```
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
```

```
$2

Val
---
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
```

FILEID**PHONEREQ

```
PPPPPPPP    HH      HH    OOOOOO    NN      NN  EEEEEEEEEE  RRRRRRRR    EEEEEEEEEE    QQQQQQ
PPPPPPPP    HH      HH    OOOOOO    NN      NN  EEEEEEEEEE  RRRRRRRR    EEEEEEEEEE    QQQQQQ
PP      PP  HH      HH  OO      OO  NN      NN  EE          RR      RR  EE          QQ      QQ
PP      PP  HH      HH  OO      OO  NN      NN  EE          RR      RR  EE          QQ      QQ
PP      PP  HH      HH  OO      OO  NNNN    NN  EE          RR      RR  EE          QQ      QQ
PP      PP  HH      HH  OO      OO  NNNN    NN  EE          RR      RR  EE          QQ      QQ
PPPPPPPP    HHHHHHHHHH  OO      OO  NN  NN  NN  EEEEEEEE    RRRRRRRR    EEEEEEEE    QQ      QQ
PPPPPPPP    HHHHHHHHHH  OO      OO  NN  NN  NN  EEEEEEEE    RRRRRRRR    EEEEEEEE    QQ      QQ
PP          HH      HH  OO      OO  NN    NNNN  EE          RR  RR      EE          QQ  QQ  QQ
PP          HH      HH  OO      OO  NN    NNNN  EE          RR  RR      EE          QQ  QQ  QQ
PP          HH      HH  OO      OO  NN      NN  EE          RR    RR    EE          QQ    QQ          ....
PP          HH      HH  OO      OO  NN      NN  EE          RR    RR    EE          QQ    QQ          ....
PP          HH      HH    OOOOOO    NN      NN  EEEEEEEEEE  RR      RR  EEEEEEEEEE    QQQQ  QQ      ....
PP          HH      HH    OOOOOO    NN      NN  EEEEEEEEEE  RR      RR  EEEEEEEEEE    QQQQ  QQ      ....

RRRRRRRR    EEEEEEEEEE    QQQQQQ
RRRRRRRR    EEEEEEEEEE    QQQQQQ
RR      RR  EE          QQ      QQ
RR      RR  EE          QQ      QQ
RR      RR  EE          QQ      QQ
RR      RR  EE          QQ      QQ
RRRRRRRR    EEEEEEEE    QQ      QQ
RRRRRRRR    EEEEEEEE    QQ      QQ
RR  RR      EE          QQ  QQ  QQ
RR  RR      EE          QQ  QQ  QQ
RR    RR    EE          QQ    QQ
RR    RR    EE          QQ    QQ
RR      RR  EEEEEEEEEE    QQQQ  QQ
RR      RR  EEEEEEEEEE    QQQQ  QQ
```

BAS
V04

```
!
! Version:      'V04-000'
!
!****************************************************************************
!*                                                                          *
!*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
!*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
!*  ALL RIGHTS RESERVED.                                                    *
!*                                                                          *
!*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
!*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
!*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
!*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
!*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
!*  TRANSFERRED.                                                            *
!*                                                                          *
!*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
!*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
!*  CORPORATION.                                                            *
!*                                                                          *
!*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
!*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
!*                                                                          *
!*                                                                          *
!****************************************************************************
!

!++
! Facility:     VAX/VMS Telephone Facility, BLISS Require File
!
! Abstract:     This is the BLISS require file for the PHONE facility.
!               It includes various useful constructs and the definitions
!               of all control blocks used by the facility.
!
!
! Environment:
!
! Author: Paul C. Anagnostopoulos, Creation Date: 29 December 1980
!
! Modified By:
!
!       V03-001 PCA1020         Paul C. Anagnostopoulos 27-May-1983
!               Add various message status codes.
!--
```

```
!
! Here we will define "extensions" to the BLISS language.
!
! First we need values for boolean variables.

literal
        false           = 0,
        true            = 1;

! Now we will define macros to generate various things associated with
! string descriptors.

field descriptor_fields = set
        len     = [0,0,16,0],
        ptr     = [4,0,32,0]
tes;

macro descriptor =
        block[8,byte] field(descriptor_fields) %;

macro describe[] =
        uplit long(%charcount(%remaining), uplit byte(%remaining)) %;

! Now we define two macros that can generate descripted buffers.  The
! first is for OWN buffers and the second for LOCAL buffers.  Note that
! the local buffer must be defined last in the declarations.

macro own_described_buffer(name,length) =
        name: block[8+length,byte] field(descriptor_fields)
                                   initial(length,name+8)
%;

macro local_described_buffer(name,length) =
        name: block[8+length,byte] field(descriptor_fields);
        name[0,0,32,0] = length;
        name[ptr] = name+8
%;

! Now we define macros to increment and decrement a variable.

macro inc(var) =
        (var = .var + 1) %,
      dec(var) =
        (var = .var - 1) %;

! We need an "infinite" loop contruct.  We also need a more elegant construct
! for terminating a loop.

macro loop =
        while 1 do %;

macro exitif[] =
        if %remaining then exitloop; %;

! Define a macro that can check statuses from routines.
```

```
macro check(status) =
        (if not status then
                signal(status);)
%;

! Declare BLISS routines that aren't defined by default.

builtin
        callg,
        insque,
        remque;

! Define literals for useful control characters.

literal
        eofrom          = %x'00',               ! Special for PHONE.
        bell            = %x'07',
        backspace       = %x'08',
        tab             = %x'09',
        linefeed        = %x'0a',
        formfeed        = %x'0c',
        ret             = %x'0d',
        ctrl_u          = %x'15',
        ctrl_w          = %x'17',
        ctrl_z          = %x'1a',
        escape          = %x'1b',
        delete          = %x'7f';
```

