NETACP

```
NN      NN  EEEEEEEEEE  TTTTTTTTTT  TTTTTTTTTT  RRRRRRR     NN      NN
NN      NN  EEEEEEEEEE  TTTTTTTTTT  TTTTTTTTTT  RRRRRRR     NN      NN
NN      NN  EE              TT          TT      RR     RR    NN      NN
NN      NN  EE              TT          TT      RR     RR    NN      NN
NNNN    NN  EE              TT          TT      RR     RR    NNNN    NN
NNNN    NN  EE              TT          TT      RR     RR    NNNN    NN
NN  NN  NN  EEEEEEE         TT          TT      RRRRRRR      NN  NN  NN
NN  NN  NN  EEEEEEE         TT          TT      RRRRRRR      NN  NN  NN
NN    NNNN  EE              TT          TT      RR  RR       NN    NNNN
NN    NNNN  EE              TT          TT      RR  RR       NN    NNNN
NN      NN  EE              TT          TT      RR    RR     NN      NN
NN      NN  EE              TT          TT      RR    RR     NN      NN
NN      NN  EEEEEEEEEE      TT          TT      RR      RR   NN      NN
NN      NN  EEEEEEEEEE      TT          TT      RR      RR   NN      NN
```

```
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
```

D 3

NETTRN
V04-000

E 3
- Main ACP loop and misc. subroutines   16-SEP-1984 01:29:26 VAX/VMS Macro V04-00   Page 1
                                           5-SEP-1984 02:21:47 [NETACP.SRC]NETTRN.MAR;1   (1)

```
0000      1          .TITLE  NETTRN - Main ACP loop and misc. subroutines
0000      2          .IDENT  'V04-000'
0000      3          .DEFAULT DISPLACEMENT,WORD
0000      4
0000      5
0000      6  ;*******************************************************************************
0000      7  ;*                                                                             *
0000      8  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                    *
0000      9  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                     *
0000     10  ;*  ALL RIGHTS RESERVED.                                                       *
0000     11  ;*                                                                             *
0000     12  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED      *
0000     13  ;*  ONLY IN  ACCORDANCE WITH THE  TERMS OF  SUCH  LICENSE  AND WITH THE        *
0000     14  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER      *
0000     15  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY      *
0000     16  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY      *
0000     17  ;*  TRANSFERRED.                                                               *
0000     18  ;*                                                                             *
0000     19  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE      *
0000     20  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT      *
0000     21  ;*  CORPORATION.                                                               *
0000     22  ;*                                                                             *
0000     23  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS      *
0000     24  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                    *
0000     25  ;*                                                                             *
0000     26  ;*                                                                             *
0000     27  ;*******************************************************************************
0000     28  ;
0000     29
0000     30
0000     31  ;++
0000     32  ; FACILITY: NETWORK ACP
0000     33  ;
0000     34  ; ABSTRACT:
0000     35  ;
0000     36  ;       This module processes the work, timer, and AQB queues.  It also
0000     37  ;       provides utility routines such as buffer management and timer/work
0000     38  ;       queue element routines.
0000     39  ;
0000     40  ; ENVIRONMENT:
0000     41  ;
0000     42  ;       MODE = KERNEL
0000     43  ;--
0000     44          .SBTTL  HISTORY
0000     45  ;
0000     46  ; AUTHOR:       SCOTT G. DAVIS, CREATION DATE: 20-APR-77
0000     47  ;
0000     48  ; MODIFIED BY:
0000     49  ;
0000     50  ;       V0007   RNG0007         Rod Gamache      14-Feb-1984
0000     51  ;               Remove reference to $XWBDEF. Add decrement of transport
0000     52  ;               entries removed from AQB queue.
0000     53  ;               Fix definitions of WQE.
0000     54  ;
0000     55  ;       V006    TMH0006         Tim Halvorsen  06-Apr-1983
0000     56  ;               Make pool automatically expand when allocation fails.
0000     57  ;
```

```
0000    58 ;        V005   TMH0005          Tim Halvorsen   05-Mar-1983
0000    59 ;               Dispatch all incoming IRPs with the PHYSIO flag set
0000    60 ;               to DLE module for processing.
0000    61 ;
0000    62 ;        V004   RNG0004          Rod Gamache     26-Jan-1983
0000    63 ;               Fix the NET$DEALLOCATE routine to not save
0000    64 ;               the caller's address in the deallocation block,
0000    65 ;               until it knows there is a block to deallocate!
0000    66 ;
0000    67 ;        V003   TMH0003          Tim Halvorsen   16-Sep-1982
0000    68 ;               Double size of journal record, and add an 8 byte
0000    69 ;               timestamp to the beginning.
0000    70 ;
0000    71 ;        V002   TMH0002          Tim Halvorsen   06-Jul-1982
0000    72 ;               Change where journalling gets the address of
0000    73 ;               the journal buffer (rather than using the local
0000    74 ;               LPD).
0000    75 ;
0000    76 ;        V001   TMH0001          Tim Halvorsen   13-Apr-1982
0000    77 ;               Change psect naming conventions.
0000    78 ;               Add WQE$FORK routine.
0000    79 ;               Pre-zero journalling buffer before filling it in.
0000    80 ;               Remove all explicit addressing mode specifiers
0000    81 ;               and make default word addressing mode.
0000    82 ;               Make all references to VMS exec general addressing.
0000    83 ;               Use SETBIT and CLRBIT macros where ever possible.
0000    84 ;---
```

```
                 0000      86              .SBTTL  DECLARATIONS
                 0000      87     ;
                 0000      88     ; INCLUDE FILES:
                 0000      89     ;
                 0000      90
                 0000      91              $AQBDEF
                 0000      92              $CCBDEF
                 0000      93              $CNFDEF
                 0000      94              $CNRDEF
                 0000      95              $CRBDEF
                 0000      96              $CXBDEF
                 0000      97              $DYNDEF
                 0000      98              $IRPDEF
                 0000      99              $IODEF
                 0000     100              $LTBDEF
                 0000     101              $NETSYMDEF
                 0000     102              $RCBDEF
                 0000     103              $UCBDEF
                 0000     104              $VECDEF
                 0000     105              $VCBDEF
                 0000     106              $WQEDEF
                 0000     107              $XMDEF
                 0000     108
                 0000     109     ;
                 0000     110     ; Add extensions to the WQE for timer entries
                 0000     111     ;
                 0000     112              $DEFINI WQE
                 0000     113
        00000024 0000     114              .= WQE$C_LENGTH
                 0024     115              ASSUME  QQE$C_LENGTH&3 EQ   0   ; Assume that we start longword aligned
                 0024     116 $DEF     WQE$Q_DUE_TIME  .BLKQ   1           ; Due time
                 002C     117 $DEF     WQE$C_TMR_LEN                       ; Length of timer WQE
                 002C     118  $EQU    WQE$C_EXT_LEN   <.-WQE$C_LENGTH> ; Length of the timer WQE Extension
                 002C     119
                 002C     120              $DEFEND WQE
                 0000     121
        00000021 0000     122 NET$C_DYN_WQE    == 33                      ; Dummy type code for WQE's
        0000000F 0000     123 MASK            = ^XF                       ; Buffer rounding mask
        00002800 0000     124 POOL_EXTEND     = 20*512                    ; Automatic pool expansion by 20 pages
                 0000     125
                 0000     126     ;
                 0000     127     ; OWN STORAGE:
                 0000     128     ;
        00000000 129              .PSECT  NET_PURE,NOWRT,NOEXE,LONG
                 0000     130
        00000000 0000     131 RANGE:  .LONG   0                           ; Range for working-set purge
        7FFFFFFF 0004     132         .LONG   <1@31>-1                    ; Do it all
                 0008     133
                 0008     134
        00000000 135              .PSECT  NET_IMPURE,WRT,NOEXE,LONG
                 0000     136
                 0000     137     ;
                 0000     138     ; Setup the timer and work queue listheads.
                 0000     139     ;
                 0000     140              ASSUME  WQE$L_FLINK     EQ      0
                 0000     141              ASSUME  WQE$L_BLINK     EQ      4
                 0000     142              ASSUME  WQE$W_SIZE      EQ      8
```

```
             0000      143              ASSUME    WQE$B_TYPE     EQ    10
             0000      144              ASSUME    WQE$B_SUB      EQ    11
             0000      145              ASSUME    WQE$L_ACTION   EQ    12
             0000      146              ASSUME    WQE$L_PM1      EQ    16
             0000      147              ASSUME    WQE$L_PM2      EQ    20
             0000      148
             0000      149  NET$GQ_WQE_WORK::                            ; ACP work queue listhead
00000000'    0000      150              .ADDRESS  .                     ; Listhead
00000000'    0004      151              .ADDRESS  .-4
             0000      152              .WORD     0                     ; Zero the size field to bugcheck
             000A      153                                              ; on attempted deallocation
        21   000A      154              .BYTE     NET$C_DYN_WQE         ; Structure type
        00   000B      155              .BYTE     WQE$C_SUB_BAS        ; Sub-type is "base"
00000024     000C      156              .BLKB     <NET$GQ_WQE_WORK+WQE$C_LENGTH>-. ; Make this a full WQE
             0024      157
             0024      158  NET$GQ_WQE_TIMR::                            ; ACP timer queue listhead
00000024'    0024      159              .ADDRESS  .                     ; Listhead
00000024'    0028      160              .ADDRESS  .-4
             0000      161              .WORD     0                     ; Zero the size field to bugcheck
             002E      162                                              ; on attempted deallocation
        21   002E      163              .BYTE     NET$C_DYN_WQE         ; Structure type
        00   002F      164              .BYTE     WQE$C_SUB_BAS        ; Sub-type is "base"
00000050     0030      165              .BLKB     <NET$GQ_WQE_TIMR+WQE$C_TMR_LEN>-. ; Make this a full WQE
             0050      166                                              ;    for Timer use.
             0050      167
00000000     0050      168  NET_TIMER:    .LONG   0                     ; Low bit used to signal timer AST
             0054      169
00000000     170                .PSECT    NET_CODE,NOWRT,EXE
```

