```
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRRRRRRRRRRR           III       VVV         VVV  EEEEEEEEEEE      RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III       VVV         VVV  EEEEEEEEEEE      RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III       VVV         VVV  EEEEEEEEEEE      RRRRRRRRRRRR
DDD          DDD RRR   RRR              III       VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III       VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III       VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR      RRR           III          VVV   VVV     EEE              RRR      RRR
DDD          DDD RRR      RRR           III          VVV   VVV     EEE              RRR      RRR
DDD          DDD RRR      RRR           III          VVV   VVV     EEE              RRR      RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII          VVV       EEEEEEEEEEEEEEE  RRR         RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII          VVV       EEEEEEEEEEEEEEE  RRR         RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII          VVV       EEEEEEEEEEEEEEE  RRR         RRR
```

FILEID**PAMAC

```
PPPPPPPP      AAAAAA    MM      MM    AAAAAA      CCCCCCCC
PPPPPPPP      AAAAAA    MM      MM    AAAAAA      CCCCCCCC
PP      PP   AA    AA   MMMM  MMMM   AA    AA   CC
PP      PP   AA    AA   MMMM  MMMM   AA    AA   CC
PP      PP   AA    AA   MM MMMM MM   AA    AA   CC
PP      PP   AA    AA   MM MMMM MM   AA    AA   CC
PPPPPPPP     AA    AA   MM  MM  MM   AA    AA   CC
PPPPPPPP     AA    AA   MM      MM   AA    AA   CC
PP           AAAAAAAAAA  MM      MM   AAAAAAAAAA   CC
PP           AAAAAAAAAA  MM      MM   AAAAAAAAAA   CC
PP           AA    AA   MM      MM   AA    AA   CC              ....
PP           AA    AA   MM      MM   AA    AA   CC              ....
PP           AA    AA   MM      MM   AA    AA     CCCCCCCC      ....
PP           AA    AA   MM      MM   AA    AA     CCCCCCCC      ....

MM      MM     AAAAAA     RRRRRRRR
MM      MM     AAAAAA     RRRRRRRR
MMMM  MMMM   AA    AA   RR      RR
MMMM  MMMM   AA    AA   RR      RR
MM  MM  MM   AA    AA   RR      RR
MM  MM  MM   AA    AA   RR      RR
MM      MM   AA    AA   RRRRRRRR
MM      MM   AA    AA   RRRRRRRR
MM      MM   AAAAAAAAAA  RR  RR
MM      MM   AAAAAAAAAA  RR  RR
MM      MM   AA    AA   RR    RR
MM      MM   AA    AA   RR    RR
MM      MM   AA    AA   RR      RR
MM      MM   AA    AA   RR      RR
```

```
        .TITLE  PADRIVER MACROS
        .IDENT  'V04-000'
```



```
;++
; FACILITY:
;
;       VAX/VMS EXECUTIVE, I/O DRIVERS
;
; ABSTRACT:
;
; AUTHOR:  N. KRONENBERG,  MAY 1981
;
; MODIFIED BY:
;
;       V03-022 NPK3055         N. Kronenberg           14-Jul-1984
;               Add PDT$W_STDGDYN and PDT$W_STDGUSED
;               Add PAER$K_ES_REVCA for ci port rev not at current
;               level and cautionary msg needed.
;
;       V03-021 NPK3054         N. Kronenberg           24-Jun-1984
;               Add error log subtype codes PAER$K_ES_REVER/CPUREV
;               for ci port ucode revision level error and CPU rev
;               level insufficient to support ci respectively.
;
;       V03-020 NPK3048         N. Kronenberg            4-Apr-1984
;               Remove PB$W_VCFAIL_RSN -- moved to port independent
;               PB.
;               Add some spare space to the port specific
;               extensions to the UCB and PDT.
;               Add two new status subtypes to PPDDEF: 5 = port detected
;               sequence number mismatch on received pkt; 6 = port
;               received a sequenced message for a VC which has the
```

```
                open bit off in the VCD.  These two status values are
                supported in experimental versions of ucode only.

        V03-019 TMK0004         Todd M. Katz            25-Mar-1984
                Add UCB$T_OPA0_TEMP to $PAUCBDEF. This field is three
                longwords in size and is used to store optional _OPA0 error
                logging information before forking to format and broadcast an
                appropriate message. Access to this field is protected by
                the UCB_V_MSGFKLOCK message fork block interlock bit.

        V03-018 NPK3047         N. Kronenberg           22-Mar-1984
                Add PDT$Q_TEMP_RSPQ and remove PDT$L_UCBQ in $PAPDTDEF.
                Also add a host shutdown datagram to PAPDTDEF.
                Add PB$W_VCFAIL_RSN to $PAPBDEF.
                Add new PPD type code and length for host shutdown dg.

        V03-017 TMK0003         Todd M. Katz            14-Feb-1984
                Add PAER$K_ES_RSCKS (remote system conflicts with known
                system) as an error signalled via a packet subtype.

        V03-016 NPK3044         N. Kronenberg           06-Feb-1984
                Add PAER$K_ES_ERRDG, subtype for logging error log dg.
                Add PPD$C_ELOG, new PPD type for error log dg.
                Add symbolic definitions of the PPD protocol revision
                levels, 0 and 1.
                Make error log portion of ucb larger to hold error log dg.
                Move message fork block ahead of error log array in ucb.

        V03-015 TMK0002         Todd M. Katz            02-Feb-1984
                Add PDT$B_PLOGMAP - a bit map of the remote ports the
                local port has logged and won't talk to because these
                remote ports have an improper SYSID and/or nodename.

        V03-014 TMK0001         Todd M. Katz            02-Feb-1984
                Add PAER$K_ES_SCSID as a software error during
                initialization subtype.

        V03-013 NPK3037         N. Kronenberg           11-Nov-1983
                Add the $DEBUGCHECK macro.

        V03-012 NPK3029         N. Kronenberg           14-Jul-1983
                Enhancements for V4.0:
                Remove PB$L_SBLINK from the PA extension to the PB.
                Add second fork block to UCB for printing msgs.

        V03-011 NPK3010         N. Kronenberg            9-Nov-1982
                Add loopback dg enable flag to local port status in
                $PAPDTDEF;  add next port to scan value to $PAPDTDEF
                for poller to probe n per cycle.
                Add own port number to PDT.

        V03-010 NPK3008         N. Kronenberg            6-Oct-1982
                Add protocol level, node name, and current time to
                the start/stack dg format.  Shorten hardware version
                from 16 to 12 bytes in start/stack dg.
```

```
;       V03-009 NPK3004         N. Kronenberg           30-Jul-1982
;               Add C1750 definitions to $PAREGDEF.
;
;       V03-008 NPK3001         N. Kronenberg           28-Jun-1982
;               Add new SB link to PA extension to PB.
;
;       V03-007 ROW0107         Ralph O. Weber          23-JUN-1982
;               Add new loopback and unexpected interrupt subtype codes to
;               $PAERDEF.  Add loopback crossed/uncrossed status bit in
;               $PAPDTDEF fields PDT$B_P0_LBSTS nad PDT$B_P1_LBSTS.
;               This change will be in a new driver image shipped in V3.1.
;
;       V03-006 ROW0091         Ralph O. Weber           1-JUN-1982
;               Add packet error constants to $PAERDEF.
;
;       V03-005 ROW0090         Ralph O. Weber          21-MAY-1982
;               Add scratch buffer for building logged messages -- used to
;               log errors where the driver has a transaction packet in
;               hand -- to the UCB extension defined in $PAUCBDEF.
;
;       V03-004 ROW0088         Ralph O. Weber          20-MAY-1982
;               Add $PAERDEF macro which defines constants related to the
;               PADRIVER error code.  Also remove space allocation for saved
;               copies of the port registers from $PAUCBDEF.
;
;       V03-003 NPK2019         N. Kronenberg           6-Apr-1982
;               Changed $QRETRY to BSBW to error addr instead of BRW.
;               Modified interlocked queue macros to report interlock
;               error to different address depending on queue operation
;               and queue being operated upon.
;
;       V03-002 NPK2018         N. Kronenberg           25-Mar-1982
;               Purged revision history, reset ident.
;               Fixed start handshake dg format to have full system rev.
;               Modified $QRETRY to use system wide retry count.
;               Added $PAPBDEF to define port specific extension to PB.
;--
```

```
        .SBTTL  GENERAL MACROS



;+
; BUGCHECK -- Call port local store save routine prior to issuing
;             a real VMS BUG_CHECK request.
;
; Inputs:
;
;       R4                      -Addr of PDT
;
; Outputs:
;
;       Microcode overwritten with local store contents
;-

        .MACRO  BUGCHECK        CODE,SEVERITY

         .IF    IDN SEVERITY,NONFATAL
         BSBW   ERR$BUGCHECKNF
         BUG_CHECK CODE
         .IFF

         BSBW   ERR$BUGCHECK
         BUG_CHECK CODE,TYPE=FATAL
         .ENDC

        .ENDM   BUGCHECK
```

```
;+
; $DEBUGCHECK -- Execute a fatal bugcheck if this type of bugcheck
;                is enabled.
;
; The purpose of this optional bugcheck is to allow selective fatal
; CIPORT bugchecks (independent of the system wide setting of nonfatal
; bugchecks to be fatal) on possible port or software problems that are
; normally recovered from.
;
; Inputs:
;
;       FLAG                    -Type of bugcheck.  If flag stored
;                                in ERR$DEBUGCHECK is set, then
;                                bugcheck is enabled.  If flag is
;                                0, then continue with recovery.
;
; Outputs:
;
;       All registers           -Preserved
;-

        .MACRO  $DEBUGCHECK     FLAG,?IGNORE

         BBC    FLAG,ERR$DEBUGCHECK,IGNORE
         BUGCHECK CIPORT,FATAL

IGNORE:

        .ENDM   $DEBUGCHECK
```

```
;+
; $DISPATCH -- Dispatch on set of index values, not necessarily dense.
;
; This macro translates into the CASEx instruction.  It calculates the
; "base" and "limit" parameters from the <index,displacement> list
; specfied in the 'vector' parameter.  The dispatch table is set up
; such that any unspecified index value within the bounds of the
; transfer vector is associated with a diplacement which transfers
; control to the first location after the CASE statement, i.e., behaves
; as if the index were out of bounds.
