```
PPPPPPPPPPPP     HHH         HHH     000000000     NNN         NNN  EEEEEEEEEEEEEEE
PPPPPPPPPPPP     HHH         HHH     000000000     NNN         NNN  EEEEEEEEEEEEEEE
PPPPPPPPPPPP     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     HHHHHHHHHHHHHHH  000         000  NNN   NNN   NNN  EEEEEEEEEEEE
PPPPPPPPPPPP     HHHHHHHHHHHHHHH  000         000  NNN   NNN   NNN  EEEEEEEEEEEE
PPPPPPPPPPPP     HHHHHHHHHHHHHHH  000         000  NNN   NNN   NNN  EEEEEEEEEEEE
PPP              HHH         HHH  000         000  NNN      NNNNNN  EEE
PPP              HHH         HHH  000         000  NNN      NNNNNN  EEE
PPP              HHH         HHH  000         000  NNN      NNNNNN  EEE
PPP              HHH         HHH  000         000  NNN         NNN  EEE
PPP              HHH         HHH  000         000  NNN         NNN  EEE
PPP              HHH         HHH  000         000  NNN         NNN  EEE
PPP              HHH         HHH     000000000     NNN         NNN  EEEEEEEEEEEEEEE
PPP              HHH         HHH     000000000     NNN         NNN  EEEEEEEEEEEEEEE
PPP              HHH         HHH     000000000     NNN         NNN  EEEEEEEEEEEEEEE
```