BA
V0

; F

```
! Now we get to stuff more specific to PHONE.  The following literals are
! used throughout the facility.

literal
        phn$k_mbxsize   = 256,          ! Maximum mailbox message size.

        phn$k_getjpiefn = 1,            ! Event flag for $GETJPI.
        phn$k_kbdefn    = 2,            ! Event flag for keyboard.
        phn$k_smbefn    = 3,            ! Event flag for steering message queue
        phn$k_decnetefn = 4,            ! Event flag for logical link I/O.
        phn$k_ourmbxefn = 5;            ! Event flag for our mailbox reads.


! The following information defines the Target Specification Block, which
! is needed to contain the parsed target specifications of people or nodes
! we wish to communicate with.

literal
        tsb_k_size = 228;

structure tsb_struc[offset,position,size,index; ] =
        [tsb_k_size]
        (tsb_struc+offset+8*index)<position,size,0>;

field tsb_fields = set
        tsb_w_flags     = [0,0,16,0],   ! Word of flags:
        tsb_v_remote    = [0,0,1,0],    !       The target is on a remote node.
        tsb_v_user      = [0,1,1,0],    !       The target is a user.
        tsb_w_tkncount  = [2,0,16,0],   ! Specification token count.
        tsb_q_tkndsc    = [4,0,0],      ! Array of token descriptors.
        tsb_t_string    = [84,0,0,0]    ! Target specification string.
tes;

macro tsb =
        tsb_struc[] field(tsb_fields) %;


! The following information defines the Phone Unit Block, which contains the
! information necessary to control the communication between us and some
! other person or node.  NOTE that it contains a TSB, as defined above.

field pub_fields = set
        pub_l_flink     = [0,0,32,0],   ! Forward link.
        pub_l_blink     = [4,0,32,0],   ! Backward link.
        pub_l_length    = [8,0,32,0],   ! Length of this PUB.
        pub_b_tsb       = [12,0,0,0],   ! TSB describing this person or node.
        pub_w_flags     = [240,0,16,0], ! Word of flags:
        pub_v_uhaveheld = [240,0,1,0],  !       You have this person on hold.
        pub_v_hasuheld  = [240,1,1,0],  !       This person has you on hold.
        pub_v_temporary = [240,2,1,0],  !       This is a temporary PUB.
        pub_v_calling   = [240,3,1,0],  !       You are calling someone.
        pub_v_answering = [240,4,1,0],  !       Someone is calling you.
        pub_w_depth     = [242,0,16,1], ! You have person at this hold depth.
        pub_w_channel   = [244,0,16,0], ! Channel number for communication.
        pub_l_busylink  = [248,0,32,0], ! Address of PUB we're busy with.
        pub_w_viewsize  = [252,0,16,0], ! Size of current viewport.
        pub_w_viewline  = [254,0,16,0], ! Starting line of current viewport.
        pub_l_ctlcount  = [256,0,32,0], ! Count of CTLs on list.
        pub_q_ctlhead0  = [260,0,32,0], ! Header for CTL list.
        pub_q_ctlhead1  = [264,0,32,0],
        pub_l_topctl    = [268,0,32,0]  ! CTL at the "top" of the viewport.
tes;

literal
        pub_k_size      = 272,          ! Overall size of PUB.