;
; Note that since the index values themselves appear in the vector
; (presumably symbolically), no ASSUME statements are needed.
;-

        .MACRO  $DISPATCH       INDX,VECTOR,TYPE=W,NMODE=S^#,?MN,?MX,?S,?SS,?ZZ
         .NOSHOW
SS:

        .MACRO  $DSP1,$DSP1_1
                .IRP    $DSP1_2,$DSP1_1
                        $DSP2   $DSP1_2
                .ENDR
        .ENDM

        .MACRO  $DSP2,$DSP2_1,$DSP2_2
                .=<$DSP2_1-MN>*2 + S
                .WORD   $DSP2_2-S
        .ENDM

        .MACRO  $BND1,$BND1_1,$BND1_2,$BND1_3
                $BND2   $BND1_1,$BND1_2
        .ENDM

        .MACRO  $BND2,$BND2_1,$BND2_2
                .IIF    $BND2_1,$BND2_2-.,      .=$BND2_2
        .ENDM

        .MACRO  $BND    $BND_1,$BND_2
                .IRP    $BND_3,<$BND_2>
                        $BND1   $BND_1,$BND_3
                .ENDR
        .ENDM

        .=0
ZZ:
        $BND    GT,<VECTOR>
MX:
        $BND    LT,<VECTOR>
MN:
        .=SS

CASE'TYPE       INDX,#<MN-ZZ>,NMODE'<MX-MN>
S:
        .REPT   MX-MN+1
```

```
        .WORD   <MX-MN>*2 + 2
        .ENDR

        .=S

        $DSP1   <<VECTOR>>

        .=<MX-MN>*2 + S + 2

         .SHOW
        .ENDM
```

```
;+
; $QRETRY -- Issue an interlocked queue instruction with retries.
;
; This macro issues the interlocked queue instruction specified
; by the opcode argument.  If the queue is interlocked, then a
; retry count set up in R0 is stepped and the instruction retried
; EXE$GL_LOCKRTRY times.  If it does not succeed in any of the retries,
; a BRW is taken to the error address specified in the error argument.
;-

        .MACRO  $QRETRY OPCODE,OPER1,OPER2,ERROR,?LOOP,?OK

        .IF     DF PA$DEBUG             ; Debug facility
        BSBW    MON$CHKQ                ; Check queue integrity
        .ENDC                           ;

        CLRL    R0                      ; Init count of retries
LOOP:   OPCODE  OPER1,OPER2             ; Try again
        BCC     OK                      ; Branch if got interlock
        AOBLSS  G^EXE$GL_LOCKRTRY,R0,LOOP  ; Step retry count, branch if more
        BSBW    ERROR                   ; Branch if failure after many retries

OK:                                     ; Instruction executed; condition
                                        ;  codes as usual for instruction
        .IF     DF PA$DEBUG             ; Debug facility
        BSBW    MON$CHKQ_POST           ; Verify all queues.
        .ENDC                           ;

        .ENDM   $QRETRY
```

```
        .SBTTL  CI PORT MACROS


;+
; $DEFFLAGS  -- Given the opcode, path select, and other options,
;               construct the fourth longword of a port command
;               except for the port number, in symbol $$$FLAGS.
;
; Inputs:
;
;       OPCODE                  -Command opcode name
;       RETFLAG                 -TRUE/FALSE for returning sent
;                                dg or msg to response/free queue
;                                May also be an opcode in which
;                                case no value for the response flag
;                                is derived in this macro.
;       PATH                    -AUTO/P0/P1 for automatic select/
;                                path 0/ path 1.  May also be an
;                                opcode = 0/1/2 for auto/P0/P1.
;                                If an opcode, then no value is set
;                                for path in this macro.
;       PKTSIZ                  -TBS
;       PKTMUL                  -TBS
;       PKFMT                   -TBS
;
; Outputs:
;       $$$FLAGS = <31,8> of the 4th longword of the PPD header,
;                  <7,0> are 0.
;-

        .MACRO  $DEFFLAGS OPCODE,RETFLAG=FALSE,PATH=AUTO,-
                          PKTSIZ=512,PKTMUL=0,PKTFMT=0
         $$$FLAGS=0

         .IF     IDN     RETFLAG,TRUE
         $$$FLAGS = 1@<PPD$V_RSP+24>
         .ENDC

         .IF     IDN     PATH,AUTO
         $$$FLAGS = $$$FLAGS!<PPD$C_PSAUTO@<PPD$V_PS+24>>
         .ENDC

         .IF     IDN     PATH,P0
         $$$FLAGS = $$$FLAGS!<PPD$C_PSP0@<PPD$V_PS+24>>
         .ENDC

         .IF     IDN     PATH,P1
         $$$FLAGS = $$$FLAGS!<PPD$C_PSP1@<PPD$V_PS+24>>
         .ENDC

         $$$FLAGS = $$$FLAGS!<PPD$C_'OPCODE'@16>

        .ENDM   $DEFFLAGS
```

```
;+
; $IFNTF -- If the specified flag argument is not identically
;           FLGVAL1 or FLGVAL2, then the specified OPERATION is
;           assembled.  Otherwise, $IFNTF is a noop.
;
; Inputs:
;
;       FLAG                    -Flag to check for specified two values
;       FLGVAL1/2               -Flag values, 'TRUE', 'FALSE' by default
;       OPERATION               -Macro instructions to execute
;
; Outputs:
;
;       Depends upon OPERATION
;-

        .MACRO  $IFNTF  FLAG,FLGVAL1=TRUE,FLGVAL2=FALSE,OPERATION

         .IF    DIF     FLAG,FLGVAL1
         .IF    DIF     FLAG,FLGVAL2
         OPERATION
         .ENDC
         .ENDC

        .ENDM   $IFNTF
```

```
;+
; $INS_DFREEQ -- Insert a datagram buffer on the tail of the
;                the datagram free queue.
; $INS_MFREEQ -- Insert a message buffer on the tail of the
;                message free queue.
;
; The port number, opcode, status, and flag bytes are all zeroed.
; The purpose of this is to make sure that the response flag = 0
; for all free queue entries so that commands can be distinguished
; from possible free queue entries in the power fail logout area.
;
; Inputs:
;
;       R2                      -Addr of buffer
;       R4                      -Addr of PDT
;       ERRADDR                 -Branch addr in case queue interlock
;                                is unobtainable
;
; Outputs:
;
;       Z bit                   -1/0 if first/not first entry in queue
;       R0                      -Destroyed
;-

        .MACRO  $INS_DFREEQ ERRADDR=INT$FATALQ_IDFQ,?DONE

         CLRL   PPD$B_PORT(R2)
        $QRETRY INSQTI  (R2),@PDT$L_DFQHDR(R4),ERROR=ERRADDR
         BNEQ   DONE
         MOVZBL #PA_DFQ_M_DFQC,@PDT$L_DFQ(R4)
DONE:

        .ENDM


        .MACRO  $INS_MFREEQ ERRADDR=INT$FATALQ_IMFQ,?DONE

         CLRL   PPD$B_PORT(R2)
        $QRETRY INSQTI  (R2),@PDT$L_MFQHDR(R4),ERROR=ERRADDR
         BNEQ   DONE
         MOVZBL #PA_MFQ_M_MFQC,@PDT$L_MFQ(R4)
DONE:

        .ENDM


        .MACRO  $INS_COMQLOW ERRADDR=INT$FATALQ_CQL,?DONE

        $QRETRY INSQTI  (R2),PDT$Q_COMQL(R4),ERROR=ERRADDR
         BNEQ   DONE
         MOVZBL #PA_CQO_M_CQC,@PDT$L_CQO(R4)
DONE:

        .ENDM
```

```
        .MACRO  $INS_COMQHIGH ERRADDR=INT$FATALQ_CQH,?DONE

        $QRETRY INSQTI  (R2),PDT$Q_COMQH(R4),ERROR=ERRADDR
         BNEQ   DONE
         MOVZBL #PA_CQ1_M_CQC,@PDT$L_CQ1(R4)
DONE:

        .ENDM
```

```
;+
; $PAERDEF -- Define PA error type code error type and subtype values
;
;                    PADRIVER ERROR TYPE CODE FORMAT
;
;  3                 2 2                 1 1 1               0 0                 0
;  1                 4 3                 6 5 4               8 7                 0
; +-----------------+-----------------+-+---------------+-----------------+
; |   UCB$B_ERTMAX  |  UCB$B_ERTCNT   |C|   ERROR TYPE  |  ERROR SUBTYPE  |
; +-----------------+-----------------+-+---------------+-----------------+
;
; WHERE:
;
;    UCB$B_ERTMAX is the maximum number of port crashes allowed (usually 10).
;
;    UCB$B_ERTCNT is the number of port crashes remaining.
;
;    C indicates whether or not the port will be crashed as a result of this
;      error.  0 ==> no, 1 ==> yes
;
;    ERROR TYPE is one of the PAER$K_ET_ constants defined below.
;
;    ERROR SUBTYPE is one of the PAER$K_ES_ constants defined below.
;
;    Together ERROR TYPE and ERROR SUBTYPE uniquely define the error condition
;    which precipitated the error log entry.
