C 8

```
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  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD         DDD  RRRRRRRRRRRR           III        VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD         DDD  RRRRRRRRRRRR           III        VVV         VVV  EEEEEEEEEEEE     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
```

```
**FILE**ID**CNDRIVER
```

```
  CCCCCCCC  NN      NN  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
  CCCCCCCC  NN      NN  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
CC          NN      NN  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
CC          NN      NN  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
CC          NNNN    NN  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
CC          NNNN    NN  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
CC          NN  NN  NN  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
CC          NN  NN  NN  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
CC          NN    NNNN  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
CC          NN    NNNN  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
CC          NN      NN  DD      DD  RR    RR        II        VV  VV    EE          RR    RR          ....
CC          NN      NN  DD      DD  RR    RR        II        VV  VV    EE          RR    RR          ....
  CCCCCCCC  NN      NN  DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....
  CCCCCCCC  NN      NN  DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000            .TITLE  CNDRIVER - VAX/VMS DECnet-CI Class Driver
0000            .IDENT  'V04-000'
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;++
0000    ; FACILITY:
0000    ;
0000    ;       VAX/VMS DECnet-CI class driver
0000    ;
0000    ; ABSTRACT:
0000    ;
0000    ;       This module contains the DECnet-CI class driver FDT routines,
0000    ;       SCS dispatcher, and fork routines.
0000    ;
0000    ;
0000    ;
0000    ;       Kerbey T. Altmann, 17-Aug-1981
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-016 ADE3004         A. Eldridge                     24-Jul-1984
0000    ;               Change name back to DECNET$PHASE_III for now.  The change in the
0000    ;               name must be phased in by updating the receiver to accept the
0000    ;               old and new name before the transmitter can be updated to send
0000    ;               the new name.  The new name should be DECNET$CI rather than
0000    ;               DECNET$PHASE_IV.
0000    ;
0000    ;       V03-015 LMP0275         L. Mark Pilant,                 12-Jul-1984  12:26
0000    ;               Initialize the ACL info in the ORB to be a null descriptor
0000    ;               list rather than an empty queue.  This avoids the overhead
0000    ;               of locking and unlocking the ACL mutex, only to find out
0000    ;               that the ACL was empty.
0000    ;
0000    ;       V03-014 LMP0221         L. Mark Pilant,                 26-Mar-1984  16:43
0000    ;               Change UCB$L_OWNUIC to ORB$L_OWNER.
0000    ;
```

```
0000   ;       V03-013 TMK0002         Todd M. Katz            24-Mar-1984
0000   ;               When connecting to a remote station over a specific local
0000   ;               port, which is what this DECnet class driver does, the name of
0000   ;               the local port together with the remote station address must be
0000   ;               specified as an arguement to the CONNECT fork process call. This
0000   ;               DECnet class driver was specifying the name of the local port
0000   ;               as PAA. It should now be specifying the name of the local port
0000   ;               as PAA0. If this is not done, the CONNECT will fail.
0000   ;
0000   ;               Change the SCS process name of the DECnet SYSAP from
0000   ;               DECNET$PHASE_III to DECNET$PHASE_IV.
0000   ;
0000   ;       V03-012 TMK0001         Todd M. Katz            08-Feb-1984
0000   ;               Use the macro SEND_DG_BUF_REG to do transmits instead of
0000   ;               SEND_DG_BUF. This allows me to remove the pseudo-CDRP which
0000   ;               is currently buried within the CDB. This false CDRP was only
0000   ;               being used to pass the application data and CDT addresses to
0000   ;               the fork process call, FPC$SENDDG. The  fork process call issued
0000   ;               by SEND_DG_BUF_REG, FPC$SENDRGDG, requires these addresses to be
0000   ;               in registers when it is invoked, and thus, doesn't require a
0000   ;               CDRP in order to obtain them.
0000   ;
0000   ;       V03-011 ADE3003         Alan D. Eldridge        19-May-1983
0000   ;               Replaced constants with appropriate SBO$ symbols.
0000   ;
0000   ;       V03-010 ADE3002         Alan D. Eldridge        19-Apr-1983
0000   ;               Modified datagram internal SCS header "size" field to
0000   ;               handle new negative offset processing option.
0000   ;
0000   ;       V03-009 ADE3001         Alan D. Eldridge        2-Feb-1983
0000   ;               Simplified connect/disconnect control.  Removed the sending of
0000   ;               XON/XOFF sequenced messages.  Issue a CONNECT only if the
0000   ;               remote sequence number is higher.  Redefined the CDB.
0000   ;
0000   ;       V03-008 NPK3010         N. Kronenberg           24-Nov-1982
0000   ;               Removed output array specifier from CONFIG_SYS call.
0000   ;
0000   ;       V03-007 KTA0109         Kerbey T. Altmann       8-Jul-1982
0000   ;               Fix bug in returning buffer info in SENSEMODE.
0000   ;
0000   ;       V03-006 NPK3004         N. Kronenberg           2-Jul-1982
0000   ;               Modify START_TRIB to connect over specific virtual
0000   ;               circuit instead of looking up the remote system and
0000   ;               connecting to that.
0000   ;
0000   ;       V03-005 NPK3003         N. Kronenberg           1-Jul-1982
0000   ;               Fixed offsets from CONFIG_PTH/SYS for new format
0000   ;               returned by those routines.
0000   ;
0000   ;       V03-004 KDM0002         Kathleen D. Morse       28-Jun-1982
0000   ;               Added $PRDEF.
0000   ;
0000   ;       V03-003 KTA0097         Kerbey T. Altmann       20-Apr-1982
0000   ;               Fix bad branch destination.
0000   ;
0000   ;
0000   ;--
```

```
0000           .SBTTL  External and local symbol definitions
0000
0000    ;
0000    ; System definitions
0000    ;
0000
0000           $ADPDEF                              ; Adapter control block
0000           $CDTDEF                              ; Connection descriptor
0000           $CRBDEF                              ; Channel request block
0000           $CXBDEF                              ; Complex buffers
0000           $DCDEF                               ; Device classes and types
0000           $DDBDEF                              ; Device data block
0000           $DEVDEF                              ; Device characteristics
0000           $DPTDEF                              ; Driver prologue table defs
0000           $DYNDEF                              ; Control block defs
0000           $IODEF                               ; I/O function codes
0000           $IPLDEF                              ; Hardware IPL definitions
0000           $JIBDEF                              ; Job info block
0000           $IRPDEF                              ; I/O request packet
0000           $NMADEF                              ; Network Management definitions
0000           $ORBDEF                              ; Object's Rights Block
0000           $PBDEF                               ; Path block defininitions
0000           $PCBDEF                              ; Process control block
0000           $PDTDEF                              ; Port Descriptor Table
0000           $PRDEF                               ; Processor register definitions
0000           $SBDEF                               ; System block definitions
0000           $SBODEF                              ; System block output definitions
0000           $SSDEF                               ; System status codes
0000           $UCBDEF                              ; Unit control block
0000           $VECDEF                              ; Interrupt vector block
0000           $XMDEF                               ; XMDRIVER symbols
0000
0000    ;
0000    ; Local macros
0000    ;
0000    .MACRO  SETBIT  POS,BAS,?L                  ; Set a single bit
0000                    BBSS    POS,BAS,L
0000                  L:
0000    .ENDM   SETBIT
0000
0000    .MACRO  CLRBIT  POS,BAS,?L                  ; Clear a single bit
0000                    BBCC    POS,BAS,L
0000                  L:
0000    .ENDM   CLRBIT
0000
0000    .MACRO  PUSHQ   ARG                         ; Push a quadword
0000                    MOVQ    ARG,-(SP)           ; Save argument on stack
0000    .ENDM   PUSHQ
0000
0000    .MACRO  POPQ    ARG                         ; Pop a quadword
0000                    MOVQ    (SP)+,ARG           ; Restore argument
0000    .ENDM   POPQ
0000
```

```
          0000  
          0000  $DEFINI PARAM
          0000  
00000004  0000      COUNT_C_ENTRY = 2*2                       ; COUNT table entry size
0000000C  0000      PARAM_C_ENTRY = 2*6                       ; PARAM table entry size
          0000     _VIELD PRM,0,<-                            ; Parameter bits and sizes
          0000                  <TYPE,12,M>,-                 ; Parameter type
          0000                  <MIN,1,M>,-                   ; Parameter minimum value
          0000                  <MAX,1,M>,-                   ; Parameter maximum value
          0000                  <REQUIRE,1,M>,-               ; Parameter required flags
          0000                  <INVALID,1,M>,-               ; Parameter invalid flags
          0000                 >
          0000  
          0000     _VIELD OFF,0,<-                            ; Offset word fields
          0000                  <VALUE,10,M>,-                ; Offset value
          0000                  <WIDTH, 6,M>,-                ; Width of field in structure
          0000                 >
          0000  
          0000  $DEFEND PARAM
          0000  
          0000  .MACRO  PARAM   TYPE,OFFSET,WIDTH=0,MIN=0,MAX=-1,REQUIRED,INVALID
          0000                          ; NOTE - The REQUIRED field can only check 1 bit!
