```
PPPPPPPPPPPP      HHH          HHH     000000000      NNN          NNN  EEEEEEEEEEEEEEE
PPPPPPPPPPPPP     HHH          HHH     000000000      NNN          NNN  EEEEEEEEEEEEEEE
PPPPPPPPPPPPP     HHH          HHH     000000000      NNN          NNN  EEEEEEEEEEEEEEE
PPP          PPP  HHH          HHH  000          000  NNN          NNN  EEE
PPP          PPP  HHH          HHH  000          000  NNN          NNN  EEE
PPP          PPP  HHH          HHH  000          000  NNN          NNN  EEE
PPP          PPP  HHH          HHH  000          000  NNNNNN       NNN  EEE
PPP          PPP  HHH          HHH  000          000  NNNNNN       NNN  EEE
PPP          PPP  HHH          HHH  000          000  NNNNNN       NNN  EEE
PPPPPPPPPPPP      HHHHHHHHHHHHHHHH  000          000  NNN   NNN    NNN  EEEEEEEEEEE
PPPPPPPPPPPP      HHHHHHHHHHHHHHHH  000          000  NNN   NNN    NNN  EEEEEEEEEEE
PPPPPPPPPPPP      HHHHHHHHHHHHHHHH  000          000  NNN   NNN    NNN  EEEEEEEEEEE
PPP               HHH          HHH  000          000  NNN      NNNNNN  EEE
PPP               HHH          HHH  000          000  NNN      NNNNNN  EEE
PPP               HHH          HHH  000          000  NNN      NNNNNN  EEE
PPP               HHH          HHH  000          000  NNN         NNN  EEE
PPP               HHH          HHH  000          000  NNN         NNN  EEE
PPP               HHH          HHH  000          000  NNN         NNN  EEE
PPP               HHH          HHH     000000000      NNN         NNN  EEEEEEEEEEEEEEE
PPP               HHH          HHH     000000000      NNN         NNN  EEEEEEEEEEEEEEE
PPP               HHH          HHH     000000000      NNN         NNN  EEEEEEEEEEEEEEE
```

FILEID**NETSLAVE

```
NN      NN  EEEEEEEEEE  TTTTTTTTTT    SSSSSSSS  LL          AAAAAA    VV      VV  EEEEEEEEEE
NN      NN  EEEEEEEEEE  TTTTTTTTTT    SSSSSSSS  LL          AAAAAA    VV      VV  EEEEEEEEEE
NN      NN  EE              TT      SS          LL         AA      AA  VV      VV  EE
NN      NN  EE              TT      SS          LL         AA      AA  VV      VV  EE
NNNN    NN  EE              TT      SS          LL         AA      AA  VV      VV  EE
NNNN    NN  EE              TT      SS          LL         AA      AA  VV      VV  EE
NN  NN  NN  EEEEEEEE        TT        SSSSSS    LL         AA      AA  VV      VV  EEEEEEEE
NN  NN  NN  EEEEEEEE        TT        SSSSSS    LL         AA      AA  VV      VV  EEEEEEEE
NN    NNNN  EE              TT              SS  LL         AAAAAAAAAA  VV      VV  EE
NN    NNNN  EE              TT              SS  LL         AAAAAAAAAA  VV      VV  EE
NN      NN  EE              TT              SS  LL         AA      AA    VV  VV    EE              ....
NN      NN  EE              TT              SS  LL         AA      AA    VV  VV    EE              ....
NN      NN  EEEEEEEEEE      TT      SSSSSSSS    LLLLLLLLLL AA      AA      VV      EEEEEEEEEE      ....
NN      NN  EEEEEEEEEE      TT      SSSSSSSS    LLLLLLLLLL AA      AA      VV      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
```

```
;  1          0001  0 %title 'NETSLAVE - Network Slave Program'
;  2          0002  0        module netslave (
;  3          0003  1                         ident='V04-000') = begin
;  4          0004  1
;  5          0005  1 !
;  6          0006  1 !*****************************************************************************
;  7          0007  1 !*                                                                           *
;  8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;  9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
; 10          0010  1 !*  ALL RIGHTS RESERVED.                                                     *
; 11          0011  1 !*                                                                           *
; 12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
; 13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
; 14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
; 15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
; 16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
; 17          0017  1 !*  TRANSFERRED.                                                             *
; 18          0018  1 !*                                                                           *
; 19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
; 20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
; 21          0021  1 !*  CORPORATION.                                                             *
; 22          0022  1 !*                                                                           *
; 23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
; 24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
; 25          0025  1 !*                                                                           *
; 26          0026  1 !*                                                                           *
; 27          0027  1 !*****************************************************************************
; 28          0028  1 !
; 29          0029  1
; 30          0030  1 !++
; 31          0031  1 ! Facility:     VAX/VMS Telephone Facility, Network Slave Program
; 32          0032  1 !
; 33          0033  1 ! Abstract:     This module is the network slave program.  When someone
; 34          0034  1 !               on a remote node wants to talk to someone on our node,
; 35          0035  1 !               this module acts as the network slave intermediary.
; 36          0036  1 !
; 37          0037  1 !
; 38          0038  1 ! Environment:  The life and times of a network slave is restricted to
; 39          0039  1 !               two controlled cases:
; 40          0040  1 !
; 41          0041  1 !               1.  The master creates the slave for information-gathering
; 42          0042  1 !                   purposes (e.g., remote directory).  The slave exists only
; 43          0043  1 !                   to gather information, and dies when the gathering is done.
; 44          0044  1 !
; 45          0045  1 !               2.  The master creates the slave so it can talk to a user
; 46          0046  1 !                   on the slave's node.  The slave provides ONE-WAY
; 47          0047  1 !                   communication from the master to the user.  The
; 48          0048  1 !                   user talks back to the master via a symmetrical slave.
; 49          0049  1 !
; 50          0050  1 !               NOTE that some errors are handled in the slave by simply
; 51          0051  1 !               dying and letting the master worry about it.
; 52          0052  1 !
; 53          0053  1 !
; 54          0054  1 ! Author: Paul C. Anagnostopoulos, Creation Date: 11 December 1980
; 55          0055  1 !
; 56          0056  1 ! Modified By:
; 57          0057  1 !
```

```
;    58          0058  1 !       V03-002 BLS0251         Benn Schreiber          8-Dec-1983
;    59          0059  1 !               Check that we are network process before translating
;    60          0060  1 !               SYS$NET.
;    61          0061  1 !
;    62          0062  1 !       V03-001 PCA0044         Paul Anagnostopoulos    26-Mar-1982
;    63          0063  1 !               Minor changes to convert from process name to user name.
;    64          0064  1 !--
```

```
;    66          0065  1 %sbttl 'Module Declarations'
;    67          0066  1 !
;    68          0067  1 ! Libraries and Requires:
;    69          0068  1 !
;    70          0069  1
;    71          0070  1 library 'sys$library:lib';
;    72          0071  1 require 'phonereq';
;    73          0400  1
;    74          0401  1 !
;    75          0402  1 ! Table of Contents:
;    76          0403  1 !