;-

        .MACRO  $PAERDEF, $GBL

        .NOSHOW

        $DEFINI PAER, $GBL

;
;   Crash port bit definitions
;
        $VIELD  PAER, 7, <-
                <CPRT,,M>-
                >

;
;   Error type values
;
        $EQULST PAER$K_ET_, $GBL, 0, , <-  ;-- DEVICE ATTENTION ERRORS --
                <INSW>,-                   ;  sofware error during initialization
                <HW>,-                     ;  hardware error
                <ILCK>,-                   ;  queue interlock failure
                <DALT>-                    ;  marker for end of list
                >
        ASSUME  PAER$K_ET_DALT LT 64

        $EQULST PAER$K_ET_, $GBL, 64, , <- ;-- LOGGED MESSAGE ERRORS --
                <PKT>,-                    ;  errors signaled via a packet
```

```
                <CBL>,-                 ;  cable change of state notifications
                <LMLT>-                 ;  marker for end of list
                >
        ASSUME  PAER$K_ET_LMLT LT 128

;
;   Software error during port initialization
;   Error subtype values
;
        $EQULST PAER$K_ES_, $GBL, 0, , <-
                <POOL>,-                ;  insufficient non-paged pool
                <CODE>,-                ;  cannot locate CI microcode
                <SCSID>,-               ;  SCSSYSTEMID is not initialized
                <LST0>-                 ;  marker for end of list
                >
        ASSUME  PAER$K_ES_LST0 LT 255

;
;   Hardware error
;   Error subtype values
;
        $EQULST PAER$K_ES_, $GBL, 0, , <-
                <UCDW>,-                ;  cannot read-back microcode
                <INIT>,-                ;  failed uninitialized to disabled
                -                       ;       transition
                <HWER>,-                ;  unspecified port hardware error
                <PDWN>,-                ;  power down
                <PUP>-                  ;  power up
                <UXIN>,-                ;  unexpected interrupt
                <REVER>,-               ;  CI port ucode rev not high enough
                <CPUREV>,-              ;  CPU rev level not high enough
                <REVCA>,-               ;  CI ucode not at current level
                <LST1>-                 ;  marker for end of list
                >
        ASSUME  PAER$K_ES_LST1 LT 255

;
;   Failed to obtain a queue interlock
;   Error subtype values
;
        $EQULST PAER$K_ES_, $GBL,  0, , <-
                <MQRM>,-                ;  message free queue for remove
                <DQRM>,-                ;  datagram free queue for remove
                <RQRM>,-                ;  response queue for remove
                <HCIN>,-                ;  high prio. cmd. queue for insert
                <LCIN>,-                ;  low prio. cmd. queue for insert
                <MQIN>,-                ;  message free queue for insert
                <DQIN>-                 ;  datagram queue for insert
                <LST2>-                 ;  marker for end of list
                >
        ASSUME  PAER$K_ES_LST2 LT 255

;
;   Errors signaled via a packet
;   Erros subtype values
;
```

```
        $EQULST PAER$K_ES_, $GBL, 0, , <-
                <UPKT>,-                ;  unrecognized packet
                <PCVC>,-                ;  port has closed the VC
                <CSHP>,-                ;  software is crashing port
                <SCVC>,-                ;  software is closing the VC
                <CNPB>,-                ;  CONNECT with no PB
                <SCA>,-                 ;  inappropriate SCA type received
                <NOPB>,-                ;  no PB found during VC closing
                <ERRDG>,-               ;  error log datagram received
                <RSCKS>,-               ;  remote system conflicts with known system
                <LST3>-                 ;  marker for end of list
                >
        ASSUME  PAER$K_ES_LST3 LT 255

;
;   Cable change of state notifications
;   Subtype values
;
        $EQULST PAER$K_ES_, $GBL, 0, , <-
                <0GB>,-                 ;  path 0 good to bad
                <1GB>,-                 ;  path 1 good to bad
                <0BG>,-                 ;  path 0 bad to good
                <1BG>,-                 ;  path 1 bad to good
                <UC>,-                  ;  uncrossed to crossed
                <CU>,-                  ;  crossed to uncrossed
                <L0GB>,-                ;  path 0 loopback good to bad
                <L1GB>,-                ;  path 1 loopback good to bad
                <L0BG>,-                ;  path 0 loopback bad to uncrossed
                <L1BG>,-                ;  path 1 loopback bad to uncrossed
                <L0BX>,-                ;  path 0 loopback bad to crossed
                <L1BX>,-                ;  path 1 loopback bad to crossed
                <LST4>-                 ;  marker for end of list
                >
        ASSUME  PAER$K_ES_LST4 LT 255
        $DEFEND PAER, $GBL, DEF
        .SHOW
        .ENDM   $PAERDEF
```

```
;+
; $PAPBDEF -- Define PA port specific extension to Path Block
;-

        .MACRO  $PAPBDEF,$GBL
         .NOSHOW
         $DEFINI PAPB,$GBL
.=.+PB$C_LENGTH                                 ; Position to end of port-
                                                ;  independent path block
         $DEF    PB$L_CLSCKT_DG  .BLKL   1      ; Addr of emergency SETCKT
                                                ;  dg, also used for cache
                                                ;  clear sequenced msg!
                                 .BLKL   2      ; Reserved
         $DEF    PB$C_PALENGTH                  ; PB size with extension
         $DEFEND PAPB,$GBL,DEF
         .SHOW
         .ENDM
```

```
;+
; $PAPDTDEF -- Define PA port specific extension to the PDT
;-

        .MACRO  $PAPDTDEF,$GBL

        .NOSHOW

        $DEFINI PAPDT,$GBL

.=.+PDT$C_LENGTH                                ; Position to end of port-
                                                ;  independent portion of PDT

        $DEF    PDT$C_PAREGBASE                 ; Start of addresses o.
                                                ;  CI registers
        $DEF    PDT$L_CNF       .BLKL   1       ; Configuration register addr
        $DEF    PDT$L_PMC       .BLKL   1       ; Port maint/control reg addr
        $DEF    PDT$L_PS        .BLKL   1       ; Port status reg addr
        $DEF    PDT$L_CQ0       .BLKL   1       ; Command Q 0 control
        $DEF    PDT$L_CQ1       .BLKL   1       ; Command Q 1 control
        $DEF    PDT$L_PSR       .BLKL   1       ; Port status release
        $DEF    PDT$L_DFQ       .BLKL   1       ; DG free Q control
        $DEF    PDT$L_MFQ       .BLKL   1       ; Msg free Q control
        $DEF    PDT$L_MTC       .BLKL   1       ; Maint timer control
        $DEF    PDT$L_PFAR      .BLKL   1       ; failing address
        $DEF    PDT$L_PPR       .BLKL   1       ; Port parameters
        $DEF    PDT$C_PAREGEND                  ; End of address of CI
                                                ;  registers

$DEF    PDT$W_LPORT_STS .BLKW   1               ; Port status

                                                ; Define port status bits
$EQU    PDT$V_PWF_CLNUP 0                       ;  Power fail cleanup in progress
$EQU    PDT$M_PWF_CLNUP 1                       ;

$EQU    PDT$V_PUP       1                       ;  Power up has occurred
$EQU    PDT$M_PUP       2                       ;

$EQU    PDT$V_LBDG      2                       ; 0/1 for LB dg's disabled/
$EQU    PDT$M_LBDG      4                       ;  enabled on this port

$DEF    PDT$W_PBCOUNT   .BLKW   1               ; # PB's associated with this
                                                ;  PDT during pwr fail recovery
$DEF    PDT$B_PORTMAP   .BLKB   32              ; Bit map of ports
                                                ;  we've heard from
$DEF    PDT$B_PLOGMAP   .BLKB   32              ; Bit map of ports we've logged
                                                ;  with improper nodename and/or
                                                ;  SYSID to whom we won't talk
$DEF    PDT$B_DQIMAP    .BLKB   32              ; Datagram inhibit mask
$DEF    PDT$Q_FORMPB    .BLKL   2               ; Listhead of formative PB's
                                                ;  from this port
$DEF    PDT$B_MAX_PORT  .BLKB   1               ; Max port number supported by
                                                ;  this CI
$DEF    PDT$B_PORT_NUM  .BLKB   1               ; Own port number
$DEF    PDT$B_NXT_PORT  .BLKB   1               ; # of next port to poll
$DEF    PDT$B_REQIDPS   .BLKB   1               ; Path select value for
```

```
                                                ;  configuration poller.
        $DEF    PDT$B_P0_LBSTS  .BLKB   1       ; Status of current and
        $DEF    PDT$B_P1_LBSTS  .BLKB   1       ;  prev LB dg tests for
                                                ;  paths 0/1
                                                ; LB status bits:
        $EQU    PDT$V_CUR_LBS   0               ;  Current LB status
        $EQU    PDT$M_CUR_LBS   1               ;
        $EQU    PDT$V_PRV_LBS   1               ;  Previous LB status
        $EQU    PDT$M_PRV_LBS   2               ;
        $EQU    PDT$V_X_LBS     2               ;  Previous LB crossed status
        $EQU    PDT$M_X_LBS     4               ;   0==>uncrossed 1==>crossed
                                .BLKB   2       ; 2 reserved bytes
        $DEF    PDT$L_LBDG      .BLKL   1       ; Addr of template loopback
                                                ;  datagram
        $DEF    PDT$L_POOLDUE   .BLKL   1       ; Due time for pool waiter check
        $DEF    PDT$L_POLLERDUE .BLKL   1       ; Duetime for config poller
        $DEF    PDT$L_DGHDRSZ   .BLKL   1       ; SCS/PPD header size
        $DEF    PDT$L_DGNETHD   .BLKL   1       ; Network header size
        $DEF    PDT$W_STDGDYN   .BLKW   1       ; # dgs queued for IDRECs
                                                ;  used in start handshakes and
                                                ;  finding out about bad paths
        $DEF    PDT$W_STDGUSED  .BLKW   1       ; # ports that we know of that
                                                ;  will be sending IDRECs
        $DEF    PDT$Q_TEMP_RSPQ .BLKL   2       ; Temp response queue to
                                                ;  hold responses dequeued
                                                ;  during send of host
                                                ;  shutdown datagram
                                .BLKL   2       ; Reserved
$$$CURSIZ = .                                   ; Make PDT stuff so far
$$$NEWSIZ = <<.+15>@-4>@+4                      ;  be multiple of 16 bytes
. = . + <$$$NEWSIZ - $$$CURSIZ>                 ;  so host shutdown dg will
                                                ;  be quadword aligned
                                                ; Start of host shutdown dg:
        $DEF    PDT$B_HSHUT_DG  .BLKL   2       ; Fwd, back links
                                .BLKW   1       ; Structure size
                                .BLKB   2       ; Structure type, subtype
                                .BLKB   1       ; Destination port number
                                .BLKB   1       ; SNDDG status
                                .BLKB   1       ; Opcode
                                .BLKB   1       ; Flags
                                .BLKW   1       ; PPD length
                                .BLKW   1       ; PPD type
        $EQU    PDT$C_HSHUT_SIZ <. - PDT$B_HSHUT_DG>
                                                ; End of host shutdown dg


$$$CURSIZ = .