FILEID**TERMINAL

```
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
```

```
0 %title 'TERMINAL - Terminal Handling Routines'
0         module terminal (
1                         ident='V04-000') = begin
1
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1
1 !++
1 ! Facility:     VAX/VMS Telephone Facility, Terminal Handling Routines
1 !
1 ! Abstract:     This module provides all of the routines necessary to
1 !               handle the user's terminal.  This includes both the
1 !               keystroke handling routines and the screen formatting
1 !               functions.
1 !
1 !
1 ! Environment:  We do not check the status returns from RTL screen package
1 !               routines.  This saves code and time.
1 !
1 ! Author: Paul C. Anagnostopoulos, Creation Date: 24 November 1980
1 !
1 ! Modified By:
1 !
1 !       V03-003 PCA1020         Paul C. Anagnostopoulos 24-May-1983
1 !               Remove !!!TEMP!!! flags.
1 !               Support 8-bit characters.
1 !               Check SYS$INPUT to ensure that it is a terminal.
1 !
1 !       V03-002 MHB0086         Mark Bramhall           17-Jan-1983
1 !               Fixed FAO argument check in PHN$INFORM.
1 !
1 !       V03-001 PCA0048         Paul Anagnostopoulos    26-Mar-1982
1 !               Minor changes to convert from process name to user name.
1 !--
```

```
%sbttl 'Module Declarations'
!
! Libraries and Requires:
!

library 'sys$library:starlet.l32';
require 'phonereq';

!
! Table of Contents:
!

forward routine
        phn$init_term,
        phn$term_characteristic,
        phn$kbd_enable: novalue,
        phn$kbd_get: novalue,
        phn$fresh_screen,
        phn$inform: novalue,
        phn$review,
        phn$fresh_view: novalue,
        phn$fresh_hold: novalue,
        phn$show_text: novalue,
        phn$scroll_prep: novalue,
        phn$scroll_line: novalue;

!
! Macro Definitions:
!
!
! This macro is used in PHN$SHOW_TEXT to add a new CTL to a PUB.

macro add_new_ctl(text) =
(       phn$make_ctl(text,c);
        insque(.c,dp[pub_q_ctlhead0]);
        inc (dp[pub_l_ctlcount]);
        latest_line_dsc = c[ctl_q_line];
)
%;

!
! External References:
!

external routine
        phn$make_ctl,
        phn$queue_smb,
        scr$erase_line: addressing_mode(general),
        scr$erase_page: addressing_mode(general),
        scr$put_buffer: addressing_mode(general),
        scr$put_line:   addressing_mode(general),
        scr$put_screen: addressing_mode(general),
        scr$set_buffer: addressing_mode(general),
        scr$set_cursor: addressing_mode(general),
        scr$set_scroll: addressing_mode(general);

!
```

```
1 ! Own Variables:
1 !
1
1 ! The following variable contains the terminal channel number for obtaining
1 ! keystrokes:
1
1 own
1         terminal_channel: word;
1
1 ! The following variable is the current position on the command line.
1
1 own
1         command_line_pos: long;
1
1 ! We need a line of dashes.  No point in wasting space with lots of them.
1
1 bind
1         dashes = describe('--------------------------------------------------------------------------------')
```

```
%sbttl 'PHN$INIT_TERM - Initialize Users Terminal'
!++
! Functional Description:
!       This routine is called at initialization to prepare the user's
!       terminal.  This includes checking for valid terminal type, channel
!       assignment, and initial formatting of the screen.
!
! Formal Parameters:
!       none
!
! Implicit Inputs:
!       global data
!
! Implicit Outputs:
!       global data
!
! Returned Value:
!       phn$_badterm    Cannot use this type of terminal as a phone.
!
! Side Effects:
!
!--


global routine phn$init_term = begin

own
        device_class: long;
bind
        get_device_class = uplit(word(4),word(dvi$_devclass),
                                 long(device_class),
                                 long(0),
                                 long(0));

local
        status: long;
local
        local_described_buffer(terminal_number,64);


! We begin by determining the user's SYS$INPUT device.  We will need to
! read keystrokes from it.  Make sure it's a device we can use.

status = $getdvi(efn=phn$k_getjpiefn,
                 devnam=describe('SYS$INPUT'),
                 itmlst=get_device_class);
check (.status);
$waitfr(efn=phn$k_getjpiefn);
if .device_class nequ dc$_term then
        return phn$_inputterm;

status = $trnlog(lognam=describe('SYS$INPUT'),
                 rsllen=terminal_number,
                 rslbuf=terminal_number);
check (.status);
if ch$rchar(.terminal_number[ptr]) eqlu escape then (
        terminal_number[len] = .terminal_number[len] - 4;
```

```
  3             terminal_number[ptr] = .terminal_number[ptr] + 4;
  2 );
  2 if not phn$term_characteristic(terminal_number,tt$m_scope) then
  2             return phn$_badterm;
  2
  2 ! Now we can assign to the user's terminal.  This channel will be used
  2 ! for reading keystrokes, but not for output.
  2
P 2 status = $assign(devnam=terminal_number,
  2                  chan=terminal_channel);
  2 check (.status);
  2
  2 ! Now we have to clear the half-duplex bit in the terminal characteristics.
  2 ! This is because we have to share the terminal with the RTL screen package.
  2
  3 begin
  3 local
  3             terminal_chars: block[8,byte];
  3 bind
  3             device_info = terminal_chars[4,0,32,0]: block[4,byte];
  3
P 3 status = $qiow(chan=.terminal_channel,
P 3                func=io$_sensemode,
  3                p1=terminal_chars);
  3 check (.status);
  3 device_info[tt$v_halfdup] = false;
P 3 status = $qiow(chan=.terminal_channel,
P 3                func=io$_setmode,
  3                p1=terminal_chars);
  3 check (.status);
  2 end;
  2
  2 ! Put up the initial screen layout.
  2
  2 phn$fresh_screen(true);
  2
  2 ! Finally, we enable the keyboard so the user can enter a command.
  2
  2 phn$kbd_enable();
  2 return phn$_ok;
  2
  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   A5  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
```

```
1 %sbttl 'PHN$KBD_ENABLE - Enable Keyboard for Input'
1 !++
1 ! Functional Description:
1 !       This routine is called to enable the keyboard to interrupt
1 !       us with input.  We use an AST to inform us of pending input.
