```
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
```

```
TTTTTTTTTT  EEEEEEEEEE  RRRRRRRR    MM      MM    IIIIII    NN      NN    AAAAAA    LL
TTTTTTTTTT  EEEEEEEEEE  RRRRRRRR    MM      MM    IIIIII    NN      NN    AAAAAA    LL
    TT      EE          RR      RR  MMMM  MMMM      II      NN      NN   AA    AA   LL
    TT      EE          RR      RR  MMMM  MMMM      II      NN      NN   AA    AA   LL
    TT      EE          RR      RR  MM  MM  MM      II      NNNN    NN   AA    AA   LL
    TT      EE          RR      RR  MM  MM  MM      II      NNNN    NN   AA    AA   LL
    TT      EEEEEEEE    RRRRRRRR    MM      MM      II      NN  NN  NN   AA    AA   LL
    TT      EEEEEEEE    RRRRRRRR    MM      MM      II      NN  NN  NN   AA    AA   LL
    TT      EE          RR  RR      MM      MM      II      NN    NNNN   AAAAAAAAAA   LL
    TT      EE          RR  RR      MM      MM      II      NN    NNNN   AAAAAAAAAA   LL
    TT      EE          RR    RR    MM      MM      II      NN      NN   AA    AA   LL              ....
    TT      EE          RR    RR    MM      MM      II      NN      NN   AA    AA   LL              ....
    TT      EEEEEEEEEE  RR      RR  MM      MM    IIIIII    NN      NN   AA    AA   LLLLLLLLLL      ....
    TT      EEEEEEEEEE  RR      RR  MM      MM    IIIIII    NN      NN   AA    AA   LLLLLLLLLL      ....

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 'TERMINAL - Terminal Handling Routines'
;     2          0002  0         module terminal (
;     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, Terminal Handling Routines
;    32          0032  1 !
;    33          0033  1 ! Abstract:     This module provides all of the routines necessary to
;    34          0034  1 !               handle the user's terminal.  This includes both the
;    35          0035  1 !               keystroke handling routines and the screen formatting
;    36          0036  1 !               functions.
;    37          0037  1 !
;    38          0038  1 !
;    39          0039  1 ! Environment:  We do not check the status returns from RTL screen package
;    40          0040  1 !               routines.  This saves code and time.
;    41          0041  1 !
;    42          0042  1 ! Author: Paul C. Anagnostopoulos, Creation Date: 24 November 1980
;    43          0043  1 !
;    44          0044  1 ! Modified By:
;    45          0045  1 !
;    46          0046  1 !       V03-003 PCA1020         Paul C. Anagnostopoulos 24-May-1983
;    47          0047  1 !               Remove !!!TEMP!!! flags.
;    48          0048  1 !               Support 8-bit characters.
;    49          0049  1 !               Check SYS$INPUT to ensure that it is a terminal.
;    50          0050  1 !
;    51          0051  1 !       V03-002 MHB0086         Mark Bramhall           17-Jan-1983
;    52          0052  1 !               Fixed FAO argument check in PHN$INFORM.
;    53          0053  1 !
;    54          0054  1 !       V03-001 PCA0048         Paul Anagnostopoulos    26-Mar-1982
;    55          0055  1 !               Minor changes to convert from process name to user name.
;    56          0056  1 !--
```

```
;    58          0057  1 %sbttl 'Module Declarations'
;    59          0058  1 !
;    60          0059  1 ! Libraries and Requires:
;    61          0060  1 !
;    62          0061  1
;    63          0062  1 library 'sys$library:starlet.l32';
;    64          0063  1 require 'phonereq';
;    65          0392  1
;    66          0393  1 !
;    67          0394  1 ! Table of Contents:
;    68          0395  1 !
;    69          0396  1
;    70          0397  1 forward routine
;    71          0398  1         phn$init_term,
;    72          0399  1         phn$term_characteristic,
;    73          0400  1         phn$kbd_enable: novalue,
;    74          0401  1         phn$kbd_get: novalue,
;    75          0402  1         phn$fresh_screen,
;    76          0403  1         phn$inform: novalue,
;    77          0404  1         phn$review,
;    78          0405  1         phn$fresh_view: novalue,
;    79          0406  1         phn$fresh_hold: novalue,
;    80          0407  1         phn$show_text: novalue,
;    81          0408  1         phn$scroll_prep: novalue,
;    82          0409  1         phn$scroll_line: novalue;
;    83          0410  1
;    84          0411  1 !
;    85          0412  1 ! Macro Definitions:
;    86          0413  1 !
;    87          0414  1 !
;    88          0415  1 ! This macro is used in PHN$SHOW_TEXT to add a new CTL to a PUB.
;    89          0416  1
;    90        M 0417  1 macro add_new_ctl(text) =
;    91        M 0418  1 (       phn$make_ctl(text,c);
;    92        M 0419  1         insque(.c,dp[pub_q_ctl[head0]);
;    93        M 0420  1         inc (dp[pub_l_ctl[count]);
;    94        M 0421  1         latest_line_dsc = c[ctl_q_line];
;    95        M 0422  1 )
;    96          0423  1 %
;    97          0424  1
;    98          0425  1 !
;    99          0426  1 ! External References:
;   100          0427  1 !
;   101          0428  1
;   102          0429  1 external routine
;   103          0430  1         phn$make_ctl,
;   104          0431  1         phn$queue_smb,
;   105          0432  1         scr$erase_line: addressing_mode(general),
;   106          0433  1         scr$erase_page: addressing_mode(general),
;   107          0434  1         scr$put_buffer: addressing_mode(general),
;   108          0435  1         scr$put_line:   addressing_mode(general),
;   109          0436  1         scr$put_screen: addressing_mode(general),
;   110          0437  1         scr$set_buffer: addressing_mode(general),
;   111          0438  1         scr$set_cursor: addressing_mode(general),
;   112          0439  1         scr$set_scroll: addressing_mode(general);
;   113          0440  1
;   114          0441  1 !
```

```
;  115          0442  1 ! Own Variables:
;  116          0443  1 !
;  117          0444  1
;  118          0445  1 ! The following variable contains the terminal channel number for obtaining
;  119          0446  1 ! keystrokes:
;  120          0447  1
;  121          0448  1 own
;  122          0449  1         terminal_channel: word;
;  123          0450  1
;  124          0451  1 ! The following variable is the current position on the command line.
;  125          0452  1
;  126          0453  1 own
;  127          0454  1         command_line_pos: long;
;  128          0455  1
;  129          0456  1 ! We need a line of dashes.  No point in wasting space with lots of them.
;  130          0457  1
;  131          0458  1 bind
;  132          0459  1         dashes = describe('----------------------------------------------------------------------------------------')
```

```
;  134          0460  1 %sbttl 'PHN$INIT_TERM - Initialize Users Terminal'
;  135          0461  1 !++
;  136          0462  1 ! Functional Description:
;  137          0463  1 !       This routine is called at initialization to prepare the user's
;  138          0464  1 !       terminal.  This includes checking for valid terminal type, channel
;  139          0465  1 !       assignment, and initial formatting of the screen.
;  140          0466  1 !
;  141          0467  1 ! Formal Parameters:
;  142          0468  1 !       none
;  143          0469  1 !
;  144          0470  1 ! Implicit Inputs:
;  145          0471  1 !       global data
;  146          0472  1 !
;  147          0473  1 ! Implicit Outputs:
;  148          0474  1 !       global data
;  149          0475  1 !
;  150          0476  1 ! Returned Value:
;  151          0477  1 !       phn$_badterm    Cannot use this type of terminal a  a phone.
;  152          0478  1 !
;  153          0479  1 ! Side Effects:
;  154          0480  1 !
;  155          0481  1 !--
;  156          0482  1
;  157          0483  1
;  158          0484  2 global routine phn$init_term = begin
;  159          0485  2
;  160          0486  2 own
;  161          0487  2         device_class: long;
;  162          0488  2 bind
;  163          0489  2         get_device_class = uplit(word(4),word(dvi$_devclass),
;  164          0490  2                                  long(device_class),
;  165          0491  2                                  long(0),
;  166          0492  2                                  long(0));
;  167          0493  2
;  168          0494  2 local
;  169          0495  2         status: long;
;  170          0496  2 local
;  171          0497  2         local_described_buffer(terminal_number,64);
;  172          0498  2
;  173          0499  2
;  174          0500  2 ! We begin by determining the user's SYS$INPUT device.  We will need to
;  175          0501  2 ! read keystrokes from it.  Make sure it's a device we can use.
;  176          0502  2
;  177        P 0503  2 status = $getdvi(efn=phn$k_getjpiefn,
;  178        P 0504  2                  devnam=describe('SYS$INPUT'),
;  179          0505  2                  itmlst=get_device_class);
;  180          0506  2 check (.status);
;  181          0507  2 $waitfr(efn=phn$k_getjpiefn);
;  182          0508  2 if .device_class nequ dc$_term then
;  183          0509  2         return phn$_inputterm;
;  184          0510  2
;  185        P 0511  2 status = $trnlog(lognam=describe('SYS$INPUT'),
;  186        P 0512  2                  rsllen=terminal_number,
;  187          0513  2                  rslbuf=terminal_number);
;  188          0514  2 check (.status);
;  189          0515  3 if ch$rchar(.terminal_number[ptr]) eglu escape then (
;  190          0516  3         terminal_number[len] = .terminal_number[len] - 4;
```

```
;  191          0517  3          terminal_number[ptr] - .terminal_number[ptr] + 4;
;  192          0518  2 );
;  193          0519  2 if not phn$term_characteristic(terminal_number,tt$m_scope) then
;  194          0520  2          return phn$_badterm;
;  195          0521  2
;  196          0522  2 ! Now we can assign to the user's terminal.  This channel will be used
;  197          0523  2 ! for reading keystrokes, but not for output.
;  198          0524  2
;  199        P 0525  2 status = $assign(devnam=terminal_number,
;  200          0526  2                  chan=terminal_channel);
;  201          0527  2 check (.status);
;  202          0528  2
;  203          0529  2 ! Now we have to clear the half-duplex bit in the terminal characteristics.
;  204          0530  2 ! This is because we have to share the terminal with the RTL screen package.
;  205          0531  2
;  206          0532  3 begin
;  207          0533  3 local
;  208          0534  3          terminal_chars: block[8,byte];
;  209          0535  3 bind
;  210          0536  3          device_info = terminal_chars[4,0,32,0]: block[4,byte];
;  211          0537  3
;  212        P 0538  3 status = $qiow(chan=.terminal_channel,
;  213        P 0539  3                func=io$_sensemode,
;  214          0540  3                p1=terminal_chars);
;  215          0541  3 check (.status);
;  216          0542  3 device_info[tt$v_halfdup] = false;
;  217        P 0543  3 status = $qiow(chan=.terminal_channel,
;  218        P 0544  3                func=io$_setmode,
;  219          0545  3                p1=terminal_chars);
;  220          0546  3 check (.status);
;  221          0547  2 end;
;  222          0548  2
;  223          0549  2 ! Put up the initial screen layout.
;  224          0550  2
;  225          0551  2 phn$fresh_screen(true);
;  226          0552  2
;  227          0553  2 ! Finally, we enable the keyboard so the user can enter a command.
;  228          0554  2
;  229          0555  2 phn$kbd_enable();
;  230          0556  2 return phn$_ok;
;  231          0557  2
;  232          0558  1 end;


                                                                .TITLE  TERMINAL TERMINAL - Terminal Handling Routines
                                                                .IDENT  \V04-000\

                                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  00000 P.AAB:  .ASCII  \----------------------------------------\  ;
2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  0000F                                                          ;
                    2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  0001E                                                          ;
2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  00028          .ASCII  \----------------------------------------\  ;
2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  2D  00037                                                          ;
                        2D  2D  2D  2D  2D  2D  2D  2D  2D  00046                                                          ;
                                                            0004F          .BLKB   1
                                                0000004F    00050 P.AAA:   .LONG   79                                      ;
```

```
                                 00000000'  00054          .ADDRESS P.AAB
                                     0004   00058 P.AAC:   .WORD   4
                                     0004   0005A          .WORD   4
                                 00000000'  0005C          .ADDRESS DEVICE_CLASS
                                 00000000   00060          .LONG   0
                                 00000000   00064          .LONG   0
54  55  50  4E  49  24  53  59  53          00068 P.AAE:   .ASCII  \SYS$INPUT\
                                           00071          .BLKB   3
                                 00000009   00074 P.AAD:   .LONG   9
                                 00000000'  00078          .ADDRESS P.AAE
54  55  50  4E  49  24  53  59  53          0007C P.AAG:   .ASCII  \SYS$INPUT\
                                           00085          .BLKB   3
                                 00000009   00088 P.AAF:   .LONG   9
                                 00000000'  0008C          .ADDRESS P.AAG

                                                          .PSECT  $OWN$,NOEXE,2

                                           00000 TERMINAL_CHANNEL:
                                                          .BLKB   2
                                           00002          .BLKB   2
                                           00004 COMMAND_LINE_POS:
                                                          .BLKB   4
                                           00008 DEVICE_CLASS:
                                                          .BLKB   4

                                                 DASHES=              P.AAA
                                                 GET_DEVICE_CLASS=    P.AAC
                                                          .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$MAKE_CTL
                                                          .EXTRN  PHN$QUEUE_SMB, SCR$ERASE_LINE
```

```
                                                   .EXTRN  SCR$ERASE_PAGE, SCR$PUT_BUFFER
                                                   .EXTRN  SCR$PUT_LINE, SCR$PUT_SCREEN
                                                   .EXTRN  SCR$SET_BUFFER, SCR$SET_CURSOR
                                                   .EXTRN  SCR$SET_SCROLL, SYS$GETDVI
                                                   .EXTRN  SYS$WAITFR, SYS$TRNLOG
                                                   .EXTRN  SYS$ASSIGN, SYS$QIOW

                                                   .PSECT  $CODE$,NOWRT,2

                              003C 00000           .ENTRY  PHN$INIT_TERM, Save R2,R3,R4,R5             ; 0484
            55 00000000G  00    9E 00002           MOVAB   SYS$QIOW, R5                                ;
            54     0000'  CF    9E 00009           MOVAB   TERMINAL_CHANNEL, R4                        ;
            53 00000000G  00    9E 0000E           MOVAB   LIB$SIGNAL, R3                              ;
            5E        B0  AE    9E 00015           MOVAB   -80(SP), SP                                 ;
     08     AE        40  8F    9A 00019           MOVZBL  #64, TERMINAL_NUMBER                        ; 0497
     0C     AE        10  AE    9E 0001E           MOVAB   TERMINAL_NUMBER+8, TERMINAL_NUMBER+4        ;
                          7E    7C 00023           CLRQ    -(SP)                                       ; 0505
                          7E    7C 00025           CLRQ    -(SP)                                       ;
                   0000'  CF    9F 00027           PUSHAB  GET_DEVICE_CLASS                            ;
                   0000'  CF    9F 0002B           PUSHAB  P.AAD                                       ;
            7E            01    7D 0002F           MOVQ    #1, -(SP)                                   ;
 00000000G  00            08    FB 00032           CALLS   #8, SYS$GETDVI                              ;
            52            50    D0 00039           MOVL    R0, STATUS                                  ;
            05            52    E8 0003C           BLBS    STATUS, 1$                                  ; 0506
                          52    DD 0003F           PUSHL   STATUS                                      ;
            63            01    FB 00041           CALLS   #1, LIB$SIGNAL                              ;
                          01    DD 00044 1$:       PUSHL   #1                                          ; 0507
 00000000G  00            01    FB 00046           CALLS   #1, SYS$WAITFR                              ;
 00000042   8F        08  A4    D1 0004D           CMPL    DEVICE_CLASS, #66                           ; 0508
                          08    13 00055           BEQL    2$                                          ;
            50 00000000G  8F    D0 00057           MOVL    #PHN$_INPUTTERM, R0                         ; 0509
                                04 0005E           RET                                                 ;
                          7E    7C 0005F 2$:       CLRQ    -(SP)                                       ; 0513
                          7E    D4 00061           CLRL    -(SP)                                       ;
                      14  AE    9F 00063           PUSHAB  TERMINAL_NUMBER                             ;
                      18  AE    9F 00066           PUSHAB  TERMINAL_NUMBER                             ;
                   0000'  CF    9F 00069           PUSHAB  P.AAF                                       ;
 00000000G  00            06    FB 0006D           CALLS   #6, SYS$TRNLOG                              ;
            52            50    D0 00074           MOVL    R0, STATUS                                  ;
            05            52    E8 00077           BLBS    STATUS, 3$                                  ; 0514
                          52    DD 0007A           PUSHL   STATUS                                      ;
            63            01    FB 0007C           CALLS   #1, LIB$SIGNAL                              ;
            1B        0C  BE    91 0007F 3$:       CMPB    @TERMINAL_NUMBER+4, #27                     ; 0515
                          08    12 00083           BNEQ    4$                                          ;
     08     AE            04    A2 00085           SUBW2   #4, TERMINAL_NUMBER                         ; 0516
     0C     AE            04    C0 00089           ADDL2   #4, TERMINAL_NUMBER+4                       ; 0517
            7E      1000  8F    3C 0008D 4$:       MOVZWL  #4096, -(SP)                                ; 0519
                      0C  AE    9F 00092           PUSHAB  TERMINAL_NUMBER                             ;
      0000V CF            02    FB 00095           CALLS   #2, PHN$TERM_CHARACTERISTIC                 ;
            08            50    E8 0009A           BLBS    R0, 5$                                      ;
            50 00000000G  8F    D0 0009D           MOVL    #PHN$_BADTERM, R0                           ; 0520
                                04 000A4           RET                                                 ;
                          7E    7C 000A5 5$:       CLRQ    -(SP)                                       ; 0526
                          54    DD 000A7           PUSHL   R4                                          ;
                      14  AE    9F 000A9           PUSHAB  TERMINAL_NUMBER                             ;
 00000000G  00            04    FB 000AC           CALLS   #4, SYS$ASSIGN                              ;
            52            50    D0 000B3           MOVL    R0, STATUS                                  ;
```

```
                        05              52  E8 000B6          BLBS    STATUS, 6$                              ; 0527
                                        52  DD 000B9          PUSHL   STATUS                                  ;
                        63              01  FB 000BB          CALLS   #1, LIB$SIGNAL                          ;
                                        7E  7C 000BE 6$:      CLRQ    -(SP)                                   ; 0540
                                        7E  7C 000C0          CLRQ    -(SP)                                   ;
                                        7E  D4 000C2          CLRL    -(SP)                                   ;
                                    14  AE  9F 000C4          PUSHAB  TERMINAL_CHARS                          ;
                                        7E  7C 000C7          CLRQ    -(SP)                                   ;
                        7E              27  7D 000C9          MOVQ    #39, -(SP)                              ;
                        7E              64  3C 000CC          MOVZWL  TERMINAL_CHANNEL, -(SP)                 ;
                                        7E  D4 000CF          CLRL    -(SP)                                   ;
                        65              0C  FB 000D1          CALLS   #12, SYS$QIOW                           ;
                        52              50  D0 000D4          MOVL    R0, STATUS                              ;
                        05              52  E8 000D7          BLBS    STATUS, 7$                              ; 0541
                                        52  DD 000DA          PUSHL   STATUS                                  ;