$$$NEWSIZ = <<.+15>@-4>@+4
. = . + <$$$NEWSIZ - $$$CURSIZ>                 ; Following Q headers must
                                                ;  be quadword aligned:
        $DEF    PDT$Q_DFREEQ    .BLKQ   1       ; Datagram free queue header
        $DEF    PDT$Q_MFREEQ    .BLKQ   1       ; Message free queue header

PDT$C_PQB = .                                   ; Base of PQB
```

```
        $DEF    PDT$Q_COMQBASE                  ; Base of queue headers
        $DEF    PDT$Q_COMQL     .BLKL   2       ; Listhead for command
                                                ;  queue 0, low priority
        $DEF    PDT$Q_COMQH     .BLKL   2       ; Listhead for command
                                                ;  queue1, high priority
        $DEF    PDT$Q_COMQ2     .BLKL   2       ; Listhead for command
                                                ;  queue 2
        $DEF    PDT$Q_COMQ3     .BLKL   2       ; Listhead for command
                                                ;  queue 3
        $DEF    PDT$Q_RSPQ      .BLKL   2       ; Listhead for response
                                                ;  queue
        $DEF    PDT$L_DFQHDR    .BLKL   1       ; Addr of dg free queue
                                                ;  listhead
        $DEF    PDT$L_MFQHDR    .BLKL   1       ; Addr of msg free queue
                                                ;  listhead
        $DEF    PDT$W_DQELEN    .BLKW   1       ; DG free Q entry length

.=.+2                                           ; MBZ word

        $DEF    PDT$W_MQELEN    .BLKW   1       ; Msg free Q entry length

.=.+2                                           ; MBZ word

        $DEF    PDT$L_VPQB      .BLKL   1       ; VA of PQB base
        $DEF    PDT$L_VBDT      .BLKL   1       ; VA of BDT base
        $DEF    PDT$W_BDTLEN    .BLKW   1       ; # of entries in BDT

.=.+2                                           ; MBZ word

        $DEF    PDT$L_SPTBASE   .BLKL   1       ; PA of base of SPT
        $DEF    PDT$L_SPTLEN    .BLKL   1       ; # longwds in SPT
        $DEF    PDT$L_GPTBASE   .BLKL   1       ; VA of GPT base
        $DEF    PDT$L_GPTLEN    .BLKL   1       ; # longwds in GPT

.=PDT$C_PQB+256                                 ; Position to offset 256
                                                ;  within PQB

        $DEF    PDT$L_DQELOGOUT .BLKL   16      ; DG's held by port on
                                                ;  powerfailure
        $DEF    PDT$L_MQELOGOUT .BLKL   16      ; MQE's held by port on
                                                ;  power failure.

PDT$C_PALENGTH = .                              ; Total size of a
                                                ;  PDT for the CI

        $DEFEND PAPDT,$GBL,DEF
        .SHOW

        .ENDM
```

```
;+
; $PAREGDEF -- Define offsets to CI registers and fields in the registers.
;-

        .MACRO  $PAREGDEF,$GBL

        .NOSHOW

        $DEFINI PAREG,$GBL

        $DEF    PA_CNF  .BLKL   1               ; Configuration register

        $FIELD  PA_CNF,0,<-                     ; Define config register fields:
        <ADPTYP,8,M>,-                          ;  Adapter type code
        <PFD,,M>,-                              ;  Power fail disable
        <TDEAD,,M>,-                            ;  Transmit dead
                                                ;  (T DCLO on CI750)
        <TFAIL,,M>,-                            ;  Transmit fail
                                                ;  (T ACLO on CI750)
        <,1>,-                                  ;  1 unused bit
                                                ;  4 CI750 only bits:
        <NOCI,,M>,-                             ;   CI750 uncabled or no pwr
        <CTO,,M>,-                              ;   CI750 port timeout
        <MAINT,,M>,-                            ;   Enable internal maint reg
        <CIBPE,,M>,-                            ;   CI bus parity error
        <CRD,,M>,-                              ;  CRD on port init'd read
        <RDS,,M>,-                              ;  RDS on port init'd read
                                                ;  (UCE on CI750)
        <CXTER,,M>,-                            ;  SBI error confirm
                                                ;  (CI780 only)
        <RDTO,,M>,-                             ;  Port init'd read timeout on SBI
                                                ;  (CI780 only)
        <CXTMO,,M>,-                            ;  Port init'd command xmit timeout
                                                ;  (NXM on CI750)
        <,1>,-                                  ;  1 unused bit
        <PUP,,M>,-                              ;  Adapter power up
        <PDN,,M>,-                              ;  Adaptor power down
        <,2>,-                                  ;  2 unused bits
                                                ;  CI780 specific bits:
        <XMTFLT,,M>,-                           ;   Transmit fault on SBI
        <MXTFLT,,M>,-                           ;   Multiple xmitter fault on SBI
        <,1>,-                                  ;   1 unused bit
        <URDFLT,,M>,-                           ;   Unexpected read data fault
        <WSQFLT,,M>,-                           ;   Write squence fault
        <PARFLT,,M>,-                           ;   Parity error
        >                                       ;

        $DEF    PA_PMC  .BLKL   1               ; Port maint control/status register

        $FIELD  PA_PMC,0,<-                     ; Define register fields:
        <MIN,,M>,-                              ;  Maint initialized
        <MTD,,M>,-                              ;  Maint timer disable
        <MIE,,M>,-                              ;  Maint interrupt enable
        <MIF,,M>,-                              ;  Maint intterupt flag
        <WP,,M>,-                               ;  Wrong parity
        <RSVD,,M>,-                             ;  Reserved for microcode
```

```
        <PSA,,M>,-                      ;  Programmable starting addr
        <UNIN,,M>,-                     ;  Port in uninited state
        <XBPE,,M>,-                     ;  Xmit buffer parity error
        <OPE,,M>,-                      ;  Output buffer parity error
        <IPE,,M>,-                      ;  Input buffer parity error
        <XMPE,,M>,-                     ;  Xmit buffer parity error
        <RBPE,,M>,-                     ;  Read packet buffer parity error
        <LSPE,,M>,-                     ;  Local store parity error
        <CSPE,,M>,-                     ;  Control store parity error
        <PE,,M>,-                       ;  Parity error, OR of <14:8>
        >                               ;

.=.+<3*4>                               ; 3 unused register slots

        $DEF    PA_MADR     .BLKL 1     ; Maint addr register (start
                                        ;  ucode addr of PSA = 1).  Bits
                                        ;  <31:14> MBZ.

        $DEF    PA_MDATR    .BLKL 1     ; Maint data register; valid
                                        ;  iff port is in uninited state

.=^X900                                 ; Gap in register space.

        $DEF    PA_PS       .BLKL 1     ; Port status register

        $FIELD  PA_PS,0,<-              ; Define PS reg fields:
        <RQA,,M>,-                      ;  Response Q entry avail
        <MFQE,,M>,-                     ;  Message free Q empty
        <PDC,,M>,-                      ;  Port disable complete
        <PIC,,M>,-                      ;  Port init complete
        <DSE,,M>,-                      ;  Data struct error
        <MSE,,M>,-                      ;  Memory system error, exact
                                        ;   in CNF
        <SE,,M>,-                       ;  Sanity/boot timer expired
        <,24>,-                         ;  24 unused bits
        <MTE,,M>,-                      ;  Maint error, exact error in
                                        ;   PMC/CNF
        >                               ;

        $DEF    PA_PQBBR    .BLKL 1     ; Port Queue Block base register,
                                        ;  contains physical addr of base
                                        ;  of page aligned PQB.

        $DEF    PA_CQ0      .BLKL 1     ; Command queue 0 control reg

        $FIELD  PA_CQ0,0,<-             ; Define only bit:
        <CQC,,M>,-                      ;  Write 1 to start command queue
        >                               ;

        $DEF    PA_CQ1      .BLKL 1     ; Command queue 1 control reg

        $FIELD  PA_CQ1,0,<-             ; Define only bit:
        <CQC,,M>,-                      ;  Write 1 to start command queue
        >                               ;

        $DEF    PA_CQ2  .BLKL   1       ; Command queue 2 control reg
```

```
        _VIELD  PA_CQ2,0,<-                     ; Define only bit:
        ?CQC,,M>,-                              ;  Write 1 to start command queue
        >

        $DEF    PA_CQ3  .BLKL   1               ; Command queue 3 control reg

        _VIELD  PA_CQ3,0,<-                     ; Define only bit:
        ?CQC,,M>,-                              ;  Write 1 to start command queue
        >                                       ;

        $DEF    PA_PSR    .BLKL 1               ; Port status release ctl reg

        _VIELD  PA_PSR,0,<-                     ; Define only bit:
        ?PSC,,M>,-                              ;  Write 1 to release PSR
        >                                       ;

        $DEF    PA_PEC    .BLKL 1               ; Port enable control reg

        _VIELD  PA_PEC,0,<-                     ; Define only bit:
        ?PEC,,M>,-                              ;  Write 1 to move port
        >                                       ;    from disabled to enabled state

        $DEF    PA_PDC    .BLKL 1               ; Port disable control reg

        _VIELD  PA_PDC,0,<-                     ; Define only bit:
        ?PDC,,M>,-                              ;  Write 1 to move port from
        >                                       ;    enabled to disabled state

        $DEF    PA_PIC    .BLKL 1               ; Port init control register

        _VIELD  PA_PIC,0,<-                     ; Define only bit:
        ?PIC,,M>,-                              ;  Write 1 to move from uninit
        >                                       ;    to enabled or disabled state

        $DEF    PA_DFQ    .BLKL 1               ; Datagram free Q control reg

        _VIELD  PA_DFQ,0,<-                     ; Define only bit:
        ?DFQC,,M>,-                             ;  Write one when 1st buffer
        >                                       ;    inserted on dg free queue

        $DEF    PA_MFQ    .BLKL 1               ; Message free queue control reg

        _VIELD  PA_MFQ,0,<-                     ; Define only bit:
        ?MFQC,,M>,-                             ;  Write one when 1st buffer
        >                                       ;    inserted on msg free queue

        $DEF    PA_MTC    .BLKL   1             ; Maint timer control reg

        _VIELD  PA_MTC,0,<-                     ; Define only bit:
        ?MTC,,M>,-                              ;  Write one to reset sanity
        >                                       ;    or boot timer

        $DEF    PA_MTEC   .BLKL 1               ; Maint timer expiration control reg

        $DEF    PA_PFAR   .BLKL 1               ; Failing addr register contains
```

```
                                        ;  addr within same page as
                                        ;  failing addr on mem or data
                                        ;  store error
        $DEF    PA_PESR    .BLKL 1      ; Port error status reg

        $DEF    PA_PPR     .BLKL 1      ; Port parameter register

        _VIELD  PA_PPR,0,<-             ; Defined fields in PPR:
        <PN,8,M>,-                      ;  Port #
        <,8>,-                          ;  8 unused bits
        <BUFLEN,12,M>,-                 ;  size of internal buffers
        <MNODE,,M>,-                    ;  0/1 for 14/244 nodes allowed
        >                               ;

;
; Define constants:
;
        $EQU    PA_C_WCSSIZ     <1024*3>    ; Size of WCS in
                                            ;  48 bit microwds
        $EQU    PA_C_LSINDX     <^X800>     ; Longwd index to local
                                            ;  store measured from
                                            ;  PA_CNF
        $EQU    PA_C_LSLENGTH   <^X800>     ; # bytes of local store
        $EQU    PA_C_MCACHESZ   <3>         ; Free msg cache size
        $EQU    PA_C_DCACHESZ   <3>         ; Free dg cache size
        $EQU    PA_C_UCODEST    <^X400>     ; Microcode start address

        $DEFEND PAREG,$GBL,DEF

.SHOW
.ENDM   $PAREGDEF
```

```
;+
; $PAUCBDEF -- Defined CI extension to UCB.