NETTRN
V04-000
         I 3

- Main ACP loop and misc. subroutines   16-SEP-1984 01:29:26 VAX/VMS Macro V04-00   Page 5
Major NETACP work dispatching loop   5-SEP-1984 02:21:47 [NETACP.SRC]NETTRN.MAR;1   (3)

```
                    0000   172              .SBTTL  Major NETACP work dispatching loop
                    0000   173  ;++
                    0000   174  ; NET$DISPATCH - Purge working set and dispatch whatever work there is to do
                    0000   175  ;
                    0000   176  ; FUNCTIONAL DESCRIPTION:
                    0000   177  ;
                    0000   178  ; The work queue must be serviced before servicing the AQB queue.  This is
                    0000   179  ; because servicing a given AQB or WQE entry may result directly result in a
                    0000   180  ; WQE being queued but will not directly result in an AQB entry being queued.
                    0000   181  ;
                    0000   182  ; The WQE entries are outline above.  The AQB entries are either IRPs or
                    0000   183  ; NET buffers as follows:
                    0000   184  ;
                    0000   185  ;       The DLE IRPs must be dispatched if the IRP$V_PHYSIO flag is set
                    0000   186  ;
                    0000   187  ;       The IRP must be dispatched by its IRP$W_FUNC value:
                    0000   188  ;
                    0000   189  ;               IO$_ACPCONTROL  - network management function if IRP$L_SVAPTE
                    0000   190  ;                                 is non-zero; $CANCEL function otherwise.
                    0000   191  ;
                    0000   192  ;               IO$_ACCESS      - logical-link connect or direct-line access
                    0000   193  ;
                    0000   194  ;               IO$_DEACESS     - logical-link disconnect or direct-line deaccess
                    0000   195  ;
                    0000   196  ;               IO$_DELETE      - datalink has gone inactive
                    0000   197  ;
                    0000   198  ;
                    0000   199  ;       The NET buffer header format is that of WQE contains the following:
                    0000   200  ;
                    0000   201  ;               WQE$B_EVT       - One of the following:
                    0000   202  ;
                    0000   203  ;                               NETMSG$C_TR  - a Transport control message
                    0000   204  ;                               NETMSG$C_ILL - an illegal message
                    0000   205  ;                               NETMSG$C_UNK - an unknown message
                    0000   206  ;                               NETMSG$C_IRP - a datalink has gone down
                    0000   207  ;
                    0000   208  ;               WQE$W_PTH       - The LPD path i.d. of the datalink
                    0000   209  ;--
                    0000   210  NET$DISPATCH::                                  ; Major NETACP work dispatching loop
                    0000   211          $PURGWS_S RANGE                         ; Purge the working set
                    000B   212          ;
                    000B   213          ;   Drain the scratch buffer queue
                    000B   214          ;
   0000'CF   D4     000B   215  GO:     CLRL    NET$GL_FLAGS                    ; Clear internal flags
50 0000'DF   0F     000F   216          REMQUE  @NET$GQ_TMP_BUF,R0              ; Get then next buffer
      05     1D     0014   217          BVS     50$                            ; Br if queue is empty
    0110'    30     0016   218          BSBW    NET$DEALLOCATE                  ; Deallocate it
      F0     11     0019   219          BRB     GO                             ; Loop for next buffer
                    001B   220          ;
                    001B   221          ;   Process any work queue entries
                    001B   222          ;
03 0050'CF  00 E7   001B   223  50$:    BBCCI   #0,NET_TIMER,70$               ; If BS then timer AST occurred
      01A2   30     0021   224          BSBW    TIMER_EXP                      ; Service the timer
      022C   30     0024   225  70$:    BSBW    WQE$REMQUE                     ; Get and process next waiting WQE
   E1 50     E8     0027   226          BLBS    R0,GO                          ; If LBS keep going
                    002A   227          ;
                    002A   228          ;   Process any AQB entries
```

NETTRN
V04-000
J 3
- Main ACP loop and misc. subroutines   16-SEP-1984 01:29:26   VAX/VMS Macro V04-00   Page 6
Major NETACP work dispatching loop   5-SEP-1984 02:21:47   [NETACP.SRC]NETTRN.MAR;1   (3)

```
52   0000'CF   D0   002A   229                  ;
  53   00 B2   0F   002F   230                  MOVL    NET$GL_PTR_AQB,R2        ; Get address of queue head
         4C     1D   0033   231                  REMQUE  @AQB$L_ACPQFL(R2),R3    ; Try to get a packet
       7C'AF    9F   0035   232                  BVS     300$                    ; If VS queue is empty
                            233                  PUSHAB  B^200$                  ; Set up return for dequeuing
                0038   234                  $DISPATCH  IRP$B_TYPE(R3),TYPE=B,-
                0038   235                  <-      ; type           ; action
                0038   236                  <DYN$C_IRP      PROC_IRP>,- ; Process an IRP
                0038   237                  <DYN$C_NDB      NET$PROC_XWB>,- ; Process and XWB
                0038   238                  <DYN$C_NET      100$>,- ; Process a transport request
                0038   239                  <DYN$C_CXB      90$>,- ; Process a counted transport request
                0038   240                  >
                0063   241
                0063   242   80$:          BUG_CHECK  NETNOSTATE,FATAL          ; Bad request packet/bad AQB count
                0067   243
50   0000'CF   D0   0067   244   90$:     MOVL    NET$GL_PTR_VCB,R0       ; Get RCB address
     00A9 C0   97   006C   245            DECB    RCB$B_AQB_CNT(R0)       ; One less transport element on queue
         F1     19   0070   246            BLSS    80$                     ; Br if bad count
  0A A3  17    90   0072   247            MOVB    #DYN$C_NET,IRP$B_TYPE(R3) ; Set the real buffer type
     55   53   D0   0076   248   100$:    MOVL    R3,R5                   ; Copy buffer address for call
       FF84'   31   0079   249            BRW     NET$DLL_RCV             ; Process the message
                007C   250
       FF81'   30   007C   251   200$:    BSBW    NET$DEC_TRANS           ; Decrement the transaction count
         8A     11   007F   252            BRB     GO                      ; Loop
                0081   253                  ;
                0081   254                  ;   Go to sleep, my baaaby
                0081   255                  ;
                0081   256   300$:    $HIBER_S                            ; Hibernate
         81     11   0088   257            BRB     GO                      ; Loop
```

NETTRN   K 3
V04-000   - Main ACP loop and misc. subroutines   16-SEP-1984 01:29:26 VAX/VMS Macro V04-00   Page 7
                   Major NETACP work dispatching loop   5-SEP-1984 02:21:47 [NETACP.SRC]NETTRN.MAR;1   (4)

```
                    008A    259  PROC_IRP:                                      ; Process IRP
                    008A    260          ;
                    008A    261          ;    If the PHYSIO bit is set, then it is a DLE IRP, and must be
                    008A    262          ;    passed directly to the DLE module.
                    008A    263          ;
              08 E5 008A    264          BBCC     #IRP$V_PHYSIO,-               ; If PHYSIO flag is set,
        03 2A A3      008C  265                   IRP$W_STS(R3),5$
            FF6E' 31  008F  266          BRW      DLE$DISPATCH                  ; Give it to DLE module
                    0092    267          ;
                    0092    268          ;    It's a normal logical link IRP
                    0092    269          ;
        55  1C A3 D0 0092   270  5$:     MOVL     IRP$L_UCB(R3),R5             ; Get UCB address
    0000'CF  55   D0 0096   271          MOVL     R5,NET$GL_SAVE_UCB           ; Save it
    0000'CF  53   D0 009B   272          MOVL     R3,NET$GL_SAVE_IRP           ; Save the IRP address
              00' 7D 00A0   273          MOVQ     S^#SS$_NORMAL,-             ; Init IOSB image
        0000'CF       00A2  274                   NET$GQ_USR_STAT
              00 EF  00A5   275          EXTZV    #IRP$V_FCODE,-              ;
              06      00A7  276                   #IRP$S_FCODE,-             ;
        57  20 A3    00A8   277                   IRP$W_FUNC(R3),R7           ; Get function code
              25 10  00AB   278          BSBB     20$                         ; Dispatch
                    00AD    279          ;
                    00AD    280          ;    Finish IRP processing - either complete or requeue to driver.
                    00AD    281          ;    If the NET$GL_SAVE_IRP is already zero then the ACP has tucked
                    00AD    282          ;    away the IRP somewhere to avoid I/O completion.  If this is the
                    00AD    283          ;    case then either mount or the transaction counts must have been
                    00AD    284          ;    updated so that pool would not be lost due to a premature shutdown.
                    00AD    285          ;
        53  0000'CF D0 00AD 286          MOVL     NET$GL_SAVE_IRP,R3          ; Get IRP
              1D 13  00B2   287          BEQL     15$                         ; If EQL its gone
        55  0000'CF D0 00B4 288          MOVL     NET$GL_SAVE_UCB,R5          ; Get UCB
        0000'CF   7D 00B9   289          MOVQ     NET$GQ_USR_STAT,-          ;
              38 A3  00BD   290                   IRP$L_MEDIA(R3)             ; Jam back the i/o status
              05 E0  00BF   291          BBS      S^#NET$V_RQIRP,-           ; Br if packet is to be given
        06 0000'CF   00C1   292                   NET$GL_FLAGS,10$           ; back to the driver
    00000000'GF  17 00C5    293          JMP      G^COM$POST                  ; Else post it for completion
    00000000'GF  17 00CB    294  10$:    JMP      G^EXE$INSIOQ                ; Queue packet to driver
              05   00D1     295  15$:    RSB                                  ; Return to caller
                    00D2    296          ;
                    00D2    297          ;
                    00D2    298  20$:    $DISPATCH R7,<-
                    00D2    299
                    00D2    300                   <IOS_ACCESS,      30$>,-
                    00D2    301                   <IOS_ACPCONTROL, 40$>,-
                    00D2    302                   <IOS_DEACCESS,   50$>,-
                    00D2    303                   >
        0000'8F B0  00E4    304          MOVW     #SS$_ILLIOFUNC,-           ; Say "illegal I/O function"
        0000'CF     00E8    305                   NET$GQ_USR_STAT           ;
              1D 11  00EB   306          BRB      60$                        ; Return
                    00ED    307          ;
                    00ED    308          ;    ACCESS function - dispatch to connect processor
                    00ED    309          ;
        0000'CF  00 FB 00ED 310  30$:    CALLS    #0,NET$CONNECT             ; Do it
              16 11  00F2   311          BRB      60$                        ; Common exit
                    00F4    312          ;
                    00F4    313          ;    ACP Control
                    00F4    314          ;
              03 E1  00F4   315  40$:    BBC      #IRP$V_COMPLX,-
```

```
        05 2A A3       00F6  316                        IRP$W_STS(R3),45$      ; If BC then I/O rundown
           FF04' 30    00F9  317              BSBW      NET$CONTROL_QIO        ; Process control function
              0C 11    00FC  318              BRB       60$                   ; Complete I/O and get next IRP
     18 A3  01 CA      00FE  319  45$:        BICL      #1,IRP$L_WIND(R3)      ; Clear interlock bit in case an
                       0102  320                                              ; IO$_ACCESS or IO$_DEACCESS is pending
         FEFB' 30      0102  321              BSBW      NET$ACP_CANCEL        ; Do cancel-related work
                       0105  322  ;           BRB       50$                   ; Requeue packet to driver
                       0105  323              :
                       0105  324              ;    DEACCESS function
                       0105  325              :
                       0105  326  50$:        SETBIT    NET$V_RQIRP,-         ; Cause packet to be requeued
                       0105  327                        NET$GL_FLAGS          ; to driver
        05             01CA  328  60$:        RSB                             ; Done
```