;    77          0404  1                                                                                         ;
;    78          0405  1 forward routine
;    79          0406  1         phn$init_slave: novalue,
;    80          0407  1         phn$slave_drive: novalue,
;    81          0408  1         phn$slave_directory2: novalue,
;    82          0409  1         phn$slave_verify: novalue,
;    83          0410  1         phn$slave_ring_out: novalue,
;    84          0411  1         phn$slave_pass_smb: novalue,
;    85          0412  1         phn$write_slave_status: novalue,
;    86          0413  1         phn$read_slave;
;    87          0414  1
;    88          0415  1 !
;    89          0416  1 ! External References:
;    90          0417  1 !
;    91          0418  1
;    92          0419  1 external routine
;    93          0420  1         phn$directory_line,
;    94          0421  1         phn$estab_link,
;    95          0422  1         phn$local_jangle,
;    96          0423  1         phn$make_pub,
;    97          0424  1         phn$send_smb;
;    98          0425  1
;    99          0426  1 !
;   100          0427  1 ! Own Variables:
;   101          0428  1 !
;   102          0429  1 ! The following word contains the DECnet logical link channel number.
;   103          0430  1
;   104          0431  1 own
;   105          0432  1         decnet_channel: word;
;   106          0433  1
;   107          0434  1 ! The following table sets up a correspondence between various PHONE
;   108          0435  1 ! status codes and the universal code transmitted as part of our network
;   109          0436  1 ! protocol.  We can't transmit VMS-format status codes because they may
;   110          0437  1 ! change.
;   111          0438  1
;   112          0439  1 own
;   113          0440  1         status_code_table: vector[10,long] initial(     ! universal code
;   114          0441  1                                 phn$_linkerror,       !       0
;   115          0442  1                                 phn$_ok,              !       1
;   116          0443  1                                 phn$_badspec,         !       2
;   117          0444  1                                 phn$_cantreach,       !       3
;   118          0445  1                                 phn$_needuser,        !       4
;   119          0446  1                                 phn$_nopriv,          !       5
;   120          0447  1                                 phn$_noproc,          !       6
;   121          0448  1                                 phn$_targterm,        !       7
;   122          0449  1                                 phn$_loggedoff,       !       8
```

```
;  123     0450  1                   phn$_unplugged          !       9
;  124     0451  1                   );
```

```
;  126          0452  1 %sbttl 'PHN$INIT_SLAVE - Initialize Network Slave'
;  127          0453  1 !++
;  128          0454  1 ! Functional Description:
;  129          0455  1 !       This routine is called by the PHONE mainline to see if indeed
;  130          0456  1 !       we are a network slave.  If not, we just return.  If so, then we
;  131          0457  1 !       establish a logical link and invoke the network slave mainline.
;  132          0458  1 !
;  133          0459  1 ! Formal Parameters:
;  134          0460  1 !       none
;  135          0461  1 !
;  136          0462  1 ! Implicit Inputs:
;  137          0463  1 !       global data
;  138          0464  1 !
;  139          0465  1 ! Implicit Outputs:
;  140          0466  1 !       global data
;  141          0467  1 !
;  142          0468  1 ! Returned Value:
;  143          0469  1 !       none
;  144          0470  1 !
;  145          0471  1 ! Side Effects:
;  146          0472  1 !
;  147          0473  1 !--
;  148          0474  1
;  149          0475  1
;  150          0476  2 global routine phn$init_slave: novalue = begin
;  151          0477  2
;  152          0478  2 local
;  153          0479  2         procsts : long volatile,
;  154          0480  2         getjpilst : $itmlst_decl(items=1),
;  155          0481  2         status: long;
;  156          0482  2
;  157          0483  2 ! If we are not a network process, then do not attempt to translate
;  158          0484  2 ! SYS$NET at all.
;  159          0485  2
;  160          0486  2 procsts = 0;
;  161        P 0487  2 $itmlst_init(itmlst=getjpilst,
;  162          0488  2         (itmcod=jpi$_sts,bufadr=procsts));
;  163          0489  2
;  164          0490  2 $getjpiw(itmlst=getjpilst);
;  165          0491  2 if not .procsts<$bitposition(pcb$v_netwrk),1> then
;  166          0492  2     return;
;  167          0493  2
;  168          0494  2 ! We will determine if we are a network slave by trying to assign to SYS$NET.
;  169          0495  2 ! If it fails, we aren't, and we just return.  If it succeeds, we are, and
;  170          0496  2 ! we have completed the logical link.
;  171          0497  2
;  172        P 0498  2 status = $assign(devnam=describe('SYS$NET'),
;  173          0499  2                  chan=decnet_channel);
;  174          0500  2 if not .status then
;  175          0501  2         return;
;  176          0502  2
;  177          0503  2 ! Now we can simply wait for steering messages from the remote node.
;  178          0504  2
;  179          0505  2 phn$slave_drive();
;  180          0506  2
;  181          0507  1 end;
```

```
                                                        .TITLE  NETSLAVE NETSLAVE - Network Slave Program
                                                        .IDENT  \V04-000\

                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                        54  45  4E  24  53  59  53  00000 P.AAB:  .ASCII  \SYS$NET\                                ;
                                                    00007       .BLKB   1
                                            00000007    00008 P.AAA:  .LONG   7                                    ;
                                            00000000'   0000C         .ADDRESS P.AAB                           ;

                                                        .PSECT  $OWN$,NOEXE,2

                                                    00000 DECNET_CHANNEL:
                                                        .BLKB   2
                                                    00002       .BLKB   2
00000000G 00000000G 00000000G 00000000G 00000000G 00000000G 00004 STATUS_CODE_TABLE:
                                                        .LONG   PHN$_LINKERROR, PHN$_OK, PHN$_BADSPEC, -         ;
                    00000000G 00000000G 00000000G 00000000G 0001C               PHN$_CANTREACH, PHN$_NEEDUSER, -  ;
                                                                PHN$_NOPRIV, PHN$_NOPROC, PHN$_TARGTERM, -       ;
                                                                PHN$_LOGGEDOFF, PHN$_UNPLUGGED                   ;

                                                        .EXTRN  PHN$_OK, PHN$_ANSWERED
                                                        .EXTRN  PHN$_BUSYCALL, PHN$_CANCALL
                                                        .EXTRN  PHN$_CANTREACH, PHN$_CONFCALL
                                                        .EXTRN  PHN$_DEAD, PHN$_DECNETLINK
                                                        .EXTRN  PHN$_DIRCAN, PHN$_FACSCAN
                                                        .EXTRN  PHN$_HELPCAN, PHN$_HUNGUP
                                                        .EXTRN  PHN$_JUSTRANG, PHN$_LOGGEDOFF