;-

        .MACRO  $PAUCBDEF,$GBL

        .NOSHOW

        $DEFINI UCB,$GBL

.=UCB$L_DPC+4                                   ; Position to end of
                                                ;  standard UCB for
                                                ;  error logging devices

        _VIELD  UCB,0,<-                        ; Define bits for UCB$W_DEVSTS
        <,1>,-                                  ;  Unused
        <FKLOCK,,M>,-                           ;  Fork block interlock bit
        <MSGFKLOCK,,M>,-                        ;  Fork block interlock for
                                                ;   printing operator msgs
        >                                       ;

        $DEF    UCB$L_MSGFKBLK  .BLKL   6       ; Fork block for starting
                                                ;  error messages to operator

        $DEF    UCB$T_OPA0_TEMP .BLKL   6       ; Field used to store optional
                                                ;  _OPA0 error logging info
                                                ; Access to this field is also
                                                ;  protected by UCB_V_MSGFKLOCK

        $EQU    UCB$K_LMPKTBYTS         64      ; Max number of data bytes
                                                ;  logged from an erroreneous
                                                ;  packet

        $EQU    UCB$K_ERRDGBYTS         180     ; Max number of data bytes
                                                ;  logged from an errorlog
                                                ;  datagram -- any more will
                                                ;  be truncated.

        ; Logged Message working buffer

        $DEF    UCB$B_LMEST     .BLKB   1       ; error subtype
        $DEF    UCB$B_LMET      .BLKB   1       ; error type (& crash port)
        $DEF    UCB$B_LMERTCNT  .BLKB   1       ; error retry count
        $DEF    UCB$B_LMERTMAX  .BLKB   1       ; max error retry count
        $DEF    UCB$W_LMERRCNT  .BLKW   1       ; accumulated errors count
                                .BLKW   1       ; unused word
        $DEF    UCB$N_LSADDR    .BLKB   6       ; local station address
        $EQU    UCB$S_LSADDR    6               ; local station address size
        $DEF    UCB$N_LSID      .BLKB   6       ; local station id
        $EQU    UCB$S_LSID      6               ; local station id size
        $DEF    UCB$N_RSADDR    .BLKB   6       ; remote station address
        $EQU    UCB$S_RSADDR    6               ; remote station address size
        $DEF    UCB$N_RSID      .BLKB   6       ; remote station id
        $EQU    UCB$S_RSID      6               ; remote station id size
        $DEF    UCB$L_CICMD     .BLKL   1       ; CI pkt. command/control/status
```

```
        $DEF    UCB$W_MSGBYTCNT .BLKW   1       ; CI packet byte count
        $DEF    UCB$W_MSGPPDTYP .BLKW   1       ; CI packet PPD type
        $DEF    UCB$T_MSGDATA                   ; CI packet message data
                        .BLKB   UCB$K_LMPKTBYTS

UCB$K_LMBUFSIZ = . - UCB$B_LMEST                ; Total size of logged message
                                                ;  buffer

                        .BLKB   UCB$K_ERRDGBYTS-UCB$K_LMPKTBYTS

UCB$K_ERRDGSIZ = . - UCB$B_LMEST                ; Total size of logged message
                                                ;  buffer in case it contains
                                                ;  an error log dg.

                        .BLKL   2               ; Reserved

UCB$C_PASIZE = .                                ; Total size of port driver UCB

        $DEFEND UCB,$GBL,DEF

        .SHOW

        .ENDM   $PAUCBDEF
```

```
;+
; PPD MESSAGE FORMAT DEFINITIONS.
;
; THIS STRUCTURE DEFINES THE PPD HEADER LAYER WHICH PRECEDES THE SCS LAYER
; AND ALSC THE FORMATS FOR PORT COMMANDS AND RESPONSES OTHER THAN SCS
; AND APPLICATION DATAGRAMS AND MESSAGES.
;-

        .MACRO  $PPDDEF,$GBL

        .NOSHOW

        $DEFINI PPD,$GBL


                                                ;MARK 0 OFFSET

$DEF    PPD$L_FLINK     .BLKL           ;QUEUE FORWARD LINK
$DEF    PPD$L_BLINK     .BLKL           ;QUEUE BACKWARD LINK
$DEF    PPD$W_SIZE      .BLKW           ;STRUCTURE SIZE
$DEF    PPD$B_TYPE      .BLKB           ;STRUCTURE TYPE
$EQU    PPD$V_DISPOSE   0               ; 0/1-->RETURN BUFFER TO
$EQU    PPD$M_DISPOSE   1               ;  POOL/SYSAP
$DEF    PPD$B_SWFLAG    .BLKB           ;SOFTWARE FLAGS
$DEF    PPD$B_PORT      .BLKB           ;PORT #
$EQU    PPD$V_ERR       0               ; 0/1 IF NO/ANY ERROR OCCURRED
$EQU    PPD$M_ERR       1
$EQU    PPD$V_PTH0      1               ; PATH 0 STATUS
$EQU    PPD$S_PTH0      2
$EQU    PPD$M_POSTS     <^X6>           ; MASK FOR ALL PATH 0 STATUS
$EQU    PPD$M_P1STS     <^X18>          ; MASK FOR ALL PATH 1 STATUS
$EQU    PPD$V_PTH1      3               ; PATH 1 STATUS
$EQU    PPD$S_PTH1      2
$EQU    PPD$V_STSTYP    5               ; STATUS TYPE CODE
$EQU    PPD$S_STSTYP    3
                                        ; PATH STATUS VALUES:
                                        ;  0 ORIGIN, INCREMENTS OF 1
$EQU    PPD$C_PTHACK    0               ;   ACK'ED (SUCCESS) OR NOT USED
$EQU    PPD$C_PTHNAK    1               ;  NEGATIVE ACK'ED (XMIT FAILURE)
$EQU    PPD$C_PTHNO_RSP 2               ;  NO RESPONSE (NO REMOTE PORT)
$EQU    PPD$C_PTHARB    3               ;  CI ARBITRATION TIMEOUT
                                        ; STATUS (STSTYP) VALUES:
                                        ;  0 ORIGIN, INCREMENTS OF 1
$EQU    PPD$C_TYPOK     0               ;  OK STATUS
$EQU    PPD$C_TYPVCC    1               ;  VC CLOSED
$EQU    PPD$C_TYPINVBN  2               ;  INVALID BUFFER NAME
$EQU    PPD$C_TYPBLV    3               ;  BUFFER LENGTH VIOLATION
$EQU    PPD$C_TYPACCV   4               ;  ACCESS VIOLATION
$EQU    PPD$C_TYPNP     5               ;  NO PATH
$EQU    PPD$C_TYPBMSE   6               ;  BUFFER MEMORY SYSTEM ERROR
$EQU    PPD$C_TYPOTHER  7               ;  OTHER -- SEE STATUS SUBTYPE
                                                ; ERROR BIT
$EQU    PPD$V_STSST     1               ; STATUS SUBTYPE
$EQU    PPD$S_STSST     4
                                        ; STATUS SUBTYPE VALUES:
                                        ;  0 ORIGIN, INCREMENTS OF 1
```

```
$EQU    PPD$C_STPSV     0               ;  PKT SIZE VIOLATION
$EQU    PPD$C_STURP     1               ;  UNRECOGNIZED PKT
$EQU    PPD$C_STINVDP   2               ;  INVALID DESTINATION PORT
$EQU    PPD$C_STURC     3               ;  UNRECOGNIZED COMMAND
$EQU    PPD$C_STABO     4               ;  ABORT (PORT DISABLED)
$EQU    PPD$C_OSEQ      5               ;  OUT OF SEQUENCE MSG RECEIVED
$EQU    PPD$C_VCDCL     6               ;  MSG RECEIVED ON CLOSED VCD

$DEF    PPD$B_STATUS    .BLKB           ;PACKET STATUS (RESPONSE ONLY)
                                        ; OPCODE VALUES:
$EQU    PPD$C_SNDDG     1               ; SEND DG/DG SENT
$EQU    PPD$C_SNDMSG    2               ; SEND MSG/MSG SENT
$EQU    PPD$C_RETCNF    3               ; CONFIRM RETURN/RETURNED
$EQU    PPD$C_REQDAT    8               ; REQUEST DATA/DATA REQUESTED
$EQU    PPD$C_REQDAT0   8               ; REQUEST DATA @ PRIORITY 0
$EQU    PPD$C_REQDAT1   9               ; REQUEST DATA @ PRIORITY 1
$EQU    PPD$C_REQDAT2   10              ; REQUEST DATA @ PRIORITY 2
$EQU    PPD$C_SNDDAT    16              ; SEND DATA/DATA SENT
$EQU    PPD$C_RETDAT    17              ; RETURN DATA/DATA RETURNED
$EQU    PPD$C_INVTC     24              ; INVALIDATE TRANSLATION CACHE
$EQU    PPD$C_SETCKT    25              ; OPEN CIRCUIT/CIRCUIT OPENED
$EQU    PPD$C_RDCNT     26              ; READ COUNTERS/COUNTERS READ
$EQU    PPD$C_SNDLB     13              ; SEND LOOPBACK DATAGRAM
$EQU    PPD$C_REQID     5               ; REQUEST ID/ID REQUESTED
$EQU    PPD$C_SNDRST    6               ; SEND RESET/RESET SENT
$EQU    PPD$C_SNDMDAT   18              ; SEND MAINT DATA/DATA SENT
$EQU    PPD$C_REQMDAT   14              ; REQ MAINT DATA/DATA REQ'D
$EQU    PPD$C_SNDSTRT   7               ; SEND START/START SENT
$EQU    PPD$C_DGREC     33              ; DATAGRAM REC'D
$EQU    PPD$C_MSGREC    34              ; MESSAGE REC'D
$EQU    PPD$C_CNFREC    35              ; CONFIRM REC'D
$EQU    PPD$C_DATREC    49              ; DATA REC'D
$EQU    PPD$C_LBREC     45              ; LOOPBACK DG REC'D
$EQU    PPD$C_IDREC     43              ; ID REC'D
$EQU    PPD$C_MDATREC   51              ; MAINT DATA REC'D
$EQU    PPD$C_MCNFREC   41              ; MAINT CONFIRM REC'D
$DEF    PPD$B_OPC       .BLKB           ; PORT OPCODE
$EQU    PPD$V_RSP       0               ; GIVE RESPONSE TO COMMAND
$EQU    PPD$M_RSP       1
$EQU    PPD$V_PS        1               ; PATH SELECT OR RECV PATH
$EQU    PPD$S_PS        2
                                        ;  ON LOOPBACK DG REC'D
                                                ; RESERVED
$EQU    PPD$V_M 4               ; MULTIPLE VALUE, BLK XFER CMD ONLY
$EQU    PPD$S_M 3
$EQU    PPD$V_P 7               ; 0/1 FOR 512/576 DATA PKT SIZE
$EQU    PPD$M_P 128
                                        ;  OR PACKING FORMAT FOR MESSAGES
                                        ;  TO PDP 10/20 PORTS.
