADR                   00000108
NWA$L_KEYXABADR                   000C010C
NWA$L_MSG_MASK                    000000D4
NWA$L_PROXABADR                   00000110
NWA$L_RDTXABADR                   00000114
NWA$L_SAVE_FLGS                   00000128
NWA$L_SUMXABADR                   00000118
NWA$L_THREAD                      000000FC
NWA$L_XLTATTR                     00000238
NWA$L_XLTBUFFLG                   0000022C
NWA$L_XLTCNT                      00000228
NWA$L_XLTMAXINDX                  00000234
NWA$L_XLTSIZ                      00000230
NWA$Q_ACS                         00000244
NWA$Q_BIGBUF                      00000170
NWA$Q_BLD                         000000F0
NWA$Q_FLG                         00000000
NWA$Q_INODE                       0000025C
NWA$Q_IOSB                        000000D8
NWA$Q_LNODE                       00000160
NWA$Q_LOGNAME                     0000023C
NWA$Q_NCB                         00000264
NWA$Q_RCV                         000000E0
NWA$Q_SAVE_DESC                   00000120
NWA$Q_XLTBUF1                     0000024C
NWA$Q_XLTBUF2                     00000254
NWA$Q_XMT                         000000E8
NWA$T_ACSBUF                      0000026C
NWA$T_AUXBUF                      000005E0
NWA$T_DAP                         00000000
NWA$T_INODEBUF                    000004AC
NWA$T_ITM_ATTR                    00000200
NWA$T_ITM_END                     00000224
NWA$T_ITM_LST                     00000200
NWA$T_ITM_MAXINDX                 00000218
NWA$T_ITM_STRING                  0000020C
NWA$T_NCBBUF                      0000052C
NWA$T_NODEBUF                     00000169
NWA$T_RCVBUF                      000001A0
NWA$T_SCAN                        00000100
NWA$T_TEMP                        00000120
NWA$T_XLTBUF1                     000002AC
NWA$T_XLTBUF2                     000003AC
NWA$T_XMTBUF                      000003C0
NWA$V_LAST_MSG                  = 00000000
NWA$W_BUILD                       000000D2
NWA$W_DAPBUFSIZ                   000000CA
NWA$W_DIR_OFF                     000000CC
NWA$W_DISPLAY                     000000D0
NWA$W_FIL_OFF                     000000CE
NWA$W_JNLXABJOP                   0000011E
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
NH$NETWORK	000000A6 (166.)	01 (1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE
ABS	00000800 (2048.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.10	00:00:01.42
Command processing	105	00:00:00.68	00:00:06.06
Pass 1	178	00:00:04.18	00:00:14.96
Symbol table sort	0	00:00:00.31	00:00:00.83
Pass 2	79	00:00:01.15	00:00:03.74
Symbol table output	14	00:00:00.09	00:00:00.33
Psect synopsis output	2	00:00:00.02	00:00:00.34
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	409	00:00:06.54	00:00:27.94

The working set limit was 1200 pages.
18530 bytes (37 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 259 non-local and 11 local symbols.
406 source lines were read in Pass 1, producing 13 object records in Pass 2.
14 pages of virtual memory were used to define 13 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	5
_255DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	9

388 GETS were required to define 9 macros.

There were no errors, warnings or information messages.

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

NT0DAPRMS
LIS

NT0GET
LIS

NT0NWASET
LIS

NT0EXTEND
LIS

NT0DECODE
LIS

NT0ENCODE
LIS

NT0ERASE
LIS

NT0DISCON
LIS

NT0DISPLY
LIS

NT0MISC
LIS