          0000  
          0000          $$$NUM = $$$NUM+1                     ; Count number of time executed
          0000          $$$TYP = NMA$C_'TYPE & PRM_M_TYPE     ; Isolate type
          0000          $$$OFF = OFFSET & OFF_M_VALUE         ; Isolate offset only
          0000  
          0000          .IIF NOT_BLANK  <MIN>,       $$$TYP = $$$TYP ! PRM_M_MIN
          0000          .IIF NOT_BLANK  <MAX>,       $$$TYP = $$$TYP ! PRM_M_MAX
          0000          .IIF NOT_BLANK  <REQUIRED>, $$$TYP = $$$TYP ! PRM_M_REQUIRE
          0000          .IIF NOT_BLANK  <INVALID>,  $$$TYP = $$$TYP ! PRM_M_INVALID
          0000  
          0000                  .WORD   $$$TYP
          0000                  .WORD   $$$OFF ! <WIDTH @ OFF_V_WIDTH>
          0000                  .WORD   MIN
          0000                  .WORD   MAX
          0000                  .WORD   REQUIRED
          0000                  .WORD   INVALID
          0000  .ENDM   PARAM
          0000  
          0000  
          0000  .MACRO  COUNT   TYPE,OFFSET,WIDTH=32
          0000  
          0000          $$$NUM = $$$NUM+1                     ; Bump number of time executed
          0000          $$$OFF = OFF_M_VALUE & OFFSET         ; Isolate offset only
          0000          $$$TYP = PRM_M_TYPE & NMA$C_'TYPE     ; Isolate type
          0000  
          0000          .IIF IDN, WIDTH, 8,     $$$TYP = $$$TYP ! <1@NMA$V_CNT_WID>
          0000          .IIF IDN, WIDTH,16,     $$$TYP = $$$TYP ! <2@NMA$V_CNT_WID>
          0000          .IIF IDN, WIDTH,32,     $$$TYP = $$$TYP ! <3@NMA$V_CNT_WID>
          0000  
          0000  
          0000                  .WORD   $$$TYP ! NMA$M_CNT_COU
          0000                  .WORD   $$$OFF ! <WIDTH @ OFF_V_WIDTH>
          0000  .ENDM   COUNT
          0000  
          0000  .MACRO  START_TABLE     NAME            ; Start Table declaration
          0000                 $$$NUM = 0                         ; Init number of entries
          0000                 'NAME'_TABLE = .                   ; Define begining of table
          0000         .ENDM   START_TABLE
          0000
          0000         .MACRO  END_TABLE        NAME              ; Terminate Table declaration
          0000                 .WORD   0                          ; Create marker
          0000                 'NAME'_NUM = $$$NUM                ; Number of entries
          0000         .ENDM   END_TABLE
          0000
          0000         ;
          0000         ; Local symbols
          0000         ;
          0000
          0000         ;
          0000         ; $QIO parameter offsets
          0000         ;
00000000  0000         P1        = 0                              ; Parameter 1
00000004  0000         P2        = 1*4                            ; Parameter 2
          0000
          0000         ;
          0000         ; Other constants
          0000         ;
00000009  0000         RBFMIN            =   9                    ; Min size of CDB_B_RCV_CNT
0000001F  0000         RBFMAX            =  31                    ; Max size of CDB_B_RCV_CNT
00000006  0000         RBFTHR            =   6                    ; CND_B_RCV_FQ threshold.  Below this
          0000                                                    ; signal XM$M_STS_BUFFAIL in IOST2
00000010  0000         MAX_TRB           = 16                     ; Max tributaries on CI device
```

```
                0000      ;
                0000      ; Overlays of IRP
                0000      ;
                0000              ASSUME IRP$L_SEGVBN EQ IRP$Q_NT_PRVMSK+8
                0000
                0000      $DEFINI IRP
                0000
      00000040  0000              . = IRP$Q_NT_PRVMSK                 ; Overlay network priv mask
                0040      $DEF    IRP$B_INDEX      .BLKB   1          ; Vector index for CDB
                0041
      00000054  0041              . = IRP$L_EXTEND
                0054      $DEF    IRP$L_CDB        .BLKL   1
                0058
                0058      $DEFEND IRP                                 ; End of IRP overlays
                0000
                0000
                0000      ;
                0000      ; Definitions that follow the standard UCB fields
                0000      ;
                0000
                0000      $DEFINI UCB                                 ; Start of UCB definitions
                0000
      00000090  0000              . = UCB$C_LENGTH                    ; Position at end of UCB
                0090
                0090      $DEF    UCB$L_LIS_CDT    .BLKL   1          ; Addr of listening CDT
                0094      $DEF    UCB$L_TWIN_CDT   .BLKL   1          ; Addr of loopbacked accept CDT
                0098      $DEF    UCB$L_DGHDRSZ    .BLKL   1          ; Size of the SCS header for DG's
                009C      $DEF    UCB$W_DUMMY      .BLKW   1          ; Dummy location for unwanted param's
                009E      $DEF    UCB$B_CN_PORT    .BLKB   1          ; Our port number
                009F      $DEF    UCB$B_RCV_CNT    .BLKB   1          ; Number of receive buffers
                00A0      $DEF    UCB$L_VEC_CDB    .BLKL   MAX_TRB    ; CDB address vector
                00E0      $DEF    UCB$W_VEC_CHAN   .BLKW   MAX_TRB    ; User channel lookup vector
                0100
      00000100  0100              UCB$C_CN_LENGTH = <.+15>&-16        ; Size of UCB padded to a quadword
                0100
                0100              ;
                0100              ;  Define device status bits
                0100              ;
                0100              $VIELD  UCB,0,<-                    ; CNDRIVER UCB$W_DEVSTS bits
                0100                              <CN_INITED,,M>,-    ; Device init'ed
                0100                      >                           ;
                0100      $DEFEND UCB                                 ; End of UCB definitions
                0000
```

```
              0000     ;
              0000     ; CNDRIVER CDB definitions
              0000     ;
              0000    $DEFINI CDB
              0000
              0000    $DEF      CDB_Q_FORK      .BLKQ   1       ; Fork Queue Linkage
              0008    $DEF      CDB_W_SIZE      .BLKW   1       ; Structure size
              000A    $DEF      CDB_B_TYPE      .BLKB   1       ; Structure type
              000B    $DEF      CDB_B_FIPL      .BLKB   1       ; Fork IPL (not UCB FIPL)
   00000006   000C              CDB_C_FIPL = 6                  ; Must be less than SCS's IPL (8)
              000C    $DEF      CDB_L_FPC       .BLKL   1       ; Fork PC
              0010    $DEF      CDB_L_FR3       .BLKL   1       ; Fork R3
              0014    $DEF      CDB_L_FR4       .BLKL   1       ; Fork R4
              0018                                              ;
              0018    $DEF      CDB_Q_XMT_IRP   .BLKQ   1       ; Transmit IRP's awaiting completion
              0020    $DEF      CDB_Q_RCV_IRP   .BLKQ   1       ; Receive IRP's awaiting buffers
              0028    $DEF      CDB_Q_RCV_MSG   .BLKQ   1       ; Receive buffers containing messages
              0030                                              ;
              0030    $DEF      CDB_L_SETMODE   .BLKL   1       ; Ptr to IO$_SETMODE
              0034    $DEF      CDB_L_ABSTIME   .BLKL   1       ; Time last DISCONNECT was issued
              0038    $DEF      CDB_W_BUFSIZ    .BLKW   1       ; Buffer size
              003A    $DEF      CDB_W_STS       .BLKW   1       ; Circuit status
              003C    $DEF      CDB_B_RCV_CNT   .BLKB   1       ; Receive buffer count
              003D    $DEF      CDB_B_RCV_FQ    .BLKB   1       ; Receive buffers on free queue
              003E    $DEF      CDB_B_TRB_ADDR  .BLKB   1       ; Tributary address
              003F    $DEF      CDB_B_STA       .BLKB   1       ; Circuit state
              0040              ;
              0040              ;
              0040              ;    Circuit counters
              0040              ;
              0040              ;
              0040    $DEF      CDB_L_BRC       .BLKL   1       ; Receive byte count
              0044    $DEF      CDB_L_BSN       .BLKL   1       ; Transmit byte count
              0048    $DEF      CDB_L_DBR       .BLKL   1       ; Data buffers received
              004C    $DEF      CDB_L_DBS       .BLKL   1       ; Data buffers sent
              0050              ;
              0050    $DEF      CDB_L_UCB       .BLKL   1       ; Addr of UCB
              0054    $DEF      CDB_L_CDT       .BLKL   1       ; Ptr to CDT
              0058    $DEF      CDB_B_REMVER    .BLKB   1       ; Remote's protocol version
              0059    $DEF      CDB_B_REMSYS    .BLKB   1       ; Remote's operating system
   00000058   005A              CDB_W_REMPROT = CDB_B_REMVER    ; Label combining two fields above
              005A    $DEF      CDB_B_DUMMY     .BLKB   1       ; Dummy location for unwanted param's
              005B    $DEF      CDB_B_RSTCNT    .BLKB   1       ; Restart counter for slowing down
              005C                                              ; restart frequency
   00000060   005C              CDB_C_LENGTH = <.+15>&-16       ; Pad structure out to a quadword
              005C
              005C     ;
              005C     ; Define status bits used in CDB_W_STS and CDB_B_STA values
              005C     ;
              005C              _VIELD  CDB,0,<-                ; Tributary status bits for CDB_W_STS
              005C                              <RUN,,M>,-      ; Tributary is in RUN state
              005C                              <CONN,,M>,-     ; Call to CONNECT pending
              005C                              <ACPT,,M>,-     ; CALL to ACCEPT pending
              005C                              <DISC,,M>,-     ; Call to DISCONNECT or FORK pending
              005C                              <REJECT,,M>,-   ; Call to REJECT pending
              005C                  >                           ;
              005C
```

```
00000000  005C          CDB_C_IDLE = 0                  ; Tributary is idle
00000001  005C          CDB_C_OPEN = 1                  ; Tributary connection has been made