NETTRN
V04-000

M 3
- Main ACP loop and misc. subroutines   16-SEP-1984 01:29:26   VAX/VMS Macro V04-00   Page   9
WQE$RESET_TIM - Cancel and reset timer   5-SEP-1984 02:21:47   [NETACP.SRC]NETTRN.MAR;1   (5)

```
                              010B   330              .SBTTL  WQE$RESET_TIM - Cancel and reset timer
                              010B   331      ;+
                              010B   332      ;  WQE$RESET_TIM - Cancel and reset timer
                              010B   333      ;
                              010B   334      ; FUNCTIONAL DESCRIPTION:
                              010B   335      ;
                              010B   336      ; The WQE timer and work queue are searched and all entries which match the
                              010B   337      ; WQE$B_EVT,WQE$B_QUAL and WQE$_REQIDT fields are deleted.  The timer is then
                              010B   338      ; reset as specified.
                              010B   339      ;
                              010B   340      ; INPUTS:           R3,R4       Quadword 100 nsec new delay
                              010B   341      ;                   R2          Action routine to call when the timer expires
                              010B   342      ;                   R1          WQE$B_EVT,WQE$B_QUAL,WQE$W_REQIDT (EVT in low byte)
                              010B   343      ;                   R0          Scratch
                              010B   344      ;
                              010B   345      ; OUTPUTS:          All registers are unchanged
                              010B   346      ;-
                              010B   347      WQE$RESET_TIM::                                 ; Cancel and reset timer
             3F    BB         010B   348              PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ;
             58    10         010D   349              BSBB    WQE$CANCEL_TIM          ; Cancel all matching entries
             03    10         010F   350              BSBB    WAIT                    ; Set new timer
             3F    BA         0111   351              POPR    #^M<R0,R1,R2,R3,R4,R5>  ;
                   05         0113   352              RSB
                              0114   353
                              0114   354      WAIT:   ASSUME  WQE$L_FLINK EQ 0        ; This assumption is made thru-out
             55    DD         0114   355              PUSHL   R5                      ; Save reg
                              0116   356              ;
                              0116   357              ;       Allocate and initialize a Work Queue Element
                              0116   358              ;
             06    BB         0116   359              PUSHR   #^M<R1,R2>              ; Save regs
          50  07    90        0118   360              MOVB    #WQE$C_SUB_TIM,R0       ; WQE subtype
          51  08    9A        011B   361              MOVZBL  #WQE$C_EXT_LEN,R1       ; Additional data bytes required
                   0151 30    011E   362              BSBW    WQE$ALLOCATE            ; Allocate the WQE
          55  52    D0        0121   363              MOVL    R2,R5                   ; Move WQE ptr, if any, to R5
             06    BA         0124   364              POPR    #^M<R1,R2>              ; Recover regs
          3A  50    E9        0126   365              BLBC    R0,30$                  ; Br on error
       0164'CF     9E         0129   366              MOVAB   TIMER_ACTION,-          ; Setup action routine address
          0C  A5              012D   367                      WQE$L_ACTION(R5)        ;
                              012F   368              ASSUME  WQE$L_PM2  EQ  4+WQE$L_PM1
       10  A5    51  7D       012F   369              MOVQ    R1,WQE$L_PM1(R5)        ; Setup action routine and parameter
                              0133   370              ;
                              0133   371              ;       Calculate expiration time and insert WQE in time ordered queue
                              0133   372              ;
       51  00000000'GF  7D    0133   373              MOVQ    G^EXE$GQ_SYSTIME,R1     ; Get current time
          53  51    C0        013A   374              ADDL    R1,R3                   ; Add low order delay
          54  52    D8        013D   375              ADWC    R2,R4                   ; Add high order with carry
       24  A5  53    7D       0140   376              MOVQ    R3,WQE$Q_DUE_TIME(R5)   ; Setup due time in WQE data
       50  0024'CF   9E       0144   377              MOVAB   NET$GQ_WQE_TIMR,R0      ; Get timer queue listhead ptr address
          50  60    D0        0149   378      10$:    MOVL    (R0),R0                 ; Advance to next entry
       09  0B  A0    E9       014C   379              BLBC    WQE$B_SUB(R0),20$       ; Br if this is the timer listhead
       51  24  A0    7D       0150   380              MOVQ    WQE$Q_DUE_TIME(R0),R1   ; Get entry's due time
          00C9  30            0154   381              BSBW    CMPTIM_32T              ; New WQE's time is in R3,R4
                              0157   382                                             ; Queued WQE's time is in R1,R2
             F0    1A         0157   383              BGTRU   10$                     ; If GTRU then R3 time may be later
       04  B0    65    0E     0159   384      20$:    INSQUE  (R5),@4(R0)             ; Insert current WQE before WQE with
                              015D   385                                             ; later expected time
             008E  30         015D   386              BSBW    SET_TIMER               ; Reset the timer
```

NETTRN
V04-000
                                    N 3
                 - Main ACP loop and misc. subroutines   16-SEP-1984 01:29:26 VAX/VMS Macro V04-00   Page 10
                 WQE$RESET_TIM - Cancel and reset timer   5-SEP-1984 02:21:47 [NETACP.SRC]NETTRN.MAR;1   (5)

```
    55 8ED0  0160  387          POPL    R5              ; Restore reg
       05    0163  388 30$:     RSB                     ; Done
             0164  389
             0164  390 TIMER_ACTION:                    ; Timer action routine
    62 16    0164  391          JSB     (R2)            ; Call action routine
       05    0166  392          RSB
```