                                                        .EXTRN  PHN$_REJECTED, PHN$_RING
                                                        .EXTRN  PHN$_REJECTJUNK
                                                        .EXTRN  PHN$_SENDINGMAIL
                                                        .EXTRN  PHN$_BADCMD, PHN$_BADHELP
                                                        .EXTRN  PHN$_BADMAILCMD
                                                        .EXTRN  PHN$_BADSMB, PHN$_BADSPEC
                                                        .EXTRN  PHN$_HELPMISSING
                                                        .EXTRN  PHN$_IVREDUNANS
                                                        .EXTRN  PHN$_IVREDUNCALL
                                                        .EXTRN  PHN$_LINKERROR, PHN$_NEEDUSER
                                                        .EXTRN  PHN$_NOCALL, PHN$_NOHOLDS
                                                        .EXTRN  PHN$_NOPORTS, PHN$_NOPRIV
                                                        .EXTRN  PHN$_NOPROC, PHN$_NOTCONV
                                                        .EXTRN  PHN$_ONLYNODE, PHN$_PHONEBUSY
                                                        .EXTRN  PHN$_REMOTEERROR
                                                        .EXTRN  PHN$_TARGTERM, PHN$_UNPLUGGED
                                                        .EXTRN  PHN$_BADTERM, PHN$_SHAREDMBX
                                                        .EXTRN  PHN$_INPUTTERM, PHN$GQ_NODE_NAME
                                                        .EXTRN  PHN$GQ_SWITCH_HOOK
                                                        .EXTRN  PHN$GL_VIEWPORT_SIZE
                                                        .EXTRN  PHN$GB_SCROLL, PHN$GQ_PUBHEAD
                                                        .EXTRN  PHN$GB_FLAGS, PHN$DIRECTORY_LINE
                                                        .EXTRN  PHN$ESTAB_LINK, PHN$LOCAL_JACKLE
                                                        .EXTRN  PHN$MAKE_PUB, PHN$SEND_SMB
                                                        .EXTRN  SYS$GETJPIW, SYS$ASSIGN

                                                        .PSECT  $CODE$,NOWRT,2
```

```
                                          0000 00000          .ENTRY  PHN$INIT_SLAVE, Save nothing                ; 0476
                                    5E    14   C2 00002          SUBL2   #20, SP                                     ;
                                      10  AE   D4 00005          CLRL    PROCSTS                                     ; 0486
                                    50    6E   9E 00008          MOVAB   GETJPILST, $$ITMBLKPTR                      ; 0488
                                    80 03050004 8F   D0 0000B    MOVL    #50659332, ($$ITMBLKPTR)+                   ;
                                    80      10  AE   9E 00012    MOVAB   PROCSTS, ($$ITMBLKPTR)+                     ;
                                            80  7C   00016      CLRQ    ($$ITMBLKPTR)+                              ;
                                            7E  7C   00018      CLRQ    -(SP)                                       ; 0490
                                            7E  D4   0001A      CLRL    -(SP)                                       ;
                                          0C    AE   9F 0001C    PUSHAB  GETJPILST                                   ;
                                            7E  7C   0001F      CLRQ    -(SP)                                       ;
                                            7E  D4   00021      CLRL    -(SP)                                       ;
                              00000000G     00   07   FB 00023  CALLS   #7, SYS$GETJPIW                            ;
                                19    12    AE   05   E1 0002A  BBC     #5, PROCSTS+2, 1$                           ; 0491
                                            7E   7C   0002F     CLRQ    -(SP)                                       ; 0499
                                      0000' CF   9F   00031     PUSHAB  DECNET_CHANNEL                              ;
                                      0000' CF   9F   00035     PUSHAB  P.AAA                                       ;
                                 00000000G  00   04   FB 00039  CALLS   #4, SYS$ASSIGN                              ;
                                            05   50   E9 00040  BLBC    STATUS, 1$                                  ; 0500
                                     0000V  CF   00   FB 00043  CALLS   #0, PHN$SLAVE_DRIVE                         ; 0505
                                                 04   00048 1$:  RET                                               ; 0507

; Routine Size:  73 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  183          0508  1 %sbttl 'PHN$SLAVE_DRIVE - Handle Steering Messages'
;  184          0509  1 !++
;  185          0510  1 ! Functional Description:
;  186          0511  1 !       This routine is the "mainline" for the network slave program.
;  187          0512  1 !       It waits for steering messages to be sent across the logical
;  188          0513  1 !       link and then acts accordingly.  Most messages can be sent
;  189          0514  1 !       on to the target task, but a few are handled herein.
;  190          0515  1 !
;  191          0516  1 ! Formal Parameters:
;  192          0517  1 !       none
;  193          0518  1 !
;  194          0519  1 ! Implicit Inputs:
;  195          0520  1 !       global data
;  196          0521  1 !
;  197          0522  1 ! Implicit Outputs:
;  198          0523  1 !       global data
;  199          0524  1 !
;  200          0525  1 ! Returned Value:
;  201          0526  1 !       none
;  202          0527  1 !
;  203          0528  1 ! Side Effects:
;  204          0529  1 !
;  205          0530  1 !--
;  206          0531  1
;  207          0532  1
;  208          0533  2 global routine phn$slave_drive: novalue = begin
;  209          0534  2
;  210          0535  2 local
;  211          0536  2         status: long,
;  212          0537  2         decnet_iosb: block[8,byte];
;  213          0538  2
;  214          0539  2
;  215          0540  2 ! Just sit in a loop processing steering messages.
;  216          0541  2
;  217          0542  3 loop (
;  218          0543  3         local
;  219          0544  3                 local_described_buffer(smb_buf,phn$k_mbxsize);
;  220          0545  3
;  221          0546  3         ! Wait for a steering message from the remote person.  If we get
;  222          0547  3         ! a transmission error, just die.
;  223          0548  3
;  224        P 0549  3         status = $qiow(efn=phn$k_decnetefn,
;  225        P 0550  3                        chan=.decnet_channel,
;  226        P 0551  3                        func=io$_readvblk,
;  227        P 0552  3                        iosb=decnet_iosb,
;  228        P 0553  3                        p1=.smb_buf[ptr],
;  229          0554  3                        p2=.smb_buf[len]);
;  230          0555  3         check (.status);
;  231          0556  3         check (.decnet_iosb[0,0,16,0]);
;  232          0557  3         smb_buf[len] = .decnet_iosb[2,0,16,0];
;  233          0558  3
;  234          0559  3         ! Now we can extract the steering message type code and decide what to
;  235          0560  3         ! do with the SMB.  Some routines are passed the steering message
;  236          0561  3         ! descriptor, which they can clobber.