00000002  005C          CDB_C_CONN = 2                  ; Tributary has CONNECT pending
00000003  005C          CDB_C_LSTN = 3                  ; Tributary is listening for connect
00000004  005C          CDB_C_ACPT = 4                  ; Tributary has ACCEPT pending
          005C
          005C     $DEFEND CDB
          0000
```

```
0000            .SBTTL  Standard tables
0000
0000    ;
0000    ; Driver prologue table
0000    ;
0000
0000            DPTAB   -                                       ; DPT-creation macro
0000                    END      = CN_END,-                     ; End of driver label
0000                    ADAPTER = NULL,-                        ; Adapter type
0000                    FLAGS    = DPT$M_SCS,-                  ; Driver requires SCS
0000                    UCBSIZE = UCB$C_CN_LENGTH,-             ; Length of UCB
0000                    NAME     = CNDRIVER,-                   ; Driver name
0000
0038            DPT_STORE INIT                                  ; Start of load
0038                                                            ; initialization table
0038            DPT_STORE UCB,UCB$B_FIPL,B,8                    ; Device fork IPL
003C            DPT_STORE UCB,UCB$B_DIPL,B,8                    ; Device interrupt IPL
0040            DPT_STORE ORB,ORB$B_FLAGS,B,-                   ; Protection block flags
0040                            <ORB$M_PROT_16>                 ; SOGW protection word
0044            DPT_STORE ORB,ORB$W_PROT,W,0                    ; default protection
0049            DPT_STORE ORB,ORB$L_OWNER,L,<^X010001>          ; [1,1] owns the device
0050            DPT_STORE UCB,UCB$L_DEVCHAR,L,-                 ; Device characteristics
0050                            <DEV$M_NET!-                    ;   e.g., network device
0050                             DEV$M_REC!-                    ;   record oriented
0050                             DEV$M_IDV!-                    ;   input device
0050                             DEV$M_ODV-                     ;   output device
0050                            >
0057            DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_SCOM         ; Sample device class
005B            DPT_STORE UCB,UCB$W_DEVBUFSIZ,@W,-              ; Default buffer size
005B                             SCS$GW_MAXDG
0062
0062            DPT_STORE REINIT                                ; Start of reload
0062                                                            ; initialization table
0062            DPT_STORE DDB,DDB$L_DDT,D,CN$DDT                ; Address of DDT
0067            DPT_STORE CRB,-                                 ; Address of device
0067                    CRB$L_INTD+VEC$L_UNITINIT,-             ; unit initialization
0067                    D,UNIT_INIT                             ; routine
006C
006C            DPT_STORE END                                   ; End of initialization
0000                                                            ; tables
```

```
0000    ;
0000    ; Driver dispatch table
0000    ;
0000            DDTAB   -                                       ; DDT-creation macro
0000                    DEVNAM  = CN,-                          ; Name of device
0000                    FUNCTB  = CN_FUNCTABLE,-                ; FDT address
0000                    CANCEL  = CANCEL,-                      ; Cancel I/O routine
0000                    ALTSTART= ALT_START                     ; Alternate start I/O
0038    ;
0038    ; Function dispatch table
0038    ;
0038    CN_FUNCTABLE:                                           ; FDT for driver
0038            FUNCTAB ,-                                      ; Valid I/O functions
0038                            <READLBLK,-                     ; Read logical
0038                             WRITELBLK,-                    ; Write logical
0038                             SETMODE,-                      ; Set device mode
0038                             SENSEMODE,-                    ; Sense mode
0038                             SETCHAR -                      ; Set device chars.
0038                            >
0040            FUNCTAB ,-                                      ; Buffered functions:
0040                            <READLBLK,-                     ; Read logical
0040                             WRITELBLK,-                    ; Write logical
0040                             SETMODE,-                      ; Set device mode
0040                             SENSEMODE,-                    ; Sense mode
0040                             SETCHAR -                      ; Set device chars.
0040                            >
0048            FUNCTAB CLR_IRP,-                               ; Init IRP fields
0048                            <READLBLK,-                     ; Read logical
0048                             WRITELBLK,-                    ; Write logical
0048                             SETMODE,-                      ; Set device mode
0048                             SENSEMODE,-                    ; Sense mode
0048                             SETCHAR -                      ; Set device chars.
0048                            >
0054            FUNCTAB RCV_FDT,-                               ; FDT read routine for
0054                            <READLBLK,-                     ; read logical,
0054                            >
0060            FUNCTAB XMT_FDT,-                               ; FDT write routine for
0060                            <WRITELBLK,-                    ; write logical,
0060                            >
006C            FUNCTAB SETMODE_FDT,-                           ; FDT set mode routine
006C                            <SETMODE,-                      ; set mode
006C                             SETCHAR -                      ; set characteristics
006C                            >
0078            FUNCTAB SENSEMODE_FDT,-                         ; FDT sense mode routine
0078                            <SENSEMODE>                     ; for sensemode
```

```
0084                 .SBTTL  P2 buffer verification tables
0084
0084    ;
0084    ; Define CDB parameters
0084    ;
0084    START_TABLE TRIB_PRM      ; Start of tributary parameter table
0084
0084            PARAM PCCI_MST, OFFSET   = CDB_B_DUMMY,-           ; Trib maint state
0084                            WIDTH    = 0,-                     ; Dummy location
0084                            MIN      = NMA$C_STATE_ON,-
0084                            MAX      = NMA$C_STATE_OFF,-
0084                            REQUIRED= 0,-
0084                            INVALID = CDB_M_RUN
0090
0090            PARAM PCCI_TRI, OFFSET   = CDB_B_TRB_ADDR,-        ; Trib address
0090                            WIDTH    = 8,-
0090                            MIN      = 0,-
0090                            MAX      = 15,-
0090                            REQUIRED= 0,-
0090                            INVALID = CDB_M_RUN
009C
009C            PARAM PCCI_MRB, OFFSET   = CDB_B_RCV_CNT,-         ; Trib max buf
009C                            WIDTH    = 8,-
009C                            MIN      = 0,-
009C                            MAX      = 255,-
009C                            REQUIRED= 0,-
009C                            INVALID = CDB_M_RUN
00A8
00A8    END_TABLE  TRIB_PRM       ; End of tributary paramerer table
00AA
00AA
00AA    ;
00AA    ; Define UCB parameters
00AA    ;
00AA    START_TABLE LINE_PRM      ; Start of device parameter table
00AA
00AA            PARAM PCLI_DUP, OFFSET   = UCB$W_DUMMY,-           ; Duplex
00AA                            WIDTH    = 0,-                     ; Dummy location
00AA                            MIN      = NMA$C_DPX_FUL,-
00AA                            MAX      = NMA$C_DPX_HAL,-
00AA                            REQUIRED= 0,-
00AA                            INVALID = UCB$M_CN_INITED
00B6
00B6            PARAM PCLI_CON, OFFSET   = UCB$W_DUMMY,-           ; Controller mode
00B6                            WIDTH    = 0,-                     ; Dummy location
00B6                            MIN      = NMA$C_LINCN_NOR,-
00B6                            MAX      = NMA$C_LINCN_LOO,-
00B6                            REQUIRED= 0,-
00B6                            INVALID = UCB$M_CN_INITED
00C2
00C2            PARAM PCLI_BUS, OFFSET   = UCB$W_DEVBUFSIZ,-       ; Block size
00C2                            WIDTH    = 16,-
00C2                            MIN      = 32,-
00C2                            MAX      = 948,-
00C2                            REQUIRED= 0,-
00C2                            INVALID = UCB$M_CN_INITED
00CE
```

```
                                      00CE              PARAM PCLI_BFN, OFFSET   = UCB$B_RCV_CNT,-       ; Maximum receive buffers
                                      00CE                              WIDTH    = 8,-
                                      00CE                              MIN      = 1,-
                                      00CE                              MAX      = 255,-
                                      00CE                              REQUIRED= 0,-
                                      00CE                              INVALID = UCB$M_CN_INITED
                                      00DA
                                      00DA      END_TABLE  LINE_PRM      ; End of device parameter tables
                                      00DC
                                      00DC      ;
                                      00DC      ; Tributary counter type codes
                                      00DC      ;
                                      00DC      START_TABLE  TRIB_CNT                ; Start of Tributary COUNTER table
                                      00DC
                                      00DC              COUNT    CTCIR_BRC, WIDTH=32, OFFSET=CDB_L_BRC    ; Bytes received
                                      00E0              COUNT    CTCIR_BSN, WIDTH=32, OFFSET=CDB_L_BSN    ; Bytes sent
                                      00E4              COUNT    CTCIR_DBR, WIDTH=32, OFFSET=CDB_L_DBR    ; Data blocks received
                                      00E8              COUNT    CTCIR_DBS, WIDTH=32, OFFSET=CDB_L_DBS    ; Data blocks sent
                                      00EC