                                        ; PATH SELECT CHOICES:
$EQU    PPD$C_PSAUTO    0               ;  AUTOMATIC SELECT
$EQU    PPD$C_PSP0      1               ;  USE PATH 0
$EQU    PPD$C_PSP1      2               ;  USE PATH 1
                                        ;
                                                ; MBZ BIT
$EQU    PPD$V_RP        1               ; REC'V PATH
```

```
$EQU    PPD$S_RP        2
                                        ; MBZ BIT
$EQU    PPD$V_SP        4               ; SEND PATH
$EQU    PPD$S_SP        2
                                        ; MBZ BITS
$EQU    PPD$V_FORCE     7               ; FORCE RESET FLAG
$EQU    PPD$M_FORCE     <^X80>          ;
$EQU    PPD$V_DSTART    7               ; DEFAULT START ADDR FLAG
$EQU    PPD$M_DSTART    <^X80>          ;

                                        ;MARK START OF OPCODE SPECIFIC FIELDS

$DEF    PPD$B_FLAGS     .BLKB           ;FLAGS ON COMMANDS
        . = 15                  ;MESSAGE AND DATAGRAM FORMAT:

                        .BLKB   1
                                        ; INCLUDING PPD$W_LENGTH
$DEF    PPD$W_LENGTH    .BLKW           ;MESSAGE SIZE IN BYTES NOT
$DEF    PPD$C_LENGTH
$DEF    PPD$K_LENGTH                    ;SIZE OF PPD LAYER
                                        ; LEGAL PPD TYPE CODES:
                                        ; ORIGIN OF 0, INCR OF 1
$EQU    PPD$C_START     0               ; START DATAGRAM
$EQU    PPD$C_STACK     1               ; STACK DATAGRAM
$EQU    PPD$C_ACK       2               ; ACK DATAGRAM
$EQU    PPD$C_SCS_DG    3               ; SCS DATAGRAM
$EQU    PPD$C_SCS_MSG   4               ; SCS MESSAGE
$EQU    PPD$C_ELOG      5               ; ERROR LOG DATAGRAM
$EQU    PPD$C_HOSTSHUT  6               ; HOST SHUTDOWN DATAGRAM

$EQU    PPD$C_CACHECLR  <^X8000>                ; CACHE CLEAR MARKER MSG

                                        ; MARK START OF PPD START
                                ;  HANDSHAKE DATAGRAMS

$DEF    PPD$W_MTYPE     .BLKW           ;PPD TYPE CODE
        . = 15                  ;BLOCK XFER COMMANDS, RESPONSES,
                                        ; AND CONFIRMS

                        .BLKB   1
$DEF    PPD$Q_XCT_ID    .BLKQ           ;TRANSACTION ID
$DEF    PPD$L_XCT_LEN   .BLKL           ;TRANSFER SIZE (BYTES)
$DEF    PPD$L_SND_NAME  .BLKL           ;NAME OF SENDING BUFFER
$DEF    PPD$L_SND_BOFF  .BLKL           ;BYTE OFFSET OF START OF SEND BUFFER
$DEF    PPD$L_REC_NAME  .BLKL           ;NAME OF RECEIVING BUFFER

$DEF    PPD$L_REC_BOFF  .BLKL           ;BYTE OFFSET OF RECEIVING BUFFER
        . = 15                  ;VIRTUAL CIRCUIT OPEN/CLOSE

                        .BLKB   1
$DEF    PPD$W_MASK      .BLKW           ;MASK TO MODIFY VCB
                                                ;RESERVED FOR SOFTWARE
                        .BLKW   1
$DEF    PPD$W_M_VAL     .BLKW           ;VCD MODIFICATION VALUE
                                                ;RESERVED FOR SOFTWARE
                        .BLKW   1
```

```
                                        ; MODIFY
                                                ; 9 MBZ BITS
$EQU    PPD$V_PSTS      9               ; PATH STATE:  0 FOR BOTH BAD;
$EQU    PPD$S_PSTS      2
                                        ;  3 FOR BOTH GOOD, ETC.
                                                ; 1 MBZ BIT
$EQU    PPD$V_DQI       12              ; INHIBIT DATAGRAMS
$EQU    PPD$M_DQI       <^X1000>
$EQU    PPD$V_NS        13              ; SEND SEQUENCE NUMBER
$EQU    PPD$M_NS        <^X2000>
$EQU    PPD$V_NR        14              ; RECV SEQUENCE NUMBER
$EQU    PPD$M_NR        <^X4000>
$EQU    PPD$V_CST       15              ; VC STATE = 0/1 FOR CLOSED/OPEN
$EQU    PPD$M_CST       <^X8000>

$DEF    PPD$L_IN_VCD    .BLKL           ;INITIAL VALUE OF VCD BEFORE
        . = 15                  ;READ EVENT COUNTER COMMAND
                                        ; AND RESPONSE

                        .BLKB   1

$DEF    PPD$L_P0_ACK    .BLKL           ;# ACK'S ON PATH 0
$DEF    PPD$L_P0_NAK    .BLKL           ;# NAK'S ON PATH 0
$DEF    PPD$L_P0_NRSP   .BLKL           ;# NO RESPONSES ON PATH 0
$DEF    PPD$L_P1_ACK    .BLKL           ;# ACK'S ON PATH 1
$DEF    PPD$L_P1_NAK    .BLKL           ;# NAK'S ON PATH 1

$DEF    PPD$L_P1_NRSP   .BLKL           ;# NO RESPONSES ON PATH 1
$DEF    PPD$L_DG_DISC   .BLKL           ;# DGS DISCARDED
        . = 15                  ;REQID COMMAND AND RESPONSE

                        .BLKB   1
                                                ;TRANSACTION ID, PREVIOUSLY DEFINED
                        .BLKQ   1
$EQU    PPD$V_PORT_TYP  0               ; PORT TYPE CODE,
$EQU    PPD$S_PORT_TYP  31
$EQU    PPD$V_DUALPATH  31              ; 0/1 FOR SINGLE/DUAL PATH
$EQU    PPD$M_DUALPATH  <^X80000000>
$DEF    PPD$L_RPORT_TYP .BLKL           ;REMOTE PORT TYPE
$DEF    PPD$L_RPORT_REV .BLKL           ;CODE REVISION OFR REMOTE PORT
$DEF    PPD$L_RPORT_FCN .BLKL           ;REMOTE PORT FUNCTION MASK
$DEF    PPD$B_RST_PORT  .BLKB           ;PORT # OF RESETTING PORT (IF ANY)
$EQU    PPD$V_MAINT     0               ; 0/1 FOR NO/YES MAINT STATE
$EQU    PPD$M_MAINT     1
$EQU    PPD$V_STATE     1               ; STATE
$EQU    PPD$S_STATE     2
                                        ; STATES ARE:
$EQU    PPD$C_UNINIT    0               ; UNINITIALIZED,
$EQU    PPD$C_DISAB     1               ; DISABLED,
$EQU    PPD$C_ENAB      2               ; ENABLED

$DEF    PPD$B_RSTATE    .BLKB           ;REMOTE PORT STATE
        . = 15                  ;SEND MAINT START COMMAND AND RESPONSE
                        .BLKB   1
                                                ;TRANSACTION ID, PREVIOUSLY DEFINED
                        .BLKQ   1
```

```
$DEF    PPD$L_ST_ADDR   .BLKL            ;START ADDRESS
$EQU    PPD$V_DS        0               ; 0/1 FOR NO/YES USE DEFAULT ADDR
$EQU    PPD$M_DS        1

$DEF    PPD$B_DEF_ST    .BLKB           ;DEFAULT FLAG:
        . = 15                  ;TEMPLATE LOOPBACK DG FORMAT:

                        .BLKB   1
                                                ;DG LENGTH (PPD$W_LENGTH)
$EQU    PPD$C_LBDAT_LEN 48                      ;# BYTES OF LOOPBACK DATA
                        .BLKW   1
$DEF    PPD$B_LBDATA    .BLKB   48      ;LOOPBACK DATA SPACE
$DEF    PPD$L_LBCRC     .BLKL           ;CRC ON LB DG
$DEF    PPD$C_LB_LENGTH

$DEF    PPD$K_LB_LENGTH                 ;SIZE OF TEMPLATE
        . = 0                   ;OFFSETS IN TEMPORARY BUFFER
                                ; USED TO COMPUTE CRC:

                        .BLKB   <3*4>   ;VMS HEADER SPACE
$DEF    PPD$W_LCB_LEN7  .BLKW           ;LENGTH + 7
$DEF    PPD$B_LCB_PORT  .BLKB           ;DESTINATION PORT
$DEF    PPD$B_LCB_NPORT .BLKB           ;COMPLEMENT OF DEST PORT
$DEF    PPD$B_LCB_LPORT .BLKB           ;LOCAL PORT (=DESTINATION)
$DEF    PPD$B_LCB_OPC   .BLKB           ;SNDLB OPCODE
$DEF    PPD$B_LCB_0     .BLKB           ;0

$DEF    PPD$C_LCB_DATA  .BLKB           ;START OF DATA AREA
        . = 18                  ;PPD START HANDSHAKE MESSAGES:
                        .BLKB   2