                        63              01  FB 000DC          CALLS   #1, LIB$SIGNAL                          ;
                    06  AE              10  8A 000DF 7$:      BICB2   #16, DEVICE_INFO+2                      ; 0542
                                        7E  7C 000E3          CLRQ    -(SP)                                   ; 0545
                                        7E  7C 000E5          CLRQ    -(SP)                                   ;
                                        7E  D4 000E7          CLRL    -(SP)                                   ;
                                    14  AE  9F 000E9          PUSHAB  TERMINAL_CHARS                          ;
                                        7E  7C 000EC          CLRQ    -(SP)                                   ;
                        7E              23  7D 000EE          MOVQ    #35, -(SP)                              ;
                        7E              64  3C 000F1          MOVZWL  TERMINAL_CHANNEL, -(SP)                 ;
                                        7E  D4 000F4          CLRL    -(SP)                                   ;
                        65              0C  FB 000F6          CALLS   #12, SYS$QIOW                           ;
                        52              50  D0 000F9          MOVL    R0, STATUS                              ;
                        05              52  E8 000FC          BLBS    STATUS, 8$                              ; 0546
                                        52  DD 000FF          PUSHL   STATUS                                  ;
                        63              01  FB 00101          CALLS   #1, LIB$SIGNAL                          ;
                                        01  DD 00104 8$:      PUSHL   #1                                      ; 0551
                0000V   CF              01  FB 00106          CALLS   #1, PHN$FRESH_SCREEN                    ;
                0000V   CF              00  FB 0010B          CALLS   #0, PHN$KBD_ENABLE                      ; 0555
                        50 00000000G    8F  D0 00110          MOVL    #PHN$_OK, R0                            ; 0556
                                            04 00117          RET                                             ; 0558

; Routine Size:  280 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  234          0559  1 %sbttl 'PHN$TERM_CHARACTERISTIC - Get a Terminal Characteristic'
;  235          0560  1 !++
;  236          0561  1 ! Functional Description:
;  237          0562  1 !       This routine is called to get one of the characteristic bits
;  238          0563  1 !       for the specified terminal.  These bits specify various features
;  239          0564  1 !       of the terminal.
;  240          0565  1 !
;  241          0566  1 ! Formal Parameters:
;  242          0567  1 !       terminal_number         Address of descriptor of terminal number.
;  243          0568  1 !       mask                    Mask for characteristic bit.
;  244          0569  1 !
;  245          0570  1 ! Implicit Inputs:
;  246          0571  1 !       global data
;  247          0572  1 !
;  248          0573  1 ! Implicit Outputs:
;  249          0574  1 !       global data
;  250          0575  1 !
;  251          0576  1 ! Returned Value:
;  252          0577  1 !       True if characteristic bit is set, false if clear.
;  253          0578  1 !
;  254          0579  1 ! Side Effects:
;  255          0580  1 !
;  256          0581  1 !--
;  257          0582  1
;  258          0583  1
;  259          0584  2 global routine phn$term_characteristic(terminal_number,mask) = begin
;  260          0585  2
;  261          0586  2 local
;  262          0587  2         status: long;
;  263          0588  2 local
;  264          0589  2         local_described_buffer(dib_buffer,12);
;  265          0590  2
;  266          0591  2
;  267          0592  2 ! First we obtain the device information in a buffer long enough to contain
;  268          0593  2 ! the device dependent characteristics.
;  269          0594  2
;  270        P 0595  2 status = $getdev(devnam=.terminal_number,
;  271          0596  2                  pribuf=dib_buffer);
;  272          0597  2 if .status nequ ss$_nonlocal then
;  273          0598  2         check (.status);
;  274          0599  2
;  275          0600  2 ! Now we can check the requested bit and return true or false.
;  276          0601  2
;  277          0602  3 begin
;  278          0603  3 bind
;  279          0604  3         dib = .dib_buffer[ptr]: block[,byte];
;  280          0605  3
;  281          0606  3 return (.dib[dib$l_devdepend] and .mask) nequ 0;
;  282          0607  3
;  283          0608  2 end;
;  284          0609  1 end;                                                                    ;


                                                        .EXTRN  SYS$GETDEV

                                0000 00000              .ENTRY  PHN$TERM_CHARACTERISTIC, Save nothing       ; 0584
```

```
                                5E              10  C2 00002          SUBL2   #16, SP                                    ;
                                                0C  DD 00005          PUSHL   #12                                     ; 0589
                          04    AE        08    AE  9E 00007          MOVAB   DIB_BUFFER+8, DIB_BUFFER+4
                                                7E  7C 0000C          CLRQ    -(SP)                                   ; 0596
                                            08  AE  9F 0000E          PUSHAB  DIB_BUFFER
                                                7E  D4 00011          CLRL    -(SP)                                   ;
                                            04  AC  DD 00013          PUSHL   TERMINAL_NUMBER                         ;
                    00000000G   00                  05  FB 00016          CALLS   #5, SYS$GETDEV                      ;
                    000008F0    8F                  50  D1 0001D          CMPL    STATUS, #2288                       ; 0597
                                                    0C  13 00024          BEQL    1$                                  ;
                                09                  50  E8 00026          BLBS    STATUS, 1$                          ; 0598
                                                    50  DD 00029          PUSHL   STATUS                              ;
                    00000000G   00                  01  FB 0002B          CALLS   #1, LIB$SIGNAL                      ;
                                50          04  AE  D0 00032  1$:         MOVL    DIB_BUFFER+4, R0                    ; 0604
                                                    51  D4 00036          CLRL    R1                                  ; 0606
                          08    AC          08  A0  D3 00038              BITL    8(R0), MASK                         ;
                                                    02  13 0003D          BEQL    2$                                  ;
                                                    51  D6 0003F          INCL    R1                                  ;
                                50                  51  D0 00041  2$:     MOVL    R1, R0                              ;
                                                        04 00044          RET                                         ; 0609

. Routine Size:  69 bytes,    Routine Base:  $CODE$ + 0118
```

```
;  286          0610  1 %sbttl 'PHN$KBD_ENABLE - Enable Keyboard for Input'
;  287          0611  1 !++
;  288          0612  1 ! Functional Description:
;  289          0613  1 !       This routine is called to enable the keyboard to interrupt
;  290          0614  1 !       us with input.  We use an AST to inform us of pending input.
;  291          0615  1 !       The AST routine sets up to read the input.
;  292          0616  1 !
;  293          0617  1 ! Formal Parameters:
;  294          0618  1 !       none
;  295          0619  1 !
;  296          0620  1 ! Implicit Inputs:
;  297          0621  1 !       global data
;  298          0622  1 !
;  299          0623  1 ! Implicit Outputs:
;  300          0624  1 !       global data
;  301          0625  1 !
;  302          0626  1 ! Returned Value:
;  303          0627  1 !       none
;  304          0628  1 !
;  305          0629  1 ! Side Effects:
;  306          0630  1 !
;  307          0631  1 !--
;  308          0632  1
;  309          0633  1
;  310          0634  2 global routine phn$kbd_enable: novalue = begin
;  311          0635  2
;  312          0636  2 own
;  313          0637  2         terminal_iosb: block[8,byte],
;  314          0638  2         own_described_buffer(single_key,1);
;  315          0639  2
;  316          0640  2 local
;  317          0641  2         status;
;  318          0642  2
;  319          0643  2
;  320          0644  2 ! The AST routine reads the pending keystroke and passes it along to a
;  321          0645  2 ! steering message routine.  The steering routine then flushes all
;  322          0646  2 ! outstanding keystrokes at the time it is called,  thus causing us to
;  323          0647  2 ! "batch" the keystrokes.  It is hoped that the delay caused by this scheme
;  324          0648  2 ! will be proportional to the system load.
;  325          0649  2
;  326          0650  3 routine kbd_ast: novalue = begin
;  327          0651  3
;  328          0652  3 check (.terminal_iosb[0,0,16,0]);
;  329          0653  3
;  330          0654  3 phn$queue_smb(smb__kbd_get,single_key);
;  331          0655  3 return;
;  332          0656  3
;  333          0657  2 end;


                                        .PSECT  $OWN$,NOEXE,2

                                0000C TERMINAL_IOSB:
                                        .BLKB   8
                      00000001  00014 SINGLE_KEY:
                                        .LONG   1                                       ;
```

```
                                00000000'  00018          .ADDRESS SINGLE_KEY+8                                   ;
                                           0001C          .BLKB   1


                                                          .PSECT  $CODE$,NOWRT,2

                                      0000 00000 KBD_AST:.WORD   Save nothing                                    ; 0650
                          0C    0000'  CF  E8 00002        BLBS    TERMINAL_IOSB, 1$                             ; 0652
                          7E    0000'  CF  3C 00007        MOVZWL  TERMINAL_IOSB, -(SP)                          ;
                00000000G   00         01  FB 0000C        CALLS   #1, LIB$SIGNAL                                ;
                                0000'  CF  9F 00013 1$:    PUSHAB  SINGLE_KEY                                    ; 0654
                                       01  DD 00017        PUSHL   #1                                            ;
                    0000G   CF         02  FB 00019        CALLS   #2, PHN$QUEUE_SMB                             ;
                                       04  0001E          RET                                                   ; 0657

; Routine Size:  31 bytes,    Routine Base:  $CODE$ + 015D
```

```
;  335          0658  2 ! To enable the keyboard, we do a read for a single keystroke, with AST
;  336          0659  2 ! to be delivered when it comes in.  Note that we specify no terminators.
;  337          0660  2
;  338        P 0661  2 status = $qio(efn=phn$k_kbdefn,
;  339        P 0662  2               chan=.terminal_channel,
;  340        P 0663  2               func=io$_readvblk + io$m_noecho + io$m_nofiltr,
;  341        P 0664  2               iosb=terminal_iosb,
;  342        P 0665  2               astadr=kbd_ast,
;  343        P 0666  2               p1=single_key+8,
;  344        P 0667  2               p2=1,
;  345          0668  2               p3=uplit long(0,0));
;  346          0669  2 check (.status);
;  347          0670  2
;  348          0671  2 return;
;  349          0672  2
;  350          0673  1 end;


                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

                                00000000  00000000  00090 P.AAH:  .LONG   0, 0                                          ;

                                                .EXTRN  SYS$QIO

                                                .PSECT  $CODE$,NOWRT,2

                                          0000 00000            .ENTRY  PHN$KBD_ENABLE, Save nothing                    ; 0634
                                       7E  7C 00002            CLRQ    -(SP)                                           ; 0668
                                       7E  D4 00004            CLRL    -(SP)                                           ;
                                  0000'  CF  9F 00006            PUSHAB  P.AAH                                           ;
                                       01  DD 0000A            PUSHL   #1                                              ;
                                  0000'  CF  9F 0000C            PUSHAB  SINGLE_KEY+8                                    ;
                                       7E  D4 00010            CLRL    -(SP)                                           ;
                                       CC  AF 9F 00012         PUSHAB  KBD_AST                                         ;
                                  0000'  CF  9F 00015            PUSHAB  TERMINAL_IOSB                                   ;
                                7E     0271  8F 3C 00019        MOVZWL  #625, -(SP)                                     ;
                                7E     0000' CF 3C 0001E        MOVZWL  TERMINAL_CHANNEL, -(SP)                         ;
                                       02  DD 00023            PUSHL   #2                                              ;
                          00000000G    00  0C FB 00025        CALLS   #12, SYS$QIO                                    ;
                                       09  50 E8 0002C         BLBS    STATUS, 1$                                      ; 0669
                                       50  DD 0002F            PUSHL   STATUS                                          ;
                          00000000G    00  01 FB 00031        CALLS   #1, LIB$SIGNAL                                  ;
                                           04 00038 1$:        RET                                                     ; 0673

; Routine Size:  57 bytes,    Routine Base:  $CODE$ + 017C
```

```
;  352          0674  1 %sbttl 'PHN$KBD_GET - Get All Outstanding Keystrokes'
;  353          0675  1 !++
;  354          0676  1 ! Functional Description:
;  355          0677  1 !       This steering message routine is invoked when the user has typed
;  356          0678  1 !       a key at the terminal.  It collects any additional keystrokes
;  357          0679  1 !       from the typeahead buffer and queues them in a batch to the
;  358          0680  1 !       input interpreter.
;  359          0681  1 !
;  360          0682  1 ! Formal Parameters:
;  361          0683  1 !       single_key      The address of a descriptor of the single keystroke
;  362          0684  1 !                       that started this mess.
;  363          0685  1 !
;  364          0686  1 ! Implicit Inputs:
;  365          0687  1 !       global data
;  366          0688  1 !
;  367          0689  1 ! Implicit Outputs:
;  368          0690  1 !       global data
;  369          0691  1 !
;  370          0692  1 ! Returned Value:
;  371          0693  1 !       none
;  372          0694  1 !
;  373          0695  1 ! Side Effects:
;  374          0696  1 !
;  375          0697  1 !--
;  376          0698  1
;  377          0699  1
;  378          0700  2 global routine phn$kbd_get(single_key): novalue = begin
;  379          0701  2
;  380          0702  2 bind
;  381          0703  2         single_key_dsc = .single_key: descriptor;
;  382          0704  2
;  383          0705  2 local
;  384          0706  2         status: long,
;  385          0707  2         terminal_iosb: block[8,byte];
;  386          0708  2 local
;  387          0709  2         local_described_buffer(input,80);
;  388          0710  2
;  389          0711  2
;  390          0712  2 ! We begin by moving the single key that was passed to us into our input
;  391          0713  2 ! buffer.
;  392          0714  2
;  393          0715  2 ch$move(1,.single_key_dsc[ptr],.input[ptr]);
;  394          0716  2
;  395          0717  2 ! Now we can flush the typeahead buffer and add it to our input buffer,
;  396          0718  2 ! forming a batch of characters.  NOTE that we specify no terminators.
;  397          0719  2
;  398        P 0720  2 status = $qiow(efn=phn$k_kbdefn,
;  399        P 0721  2                chan=.terminal_channel,
;  400        P 0722  2                func=io$_readvblk + io$m_noecho + io$m_nofiltr + io$m_timed,
;  401        P 0723  2                iosb=terminal_iosb,
;  402        P 0724  2                p1=.input[ptr]+1,
;  403        P 0725  2                p2=79,
;  404        P 0726  2                p3=0,
;  405          0727  2                p4=uplit long(0,0));
;  406          0728  2 check (.status);
;  407          0729  2 if .terminal_iosb[0,0,16,0] nequ ss$_timeout then
;  408          0730  2         check (.terminal_iosb[0,0,16,0]);
```

```
;  409          0731  2
;  410          0732  2 ! Fill in the character count in the input buffer and queue a steering
;  411          0733  2 ! message to route the input.
;  412          0734  2
;  413          0735  2 input[len] = 1 + .terminal_iosb[2,0,16,0];
;  414          0736  2 phn$queue_smb(smb__kbd_route,input);
;  415          0737  2
;  416          0738  2 ! Re-enable the keyboard for further input.
;  417          0739  2
;  418          0740  2 phn$kbd_enable();
;  419          0741  2 return;
;  420          0742  2
;  421          0743  1 end;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                                00000000  00000000  00098 P.AAI:  .LONG   0, 0                                          ;


                                                        .PSECT  $CODE$,NOWRT,2

                                            0004 00000          .ENTRY  PHN$KBD_GET, Save R2                            ; 0700
                                    52 00000000G   00    9E 00002  MOVAB   LIB$SIGNAL, R2                              ;
                                    5E       A4    AE    9E 00009  MOVAB   -92(SP), SP                                 ;
                                    50       04    AC    D0 0000D  MOVL    SINGLE_KEY, R0                              ; 0703
                                    7E       50    8F    9A 00011  MOVZBL  #80, INPUT                                  ; 0709
                              04    AE       08    AE    9E 00015  MOVAB   INPUT+8, INPUT+4                            ;
                              04    BE       04    B0    90 0001A  MOVB    @4(R0), @INPUT+4                            ; 0715
                        50    04    AE             01    C1 0001F  ADDL3   #1, INPUT+4, R0                             ;
                                                   7E    7C 00024  CLRQ    -(SP)                                       ; 0727
                                          0000'    CF    9F 00026  PUSHAB  P.AAI                                       ;
                                                   7E    D4 0002A  CLRL    -(SP)                                       ;
                                          7E       4F    8F    9A 0002C  MOVZBL  #79, -(SP)                            ;
                                                   50    DD 00030  PUSHL   R0                                          ;
                                                   7E    7C 00032  CLRQ    -(SP)                                       ;
                                             78    AE    9F 00034  PUSHAB  TERMINAL_IOSB                               ;
                                          7E       02F1  8F    3C 00037  MOVZWL  #753, -(SP)                           ;
                                          7E       0000' CF    3C 0003C  MOVZWL  TERMINAL_CHANNEL, -(SP)               ;
                                                   02    DD 00041  PUSHL   #2                                          ;
                           000000000G     00             0C    FB 00043  CALLS   #12, SYS$QIOW                       ;
                                          05             50    E8 0004A  BLBS    STATUS, 1$                          ; 0728
                                                         50    D0 0004D  PUSHL   STATUS                              ;
                                          62             01    FB 0004F  CALLS   #1, LIB$SIGNAL                      ;
                                022C      8F       58    AE    B1 00052  1$:     CMPW    TERMINAL_IOSB, #556          ; 0729
                                                   0B    13 00058  BEQL    2$                                          ;
                                          07       58    AE    E8 0005A  BLBS    TERMINAL_IOSB, 2$                   ; 0730
                                          7E       58    AE    3C 0005E  MOVZWL  TERMINAL_IOSB, -(SP)                  ;
                                          62             01    FB 00062  CALLS   #1, LIB$SIGNAL                      ;
                              6E    5A    AE             01    A1 00065  2$:     ADDW3   #1, TERMINAL_IOSB+2, INPUT   ; 0735
                                                   5E    DD 0006A  PUSHL   SP                                          ; 0736
                                                   02    DD 0006C  PUSHL   #2                                          ;
                                    0000G  CF              02    FB 0006E  CALLS   #2, PHN$QUEUE_SMB                   ;
                                    FF4F   CF              00    FB 00073  CALLS   #0, PHN$KBD_ENABLE                  ; 0740
                                                           04 00078  RET                                               ; 0743
```