                                      00EC      END_TABLE    TRIB_CNT                ; End of Tributary COUNTER table
                                      00EE
                                      00EE
                                      00EE      START_TABLE  LINE_CNT                ; Start of device COUNTER table
                                      00EE      END_TABLE    LINE_CNT                ; - null table
                                      00F0
                                      00F0      ;
                                      00F0      ;  Our SCS process name and connect data
                                      00F0      ;
                            00000006  00F0      PROC_C_NAM = 6                               ; How much of PROC_NAM must match
                                      00F0      PROC_NAM:
 45 53 41 48 50 24 54 45 4E 43 45 44  00F0              .ASCII  'DECNET$PHASE_III'           ; How SCS knows us -- 16 characters long
                         49 49 49 5F  00FC
                                      0100      CONN_DATA:
                                  01  0100              .BYTE   1                            ; Protocol version
                                  00  0101              .BYTE   0                            ; Operating system (VMS) id
00'00'00'00'00'00'00'00'00'00'00'00'  0102              .BYTE   0[14]                        ; Remaining fields must be zero
                              00'00'  010E
                                      0110
                            00000000  0110      OLD_C_PROT = 0                               ; Use for original protocol
                                      0110
```

```
                     0110              .SBTTL  UNIT_INIT,  Unit initialization routine
                     0110
                     0110      ;++
                     0110      ; UNIT_INIT - Readies unit for I/O operations
                     0110      ;
                     0110      ;
                     0110      ; The operating system calls this routine after calling the
                     0110      ; controller initialization routine:
                     0110      ;
                     0110      ;         - at system startup
                     0110      ;         - during driver loading
                     0110      ;         - during recovery from a power failure
                     0110      ;
                     0110      ; The unit is put online.
                     0110      ;
                     0110      ; Inputs:         R5 = UCB address
                     0110      ;
                     0110      ; Outputs:        All registers are preserved
                     0110      ;
                     0110      ;
                     0110      ;--
                     0110
                     0110      UNIT_INIT:                                      ; Initialize unit
 64 A5   10      A8  0110              BISW    #UCB$M_ONLINE,UCB$W_STS(R5)     ; Set unit online
                 05  0114              RSB                                     ; Return
                     0115
                 0115           .SBTTL  CLR_IRP - Initialize IRP fields
                 0115
                 0115   ;++
                 0115   ; CLR_IRP  - Initialize IRP fields
                 0115   ;
                 0115   ;
                 0115   ; Selected IRP fields are initialized.  The function code with modifiers
                 0115   ; is setup.
                 0115   ;
                 0115   ; Inputs:         R3   IRP address
                 0115   ;
                 0115   ; Outputs:        All other registers are preserved.
                 0115   ;
                 0115   ;                 IPL may be FIPL or ASTDEL
                 0115   ;
                 0115   ;--
                 0115   CLR_IRP:                                        ; Initialize IRP fields
  38 A3     7C   0115           CLRQ    IRP$L_IOST1(R3)                 ; Clear IOSB image
  2C A3     D4   0118           CLRL    IRP$L_SVAPTE(R3)                ; Init buffer pointer
  30 A3     B4   011B           CLRW    IRP$W_BOFF(R3)                  ; No quota to return yet at I/O post
  54 A3     D4   011E           CLRL    IRP$L_CDB(R3)                   ; No CDB yet
  40 A3     7C   0121           CLRQ    IRP$B_INDEX(R3)                 ; No trib i.d. yet
            05   0124           RSB                                     ; Done
                 0125
```

```
                             0125          .SBTTL  XMT_FDT,  Transmit I/O Operation FDT Routine
                             0125
                             0125  ;++
                             0125  ; XMT_FDT - Transmit I/O Operation FDT Routine
                             0125  ;
                             0125  ;
                             0125  ; This routine is called by the SYS$QIO system service to dispatch a
                             0125  ; WRITE I/O request.  The buffer is validated for access and copied to a
                             0125  ; system buffer.
                             0125  ;
                             0125  ; The QIO parameters used for WRITEs are:
                             0125  ;
                             0125  ;         P1 = address of the buffer
                             0125  ;         P2 = size of the buffer
                             0125  ;
                             0125  ; Inputs:                R3  - IRP address (I/O request packet)
                             0125  ;                        R4  - PCB address (process control block)
                             0125  ;                        R5  - UCB address (unit control block)
                             0125  ;                        R6  - CCB address (channel control block)
                             0125  ;                        R7  - bit number of the I/O function code
                             0125  ;
                             0125  ;                        IPL = ASTDEL (2)
                             0125  ;
                             0125  ; Outputs:               R0 = status of transmit request initiation
                             0125  ;
                             0125  ;                        R1,R2 are clobbered, all others are preserved.
                             0125  ;--
                             0125  XMT_FDT:                                                        ; Transmit FDT routine
                    69   10  0125          BSBB    XMT_RCV_FDT_CO                                  ; Get user buffer
                             0127                                                                  ; - no return on error
           00000000'GF   16  0127          JSB     G^EXE$WRITECHK                                  ; Check buffer access
                             012D                                                                  ; (no return means no access)
                             012D  GET_BUF:                                                        ; Get buffer
                    3E   BB  012D          PUSHR   #^M<R1,R2,R3,R4,R5>                             ; Save registers
                             012F
           00000000'GF   16  012F          JSB     G^EXE$BUFFRQUOTA                                ; Check if process has sufficient qu
                 49 50   E9  0135          BLBC    R0,20$                                          ; If LBC quota check failure
      51   0000004C 8F   C0  0138          ADDL    #CXB$C_OVERHEAD,R1                              ; Add in overhead
           00000000'GF   16  013F          JSB     G^EXE$ALONONPAGED                               ; Allocate buffer for output
                 39 50   E9  0145          BLBC    R0,20$                                          ; If LBC allocation failure
08 A2 51   001B0000 8F   C1  0148          ADDL3   #<DYN$C_CXB@16>,R1,IRP$W_SIZE(R2)               ; Set the size
           62    48 A2   9E  0151          MOVAB   CXB$C_HEADER(R2),(R2)                           ; Store pointer to data area
                 51 6E   D0  0155          MOVL    (SP),R1                                         ; Get back message size
           04 AE    52   D0  0158          MOVL    R2,4(SP)                                        ; Save buffer address
           53    08 AE   D0  015C          MOVL    8(SP),R3                                        ; Retrieve address of IRP
        50    0080 C4    D0  0160          MOVL    PCB$L_JIB(R4),R0                                ; Get JIB address
           20 A0    51   A2  0165          SUBW    R1,JIB$L_BYTCNT(R0)                             ; Adjust buffered I/O quota
           2C A3    52   D0  0169          MOVL    R2,IRP$L_SVAPTE(R3)                             ; Setup buffer pointer
           30 A3    51   B0  016D          MOVW    R1,IRP$W_BOFF(R3)                               ; Set number of bytes charged to quo
                    0B   13  0171          BEQL    10$                                             ; If EQL then none
        06 2A A3    01   E0  0173          BBS     #IRP$V_FUNC,IRP$W_STS(R3),10$                   ; If BS then "read" function
   00 B2   3C B3    51   28  0178          MOVC3   R1,@IRP$L_IOST2(R3),@(R2)                       ; Move data
              50    01   D0  017E  10$:    MOVL    #1,R0                                           ; Indicate success
                             0181                                                                  ;
                    3E   BA  0181  20$:    POPR    #^M<R1,R2,R3,R4,R5>                             ; Restore registers
                         05  0183          RSB                                                     ; Return to co-routine with
                         0184                   .SBTTL  RCV_FDT,  Read I/O Operation FDT Routine
                         0184
                         0184     ;++
                         0184     ; RCV_FDT - Read I/O Operation FDT Routine
                         0184     ;
                         0184     ;
                         0184     ; This routine is called by the SYS$QIO system service to dispatch a
                         0184     ; READ I/O request.