1 !       The AST routine sets up to read the input.
1 !
1 ! Formal Parameters:
1 !       none
1 !
1 ! Implicit Inputs:
1 !       global data
1 !
1 ! Implicit Outputs:
1 !       global data
1 !
1 ! Returned Value:
1 !       none
1 !
1 ! Side Effects:
1 !
1 !--
1
1
2 global routine phn$kbd_enable: novalue = begin
2
2 own
2       terminal_iosb: block[8,byte],
2       own_described_buffer(single_key,1);
2
2 local
2       status;
2
2
2 ! The AST routine reads the pending keystroke and passes it along to a
2 ! steering message routine.  The steering routine then flushes all
2 ! outstanding keystrokes at the time it is called,  thus causing us to
2 ! "batch" the keystrokes.  It is hoped that the delay caused by this scheme
2 ! will be proportional to the system load.
2
3 routine kbd_ast: novalue = begin
3
3 check (.terminal_iosb[0,0,16,0]);
3
3 phn$queue_smb(smb__kbd_get,single_key);
3 return;
3
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
```

```
%sbttl 'PHN$KBD_GET - Get All Outstanding Keystrokes'
!++
! Functional Description:
!       This steering message routine is invoked when the user has typed
!       a key at the terminal.  It collects any additional keystrokes
!       from the typeahead buffer and queues them in a batch to the
!       input interpreter.
!
! Formal Parameters:
!       single_key      The address of a descriptor of the single keystroke
!                       that started this mess.
!
! Implicit Inputs:
!       global data
!
! Implicit Outputs:
!       global data
!
! Returned Value:
!       none
!
! Side Effects:
!
!--


global routine phn$kbd_get(single_key): novalue = begin

bind
        single_key_dsc = .single_key: descriptor;

local
        status: long,
        terminal_iosb: block[8,byte];
local
        local_described_buffer(input,80);


! We begin by moving the single key that was passed to us into our input
! buffer.

ch$move(1,.single_key_dsc[ptr],.input[ptr]);

! Now we can flush the typeahead buffer and add it to our input buffer,
! forming a batch of characters.  NOTE that we specify no terminators.

status = $qiow(efn=phn$k_kbdefn,
                chan=.terminal_channel,
                func=io$_readvblk + io$m_noecho + io$m_nofiltr + io$m_timed,
                iosb=terminal_iosb,
                p1=.input[ptr]+1,
                p2=79,
                p3=0,
                p4=uplit long(0,0));
check (.status);
if .terminal_iosb[0,0,16,0] nequ ss$_timeout then
        check (.terminal_iosb[0,0,16,0]);
```

```
2
2 ! Fill in the character count in the input buffer and queue a steering
2 ! message to route the input.
2
2 input[len] = 1 + .terminal_iosb[2,0,16,0];
2 phn$queue_smb(smb__kbd_route,input);
2
2 ! Re-enable the keyboard for further input.
2
2 phn$kbd_enable();
2 return;
2
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                                      ;
       00000000G  00             0C   FB 00043               CALLS   #12, SYS$QIOW                           ;
                  05             50   E8 0004A               BLBS    STATUS, 1$                              ; 0728
                                 50   DD 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
```

```
1 %sbttl 'PHN$FRESH_SCREEN - Refresh the Screen'
1 !++
1 ! Functional Description:
1 !       This routine is called to refresh part or all of the screen, as
1 !       specified by the parameter.
1 !
1 ! Formal Parameters:
1 !       top_too         If false, we refresh the viewports.  If true, we
1 !                       also refresh the top four lines.
1 !
1 ! Implicit Inputs:
1 !       global data
1 !
1 ! Implicit Outputs:
1 !       global data
1 !
1 ! Returned Value:
1 !       phn$_noports    Too many viewports required for active people.
1 !
1 ! Side Effects:
1 !
1 !--
1
1
2 global routine phn$fresh_screen(top_too) = begin
2
2 local
2       status: long,
2       p: ref pub;
2
2
2 ! If a scrolling-type command is in progress (e.g., HELP), then we can't
2 ! refresh or we'll clobber whatever is being scrolled.
2
2 if .phn$gv_scroller then
2       return phn$_ok;
2
2 ! We begin by recalculating the viewport parameters to assign all active
2 ! participants a viewport.  If this cannot be done, we return an error
2 ! status without touching the screen.
2
2 status = phn$review();
2 if .status nequ phn$_ok then
2       return .status;
2
2 ! If we are to refresh the top four lines of the screen, then do it.
2 ! This includes a heading, today's date, the switch hook character on the
2 ! command line, and a line of dashes.
2
3 if .top_too then (
3       scr$erase_page(1,1);
3
3       scr$put_screen(describe(' VAX/VMS Phone Facility '),1,28,%b'0010');
3
4       begin
4       local
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     0002D          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
```

```
%sbttl 'PHN$INFORM - Display Message to User'
!++
! Functional Description:
!       This routine is called to display a message to the user.  All messages
!       are defined in the message file and referenced by status codes.
!       $FAO arguments can also be passed, and they will be filled into
!       the message text.
!
! Formal Parameters:
!       status_code     The message status code.  Zero means to clear the line.
!       fao1...         Zero or more $FAO arguments to be filled in.
!
! Implicit Inputs:
!       global data
!
! Implicit Outputs:
!       global data
!
! Returned Value:
!       none
!
! Side Effects:
!
!--


global routine phn$inform(status_code,fao1): novalue = begin

local
        status: long,
        byte_array: vector[4,byte],
        msg_text_ptr: ref descriptor;

builtin
        actualcount,
        nullparameter;


! If a scrolling-type command (e.g., HELP) is in progress, we can't display
! the message.  However, we will ring the bell to tell the user something's up.

if .phn$gv_scroller then (
        scr$put_screen(describe(%char(bell)));
        return;
);

! Now, if we are only to clear the message line, do it.

if nullparameter(1) then (
        scr$erase_line(3,1);
        return;
);

! Now we can obtain the message text and fill in the $FAO arguments.  If
! the message was not defined with enough $FAO codes, we don't fill anything
! into it.