NETTRN   B 4
V04-000   - Main ACP loop and misc. subroutines   16-SEP-1984 01:29:26   VAX/VMS Macro V04-00   Page 11
               WQE$CANCEL_TIM - Cancel work timer   5-SEP-1984 02:21:47   [NETACP.SRC]NETTRN.MAR;1   (6)

```
                        0167    394              .SBTTL   WQE$CANCEL_TIM - Cancel work timer
                        0167    395    ;+
                        0167    396    ; WQE$CANCEL_TIM - Cancel timer
                        0167    397    ;
                        0167    398    ; FUNCTIONAL DESCRIPTION:
                        0167    399    ;
                        0167    400    ; The WQE timer and work queue are searched and all entries which match the
                        0167    401    ; WQE$B_EVT, WQE$B_QUAL and WQE$_REQIDT fields are deleted.  WQE$B_EVT = 0
                        0167    402    ; matches all events.
                        0167    403    ;
                        0167    404    ; INPUTS:              R1       WQE$B_EVT,WQE$B_QUAL,WQE$W_REQIDT (EVT in low byte)
                        0167    405    ;                     R0       Scratch
                        0167    406    ; OUTPUTS:
                        0167    407    ;                     R0       Clobbered
                        0167    408    ;
                        0167    409    ;                     All other registers are unchanged
                        0167    410    ;
                        0167    411    ;                     ****  MUST BE CALLED AT IPL 0  ****
                        0167    412    ;-
                        0167    413    WQE$CANCEL_TIM::                                ; Cancel all matching timer entries
        50     00'  DB  0167    414              MFPR     S^#PR$_IPL,R0             ; Get current IPL
               50  95  016A    415              TSTB     R0                       ; Is it zero
               04  13  016C    416              BEQL     3$                       ; If EQL then okay
                        016E    417              BUG_CHECK NETNOSTATE,FATAL        ; Else race conditions could exist
        1C     BB  0172    418    3$:            PUSHR    #^M<R2,R3,R4>            ; Save regs
        53     D4  0174    419              CLRL     R3                       ; Nullify event mask
        51     95  0176    420              TSTB     R1                       ; Cancel all ?
        02     12  0178    421              BNEQ     5$                       ; If NEQ then no
        53     97  017A    422              DECB     R3                       ; Set all low order bits
   52  0004'CF  D0  017C    423    5$:            MOVL     NET$GQ_WQE_WORK+4,R2     ; Point to last item in the work queue
        0A     10  0181    424              BSBB     10$                      ; Remove all matching entries
   52  0028'CF  D0  0183    425              MOVL     NET$GQ_WQE_TIMR+4,R2     ; Get last item in the timer queue
        03     10  0188    426              BSBB     10$                      ; Remove all matching entries
        1C     BA  018A    427              POPR     #^M<R2,R3,R4>            ; Restore regs
        05     018C    428              RSB
                        018D    429
        52  62  D0  018D    430    10$:           MOVL     (R2),R2                 ; Chain down the list
        50  62  D0  0190    431    20$:           MOVL     (R2),R0                 ; Get next entry
     0B A0  91  0193    432              CMPB     WQE$B_SUB(R0),-          ; Is the listhead ?
        00     0196    433                       #WQE$C_SUB_BAS
        18  13  0197    434              BEQL     30$                      ; If EQL then yes, we're done
     0B A0  91  0199    435              CMPB     WQE$B_SUB(R0),-          ; Is a timer element?
        07     019C    436                       #WQE$C_SUB_TIM
        EE  12  019D    437              BNEQ     10$                      ; If NEQ then no, try next element
  54  10 A0  53  CB  019F    438              BICL3    R3,WQE$B_EVT(R0),R4     ; Get event longword
     54  51  D1  01A4    439              CMPL     R1,R4                   ; Does the event match ?
        E4  12  01A7    440              BNEQ     10$                      ; If not, loop
     50  60  0F  01A9    441              REMQUE   (R0),R0                 ; Remove the entry
        00E4  30  01AC    442              BSBW     WQE$DEALLOCATE          ; Deallocate it
        DF  11  01AF    443              BRB      20$                     ; Loop
        05     01B1    444    30$:           RSB
```

```
                                    01B2      446              .SBTTL  WQE$TIMER_AST - Work timer AST
                                    01B2      447     ;+
                                    01B2      448     ; WQE$TIMER_AST - Receive timer AST
                                    01B2      449     ;
                                    01B2      450     ;-
                                    01B2      451     WQE$TIMER_AST::                              ; Receive timer AST
                           0000     01B2      452              .WORD   0                           ; Save regs
        00 0050'CF    00    E6      01B4      453              BBSSI   #0,NET_TIMER,10$            ; Flag fact that the AST occurred
                              04    01BA      454     10$:     $WAKE_S                             ; Wake the ACP
                                    01C5      455              RET
                                    01C6      456
                                    01C6      457     TIMER_EXP:
                    3F    BB        01C6      458              PUSHR   #^M<R0,R1,R2,R3,R4,R5>      ; Save regs
                                    01C8      459
                                    01C8      460              ASSUME  WQE$L_FLINK EQ 0
                                    01C8      461
        55  0024'CF    9E           01C8      462              MOVAB   NET$GQ_WQE_TIMR,R5          ; Get address of TIME queue listhead
            51  24 A5  7D           01CD      463              MOVQ    WQE$Q_DUE_TIME(R5),R1       ; Get due time
                24 A5  7C           01D1      464              CLRQ    WQE$Q_DUE_TIME(R5)          ; Zero due time indicator to indicate
                                    01D4      465                                                  ; that no timer is active
        50      65    D0           01D4      466     5$:      MOVL    (R5),R0                     ; Advance to next element
            53  24 A0  7D           01D7      467              MOVQ    WQE$Q_DUE_TIME(R0),R3       ; Get its due time
                    0B  13           01DB      468              BEQL    10$                         ; If EQL then listhead
                    41  10           01DD      469              BSBB    CMPTIM_321                  ; Compare due time to current time
                    07  1A           01DF      470              BGTRU   10$                         ; If GTRU then it must wait
        50      60    0F           01E1      471              REMQUE  (R0),R0                     ; Else deque the WQE
                43      10           01E4      472              BSBB    WQE$INSQUE                  ; and insert it on work queue
                EC      11           01E6      473              BRB     5$                          ; Loop for as many WQEs as possible
                0003    30           01E8      474     10$:     BSBW    SET_TIMER                   ; Reset the timer
                3F      BA           01EB      475              POPR    #^M<R0,R1,R2,R3,R4,R5>      ; Restore regs
                        05           01ED      476              RSB
                                    01EE      477
                                    01EE      478
                                    01EE      479     SET_TIMER:                                   ; Conditionally reset timer
        50  0024'CF    9E           01EE      480              MOVAB   NET$GQ_WQE_TIMR,R0          ; Get timer queue listhead ptr
            53  60    D0           01F3      481              MOVL    (R0),R3                     ; Get first timer entry -- it may will
                                    01F6      482                                                  ; be the listhead if the queue is empty
        53  24 A3    9E           01F6      483              MOVAB   WQE$Q_DUE_TIME(R3),R3       ; Get its expiration time address
            24 A0  63  7D           01FA      484              MOVQ    (R3),WQE$Q_DUE_TIME(R0)     ; Setup new expiration time
                1C      13           01FE      485              BEQL    20$                         ; If EQL then queue is empty
                0200    486              $CANTIM_S  ACMODE = #0,-                ; Cancel timer - kernel mode
                0200    487                         REQIDT = #0                  ; Cancel all timers (there's only one)
                0209    488     10$:     $SETIMR_S  DAYTIM = (R3),-              ; Expiration time
                0209    489                         REQIDT = #0,-                ; Timer i.d.
                0209    490                         ASTADR = WQE$TIMER_AST       ; AST routine address
        ED 50    E9    0219    491              BLBC    R0,10$                      ;!Keep trying      *** FIX THIS ***
        50  01    90    021C    492     20$:     MOVB    #1,R0                       ; Set low bit to indicate success
                05    021F    493              RSB                                  ; Done
                        0220    494
                        0220    495     CMPTIM_321:                                  ; Compare R3,R4 to R1,R2 quadwords
        52  54    D1    0220    496              CMPL    R4,R2                       ; Compare high order expected time
            03    12    0223    497              BNEQ    10$                         ; Condition codes are valid if NEQ
        51  53    D1    0225    498              CMPL    R3,R1                       ; Compare low order expected time
                05    0228    499     10$:     RSB                                  ;
```

NETTRN   D 4
V04-000   - Main ACP loop and misc. subroutines   16-SEP-1984 01:29:26   VAX/VMS Macro V04-00   Page 13
                 WQE$INSQUE - Insert WQE into work queue   5-SEP-1984 02:21:47   [NETACP.SRC]NETTRN.MAR;1   (8)

```
                        0229      501              .SBTTL  WQE$INSQUE - Insert WQE into work queue
                        0229      502     ;+
                        0229      503     ; WQE$INSQUE - Insert WQE into work queue
                        0229      504     ;
                        0229      505     ; INPUTS:          RO - address of WQE to be inserted
                        0229      506     ;
                        0229      507     ;-
                        0229      508     WQE$INSQUE::                                ; Insert WQE into work queue
         0004'DF  60  0E 0229      509              INSQUE  (RO),@NET$GQ_WQE_WORK+4  ; Insert the element
                  22  12 022E      510              BNEQ    20$                      ; If NEQ then not first
                        0230      511              $DISPATCH TYPE=B,WQE$B_SUB(RO),-  ; Dispatch by subtype
                        0230      512              <-
                        0230      513                      <WQE$C_SUB_AST, 10$>,-    ; Miscellaneous AST
                        0230      514                      <WQE$C_SUB_TIM, 10$>,-    ; Timer AST
                        0230      515                      <WQE$C_SUB_MBX, 10$>,-    ; Mailbox read AST
                        0230      516                      <WQE$C_SUB_ACP, 20$>,-    ; Internal non-AST routine
                        0230      517              >
                        0243      518              BUG_CHECK NETNOSTATE,FATAL        ; Invalid subtype
                        0247      519
                        0247      520     10$:     $WAKE_S                           ; Wake-up the ACP
                  05    0252      521     20$:     RSB
```

NETTRN   E 4
V04-000   - Main ACP loop and misc. subroutines   16-SEP-1984 01:29:26 VAX/VMS Macro V04-00   Page 14
                      WQE$REMQUE - Dispatch next work element   5-SEP-1984 02:21:47 [NETACP.SRC]NETTRN.MAR;1   (9)

```
                        0253   523              .SBTTL  WQE$REMQUE - Dispatch next work element
                        0253   524    ;+
                        0253   525    ; WQE$REMQUE  - Dispatch next work queue entry
                        0253   526    ;
                        0253   527    ; DISPATCH WITH:   R5    WQE address
                        0253   528    ;                  R1    1st WQE parameter
                        0253   529    ;                  R2    2nd WQE parameter
                        0253   530    ;
                        0253   531    ;         All registers may be destroyed before returning
                        0253   532    ;-
                        0253   533    WQE$REMQUE::                             ; Remove entry form work queue
                   50   D4 0253   534              CLRL    R0                  ; In case nothing in queue
         55   0000'DF   0F 0255   535              REMQUE  @NET$GQ_WQE_WORK,R5 ; Get next work queue entry
                   15   1D 025A   536              BVS     20$                 ; If VS then none
                        025C   537              ASSUME  WQE$L_PM2  EQ  4+WQE$L_PM1
         51   10 A5     7D 025C   538              MOVQ    WQE$L_PM1(R5),R1    ; Get the parameters
                   07   91 0260   539              CMPB    #WQE$C_SUB_TIM,-    ; Queued by timer?
              0B A5        0262   540                      WQE$B_SUB(R5)       ;
                   05   12 0264   541              BNEQ    10$                 ; If NEQ no
                        0266   542              SETBIT  NET$V_TIMER,NET$GL_FLAGS ; Set flag to inform action routine
            0C B5   16 026B   543    10$:       JSB     @WQE$C_ACTION(R5)   ; Dispatch to action routine
         50   01   D0 026E   544              MOVL    #1,R0               ; Indicate should try again
                   05 0271   545    20$:       RSB
```