; Routine Size: 121 bytes,   Routine Base: $CODE$ + 01B5

```
;  423     0744  1 %sbttl 'PHN$FRESH_SCREEN - Refresh the S reen'
;  424     0745  1 !++
;  425     0746  1 ! Functional Description:
;  426     0747  1 !       This routine is called to refresh part or all of the screen, as
;  427     0748  1 !       specified by the parameter.
;  428     0749  1 !
;  429     0750  1 ! Formal Parameters:
;  430     0751  1 !       top_too         If false, we refresh the viewports.  If true, we
;  431     0752  1 !                       also refresh the top four lines.
;  432     0753  1 !
;  433     0754  1 ! Implicit Inputs:
;  434     0755  1 !       global data
;  435     0756  1 !
;  436     0757  1 ! Implicit Outputs:
;  437     0758  1 !       global data
;  438     0759  1 !
;  439     0760  1 ! Returned Value:
;  440     0761  1 !       phn$_noports    Too many viewports required for active people.
;  441     0762  1 !
;  442     0763  1 ! Side Effects:
;  443     0764  1 !
;  444     0765  1 !--
;  445     0766  1
;  446     0767  1
;  447     0768  2 global routine phn$fresh_screen(top_too) = begin
;  448     0769  2
;  449     0770  2 local
;  450     0771  2         status: long,
;  451     0772  2         p: ref pub;
;  452     0773  2
;  453     0774  2
;  454     0775  2 ! If a scrolling-type command is in progress (e.g., HELP), then we can't
;  455     0776  2 ! refresh or we'll clobber whatever is being scrolled.
;  456     0777  2
;  457     0778  2 if .phn$gv_scroller then
;  458     0779  2         return phn$_ok;
;  459     0780  2
;  460     0781  2 ! We begin by recalculating the viewport parameters to assign all active
;  461     0782  2 ! participants a viewport.  If this cannot be done, we return an error
;  462     0783  2 ! status without touching the screen.
;  463     0784  2
;  464     0785  2 status = phn$review();
;  465     0786  2 if .status nequ phn$_ok then
;  466     0787  2         return .status;
;  467     0788  2
;  468     0789  2 ! If we are to refresh the top four lines of the screen, then do it.
;  469     0790  2 ! This includes a heading, today's date, the switch hook character on the
;  470     0791  2 ! command line, and a line of dashes.
;  471     0792  2
;  472     0793  3 if .top_too then (
;  473     0794  3         scr$erase_page(1,1);
;  474     0795  3
;  475     0796  3         scr$put_screen(describe(' VAX/VMS Phone Facility '),1,28,%b'0010');
;  476     0797  3
;  477     0798  4         begin
;  478     0799  4         local
;  479     0800  4                 local_described_buffer(todays_date,11);
```

```
4
4               $asctim(timbuf=todays_date);
4               scr$put_screen(todays_date,1,69);
3               end;
3
3               scr$put_screen(phn$gq_switch_hook,2,1);
3               command_line_pos = 2;
3
3               scr$put_screen(dashes,4,1);
2 );
2
2 ! Now we can display the viewports.  We scan the PUB chain and find each
2 ! PUB that has a viewport assigned.  These viewports are displayed.
2
2 scr$erase_page(5,1);
2
2 p = .phn$gq_pubhead[0];
3 until .p eqla phn$gq_pubhead do (
3               if .p[pub_w_viewsize] nequ 0 then
3                       phn$fresh_view(.p,true,true);
3
3               p = .p[pub_l_flink];
2 );
2
2 return phn$_ok;
2
1 end;
```

```
                                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

20 65 6E 6F 68 50 20 53 4D 56 2F 58 41 56 20 000A0 P.AAK:      .ASCII  \ VAX/VMS Phone Facility \        ;
                  20 79 74 69 6C 69 63 61 46 000AF                                                  ;
                                   00000018 000B8 P.AAJ:       .LONG   24                            ;
                                   00000000' 000BC             .ADDRESS P.AAK                        ;

                                                                .EXTRN  SYS$ASCTIM

                                                                .PSECT  $CODE$,NOWRT,2

                                        003C 00000              .ENTRY  PHN$FRESH_SCREEN, Save R2,R3,R4,R5   ; 0768
                        55 00000000G 00  9E 00002              MOVAB   SCR$ERASE_PAGE, R5                   ;
                        54 00000000G 8F  D0 00009              MOVL    #PHN$_OK, R4                         ;
                        53 00000000G 00  9E 00010              MOVAB   SCR$PUT_SCREEN, R3                   ;
                        5E           14  C2 00017              SUBL2   #20, SP                              ;
           03     0000G CF           01  E1 0001A              BBC     #1, PHN$GB_FLAGS, 1$                 ; 0778
                                   008C  31 00020              BRW     6$                                   ;
                  0000V CF           00  FB 00023 1$:          CALLS   #0, PHN$REVIEW                       ; 0785
                        54           50  D1 00028              CMPL    STATUS, R4                           ; 0786
                                     01  13 0002B              BEQL    2$                                   ;
                                     04  00002D                RET                                          ;
                        51     04    AC  E9 0002E 2$:          BLBC    TOP_TOO, 3$                          ; 0793
                                     01  DD 00032              PUSHL   #1                                   ; 0794
                                     01  DD 00034              PUSHL   #1                                   ;
                        65           02  FB 00036              CALLS   #2, SCR$ERASE_PAGE                   ;
                                     02  DD 00039              PUSHL   #2                                   ; 0796
```

```
                                 1C  DD 0003B          PUSHL   #28                                                     ;
                                 01  DD 0003D          PUSHL   #1                                                      ;
                           0000' CF  9F 0003F          PUSHAB  P.AAJ                                                   ;
                              63     04 FB 00043          CALLS   #4, SCR$PUT_SCREEN                                      ;
                              6E     0B D0 00046          MOVL    #11, TODAYS_DATE                                        ; 0800
                           04 AE     08 AE 9E 00049       MOVAB   TODAYS_DATE+8, TODAYS_DATE+4                            ;
                                    7E 7C 0004E          CLRQ    -(SP)                                                   ; 0802
                              08 AE     9F 00050          PUSHAB  TODAYS_DATE                                             ;
                                    7E D4 00053          CLRL    -(SP)                                                   ;
                  00000000G   00     04 FB 00055          CALLS   #4, SYS$ASCTIM                                          ;
                              7E     45 8F 9A 0005C       MOVZBL  #69, -(SP)                                              ; 0803
                                 01  DD 00060          PUSHL   #1                                                      ;
                              08 AE     9F 00062          PUSHAB  TODAYS_DATE                                             ;
                              63     03 FB 00065          CALLS   #3, SCR$PUT_SCREEN                                      ;
                                 01  DD 00068          PUSHL   #1                                                      ; 0806
                                 02  DD 0006A          PUSHL   #2                                                      ;
                           0000G CF  9F 0006C          PUSHAB  PHN$GQ_SWITCH_HOOK                                      ;
                              63     03 FB 00070          CALLS   #3, SCR$PUT_SCREEN                                      ;
                        0000' CF     02 D0 00073          MOVL    #2, COMMAND_LINE_POS                                    ; 0807
                                 01  DD 00078          PUSHL   #1                                                      ; 0809
                                 04  DD 0007A          PUSHL   #4                                                      ;
                           0000' CF  9F 0007C          PUSHAB  DASHES                                                  ;
                              63     03 FB 00080          CALLS   #3, SCR$PUT_SCREEN                                      ;
                                 01  DD 00083 3$:       PUSHL   #1                                                      ; 0815
                                 05  DD 00085          PUSHL   #5                                                      ;
                              65     02 FB 00087          CALLS   #2, SCR$ERASE_PAGE                                      ;
                              52     0000G CF D0 0008A    MOVL    PHN$GQ_PUBHEAD, P                                       ; 0817
                              50     0000G CF 9E 0008F 4$:    MOVAB   PHN$GQ_PUBHEAD, R0                                      ; 0818
                              50        52 D1 00094       CMPL    P, R0                                                   ;
                                    16 13 00097          BEQL    6$                                                      ;
                                 00FC C2  B5 00099       TSTW    252(P)                                                  ; 0819
                                    0B 13 0009D          BEQL    5$                                                      ;
                                 01  DD 0009F          PUSHL   #1                                                      ; 0820
                                 01  DD 000A1          PUSHL   #1                                                      ;
                                 52  DD 000A3          PUSHL   P                                                       ;
                        0000V CF     03 FB 000A5          CALLS   #3, PHN$FRESH_VIEW                                      ;
                              52     62 D0 000AA 5$:      MOVL    (P), P                                                  ; 0822
                                    E0 11 000AD          BRB     4$                                                      ; 0818
                              50     54 D0 000AF 6$:      MOVL    R4, R0                                                  ; 0825
                                    04 000B2          RET                                                             ; 0827

; Routine Size:  179 bytes,    Routine Base:  $CODE$ + 022E
```

```
;  508          0828  1 %sbttl 'PHN$INFORM - Display Message to User'
;  509          0829  1 !++
;  510          0830  1 ! Functional Description:
;  511          0831  1 !       This routine is called to display a message to the user.  All messages
;  512          0832  1 !       are defined in the message file and referenced by status codes.
;  513          0833  1 !       $FAO arguments can also be passed, and they will be filled into
;  514          0834  1 !       the message text.
;  515          0835  1 !
;  516          0836  1 ! Formal Parameters:
;  517          0837  1 !       status_code     The message status code.  Zero means to clear the line.
;  518          0838  1 !       fao1...         Zero or more $FAO arguments to be filled in.
;  519          0839  1 !
;  520          0840  1 ! Implicit Inputs:
;  521          0841  1 !       global data
;  522          0842  1 !
;  523          0843  1 ! Implicit Outputs:
;  524          0844  1 !       global data
;  525          0845  1 !
;  526          0846  1 ! Returned Value:
;  527          0847  1 !       none
;  528          0848  1 !
;  529          0849  1 ! Side Effects:
;  530          0850  1 !
;  531          0851  1 !--
;  532          0852  1
;  533          0853  1
;  534          0854  2 global routine phn$inform(status_code,fao1): novalue = begin
;  535          0855  2
;  536          0856  2 local
;  537          0857  2         status: long,
;  538          0858  2         byte_array: vector[4,byte],
;  539          0859  2         msg_text_ptr: ref descriptor;
;  540          0860  2
;  541          0861  2 builtin
;  542          0862  2         actualcount,
;  543          0863  2         nullparameter;
;  544          0864  2
;  545          0865  2
;  546          0866  2 ! If a scrolling-type command (e.g., HELP) is in progress, we can't display
;  547          0867  2 ! the message.  However, we will ring the bell to tell the user something's up.
;  548          0868  2
;  549          0869  3 if .phn$gv_scroller then (
;  550          0870  3         scr$put_screen(describe(%char(bell)));
;  551          0871  3         return;
;  552          0872  2 );
;  553          0873  2
;  554          0874  2 ! Now, if we are only to clear the message line, do it.
;  555          0875  2
;  556          0876  3 if nullparameter(1) then (
;  557          0877  3         scr$erase_line(3,1);
;  558          0878  3         return;
;  559          0879  2 );
;  560          0880  2
;  561          0881  2 ! Now we can obtain the message text and fill in the $FAO arguments.  If
;  562          0882  2 ! the message was not defined with enough $FAO codes, we don't fill anything
;  563          0883  2 ! into it.
;  564          0884  2
```

```
;  565          0885  3 begin
;  566          0886  3 local
;  567          0887  3         local_described_buffer(msg_buf,79);
;  568          0888  3
;  569        P 0889  3 status = $getmsg(msgid=.status_code,
;  570        P 0890  3                  msglen=msg_buf,
;  571        P 0891  3                  bufadr=msg_buf,
;  572        P 0892  3                  flags=%b'0001',
;  573          0893  3                  outadr=byte_array);
;  574          0894  3 check (.status);
;  575          0895  3 msg_text_ptr = msg_buf;
;  576          0896  3
;  577          0897  4 begin
;  578          0898  4 local
;  579          0899  4         local_described_buffer(fao_buf,79);
;  580          0900  4
;  581          0901  5 if (.byte_array[1] neq 0) and (actualcount() gtru .byte_array[1]) then (
;  582        P 0902  5         status = $faol(ctrstr=msg_buf,
;  583        P 0903  5                        outlen=fao_buf,
;  584        P 0904  5                        outbuf=fao_buf,
;  585          0905  5                        prmlst=faoT);
;  586          0906  5         check (.status);
;  587          0907  5         msg_text_ptr = fao_buf;
;  588          0908  4 );
;  589          0909  4
;  590          0910  4 ! Now we can clear the message line and display the message text.
;  591          0911  4
;  592          0912  4 scr$erase_line(3,1);
;  593          0913  4 scr$put_screen(.msg_text_ptr);
;  594          0914  3 end;
;  595          0915  2 end;
;  596          0916  2
;  597          0917  2 ! Set the flag that says there's a message on the message line.
;  598          0918  2
;  599          0919  2 phn$gv_message = true;
;  600          0920  2
;  601          0921  2 return;
;  602          0922  2
;  603          0923  1 end;


                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

                                      07  000C0 P.AAM:  .ASCII  <7>                                    ;
                                          000C1         .BLKB   3
                                00000001  000C4 P.AAL:  .LONG   1                                      ;
                                00000000' 000C8         .ADDRESS P.AAM                                 ;

                                                .EXTRN  SYS$GETMSG, SYS$FAOL

                                                .PSECT  $CODE$,NOWRT,2

                                          007C 00000    .ENTRY  PHN$INFORM, Save R2,R3,R4,R5,R6        ; 0854
                    56 00000000G   00   9E 00002        MOVAB   LIB$SIGNAL, R6                         ;
                    55 00000000G   00   9E 00009        MOVAB   SCR$ERASE_LINE, R5                     ;
                    54 00000000G   00   9E 00010        MOVAB   SCR$PUT_SCREEN, R4                     ;
```

```
                        5E      FF4C    CE  9E 00017          MOVAB   -180(SP), SP
                08      0000G   CF          01  E1 0001C          BBC     #1, PHN$GB_FLAGS, 1$            ; 0869
                                        0000'   CF  9F 00022          PUSHAB  P.AAL                           ; 0870
                                64              01  FB 00026          CALLS   #1, SCR$PUT_SCREEN
                                                04  00029          RET                                     ; 0869
                                                6C  95 0002A 1$:       TSTB    (AP)                            ; 0876
                                                05  13 0002C          BEQL    2$
                                          04    AC  D5 0002E          TSTL    4(AP)
                                                08  12 00031          BNEQ    3$
                                                01  DD 00033 2$:       PUSHL   #1                              ; 0877
                                                03  DD 00035          PUSHL   #3
                                65              02  FB 00037          CALLS   #2, SCR$ERASE_LINE
                                                04  0003A          RET                                     ; 0876
                            5C  AE      4F  8F  9A 0003B 3$:       MOVZBL  #79, MSG_BUF                    ; 0887
                            60  AE      64  AE  9E 00040          MOVAB   MSG_BUF+8, MSG_BUF+4
                                                5E  DD 00045          PUSHL   SP                              ; 0893
                                                01  DD 00047          PUSHL   #1
                                          64    AE  9F 00049          PUSHAB  MSG_BUF
                                          68    AE  9F 0004C          PUSHAB  MSG_BUF
                                          04    AC  DD 0004F          PUSHL   STATUS_CODE
                        00000000G       00          05  FB 00052          CALLS   #5, SYS$GETMSG
                                        52          50  D0 00059          MOVL    R0, STATUS
                                        05          52  E8 0005C          BLBS    STATUS, 4$                  ; 0894
                                                52  DD 0005F          PUSHL   STATUS
                                        66          01  FB 00061          CALLS   #1, LIB$SIGNAL
                                        53      5C  AE  9E 00064 4$:       MOVAB   MSG_BUF, MSG_TEXT_PTR           ; 0895
                                04      AE      4F  8F  9A 00068          MOVZBL  #79, FAO_BUF                    ; 0899
                                08      AE      0C  AE  9E 0006D          MOVAB   FAO_BUF+8, FAO_BUF+4
                                                01  AE  95 00072          TSTB    BYTE_ARRAY+1                    ; 0901
                                                28  13 00075          BEQL    6$
                                01      AE          6C  91 00077          CMPB    (AP), BYTE_ARRAY+1
                                                22  1B 0007B          BLEQU   6$
                                          08    AC  9F 0007D          PUSHAB  FAO1                            ; 0905
                                          08    AE  9F 00080          PUSHAB  FAO_BUF
                                          0C    AE  9F 00083          PUSHAB  FAO_BUF
                                          68    AE  9F 00086          PUSHAB  MSG_BUF
                        00000000G       00          04  FB 00089          CALLS   #4, SYS$FAOL
                                        52          50  D0 00090          MOVL    R0, STATUS
                                        05          52  E8 00093          BLBS    STATUS, 5$                  ; 0906
                                                52  DD 00096          PUSHL   STATUS
                                        66          01  FB 00098          CALLS   #1, LIB$SIGNAL
                                        53      04  AE  9E 0009B 5$:       MOVAB   FAO_BUF, MSG_TEXT_PTR           ; 0907
                                                01  DD 0009F 6$:       PUSHL   #1                              ; 0912
                                                03  DD 000A1          PUSHL   #3
                                65              02  FB 000A3          CALLS   #2, SCR$ERASE_LINE
                                                53  DD 000A6          PUSHL   MSG_TEXT_PTR                    ; 0913
                                64              01  FB 000A8          CALLS   #1, SCR$PUT_SCREEN
                            0000G   CF          01  88 000AB          BISB2   #1, PHN$GB_FLAGS                ; 0919
                                                04  000B0          RET                                     ; 0923

; Routine Size:  177 bytes,    Routine Base:  $CODE$ + 02E1
```

```
;  605          0924  1 %sbttl 'PHN$REVIEW - Recalculate Viewport Parameters'
;  606          0925  1 !++
;  607          0926  1 ! Functional Description:
;  608          0927  1 !       This routine is called to recalculate the viewport parameters
;  609          0928  1 !       in each of the existing PUBs.
;  610          0929  1 !
;  611          0930  1 ! Formal Parameters:
;  612          0931  1 !       none
;  613          0932  1 !
;  614          0933  1 ! Implicit Inputs:
;  615          0934  1 !       global data
;  616          0935  1 !
;  617          0936  1 ! Implicit Outputs:
;  618          0937  1 !       global data
;  619          0938  1 !
;  620          0939  1 ! Returned Value:
;  621          0940  1 !       phn$_noports    There is not enought room on the screen for all
;  622          0941  1 !                       required viewports.
;  623          0942  1 !
;  624          0943  1 ! Side Effects:
;  625          0944  1 !
;  626          0945  1 !--
;  627          0946  1
;  628          0947  1
;  629          0948  2 global routine phn$review = begin
;  630          0949  2
;  631          0950  2 local
;  632          0951  2         active_count: long, held_count: long,
;  633          0952  2         viewport_size: long, viewport_line: long,
;  634          0953  2         p: ref pub;
;  635          0954  2
;  636          0955  2 literal
;  637          0956  2         max_viewports = 20/pub_k_minlines;
;  638          0957  2
;  639          0958  2
;  640          0959  2 ! First we have to scan all the PUBs and make two counts:  the number of
;  641          0960  2 ! PUBs we don't have on hold and the number we do have on hold.