```

```
;     3 begin
;     3 local
;     3         local_described_buffer(msg_buf,79);
;     3
;   P 3 status = $getmsg(msgid=.status_code,
;   P 3                  msglen=msg_buf,
;   P 3                  bufadr=msg_buf,
;   P 3                  flags=%b'0001',
;     3                  outadr=byte_array);
;     3 check (.status);
;     3 msg_text_ptr = msg_buf;
;     3
;     4 begin
;     4 local
;     4         local_described_buffer(fao_buf,79);
;     4
;     5 if (.byte_array[1] neq 0) and (actualcount() gtru .byte_array[1]) then (
;   P 5         status = $faol(ctrstr=msg_buf,
;   P 5                        outlen=fao_buf,
;   P 5                        outbuf=fao_buf,
;     5                        prmlst=faol);
;     5         check (.status);
;     5         msg_text_ptr = fao_buf;
;     4 );
;     4
;     4 ! Now we can clear the message line and display the message text.
;     4
;     4 scr$erase_line(3,1);
;     4 scr$put_screen(.msg_text_ptr);
;     3 end;
;     2 end;
;     2
;     2 ! Set the flag that says there's a message on the message line.
;     2
;     2 phn$gv_message = true;
;     2
;     2 return;
;     2
;     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
```

```
%sbttl 'PHN$REVIEW - Recalculate Viewport Parameters'
!++
! Functional Description:
!       This routine is called to recalculate the viewport parameters
!       in each of the existing PUBs.
!
! Formal Parameters:
!       none
!
! Implicit Inputs:
!       global data
!
! Implicit Outputs:
!       global data
!
! Returned Value:
!       phn$_noports    There is not enought room on the screen for all
!                       required viewports.
!
! Side Effects:
!
!--


global routine phn$review = begin

local
        active_count: long, held_count: long,
        viewport_size: long, viewport_line: long,
        p: ref pub;

literal
        max_viewports = 20/pub_k_minlines;


! First we have to scan all the PUBs and make two counts:  the number of
! PUBs we don't have on hold and the number we do have on hold.

active_count = held_count = 0;
p = .phn$gq_pubhead[0];
while .p neqa phn$gq_pubhead do (
        if not .p[pub_v_temporary] then
                if .p[pub_v_uhaveheld] then
                        inc (held_count)
                else
                        inc (active_count);

        p = .p[pub_l_flink];
);

! We need a viewport for at least all the active units.  Make sure the
! required viewports won't be too small.

if .active_count gtr max_viewports then
        return phn$_noports;

! 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                        ;
                           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
```

```
1 %sbttl 'PHN$VIEW_GOODIES - Calculate Viewport Information'
1 !++
1 ! Functional Description:
1 !       This routine is called to calculate various information about a
1 !       viewport.  The information that it returns is determined by the
1 !       parameters.
1 !
1 ! Formal Parameters:
1 !       goodies_pub     The address of the PUB describing the viewport.
1 !
1 !       cursor_pos      The address of a vector of 2 longwords.  We return
1 !                       the line and column numbers specifying the current
1 !                       position of the cursor.
1 !
1 !       line_numbers    The address of a longword vector.  We return the line
1 !                       numbers of the text lines in the viewport.  The zeroth
1 !                       entry contains the line with the latest CTL.
1 !
1 !       text_dscs       The address of a longword vector.  We return the
1 !                       address of the descriptor of the text to go on the line
1 !                       specified in the corresponding line_numbers entry.
1 !
1 ! Implicit Inputs:
1 !       global data
1 !
1 ! Implicit Outputs:
1 !       global data
1 !
1 ! Returned Value:
1 !       none
1 !
1 ! Side Effects:
1 !
1 !--
1
1
1 global routine phn$view_goodies(goodies_pub,cursor_pos,line_numbers,text_dscs):
2                                                       novalue = begin
2
2 bind
2       gp = .goodies_pub: pub;
2
2 bind
2       latest_ctl = .gp[pub_q_ctlhead0]: ctl,
2       latest_text_dsc = latest_ctl[ctl_q_line]: descriptor;
2
2 local
2       latest_line: long,
2       text_count: long,
2       c: ref ctl;
2
2 builtin
2       nullparameter;
2
2
2 ! First let's get some terminology straight:
2 !
```

```
!       oldest CTL      The CTL at the end of the CTL list.  It is the CTL
!                       representing the oldest line of the conversation.
!
!       latest CTL      The CTL at the front of the CTL list.  It represents
!                       the current line of the conversation.
!
!       top CTL         The CTL representing the line that would be at the
!                       top of the viewport if the viewport were infinitely
!                       large.  Usually the oldest CTL, but not if the user
!                       has typed a CTRL/L.

! We begin by calculating the line on which the latest CTL text should be
! placed.  There are two algorithms.

if .phn$gb_scroll then

        ! If we are scrolling, then the latest CTL goes on the bottom line
        ! of the viewport, unless we are not very many CTLs from the top.

        latest_line = .gp[pub_w_viewline] + 1 +
                      minu(.gp[pub_w_viewsize] - 3,
                           .gp[pub_l_ctlcount] - .gp[pub_l_topctl])

else

        ! If we are wrapping, then the latest CTL goes on a line determined
        ! by the number of lines from the top MOD the size of the viewport.

        latest_line = .gp[pub_w_viewline] + 1 +
                      (.gp[pub_l_ctlcount] - .gp[pub_l_topctl]) mod
                           (.gp[pub_w_viewsize] - 2);

! Now if the caller wants to know the current position, fill it in.

if not nullparameter(2) then (
        bind
                cursor_pos_array = .cursor_pos: vector[2,long];

        cursor_pos_array[0] = .latest_line;
        cursor_pos_array[1] = .latest_text_dsc[len] + 1;
);

! Now if the caller only wanted to know the current position, we're done.

if nullparameter(3) or nullparameter(4) then
        return;

! We are to fill in the line number and text descriptor arrays.
! Now we calculate how many text lines we will display in the viewport.
! This is equal to the number of text lines in the viewport, unless we
! are fewer than that from the top.

text_count = minu(.gp[pub_l_ctlcount] - .gp[pub_l_topctl] + 1,
                  .gp[pub_w_viewsize] - 2);

! Now we will fill in as many array entries as there are lines in the
! viewport.  Each line number entry gets the corresponding viewport line
```

```
2 ! number.  Each text descriptor either gets the address of a descriptor
2 ! from a CTL, or the address of a null descriptor if the line is to be
2 ! blanked.
2
3 begin
3 bind
3         line_numbers_array = .line_numbers: vector[pub_k_maxlines-2,long],
3         text_dscs_array = .text_dscs: vector[pub_k_maxlines-2,long];
3
3 c = latest_ctl;
4 incru i from 0 to .gp[pub_w_viewsize]-2-1 do (
4         line_numbers_array[.i] = .latest_line;
4         dec (latest_line);
4         if .latest_line lequ .gp[pub_w_viewline] then
4                 latest_line = .gp[pub_w_viewline] + .gp[pub_w_viewsize]  - 2;
4
5         if dec (text_count) geq 0 then (
5                 text_dscs_array[.i] = c[ctl_q_line];
5                 c = .c[ctl_l_flink];
4         ) else
4                 text_dscs_array[.i] = describe('');
3 );
2 end;
2
2 return;
2
1 end;
```