NETTRN
V04-000

F 4
- Main ACP loop and misc. subroutines   16-SEP-1984 01:29:26   VAX/VMS Macro V04-00   Page 15
WQE$ALLOCATE - Allocate a work element   5-SEP-1984 02:21:47   [NETACP.SRC]NETTRN.MAR;1   (10)

```
                    0272    547              .SBTTL   WQE$ALLOCATE - Allocate a work element
                    0272    548    ;+
                    0272    549    ; WQE$ALLOCATE  - Allocate a work queue element
                    0272    550    ;
                    0272    551    ; FUNCTIONAL DESCRIPTION:
                    0272    552    ;
                    0272    553    ; Allocate and initialize a work queue element.
                    0272    554    ;
                    0272    555    ; INPUTS:          R2       Scratch
                    0272    556    ;                  R1       Bytes in data area at end of block
                    0272    557    ;                  R0       WQE subtype code
                    0272    558    ;
                    0272    559    ; OUTPUTS:         R2       Address of block
                    0272    560    ;                  R1       Garbage
                    0272    561    ;                  R0       Status
                    0272    562    ;-
                    0272    563    WQE$ALLOCATE::                              ; Allocate a work queue element
              50 DD 0272    564              PUSHL    R0                      ; Save subtype
        51 24 C0 0274    565              ADDL     #WQE$C_LENGTH,R1        ; Get total size
      FE24' 30 0277    566              BSBW     NET$ALLOCATE            ; Allocate the block
         51 8ED0 027A    567              POPL     R1                      ; Recover the subtype
      0F 50 E9 027D    568              BLBC     R0,10$                  ; Br on error
         21 90 0280    569              MOVB     #NET$C_DYN_WQE,-        ;
         0A A2         0282    570                       WQE$B_TYPE(R2)         ; Setup the block type
   0B A2 51 90 0284    571              MOVB     R1,WQE$B_SUB(R2)        ; Setup the subtype
      14 A2 D4 0288    572              CLRL     WQE$L_PM2(R2)           ; Initialize some fields
      20 A2 B4 028B    573              CLRW     WQE$W_ADJ_INX(R2)      ; DLLTRN wants this to be initially 0
            05 028E    574              RSB
                    028F    575
                    028F    576    10$:     BUG_CHECK NETNOBUF,FATAL        ; No WQE available
```

NETTRN
V04-000
             G 4
- Main ACP loop and misc. subroutines   16-SEP-1984 01:29:26 VAX/VMS Macro V04-00   Page 16
WQE$DEALLOCATE - Deallocate a work eleme 5-SEP-1984 02:21:47 [NETACP.SRC]NETTRN.MAR;1   (11)

```
                          0293    578              .SBTTL  WQE$DEALLOCATE - Deallocate a work element
                          0293    579    ;+
                          0293    580    ; WQE$DEALLOCATE - Deallocate work queue element
                          0293    581    ;
                          0293    582    ; FUNCIONAL DESCRIPTION:
                          0293    583    ;
                          0293    584    ; Deallocate work queue element.  This routine calls NET$DEALLOCATE to
                          0293    585    ; deallocate the block is and is therefore currently unnecessary.  It is
                          0293    586    ; used as a possible hook for the furture when it may be used to recycle
                          0293    587    ; a WQE for a waiting caller to WQE$ALLOCATE.
                          0293    588    ;
                          0293    589    ; INPUTS:         RO        Address of block to be deallocated
                          0293    590    ;
                          0293    591    ; OUTPUTS:        RO        Garbage.
                          0293    592    ;
                          0293    593    ;                 All other registers are preserved.
                          0293    594    ;
                          0293    595    ;-
                          0293    596    WQE$DEALLOCATE::                                 ; Deallocate a WQE
        21       91       0293    597              CMPB    #NET$C_DYN_WQE,-            ; Is this really a WQE ?
     OA AO                0295    598                      WQE$B_TYPE(RO)
        06       13       0297    599              BEQL    5$                         ; If so, deallocate it
        17       91       0299    600              CMPB    #DYN$C_NET,-               ; This type code comes from NETDRIVER
     OA AO                029B    601                      WQE$B_TYPE(RO)             ; evnets
        04       12       029D    602              BNEQ    10$                        ; If NEQ then bug
     FE87'     30         029F    603    5$:        BSBW    NET$DEALLOCATE             ; Deallocate the block
                 05       02A2    604              RSB                                ; Return
                          02A3    605
                          02A3    606    10$:       BUG_CHECK NETNOSTATE,FATAL         ; Invalid WQE
```

NETTRN   H 4
V04-000   - Main ACP loop and misc. subroutines   16-SEP-1984 01:29:26 VAX/VMS Macro V04-00   Page 17
                    WQE$FORK - Switch to work queue level   5-SEP-1984 02:21:47 [NETACP.SRC]NETTRN.MAR;1   (12)

```
                        02A7    608            .SBTTL  WQE$FORK - Switch to work queue level
                        02A7    609     ;+
                        02A7    610     ; WQE$FORK - Switch to work queue level
                        02A7    611     ;
                        02A7    612     ; This routine is called to cause a code sequence to be executed
                        02A7    613     ; at "work level", which is a serial queue of tasks which are executed
                        02A7    614     ; at the ACP main dispatch routine.  This can be used to defer execution
                        02A7    615     ; of a code sequence to serialize access to a resource or eliminate stack
                        02A7    616     ; overflow due to excessive call frames.
                        02A7    617     ;
                        02A7    618     ; Inputs:
                        02A7    619     ;
                        02A7    620     ;       4(SP) = Address of caller's caller
                        02A7    621     ;       (SP) = Address of routine to execute
                        02A7    622     ;       R1/R2 = Arguments passed to routine
                        02A7    623     ;
                        02A7    624     ;       Only R1 and R2 are "passed" to the routine.  All other registers
                        02A7    625     ;       will not be available at the time the routine executes.  If more
                        02A7    626     ;       context needs to be passed, a longer WQE must be allocated to
                        02A7    627     ;       handle such needs.
                        02A7    628     ;
                        02A7    629     ; Outputs:
                        02A7    630     ;
                        02A7    631     ;       R0 = success
                        02A7    632     ;
                        02A7    633     ;       The WQE is queued and control is returned to the
                        02A7    634     ;       caller's caller.
                        02A7    635     ;-
                        02A7    636
                        02A7    637     WQE$FORK::
               OE    BB  02A7    638            PUSHR   #^M<R1,R2,R3>              ; Save regs
               51    D4  02A9    639            CLRL    R1                        ; Indicate no extra space needed
         50    01    DO  02AB    640            MOVL    #WQE$C_SUB_ACP,R0         ; Set WQE subtype
            FFC1    30  02AE    641            BSBW    WQE$ALLOCATE              ; Allocate a WQE
         50    52    DO  02B1    642            MOVL    R2,R0                     ; Transfer address of WQE
      10 AO    6E    7D  02B4    643            MOVQ    (SP),WQE$L_PM1(R0)        ; Store routine arguments
   18 AO  OC AE    DO  02B8    644            MOVL    12(SP),WQE$L_EVL_PKT(R0)  ; Set address of user's routine
   OC AO  CE'AF    9E  02BD    645            MOVAB   B^50$,WQE$L_ACTION(R0)    ; Set action routine address
            FF64    30  02C2    646            BSBW    WQE$INSQUE               ; Queue the work
               OE    BA  02C5    647            POPR    #^M<R1,R2,R3>             ; Restore regs
         50    01    DO  02C7    648            MOVL    #1,R0                     ; Set successful
         5E    04    CO  02CA    649            ADDL    #4,SP                     ; Return to caller's caller
                     05  02CD    650            RSB
                        02CE    651
                        02CE    652     ;
                        02CE    653     ; Come here when the work element is triggered.  R1/R2 already setup.
                        02CE    654     ;
                        02CE    655
               55    DD  02CE    656     50$:    PUSHL   R5                       ; Save WQE address
         18 B5    16  02D0    657            JSB     @WQE$L_EVL_PKT(R5)       ; Call user's routine at work level
                        02D3    658                                             ; All registers may be clobbered
         50 8EDO  02D3    659            POPL    R0                       ; Restore WQE address
            FFBA    30  02D6    660            BSBW    WQE$DEALLOCATE           ; Deallocate the WQE
                     05  02D9    661            RSB
```

NETTRN
V04-000
```
                                    I  4
              - Main ACP loop and misc. subroutines    16-SEP-1984 01:29:26  VAX/VMS Macro V04-00    Page 18
              NET$GETUTLBUF - Get use of utility buffe  5-SEP-1984 02:21:47  [NETACP.SRC]NETTRN.MAR;1    (13)
```

```
                02DA   663              .SBTTL  NET$GETUTLBUF - Get use of utility buffer
                02DA   664         ;+
                02DA   665         ; NET$GETUTLBUF - Acquire use of utility buffer (co-routine)
                02DA   666         ;
                02DA   667         ; FUNCTIONAL DESCRIPTION:
                02DA   668         ;
                02DA   669         ; Authorize the use of the utility buffer.  This is coded as a co-routine
                02DA   670         ; so that the utility buffer may be automatically released when the requesting
                02DA   671         ; routine exits.
                02DA   672         ;
                02DA   673         ; CALLED VIA:
                02DA   674         ;
                02DA   675         ;       BSB  NET$GETUTLBUF  or  JSB NET$GETUTLBUF
                02DA   676         ;
                02DA   677         ;-
                02DA   678  NET$GETUTLBUF::                              ; Acquire use of utility buffer
           06  E2  02DA   679              BBSS    S^#NET$V_UTLBUF,-        ; Obtain buffer if possible
 09 0000'CF       02DC   680                      NET$GL_FLAGS,10$
           9E  16  02E0   681              JSB     @(SP)+                  ; Call our caller
                  02E2   682              CLRBIT  NET$V_UTLBUF,NET$GL_FLAGS ; Free the utility buffer
           05      02E8   683              RSB                             ; Return to caller's original
                  02E9   684                                              ; caller.
                  02E9   685  10$:     BUG_CHECK NETNOSTATE,FATAL          ; UTLBUF is already in use
```