;  642          0961  2
;  643          0962  2 active_count = held_count = 0;
;  644          0963  2 p = .phn$gq_pubhead[0];
;  645          0964  3 while .p neqa phn$gq_pubhead do (
;  646          0965  3         if not .p[pub_v_temporary] then
;  647          0966  3                 if .p[pub_v_uhaveheld] then
;  648          0967  4                         inc (held_count)
;  649          0968  3                 else
;  650          0969  3                         inc (active_count);
;  651          0970  3
;  652          0971  3         p = .p[pub_l_flink];
;  653          0972  2 );
;  654          0973  2
;  655          0974  2 ! We need a viewport for at least all the active units.  Make sure the
;  656          0975  2 ! required viewports won't be too small.
;  657          0976  2
;  658          0977  2 if .active_count gtr max_viewports then
;  659          0978  2         return phn$_noports;
;  660          0979  2
;  661          0980  2 ! Although we need a viewport for all the active units, we may also be
```

```
;  662          0981  2 ! able to accomodate some of the held units.  Calculate how many of them
;  663          0982  2 ! we can accomodate.
;  664          0983  2
;  665          0984  2 held_count = min(.held_count,
;  666          0985  2                  max_viewports - .active_count);
;  667          0986  2
;  668          0987  2 ! Now we know how many viewports we will assign.  Calculate how many lines
;  669          0988  2 ! in each viewport, but make sure it's not above the limit set by the user.
;  670          0989  2
;  671          0990  2 viewport_size = minu(20/(.active_count+.held_count),
;  672          0991  2                       .phn$gl_viewport_size);
;  673          0992  2
;  674          0993  2 ! Now we can make another pass through the PUBs and assign viewport
;  675          0994  2 ! parameters to all active PUBs and however many held PUBs we can
;  676          0995  2 ! accomodate.
;  677          0996  2
;  678          0997  2 viewport_line = 5;
;  679          0998  2 p = .phn$gq_pubhead[0];                                                  ;
;  680          0999  3 until .p eqla phn$gq_pubhead do (
;  681          1000  3         if not .p[pub_v_temporary] then
;  682          1001  3                                                                         ;
;  683          1002  3                 if .p[pub_v_uhaveheld] and                              ;
;  684          1003  3                    (dec (held_count) lss 0) then                        ;
;  685          1004  3                                                                         ;
;  686          1005  3                         p[pub_w_viewsize] = p[pub_w_viewline] = 0
;  687          1006  4                 else (
;  688          1007  4                         p[pub_w_viewsize] = .viewport_size;
;  689          1008  4                         p[pub_w_viewline] = .viewport_line;
;  690          1009  4                         viewport_line = .viewport_line + .viewport_size;
;  691          1010  3                 );
;  692          1011  3
;  693          1012  3         p = .p[pub_l_flink];
;  694          1013  2 );
;  695          1014  2
;  696          1015  2 return phn$_ok;
;  697          1016  2
;  698          1017  1 end;


                                                 003C 00000          .ENTRY  PHN$REVIEW, Save R2,R3,R4,R5                    ; 0948
                                      55  0000G  CF   9E 00002          MOVAB   PHN$GQ_PUBHEAD, R5                              ;
                                          0000G  50   7C 00007          CLRQ    ACTIVE_COUNT                                    ; 0962
                                      52         65   D0 00009          MOVL    PHN$GQ_PUBHEAD, P                               ; 0963
                                      53         65   9E 0000C 1$:      MOVAB   PHN$GQ_PUBHEAD, R3                              ; 0964
                                      53         52   D1 0000F          CMPL    P, R3                                           ;
                                                 16   13 00012          BEQL    4$                                              ;
                                0B    00F0      C2   02   E0 00014          BBS     #2, 240(P), 3$                          ; 0965
                                      04  00F0  C2   E9 0001A          BLBC    240(P), 2$                                      ; 0966
                                                 51   D6 0001F          INCL    HELD_COUNT                                      ; 0967
                                                 02   11 00021          BRB     3$                                              ;
                                                 50   D6 00023 2$:      INCL    ACTIVE_COUNT                                    ; 0969
                                      52         62   D0 00025 3$:      MOVL    (P), P                                          ; 0971
                                                 E2   11 00028          BRB     1$                                              ; 0964
                                      06         50   D1 0002A 4$:      CMPL    ACTIVE_COUNT, #6                                ; 0977
```

```
                                  08  15 0002D        BLEQ    5$
             50 00000000G         8F  D0 0002F        MOVL    #PHN$_NOPORTS, R0                          ; 0978
                                  04     00036        RET
      54              06          50  C3 00037 5$:    SUBL3   ACTIVE_COUNT, #6, R4                       ; 0985
                      53          51  D0 0003B        MOVL    HELD_COUNT, R3                             ;
                      54          53  D1 0003E        CMPL    R3, R4                                     ;
                                  03  15 00041        BLEQ    6$                                         ;
                      53          54  D0 00043        MOVL    R4, R3                                     ;
                      51          53  D0 00046 6$:    MOVL    R3, HELD_COUNT                             ; 0984
                      50          51  C0 00049        ADDL2   HELD_COUNT, R0                             ; 0990
      50              14          50  C7 0004C        DIVL3   R0, #20, R0                                ;
              0000G   CF          50  D1 00050        CMPL    R0, PHN$GL_VIEWPORT_SIZE                   ; 0991
                                  05  1B 00055        BLEQU   7$                                         ;
                      50   0000G  CF  D0 00057        MOVL    PHN$GL_VIEWPORT_SIZE, R0                   ;
                      54          50  D0 0005C 7$:    MOVL    R0, VIEWPORT_SIZE                          ; 0990
                      53          05  D0 0005F        MOVL    #5, VIEWPORT_LINE                          ; 0997
                      52          65  D0 00062        MOVL    PHN$GQ_PUBHEAD, P                          ; 0998
                      50          65  9E 00065 8$:    MOVAB   PHN$GQ_PUBHEAD, R0                         ; 0999
                      50          52  D1 00068        CMPL    P, R0                                      ;
                                  26  13 0006B        BEQL    11$                                        ;
      1B      00F0    C2          02  E0 0006D        BBS     #2, 240(P), 10$                            ; 1000
                      09   00F0   C2  E9 00073        BLBC    240(P), 9$                                 ; 1002
                      06          51  F4 00078        SOBGEQ  HELD_COUNT, 9$                             ; 1003
                           00FC   C2  D4 0007B        CLRL    252(P)                                     ; 1005
                                  0D  11 0007F        BRB     10$                                        ;
              00FC    C2          54  B0 00081 9$:    MOVW    VIEWPORT_SIZE, 252(P)                      ; 1007
              00FE    C2          53  B0 00086        MOVW    VIEWPORT_LINE, 254(P)                      ; 1008
                      53          54  C0 0008B        ADDL2   VIEWPORT_SIZE, VIEWPORT_LINE               ; 1009
                      52          62  D0 0008E 10$:   MOVL    (P), P                                     ; 1012
                                  D2  11 00091        BRB     8$                                         ; 0999
             50 00000000G         8F  D0 00093 11$:   MOVL    #PHN$_OK, R0                               ; 1015
                                  04     0009A        RET                                                ; 1017

; Routine Size:  155 bytes,    Routine Base:  $CODE$ + 0392
```

```
;  700          1018  1 %sbttl 'PHN$VIEW_GOODIES - Calculate Viewport Information'
;  701          1019  1 !++
;  702          1020  1 ! Functional Description:
;  703          1021  1 !       This routine is called to calculate various information about a
;  704          1022  1 !       viewport.  The information that it returns is determined by the
;  705          1023  1 !       parameters.
;  706          1024  1 !
;  707          1025  1 ! Formal Parameters:
;  708          1026  1 !       goodies_pub     The address of the PL3 describing the viewport.
;  709          1027  1 !
;  710          1028  1 !       cursor_pos      The address of a vector of 2 longwords.  We return
;  711          1029  1 !                       the line and column numbers specifying the current
;  712          1030  1 !                       position of the cursor.
;  713          1031  1 !
;  714          1032  1 !       line_numbers    The address of a longword vector.  We return the line
;  715          1033  1 !                       numbers of the text lines in the viewport.  The zeroth
;  716          1034  1 !                       entry contains the line with the latest CTL.
;  717          1035  1 !
;  718          1036  1 !       text_dscs       The address of a longword vector.  We return the
;  719          1037  1 !                       address of the descriptor of the text to go on the line
;  720          1038  1 !                       specified in the corresponding line_numbers entry.
;  721          1039  1 !
;  722          1040  1 ! Implicit Inputs:
;  723          1041  1 !       global data
;  724          1042  1 !
;  725          1043  1 ! Implicit Outputs:
;  726          1044  1 !       global data
;  727          1045  1 !
;  728          1046  1 ! Returned Value:
;  729          1047  1 !       none
;  730          1048  1 !
;  731          1049  1 ! Side Effects:
;  732          1050  1 !
;  733          1051  1 !--
;  734          1052  1
;  735          1053  1
;  736          1054  1 global routine phn$view_goodies(goodies_pub,cursor_pos,line_numbers,text_dscs):
;  737          1055  2                                                       novalue = begin
;  738          1056  2
;  739          1057  2 bind
;  740          1058  2         gp = .goodies_pub: pub;
;  741          1059  2
;  742          1060  2 bind
;  743          1061  2         latest_ctl = .gp[pub_q_ctlhead0]: ctl,
;  744          1062  2         latest_text_dsc = latest_ctl[ctl_q_line]: descriptor;
;  745          1063  2
;  746          1064  2 local
;  747          1065  2         latest_line: long,
;  748          1066  2         text_count: long,
;  749          1067  2         c: ref ctl;
;  750          1068  2
;  751          1069  2 builtin
;  752          1070  2         nullparameter;
;  753          1071  2
;  754          1072  2
;  755          1073  2 ! First let's get some terminology straight:
;  756          1074  2 !
```

```
;  757     1075  2 !       oldest CTL      The CTL at the end of the CTL list.  It is the CTL
;  758     1076  2 !                       representing the oldest line of the conversation.
;  759     1077  2 !
;  760     1078  2 !       latest CTL      The CTL at the front of the CTL list.  It represents
;  761     1079  2 !                       the current line of the conversation.
;  762     1080  2 !
;  763     1081  2 !       top CTL         The CTL representing the line that would be at the
;  764     1082  2 !                       top of the viewport if the viewport were infinitely
;  765     1083  2 !                       large.  Usually the oldest CTL, but not if the user
;  766     1084  2 !                       has typed a CTRL/L.
;  767     1085  2
;  768     1086  2 ! We begin by calculating the line on which the latest CTL text should be
;  769     1087  2 ! placed.  There are two algorithms.
;  770     1088  2
;  771     1089  2 if .phn$qb_scroll then
;  772     1090  2
;  773     1091  2         ! If we are scrolling, then the latest CTL goes on the bottom line
;  774     1092  2         ! of the viewport, unless we are not very many CTLs from the top.
;  775     1093  2
;  776     1094  2         latest_line = .gp[pub_w_viewline] + 1 +
;  777     1095  2                       minu(.gp[pub_w_viewsize] - 3,
;  778     1096  2                            .gp[pub_l_ctlcount] - .gp[pub_l_topctl])
;  779     1097  2
;  780     1098  2 else
;  781     1099  2
;  782     1100  2         ! If we are wrapping, then the latest CTL goes on a line determined
;  783     1101  2         ! by the number of lines from the top MOD the size of the viewport.
;  784     1102  2
;  785     1103  2         latest_line = .gp[pub_w_viewline] + 1 +
;  786     1104  2                       (.gp[pub_l_ctlcount] - .gp[pub_l_topctl]) mod
;  787     1105  2                               (.gp[pub_w_viewsize] - 2);
;  788     1106  2
;  789     1107  2 ! Now if the caller wants to know the current position, fill it in.
;  790     1108  2
;  791     1109  3 if not nullparameter(2) then (
;  792     1110  3         bind
;  793     1111  3                 cursor_pos_array = .cursor_pos: vector[2,long];
;  794     1112  3
;  795     1113  3         cursor_pos_array[0] = .latest_line;
;  796     1114  3         cursor_pos_array[1] = .latest_text_dsc[len] + 1;
;  797     1115  2 );
;  798     1116  2
;  799     1117  2 ! Now if the caller only wanted to know the current position, we're done.
;  800     1118  2
;  801     1119  2 if nullparameter(3) or nullparameter(4) then
;  802     1120  2         return;
;  803     1121  2
;  804     1122  2 ! We are to fill in the line number and text descriptor arrays.
;  805     1123  2 ! Now we calculate how many text lines we will display in the viewport.
;  806     1124  2 ! This is equal to the number of text lines in the viewport, unless we
;  807     1125  2 ! are fewer than that from the top.
;  808     1126  2
;  809     1127  2 text_count = minu(.gp[pub_l_ctlcount] - .gp[pub_l_topctl] + 1,
;  810     1128  2                  .gp[pub_w_viewsize] - 2);
;  811     1129  2
;  812     1130  2 ! Now we will fill in as many array entries as there are lines in the
;  813     1131  2 ! viewport.  Each line number entry gets the corresponding viewport line
```

```
;  814          1132  2 ! number.  Each text descriptor either gets the address of a descriptor
;  815          1133  2 ! from a CTL, or the address of a null descriptor if the line is to be
;  816          1134  2 ! blanked.
;  817          1135  2
;  818          1136  3 begin
;  819          1137  3 bind
;  820          1138  3         line_numbers_array = .line_numbers: vector[pub_k_maxlines-2,long],
;  821          1139  3         text_dscs_array = .text_dscs: vector[pub_k_maxlines-2,long];
;  822          1140  3
;  823          1141  3 c = latest_ctl;
;  824          1142  4 incru i from 0 to .gp[pub_w_viewsize]-2-1 do (
;  825          1143  4         line_numbers_array[.i] = .latest_line;
;  826          1144  4         dec (latest_line);
;  827          1145  4         if .latest_line lequ .gp[pub_w_viewline] then
;  828          1146  4                 latest_line = .gp[pub_w_viewline] + .gp[pub_w_viewsize]  - 2;
;  829          1147  4
;  830          1148  5         if dec (text_count) geq 0 then (
;  831          1149  5                 text_dscs_array[.i] = c[ctl_q_line];
;  832          1150  5                 c = .c[ctl_l_flink];
;  833          1151  4         ) else
;  834          1152  4                 text_dscs_array[.i] = describe('');
   835          1153  3 );
;  836          1154  2 end;
;  837          1155  2
;  838          1156  2 return;
;  839          1157  2
;  840          1158  1 end;


                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

                                00000CC P.AAO:  .BLKB   0                                       ;
                       00000000 00000CC P.AAN:  .LONG   0                                       ;
                       00000000' 00000D0        .ADDRESS P.AAO                                  ;


                                                .PSECT  $CODE$,NOWRT,2

                                  00FC 00000            .ENTRY  PHN$VIEW_GOODIES, Save R2,R3,R4,R5,R6,R7 ; 1054
                    52      04    AC  D0 00002        MOVL    GOODIES_PUB, R2                         ; 1058
57      0104        C2            10  C1 00006        ADDL3   #16, 260(R2), R7                        ; 1062
                    2B  0000G     CF  E9 0000C        BLBC    PHN$GB_SCROLL, 2$                       ; 1089
                    50  00FE      C2  9E 00011        MOVAB   254(R2), R0                             ; 1094
                    51            60  3C 00016        MOVZWL  (R0), R1                                ;
                    50  00FC      C2  9E 00019        MOVAB   252(R2), R0                             ; 1095
                    55            60  3C 0001E        MOVZWL  (R0), R5                                ;
                    53      FD    A5  9E 00021        MOVAB   -3(R5), R3                              ;
54      0100        C2      010C  C2  C3 00025        SUBL3   268(R2), 256(R2), R4                    ; 1096
                    54            53  D1 0002D        CMPL    R3, R4                                  ;
                                  03  1B 00030        BLEQU   1$                                      ;
                    53            54  D0 00032        MOVL    R4, R3                                  ;
                    56      01 A341  9E 00035 1$:     MOVAB   1(R3)[R1], LATEST_LINE                  ; 1094
                                  2B  11 0003A        BRB     3$                                      ;
                    50      00FE      C2  9E 0003C 2$:  MOVAB   254(R2), R0                         ; 1103
                    51                60  3C 00041    MOVZWL  (R0), R1                              ;
```

```
                54      0100    C2      010C    C2  C3 00044          SUBL3   268(R2), 256(R2), R4                    ; 1104
                50      00FC    C2      9E 0004C          MOVAB   252(R2), R0                     ; 1105
                55              60      3C 00051          MOVZWL  (R0), R5
                53        FE    A5      9E 00054          MOVAB   -2(R5), R3
      7E        00              54      01 7A 00058       EMUL    #1, R4, #0, -(SP)
      50        50              8E      53 7B 0005D       EDIV    R3, (SP)+, R0, R0
                56        01 A041       9E 00062          MOVAB   1(R0)[R1], LATEST_LINE                  ; 1103
                02              6C      91 00067 3$:      CMPB    (AP), #2                        ; 1109
                                13      1F 0006A          BLSSU   4$
                          08    AC      D5 0006C          TSTL    8(AP)
                                0E      13 0006F          BEQL    4$
                50        08    AC      D0 00071          MOVL    CURSOR_POS, R0                  ; 1111
                60              56      D0 00075          MOVL    LATEST_LINE, (R0)               ; 1113
              04    A0                  67      3C 00078  MOVZWL  (R7), 4(R0)                     ; 1114
                          04    A0      D6 0007C          INCL    4(R0)
                03              6C      91 0007F 4$:      CMPB    (AP), #3                        ; 1119
                                58      1F 00082          BLSSU   11$
                          0C    AC      D5 00084          TSTL    12(AP)
                                53      13 00087          BEQL    11$
                04              6C      91 00089          CMPB    (AP), #4
                                4E      1F 0008C          BLSSU   11$
                          10    AC      D5 0008E          TSTL    16(AP)
                                49      13 00091          BEQL    11$
                50        01    A4      9E 00093          MOVAB   1(R4), R0                       ; 1127
                53        FE    A5      9E 00097          MOVAB   -2(R5), R3                      ; 1128
                53              50      D1 0009B          CMPL    R0, R3
                                03      1B 0009E          BLEQU   5$
                50              53      D0 000A0          MOVL    R3, R0
                54      0104    C2      D0 000A3 5$:      MOVL    260(R2), C                      ; 1141
                                53      D7 000A8          DECL    R3                              ; 1142
                                52      D4 000AA          CLRL    I
                                29      11 000AC          BRB     10$
            0C BC42             56      D0 000AE 6$:      MOVL    LATEST_LINE, @LINE_NUMBERS[I]   ; 1143
                                56      D7 000B3          DECL    LATEST_LINE                     ; 1144
                51              56      D1 000B5          CMPL    LATEST_LINE, R1                 ; 1145
                                05      1A 000B8          BGTRU   7$
                56        FE A541       9E 000BA          MOVAB   -2(R5)[R1], LATEST_LINE         ; 1146
                                50      D7 000BF 7$:      DECL    TEXT_COUNT                      ; 1148
                                0B      19 000C1          BLSS    8$
            10 BC42       10    A4      9E 000C3          MOVAB   16(R4), @TEXT_DSCS[I]           ; 1149
                54              64      D0 000C9          MOVL    (C), C                          ; 1150
                                07      11 000CC          BRB     9$                              ; 1148
            10 BC42     0000'   CF      9E 000CE 8$:      MOVAB   P.AAN, @TEXT_DSCS[I]            ; 1152
                                52      D6 000D5 9$:      INCL    I                               ; 1142
                53              52      D1 000D7 10$:     CMPL    I, R3
                                D2      1B 000DA          BLEQU   6$
                                        04 000DC 11$:     RET                                     ; 1158