;  237          0562  3
;  238          0563  3         selectoneu ch$rchar(.smb_buf[ptr]) of set
;  239          0564  3
```

```
;  240          0565  3          [smb__directory2]:
;  241          0566  3                          phn$slave_directory2();
;  242          0567  3
;  243          0568  3          [smb__slave_verify]:
;  244          0569  3                          phn$slave_verify(smb_buf);
;  245          0570  3
;  246          0571  3          [smb__rang_in]:
;  247          0572  3                          phn$slave_ring_out(smb_buf);
;  248          0573  3
;  249          0574  3          [smb__slave_done]:
;  250          0575  3                          $exit(code=ss$_normal);
;  251          0576  3
;  252          0577  3          [otherwise]:    phn$slave_pass_smb(smb_buf);
;  253          0578  3          tes;
;  254          0579  3
;  255          0580  2 );
;  256          0581  2
;  257          0582  1 end;


                                                                .EXTRN  SYS$QIOW, SYS$EXIT

                                                  001C 00000    .ENTRY  PHN$SLAVE_DRIVE, Save R2,R3,R4                  ; 0533
                    54 00000000G   00    9E 00002             MOVAB   LIB$SIGNAL, R4                                  ;
                    5E     FEF0    CE    9E 00009             MOVAB   -272(SP), SP                                    ;
                    6E     0100    8F    3C 0000E 1$:         MOVZWL  #256, SMB_BUF                                   ; 0544
              04    AE       08    AE    9E 00013             MOVAB   SMB_BUF+8, SMB_BUF+4                            ;
                                   7E    7C 00018             CLRQ    -(SP)                                           ; 0554
                                   7E    7C 0001A             CLRQ    -(SP)                                           ;
                    7E       10    AE    3C 0001C             MOVZWL  SMB_BUF, -(SP)                                  ;
                             18    AE    DD 00020             PUSHL   SMB_BUF+4                                       ;
                                   7E    7C 00023             CLRQ    -(SP)                                           ;
                             F8    AD    9F 00025             PUSHAB  DECNET_IOSB                                     ;
                                   31    DD 00028             PUSHL   #49                                             ;
                          7E     0000'   CF    3C 0002A       MOVZWL  DECNET_CHANNEL, -(SP)                           ;
                                   04    DD 0002F             PUSHL   #4                                              ;
              00000000G   00             0C    FB 00031       CALLS   #12, SYS$QIOW                                   ;
                          53             50    DD 00038       MOVL    R0, STATUS                                      ;
                          05             53    E8 0003B       BLBS    STATUS, 2$                                      ; 0555
                                         53    DD 0003E       PUSHL   STATUS                                          ;
                          64             01    FB 00040       CALLS   #1, LIB$SIGNAL                                  ;
                          07       F8    AD    E8 00043 2$:   BLBS    DECNET_IOSB, 3$                                 ; 0556
                          7E       F8    AD    3C 00047       MOVZWL  DECNET_IOSB, -(SP)                              ;
                          64             01    FB 0004B       CALLS   #1, LIB$SIGNAL                                  ;
                          6E       FA    AD    B0 0004E 3$:   MOVW    DECNET_IOSB+2, SMB_BUF                          ; 0557
                          52       04    BE    9A 00052       MOVZBL  @SMB_BUF+4, R2                                  ; 0563
                          0F             52    91 00056       CMPB    R2, #15                                         ; 0565
                                         07    12 00059       BNEQ    5$                                              ;
                    0000V CF             00    FB 0005B       CALLS   #0, PHN$SLAVE_DIRECTORY2                        ; 0566
                                         AC    11 00060 4$:   BRB     1$                                              ;
                          07             52    91 00062 5$:   CMPB    R2, #7                                          ; 0568
                                         09    12 00065       BNEQ    6$                                              ;
                                         5E    DD 00067       PUSHL   SP                                              ; 0569
                    0000V CF             01    FB 00069       CALLS   #1, PHN$SLAVE_VERIFY                            ;
                                         9E    11 0006E       BRB     1$                                              ;
                          08             52    91 00070 6$:   CMPB    R2, #8                                          ; 0571
```

```
                                09  12 00073          BNEQ    7$                                      ;
                                5E  DD 00075          PUSHL   SP                                      ; 0572
                      0000V  CF 01  FB 00077          CALLS   #1, PHN$SLAVE_RING_OUT                  ;
                                90  11 0007C          BRB     1$                                      ;
                             0D 52  91 0007E 7$:      CMPB    R2, #13                                 ; 0574
                                0B  12 00081          BNEQ    8$                                      ;
                                01  DD 00083          PUSHL   #1                                      ; 0575
                  00000000G  00 01  FB 00085          CALLS   #1, SYS$EXIT                            ;
                                80  11 0008C          BRB     1$                                      ;
                                5E  DD 0008E 8$:      PUSHL   SP                                      ; 0577
                      0000V  CF 01  FB 00090          CALLS   #1, PHN$SLAVE_PASS_SMB                  ;
                                C9  11 00095          BRB     4$                                      ; 0537

; Routine Size:  151 bytes,    Routine Base:  $CODE$ + 0049
```

```
;  259          0583  1 %sbttl 'PHN$SLAVE_DIRECTORY2 - Handle Remote Directory'
;  260          0584  1 !++
;  261          0585  1 ! Functional Description:
;  262          0586  1 !       This steering message routine is called when the slave has been
;  263          0587  1 !       created to gather directory information for the master.  Each
;  264          0588  1 !       time the steering message is received, this routine transmits
;  265          0589  1 !       a directory line for the next process on its node.  The slave
;  266          0590  1 !       exists only for the duration of ONE remote directory request.
;  267          0591  1 !
;  268          0592  1 ! Formal Parameters:
;  269          0593  1 !       none
;  270          0594  1 !
;  271          0595  1 ! Implicit Inputs:
;  272          0596  1 !       global data
;  273          0597  1 !
;  274          0598  1 ! Implicit Outputs:
;  275          0599  1 !       global data
;  276          0600  1 !
;  277          0601  1 ! Returned Value:
;  278          0602  1 !       none
;  279          0603  1 !
;  280          0604  1 ! Side Effects:
;  281          0605  1 !
;  282          0606  1 !--
;  283          0607  1
;  284          0608  1
;  285          0609  2 global routine phn$slave_directory2: novalue = begin
;  286          0610  2
;  287          0611  2 own
;  288          0612  2         wild_pid: long initial(-1);
;  289          0613  2
;  290          0614  2 local
;  291          0615  2         status: long;
;  292          0616  2 local
;  293          0617  2         local_described_buffer(display_line,79);
;  294          0618  2
;  295          0619  2
;  296          0620  2 ! Get the directory line for the next process on our node.
;  297          0621  2
;  298          0622  2 status = phn$directory_line(wild_pid,display_line);
;  299          0623  2
;  300          0624  2 ! If we ran out of processes, then send a null line back to the master.
;  301          0625  2 ! Otherwise send the directory line.