        pub_k_minlines  =   3,          ! Minimum allowable viewport size.
        pub_k_maxlines  =  10;          ! Maximum allowable viewport size.

macro pub =
        block[pub_k_size,byte] field(pub_fields) %;


! The following information defines a Conversation Text Line buffer, which
! contains one line of text from the conversation.  These blocks are chained
! off of the PUB.

field ctl_fields = set
        ctl_l_flink     = [0,0,32,0],   ! Forward link.
        ctl_l_blink     = [4,0,32,0],   ! Backward link.
        ctl_l_length    = [8,0,32,0],   ! Length of this CTL.
        ctl_l_stamp     = [12,0,32,0],  ! Synchronization stamp for transcript.
        ctl_q_line      = [16,0,0,0],   ! Descriptor for conversation line.
        ctl_t_linebuf   = [24,0,0,0]    ! The conversation line text.
tes;

literal
        ctl_k_size      = 103;          ! Overall size of a CTL.

macro ctl =
        block[ctl_k_size,byte] field(ctl_fields) %;


! The following information defines the Steering Message Block, which is
! used to control the sequencing of events in PHONE.

field smb_fields = set
        smb_l_flink     = [0,0,32,0],   ! Forward link.
        smb_l_blink     = [4,0,32,0],   ! Backward link.
        smb_w_length    = [8,0,16,0],   ! Length of this SMB.
        smb_w_type      = [10,0,16,0],  ! Type code for this message.
        smb_q_msg       = [12,0,0,0],   ! Descriptor for the message text.
        smb_t_msgbuf    = [20,0,0,0]    ! Start of the message text.
tes;

literal
        smb_k_size      = 20;           ! Base size of SMB.

macro smb =
        block[,byte] field(smb_fields) %;

literal                                 ! MESSAGE TYPES:
        smb__kbd_get            =  1,   ! Get keyboard input.
        smb__kbd_route          =  2,   ! Route keyboard input.
        smb__cmd_parse          =  3,   ! Collect and parse command.
        smb__talk               =  4,   ! Handle text we typed.
        smb__help2              =  5,   ! Handle help info scrolling.
        smb__ring_out           =  6,   ! Ring someone's phone.
        smb__slave_verify       =  7,   ! Network Slave: verify user.
        smb__rang_in            =  8,   ! Someone is ringing us.
        smb__hungup             =  9,   ! Someone hung up on us.
        smb__busy               = 10,   ! Phone we're calling is busy.
        smb__answered           = 11,   ! Person has answered a call.
        smb__rejected           = 12,   ! Person has rejected a call.
        smb__slave_done         = 13,   ! Network Slave: processing complete.
        smb__listen             = 14,   ! Handle text someone else typed.
        smb__directory2         = 15,   ! Handle directory info scrolling.
        smb__facsimile2         = 16,   ! Handle facsing a record.
        smb__forced_link        = 17,   ! Handle a forced link.
        smb__held               = 18,   ! Someone put us on hold.
        smb__unheld             = 19;   ! Someone took us off hold.
```