; Routine Size:  221 bytes,    Routine Base:  $CODE$ + 042D
```

```
;  842          1159  1 %sbttl 'PHN$FRESH_VIEW - Refresh a Viewport'
;  843          1160  1 !++
;  844          1161  1 ! Functional Description:
;  845          1162  1 !       This routine is called to refresh a specific viewport on the screen.
;  846          1163  1 !
;  847          1164  1 ! Formal Parameters:
;  848          1165  1 !       fresh_pub       Address of PUB representing viewport to be refreshed.
;  849          1166  1 !       viewport_clear  True if viewport is known to be erased upon call;
;  850          1167  1 !                       false otherwise.  This is used for optimization.
;  851          1168  1 !       top_bottom_too  True if entire viewport is to be refreshed; false if
;  852          1169  1 !                       only the text lines.
;  853          1170  1 !
;  854          1171  1 ! Implicit Inputs:
;  855          1172  1 !       global data
;  856          1173  1 !
;  857          1174  1 ! Implicit Outputs:
;  858          1175  1 !       global data
;  859          1176  1 !
;  860          1177  1 ! Returned Value:
;  861          1178  1 !       none
;  862          1179  1 !
;  863          1180  1 ! Side Effects:
;  864          1181  1 !
;  865          1182  1 !--
;  866          1183  1
;  867          1184  1
;  868          1185  2 global routine phn$fresh_view(fresh_pub,viewport_clear,top_bottom_too): novalue = begin
;  869          1186  2
;  870          1187  2 bind
;  871          1188  2         fp = .fresh_pub: pub,
;  872          1189  2         fresh_tsb = fp[pub_b_tsb]: tsb;
;  873          1190  2
;  874          1191  2 local
;  875          1192  2         cursor_pos: vector[2,long],
;  876          1193  2         line_numbers: vector[pub_k_maxlines-2,long],
;  877          1194  2         text_dscs:    vector[pub_k_maxlines-2,long],
;  878          1195  2         c: ref ctl;
;  879          1196  2
;  880          1197  2
;  881          1198  2 ! If a scrolling-type command (e.g., HELP) is in progress, we can't refresh
;  882          1199  2 ! the viewport.
;  883          1200  2
;  884          1201  2 if .phn$gv_scroller then
;  885          1202  2         return;
;  886          1203  2
;  887          1204  2 ! If we are to refresh the entire viewport, then we have to clear the first
;  888          1205  2 ! line and center the user name on it, display the hold indicators, and
;  889          1206  2 ! throw up a lines of dashes at the bottom.
;  890          1207  2
;  891          1208  3 if .top_bottom_too then (
;  892          1209  4         begin
;  893          1210  4         bind
;  894          1211  4                 spec_dsc = fresh_tsb[tsb_q_tkndsc,0]: descriptor;
;  895          1212  4
;  896          1213  4         if not .viewport_clear then
;  897          1214  4                 scr$erase_line(.fp[pub_w_viewline],1);
;  898          1215  4         scr$put_screen(spec_dsc,.fp[pub_w_viewline],(81-.spec_dsc[len])/2,%b'0001');
```

```
;  899          1216  3          end;
;  900          1217  3
;  901          1218  3          phn$fresh_hold(fp);
;  902          1219  3
;  903          1220  3          scr$put_screen(dashes,.fp[pub_w_viewline]+.fp[pub_w_viewsize]-1,1);
;  904          1221  2 );
;  905          1222  2
;  906          1223  2 ! Now we want to refresh the text lines themselves.  We call a routine
;  907          1224  2 ! to tell us which lines and what text goes on them.
;  908          1225  2
;  909          1226  2 phn$view_goodies(fp,cursor_pos,line_numbers,text_dscs);
;  910          1227  2
;  911          1228  2 ! Now we loop through the arrays, acting on each active entry.  The
;  912          1229  2 ! line number array tells us which viewport line the text goes on.  The
;  913          1230  2 ! text descriptor array tells us what text to put there.
;  914          1231  2
;  915          1232  3 incru i from 0 to .fp[pub_w_viewsize]-2-1 do (
;  916          1233  3          bind
;  917          1234  3                  line_dsc = .text_dscs[.i]: descriptor;
;  918          1235  3
;  919          1236  4          if not .viewport_clear or .line_dsc[len] nequ 0 then (
;  920          1237  4                  scr$erase_line(.line_numbers[.i],1);
;  921          1238  4                  scr$put_screen(line_dsc);
;  922          1239  3          );
;  923          1240  2 );
;  924          1241  2
;  925          1242  2 ! Finally, position the cursor at it's current point in the viewport.
;  926          1243  2
;  927          1244  2 scr$set_cursor(.cursor_pos[0],.cursor_pos[1]);
;  928          1245  2
;  929          1246  2 return;
;  930          1247  2
;  931          1248  1 end;


                                        00FC 00000          .ENTRY  PHN$FRESH_VIEW, Save R2,R3,R4,R5,R6,R7      ; 1185
                                57 00000000G   00    9E 00002       MOVAB   SCR$ERASE_LINE, R7                  ;
                                56 00000000G   00    9E 00009       MOVAB   SCR$PUT_SCREEN, R6                  ;
                                5E        B8   AE    9E 00010       MOVAB   -72(SP), SP                         ;
                                52        04   AC    D0 00014       MOVL    FRESH_PUB, R2                       ; 1188
                          01    0000G   CF             01    E1 00018       BBC     #1, PHN$GB_FLAGS, 1$        ; 1201
                                                       04 0001E       RET                                         ;
                                49        0C   AC    E9 0001F 1$:   BLBC    TOP_BOTTOM_TOO, 3$                  ; 1208
                                0A        08   AC    E8 00023       BLBS    VIEWPORT_CLEAR, 2$                  ; 1213
                                                       01    DD 00027       PUSHL   #1                          ; 1214
                                7E      00FE   C2    3C 00029       MOVZWL  254(R2), -(SP)                      ;
                                67             02    FB 0002E       CALLS   #2, SCR$ERASE_LINE                  ;
                                                       01    DD 00031 2$:   PUSHL   #1                          ; 1215
                                50        10   A2    3C 00033       MOVZWL  16(R2), R0                          ;
                                50        AF   A0    9E 00037       MOVAB   -81(R0), R0                         ;
                                50             02    C6 0003B       DIVL2   #2, R0                              ;
                                7E             50    CE 0003E       MNEGL   R0, -(SP)                           ;
                                7E      00FC   C2    3C 00041       MOVZWL  254(R2), -(SP)                      ;
                                         10    A2    9F 00046       PUSHAB  16(R2)                              ;
```

```
                         66               04   FB 00049          CALLS   #4, SCR$PUT_SCREEN                    ;
                                          52   DD 0004C          PUSHL   R2                                   ; 1218
                  0000V  CF               01   FB 0004E          CALLS   #1, PHN$FRESH_HOLD                   ;
                                          01   DD 00053          PUSHL   #1                                   ; 1220
                         50       00FE    C2   3C 00055          MOVZWL  254(R2), R0                          ;
                         51       00FC    C2   3C 0005A          MOVZWL  252(R2), R1                          ;
                         50               51   C0 0005F          ADDL2   R1, R0                               ;
                                    FF    A0   9F 00062          PUSHAB  -1(R0)                               ;
                                  0000'   CF   9F 00065          PUSHAB  DASHES                               ;
                         66               03   FB 00069          CALLS   #3, SCR$PUT_SCREEN                   ;
                                          5E   DD 0006C 3$:      PUSHL   SP                                   ; 1226
                                    24    AE   9F 0006E          PUSHAB  LINE_NUMBERS                         ;
                                    48    AE   9F 00071          PUSHAB  CURSOR_POS                           ;
                                          52   DD 00074          PUSHL   R2                                   ;
                  FEA8   CF               04   FB 00076          CALLS   #4, PHN$VIEW_GOODIES                 ;
                         54       00FC    C2   3C 00078          MOVZWL  252(R2), R4                          ; 1232
                         54               03   C2 00080          SUBL2   #3, R4                               ;
                         55         08    AC   D2 00083          MCOML   VIEWPORT_CLEAR, R5                   ; 1236
                                          52   D4 00087          CLRL    I                                    ;
                                          1B   11 00089          BRB     7$                                   ;
                         53             6E42  D0 0008B 4$:      MOVL    TEXT_DSCS[I], R3                     ; 1234
                         04               55  E8 0008F          BLBS    R5, 5$                               ; 1236
                                          63   B5 00092          TSTW    (R3)                                 ;
                                          0E   13 00094          BEQL    6$                                   ;
                                          01   DD 00096 5$:      PUSHL   #1                                   ; 1237
                                    24 AE42    DD 00098          PUSHL   LINE_NUMBERS[I]                      ;
                         67               02   FB 0009C          CALLS   #2, SCR$ERASE_LINE                   ;
                                          53   DD 0009F          PUSHL   R3                                   ; 1238
                         66               01   FB 000A1          CALLS   #1, SCR$PUT_SCREEN                   ;
                                          52   D6 000A4 6$:      INCL    I                                    ; 1232
                         54               52   D1 000A6 7$:      CMPL    I, R4                                ;
                                          E0   1B 000A9          BLEQU   4$                                   ;
                                    44    A    DD 000AB          PUSHL   CURSOR_POS+4                         ; 1244
                                    44    A    DD 000AE          PUSHL   CURSOR_POS                           ;
                00000000G         00         02   FB 000B1          CALLS   #2, SCR$SET_CURSOR                   ;
                                               04 000B8          RET                                          ; 1248

; Routine Size:  185 bytes,    Routine Base:  $CODE$ + 050A
```

```
;  933          1249  1 %sbttl 'PHN$FRESH_HOLD - Display Hold Indicators'
;  934          1250  1 !++
;  935          1251  1 ! Functional Description:
;  936          1252  1 !       This routine is called to display the hold indicators on the first
;  937          1253  1 !       line of a viewport.  One indicator says whether or not we have the
;  938          1254  1 !       person on hold, the other says whether or not they have us on hold.
;  939          1255  1 !
;  940          1256  1 ! Formal Parameters:
;  941          1257  1 !       fresh_pub       The address of the PUB representing the viewport.
;  942          1258  1 !
;  943          1259  1 ! Implicit Inputs:
;  944          1260  1 !       global data
;  945          1261  1 !
;  946          1262  1 ! Implicit Outputs:
;  947          1263  1 !       global data
;  948          1264  1 !
;  949          1265  1 ! Returned Value:
;  950          1266  1 !       none
;  951          1267  1 !
;  952          1268  1 ! Side Effects:
;  953          1269  1 !
;  954          1270  1 !--
;  955          1271  1
;  956          1272  1
;  957          1273  2 global routine phn$fresh_hold(fresh_pub): novalue = begin
;  958          1274  2
;  959          1275  2 bind
;  960          1276  2         fp = .fresh_pub: pub;
;  961          1277  2
;  962          1278  2
;  963          1279  2 ! If a scrolling-type command (e.g., HELP) is in progress, we can't display
;  964          1280  2 ! the indicators.
;  965          1281  2
;  966          1282  2 if .phn$gv_scroller then
;  967          1283  2         return;
;  968          1284  2
;  969          1285  2 ! Display an indicator to say if we have them on hold.
;  970          1286  2
;  971          1287  2 scr$put_screen(if .fp[pub_v_uhaveheld] then describe('(YOU HAVE HELD)')
;  972          1288  2                                       else describe('                '),
;  973          1289  2                 .fp[pub_w_viewline],1);
;  974          1290  2
;  975          1291  2 ! Display an indicator to say if they have us on hold.
;  976          1292  2
;  977          1293  2 if .fp[pub_v_hasuheld] then
;  978          1294  2         scr$put_screen(describe('(HAS YOU HELD)'),.fp[pub_w_viewline],66)
;  979          1295  2 else
;  980          1296  2         scr$erase_line(.fp[pub_w_viewline],66);
;  981          1297  2
;  982          1298  2 return;
;  983          1299  2
;  984          1300  1 end;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2
```

```
29 44 4C 45 48 20 45 56 41 48 20 55 4F 59 28  000D4 P.AAQ:  .ASCII  \(YOU HAVE HELD)\                ;
                                              000E3        .BLKB   1
                                   0000000F   000E4 P.AAP:  .LONG   15                              ;
                                   00000000'  000E8        .ADDRESS P.AAQ                          ;
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20  000EC P.AAS:  .ASCII  \                \             ;
                                              000FB        .BLKB   1
                                   0000000F   000FC P.AAR:  .LONG   15                              ;
                                   00000000'  00100        .ADDRESS P.AAS                          ;
   29 44 4C 45 48 20 55 4F 59 20 53 41 48 28  00104 P.AAU:  .ASCII  \(HAS YOU HELD)\                ;
                                              00112        .BLKB   2
                                   0000000E   00114 P.AAT:  .LONG   14                              ;
                                   00000000'  00118        .ADDRESS P.AAU                          ;

                                                           .PSECT  $CODE$,NOWRT,2

                                         001C 00000        .ENTRY  PHN$FRESH_HOLD, Save R2,R3,R4    ; 1273
                          54 00000000G   00  9E 00002        MOVAB   SCR$PUT_SCREEN, R4              ;
                          52       04    AC  D0 00009        MOVL    FRESH_POB, R2                   ; 1276
                40  0000G CF             01  EC 0000D        BBS     #1, PHN$GB_FLAGS, 4$            ; 1282
                                         01  DD 00013        PUSHL   #1                              ; 1287
                          53     00FE    C2  3C 00015        MOVZWL  254(R2), R3                     ; 1289
                                         53  DD 0001A        PUSHL   R3                              ;
                          07     00F0    C2  E9 0001C        BLBC    240(R2), 1$                     ; 1287
                          50     0000'   CF  9E 00021        MOVAB   P.AAP, R0                       ;
                                         05  11 00026        BRB     2$                              ;
                          50     0000'   CF  9E 00028 1$:     MOVAB   P.AAR, R0                       ; 1288
                                         50  DD 0002D 2$:     PUSHL   R0                              ;
                          64             03  FB 0002F        CALLS   #3, SCR$PUT_SCREEN              ; 1287
                0E  00F0  C2             01  E1 00032        BBC     #1, 240(R2), 3$                 ; 1293
                          7E       42    8F  9A 00038        MOVZBL  #66, -(SP)                      ; 1294
                                         53  DD 0003C        PUSHL   R3                              ;
                                   0000' CF  9F 0003E        PUSHAB  P.AAT                           ;
                          64             03  FB 00042        CALLS   #3, SCR$PUT_SCREEN              ;
                                         04  00045        RET                                     ;
                          7E       42    8F  9A 00046 3$:     MOVZBL  #66, -(SP)                      ; 1296
                                         53  DD 0004A        PUSHL   R3                              ;
                00000000G 00             02  FB 0004C        CALLS   #2, SCR$ERASE_LINE              ;
                                         04  00053 4$:     RET                                     ; 1300

; Routine Size:  84 bytes,    Routine Base:  $CODE$ + 05C3
```

```
;  986          1301  1 %sbttl 'PHN$SHOW_TEXT - Display Text on Screen'
;  987          1302  1 !++
;  988          1303  1 ! Functional Description:
;  989          1304  1 !       This routine is called to display text on the screen.  It can be
;  990          1305  1 !       passed the address of a PUB describing the viewport in which the
;  991          1306  1 !       text is to be displayed.  It can also be passed an address of zero,
;  992          1307  1 !       in which case the text is displayed on the command line.
;  993          1308  1 !
;  994          1309  1 ! Formal Parameters:
;  995          1310  1 !       display_pub     The address of the PUB describing the viewport to
;  996          1311  1 !                       contain the text (viewport parameters are assured to
;  997          1312  1 !                       be filled in).  An address of zero means the text
;  998          1313  1 !                       goes on the command line.
;  999          1314  1 !       text            Address of the descriptor of the text.
; 1000          1315  1 !
; 1001          1316  1 ! Implicit Inputs:
; 1002          1317  1 !       global data
; 1003          1318  1 !
; 1004          1319  1 ! Implicit Outputs:
; 1005          1320  1 !       global data
; 1006          1321  1 !
; 1007          1322  1 ! Returned Value:
; 1008          1323  1 !       none
; 1009          1324  1 !
; 1010          1325  1 ! Side Effects:
; 1011          1326  1 !
; 1012          1327  1 !--
; 1013          1328  1
; 1014          1329  1
; 1015          1330  2 global routine phn$show_text(display_pub,text): novalue = begin
; 1016          1331  2
; 1017          1332  2 bind
; 1018          1333  2         dp = .display_pub: pub;
; 1019          1334  2
; 1020          1335  2 local
; 1021          1336  2         text_dsc: descriptor,
; 1022          1337  2         char: byte,
; 1023          1338  2         c: ref ctl,
; 1024          1339  2         latest_line_dsc: ref descriptor,
; 1025          1340  2         cursor_pos: vector[2,long],
; 1026          1341  2         word_wrap: byte;
; 1027          1342  2
; 1028          1343  2 builtin
; 1029          1344  2         nullparameter;
```

```
;  1031          1345  2   This routine is called from the screen update logic.  It determines the
;  1032          1346  2 ! length of the last word in the text and returns that length.  This
;  1033          1347  2 ! information is needed when we are to word-wrap or delete the last word
;  1034          1348  2 ! on the line.
;  1035          1349  2
;  1036          1350  3 routine last_word(text) = begin                                              ; R
;  1037          1351  3
;  1038          1352  3 bind
;  1039          1353  3         text_dsc = .text: descriptor,
;  1040          1354  3         text_string = .text_dsc[ptr]: vector[,byte];
;  1041          1355  3
;  1042          1356  3 local
;  1043          1357  3         i: long;
;  1044          1358  3
;  1045          1359  3
;  1046          1360  3 ! First we scan backwards for the last non-blank in the text.  This marks
;  1047          1361  3 ! the end of the last word.
;  1048          1362  3
;  1049          1363  4 i = (
;  1050          1364  5         decr i from .text_dsc[len]-1 to -1 do (
;  1051          1365  5         if .i eql -1 then exitloop .i;
;  1052          1366  5         if .text_string[.i] nequ ' ' then exitloop .i;
;  1053          1367  5         )
;  1054          1368  3 );
;  1055          1369  3
;  1056          1370  3 ! Now we scan backwards for the preceeding blank in the text.  This marks
;  1057          1371  3 ! the beginning of the last word.
;  1058          1372  3
;  1059          1373  4 i = (
;  1060          1374  5         decr i from .i to -1 do (
;  1061          1375  5         if .i eql -1 then exitloop .i;
;  1062          1376  5         if .text_string[.i] eqlu ' ' then exitloop .i;
;  1063          1377  5         )
;  1064          1378  3 );
;  1065          1379  3
;  1066          1380  3 ! Return the length of the last word.
;  1067          1381  3
;  1068          1382  3 return .text_dsc[len] - .i - 1;
;  1069          1383  2 end;


                                0004 00000 LAST_WORD:
                                           .WORD   Save R2                                          ; 1350
                     52    04   AC  D0 00002         MOVL    TEXT, R2                               ; 1353
                     50         62  3C 00006         MOVZWL  (R2), I                                ; 1364
                                0E  11 00009         BRB     3$                                     ;
                                07  13 0000B 1$:     BEQL    2$                                     ; 1365
                     20    04 B240  91 0000D         CMPB    @4'R2)[I], #32                         ; 1366
                                05  13 00012         BEQL    3$                                     ;
                     51         50  D0 00014 2$:     MOVL    I, I                                   ;
                                0E  11 00017         BRB     4$                                     ;
                                50  D7 00019 3$:     DECL    I                                      ; 1364
                 FFFFFFFF   8F   50  D1 0001B         CMPL    I, #-1                                 ;
                                E7  18 0002?         BGEQ    1$                                     ;
```

```
                                 51            01  CE 00024          MNEGL   #1, I                                    ;