;  302          0626  2
;  303          0627  2 if .status nequ phn$_ok then
;  304          0628  2         display_line[len] = 0;
;  305        P 0629  2 $qiow(efn=phn$k_decnetefn,
;  306        P 0630  2       chan=.decnet_channel,
;  307        P 0631  2       func=io$_writevblk,
;  308        P 0632  2       p1=.display_line[ptr],
;  309          0633  2       p2=.display_line[len]);
;  310          0634  2
;  311          0635  2 return;
;  312          0636  2
;  313          0637  1 end;
```

```
                                                        .PSECT  $OWN$,NOEXE,2

                                FFFFFFFF  0002C WILD_PID:
                                                        .LONG   -1                                      ;


                                                        .PSECT  $CODE$,NOWRT,2

                                      0000 00000        .ENTRY  PHN$SLAVE_DIRECTORY2, Save nothing      ; 0609
                           5E      AC   AE  9E 00002    MOVAB   -84(SP), SP                             ;
                           7E      4F   8F  9A 00006    MOVZBL  #79, DISPLAY_LINE                       ; 0617
                     04    AE      08   AE  9E 0000A    MOVAB   DISPLAY_LINE+8, DISPLAY_LINE+4          ;
                                        5E  DD 0000F    PUSHL   SP                                      ; 0622
                                 0000'  CF  9F 00011    PUSHAB  WILD_PID                                ;
                       0000G     CF          02  FB 00015    CALLS   #2, PHN$DIRECTORY_LINE              ;
                   00000000G     8F          50  D1 0001A    CMPL    STATUS, #PHN$_OK                    ; 0627
                                             02  13 00021    BEQL    1$                                  ;
                                             6E  B4 00023    CLRW    DISPLAY_LINE                        ; 0628
                                             7E  7C 00025 1$: CLRQ    -(SP)                               ; 0633
                                             7E  7C 00027    CLRQ    -(SP)                               ;
                           7E        10      AE  3C 00029    MOVZWL  DISPLAY_LINE, -(SP)                 ;
                                     18      AE  DD 0002D    PUSHL   DISPLAY_LINE+4                      ;
                                             7E  7C 00030    CLRQ    -(SP)                               ;
                           7E                30  7D 00032    MOVQ    #48, -(SP)                          ;
                           7E        0000'   CF  3C 00035    MOVZWL  DECNET_CHANNEL, -(SP)               ;
                                             04  DD 0003A    PUSHL   #4                                  ;
                   00000000G 00              0C  FB 0003C    CALLS   #12, SYS$QIOW                       ;
                                                 04 00043    RET                                         ; 0637

; Routine Size:  68 bytes,    Routine Base:  $CODE$ + 00E0
```

```
;  315          0638  1 %sbttl 'PHN$SLAVE_VERIFY - Verify Existence of Process'
;  316          0639  1 !++
;  317          0640  1 ! Functional Description:
;  318          0641  1 !       This steering message routine is called when the master sends us
;  319          0642  1 !       a slave_verify steering message.  It does this as its first action
;  320          0643  1 !       when it wants to set up a conversation with a remote user.
;  321          0644  1 !       We are required to verify the existence of the user and set up
;  322          0645  1 !       a link to it.
;  323          0646  1 !
;  324          0647  1 ! Formal Parameters:
;  325          0648  1 !       verify_smb      The SMB as it was sent to us.  This includes:
;  326          0649  1 !                       1.      The steering message type code.
;  327          0650  1 !
;  328          0651  1 !                       2.      The home node/user name of the master,
;  329          0652  1 !                               followed by an eofrom character.
;  330          0653  1 !
;  331          0654  1 !                       3.      The home node/user name of the target.
;  332          0655  1 !
;  333          0656  1 ! Implicit Inputs:
;  334          0657  1 !       global data
;  335          0658  1 !
;  336          0659  1 ! Implicit Outputs:
;  337          0660  1 !       global data
;  338          0661  1 !
;  339          0662  1 ! Returned Value:
;  340          0663  1 !       none
;  341          0664  1 !
;  342          0665  1 ! Side Effects:
;  343          0666  1 !
;  344          0667  1 !--
;  345          0668  1
;  346          0669  1
;  347          0670  2 global routine phn$slave_verify(verify_smb): novalue = begin
;  348          0671  2
;  349          0672  2 bind
;  350          0673  2         smb_dsc = .verify_smb: descriptor;
;  351          0674  2
;  352          0675  2 local
;  353          0676  2         status: long,
;  354          0677  2         work_dsc: descriptor,
;  355          0678  2         op: ref pub,
;  356          0679  2         tp: ref pub;
;  357          0680  2
;  358          0681  2
;  359          0682  2 ! Many routines assume that the first PUB on the chain is "our PUB",
;  360          0683  2 ! representing this user's phone.  In a network slave that doesn't really
;  361          0684  2 ! make sense, but we will create a PUB to represent the master who
;  362          0685  2 ! created us.  This PUB will be a reflection of "our PUB" in the master.
;  363          0686  2
;  364          0687  2 work_dsc[ptr] = .smb_dsc[ptr] + 1;
;  365          0688  2 work_dsc[len] = ch$find_ch(.smb_dsc[len],.smb_dsc[ptr],eofrom) - .smb_dsc[ptr] - 1;
;  366          0689  2 status = phn$make_pub(work_dsc,op);
;  367          0690  2 check (.status);
;  368          0691  2 op[pub_v_temporary] = false;
;  369          0692  2
;  370          0693  2 ! Now we can set up a PUB representing the target user on this remote
;  371          0694  2 ! node that the master wants to talk to.
```

```
;  372          0695  2
;  373          0696  2 work_dsc[ptr] = ch$find_ch(.smb_dsc[len],.smb_dsc[ptr],eofrom) + 1;
;  374          0697  2 work_dsc[len] = .smb_dsc[len] - .work_dsc[ptr] + .smb_dsc[ptr];
;  375          0698  2 status = phn$estab_link(work_dsc,tp);
;  376          0699  2 tp[pub_v_temporary] = false;
;  377          0700  2
;  378          0701  2 ! Finally, we send that status back to the master so it knows how things went.