                         0184     ;
                         0184     ; The QIO parameters for READs are:
                         0184     ;
                         0184     ;         P1 = address of the buffer
                         0184     ;         P2 = size of the buffer
                         0184     ;         All other parameters are unused.
                         0184     ;
                         0184     ; The specified buffer is checked for accessibility. The buffer address and
                         0184     ; count are saved in the packet. Then IPL is raised to device fork IPL and if
                         0184     ; a message is available the operation is complete. Otherwise the packet is
                         0184     ; queued onto the waiting receive list of the CDB.
                         0184     ;
                         0184     ;
                         0184     ; Inputs:           R3  - IRP address (I/O request packet)
                         0184     ;                   R4  - PCB address (process control block)
                         0184     ;                   R5  - UCB address (unit control block)
                         0184     ;                   R6  - CCB address (channel control block)
                         0184     ;                   R7  - bit number of the I/O function code
                         0184     ;
                         0184     ;                   IPL = ASTDEL (2)
                         0184     ;
                         0184     ; Outputs:          R0 = status of transmit request initiation
                         0184     ;
                         0184     ;                   R1,R2 are clobbered, all others are preserved.
                         0184     ;
                         0184     ;--
                         0184     RCV_FDT:                                                       ; Read FDT process routine
             0A    10    0184               BSBB    XMT_RCV_FDT_CO                                 ; Get user buffer
                         0186                                                                  ; - no return on error
    00000000'GF    16    0186               JSB     G^EXE$READCHK                                  ; Check accessibility
                         018C                                                                  ; (No return on no access)
          50   01  D0    018C               MOVL    #1,R0                                          ; Say "success"
                   05    018F               RSB                                                    ; Return status to co-routine
                         0190
```

```
                      0190
                      0190  XMT_RCV_FDT_CO:
      50    14    3C  0190          MOVZWL  S^#SS$_BADPARAM,R0              ; Assume bad parameters
51    04    AC    3C  0193          MOVZWL  P2(AP),R1                       ; Get buffer size
            12    13  0197          BEQL    10$                             ; If zero, abort I/O request
   42 A5    51    B1  0199          CMPW    R1,UCB$W_DEVBUFSIZ(R5)          ; Is buffer too big?
            0C    1A  019D          BGTRU   10$                             ; If GTRU yes, abort I/O request
      50    6C    D0  019F          MOVL    P1(AP),R0                       ; Get user buffer virt address
   3C A3    50    D0  01A2          MOVL    R0,IRP$L_IOST2(R3)              ; Save it for MOVC
            9E    16  01A6          JSB     @(SP)+                          ; Call back our caller
         03 50    E8  01A8          BLBS    R0,20$                          ; If LBS, continue
          010D    31  01AB  10$:    BRW     ABORT_REQ                       ; Abort the request
                      01AE  20$:    ;
                      01AE          ;
                      01AE          ; Okay so far.  Setup to return to EXE$QIORETURN -- which returns to
                      01AE          ; the user with SS$_NORMAL in R0.  This means that all subsequent
                      01AE          ; errors must be reported via the IOSB.
                      01AE          ;
                      01AE          ;
  00000000'GF     9F  01AE          PUSHAB  G^EXE$QIORETURN                 ; Setup return address on stack
                      01B4          SETIPL  UCB$B_FIPL(R5)                  ; Raise IPL to fork level
                      01B8          ;                                       ;  to lock the data base
                      01B8          ; Fall thru to ALT_START
                      01B8          ;
```

```
                     01B8          .SBTTL  ALT_START,  Alternate Start I/O Routine
                     01B8
                     01B8  ;++
                     01B8  ; ALT_START - Alternate Start I/O Routine
                     01B8  ;
                     01B8  ;
                     01B8  ; This entry point is used to dispatch IO$_READLBLK and IO$_WRITELBLK requests.
                     01B8  ; The IRP is either built by our own FDT routines, or by some higher level
                     01B8  ; Executive agent (e.g. NETDRIVER).  All I/O status, including errors, must
                     01B8  ; be passed via IOPOST in the IOSB.
                     01B8  ;
                     01B8  ;
                     01B8  ; NOTE: The CHAN field of the IRP is sufficient to map to a CDB.
                     01B8  ;
                     01B8  ; Inputs:        R3 - IRP address
                     01B8  ;                R5 - UCB address
                     01B8  ;
                     01B8  ;                All pertinent fields of the IRP are assumed to be valid.
                     01B8  ;
                     01B8  ;                IPL = FIPL
                     01B8  ;
                     01B8  ; Outputs:       R0-R4   Garbage
                     01B8  ;
                     01B8  ;--
                     01B8  ALT_START:
   0210 8F    BB     01B8          PUSHR   #^M<R4,R9>                     ; Save reg
        05    10     01BC          BSBB    5$                             ; Process request
   0210 8F    BA     01BE          POPR    #^M<R4,R9>                     ; Restore regs
              05     01C2          RSB                                    ; Return to caller with garbage in R0
                     01C3                                                 ;
      079D    30     01C3  5$:     BSBW    XLATE                          ; Get CDB from IRP$W_CHAN
     5D 50    E9     01C6          BLBC    R0,ABORT_START                 ; If LBC then error
                     01C9          ASSUME  CDB_V_RUN  EQ  0               ;
  59 3A A9    E9     01C9          BLBC    CDB_W_STS(R9),ABORT_START; If LBC then not in RUN state
        01    E0     01CD          BBS     #IRP$V_FUNC,-                  ;
  57 2A A3           01CF                  IRP$W_STS(R3),RCV_START ; If BS then IO$_READ else IO$_WRITE
                     01D2          ;
                     01D2          ;  Fail thru to XMT_START
                     01D2          ;
```

```
                       01D2             .SBTTL  XMT_START,  Start Transmit Operation
                       01D2
                       01D2  ;++
                       01D2  ; XMT_START - Start Transmit Operation
                       01D2  ;
                       01D2  ;
                       01D2  ; This routine is called to start a transmit operation.  The tributary is
                       01D2  ; known to be up and running at this point.  All status must be returned via
                       01D2  ; the IOSB.
                       01D2  ;
                       01D2  ;
                       01D2  ; Inputs:         R3 = IRP address
                       01D2  ;                 R5 = UCB address
                       01D2  ;                 R9 = CDB address
                       01D2  ;
                       01D2  ;                 IPL = FIPL
                       01D2  ;
                       01D2  ; Outputs:        R0 = status of transmit request
                       01D2  ;
                       01D2  ;                 R5-R7 are preserved.
                       01D2  ;
                       01D2  ;--
                       01D2  XMT_START:
 51    32 A3     3C    01D2             MOVZWL  IRP$L_BCNT(R3),R1                       ; Pick up length
 50    2C A3     D0    01D6             MOVL    IRP$L_SVAPTE(R3),R0                     ; Pick up head of buffer
    52    60     D0    01DA             MOVL    (R0),R2                                 ; Get beginning of user message
                       01DD  10$:       ;
                       01DD             ;   Add CI padding to keep beginning quadword aligned
                       01DD             ;
    52    07     93    01DD             BITB    #^X<07>,R2                              ; Need padding ?
          07     13    01E0             BEQL    20$                                     ; If EQL no
    72    01     8E    01E2             MNEGB   #1,-(R2)                                ; Pad
          51     D6    01E5             INCL    R1                                      ; Adjust byte count
          F4     11    01E7             BRB     10$
                       01E9  20$:       ;
                       01E9             ;   Send it to SCS requesting that the datagram be returned when done.