BAS
V0

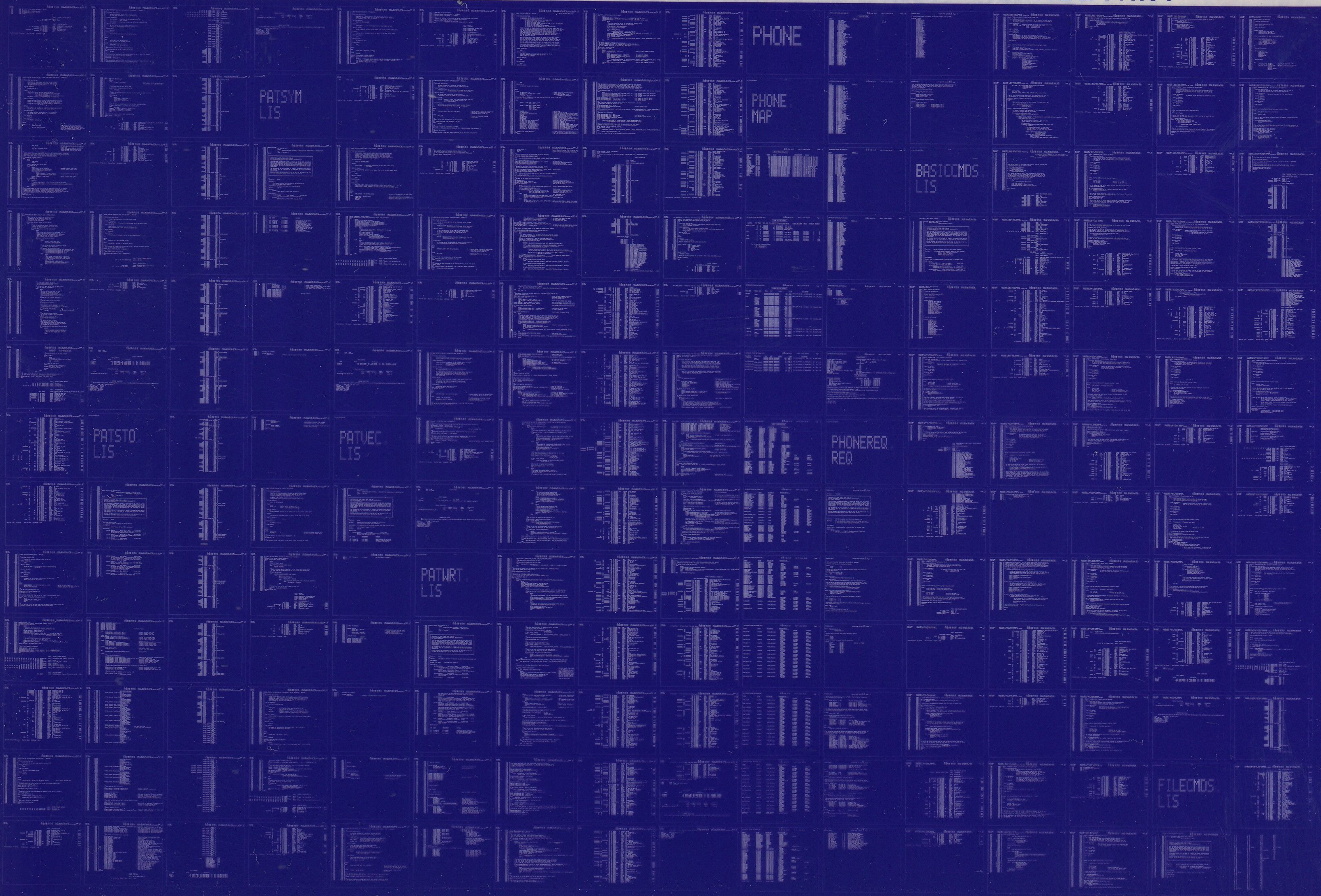
; F

```
! The following table of literals defines the messages used by PHONE.

external literal
        phn$_ok,
        phn$_answered,
        phn$_busycall,
        phn$_cancall,
        phn$_cantreach,
        phn$_confcall,
        phn$_dead,
        phn$_decnetlink,
        phn$_dircan,
        phn$_facscan,
        phn$_helpcan,
        phn$_hungup,
        phn$_justrang,
        phn$_loggedoff,
        phn$_rejected,
        phn$_ring,
        phn$_rejectjunk,
        phn$_sendingmail,
        phn$_badcmd,
        phn$_badhelp,
        phn$_badmailcmd,
        phn$_badsmb,
        phn$_badspec,
        phn$_helpmissing,
        phn$_ivredunans,
        phn$_ivreduncall,
        phn$_linkerror,
        phn$_needuser,
        phn$_nocall,
        phn$_noholds,
        phn$_noports,
        phn$_nopriv,
        phn$_noproc,
        phn$_notconv,
        phn$_onlynode,
        phn$_phonebusy,
        phn$_remoteerror,
        phn$_targterm,
        phn$_unplugged,
        phn$_badterm,
        phn$_sharedmbx,
        phn$_inputterm;
```

```
! The following declarations declare ALL the global data used in the facility.
! We only declare the data if the symbol GLOBAL_DATA is not defined; if
! defined, it means we are compiling the main module, which contains the
! definitions themselves.

%if not %declared(global_data) %then

external
        phn$gq_node_name: descriptor;

external
        phn$gq_switch_hook: descriptor,
        phn$gl_viewport_size: long,
        phn$gb_scroll: byte;

external
        phn$gq_pubhead: vector[2,long];

external
        phn$gb_flags: byte;
macro
        phn$gv_message          = phn$gb_flags<0,1,0> %,
        phn$gv_scroller         = phn$gb_flags<1,1,0> %,
        phn$gv_scrollprep       = phn$gb_flags<2,1,0> %,
        phn$gv_facsimile        = phn$gb_flags<3,1,0> %;
%fi
```

0304 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY


PHONE

PATSYM
LIS

PHONE
MAP

BASICCMDS
LIS

PATSTO
LIS

PATVEC
LIS

PHONEREQ
REQ

PATWRT
LIS

FILECMDS
LIS