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

                                   000CC P.AAO:         .BLKB   0
                         00000000  000CC P.AAN:         .LONG   0
                         00000000' 000D0                .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
```

```
%sbttl 'PHN$FRESH_VIEW - Refresh a Viewport'
!++
! Functional Description:
!       This routine is called to refresh a specific viewport on the screen.
!
! Formal Parameters:
!       fresh_pub         Address of PUB representing viewport to be refreshed.
!       viewport_clear    True if viewport is known to be erased upon call;
!                         false otherwise.  This is used for optimization.
!       top_bottom_too    True if entire viewport is to be refreshed; false if
!                         only the text lines.
!
! Implicit Inputs:
!       global data
!
! Implicit Outputs:
!       global data
!
! Returned Value:
!       none
!
! Side Effects:
!
!--


global routine phn$fresh_view(fresh_pub,viewport_clear,top_bottom_too): novalue = begin

bind
        fp = .fresh_pub: pub,
        fresh_tsb = fp[pub_b_tsb]: tsb;

local
        cursor_pos: vector[2,long],
        line_numbers: vector[pub_k_maxlines-2,long],
        text_dscs:    vector[pub_k_maxlines-2,long],
        c: ref ctl;


! If a scrolling-type command (e.g., HELP) is in progress, we can't refresh
! the viewport.

if .phn$gv_scroller then
        return;

! If we are to refresh the entire viewport, then we have to clear the first
! line and center the user name on it, display the hold indicators, and
! throw up a lines of dashes at the bottom.

if .top_bottom_too then (
        begin
        bind
                spec_dsc = fresh_tsb[tsb_q_tkndsc,0]: descriptor;

        if not .viewport_clear then
                scr$erase_line(.fp[pub_w_viewline],1);
        scr$put_screen(spec_dsc,.fp[pub_w_viewline],(81-.spec_dsc[len])/2,%b'0001');
```

```
3               end;
3
3               phn$fresh_hold(fp);
3
3               scr$put_screen(dashes,.fp[pub_w_viewline]+.io[pub_w_viewsize]-1,1);
2       );
2
2       ! Now we want to refresh the text lines themselves.  We call a routine
2       ! to tell us which lines and what text goes on them.
2
2       phn$view_goodies(fp,cursor_pos,line_numbers,text_dscs);
2
2       ! Now we loop through the arrays, acting on each active entry.  The
2       ! line number array tells us which viewport line the text goes on.  The
2       ! text descriptor array tells us what text to put there.
2
3       incru i from 0 to .fp[pub_w_viewsize]-2-1 do (
3               bind
3                       line_dsc = .text_dscs[.i]: descriptor;
3
4               if not .viewport_clear or .line_dsc[len] nequ 0 then (
4                       scr$erase_line(.line_numbers[.i],1);
4                       scr$put_screen(line_dsc);
3               );
2       );
2
2       ! Finally, position the cursor at it's current point in the viewport.
2
2       scr$set_cursor(.cursor_pos[0],.cursor_pos[1]);
2
2       return;
2
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       00FE  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$VIEW_GOODIES                           ;
                  54      00FC   C2  3C 0007B         MOVZWL  252(R2), R4                                    ; 1232
                  54             03  C2 00080         SUBL2   #3, R4                                         ;
                  55        08   AC  D2 00083         MCOML   VIEWPORT_CLEAR, R5                             ; 1236
                                 52  D4 00087         CLRL    I                                              ;
                                 1B  11 00089         BRB     7$                                             ;
                  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   AE  DD 000AB         PUSHL   CURSOR_POS+4                                   ; 1244
                            44   AE  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
```

```
%sbttl 'PHN$FRESH_HOLD - Display Hold Indicators'
!++
! Functional Description:
!	This routine is called to display the hold indicators on the first
!	line of a viewport.  One indicator says whether or not we have the
!	person on hold, the other says whether or not they have us on hold.
!
! Formal Parameters:
!	fresh_pub	The address of the PUB representing the viewport.
!
! Implicit Inputs:
!	global data
!
! Implicit Outputs:
!	global data
!
! Returned Value:
!	none
!
! Side Effects:
!
!--


global routine phn$fresh_hold(fresh_pub): novalue = begin

bind
	fp = .fresh_pub: pub;


! If a scrolling-type command (e.g., HELP) is in progress, we can't display
! the indicators.

if .phn$gv_scroller then
	return;

! Display an indicator to say if we have them on hold.

scr$put_screen(if .fp[pub_v_uhaveheld] then describe('(YOU HAVE HELD)')
					else describe('               '),
		.fp[pub_w_viewline],1);

! Display an indicator to say if they have us on hold.

if .fp[pub_v_hasuheld] then
	scr$put_screen(describe('(HAS YOU HELD)'),.fp[pub_w_viewline],66)
else
	scr$erase_line(.fp[pub_w_viewline],66);

return;

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_PDB, R2                            ; 1276
           40     0000G CF         01   E0 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
```

```
1 %sbttl 'PHN$SHOW_TEXT - Display Text on Screen'
1 !++
1 ! Functional Description:
1 !       This routine is called to display text on the screen.  It can be
1 !       passed the address of a PUB describing the viewport in which the
1 !       text is to be displayed.  It can also be passed an address of zero,
1 !       in which case the text is displayed on the command line.
1 !
1 ! Formal Parameters:
1 !       display_pub     The address of the PUB describing the viewport to
1 !                       contain the text (viewport parameters are assumed to
1 !                       be filled in).  An address of zero means the text
1 !                       goes on the command line.
1 !       text            Address of the descriptor of the text.
1 !
1 ! Implicit Inputs:
1 !       global data
1 !
1 ! Implicit Outputs:
1 !       global data
1 !
1 ! Returned Value:
1 !       none
1 !
1 ! Side Effects:
1 !
1 !--
1
1
2 global routine phn$show_text(display_pub,text): novalue = begin
2
2 bind
2       dp = .display_pub: pub;
2
2 local
2       text_dsc: descriptor,
2       char: byte,
2       c: ref ctl,
2       latest_line_dsc: ref descriptor,
2       cursor_pos: vector[2,long],
2       word_wrap: byte;
2
2 builtin
2       nullparameter;
```

```
! This routine is called from the screen update logic.  It determines the
! length of the last word in the text and returns that length.  This
! information is needed when we are to word-wrap or delete the last word
! on the line.

routine last_word(text) = begin

bind
        text_dsc = .text: descriptor,
        text_string = .text_dsc[ptr]: vector[,byte];

local
        i: long;