NETTRN   J 4
V04-000   - Main ACP loop and misc. subroutines   16-SEP-1984 01:29:26 VAX/VMS Macro V04-00   Page 19
                 NET$BIN2ASC - Convert binary to ASCII   5-SEP-1984 02:21:47 [NETACP.SRC]NETTRN.MAR;1   (14)

```
                      02ED    687              .SBTTL   NET$BIN2ASC - Convert binary to ASCII
                      02ED    688       ;++
                      02ED    689       ;
                      02ED    690       ; NET$BIN2ASC    - Convert binary to ASCII string
                      02ED    691       ;
                      02ED    692       ; FUNCTIONAL DESCRIPTION
                      02ED    693       ;
                      02ED    694       ; A binary number is converted to its ASCII representation.  The most
                      02ED    695       ; significant character is stored in the low order byte of the
                      02ED    696       ; destination string.
                      02ED    697       ;
                      02ED    698       ; INPUTS:    R0   Binary value to be converted
                      02ED    699       ;            R3   Pointer to byte to receive most significant ASCII character
                      02ED    700       ;
                      02ED    701       ; OUTPUTS:   R0   Garbage
                      02ED    702       ;            R3   Pointer to first byte passed the least significant ASCII
                      02ED    703       ;                 character
                      02ED    704       ;--
                      02ED    705       NET$BIN2ASC::                                        ; Convert binary to ASCII
              06  BB  02ED    706              PUSHR    #^M<R1,R2>                           ; Save regs
              51  7C  02EF    707              CLRQ     R1                                   ; Init high order dividend,
                      02F1    708                                                            ; character count
                      02F1    709              ;
                      02F1    710              ; Note that this algorithm yields an ASCII "0" if R0=0
                      02F1    711              ;
              52  D6  02F1    712       10$:    INCL     R2                                   ; Account for char to be moved
7E  50  50  0A  7B  02F3    713              EDIV     #10,R0,R0,-(SP)                      ; Get decimal digit
          F7  12  02F8    714              BNEQ     10$                                  ; Done if EQL
      50  8E  30  C1  02FA    715       20$:    ADDL3    #^A'0',(SP)+,R0                      ; Convert digit to ascii
          83  50  90  02FE    716              MOVB     R0,(R3)+                             ; Move it to dest buffer
      F6  52  F5  0301    717              SOBGTR   R2,20$                               ;  most significant character first
          06  BA  0304    718              POPR     #^M<R1,R2>                           ; Restore regs
              05  0306    719              RSB
```

```
                          0307    721              .SBTTL  NET$JNX_CO - Journalling routine
                          0307    722          ;+
                          0307    723          ; NET$JNX_CO    - Journalling co-routine
                          0307    724          ;
                          0307    725          ; This is common co-routine to facilitate "journalling".  Journalling is a
                          0307    726          ; debugging aid and is not part of the DECnet product.
                          0307    727          ;
                          0307    728          ;-
              00000040    0307    729                  BUF_SIZ = 64                    ; Must divide evenly into a block
                          0307    730
                          0307    731          NET$JNX_CO::                            ; Journalling co-routine
        0071'CF    16     0307    732                  JSB     FIND_JNX                ; Find the journal bufer
              01    12     030B    733                  BNEQ    10$                     ; If NEQ the journalling is active
                    05     030D    734                  RSB                             ; Return with low bit clear in R0
                          030E    735
        0000'CF    17     030E    736          10$:    JMP     JNX_CALLBACK            ; Setup record and call back user
                          0312    737
                          0312    738                  .SAVE_PSECT
              00000000    739                          .PSECT  NET_LOCK_CODE,NOWRT,GBL
                          0000    740
                          0000    741          JNX_CALLBACK:
        50  8ED0    0000    742                  POPL    R0                      ; Get caller's address, cleanup stack
        3E    BB     0003    743                  PUSHR   #^M<R1,R2,R3,R4,R5>     ; Save regs
                      0005    744                  ASSUME  BUF_SIZ EQ 64
        7E    7C     0005    745                  CLRQ    -(SP)                   ; Create/zero nonpaged scratch buffer
        7E    7C     0007    746                  CLRQ    -(SP)
        7E    7C     0009    747                  CLRQ    -(SP)
        7E    7C     000B    748                  CLRQ    -(SP)
        7E    7C     000D    749                  CLRQ    -(SP)
        7E    7C     000F    750                  CLRQ    -(SP)
        7E    7C     0011    751                  CLRQ    -(SP)
        51    5E  D0  0013   752                  MOVL    SP,R1                   ; Point to data buffer
7E  00000000'GF  7D  0016   753                  MOVQ    G^EXE$GQ_SYSTIME,-(SP)  ; Store timestamp in first 8 bytes
        50    DD     001D    754                  PUSHL   R0                      ; Push caller's address
        50    01  D0  001F   755                  MOVL    #1,R0                   ; Set success
        9E    16     0022    756                  JSB     a(SP)+                  ; Call back for data
    52  04  AE  9E  0024    757                  MOVAB   4(SP),R2                ; Get 'buffer' address
        51    52  C2  0028   758                  SUBL    R2,R1                   ; Get number of bytes moved
00000040 8F  51  D1  002B   759                  CMPL    R1,#BUF_SIZ             ; Too much data moved?
        04    1B     0032    760                  BLEQU   20$                     ; If LEQU then okay
                      0034    761
                      0034    762                  BUG_CHECK NETNOSTATE,FATAL      ; Too much journalling data
                      0038    763
                      0038    764          20$:    DSBINT  #NET$C_IPL              ; Synchronize with NETDRIVER and I/O
                      003E    765                                                  ; data base changes.
        31    10     003E    766                  BSBB    FIND_JNX                ; JNX buffer still around?
        1E    13     0040    767                  BEQL    30$                     ; If not, we're done
        53    60  D0  0042   768                  MOVL    (R0),R3                 ; Get pointer to free area
06  A0  0040 8F  B1  0045   769                  CMPW    #BUF_SIZ,6(R0)          ; Enough bytes left
        13    1E     004B    770                  BGEQU   30$                     ; If LEQU then no
06  A0  0040 8F  A2  004D   771                  SUBW    #BUF_SIZ,6(R0)          ; Take the space
60  00000040 8F  C0  0053   772                  ADDL    #BUF_SIZ,(R0)           ; Update pointer
63  62  0040 8F  28  005A   773                  MOVC3   #BUF_SIZ,(R2),(R3)      ; Enter JNX record
                      0060    774          30$:    ENBINT                          ; Restore IPL, fix stack
        50  8ED0    0063    775                  POPL    R0                      ; Get return address
5E  00000040 8F  C0  0066   776                  ADDL    #BUF_SIZ,SP             ; Fix stack
        3E    BA     006D    777                  POPR    #^M<R1,R2,R3,R4,R5>     ; Restore regs
```

```
         60   17  006F    778             JMP      (R0)                      ; Return
                  0071    779
                  0071    780  FIND_JNX:
50   0000'CF  D0  0071    781             MOVL     NET$GL_PTR_VCB,R0         ; Get RCB
         04   13  0076    782             BEQL     10$                       ; If EQL then none
50   18  A0   D0  0078    783             MOVL     RCB$L_PTR_JNX(R0),R0      ; Get journal buffer (0 if none)
         05       007C    784  10$:       RSB
                  007D    785
              00000312    786             .RESTORE
```