;  379          0702  2
;  380          0703  2 phn$write_slave_status(.status);
;  381          0704  2 return;
;  38           0705  2
;  383          0706  1 end;


                                                001C 00000          .ENTRY  PHN$SLAVE_VERIFY, Save R2,R3,R4                 ; 0670
                                             5E                 10  C2 00002          SUBL2   #16, SP                         ;
                                             52          04     AC  D0 00005          MOVL    VERIFY_SMB, R2                  ; 0673
                                             53          04     A2  D0 00009          MOVL    4(R2), R3                       ; 0687
                                       0C    AE          01     A3  9E 0000D          MOVAB   1(R3), WORK_DSC+4               ;
                                 63          62                 00  3A 00012          LOCC    #0, (R2), (R3)                  ; 0688
                                                                02  12 00016          BNEQ    1$                              ;
                                                                51  D4 00018          CLRL    R1                              ;
                                             51                 53  C2 0001A 1$:      SUBL2   R3, R1                          ;
                                 08    AE    51                 01  A3 0001D          SUBW3   #1, R1, WORK_DSC                ;
                                                                5E  DD 00022          PUSHL   SP                              ; 0689
                                                          0C    AE  9F 00024          PUSHAB  WORK_DSC                        ;
                                                    0000G CF        02  FB 00027          CALLS   #2, PHN$MAKE_PUB                ;
                                             54                 50  D0 0002C          MOVL    R0, STATUS                      ;
                                             09                 54  E8 0002F          BLBS    STATUS, 2$                      ; 0690
                                       00000000G 00             01  FB 00032          CALLS   #1, LIB$SIGNAL                  ;
                                             50                 6E  D0 00034  2$:      MOVL    OP, R0                          ; 0691
                                          00F0 C0               04  8A 0003B          BICB2   #4, 240(R0)                     ;
                                 63          62                 00  3A 0003E          LOCC    #0, (R2), (R3)                  ; 0696
                                                                02  12 00043          BNEQ    3$                              ;
                                                                51  D4 00047          CLRL    R1                              ;
                                          0C    AE    01        A1  9E 00049  3$:      MOVAB   1(R1), WORK_DSC+4               ;
                                             50                 62  3C 0004B          MOVZWL  (R2), R0                        ; 0697
                                             50          0C     AE  C2 00050          SUBL2   WORK_DSC+4, R0                  ;
                                 08    AE    50                 53  A1 00053          ADDW3   R3, R0, WORK_DSC                ;
                                                          04    AE  9F 00057          PUSHAB  TP                              ; 0698
                                                          0C    AE  9F 0005A          PUSHAB  WORK_DSC                        ;
                                                    0000G CF        02  FB 0005D          CALLS   #2, PHN$ESTAB_LINK              ;
                                             54                 50  D0 00062          MOVL    R0, STATUS                      ;
                                             50          04     AE  D0 00065          MOVL    TP, R0                          ; 0699
                                          00F0 C0               04  8A 00069          BICB2   #4, 240(R0)                     ;
                                                                54  DD 0006E          PUSHL   STATUS                          ; 0703
                                                    0000V CF        01  FB 00070          CALLS   #1, PHN$WRITE_SLAVE_STATUS      ;
                                                                    04 00075          RET                                     ; 0706

; Routine Size:  118 bytes,    Routine Base:  $CODE$ + 0124
```

```
;  385          0707  1 %sbttl 'PHN$SLAVE_RING_OUT - Ring a Remote Phone'
;  386          0708  1 !++
;  387          0709  1 ! Functional Description:
;  388          0710  1 !       This steering message routine is executed when the master sends
;  389          0711  1 !       a rang_in message to the slave.  The slave is supposed to ring
;  390          0712  1 !       the remote person's phone, and return a status to the master.
;  391          0713  1 !
;  392          0714  1 ! Formal Parameters:
;  393          0715  1 !       ring_smb        The address of a descriptor of the SMB as sent
;  394          0716  1 !                       by the master.  We are only interested in the
;  395          0717  1 !                       first time flag.
;  396          0718  1 !
;  397          0719  1 ! Implicit Inputs:
;  398          0720  1 !       global data
;  399          0721  1 !
;  400          0722  1 ! Implicit Outputs:
;  401          0723  1 !       global data
;  402          0724  1 !
;  403          0725  1 ! Returned Value:
;  404          0726  1 !       none
;  405          0727  1 !
;  406          0728  1 ! Side Effects:
;  407          0729  1 !
;  408          0730  1 !--
;  409          0731  1
;  410          0732  1
;  411          0733  2 global routine phn$slave_ring_out(ring_smb): novalue = begin
;  412          0734  2
;  413          0735  2 bind
;  414          0736  2         smb_dsc = .ring_smb: descriptor;
;  415          0737  2
;  416          0738  2
;  417          0739  2 local
;  418          0740  2         status: long,
;  419          0741  2         first_ring_flag: byte;
;  420          0742  2
;  421          0743  2
;  422          0744  2 ! Begin by extracting the first time flag from the steering message.
;  423          0745  2
;  424          0746  2 first_ring_flag = ch$rchar(ch$find_ch(.smb_dsc[len],.smb_dsc[ptr],eofrom)+1);
;  425          0747  2
;  426          0748  2 ! Now we can actually ring the person's phone.  Call a routine to do this,
;  427          0749  2 ! and send the resulting status back to the master.
;  428          0750  2
;  429          0751  2 status = phn$local_jangle(..phn$gq_pubhead[0],.first_ring_flag);
;  430          0752  2 phn$write_slave_status(.status);
;  431          0753  2
;  432          0754  2 return;
;  433          0755  2
;  434          0756  1 end;


                                0004 00000          .ENTRY  PHN$SLAVE_RING_OUT, Save R2                     ; 0733
                     50    04   AC  D0 00002          MOVL    RING_SMB, R0                                    ; 0736
```

```
                04   B0          60              00  3A 00006          LOCC    #0, (R0), @4(R0)                ; 0746
                                                 02  12 0000B          BNEQ    1$                              ;
                                                 51  D4 0000D          CLRL    R1                              ;
                             52      01          A1  90 0000F 1$:      MOVB    1(R1), FIRST_RING_FLAG          ;
                             7E                  52  9A 00013          MOVZBL  FIRST_RING_FLAG, -(SP)          ; 0751
                                     0000G       DF  DD 00016          PUSHL   @PHN$GQ_PUBHEAD                 ;
                         0000G   CF              02  FB 0001A          CALLS   #2, PHN$LOCAL_JANGLE            ;
                                                 50  DD 0001F          PUSHL   STATUS                          ; 0752
                         0000V   CF              01  FB 00021          CALLS   #1, PHN$WRITE_SLAVE_STATUS      ;
                                                     04 00026          RET                                     ; 0756

; Routine Size:  39 bytes,    Routine Base:  $CODE$ + 019F
```

```
;  436          0757  1 %sbttl 'PHN$SLAVE_PASS_SMB - Pass SMB on to Process'
;  437          0758  1 !++
;  438          0759  1 ! Functional Description:
;  439          0760  1 !       This steering message routine handles all those steering messages
;  440          0761  1 !       that do not require any special action on the part of the slave.
;  441          0762  1 !       We simply need to pass the SMB on to the target user.
;  442          0763  1 !
;  443          0764  1 ! Formal Parameters:
;  444          0765  1 !       pass_smb        Address of descriptor of SMB.
;  445          0766  1 !
;  446          0767  1 ! Implicit Inputs:
;  447          0768  1 !       global data
;  448          0769  1 !
;  449          0770  1 ! Implicit Outputs:
;  450          0771  1 !       global data
;  451          0772  1 !
;  452          0773  1 ! Returned Value:
;  453          0774  1 !       none
;  454          0775  1 !
;  455          0776  1 ! Side Effects:
;  456          0777  1 !
;  457          0778  1 !--
;  458          0779  1
;  459          0780  1
;  460          0781  2 global routine phn$slave_pass_smb(pass_smb): novalue = begin
;  461          0782  2
;  462          0783  2 bind
;  463          0784  2         smb_dsc = .pass_smb: descriptor;
;  464          0785  2
;  465          0786  2 local
;  466          0787  2         status: long,
;  467          0788  2         char_ptr: long,
;  468          0789  2         tp: ref pub,
;  469          0790  2         mbx_iosb: block[8,byte];
;  470          0791  2
;  471          0792  2
;  472          0793  2 ! We can just send the SMB to the target user via its receive mailbox.
;  473          0794  2 ! We'll do it ourselves, rather than calling PHN$SEND_SMB, so we don't
;  474          0795  2 ! have to split apart the SMB just to have it reassembled.