! First we scan backwards for the last non-blank in the text.  This marks
! the end of the last word.

i = (
        decr i from .text_dsc[len]-1 to -1 do (
        if .i eql -1 then exitloop .i;
        if .text_string[.i] nequ ' ' then exitloop .i;
        )
);

! Now we scan backwards for the preceeding blank in the text.  This marks
! the beginning of the last word.

i = (
        decr i from .i to -1 do (
        if .i eql -1 then exitloop .i;
        if .text_string[.i] eqlu ' ' then exitloop .i;
        )
);

! Return the length of the last word.

return .text_dsc[len] - .i - 1;
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$:     DECL     I                                ; 1364
        FFFFFFFF  8F            50  D1 0001B         CMPL     I, #-1                           ;
                                E7  18 00022         BGEQ     1$                               ;
```

```
                      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);
```

```
2 ! We initialize by copying the descriptor of the text because we are going
2 ! go clobber it.
2
2 ch$move(8,.text,text_dsc);
2
2 ! Now we see if we are to display text on the command line.  If so, we
2 ! process each text character and handle it appropriately.
2
3 if nullparameter(1) then (
3
4         while dec (text_dsc[len]) geq 0 do (
4
4                 char = ch$rchar_a(text_dsc[ptr]);
4
4                 selectoneu .char of set
4                 [%x'20' to %x'7e',
4                  %x'80' to %x'ff']:
4
4                                         ! We have a normal character.  Place it on the
4                                         ! command line and advance the cursor position.
4
5                                         (local
5                                                 char_dsc: descriptor;
5
5                                         char_dsc[0,0,32,0] = 1;
5                                         char_dsc[ptr] = char;
5                                         scr$put_screen(char_dsc,2,.command_line_pos);
4                                         command_line_pos = minu(.command_line_pos+1,79););
4
4                 [delete]:               ! We have a delete.  Decrement the cursor
4                                         ! position and clear away the character.
4
5                                         (command_line_pos = maxu(.command_line_pos-1,2);
4                                         scr$erase_line(2,.command_line_pos););
4
4
4                 [ctrl_u].               ! We have a CTRL/U.  Reset the cursor to column
4                                         ! 2 and clear the line.
4
5                                         (command_line_pos = 2;
4                                         scr$erase_line(2,2););
4
4
4                 [ctrl_w]:               ! CTRL/W causes us to redraw the entire screen.
4
4                                         phn$fresh_screen(true);
4                 tes;
4
3         );
3
3         return;
2 );
```

```
2 ! Oh my God!  We have to display text in a participant's viewport.  This
2 ! is a very complex process, so I will try to comment well.
2 !
2 ! latest_line_dsc will point to the line text descriptor in the latest CTL.
2 ! It will always be updated so that is the case.
2
2 c = .dp[pub_q_ctlhead0];
2 latest_line_dsc = c[ctl_q_line];
2
2 ! It is not always the case that we can update the screen in this routine.
2 ! If a scrolling-type command is in progress, we can't.  If we can, though,
2 ! let's allocate a buffer for the screen package and position the cursor
2 ! at the current point.  From now on, we will assume that the cursor is
2 ! dynamically updated.
2
3 if not .phn$gv_scroller then (
3         scr$set_buffer(screen_buf);
3         phn$view_goodies(dp,cursor_pos);
3         scr$set_cursor(.cursor_pos[0],.cursor_pos[1]);
2 );
```

```
! Now we will look at the text character by character and update the CTL
! buffers to reflect the new conversation.  We always keep the CTL buffers
! up-to-date so that if we ever have to refresh the screen (e.g., after
! someone hangs up the phone), we can do it directly from the buffers.
! We won't indent this loop because it is so big.

while dec (text_dsc[len]) geq 0 do (

char = ch$rchar_a(text_dsc[ptr]);

selectoneu .char of set
[%x'20' to %x'7e',
 %x'80' to %x'ff']:

                        ! We have a normal text character.  What we do depends upon
                        ! whether it fits on the current line or not.

                        (word_wrap = .latest_line_dsc[len] gequ 79;
                        if .word_wrap then (

                                ! We can't fit it on this line.  Find the last word on
                                ! the line if there is one and remove it from the line.
                                ! Save resulting cursor position for later use.  Make
                                ! a new CTL with wrapped word.

                                local
                                        word_dsc: descriptor;

                                word_dsc[0,0,32,0] = last_word(.latest_line_dsc);
                                if .word_dsc[len] gequ 79 then
                                        word_dsc[len] = 0;
                                latest_line_dsc[len] = .latest_line_dsc[len] - .word_dsc[len];
                                phn$view_goodies(dp,cursor_pos);
                                word_dsc[ptr] = .latest_line_dsc[ptr] + .latest_line_dsc[len];
                                add_new_ctl(word_dsc);
                        );

                        ! Now we can put the new character at the end of the (possibly
                        ! new) current line.

                        ch$wchar(.char,.latest_line_dsc[ptr]+.latest_line_dsc[len]);
                        inc (latest_line_dsc[len]););


[ret]:                  ! We have a carriage return.  Remember the current cursor
                        ! position for later.  Add a new CTL with no text.

                        (phn$view_goodies(dp,cursor_pos);
                        add_new_ctl(0););


[tab]:                  ! We have a tab character.  If we are not too far along on the
                        ! line, we can honor it.  Compute the number of columns to the
                        ! next tab stop and add that many blanks to the line.