                                 50            51  D0 00027 4$:       MOVL    I, I                                    ; 1374
                                               10  11 0002A          BRB     8$
                                               07  13 0002C 5$:       BEQL    6$                                      ; 1375
                                 20  04 B240    91  0002E          CMPB    @4(R2)[I], #32                          ; 1376
                                               05  12 00033          BNEQ    7$
                                 51            50  D0 00035 6$:       MOVL    I, I
                                               0E  11 00038          BRB     9$
                                               50  D7 0003A 7$:       DECL    I                                       ; 1374
                          FFFFFFFF   8F        50  D1 0003C 8$:       CMPL    I, #-1
                                               E7  18 00043          BGEQ    5$
                                 51            01  CE 00045          MNEGL   #1, I
                                 52            62  3C 00048 9$:       MOVZWL  (R2), R2                                ; 1382
                                 52            51  C2 0004B          SUBL2   I, R2
                                 50            72  9E 0004E          MOVAB   -(R2), R0
                                                   04 00051          RET                                             ; 1383

; Routine Size:  82 bytes,    Routine Base:  $CODE$ + 0617

;  1070          1384  2
;  1071          1385  2
;  1072          1386  2 local
;  1073          1387  2         local_described_buffer(screen_buf,1024);
```

```
; 1075          1388  2 ! We initialize by copying the descriptor of the text because we are going
; 1076          1389  2 ! go clobber it.
; 1077          1390  2
; 1078          1391  2 ch$move(8,.text,text_dsc);
; 1079          1392  2
; 1080          1393  2 ! Now we see if we are to display text on the command line.  If so, we
; 1081          1394  2 ! process each text character and handle it appropriately.
; 1082          1395  2
; 1083          1396  3 if nullparameter(1) then (
; 1084          1397  3
; 1085          1398  4         while dec (text_dsc[len]) geq 0 do (
; 1086          1399  4
; 1087          1400  4                 char = ch$rchar_a(text_dsc[ptr]);
; 1088          1401  4
; 1089          1402  4                 selectoneu .char of set
; 1090          1403  4                 [%x'20' to %x'7e',
; 1091          1404  4                  %x'80' to %x'ff']:
; 1092          1405  4
; 1093          1406  4                                 ! We have a normal character.  Place it on the
; 1094          1407  4                                 ! command line and advance the cursor position.
; 1095          1408  4
; 1096          1409  5                                 (local
; 1097          1410  5                                         char_dsc: descriptor;
; 1098          1411  5
; 1099          1412  5                                 char_dsc[0,0,32,0] = 1;
; 1100          1413  5                                 char_dsc[ptr] = char;
; 1101          1414  5                                 scr$put_screen(char_dsc,2,.command_line_pos);
; 1102          1415  4                                 command_line_pos = minu(.command_line_pos+1,79););
; 1103          1416  4
; 1104          1417  4                 [delete]:       ! We have a delete   Decrement the cursor
; 1105          1418  4                                 ! position and clear away the character.
; 1106          1419  4
; 1107          1420  5                                 (command_line_pos = maxu(.command_line_pos-1,2);
; 1108          1421  4                                 scr$erase_line(2,.command_line_pos););
; 1109          1422  4
; 1110          1423  4
; 1111          1424  4                         [ctrl_u].       ! We have a CTRL/U.  Reset the cursor to column
; 1112          1425  4                                         ! 2 and clear the line.
; 1113          1426  4
; 1114          1427  5                                         (command_line_pos = 2;
; 1115          1428  4                                         scr$erase_line(2,2););
; 1116          1429  4
; 1117          1430  4
; 1118          1431  4                         [ctrl_w]:       ! CTRL/W causes us to redraw the entire screen.
; 1119          1432  4
; 1120          1433  4                                         phn$fresh_screen(true);
; 1121          1434  4                         tes;
; 1122          1435  4
; 1123          1436  3         );
; 1124          1437  3
; 1125          1438  3         return;
; 1126          1439  2 );
```

```
; 1128          1440  2 ! Oh my God!  We have to display text in a participant's viewport.  This
; 1129          1441  2 ! is a very complex process, so I will try to comment well.
; 1130          1442  2 !
; 1131          1443  2 ! latest_line_dsc will point to the line text descriptor in the latest CTL.
; 1132          1444  2 ! It will always be updated so that is the case.
; 1133          1445  2
; 1134          1446  2 c = .dp[pub_g_ctlhead0];
; 1135          1447  2 latest_line_dsc = c[ctl_g_line];
; 1136          1448  2
; 1137          1449  2 ! It is not always the case that we can update the screen in this routine.
; 1138          1450  2 ! If a scrolling-type command is in progress, we can't.  If we can, though,
; 1139          1451  2 ! let's allocate a buffer for the screen package and position the cursor
; 1140          1452  2 ! at the current point.  From now on, we will assume that the cursor is
; 1141          1453  2 ! dynamically updated.
; 1142          1454  2
; 1143          1455  3 if not .phn$gv_scroller then (
; 1144          1456  3         scr$set_buffer(screen_buf);
; 1145          1457  3         phn$view_goodies(dp,cursor_pos);
; 1146          1458  3         scr$set_cursor(.cursor_pos[0],.cursor_pos[1]);
; 1147          1459  2 );
```

```
; 1149          1460  2 ! Now we will look at the text character by character and update the CTL
; 1150          1461  2 ! buffers to reflect the new conversation.  We always keep the CTL buffers
; 1151          1462  2 ! up-to-date so that if we ever have to refresh the screen (e.g., after
; 1152          1463  2 ! someone hangs up the phone), we can do it directly from the buffers.
; 1153          1464  2 ! We won't indent this loop because it is so big.
; 1154          1465  2
; 1155          1466  2 while dec (text_dsc[len]) geq 0 do (
; 1156          1467  2
; 1157          1468  2 char = ch$rchar_a(text_dsc[ptr]);
; 1158          1469  2
; 1159          1470  2 selectoneu .char of set
; 1160          1471  2 [%x'20' to %x'7e',
; 1161          1472  2  %x'80' to %x'ff']:
; 1162          1473  2
; 1163          1474  3                     ! We have a normal text character.  What we do depends upon
; 1164          1475  3                     ! whether it fits on the current line or not.
; 1165          1476  3
; 1166          1477  4                     (word_wrap = .latest_line_dsc[len] gequ 79;
; 1167          1478  5                     if .word_wrap then (
; 1168          1479  5
; 1169          1480  5                             ! We can't fit it on this line.  Find the last word on
; 1170          1481  5                             ! the line if there is one and remove it from the line.
; 1171          1482  5                             ! Save resulting cursor position for later use.  Make
; 1172          1483  5                             ! a new CTL with wrapped word.
; 1173          1484  5
; 1174          1485  5                             local
; 1175          1486  5                                     word_dsc: descriptor;
; 1176          1487  5
; 1177          1488  5                             word_dsc[0,0,32,0] = last_word(.latest_line_dsc);
; 1178          1489  5                             if .word_dsc[len] gegu 79 then
; 1179          1490  5                                     word_dsc[len] = 0;
; 1180          1491  5                             latest_line_dsc[len] = .latest_line_dsc[len] - .word_dsc[len];
; 1181          1492  5                             phn$view_goodies(dp,cursor_pos);
; 1182          1493  5                             word_dsc[ptr] = .latest_line_dsc[ptr] + .latest_line_dsc[len];
; 1183          1494  5                             add_new_ctl(word_dsc);
; 1184          1495  4                     );
; 1185          1496  4
; 1186          1497  4                     ! Now we can put the new character at the end of the (possibly
; 1187          1498  4                     ! new) current line.
; 1188          1499  4
; 1189          1500  4                     ch$wchar(.char,.latest_line_dsc[ptr]+.latest_line_dsc[len]);
; 1190          1501  3                     inc (latest_line_dsc[len]););
; 1191          1502  3
; 1192          1503  3
; 1193          1504  3 [ret]:              ! We have a carriage return.  Remember the current cursor
; 1194          1505  3                     ! position for later.  Add a new CTL with no text.
; 1195          1506  3
; 1196          1507  4                     (phn$view_goodies(dp,cursor_pos);
; 1197          1508  3                     add_new_ctl(0););
; 1198          1509  3
; 1199          1510  3
; 1200          1511  3 [tab]:              ! We have a tab character.  If we are not too far along on the
; 1201          1512  3                     ! line, we can honor it.  Compute the number of columns to the
; 1202          1513  3                     ! next tab stop and add that many blanks to the line.
; 1203          1514  3
; 1204          1515  4                     if .latest_line_dsc[len]+1 lequ 72 then (
; 1205          1516  4                             local
```

```
;  1206          1517  4                                    count: long;
;  1207          1518  4
;  1208          1519  4                            count = 8 - (.latest_line_dsc[len] mod 8);
;  1209          1520  4                            ch$fill(' ',.count,.latest_line_dsc[ptr]+.latest_line_dsc[len]);
;  1210          1521  4                            latest_line_dsc[len] = .latest_line_dsc[len] + .count;
;  1211          1522                      );
;  1212          1523
;  1213          1524
;  1214          1525      [delete]:       ! We have a delete character.  Just decrement the line length.
;  1215          1526
;  1216          1527                      latest_line_dsc[len] = max(.latest_line_dsc[len]-1,0);
;  1217          1528
;  1218          1529
;  1219          1530      [linefeed]:     ! We have a line feed character, which means we are to delete
;  1220          1531                      ! the last word on the line.  Just reduce the line length by
;  1221          1532                      ! the length of the word.
;  1222          1533
;  1223          1534                      latest_line_dsc[len] = .latest_line_dsc[len] - last_word(.latest_line_dsc);
;  1224          1535
;  1225          1536
;  1226          1537      [ctrl_u].       ! CTRL/U is used to clear the entire line.
;  1227          1538
;  1228          1539                      latest_line_dsc[len] = 0;
;  1229          1540
;  1230          1541
;  1231          1542      [formfeed]:     ! We have a CTRL/L.  Add a new CTL with no text, and remember
;  1232          1543                      ! it as the new top of the viewport.
;  1233          1544
;  1234          1545  4                   (add_new_ctl(0);
;  1235          1546  4                   dp[pub_l_topctl] = .dp[pub_l_ctlcount];);
;  1236          1547
;  1237          1548  3 tes;
```

```
; 1239          1549  3 ! Well, we have updated the CTL buffers.  Now, if we are allowed to, we
; 1240          1550  3 ! should update the screen.  This update logic is based on the fact that
; 1241          1551  3 ! the CTL buffers are already updated.  Each bit of logic also assumes that
; 1242          1552  3 ! the cursor is positioned correctly, which is why we positioned it in
; 1243          1553  3 ! the initialization code.
; 1244          1554  3
; 1245          1555  3 if not .phn$gv_scroller then
; 1246          1556  3
; 1247          1557  3 selectoneu .char of set
; 1248          1558  3 [%x'20' to %x'7e',
; 1249          1559  3  %x'80' to %x'ff']:
; 1250          1560  3
; 1251          1561  3                     ! We have a normal character.  What we do depends upon whether
; 1252          1562  3                     ! we word-wrapped or not.
; 1253          1563  3
; 1254          1564  4                     if .word_wrap then (
; 1255          1565  4
; 1256          1566  4                             ! We word-wrapped.  Clear the word from the end of the
; 1257          1567  4                             ! line.
; 1258          1568  4
; 1259          1569  4                             scr$erase_line(.cursor_pos[0],.cursor_pos[1]);
; 1260          1570  4
; 1261          1571  4                             ! Now we have to place the wrapped word on the next
; 1262          1572  4                             ! line.  If we are scrolling and are at the bottom
; 1263          1573  4                             ! of the viewport, then scroll.  Otherwise position
; 1264          1574  4                             ! on the next line and display the word.
; 1265          1575  4
; 1266          1576  4                             if .phn$gb_scroll and
; 1267          1577  4                                (.cursor_pos[0] gequ .dp[pub_w_viewline]+.dp[pub_w_viewsize]-2) then
; 1268          1578  4                                     phn$fresh_view(dp,false,false)
; 1269          1579  5                             else (
; 1270          1580  5                                     phn$view_goodies(dp,cursor_pos);
; 1271          1581  5                                     scr$put_screen(.latest_line_dsc,.cursor_pos[0],1);
; 1272          1582  5                                     scr$erase_line();
; 1273          1583  4                             );
; 1274          1584  4                     ) else (
; 1275          1585  4
; 1276          1586  4                             ! We did not word-wrap, so we can just display the
; 1277          1587  4                             ! character.
; 1278          1588  4
; 1279          1589  4                             local
; 1280          1590  4                                     char_dsc: descriptor;
; 1281          1591  4
; 1282          1592  4                             char_dsc[0,0,32,0] = 1;
; 1283          1593  4                             char_dsc[ptr] = .char;
; 1284          1594  4                             scr$put_screen(char_dsc);
; 1285          1595  3                     );
; 1286          1596  3
; 1287          1597  3
; 1288          1598  3 [ret]:
; 1289          1599  3                     ! We have a carriage return.  If we are scrolling and are at
; 1290          1600  3                     ! the bottom of the viewport, then scroll.  Otherwise position
; 1291          1601  3                     ! at the new line.
; 1292          1602  3
; 1293          1603  3                     if .phn$gb_scroll and
; 1294          1604  3                        (.cursor_pos[0] gequ .dp[pub_w_viewline]+.dp[pub_w_viewsize]-2) then
; 1295          1605  3                             phn$fresh_view(dp,false,false)
```

```
;  1296          1606  4                   else (
;  1297          1607  4                           phn$view_goodies(dp,cursor_pos);
;  1298          1608  4                           scr$erase_line(.cursor_pos[0],1);
;  1299          1609  3                   );
;  1300          1610  3
;  1301          1611  3
;  1302          1612  3 [tab]:            ! If we are not too far along on the line, we can tab.
;  1303          1613  3
;  1304          1614  3                   if .latest_line_dsc[len]+1 lequ 73 then
;  1305          1615  3                           scr$put_screen(describe(%char(tab)));
;  1306          1616  3
;  1307          1617  3
;  1308          1618  3 [delete,
;  1309          1619  3  linefeed,
;  1310          1620  3  ctrl_u]:        ! We have a character that deleted some text.  Position to
;  1311          1621  3                   ! the new cursor point and erase the rest of the line.
;  1312          1622  3
;  1313          1623  3                   (phn$view_goodies(dp,cursor_pos);
;  1314          1624  3                   scr$erase_line(.cursor_pos[0],.cursor_pos[1]););
;  1315          1625  3
;  1316          1626  3
;  1317          1627  3 [formfeed]:      ! We are clearing the viewport.  Just refresh all text lines.
;  1318          1628  3
;  1319          1629  3                   phn$fresh_view(dp,false,false);
;  1320          1630  3
;  1321          1631  3 tes;
```

```
;  1323          1632  3 ! The bell character (CTRL/G) is a very special case.  First of all,
;  1324          1633  3 ! we don't save it in the CTL buffer, because then it will beep every
;  1325          1634  3 ! time the viewport is scrolled.  Secondly, we always want to display it,
;  1326          1635  3 ! even if a scrolling command is in progress.
;  1327          1636  3
;  1328          1637  3 if .char eqlu bell then
;  1329          1638  3         scr$put_screen(describe(%char(bell)));
;  1330          1639  3
;  1331          1640  3 ! All done processing a character.  Continue until all text is done.
;  1332          1641  3
;  1333          1642  2 );
;  1334          1643  2
;  1335          1644  2
;  1336          1645  2 ! Now we can put the buffer to the screen, if allowable.
;  1337          1646  2
;  1338          1647  2 if not .phn$gv_scroller then
;  1339          1648  2         scr$put_buffer();
;  1340          1649  2
;  1341          1650  2 return;
;  1342          1651  2
;  1343          1652  1 end;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                                              09  0011C P.AAW:  .ASCII  <9>                                      ;
                                                  0011D        .BLKB   3                                        ;
                                        00000001  00120 P.AAV:  .LONG   1                                        ;
                                        00000000' 00124        .ADDRESS P.AAW                                    ;
                                              07  00128 P.AAY:  .ASCII  <7>                                      ;
                                                  00129        .BLKB   3                                        ;
                                        00000001  0012C P.AAX:  .LONG   1                                        ;
                                        00000000' 00130        .ADDRESS P.AAY                                    ;


                                                        .PSECT  $CODE$,NOWRT,2

                                                    0FFC 00000         .ENTRY  PHN$SHOW_TEXT, Save R2,R3,R4,R5,R6,R7,R8,- ; 1330
                                                                               R9,R10,R11                           ;
                                          5B    FDBE    CF  9E 00002         MOVAB   PHN$VIEW_GOODIES, R11        ;
                                          5E    FBD8    CE  9E 00007         MOVAB   -1064(SP), SP                ;
                                          58      04    AC  D0 0000C         MOVL    DISPLAY_PUB, R8              ; 1333
                                    10    AE    0400    8F  3C 00010         MOVZWL  #1024, SCREEN_BUF            ; 1387
                                    14    AE      18    AE  9E 00016         MOVAB   SCREEN_BUF+8, SCREEN_BUF+4   ;
                              F8    AD      08    BC        08  28 0001B         MOVC3   #8, @TEXT, TEXT_DSC          ; 1391
                                                          6C  95 00021         TSTB    (AP)                         ; 1396
                                                          08  13 00023         BEQL    1$                           ;
                                                    04    AC  D5 00025         TSTL    4(AP)                        ;
                                                          03  13 00028         BEQL    1$                           ;
                                                      00A3  31 0002A         BRW     12$                          ;
                                          50        F8    AD  3C 0002D 1$:     MOVZWL  TEXT_DSC, R0                 ; 1398
                                                          50  D7 00031         DECL    R0                           ;
                                    F8    AD              50  B0 00033         MOVW    R0, TEXT_DSC                 ;
                                                          50  D5 00037         TSTL    R0                           ;
                                                          01  18 00039         BGEQ    2$                           ;
```

```
                                          04 0003B        RET                                              ;
        04   AE          FC  BD  90 0003C 2$:     MOVB    @TEXT_DSC+4, CHAR                        ; 1400
                         FC  AD  D6 00041         INCL    TEXT_DSC+4                               ;
             50          04  AE  9A 00044         MOVZBL  CHAR, R0                                 ; 1402
             20              50  91 00048         CMPB    R0, #32                                  ; 1403
                             06  1F 0004B         BLSSU   3$                                       ;
        7E   8F              50  91 0004D         CMPB    R0, #126                                 ;
                             06  1B 00051         BLEQU   4$                                       ;
        80   8F              50  91 00053 3$:     CMPB    R0, #128                                 ;
                             33  1F 00057         BLSSU   7$                                       ;
        08   AE              01  D0 00059 4$:     MOVL    #1, CHAR_DSC                             ; 1412
        0C   AE          04  AE  9E 0005D         MOVAB   CHAR, CHAR_DSC+4                         ; 1413
                      0000'  CF  DD 00062         PUSHL   COMMAND_LINE_POS                         ; 1414
                             02  DD 00066         PUSHL   #2                                       ;
                         10  AE  9F 00068         PUSHAB  CHAR_DSC                                 ;
    00000000G  00            03  FB 0006B         CALLS   #3, SCR$PUT_SCREEN                       ;
 50     0000'  CF            01  C1 00072         ADDL3   #1, COMMAND_LINE_POS, R0                 ; 1415