NETTRN
V04-000
M 4
- Main ACP loop and misc. subroutines   16-SEP-1984 01:29:26   VAX/VMS Macro V04-00   Page 22
Pool allocation routines   5-SEP-1984 02:21:47   [NETACP.SRC]NETTRN.MAR;1   (16)

```
                      0312    788              .SBTTL  Pool allocation routines
                      0312    789    ;++
                      0312    790    ; NET$ALONPGD_Z - Allocate and zero a block of nonpaged system pool
                      0312    791    ; NET$ALONPAGED - Allocate a block of nonpaged system pool
                      0312    792    ; NET$ALLOCATE  - Allocate a process space memory block
                      0312    793    ; NET$DEALLOCATE- Deallocate memory to either process or non-paged pool
                      0312    794    ;
                      0312    795    ; FUNCTIONAL DESCRIPTION:
                      0312    796    ;
                      0312    797    ; A block is allocated and the block header is initialized as follows:
                      0312    798    ;
                      0312    799    ;          +----------------------------------+
                      0312    800    ;          !                0                 !
                      0312    801    ;          +----------------------------------+
                      0312    802    ;          !                0                 !
                      0312    803    ;          +----------------------------------+
                      0312    804    ;          !    0   !DYN$C_NET!     size       !
                      0312    805    ;          +--------+---------+---------------+
                      0312    806    ;          !                                  !
                      0312    807    ;          :                                  :
                      0312    808    ;          :                                  :
                      0312    809    ;
                      0312    810    ;
                      0312    811    ; INPUTS:        R1 - Block size
                      0312    812    ;--
                      0312    813              .SAVE_PSECT
                  0000007D    814              .PSECT  NET_LOCK_CODE,NOWRT,GBL
                      007D    815
                      007D    816    NET$ALONPGD_Z::                            ; Allocate/zero non-paged pool
                13    10    007D    817              BSBB    NET$ALONPAGED      ; Get the block
          0F    50    E9    007F    818              BLBC    R0,10$             ; Br on error
          51    0C    A2    0082    819              SUBW    #12,R1             ; Get number of bytes to zero
          3F    BB    0085    820              PUSHR   #^M<R0,R1,R2,R3,R4,R5>   ; Save regs
0C A2  51  00  0C A2  00    2C    0087    821              MOVC5   #0,12(R2),#0,R1,12(R2)   ; Zero the block
          3F    BA    008F    822              POPR    #^M<R0,R1,R2,R3,R4,R5>   ; Restore regs
          05    0091    823    10$:         RSB                                ; Done
                0092    824
                0092    825    NET$ALONPAGED::                            ; Allocate non-paged memory
          3A    BB    0092    826              PUSHR   #^M<R1,R3,R4,R5>   ; Save regs
  00000000'GF    16    0094    827              JSB     G^EXE$ALONONPAGED  ; Get the block
          3A    BA    009A    828              POPR    #^M<R1,R3,R4,R5>   ; Restore regs
          7A    11    009C    829              BRB     INIT               ; 
                009E    830
                009E    831
                009E    832    NET$ALLOCATE::                            ; Allocate virtual memory
      51  0F  C0    009E    833              ADDL    #MASK,R1           ; Round size to next boundary
      51  0F  CA    00A1    834              BICL    #MASK,R1           ; 
          04    12    00A4    835              BNEQ    5$                 ; If EQL bad allocation request
                00A6    836
                00A6    837              BUG_CHECK BADALORQSZ,FATAL       ; Bad allocation request size
                00AA    838
    7E  53    7D    00AA    839    5$:          MOVQ    R3,-(SP)           ; Save regs
  53  0000'CF    9E    00AD    840              MOVAB   NET$GL_MY_POOL,R3  ; Point to memory listhead
                00B2    841              DSBINT  (R3)+              ; Synchronize pool
  00000000'GF    16    00B8    842              JSB     G^EXE$ALLOCATE     ; Allocate block
                00BE    843              ENBINT
                00C1    844    ;
```

NETTRN   N 4
V04-000   - Main ACP loop and misc. subroutines   16-SEP-1984 01:29:26   VAX/VMS Macro V04-00   Page 23
                       Pool allocation routines   5-SEP-1984 02:21:47   [NETACP.SRC]NETTRN.MAR;1   (16)

```
                        00C1   845          ;          If there was insufficient memory in the virtual pool, then
                        00C1   846          ;          allocate more virtual address space and add it to the pool.
                        00C1   847          ;
        5E   08   C2    00C1   848                 SUBL    #8,SP                    ; Allocate quadword on stack
     4B 50   E8    00C4 849                 BLBS    R0,20$                   ; If insufficient memory,
        50   5E   D0    00C7   850                 MOVL    SP,R0                    ; Set address of scratch quadword
        51   DD    00CA   851                 PUSHL   R1                       ; Save length of requested block
                        00CC   852                 $EXPREG_S -
                        00CC   853                   PAGCNT = #<POOL_EXTEND+511>/512,- ; no. of pages
                        00CC   854                   RETADR = (R0)            ; Address for pool address
        51 8ED0  00DB   855                 POPL    R1                       ; Restore length of requested block
     31 50   E9    00DE 856                 BLBC    R0,20$                   ; If error, then nothing more to give
        50   6E   D0    00E1   857                 MOVL    (SP),R0                  ; Get address of new storage
        60   D4    00E4   858                 CLRL    (R0)                     ; Zero link pointer in free block
 04 A0  00002800 8F DO 00E6  859                 MOVL    #POOL_EXTEND,4(R0)       ; Set length of free block
     53 0000'CF  9E   00EE 860                 MOVAB   NET$GL_MY_POOL,R3        ; Point to memory listhead
                        00F3   861                 DSBINT  (R3)+                    ; Synchronize pool
        52   53   D0    00F9   862  15$:       MOVL    R3,R2                    ; Save address of previous free block
        53   63   D0    00FC   863                 MOVL    (R3),R3                  ; Get address of next free block
        F8   12    00FF   864                 BNEQ    15$                      ; Loop until end of free list found
        62   50   D0    0101   865                 MOVL    R0,(R2)                  ; Append new storage to end of list
     53 0004'CF  9E   0104 866                 MOVAB   NET$GL_MY_POOL+4,R3      ; Point to memory listhead
  00000000'GF  16   0109 867                 JSB     G^EXE$ALLOCATE           ; Try to allocate block once more
                        010F   868                 ENBINT
        5E   08   C0    0112   869  20$:       ADDL    #8,SP                    ; Pop scratch quadword off stack
        53   8E   7D    0115   870                 MOVQ    (SP)+,R3                 ; Restore regs
                        0118   871          ;
                        0118   872          ;        Initialize the storage
                        0118   873          ;
        0B 50   E9    0118   874  INIT:      BLBC    R0,10$                   ; If LBC error
        62   7C    011B   875                 CLRQ    (R2)                     ; Clear the first 2 header longwords
     08 A2   51   B0   011D   876                 MOVW    R1,IRP$W_SIZE(R2)        ; Enter size
     0A A2   17   9B   0121   877                 MOVZBW  #DYN$C_NET,IRP$B_TYPE(R2) ; Enter default type, clear 11th byte
        05    0125   878                 RSB
                        0126   879
        52   D4    0126   880  10$:       CLRL    R2                       ; Nullify block pointer
        05    0128   881                 RSB                               ;
                        0129   882
                        0129   883  ;++
                        0129   884  ;
                        0129   885  ;  NET$DEALLOCATE - Deallocate memory
                        0129   886  ;
                        0129   887  ;  INPUTS:        R0 - Address of block to be deallocated
                        0129   888  ;--
                        0129   889  NET$DEALLOCATE::
        50   D5    0129   890                 TSTL    R0                       ; Is there a block to deallocate?
        5C   13    012B   891                 BEQL    70$                      ; If EQL then there's no block
     04 A0   6E   D0   012D   892                 MOVL    (SP),4(R0)               ; Save caller's address for journalling
        1E   BB    0131   893                 PUSHR   #^M<R1,R2,R3,R4>
        50   D5    0133   894                 TSTL    R0                       ; Is block from system pool?
        2B   19    0135   895                 BLSS    50$                      ; If LSS yes
        50   0F   D3    0137   896                 BITL    #MASK,R0                 ; Block aligned on boundary?
        0C   12    013A   897                 BNEQ    10$                      ; If NEQ no - bad deallocation
     51 08 A0  3C   013C 898                 MOVZWL  IRP$W_SIZE(R0),R1        ; Get size of block in bytes
        51   0F   C0    0140   899                 ADDL    #MASK,R1                 ; Round size up to next boundary
        51   0F   CA    0143   900                 BICL    #MASK,R1                 ; Truncate size back to multiple
        04   12    0146   901                 BNEQ    20$                      ; If NEQ okay
```

NETTRN   - Main ACP loop and misc. subroutines   16-SEP-1984 01:29:26   VAX/VMS Macro V04-00   Page 24
V04-000   Pool allocation routines   5-SEP-1984 02:21:47   [NETACP.SRC]NETTRN.MAR;1   (16)