                       01E9             ;
                       01E9             PUSHQ   R2                                      ; Save user msg & IRP addresses
  38 A9   51     B1    01EC             CMPW    R1,CDB_W_BUFSIZ(R9)                     ; Msg size within bounds?
          31     1A    01F0             BGTRU   60$                                     ; If GTRU then no
    52    20     C2    01F2             SUBL    #32,R2                                  ; Go to begining of PPD header
 54    52    50  C3    01F5             SUBL3   R0,R2,R4                                ; Get offset to top of buffer
    28    54     D1    01F9             CMPL    R4,#CXB$C_HEADER-32                     ; Is header big enough ?
          25     19    01FC             BLSS    60$                                     ; If LSS then header too small
  08 A2   54     AE    01FE             MNEGW   R4,8(R2)                                ; Neg. offset to top of buffer
  0A A2   3B     B0    0202             MOVW    #DYN$C_CIDG,10(R2)                      ; Sturcture type
 54    0084 C5   D0    0206             MOVL    UCB$L_PDT(R5),R4                        ; Recover the PDT
                       020B             SEND_DG_BUF_REG  #1,-                           ; Control returns immediately
                       020B                     CDT=CDB_L_CDT(R9),BUFFER=(SP)
       08 50     E9    0218             BLBC    R0,60$                                  ; If LBC, datagram not queued
                       021B             POPQ    R2                                      ; Restore IRP address
  1C B9   63     0E    021E             INSQUE  (R3),@CDB_Q_XMT_IRP+4(R9)               ; Queue IRP
                 05    0222  40$:       RSB                                             ; Return to await completion
                       0223
                       0223  60$:       POPQ    R2                                      ; Restore IRP address
                       0226
                       0226  ABORT_START:
```

```
                    06B6   31  0226   822          BRW     ABORT_IRP_POST                          ; Report SS$_ABORT via IOSB
                               0229   823
                    0229             .SBTTL  RCV_START,  Start Receive Operation
                    0229
                    0229   ;++
                    0229   ; RCV_START - Start Receive Operation
                    0229   ;
                    0229   ;
                    0229   ; This routine is called to start a receive operation.  The tributary is
                    0229   ; known to be up and running at this point.  If IRP$L_SVAPTE is none zero
                    0229   ; then it is assumed to be system buffer to be added to the receive free list.
                    0229   ; All status must be returned via the IOSB.
                    0229   ;
                    0229   ; Inputs:         R3 = IRP address
                    0229   ;                 R5 = UCB address
                    0229   ;                 R9 = CDB address
                    0229   ;
                    0229   ;                 IPL = FIPL
                    0229   ;
                    0229   ; Outputs:        R0 = return status of receive request
                    0229   ;
                    0229   ;--
                    0229   RCV_START:
  54     59    D0   0229           MOVL    R9,R4                           ; Transfer CDB pointer
       06D3    30   022C           BSBW    ADDRCVLIST                      ; Add IRP buffer to free list
                    022F           ;
                    022F           ;  Check to see if message is available
                    022F           ;
52    28 B9    0F   022F           REMQUE  @CDB_Q_RCV_MSG(R9),R2           ; Dequeue a received message
         03    1D   0233           BVS     100$                            ; Br if none
       0677    31   0235           BRW     FINISH_RCV_IO                   ; Complete the I/O request
                    0238   100$:   ;
                    0238           ;  No message available.  Queue IRP to await arrival of message.
                    0238           ;
24 B9    63    0E   0238           INSQUE  (R3),@CDB_Q_RCV_IRP+4(R9)       ; Queue IRP to await message
         05         023C           RSB                                     ; Return
                    023D
                    023D   861          .SBTTL  SETMODE_FDT,  Set mode I/O operation FDT routine
                    023D   862
                    023D   863  ;++
                    023D   864  ; SETMODE_FDT - Set mode I/O operation FDT routine
                    023D   865  ;
                    023D   866  ;
                    023D   867  ; Setup control parameters.  Optionally startup/shutdownt the device or one
                    023D   868  ; of the tributaries.  The subfunction modifiers are as follows:
                    023D   869  ;
                    023D   870  ;       IO$M_CTRL       - If set, request is for device.  Else, for tributary.
                    023D   871  ;       IO$M_STARTUP    - Start device or establish tributary connection.
                    023D   872  ;       IO$M_SHUTDOWN   - Shutdown device or disconnect tributary.
                    023D   873  ;
                    023D   874  ;
                    023D   875  ; The QIO parameter for SETMODE is:
                    023D   876  ;
                    023D   877  ;       P2 = Optional address of buffer descriptor for extended characteristics
                    023D   878  ;
                    023D   879  ;
                    023D   880  ; Inputs:       R3 = IRP address
                    023D   881  ;               R4 = PCB address
                    023D   882  ;               R5 = UCB address
                    023D   883  ;               R6 = CCB address
                    023D   884  ;               R7 = Function code
                    023D   885  ;               AP = address of first QIO parameter
                    023D   886  ;
                    023D   887  ; Outputs:      R0 = status of setmode request
                    023D   888  ;
                    023D   889  ;               R3-R5 are preserved.
                    023D   890  ;               R7-R9 = destroyed
                    023D   891  ;
                    023D   892  ;--
                    023D   893 SETMODE_FDT:                                     ; Setmode FDT processing
                    023D   894          ;
                    023D   895          ;  Copy the characteristics buffer, if any.  No return on error.
                    023D   896          ;  On return, there's a buffer attached to IRP$L_SVAPTE containing a
                    023D   897          ;  copy of the user buffer -- hence we cannot "abort" the QIO passed
                    023D   898          ;  this point but must return all errors via the IOSB.
                    023D   899          ;
                    023D   900          ;  Upon return, the IPL has been raised to FIPL
                    023D   901          ;
          029D   30 023D   902          BSBW    GET_CHAR_WBUF                    ; Get P2 characteristics buffer
                    0240   903                                                   ; - no return on error
     57  20 A3   3C 0240   904          MOVZWL  IRP$W_FUNC(R3),R7                ; Get full function code.
     03 57   09  E1 0244   905          BBC     S^#IO$V_CTRL,R7,10$              ; Br if not controller request
          0113   31 0248   906          BRW     SETMODE_CTRL                     ; Process controller request
                    024B   907 10$:     ;
                    024B   908          ;  Perform setmode request on a tributary
                    024B   909          ;
          0715   30 024B   910          BSBW    XLATE                            ; Get CDB address if any
     08 57   07  E1 024E   911          BBC     S^#IO$V_SHUTDOWN,R7,40$          ; Branch if not trib shutdown
                    0252   912          ;
                    0252   913          ;  Shutdown tributary modifier specified -- always successful.
                    0252   914          ;  Shutdown may complete ahead of other queued I/O for this tributary.
                    0252   915          ;
         6C 50   E9 0252   916          BLBC    R0,FINISH_SUC                    ; If LBC then no CDB
          0513   30 0255   917          BSBW    ZAP_CDB_R9                       ; Do the dirty work
```

```
            67  11  0258        BRB     FINISH_SUC                      ; Always return "success"
                    025A  40$:  ;
                    025A        ;  IO$M_STARTUP tributary modifier specified or no modifier.
                    025A        ;  Validate the P2 buffer and its contents.
                    025A        ;
 51    FE26 CF  9E  025A        MOVAB   TRIB_PRM_TABLE,R1               ; Set address of verification table
            52  D4  025F        CLRL    R2                              ; No status flags yet
         04 59  E8  0261        BLBS    R9,50$                          ; If LBS then no CDB
  52     3A A9  3C  0264        MOVZWL  CDB_W_STS(R9),R2                ; Get status flags
          072E  30  0268  50$:  BSBW    VALIDATE_P2                     ; Validate the P2 buffer
         67 50  E9  026B        BLBC    R0,FINISH_REQ                   ; If LBC, report error via IOSB
                    026E        ;
                    026E        ;  Check trib address.  If this is a trib address change for this
                    026E        ;  channel  (in which case unconditionally give up the old CDB even if
                    026E        ;  the QIO subsequently fails), or if there is no current CDB, then
                    026E        ;  attempt to bind this channel to the CDB for the new trib address.
                    026E        ;
 51    0474 8F  3C  026E        MOVZWL  #NMA$C_PCCI_TRI,R1              ; Get trib address param i.d.
          0796  30  0273        BSBW    UNPACK_P2_BUF                   ; From P2 buffer
         05 50  E8  0276        BLBS    R0,60$                          ; If LBS, trib was specified
         59 59  E8  0279        BLBS    R9,FINISH_REQ                   ; If LBS, no CDB - return R0,R1
            16  11  027C        BRB     80$                             ; No trib addr, use current CDB
         0E 59  E8  027E  60$:  BLBS    R9,70$                          ; If LBS, no CDB
  3E A9     52  91  0281        CMPB    R2,CDB_B_TRB_ADDR(R9)           ; Address being changed ?