                        if .latest_line_dsc[len]+1 lequ 72 then (
                                local
```

```
                                count: long;

                        count = 8 - (.latest_line_dsc[len] mod 8);
                        ch$fill(' ',.count,.latest_line_dsc[ptr]+.latest_line_dsc[len]);
                        latest_line_dsc[len] = .latest_line_dsc[len] + .count;
                );


[delete]:       ! We have a delete character.  Just decrement the line length.

                latest_line_dsc[len] = max(.latest_line_dsc[len]-1,0);


[linefeed]:     ! We have a line feed character, which means we are to delete
                ! the last word on the line.  Just reduce the line length by
                ! the length of the word.

                latest_line_dsc[len] = .latest_line_dsc[len] - last_word(.latest_line_dsc);


[ctrl_u]:       ! CTRL/U is used to clear the entire line.

                latest_line_dsc[len] = 0;


[formfeed]:     ! We have a CTRL/L.  Add a new CTL with no text, and remember
                ! it as the new top of the viewport.

                (add_new_ctl(0);
                dp[pub_l_topctl] = .dp[pub_l_ctlcount];);

tes;
```

```
! Well, we have updated the CTL buffers.  Now, if we are allowed to, we
! should update the screen.  This update logic is based on the fact that
! the CTL buffers are already updated.  Each bit of logic also assumes that
! the cursor is positioned correctly, which is why we positioned it in
! the initialization code.

if not .phn$gv_scroller then

selectoneu .char of set
[%x'20' to %x'7e',
 %x'80' to %x'ff']:

                        ! We have a normal character.  What we do depends upon whether
                        ! we word-wrapped or not.

                        if .word_wrap then (

                                ! We word-wrapped.  Clear the word from the end of the
                                ! line.

                                scr$erase_line(.cursor_pos[0],.cursor_pos[1]);

                                ! Now we have to place the wrapped word on the next
                                ! line.  If we are scrolling and are at the bottom
                                ! of the viewport, then scroll.  Otherwise position
                                ! on the next line and display the word.

                                if .phn$gb_scroll and
                                   (.cursor_pos[0] gequ .dp[pub_w_viewline]+.dp[pub_w_viewsize]-2) then
                                        phn$fresh_view(dp,false,false)
                                else (
                                        phn$view_goodies(dp,cursor_pos);
                                        scr$put_screen(.latest_line_dsc,.cursor_pos[0],1);
                                        scr$erase_line();
                                );
                        ) else (

                                ! We did not word-wrap, so we can just display the
                                ! character.

                                local
                                        char_dsc: descriptor;

                                char_dsc[0,0,32,0] = 1;
                                char_dsc[ptr] = char;
                                scr$put_screen(char_dsc);
                        );


[ret]:
                        ! We have a carriage return.  If we are scrolling and are at
                        ! the bottom of the viewport, then scroll.  Otherwise position
                        ! at the new line.

                        if .phn$gb_scroll and
                           (.cursor_pos[0] gequ .dp[pub_w_viewline]+.dp[pub_w_viewsize]-2) then
                                phn$fresh_view(dp,false,false)
```

```
4                       else (
4                               phn$view_goodies(dp,cursor_pos);
4                               scr$erase_line(.cursor_pos[0],1);
3                       );
3
3
3 [tab]:                ! If we are not too far along on the line, we can tab.
3
3                       if .latest_line_dsc[len]+1 lequ 73 then
3                               scr$put_screen(describe(%char(tab)));
3
3
3 [delete,
3  linefeed,
3  ctrl_u]:             ! We have a character that deleted some text.  Position to
3                       ! the new cursor point and erase the rest of the line.
3
4                       (phn$view_goodies(dp,cursor_pos);
3                       scr$erase_line(.cursor_pos[0],.cursor_pos[1]););
3
3
3 [formfeed]:           ! We are clearing the viewport.  Just refresh all text lines.
3
3                       phn$fresh_view(dp,false,false);
3
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          52  91 0018E 20$:  CMPB    R2, #13                                 ; 1504
                          11  12 00191       BNEQ    21$                                     ;
                      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                                          :
                6B                02  FB 0027F        CALLS    #2, PHN$VIEW_GOODIES                        :
                                  01  DD 00282        PUSHL    #1                                          : 1581
                          F0      AD  DD 00284        PUSHL    CURSOR_POS                                  :
                                  56  DD 00287        PUSHL    LATEST_LINE_DSC                             :
     00000000G  00                03  FB 00289        CALLS    #3, SCR$PUT_SCREEN                          :
     00000000G  00                00  FB 00290        CALLS    #0, SCR$ERASE_LINE                          : 1582
                                  7F  11 00297 33$:   BRB      43$                                         : 1564
          08    AE                01  D0 00299 34$:   MOVL     #1, CHAR_DSC                                : 1592
          0C    AE        04      AE  9E 0029D        MOVAB    CHAR, CHAR_DSC+4                            : 1593
                          08      AE  9F 002A2        PUSHAB   CHAR_DSC                                    : 1594
                                  43  11 002A5        BRB      39$                                         :
                0D                50  91 002A7 35$:   CMPB     R0, #13                                     : 1598
                                  27  12 002AA        BNEQ     38$                                         :
                16      0000G     CF  E9 002AC        BLBC     PHN$GB_SCROLL, 37$                          : 1603
                50      00FE      C8  3C 002B1        MOVZWL   254(R8), R0                                 : 1604
                51      00FC      C8  3C 002B6        MOVZWL   252(R8), R1                                 :
                50                51  C0 002BB        ADDL2    R1, R0                                      :
                50                02  C2 002BE        SUBL2    #2, R0                                      :
                50        F0      AD  D1 002C1        CMPL     CURSOR_POS, R0                              :
                                  58  1E 002C5 36$:   BGEQU    45$                                         :
                          F0      AD  9F 002C7 37$:   PUSHAB   CURSOR_POS                                  : 1607
                                  58  DD 002CA        PUSHL    R8                                          :
                6B                02  FB 002CC        CALLS    #2, PHN$VIEW_GOODIES                        :
                                  01  DD 002CF        PUSHL    #1                                          : 1608
                                  3B  11 002D1        BRB      42$                                         :
                09                50  91 002D3 38$:   CMPB     R0, #9                                      : 1612
                                  1B  12 002D6        BNEQ     40$                                         :
                50                66  3C 002D8        MOVZWL   (LATEST_LINE_DSC), R0                       : 1614
                                  50  D6 002DB        INCL     R0                                          :
      00000049  8F                50  D1 002DD        CMPL     R0, #73                                     :
                                  42  1A 002E4        BGTRU    46$                                         :
                        0000'     CF  9F 002E6        PUSHAB   P.AAV                                       : 1615
     00000000G  00                01  FB 002EA 39$:   CALLS    #1, SCR$PUT_SCREEN                          :
                                  35  11 002F1        BRB      46$                                         : 1614
                0A                50  91 002F3 40$:   CMPB     R0, #10                                     : 1618
                                  0B  13 002F6        BEQL     41$                                         :
                15                50  91 002F8        CMPB     R0, #21                                     :
                                  06  13 002FB        BEQL     41$                                         :
           7F   8F                50  91 002FD        CMPB     R0, #127                                    :
                                  17  12 00301        BNEQ     44$                                         :
                          F0      AD  9F 00303 41$:   PUSHAB   CURSOR_POS                                  : 1623
                                  58  DD 00306        PUSHL    R8                                          :
                6B                02  FB 00308        CALLS    #2, PHN$VIEW_GOODIES                        :
                          F4      AD  DD 0030B        PUSHL    CURSOR_POS+4                                : 1624
                          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
                0C                50  91 0031A 44$:   CMPB     R0, #12                                     : 1627
                                  09  12 0031D        BNEQ     46$                                         :
                                  7E  7C 0031F 45$:   CLRQ     -(SP)                                       : 1629
                                  58  DD 00321        PUSHL    R8                                          :
          00DD  CB                03  FB 00323        CALLS    #3, PHN$FRESH_VIEW                          :
                07        04      AE  91 00328 46$:   CMPB     CHAR, #7                                    : 1637
                                  0B  12 0032C        BNEQ     47$                                         :
                        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
```