B 5

```
                        0148    902 10$:    BUG_CHECK BADDALRQSZ,FATAL    ; Bad deallocation request size or address
       53    0000'CF    9E 014C 903 20$:    MOVAB     NET$GL_MY_POOL,R3    ; Point to memory listhead
                           0151 904         DSBINT    (R3)+                ; Synchronize pool
          00000000'GF    16 0157 905         JSB       G^EXE$DEALLOCATE     ; Deallocate the block
                           015D 906         ENBINT                         ; Restore IPL
                   25    11 0160 907         BRB       60$                  ; Continue
       54    50.    D0 0162 908 50$:        MOVL      R0,R4                ; Save buffer address
             019F'    30 0165 909           BSBW      NET$JNX_CO           ; See if journalling is enabled
          13    50    E9 0168 910           BLBC      R0,55$               ; If LBC then no
    81 000000DE 8F    D0 016B 911           MOVL      #^X<DE>,(R1)+        ; Enter record type
          81    54    D0 0172 912           MOVL      R4,(R1)+             ; Enter buffer address
          81    64    7D 0175 913           MOVQ      (R4),(R1)+           ; Enter buffer header
    81    08 A4    7D 0178 914             MOVQ      8(R4),(R1)+
             9E    16 017C 915             JSB       @(SP)+               ; Return to co-routine to fill journal
          50    54    D0 017E 916 55$:      MOVL      R4,R0                ; Restore buffer address
          00000000'GF    16 0181 917         JSB       G^EXE$DEANONPAGED    ; Deallocate and return
                   1E    BA 0187 918 60$:    POPR      #^M<R1,R2,R3,R4>
                   05 0189 919 70$:          RSB
                           018A 920
                   00000312 921             .RESTORE
                           0312 922
                           0312 923 .END
```

```
$$T1                      = 00000000           NET$C_ACT_TIMER          = 0000001E
ACP$C_STA_F               = 00000004           NET$C_DYN_WQE            = 00000021  G
ACP$C_STA_H               = 00000005           NET$C_EFN_ASYN           = 00000002
ACP$C_STA_I               = 00000000           NET$C_EFN_WAIT           = 00000001
ACP$C_STA_N               = 00000001           NET$C_IPL               = 00000008
ACP$C_STA_R               = 00000002           NET$C_MAXACCFLD          = 00000027
ACP$C_STA_S               = 00000003           NET$C_MAXLINNAM          = 0000000F
AQB$L_ACPQFL              = 00000000           NET$C_MAXLNK             = 000003FF
BIT...                    = 00000006           NET$C_MAXNODNAM          = 00000006
BUF_SIZ                   = 00000040           NET$C_MAXOBJNAM          = 0000000C
BUG$_BADALORQSZ           ********   X   05     NET$C_MAX_AREAS          = 0000003F
BUG$_BADDALRQSZ           ********   X   05     NET$C_MAX_LINES          = 00000040
BUG$_NETNOBUF             ********   X   04     NET$C_MAX_NCB            = 0000006E
BUG$_NETNOSTATE           ********   X   04     NET$C_MAX_NODES          = 000003FF
CMPTIM_321                00000220 R     04     NET$C_MAX_OBJ            = 000000FF
CNF$_ADVANCE              = 00000000           NET$C_MAX_WQE            = 00000014
CNF$_QUIT                 = 00000002           NET$C_MINBUFSIZ          = 000000C0
CNF$_TAKE_CURR            = 00000003           NET$C_TID_ACT            = 00000003
CNF$_TAKE_PREV            = 00000001           NET$C_TID_RUS            = 00000001
COM$POST                  ********   X   04     NET$C_TID_XRT            = 00000002
DLE$DISPATCH              ********   X   04     NET$C_TRCTL_CEL          = 00000002
DYN$C_CXB                 = 0000001B           NET$C_TRCTL_OVR          = 00000005
DYN$C_IRP                 = 0000000A           NET$C_UTLBUFSIZ          = 00001000
DYN$C_NDB                 = 0000001C           NET$DEALLOCATE           00000129 RG    05
DYN$C_NET                 = 00000017           NET$DEC_TRANS            ********   X   04
EXE$ALLOCATE              ********   X   05     NET$DISPATCH             00000000 RG    04
EXE$ALONONPAGED           ********   X   05     NET$DLL_RCV              ********   X   04
EXE$DEALLOCATE            ********   X   05     NET$GETUTLBUF            000002DA RG    04
EXE$DEANONPAGED           ********   X   05     NET$GL_FLAGS             ********   X   04
EXE$GQ_SYSTIME            ********   X   04     NET$GL_MY_POOL           ********   X   05
EXE$INSIOQ                ********   X   04     NET$GL_PTR_AQB           ********   X   04
FIND_JNX                  00000071 R     05     NET$GL_PTR_VCB           ********   X   04
GO                        0000000B R     04     NET$GL_SAVE_IRP          ********   X   04
INIT                      00000118 R     05     NET$GL_SAVE_UCB          ********   X   04
IO$_ACCESS                = 00000032           NET$GQ_TMP_BUF           ********   X   04
IO$_ACPCONTROL            = 00000038           NET$GQ_USR_STAT          ********   X   04
IO$_DEACCESS              = 00000034           NET$GQ_WQE_TIMR          00000024 RG    03
IRP$B_TYPE                = 0000000A           NET$GQ_WQE_WORK          00000000 RG    03
IRP$L_MEDIA               = 00000038           NET$JNX_CO               00000307 RG    04
IRP$L_UCB                 = 0000001C           NET$M_MAXLNKMSK          = 000003FF
IRP$L_WIND                = 00000018           NET$PROC_XWB             ********   X   04
IRP$S_FCODE               = 00000006           NET$V_RQIRP              = 00000005
IRP$V_COMPLX              = 00000003           NET$V_TIMER              = 00000004
IRP$V_FCODE               = 00000000           NET$V_UTLBUF             = 00000006
IRP$V_PHYSIO              = 00000008           NET_TIMER                00000050 R     03
IRP$W_FUNC                = 00000020           NSP$C_EXT_LNK            = 0000001E
IRP$W_SIZE                = 00000008           NSP$C_MAXADR             = 00000009
IRP$W_STS                 = 0000002A           POOL_EXTEND              = 00002800
JNX_CALLBACK              00000000 R     05     PR$_IPL                  ********   X   04
MASK                      = 0000000F           PROC_IRP                 0000008A R     04
NET$ACP_CANCEL            ********   X   04     RANGE                    00000000 R     02
NET$ALLOCATE              0000009E RG    05     RCB$B_AQB_CNT            = 000000A9
NET$ALONPAGED             00000092 RG    05     RCB$L_PTR_JNX            = 00000018
NET$ALONPGD_Z             0000007D RG    05     SET_TIMER                000001EE R     04
NET$BIN2ASC               000002ED RG    04     SIZ...                   = 00000001
NET$CONNECT               ********   X   04     SS$_ILLIOFUNC            ********   X   04
NET$CONTROL_QIO           ********   X   04     SS$_NORMAL               ********   X   04
```

```
SYS$CANTIM                      ********  GX   04
SYS$EXPREG                      ********  GX   05
SYS$HIBER                       ********  GX   04
SYS$PURGWS                      ********  GX   04
SYS$SETIMR                      ********  GX   04
SYS$WAKE                        ********  GX   04
TIMER_ACTION                    00000164  R    04
TIMER_EXP                       000001C6  R    04
TR$C_MAXHDR                  =  0000001C
TR$C_NI_ALLEND1              =  040000AB
TR$C_NI_ALLEND2              =  00000000
TR$C_NI_ALLROU1              =  030000AB
TR$C_NI_ALLROU2              =  00000000
TR$C_NI_PREFIX               =  000400AA
TR$C_NI_PROT                 =  00000360
TR$C_PRI_ECL                 =  0000001F
TR$C_PRI_RTHRU               =  0000001F
WAIT                            00000114  R    04
WQE$ALLOCATE                    00000272  RG   04
WQE$B_EVT                    =  00000010
WQE$B_SUB                    =  0000000B
WQE$B_TYPE                   =  0000000A
WQE$CANCEL_TIM                  00000167  RG   04
WQE$C_EXT_LEN                =  00000008
WQE$C_LENGTH                 =  00000024
WQE$C_SUB_ACP                =  00000001
WQE$C_SUB_AST                =  00000003
WQE$C_SUB_BAS                =  00000000
WQE$C_SUB_MBX                =  00000005
WQE$C_SUB_TIM                =  00000007
WQE$C_TMR_LEN                   0000002C
WQE$DEALLOCATE                  00000293  RG   04
WQE$FORK                        000002A7  RG   04
WQE$INSQUE                      00000229  RG   04
WQE$L_ACTION                 =  0000000C
WQE$L_BLINK                  =  00000004
WQE$L_EVL_PKT                =  00000018
WQE$L_FLINK                  =  00000000
WQE$L_PM1                    =  00000010
WQE$L_PM2                    =  00000014
WQE$Q_DUE_TIME                  00000024
WQE$REMQUE                      00000253  RG   04
WQE$RESET_TIM                   0000010B  RG   04
WQE$TIMER_AST                   000001B2  RG   04
WQE$W_ADJ_INX                =  00000020
WQE$W_SIZE                   =  00000008
_$$_                         =  000000EF
```

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

PSECT name                 Allocation          PSECT No.   Attributes
----------                 ----------          ---------   ----------
.  ABS  .                  00000000  (    0.)   00 (  0.)   NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                      0000002C  (   44.)   01 (  1.)   NOPIC   USR   CON   ABS   LCL NOSHR  EXE   RD     WRT NOVEC BYTE
NET_PURE                   00000008  (    8.)   02 (  2.)   NOPIC   USR   CON   REL   LCL NOSHR NOEXE  RD   NOWRT NOVEC LONG
NET_IMPURE                 00000054  (   84.)   03 (  3.)   NOPIC   USR   CON   REL   LCL NOSHR NOEXE  RD     WRT NOVEC LONG
NET_CODE                   00000312  (  786.)   04 (  4.)   NOPIC   USR   CON   REL   LCL NOSHR  EXE   RD   NOWRT NOVEC BYTE
NET_LOCK_CODE              0000018A  (  394.)   05 (  5.)   NOPIC   USR   CON   REL   GBL NOSHR  EXE   RD   NOWRT NOVEC BYTE

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

Phase                 Page faults     CPU Time        Elapsed Time
-----                 -----------     --------        ------------
Initialization                 29     00:00:00.15     00:00:00.63
Command processing            124     00:00:01.03     00:00:04.46
Pass 1                        534     00:00:19.52     00:00:41.20
Symbol table sort               0     00:00:02.77     00:00:05.28
Pass 2                        175     00:00:03.88     00:00:09.46
Symbol table output            20     00:00:00.19     00:00:00.31
Psect synopsis output           2     00:00:00.03     00:00:00.03
Cross-reference output          0     00:00:00.00     00:00:00.00
Assembler run totals          886     00:00:27.57     00:01:01.37
```

The working set limit was 1800 pages.
105928 bytes (207 pages) of virtual memory were used to buffer the intermediate code.
There were 100 pages of symbol table space allocated to hold 1779 non-local and 71 local symbols.
923 source lines were read in Pass 1, producing 25 object records in Pass 2.
52 pages of virtual memory were used to define 48 macros.

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

Macro library name                          Macros defined
------------------                          --------------
_$255$DUA28:[SHRLIB]NMALIBRY.MLB;1                  0
_$255$DUA28:[SHRLIB]EVCDEF.MLB;1                    0
_$255$DUA28:[NETACP.OBJ]NETDRV.MLB;1               0
_$255$DUA28:[NETACP.OBJ]NET.MLB;1                   9
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                     12
_$255$DUA28:[SYSLIB]STARLET.MLB;2                  18
TOTALS (all libraries)                             39
```

2052 GETS were required to define 39 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:NETTRN/OBJ=OBJ$:NETTRN MSRC$:NETTRN/UPDATE=(ENH$:NETTRN)+EXECML$/LIB+LIB$:NET/LIB+LIB$:NETDRV/LIB+SHRLIB$:EVCDEF/LIB+

NICNF

CNFDEF
SDL

CNFDEF
LIS

NETTRN
LIS

NICONFIG
MAP

NETTREE
LIS

CNFMAIN
LIS

CNFREQUES
LIS

SERVER
LIS

CNFINTRPT
LIS

CNFWQDEF
SDL

CNFPREFIX
REQ

CNFMSG
LIS