    0000004F   8F            50  D1 00078         CMPL    R0, #79                                  ;
                             04  1B 0007F         BLEQU   5$                                       ;
             50          4F  8F  9A 00081         MOVZBL  #79, R0                                  ;
        0000'  CF            50  D0 00085 5$:     MOVL    R0, COMMAND_LINE_POS                     ;
                             A1  11 0008A 6$:     BRB     1$                                       ; 1402
        7F   8F              50  91 0008C 7$:     CMPB    R0, #127                                 ; 1417
                             19  12 00090         BNEQ    9$                                       ;
 50     0000'  CF            01  C3 00092         SUBL3   #1, COMMAND_LINE_POS, R0                 ; 1420
             02              50  D1 00098         CMPL    R0, #2                                   ;
                             03  1E 0009B         BGEQU   8$                                       ;
             50              02  D0 0009D         MOVL    #2, R0                                   ;
        0000'  CF            50  D0 000A0 8$:     MOVL    R0, COMMAND_LINE_POS                     ;
                      0000'  CF  DD 000A5         PUSHL   COMMAND_LINE_POS                         ; 1421
                             0C  11 000A9         BRB     10$                                      ;
             15              50  91 000AB 9$:     CMPB    R0, #21                                  ; 1424
                             12  12 000AE         BNEQ    11$                                      ;
        0000'  CF            02  D0 000B0         MOVL    #2, COMMAND_LINE_POS                     ; 1427
                             02  DD 000B5         PUSHL   #2                                       ; 1428
                             02  DD 000B7 10$:    PUSHL   #2                                       ;
    00000000G  00            02  FB 000B9         CALLS   #2, SCR$ERASE_LINE                       ;
                             C8  11 000C0         BRB     6$                                       ; 1402
             17              50  91 000C2 11$:    CMPB    R0, #23                                  ; 1431
                             C3  12 000C5         BNEQ    6$                                       ;
                             01  DD 000C7         PUSHL   #1                                       ; 1433
        FE01   CB            01  FB 000C9         CALLS   #1, PHN$FRESH_SCREEN                     ;
                             BA  11 000CE         BRB     6$                                       ; 1398
             59        0104  C8  9E 000D0 12$:    MOVAB   260(R8), R9                              ; 1446
             6E              69  D0 000D5         MOVL    (R9), C                                  ;
 56          6E              10  C1 000D8         ADDL3   #16, C, LATEST_LINE_DSC                  ; 1447
 1D     0000G  CF            01  E0 000DC         BBS     #1, PHN$GB_FLAGS, 13$                    ; 1455
                         10  AE  9F 000E2         PUSHAB  SCREEN_BUF                               ; 1456
    00000000G  00            01  FB 000E5         CALLS   #1, SCR$SET_BUFFER                       ;
                         F0  AD  9F 000EC         PUSHAB  CURSOR_POS                               ; 1457
                             58  DD 000EF         PUSHL   R8                                       ;
             6B              02  FB 000F1         CALLS   #2, PHN$VIEW_GOODIES                     ;
             7E          F0  AD  7D 000F4         MOVQ    CURSOR_POS, -(SP)                        ; 1458
    00000000G  00            02  FB 000F8         CALLS   #2, SCR$SET_CURSOR                       ;
             50          F8  AD  3C 000FF 13$:    MOVZWL  TEXT_DSC, R0                             ; 1466
                             50  D7 00103         DECL    R0                                       ;
        F8   AD              50  B0 00105         MOVW    R0, TEXT_DSC                             ;
```

```
                                50  D5 00109          TSTL    R0                                          ;
                                03  18 0010B          BGEQ    14$                                         ;
                             022C   31 0010D          BRW     48$                                         ;
            04    AE        FC    BD 90 00110 14$:    MOVB    @TEXT_DSC+4, CHAR                           ; 1468
                            FC    AD D6 00115          INCL    TEXT_DSC+4                                  ;
                  52        04    AE 9A 00118          MOVZBL  CHAR, R2                                    ; 1470
                  20              52 91 0011C          CMPB    R2, #32                                     ; 1471
                                  06 1F 0011F          BLSSU   15$                                         ;
            7E    8F              52 91 00121          CMPB    R2, #126                                    ;
                                  06 1B 00125          BLEQU   16$                                         ;
            80    8F              52 91 00127 15$:     CMPB    R2, #128                                    ;
                                  61 1F 0012B          BLSSU   20$                                         ;
                                  50 D4 0012D 16$:     CLRL    R0                                          ; 1477
          004F    8F              66 B1 0012F          CMPW    (LATEST_LINE_DSC), #79                      ;
                                  02 1F 00134          BLSSU   17$                                         ;
                                  50 D6 00136          INCL    R0                                          ;
                  5A              50 90 00138 17$:     MOVB    R0, WORD_WRAP                               ;
                  41              5A E9 0013B          BLBC    WORD_WRAP, 19$                              ; 1478
                                  56 DD 0013E          PUSHL   LATEST_LINE_DSC                             ; 1488
          01EA    CB              01 FB 00140          CALLS   #1, LAST_WORD                               ;
            08    AE              50 D0 00145          MOVL    R0, WORD_DSC                                ;
          004F    8F        08    AE B1 00149          CMPW    WORD_DSC, #79                               ; 1489
                                  03 1F 0014F          BLSSU   18$                                         ;
                            08    AE B4 00151          CLRW    WORD_DSC                                    ; 1490
                  66        08    AE A2 00154 18$:     SUBW2   WORD_DSC, (LATEST_LINE_DSC)                 ; 1491
                            F0    AD 9F 00158          PUSHAB  CURSOR_POS                                  ; 1492
                                  58 DD 0015B          PUSHL   R8                                          ;
                  6B              02 FB 0015D          CALLS   #2, PHN$VIEW_GOODIES                        ;
                  50              66 3C 00160          MOVZWL  (LATEST_LINE_DSC), R0                       ; 1493
            0C    AE        04 B640 9E 00163          MOVAB   @4(LATEST_LINE_DSC)[R0], WORD_DSC+4         ;
                                  5E DD 00169          PUSHL   SP                                          ; 1494
                            0C    AE 9F 0016B          PUSHAB  WORD_DSC                                    ;
                 0000G    CF        02 FB 0016E          CALLS   #2, PHN$MAKE_CTL                            ;
                  69        00    BE 0E 00173          INSQUE  @C, (R9)                                    ;
                          0100    C8 D6 00177          INCL    256(R8)                                     ;
          56              6E        10 C1 0017B          ADDL3   #16, C, LATEST_LINE_DSC                     ;
                  50              66 3C 0017F 19$:     MOVZWL  (LATEST_LINE_DSC), R0                       ; 1500
                  50        04    A6 C0 00182          ADDL2   4(LATEST_LINE_DSC), R0                      ;
                  60        04    AE 90 00186          MOVB    CHAR, (R0)                                  ;
                                  66 B6 0018A          INCW    (LATEST_LINE_DSC)                           ; 1501
                                  7C 11 0018C          BRB     25$                                         ; 1470
                  0D              .2 91 0018E 20$:     CMPB    R2, #13                                     ; 1504
                                  1; 12 00191          BNEQ    21$                                         ;
                            F0    AD 9F 00193          PUSHAB  CURSOR_POS                                  ; 1507
                                  58 DD 00196          PUSHL   R8                                          ;
                  6B              02 FB 00198          CALLS   #2, PHN$VIEW_GOODIES                        ;
                                  5E DD 0019B          PUSHL   SP                                          ; 1508
                                  7E D4 0019D          CLRL    -(SP)                                       ;
                 0000G    CF        02 FB 0019F          CALLS   #2, PHN$MAKE_CTL                            ;
                  69        00    BE 0E 001A4          INSQUE  @C, (R9)                                    ;
                          0100    C8 D6 001A8          INCL    256(R8)                                     ;
          56              6E        10 C1 001AC          ADDL3   #16, C, LATEST_LINE_DSC                     ;
                                  61 11 001B0          BRB     27$                                         ; 1470
                  09              52 91 001B2 21$:     CMPB    R2, #9                                      ; 1511
                                  31 12 001B5          BNEQ    22$                                         ;
                  50              66 3C 001B7          MOVZWL  (LATEST_LINE_DSC), R0                       ; 1515
                                  50 D6 001BA          INCL    R0                                          ;
```

```
               00000048    8F           50 D1 001BC          CMPL    R0, #72                        ;
                                        71 1A 001C3          BGTRU   29$                            ;
                           57           66 3C 001C5          MOVZWL  (LATEST_LINE_DSC), R7          ; 1519
7E        00               57           01 7A 001C8          EMUL    #1, R7, #0, -(SP)              ;
57        57               8E           08 7B 001CD          EDIV    #8, (SP)+, R7, R7              ;
          57               08           57 C3 001D2          SUBL3   R7, #8, COUNT                  ;
                           50           66 3C 001D6          MOVZWL  (LATEST_LINE_DSC), R0          ; 1520
                           50     04    A6 C0 001D9          ADDL2   4(LATEST_LINE_DSC), R0         ;
57        20               6E           00 2C 001DD          MOVC5   #0, (SP), #32, COUNT, (R0)     ;
                                        60    001E2                                                 ;
                           66           57 A0 001E3          ADDW2   COUNT, (LATEST_LINE_DSC)       ; 1521
                                        4E 11 001E6          BRB     29$                            ; 1515
               7F          8F           52 91 001E8 22$:     CMPB    R2, #127                       ; 1525
                                        0D 12 001EC          BNEQ    24$                            ;
                           50           66 3C 001EE          MOVZWL  (LATEST_LINE_DSC), R0          ; 1527
                           02           50 F4 001F1          SOBGEQ  R0, 23$                        ;
                                        50 D4 001F4          CLRL    R0                             ;
                           66           50 B0 001F6 23$:     MOVW    R0, (LATEST_LINE_DSC)          ;
                                        3B 11 001F9          BRB     29$                            ;
                           0A           52 91 001FB 24$:     CMPB    R2, #10                        ; 1530
                                        0C 12 001FE          BNEQ    26$                            ;
                                        56 DD 00200          PUSHL   LATEST_LINE_DSC                ; 1534
               01EA        CB           01 FB 00202          CALLS   #1, LAST_WORD                  ;
                           66           50 A2 00207          SUBW2   R0, (LATEST_LINE_DSC)          ;
                                        2A 11 0020A 25$:     BRB     29$                            ;
                           15           52 91 0020C 26$:     CMPB    R2, #21                        ; 1537
                                        04 12 0020F          BNEQ    28$                            ;
                                        66 B4 00211          CLRW    (LATEST_LINE_DSC)              ; 1539
                                        21 11 00213 27$:     BRB     29$                            ;
                           0C           52 91 00215 28$:     CMPB    R2, #12                        ; 1542
                                        1C 12 00218          BNEQ    29$                            ;
                                        5E DD 0021A          PUSHL   SP                             ; 1545
                                        7E D4 0021C          CLRL    -(SP)                          ;
               0000G       CF           02 FB 0021E          CALLS   #2, PHN$MAKE_CTL               ;
                           69     00    BE 0E 00223          INSQUE  @C, (R9)                       ;
                                0100    C8 D6 00227          INCL    256(R8)                        ;
          56               6E           10 C1 0022B          ADDL3   #16, C, LATEST_LINE_DSC        ;
               010C        C8   0100    C8 D0 0022F          MOVL    256(R8), 268(R8)               ; 1546
          5B   0000G       CF           01 E0 00236 29$:     BBS     #1, PHN$GB_FLAGS, 33$          ; 1555
                           50     04    AE 9A 0023C          MOVZBL  CHAR, R0                       ; 1557
                           20           50 91 00240          CMPB    R0, #32                        ; 1558
                                        06 1F 00243          BLSSU   30$                            ;
               7E          8F           50 91 00245          CMPB    R0, #126                       ;
                                        06 1B 00249          BLEQU   31$                            ;
               80          8F           50 91 0024B 30$:     CMPB    R0, #128                       ;
                                        56 1F 0024F          BLSSU   35$                            ;
                           45           5A E9 00251 31$:     BLBC    WORD_WRAP, 34$                 ; 1564
                           7E     F0    AD 7D 00254          MOVQ    CURSOR_POS, -(SP)              ; 1569
               00000000G   00           02 FB 00258          CALLS   #2, SCR$ERASE_LINE             ;
                           16   0000G   CF E9 0025F          BLBC    PHN$GB_SCROLL, 32$             ; 1576
                           50   00FE    C8 3C 00264          MOVZWL  254(R8), R0                    ; 1577
                           51   00FC    C8 3C 00269          MOVZWL  252(R8), R1                    ;
                           50           51 C0 0026E          ADDL2   R1, R0                         ;
                           50           02 C2 00271          SUBL2   #2, R0                         ;
                           50     F0    AD D1 00274          CMPL    CURSOR_POS, R0                 ;
                                        4B 1E 00278          BGEQU   36$                            ;
                                  F0    AD 9F 0027A 32$:     PUSHAB  CURSOR_POS                     ; 1580
```

```
                                58  DD 0027D        PUSHL   R8                                  ;          Sym
                     6B         02  FB 0027F        CALLS   #2, PHN$VIEW_GOODIES                ;          ---
                                01  DD 00282        PUSHL   #1                                  ; 1581     FOR
                        F0      AD  DD 00284        PUSHL   CURSOR_POS                          ;          FOR
                                56  DD 00287        PUSHL   LATEST_LINE_DSC                     ;          FOR
         00000000G   00         03  FB 00289        CALLS   #3, SCR$PUT_SCREEN                  ;          FOR
         00000000G   00         00  FB 00290        CALLS   #0, SCR$ERASE_LINE                  ; 1582     FOR
                                7F  11 00297 33$:   BRB     43$                                 ; 1564     FOR
                08   AE         01  D0 00299 34$:   MOVL    #1, CHAR_DSC                        ; 1592     FOR
                0C   AE    04   AE  9E 0029D        MOVAB   CHAR, CHAR_DSC+4                    ; 1593     FOR
                           08   AE  9F 002A2        PUSHAB  CHAR_DSC                            ; 1594     FOR
                                43  11 002A5        BRB     39$                                 ;          FOR
                     0D         50  91 002A7 35$:   CMPB    R0, #13                             ; 1598     FOR
                                27  12 002AA        BNEQ    38$                                 ;          FOR
                     16  0000G  CF  E9 002AC        BLBC    PHN$GB_SCROLL, 37$                  ; 1603     FOR
                     50  00FE   C8  3C 002B1        MOVZWL  254(R8), R0                         ; 1604     FOR
                     51  00FC   C8  3C 002B6        MOVZWL  252(R8), R1                         ;          FOR
                     50         51  C0 002BB        ADDL2   R1, R0                              ;          FOR
                     50         02  C2 002BE        SUBL2   #2, R0                              ;          FOR
                     50    F0   AD  D1 002C1        CMPL    CURSOR_POS, R0                      ;          FOR
                                58  1E 002C5 36$:   BGEQU   45$                                 ;          FOR
                           F0   AD  9F 002C7 37$:   PUSHAB  CURSOR_POS                          ; 1607     FOR
                                58  DD 002CA        PUSHL   R8                                  ;          FOR
                     6B         02  FB 002CC        CALLS   #2, PHN$VIEW_GOODIES                ;          FOR
                                01  DD 002CF        PUSHL   #1                                  ; 1608     FOR
                                3B  11 002D1        BRB     42$                                 ;          FOR
                     09         50  91 002D3 38$:   CMPB    R0, #9                              ; 1612     FOR
                                1B  12 002D6        BNEQ    40$                                 ;          FOR
                     50         66  3C 002D8        MOVZWL  (LATEST_LINE_DSC), R0               ; 1614     FOR
                                50  D6 002DB        INCL    R0                                  ;          FOR
         00000049    8F         50  D1 002DD        CMPL    R0, #73                             ;          FOR
                                42  1A 002E4        BGTRU   46$                                 ;          FOR
                         0000'  CF  9F 002E6        PUSHAB  P.AAV                               ; 1615     FOR
         00000000G   00         01  FB 002EA 39$:   CALLS   #1, SCR$PUT_SCREEN                  ;          FOR
                                35  11 002F1        BRB     46$                                 ; 1614     FOR
                     0A         50  91 002F3 40$:   CMPB    R0, #10                             ; 1618     FOR
                                0B  13 002F6        BEQL    41$                                 ;          FOR
                     15         50  91 002F8        CMPB    R0, #21                             ;          FOR
                                06  13 002FB        BEQL    41$                                 ;          FOR
                7F   8F         50  91 002FD        CMPB    R0, #127                            ;          LIB
                                17  12 00301        BNEQ    44$                                 ;          LIB
                           F0   AD  9F 00303 41$:   PUSHAB  CURSOR_POS                          ; 1623
                                58  DD 00306        PUSHL   R8                                  ;
                     6B         02  FB 00308        CALLS   #2, PHN$VIEW_GOODIES                ;          LIB
                           F4   AD  DD 0030B        PUSHL   CURSOR_POS+4                        ; 1624     LIB
                           F0   AD  DD 0030E 42$:   PUSHL   CURSOR_POS                          ;
         00000000G   00         02  FB 00311        CALLS   #2, SCR$ERASE_LINE                  ;
                                0E  11 00318 43$:   BRB     46$                                 ; 1557     OTS
                     0C         50  91 0031A 44$:   CMPB    R0, #12                             ; 1627     OTS
                                09  12 0031D        BNEQ    46$                                 ;          OTS
                                7E  7C 0031F 45$:   CLRQ    -(SP)                               ; 1629     OTS
                                58  DD 00321        PUSHL   R8                                  ;          OTS
             00DD    CB         03  FB 00323        CALLS   #3, PHN$FRESH_VIEW                  ;          OTS
                     07    04   AE  91 00328 46$:   CMPB    CHAR, #7                            ; 1637     OTS
                                0B  12 0032C        BNEQ    47$                                 ;
                         0000'  CF  9F 0032E        PUSHAB  P.AAX                               ; 1638
         00000000G   00         01  FB 00332        CALLS   #1, SCR$PUT_SCREEN                  ;
```

```
                              FDC3  31 00339 47$:   BRW     13$                          ; 1466
        07      0000G  CF           01  E0 0033C 48$:   BBS     #1, PHN$GB_FLAGS, 49$        ; 1647
           00000000G  00           00  FB 00342        CALLS   #0, SCR$PUT_BUFFER           ; 1648
                                       04 00349 49$:   RET                                  ; 1652

; Routine Size:  842 bytes,    Routine Base:  $CODE$ + 0669
```