```
1 %sbttl 'PHN$SCROLL_PREP - Prepare to Scroll Information'
1 !++
1 ! Functional Description:
1 !       This routine is called when a command wants to begin scrolling
1 !       information onto the screen.  A command like HELP scrolls info
1 !       into the screen beginning at line 5 and lasting for one or more
1 !       lines.
1 !
1 ! Formal Parameters:
1 !       heading         Address of descriptor of heading line to be
1 !                       placed on line 5.  Optional.
1 !
1 ! Implicit Inputs:
1 !       global data
1 !
1 ! Implicit Outputs:
1 !       global data
1 !
1 ! Returned Value:
1 !       none
1 !
1 ! Side Effects:
1 !
1 !--
1
1
2 global routine phn$scroll_prep(heading): novalue = begin
2
2 builtin
2       nullparameter;
2
2
2 ! Erase the screen beginning at line 5.
2
2 scr$erase_page(5,1);
2
2 ! If we are to display a heading, do so.  Position cursor at line 7.
2
2 if not nullparameter(1) then
2       scr$put_line(.heading,2);
2
2 ! If this is a VT100, it will look a lot sexier if we don't scroll off
2 ! the top of the screen.
2
2 scr$set_scroll((if nullparameter(1) then 5 else 7), 24);
2
2 return;
2
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$SCROLL_LINE - Scroll Line Onto Screen'
; 1396          1703  1 !++
; 1397          1704  1 ! Functional Description:
; 1398          1705  1 !       This routine is called to scroll a line of information (e.g., HELP)
; 1399          1706  1 !       onto what is normally the viewport area of the screen.  The area
; 1400          1707  1 !       must have been previously prepared by PHN$SCROLL_PREP.
; 1401          1708  1 !
; 1402          1709  1 ! Formal Parameters:
; 1403          1710  1 !       line            Address of descriptor of line to be scrolled.
; 1404          1711  1 !
; 1405          1712  1 ! Implicit Inputs:
; 1406          1713  1 !       global data
; 1407          1714  1 !
; 1408          1715  1 ! Implicit Outputs:
; 1409          1716  1 !       global data
; 1410          1717  1 !
; 1411          1718  1 ! Returned Value:
; 1412          1719  1 !       none
; 1413          1720  1 !
; 1414          1721  1 ! Side Effects:
; 1415          1722  1 !
; 1416          1723  1 !--
; 1417          1724  1
; 1418          1725  1
; 1419          1726  2 global routine phn$scroll_line(line): novalue = begin
; 1420          1727  2
; 1421          1728  2
; 1422          1729  2 ! Just display it with a carriage return/line feed.
; 1423          1730  2
; 1424          1731  2 scr$put_line(.line,1);
; 1425          1732  2
; 1426          1733  2 return;
; 1427          1734  2
; 1428          1735  1 end;


                                      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
; 1430          1737  0 end eludom


                                                          .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
;
;                                             -------- Symbols --------      Pages        Processing
;       File                                  Total    Loaded   Percent     Mapped        Time
;
;   _$255$DUA28:[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
```

0305 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

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