;  475          0796  2
;  476          0797  2 tp = ..phn$gq_pubhead[0];
;  477        P 0798  2 status = $qiow(chan=.tp[pub_w_channel],
;  478        P 0799  2                func=io$_writevblk + io$m_now,
;  479        P 0800  2                iosb=mbx_iosb,
;  480        P 0801  2                p1=.smb_dsc[ptr],
;  481          0802  2                p2=.smb_dsc[len]);
;  482          0803  2 check (.status);
;  483          0804  2 check (.mbx_iosb[0,0,16,0]);
;  484          0805  2
;  485          0806  2 return;
;  486          0807  2
;  487          0808  1 end;
```

```
                                        0004 00000          .ENTRY  PHN$SLAVE_PASS_SMB, Save R2                 ; 0781
                              52 00000000G  00  9E 00002    MOVAB   LIB$SIGNAL, R2                              ;
                              5E            08  C2 00009    SUBL2   #8, SP                                      ;
                              50        04  AC  D0 0000C    MOVL    PASS_SMB, R0                                ; 0784
                              51      0000G DF  D0 00010    MOVL    @PHN$GQ_PUBHEAD, TP                         ; 0797
                                            7E  7C 00015    CLRQ    -(SP)                                       ; 0802
                                            7E  7C 00017    CLRQ    -(SP)                                       ;
                              7E            60  3C 00019    MOVZWL  (R0), -(SP)                                 ;
                                 04         A0  DD 0001C    PUSHL   4(R0)                                       ;
                                            7E  7C 0001F    CLRQ    -(SP)                                       ;
                                 20         AE  9F 00021    PUSHAB  MBX_IOSB                                    ;
                              7E 70         8F  9A 00024    MOVZBL  #112, -(SP)                                 ;
                              7E   00F4     C1  3C 00028    MOVZWL  244(TP), -(SP)                              ;
                                            7E  D4 0002D    CLRL    -(SP)                                       ;
                    00000000G 00            0C  FB 0002F    CALLS   #12, SYS$QIOW                               ;
                              05            50  E8 00036    BLBS    STATUS, 1$                                  ; 0803
                                            50  DD 00039    PUSHL   STATUS                                      ;
                              62            01  FB 0003B    CALLS   #1, LIB$SIGNAL                              ;
                              06            6E  E8 0003E 1$: BLBS    MBX_IOSB, 2$                                ; 0804
                              7E            6E  3C 00041    MOVZWL  MBX_IOSB, -(SP)                             ;
                              62            01  FB 00044    CALLS   #1, LIB$SIGNAL                              ;
                                            04  00047 2$:    RET                                                 ; 0808

; Routine Size:  72 bytes,    Routine Base:  $CODE$ + 01C6
```

```
;  489          0809  1 %sbttl 'PHN$WRITE_SLAVE_STATUS - Send Status Back to Master'
;  490          0810  1 !++
;  491          0811  1 ! Functional Description:
;  492          0812  1 !       This little routine can be called to network slave routines to send
;  493          0813  1 !       a status back to the master.
;  494          0814  1 !
;  495          0815  1 ! Formal Parameters:
;  496          0816  1 !       phone_status    VMS-format status to send to the master.
;  497          0817  1 !
;  498          0818  1 ! Implicit Inputs:
;  499          0819  1 !       global data
;  500          0820  1 !
;  501          0821  1 ! Implicit Outputs:
;  502          0822  1 !       global data
;  503          0823  1 !
;  504          0824  1 ! Returned Value:
;  505          0825  1 !       none
;  506          0826  1 !
;  507          0827  1 ! Side Effects:
;  508          0828  1 !
;  509          0829  1 !--
;  510          0830  1
;  511          0831  1
;  512          0832  2 global routine phn$write_slave_status(phone_status): novalue = begin
;  513          0833  2
;  514          0834  2 local
;  515          0835  2         universal_status: byte,
;  516          0836  2         status: long;
;  517          0837  2
;  518          0838  2
;  519          0839  2 ! Because we don't want to use VMS-format status codes in our protocol, we
;  520          0840  2 ! must translate the status to a universal protocol status.  This is done
;  521          0841  2 ! by looking it up in the status table and using the index as the universal
;  522          0842  2 ! status.  If we don't find it, just use code 0 (anl$_linkerror).
;  523          0843  2
;  524          0844  2 universal_status = 0;
;  525          0845  2 incru i from 0 to %allocation(status_code_table)/4-1 do
;  526          0846  3         if .phone_status eqlu .status_code_table[.i] then (
;  527          0847  3                 universal_status = .i;
;  528          0848  3 exitloop;
;  529          0849  2         );
;  530          0850  2
;  531          0851  2 ! Write the universal status over the logical link.  Die if we can't do that,
;  532          0852  2 ! and let the master worry about completion of the I/O.
;  533          0853  2
;  534        P 0854  2 status = $qiow(efn=phn$k_decnetefn,
;  535        P 0855  2                chan=.decnet_channel,
;  536        P 0856  2                func=io$_writevblk,
;  537        P 0857  2                p1=universal_status,
;  538          0858  2                p2=1);
;  539          0859  2 check (.status);
;  540          0860  2
;  541          0861  2 return;
;  542          0862  2
;  543          0863  1 end;
```

```
                                  0000 00000          .ENTRY  PHN$WRITE_SLAVE_STATUS, Save nothing        ; 0832
                          5E        04  C2 00002          SUBL2   #4, SP                                      ;
                                    6E  94 00005          CLRB    UNIVERSAL_STATUS                            ; 0844
                                    50  D4 00007          CLRL    I                                           ; 0846
                0000'CF40       04  AC  D1 00009 1$:      CMPL    PHONE_STATUS, STATUS_CODE_TABLE[I]          ;
                                    05  12 00010          BNEQ    2$                                          ;
                          6E        50  90 00012          MOVB    I, UNIVERSAL_STATUS                         ; 0847
                                    07  11 00015          BRB     3$                                          ; 0846
                                    50  D6 00017 2$:      INCL    I                                           ;
                          09        50  D1 00019          CMPL    I, #9                                       ;
                                    EB  1B 0001C          BLEQU   1$                                          ;
                                    7E  7C 0001E 3$:      CLRQ    -(SP)                                       ; 0858
                                    7E  7C 00020          CLRQ    -(SP)                                       ;
                                    01  DD 00022          PUSHL   #1                                          ;
                              14    AE  9F 00024          PUSHAB  UNIVERSAL_STATUS                            ;
                                    7E  7C 00027          CLRQ    -(SP)                                       ;
                          7E        30  7D 00029          MOVQ    #48, -(SP)                                  ;
                          7E  0000' CF  3C 0002C          MOVZWL  DECNET_CHANNEL, -(SP)                       ;
                                    04  DD 00031          PUSHL   #4                                          ;
                00000000G 00        0C  FB 00033          CALLS   #12, SYS$QIOW                               ;
                          09        50  E8 0003A          BLBS    STATUS, 4$                                  ; 0859
                                    50  DD 0003D          PUSHL   STATUS                                      ;
                00000000G 00        01  FB 0003F          CALLS   #1, LIB$SIGNAL                              ;
                                        04 00046 4$:      RET                                                 ; 0863

; Routine Size:  71 bytes,    Routine Base:  $CODE$ + 020E
```

```
;  545          0864  1 %sbttl 'PHN$READ_SLAVE - Read Data from Logical Link'
;  546          0865  1 !++
;  547          0866  1 ! Functional Description:
;  548          0867  1 !       This routine is not actually part of the network slave program, but
;  549          0868  1 !       it belongs in this module.  It is called by the master to read
;  550          0869  1 !       data sent over the logical link by the slave.