_$2

Sym

PL1
PLI

PLI

PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI

PLI
PLI
PLI

PLI

PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI

PLI

PLI
PLI
PLI
PLI

PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI

```
;1345     1653  1 %sbttl 'PHN$SCROLL_PREP - Prepare to Scroll Information'
;1346     1654  1 !++
;1347     1655  1 ! Functional Description:
;1348     1656  1 !       This routine is called when a command wants to begin scrolling
;1349     1657  1 !       information onto the screen.  A command like HELP scrolls info
;1350     1658  1 !       into the screen beginning at line 5 and lasting for one or more
;1351     1659  1 !       lines.
;1352     1660  1 !
;1353     1661  1 ! Formal Parameters:
;1354     1662  1 !       heading         Address of descriptor of heading line to be
;1355     1663  1 !                       placed on line 5.  Optional.
;1356     1664  1 !
;1357     1665  1 ! Implicit Inputs:
;1358     1666  1 !       global data
;1359     1667  1 !
;1360     1668  1 ! Implicit Outputs:
;1361     1669  1 !       global data
;1362     1670  1 !
;1363     1671  1 ! Returned Value:
;1364     1672  1 !       none
;1365     1673  1 !
;1366     1674  1 ! Side Effects:
;1367     1675  1 !
;1368     1676  1 !--
;1369     1677  1
;1370     1678  1
;1371     1679  2 global routine phn$scroll_prep(heading): novalue = begin
;1372     1680  2
;1373     1681  2 builtin
;1374     1682  2         nullparameter;
;1375     1683  2
;1376     1684  2
;1377     1685  2 ! Erase the screen beginning at line 5.
;1378     1686  2
;1379     1687  2 scr$erase_page(5,1);
;1380     1688  2
;1381     1689  2 ! If we are to display a heading, do so.  Position cursor at line 7.
;1382     1690  2
;1383     1691  2 if not nullparameter(1) then
;1384     1692  2         scr$put_line(.heading,2);
;1385     1693  2
;1386     1694  2 ! If this is a VT100, it will look a lot sexier if we don't scroll off
;1387     1695  2 ! the top of the screen.
;1388     1696  2
;1389     1697  2 scr$set_scroll((if nullparameter(1) then 5 else 7), 24);
;1390     1698  2
;1391     1699  2 return;
;1392     1700  2
;1393     1701  1 end;


                                0000 00000          .ENTRY  PHN$SCROLL_PREP, Save nothing                ;1679
                              01  DD 00002          PUSHL   #1                                           ;1687
                              05  DD 00004          PUSHL   #5                                           ;
```

```
                    00000000G  00           02  FB 00006          CALLS   #2, SCR$ERASE_PAGE                 ;
                                            6C  95 0000D          TSTB    (AP)                               ; 1691
                                            11  13 0000F          BEQL    1$                                 ;
                                      04    AC  D5 00011          TSTL    4(AP)                              ;
                                            0C  13 00014          BEQL    1$                                 ;
                                            02  DD 00016          PUSHL   #2                                 ; 1692
                                      04    AC  DD 00018          PUSHL   HEADING                            ;
                    00000000G  00           02  FB 0001B          CALLS   #2, SCR$PUT_LINE                   ;
                                            18  DD 00022 1$:      PUSHL   #24                                ; 1697
                                            6C  95 00024          TSTB    (AP)                               ;
                                            05  13 00026          BEQL    2$                                 ;
                                      04    AC  D5 00028          TSTL    4(AP)                              ;
                                            04  12 0002B          BNEQ    3$                                 ;
                                            05  DD 0002D 2$:      PUSHL   #5                                 ;
                                            02  11 0002F          BRB     4$                                 ;
                                            07  DD 00031 3$:      PUSHL   #7                                 ;
                    00000000G  00           02  FB 00033 4$:      CALLS   #2, SCR$SET_SCROLL                 ;
                                                04 0003A          RET                                        ; 1701

; Routine Size:  59 bytes,    Routine Base:  $CODE$ + 09B3
```

```
;  1395          1702  1 %sbttl 'PHN$SCROLL_LINE - Scroll Line Onto Screen'                                                    Sym
;  1396          1703  1 !++                                                                                                    ---
;  1397          1704  1 ! Functional Description:                                                                            PLI
;  1398          1705  1 !       This routine is called to scroll a line of information (e.g., HELP)                       PLI
;  1399          1706  1 !       onto what is normally the viewport area of the screen.  The area                          PLI
;  1400          1707  1 !       must have been previously prepared by PHN$SCROLL_PREP.                                     PLI
;  1401          1708  1 !                                                                                                 PLI
;  1402          1709  1 ! Formal Parameters:                                                                               PLI
;  1403          1710  1 !       line            Address of descriptor of line to be scrolled.                             PLI
;  1404          1711  1 !                                                                                                 PLI
;  1405          1712  1 ! Implicit Inputs:                                                                                 PLI
;  1406          1713  1 !       global data                                                                               PLI
;  1407          1714  1 !                                                                                                 PLI
;  1408          1715  1 ! Implicit Outputs:                                                                                PLI
;  1409          1716  1 !       global data                                                                               PLI
;  1410          1717  1 !                                                                                                 PLI
;  1411          1718  1 ! Returned Value:                                                                                  PLI
;  1412          1719  1 !       none                                                                                      PLI
;  1413          1720  1 !                                                                                                 PLI
;  1414          1721  1 ! Side Effects:                                                                                   PLI
;  1415          1722  1 !                                                                                                 PLI
;  1416          1723  1 !--                                                                                               PLI
;  1417          1724  1                                                                                                   PLI
;  1418          1725  1                                                                                                   PLI
;  1419          1726  2 global routine phn$scroll_line(line): novalue = begin                                             PLI
;  1420          1727  2                                                                                                   PLI
;  1421          1728  2                                                                                                   PLI
;  1422          1729  2 ! Just display it with a carriage return/line feed.                                               PLI
;  1423          1730  2                                                                                                   PLI
;  1424          1731  2 scr$put_line(.line,1);                                                                            PLI
;  1425          1732  2                                                                                                   PLI
;  1426          1733  2 return;                                                                                           PLI
;  1427          1734  2                                                                                                   PLI
;  1428          1735  1 end;                                                                                             PLI

                                                0000 00000      .ENTRY  PHN$SCROLL_LINE, Save nothing                   ; 1726
                                              01  DD 00002      PUSHL   #1                                              ; 1731
                                          04  AC  DD 00004      PUSHL   LINE                                            ;
                              00000000G   00    02  FB 00007  CALLS   #2, SCR$PUT_LINE                                ;
                                                04  0000E      RET                                                     ; 1735

; Routine Size:  15 bytes,    Routine Base:  $CODE$ + 09EE

;  1429          1736  1                                                                                                   PLI
;  1430          1737  0 end eludom                                                                                       PLI

                                                            .EXTRN  LIB$SIGNAL

;                                   PSECT SUMMARY
```

Name	Bytes	Attributes
OWN	29	NOVEC, WRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
$PLIT$	308	NOVEC,NOWRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
$CODE$	2557	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

; Library Statistics

File	Symbols Total	Symbols Loaded	Symbols Percent	Pages Mapped	Processing Time
_255DUA28:[SYSLIB]STARLET.L32;1	9776	25	0	581	00:00.8

; COMMAND QUALIFIERS

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

```
; Size:          2557 code + 337 data bytes
; Run Time:         00:30.3
; Elapsed Time:     01:53.8
; Lines/CPU Min:     3443
; Lexemes/CPU-Min: 30087
; Memory Used:  260 pages
; Compilation Complete
```

PHONE
LIS

NETSLAVE
LIS

LINKSUBS
LIS

PHONEMSGS
LIS

STACKCMDS
LIS

TERMINAL
LIS

MISCCMDS
LIS

PUBSUBS
LIS

INPUT
LIS

PLICASERR
LIS

PLICONDIT
LIS

DATDEF
SDL

FILEDEF
IN

PLIRTDEF
MAR

PLICHRSTR
LIS

PLICHAR
LIS

PLICLOSE
LIS

PLIRTL

PLIFILDSP
IN

PLIBYTSIZ
LIS

PLIRTL
MAP

ENVCODES
IN

PLIBIT
LIS