$DEF    PPD$B_SYSTEMID  .BLKB   6       ;SENDING SYSTEM ID
$DEF    PPD$B_PROTOCOL  .BLKB   1       ;PPD PROTOCOL REV LEVEL

$EQU    PPD$C_PRT_BASE  0               ; 1ST PPD PROTOCOL REV
$EQU    PPD$C_PRT_ELOG  1               ; 2ND REV, SUPPORTS ERROR LOG DGS

                        .BLKB   1       ;RESERVED MBZ BYTE
$DEF    PPD$W_MAXDG     .BLKW           ;MAX DG SIZE
$DEF    PPD$W_MAXMSG    .BLKW           ;MAX MSG SIZE
$DEF    PPD$T_SWTYPE    .BLKB   4       ;SOFTWARE TYPE
$DEF    PPD$T_SWVERS    .BLKB   4       ;SOFTWARE VERSION
$DEF    PPD$Q_SWINCARN  .BLKQ           ;SOFTWARE INCARNATION #
$DEF    PPD$T_HWTYPE    .BLKB   4       ;PROCESSOR HARDWARE TYPE
                                        ;PPD MSG/DG LENGTHS:
$EQU    PPD$C_START_LEN 62              ; START DG
$EQU    PPD$C_STACK_LEN 62              ; STACK DG
$EQU    PPD$C_ACK_LEN   4               ; ACK DG
$EQU    PPD$C_CACHE_LEN 2               ; CACHE CLEAR MARKER MSG
$EQU    PPD$C_HSHUT_LEN 2               ; HOST SHUTDOWN DATAGRAM

$DEF    PPD$B_HWVERS    .BLKB   12      ;PROCESSOR HARDWARE VERSION
$DEF    PPD$Q_NODENAME  .BLKQ   1       ;NODE NAME
$DEF    PPD$Q_CURTIME   .BLKQ   1       ;CURRENT SYSTEM TIME MEASURED IN
                                        ;  100 NSEC UNITS
$DEF    PPD$C_MIN_DGSIZ                 ;MINIMUM ALLOWED DG SIZE (INCLUDING
                                        ; PPD HEADER)
```

```
        $DEFEND PPD,$GBL,DEF
        .SHOW
        .ENDM   $PPDDEF
```

```
;+
; $REM_DFREEQ -- Remove a datagram buffer from the tail of the datagram free queue.
; $REM_MFREEQ -- Remove a message buffer from the tail of the message free queue.
;
; Inputs:
;
;       R4                      -Addr of PDT
;       ERRADDR                 -Branch addr in case queue interlock unobtainable
;
; Outputs:
;
;       V bit                   -0/1 = success/no queue entry
;       R0                      -Destroyed
;       R2                      -Addr of buffer if success
;-

        .MACRO  $REM_DFREEQ ERRADDR=INT$FATALQ_RDFQ

         $QRETRY        REMQTI @PDT$L_DFQHDR(R4),R2,ERROR=ERRADDR

        .ENDM

        .MACRO  $REM_MFREEQ ERRADDR=INT$FATALQ_RMFQ

         $QRETRY        REMQTI @PDT$L_MFQHDR(R4),R2,ERROR=ERRADDR

        .ENDM

        .MACRO  $REM_RESPQ ERRADDR=INT$FATALQ_RSPQ

         $QRETRY        REMQHI PDT$Q_RSPQ(R4),R2,ERROR=ERRADDR

        .ENDM
```

```
;+
; $REQID -- Format the PPD datagram and put on low priority port command queue.
;
; Inputs:
;
;       R2                      -Addr of datagram buffer
;       R4                      -Addr of PDT
;       RETFLAG                 -TRUE/FALSE or 1/0 for response
;                                wanted.  If an opcode, must be a longword
;       PATH                    -AUTO/P0/P1 or 0/1/2.  If an
;                                opcode, must be a longword.
;
;       PORT                    -Remote port # to request ID of
;       XCT_ID                  -Transaction ID to place in REQID
;
; Outputs:
;
;       R0                      -Destroyed
;-

        .MACRO  $REQID RETFLAG=TRUE,PRIORITY=LOW,PATH=AUTO,PORT,XCT_ID

        $DEFFLAGS REQID,RETFLAG,PATH
         CLRL   R0                      ; Zero flags, opcode, port

         .IF    DIF     RETFLAG,TRUE    ; If RETFLAG is neither
         .IF    DIF     RETFLAG,FALSE   ;  true nor false string, then
         ASHL   #<PPD$V_RSP+24>,-       ;  it is an opcode
                RETFLAG,R0              ; Position flag in R0
         .ENDC
         .ENDC

        MOVB    PORT,R0                 ; Insert port number
        BISL3   #$$$$FLAGS,R0,PPD$B_PORT(R2)  ; Add flags and opcode

        .IF     DIF     PATH,AUTO       ; If PATH argument wasn't
        .IF     DIF     PATH,P0         ;  a string, then it was
        .IF     DIF     PATH,P1         ;  an opcode.
        ASHL    #<PPD$V_PS+24>,-        ; Position path select
                PATH,R0                 ;
        BISL    R0,PPD$B_PORT(R2)       ; Set in message header
        .ENDC
        .ENDC
        .ENDC

        .IF     B       XCT_ID
        CLRQ    PPD$Q_XCT_ID(R2)

        .IFF
        MOVQ    XCT_ID,PPD$Q_XCT_ID(R2)
        .ENDC

        $INS_COMQ'PRIORITY

.ENDM
```

```
;+
; $SETCKT -- Fill in a dg buffer with a SETCKT command and issue to
;            the port.
;
; Inputs:
;
;       R2                      -Addr of datagram buffer
;       R4                      -PDT addr
;
;       RETFLAG                 -TRUE/FALSE if response on success
;                                is/is'nt wanted
;       PRIORITY                -LOW/HIGH
;       PORT                    -Port # to do SETCKT on
;       MASK                    -Mask of bits to modify in VC descriptor
;       MVAL                    -Values to set for bits specified in MASK
;       DQI                     -INHIB/ENAB for dgs are/aren't being
;                                inhibited.  NOCH means no change.
;       NOTIFY                  -TRUE/FALSE for do/don't notify
;                                ERR$VCCLOSED_MSG
;
; Outputs:
;
;       R0                      -Destroyed
;       Other registers         -Preserved
;       PDT$B_DQIMAP(R4)        -Bit set/clear if dgs inhibited/enabled
;                                by DQI parameter.
;-

        .MACRO  $SETCKT RETFLAG=FALSE,PRIORITY=LOW,PORT,MASK,MVAL,DQI=NOCH,NOTIFY=FALSE,?INQ

        $DEFFLAGS SETCKT,RETFLAG                ; Define $$$FLAGS

        MOVZBL  PORT,R0                         ; Get port #
        BISL3   #$$$FLAGS,R0,-                  ; Set up opcode, port, response
                PPD$B_PORT(R2)                  ;  bit in dg buffer
        MOVZWL  MASK,PPD$W_MASK(R2)             ; Set mask of bits to modify
        MOVZWL  MVAL,PPD$W_M_VAL(R2)            ;  and values to modify to

        .IF     IDN     NOTIFY,TRUE             ; If this is for ERR$VCCLOSED,
        MOVB    #PPD$M_DISPOSE,-                ;  then flag it
                PPD$B_SWFLAG(R2)                ;
        .ENDC

        .IF     IDN     DQI,TRUE                ; If dgs being inhibited, then
        BBSS    R0,PDT$B_DQIMAP(R4),INQ         ;  set dg inhibit
        .ENDC                                   ;
        .IF     IDN     DQI,FALSE               ; If dgs being enabled, then
        BBCC    R0,PDT$B_DQIMAP(R4),INQ         ;  clear dg inhibit
        .ENDC                                   ;

INQ:
        .IF     DF      PA$DEBUG                ; Debug facility
        BSBW    TRC$LOGMSG                      ; Log packet
        .ENDC

        $INS_COMQ'PRIORITY'                     ; !ssue command
```

```
        .FNDM   $SETCKT
```

```
;+
;$SNDDG -- Fill in PPD header and queue datagram to port.
;
; Inputs:
;
;       R2                      -Addr of dg buffer
;       R3                      -Addr of CDT, optional
;       R4                      -Addr of PDT
;
;       RETFLAG                 -TRUE/FALSE/opcode. FALSE --> sent dg goes
;                                to free queue;  TRUE --> sent dg goes
;                                to response queue and is disposed of
;                                according to the DISPOSE param.