            0D  13  0285        BEQL    80$                             ; If EQL no
   00E0 C542    B4  0287        CLRW    UCB$W_VEC_CHAN(R5)[R2]          ; Give-up previous CDB
          04DC  30  028C        BSBW    ZAP_CDB_R9                      ; Shut it down
            4A  10  028F  70$:  BSBB    NEW_TRIB                        ; Init/allocate CDB
         30 50  E9  0291        BLBC    R0,FINISH_ERR                   ; If LBC, report error
                    0294  80$:  ;
                    0294        ;  Tributary now exists change its characteristics and set them
                    0294        ;  if trib is established.
                    0294        ;
  29 57     06  E1  0294        BBC     S^#IO$V_STARTUP,R7,FINISH_SUC   ; Br if not startup request
 50    02C4 8F  3C  0298        MOVZWL  #SS$_DEVACTIVE,R0               ; Assume trib already active
 51      3A A9  3C  029D        MOVZWL  CDB_W_STS(R9),R1                ; Get current status
                    02A1        ASSUME  CDB_C_IDLE  EQ  0               ;
 51      3F A9  88  02A1        BISB    CDB_B_STA(R9),R1                ; OR in the state
 51      30 A9  C8  02A5        BISL    CDB_L_SETMODE(R9),R1            ; OR in pending SETMODE address
            19  12  02A9        BNEQ    FINISH_ERR                      ; If NEQ then can't do startup
  30 A9     53  D0  02AB        MOVL    R3,CDB_L_SETMODE(R9)            ; Save IRP address
          027C  30  02AF        BSBW    START_TRIB                      ; Startup the trib
                    02B2                                                ; Fall thru to QIORET
                    02B2
```

```
                        02B2          .SBTTL  Complete QIO request routines
                        02B2
                        02B2    ;+
                        02B2    ;  The following routines all exit the $QIO system service with status in R0.
                        02B2    ;  If an error is not being returned, further status will be eventually passed
                        02B2    ;  via the IOSB when the IRP undergoes post processing.
                        02B2    ;
                        02B2    ;
                        02B2    ;  Inputs:        R3  IRP address
                        02B2    ;                 R5  UCB address
                        02B2    ;
                        02B2    ;                 IPL may be FIPL or IPL$_ASTDEL
                        02B2    ;
                        02B2    ;-
      00000000'GF   17  02B2    QIORET: JMP     G^EXE$QIORETURN                 ; Return success in R0 to user
                        02B8    ABORT_REQ1:
         50    2C   D0  02B8            MOVL    S^#SS$_ABORT,R0                 ; Setup error status
                        02BB    ABORT_REQ:
      00000000'GF   17  02BB            JMP     G^EXE$ABORTIO                   ; Exit QIO service with error
                        02C1
                        02C1
                        02C1
                        02C1    ;+
                        02C1    ;  The following routines exit the $QIO system service with SS$_NORMAL and
                        02C1    ;  send the IRP back to IOPOST to return final status via the IOSB.
                        02C1    ;
                        02C1    ;  Inputs:        R3  IRP address
                        02C1    ;                 R5  UCB address
                        02C1    ;
                        02C1    ;                 IPL may be either FIPL or IPL$_ASTDEL
                        02C1    ;
                        02C1    ;-
                        02C1    FINISH_SUC:
         50    01   3C  02C1            MOVZWL  S^#SS$_NORMAL,R0                ; Set success
                        02C4    FINISH_ERR:
               51   D4  02C4            CLRL    R1                              ; Clear second IOSB longword
               09   E0  02C6            BBS     S^#IO$V_CTRL,-                  ; Skip for controllers
         0A 20 A3       02C8                    IRP$W_FUNC(R3),FINISH_REQ       ;
                        02CB            ASSUME  CDB_V_RUN  EQ  0
            07 50   E9  02CB            BLBC    R0,FINISH_REQ                   ; If LBC then circuit not up
 51   00002800 8F   D0  02CE            MOVL    #XM$M_STS_ACTIVE!-              ; Indicate circuit up
                        02D5                     XM$M_STS_RUNNING,R1            ;
                        02D5    FINISH_REQ:
      00000000'GF   17  02D5            JMP     G^EXE$FINISHIO                  ; Complete the I/O
                        02DB
```

```
                         02DB              .SBTTL  NEW_TRIB  - Allocate and init new CDB
                         02DB
                         02DB      ;+
                         02DB      ;  NEW_TRIB      - allocate and init new CDB
                         02DB      ;
                         02DB      ;
                         02DB      ; A CDB is allocated and initialized and stored in the UCB CDB vector.
                         02DB      ; The address is also stored in IRP$L_CDB.
                         02DB      ;
                         02DB      ;
                         02DB      ; Inputs:        R2  Trib address
                         02DB      ;                R3  IRP address
                         02DB      ;                R5  UCB address
                         02DB      ;
                         02DB      ; Outputs:       R9  CDB address
                         02DB      ;
                         02DB      ;                R0-R2  are clobbered.  All other registers are preserved.
                         02DB      ;
                         02DB      ;-
                         02DB      NEW_TRIB:
     40 A3    52    90   02DB              MOVB    R2,IRP$B_INDEX(R3)                 ; Set the trib number
  50   0641 8F    3C   02DF              MOVZWL  #SS$_DEVALRALLOC,R0                ; Assume error
     00E0 C542    B5   02E4              TSTW    UCB$Q_VEC_CHAN(R5)[R2]             ; Claimed by another channel?
             1E    12   02E9              BNEQ    40$                                ; If NEQ yes, report error
  59   00A0 C542    D0   02EB              MOVL    UCB$L_VEC_CDB(R5)[R2],R9           ; Get associated CDB
             0F    12   02F1              BNEQ    30$                                ; If NEQ, CDB exits - claim it
             15    10   02F3              BSBB    NEW_CDB                            ; Create a new CDB
          11 50    E9   02F5              BLBC    R0,40$                             ; If LBC then error
     52   40 A3    9A   02F8              MOVZBL  IRP$B_INDEX(R3),R2                 ; Restore trib address
00A0 C542    59    D0   02FC              MOVL    R9,UCB$L_VEC_CDB(R5)[R2]           ; Store CDB address in UCB
          28 A3    B0   0302      30$:    MOVW    IRP$W_CHAN(R3),-                   ; Save channel index in UCB
     00E0 C542          0305                      UCB$W_VEC_CHAN(R5)[R2]             ;
                   05   0309      40$:    RSB                                        ; Done
                        030A
                        030A
                        030A      NEW_CDB:                                           ; Create new CDB
  51   0060 8F    3C   030A              MOVZWL  #CDB_C_LENGTH,R1                   ; Get size of CDB
             53    DD   030F              PUSHL   R3                                 ; Save reg
    00000000'GF    16   0311              JSB     G^EXE$ALONONPAGED                  ; Allocate the CDB
             53  8ED0   0317              POPL    R3                                 ; Restore reg
          40 50    E9   031A              BLBC    R0,100$                            ; Br if error
                        031D              ;
                        031D              ;  Initialize CDB
                        031D              ;
       59    52    D0   031D              MOVL    R2,R9                              ; Copy CDB address
    54 A3    52    D0   0320              MOVL    R2,IRP$L_CDB(R3)                   ; Save it in IRP
    52   08 A2    9E   0324              MOVAB   CDB_W_SIZE(R2),R2                  ; Setup ptr to init CDB
                        0328
                        0328              ASSUME  CDB_B_TYPE      EQ  2+CDB_W_SIZE
                        0328              ASSUME  CDB_B_FIPL      EQ  1+CDB_B_TYPE
                        0328              ASSUME  CDB_L_FPC       EQ  1+CDB_B_FIPL
                        0328              ASSUME  CDB_L_FR3       EQ  4+CDB_L_FPC
                        0328              ASSUME  CDB_L_FR4       EQ  4+CDB_L_FR3
                        0328
       82    51    B0   0328              MOVW    R1,(R2)+                           ; CDB_W_SIZE
  82   0617 8F    B0   032B              MOVW    #<CDB_C_FIPL@8>+DYN$C_NET,(R2)+    ; CDB_B_TYPE and CDB_B_FIPL
       52    0C    C0   0330              ADDL    #3*4,R2                            ; Advance passed CDB_L_FR4
```

```
                     0333
                     0333            ASSUME  CDB_Q_XMT_IRP     EQ  4+CDB_L_FR4
                     0333            ASSUME  CDB_Q_RCV_IRP     EQ  8+CDB_Q_XMT_IRP
                     0333            ASSUME  CDB_Q_RCV_MSG     EQ  8+CDB_Q_RCV_IRP
                     0333
        51    03  D0 0333            MOVL    #3,R1                                  ; Set number of queue heads