;  551          0870  1 !
;  552          0871  1 ! Formal Parameters:
;  553          0872  1 !       channel         The channel number of the logical link.
;  554          0873  1 !       buffer          Address of descriptor of buffer.  We set the length.
;  555          0874  1 !       status_flag     A boolean, true if we are reading a status code.
;  556          0875  1 !
;  557          0876  1 ! Implicit Inputs:
;  558          0877  1 !       global data
;  559          0878  1 !
;  560          0879  1 ! Implicit Outputs:
;  561          0880  1 !       global data
;  562          0881  1 !
;  563          0882  1 ! Returned Value:
;  564          0883  1 !       phn$_remoteerror        Error occured while reading data.
;  565          0884  1 !
;  566          0885  1 ! Side Effects:
;  567          0886  1 !
;  568          0887  1 !--
;  569          0888  1
;  570          0889  1
;  571          0890  2 global routine phn$read_slave(channel,buffer,status_flag) = begin
;  572          0891  2
;  573          0892  2 bind
;  574          0893  2         buffer_dsc = .buffer: descriptor;
;  575          0894  2
;  576          0895  2 local
;  577          0896  2         status: long,
;  578          0897  2         decnet_iosb: block[8,byte];
;  579          0898  2
;  580          0899  2
;  581          0900  2 ! Read the data from the slave.
;  582          0901  2
;  583        P 0902  2 status = $qiow(efn=phn$k_decnetefn,
;  584        P 0903  2                chan=.channel,
;  585        P 0904  2                func=io$_readvblk,
;  586        P 0905  2                iosb=decnet_iosb,
;  587        P 0906  2                p1=.buffer_dsc[ptr],
;  588          0907  2                p2=.buffer_dsc[len]);
;  589          0908  2
;  590          0909  2 ! Make sure the read went OK.  If not, return a failure status.
;  591          0910  2
;  592          0911  2 if not .status or not .decnet_iosb[0,0,16,0] then
;  593          0912  2         return phn$_remoteerror;
;  594          0913  2
;  595          0914  2 ! We have to do different things depending upon whether this is a status
;  596          0915  2 ! we read or just data.
;  597          0916  2
;  598          0917  3 if .status_flag then (
;  599          0918  3
;  600          0919  3         ! We just read a universal status, defined as part of the PHONE
;  601          0920  3         ! protocol.  We want to translate it back to a VMS-format status
```

```
;  602          0921  3          ! using the status code table.
;  603          0922  3
;  604          0923  3          bind
;  605          0924  3                  universal_status = .buffer_dsc[ptr]: byte;
;  606          0925  3
;  607          0926  3          buffer_dsc[len] = 4;
;  608          0927  3          .buffer_dsc[ptr] = .status_code_table[.universal_status];
;  609          0928  3
;  610          0929  2 ) else
;  611          0930  2
;  612          0931  2          ! It's just some data we read.
;  613          0932  2
;  614          0933  2          buffer_dsc[len] = .decnet_iosb[2,0,16,0];
;  615          0934  2
;  616          0935  2 return phn$_ok;
;  617          0936  2
;  618          0937  1 end;


                                        0004 00000          .ENTRY  PHN$READ_SLAVE, Save R2                         ; 0890
                                5E        08 C2 00002       SUBL2   #8, SP                                          ;
                                52    08  AC D0 00005       MOVL    BUFFER, R2                                      ; 0893
                                      7E  7C 00009          CLRQ    -(SP)                                           ; 0907
                                      7E  7C 0000B          CLRQ    -(SP)                                           ;
                                7E        62 3C 0000D       MOVZWL  (R2), -(SP)                                     ;
                                      04  A2 DD 00010       PUSHL   4(R2)                                           ;
                                      7E  7C 00013          CLRQ    -(SP)                                           ;
                                      20  AE 9F 00015       PUSHAB  DECNET_IOSB                                     ;
                                      31  DD 00018          PUSHL   #49                                             ;
                                      04  AC DD 0001A       PUSHL   CHANNEL                                         ;
                                      04  DD 0001D          PUSHL   #4                                              ;
                          00000000G   00    FB 0001F        CALLS   #12, SYS$QIOW                                   ;
                                      03    E9 00026        BLBC    STATUS, 1$                                      ; 0911
                                      08    E8 00029        BLBS    DECNET_IOSB, 2$                                 ;
                                      50 00000000G  8F D0 0002C 1$: MOVL    #PHN$_REMOTEERROR, R0                   ; 0912
                                                04 00033    RET                                                     ;
                                      13    0C  AC E9 00034 2$:    BLBC    STATUS_FLAG, 3$                         ; 0917
                                      50    04  A2 D0 00038        MOVL    4(R2), R0                               ; 0924
                                      62        04 B0 0003C        MOVW    #4, (R2)                                ; 0926
                                      50        60 9A 0003F        MOVZBL  (R0), R0                                ; 0927
                                04    B2  0000'CF40 D0 00042       MOVL    STATUS_CODE_TABLE[R0], @4(R2)           ;
                                                04 11 00049       BRB     4$                                      ; 0917
                                      62    02  AE B0 0004B 3$:    MOVW    DECNET_IOSB+2, (R2)                     ; 0933
                                      50 00000000G  8F D0 0004F 4$: MOVL    #PHN$_OK, R0                            ; 0935
                                                04 00056    RET                                                     ; 0937

; Routine Size:  87 bytes,    Routine Base:  $CODE$ + 0255

;  619          0938  1
;  620          0939  0 end eludom
```

```
                                                .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;  $OWN$                          48  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $PLIT$                         16  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                        684  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                           Library Statistics
;                                       -------- Symbols --------      Pages       Processing
;       File                            Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1        18619        24         0      1000          00:01.4



;                                   COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:NETSLAVE/OBJ=OBJ$:NETSLAVE MSRC$:NETSLAVE/UPDATE=(ENH$:NETSLAVE)

; Size:          684 code + 64 data bytes
; Run Time:          00:12.8
; Elapsed Time:      00:48.2
; Lines/CPU Min:      4401
; Lexemes/CPU-Min:  33792
; Memory Used:  123 pages
; Compilation Complete
```

PHONE
LIS

NETSLAVE
LIS

LINKSUBS
LIS

PHONEMSGS
LIS

STACKCMDS
LIS

TERMINAL
LIS

MISCCMDS
LIS

PUBSUBS
LIS

INPUT
LIS