;                                May be expressed as an opcode = 1/0
;                                for TRUE/FALSE.  PPD$V_RSP is assumed
;                                = 0.
;       PRIORITY                -LOW/HIGH
;       PATH                    -AUTO/P0/P1
;       DISPOSAL                -POOL/SYSAP/opcode.  Sent dg is deallocated
;                                to nonpaged pool or give to SYSAP dg
;                                input routine.  Irrelevant if RETFLAG =
;                                FALSE.  May be expressed as an opcode =
;                                1/0 for SYSAP/POOL.
;       PORT                    -Optional remote port #.  If PORT is
;                                omitted, then R# is assuemd to have a
;                                CDT with valid remote station addr.
;
; Outputs:
;
;       R0                      -Destroyed
;       Other registers         -Preserved
;-

        .MACRO  $SNDDG  RETFLAG=FALSE,PRIORITY=HIGH,PATH=AUTO,DISPOSAL=POOL,-
                        PORT

        $DEFFLAGS SNDDG,RETFLAG,PATH

         .IF B PORT
         BISL3  #$$$FLAGS,-                     ; To flags, add
                CDT$B_RSTATION(R3),-            ;  remote port and put
                PPD$B_PORT(R2)                  ;  into dg buffer

         .IFF
         MOVZBL PORT,PPD$B_PORT(R2)             ; Get caller-specified port
         BISL   #$$$FLAGS,-                     ; Add flags to port
                PPD$B_PORT(R2)                  ;  in dg buffer header
         .ENDC

         $IFNTF RETFLAG,,-                      ; If RETFLAG is an opcode,
         <BISB  RETFLAG,PPD$B_FLAGS(R2)>        ;  OR it into the flags

         .IF IDN DISPOSAL,POOL                  ; If return dg to pool
         CLRB   PPD$B_SWFLAG(R2)                ;  clear flag
         .ENDC                                  ;
```

```
        .IF  IDN DISPOSAL,SYSAP                 ; If returning dg to SYSAP,
        BISB    #PPD$M_DISPOSE,-                ;  set flag to show return
                PPD$B_SWFLAG(R2)                ;  to sysap
        .ENDC

        $IFNTF  DISPOSAL,POOL,SYSAP,-           ; If DISPOSAL is opcode,
        <MOVB   DISPOSAL,PPD$B_SWFLAG(R2)>      ;  set it now

        .IF     DF PA$DEBUG                     ; Debug facility
        BSBW    TRC$LOGMSG                      ; Log dg send
        .ENDC

        $INS_COMQ'PRIORITY                      ; Send dg

       .ENDM    $SNDDG
```

```
;+
; $SNDMSG -- Fill in PPD header and queue sequenced message to port.
; $TURNMSG -- Turn a received message around for send.
;
; Inputs:
;
;       R2                      -Addr of message buffer
;       R3                      -Addr of CDT ($SNDMSG only)
;       R4                      -Addr of PDT
;
;       RETFLAG                 -TRUE/FALSE, for sent msg goes to
;                                response/free queue.
;       PRIORITY                -LOW/HIGH
;       PATH                    -AUTO/P0/P1
;       PPDTYP                  -PPD type code
;
; Outputs:
;
;       R0                      -Destroyed
;-

        .MACRO  $SNDMSG RETFLAG=FALSE,PRIORITY=HIGH,PATH=AUTO,PPDTYP=SCS_MSG

         $DEFFLAGS SNDMSG,RETFLAG,PATH          ; Define opcode+path+resp flag

         MOVW   #PPD$C_'PPDTYP',PPD$W_MTYPE(R2) ; Set PPD type = caller specified
         BISL3  #$$$FLAGS,-                     ; Combine flags
                CDT$B_RSTATION(R3),-            ;  remote port, and
                PPD$B_PORT(R2)                  ;  deposit in message
         $IFNTF RETFLAG,,,-                     ; If RETFLAG is an opcode,
         <BISB  RETFLAG,PPD$B_FLAGS(R2)>        ;  then or it into flags

         .IF    DF PA$DEBUG                     ; Debug facility
         BSBW   TRC$LOGMSG                      ; Log send msg
         .ENDC                                  ;

         $INS_COMQ'PRIORITY                     ; Send message

        .ENDM   $SNDMSG



        .MACRO  $TURNMSG RETFLAG=FALSE,PRIORITY=HIGH,PATH=AUTO

         $DEFFLAGS SNDMSG,RETFLAG,PATH          ; Define opcode+path+resp flag

         INSV   #$$$FLAGS@-8,#0,#24,-           ; Put opcode+flags in msg and
                PPD$B_STATUS(R2)                ;  zero status byte

         $IFNTF RETFLAG,,,-                     ; If RETFLAG is an opcode,
         <BISB  RETFLAG,PPD$B_FLAGS(R2)>        ;  then or it into the flags

         .IF    DF PA$DEBUG                     ; Debug facility
         BSBW   TRC$LOGMSG                      ; Log send message
```

```
        .ENDC                           ;
        $INS_COMQ'PRIORITY              ; Send message
       .ENDM    $TURNMSG
```

```
        .SBTTL  SCS MACROS

;+
; $CHK_CDTSTATE -- Verify that the CDT is in the specified state.
;                  If the ERROR argument is specified, fall through
;                  on matching state and branch to the ERROR addr
;                  on match failure.  If the SUCCESS argument is
;                  specified, then fall through on mismatch and
;                  branch to the SUCCESS address on match.
;
; Inputs:
;
;       CDT                     -Addr of CDT, must be a register
;       STATE                   -Desired CDT state
;       ERROR                   -Addr to got to on mismatch
;       SUCCESS                 -Addr to go to on match
;
; Outputs:
;
;       All registers           -Preserved
;-

        .MACRO  $CHK_CDTSTATE   STATE,ERROR,CDT=R3,SUCCESS,?CONTINUE

         CMPW   CDT$W_STATE('CDT'),-
                #CDT$C_'STATE'

         .IF    NB ERROR
         BEQL   CONTINUE

         BRW    ERROR
         .ENDC

         .IF    NB SUCCESS
         BNEQ   CONTINUE
         BRW    SUCCESS
         .ENDC

CONTINUE:

        .ENDM   $CHK_CDTSTATE
```

```
;+
; $RESUME_FP -- Resumes a suspended fork process resulting from
; $SUSP_SCS or $SUSP_FP.  Removes the next CDRP from the wait queue,
; restores context from the CDRP and calls the fork process thread
; back.
;
; Inputs:
;
;       WAITQHDR                -Header of queue of waiting CDRP's
;       QEMPTY                  -Addr to go to if nobody is waiting
;
; Outputs seen by routine doing $RESUME_FP:
;
;       R0-R2                   -Destroyed
;       Other registers         -Preserved
;-

        .MACRO  $RESUME_FP      WAITQHDR,QEMPTY,?CONTINUE

         REMQUE WAITQHDR,R0             ; Get next waiting CDRP

         .IF    B QEMPTY                ; If no queue empty addr,
         BVS    CONTINUE                ;  branch to continue addr
         .IFF                           ; Else,
         BVS    QEMPTY                  ; Branch if none
         .ENDC

         PUSHR  #^M<R3,R4,R5>           ; Save current context
         MOVL   R0,R5                   ; Get waiting CDRP addr
         JSB    G^SCS$RESUMEWAITR       ; Resume waiter
         POPR   #^M<R3,R4,R5>           ; Restore previous context

CONTINUE:

        .ENDM   $RESUME_FP
```

```
;+
; $SUSP_FP -- Suspends a fork process and returns to its caller's
; caller.  The resource wait counter is not incremented
; since this macro is used to suspend a process pending receipt
; of a response.  In this case no resource is awaited.
;
; Inputs:
;
;       R5                      -Addr of CDRP
;       WAITQHDR                -Optional addr of queue header
;                                on which to queue fork block
;       0(SP)                   -Fork process PC
;
; Outputs:
;
;       N/A
;-

        .MACRO  $SUSP_FP        WAITQHDR

         MOVQ   R3,CDRP$L_FR3(R5)       ; Save process' context
         POPL   CDRP$L_FPC(R5)          ; Copy return to process from
                                        ;  stack to CDRP

         .IF    NB WAITQHDR             ; If queuing is desired,
         INSQUE (R5),WAITQHDR           ;  queue CDRP on wait queue
         .ENDC

         RSB                            ; Return to caller's caller

        .ENDM   $SUSP_FP
```

```
;+
; $SUSP_SCS -- Suspends an SCS routine by saving context in the
; CDRP.  The saved PC is a continuation address in the SCS routine.
; Return is taken to the PC on the top of the stack.  Since
; suspension of the SCS routine normally implies suspension of the
; calling SYSAP too, the SCS routine should pop the return to the
; SYSAP from the stack into the level 1 return field of the CDRP
; (CDRP$L_SAVD_RTN) prior to invoking $SUSP_SCS.
;
; Inputs:
;
;       R5                      -Addr of CDRP
;       WAITQHDR                -Header of queue on which to insert
;                                waiting CDRP
;
; Outputs:
;
;       R0                      -Destroyed
;       @CDRP$L_RWCPTR(R5)      -Incremented if CDRP$L_RWCPTR is nonzero
;-

        .MACRO  $SUSP_SCS       WAITQHDR,?NO_RWC,?CONTINUE

         MOVQ   R3,CDRP$L_FR3(R5)       ; Save process' context
         MOVAL  CONTINUE,CDRP$L_FPC(R5) ; Put SCS continuation addr in CDRP
         MOVL   CDRP$L_RWCPTR(R5),R0    ; Get addr of resource wait count
         BEQL   NO_RWC                  ; Branch if none
         INCW   (R0)                    ; Else step count of threads waiting

NO_RWC:
         INSQUE (R5),WAITQHDR           ; Queue CDRP on wait queue
         RSB                            ; Return to PC on top of stack

CONTINUE:

        .ENDM   $SUSP_SCS
```

XADRIVER
MAR

CONINTERR
LIS

XIDRIVER
MAR

PAMAC
MAR

CNDRIVER
LIS