        62    62  DE 0336    20$:    MOVAL   (R2),(R2)                              ; Init forward link pointer
        82    82  DE 0339            MOVAL   (R2)+,(R2)+                            ; Init backward link pointer
           F7 51  F5 033C            SOBGTR  R1,20$                                 ; Loop if more queues
                     033F
                     033F            ASSUME  CDB_L_SETMODE     EQ  8+CDB_Q_RCV_MSG
                     033F            ASSUME  CDB_L_ABSTIME     EQ  4+CDB_L_SETMODE
                     033F            ASSUME  CDB_W_BUFSIZ      EQ  4+CDB_L_ABSTIME
                     033F            ASSUME  CDB_W_STS         EQ  2+CDB_W_BUFSIZ
                     033F
              82  7C 033F            CLRQ    (R2)+                                  ; Init CDB_L_SETMODE,ABSTIME
     82    42 A5  3C 0341            MOVZWL  UCB$W_DEVBUFSIZ(R5),(R2)+              ; CDB_W_BUFSIZ and CDB_W_STS
                     0345
                     0345            ASSUME  CDB_B_RCV_CNT     EQ  2+CDB_W_STS
                     0345            ASSUME  CDB_B_RCV_FQ      EQ  1+CDB_B_RCV_CNT
                     0345            ASSUME  CDB_B_TRB_ADDR    EQ  1+CDB_B_RCV_FQ
                     0345            ASSUME  CDB_B_STA         EQ  1+CDB_B_TRB_ADDR
                     0345            ASSUME  CDB_C_IDLE        EQ  0
                     0345
  62    009F C5   90 0345            MOVB    UCB$B_RCV_CNT(R5),(R2)                 ; CDB_B_RCV_CNT (default)
          82 82   90 034A            MOVB    (R2)+,(R2)+                            ; CDB_B_RCV_FQ  (default)
     82    40 A3  9B 034D            MOVZBW  IRP$B_INDEX(R3),(R2)+                  ; CDB_B_TRB_ADDR, CDB_B_STA
                     0351
                     0351            ASSUME  CDB_L_BRC         EQ  1+CDB_B_STA
                     0351            ASSUME  CDB_L_BSN         EQ  4+CDB_L_BRC
                     0351            ASSUME  CDB_L_DBR         EQ  4+CDB_L_BSN
                     0351            ASSUME  CDB_L_DBS         EQ  4+CDB_L_DBR
                     0351
              82  7C 0351            CLRQ    (R2)+                                  ; CDB_L_BRC and CDB_L_BSN
              82  7C 0353            CLRQ    (R2)+                                  ; CDB_L_DBR and CDB_L_DBS
                     0355                                                           ;
                     0355            ASSUME  CDB_L_UCB         EQ  4+CDB_L_DBS
                     0355            ASSUME  CDB_L_CDT         EQ  4+CDB_L_UCB
                     0355
        82    55  D0 0355            MOVL    R5,(R2)+                               ; CDB_L_UCB
              82  D4 0358            CLRL    (R2)+                                  ; CDB_L_CDT
        50    01  D0 035A            MOVL    #1,R0                                  ; Indicate success
                  05 035D    100$:   RSB                                            ; Done
                     035E
                     035E
```

```
                    035E                 .SBTTL  SETMODE_CTRL,  Perform setmode FDT operation on controller
                    035E
                    035E     ;++
                    035E     ; SETMODE_CTRL - Perform setmode FDT operation on controller
                    035E     ;
                    035E     ;
                    035E     ; This routine performs the SETMODE FDT setup for the controller.
                    035E     ;
                    035E     ; Inputs:          R3 = IRP address
                    035E     ;                  R4 = PCB address
                    035E     ;                  R5 = UCB address
                    035E     ;                  R7 = IRP function word
                    035E     ;
                    035E     ; Outputs:         R0 = status of setmode request
                    035E     ;
                    035E     ;                  R3-R5 are preserved.
                    035E     ;
                    035E     ;--
                    035E     SETMODE_CTRL:                                           ; Perform setmode on controller
   05 57    07  E1  035E             BBC     S^#IO$V_SHUTDOWN,R7,10$                 ; Br if not shutdown request
                    0362             ;
                    0362             ;  Shutdown modifier specified
                    0362             ;
         03D0   30  0362             BSBW    CAN_DEV                                 ; Shutdown the device
           3F   11  0365             BRB     50$                                     ; Finish the QIO with success
                    0367     10$:    ;
                    0367             ;  Startup line modifier specified or no modifier
                    0367             ;
           00   E0  0367             BBS     #UCB$V_CN_INITED,-                      ; Br if controller up already
     31 68 A5       0369              UCB$W_DEVSTS(R5),40$
                    036C             ;
                    036C             ;  Validate P2
                    036C             ;
51    FD3A CF   9E  036C             MOVAB   LINE_PRM_TABLE,R1                       ; Address of verif table
        59 55   D0  0371             MOVL    R5,R9                                   ; Address of current param's
52     68 A5    3C  0374             MOVZWL  UCB$W_DEVSTS(R5),R2                     ; Status flags
         061E   30  0378             BSBW    VALIDATE_P2                             ; Validate P2 buffer
        2B 50   E9  037B             BLBC    R0,70$                                  ; If LBC, return R0,R1 in IOSB
                    037E             ;
                    037E             ;  Setup Maximum receive buffers
                    037E             ;
51    0451 8F   3C  037E             MOVZWL  #NMA$C_PCLI_BFN,R1                      ; Set to find MAX RCV
         0686   30  0383             BSBW    UNPACK_P2_BUF                           ; In P2 buffer
        05 50   E9  0386             BLBC    R0,30$                                  ; Br if not found
009F C5    52   90  0389             MOVB    R2,UCB$B_RCV_CNT(R5)                    ; Initialize number of RCV
                    038E     30$:    ;
                    038E             ;  Setup Blocksize
                    038E             ;
51    0AF1 8F   3C  038E             MOVZWL  #NMA$C_PCLI_BUS,R1                      ; Get buffer size
         0676   30  0393             BSBW    UNPACK_P2_BUF                           ; From P2 buffer
        04 50   E9  0396             BLBC    R0,40$                                  ; Br if not found
   42 A5   52   B0  0399             MOVW    R2,UCB$W_DEVBUFSIZ(R5)                  ;  and in UCB
                    039D     40$:    ;
                    039D             ;  Device initialized - then do a LISTEN if IO$V_STARTUP
                    039D             ;
   05 57    06  E1  039D             BBC     S^#IO$V_STARTUP,R7,50$                  ; Finish up if not starting
           0C   10  03A1             BSBB    LISTEN                                  ; Do a LISTEN
```

```
            06 50   E9  03A3          BLBC    R0,100$                         ; If LBC then failed
             FF18   31  03A6  50$:    BRW     FINISH_SUC                      ; Finish - SS$_NORMAL for IOSB
             FF29   31  03A9  70$:    BRW     FINISH_REQ                      ; Finish - R0,R1 for IOSB
             FF0C   31  03AC  100$:   BRW     ABORT_REQ                       ; Abort the I/O request
                        03AF
                        03AF  LISTEN: ;
                        03AF          ;  Do all the wonderful SCS magic needed to start up.  The buffer
                        03AF          ;  created on the stack is pointed to by R7 in case the CONFIG_SYS
                        03AF          ;  macro is updated someday to modify SP as it pushes arguements.
                        03AF          ;
                        03AF          ;  NOTE: The following code assumes that we have only 1 CI port on
                        03AF          ;        the current system!!!!!
                        03AF          ;
                        03AF
         00000070       03AF          SBO_LNG = SBO$C_LENGTH + 32             ; SBO length plus random amount
                        03AF                                                  ; of padding merely for merely
                        03AF                                                  ; hysterical purposes.
            59 53   D0  03AF          MOVL    R3,R9                           ; Save R3
   5E 00000070 8F   C2  03B2          SUBL    #SBO_LNG,SP                     ; Create buffer on stack
                        03B9                                                  ;
            57 5E   D0  03B9          MOVL    SP,R7                           ; Preserve value of buffer
                        03BC          CONFIG_SYS  G^SCS$GB_SYSTEMID,(R7)      ; Get our system block
            08 50   E9  03CC          BLBC    R0,200$                         ; If LBC, not ready yet
            3C A7   90  03CF          MOVB    SBO$B_RSTATION1(R7),-           ; Get our port number
             009E C5    03D2                  UCB$B_CN_PORT(R5)               ;
               07   18  03D5          BGEQ    210$                            ; If LSS then not ready yet
       50 0084 8F   3C  03D7  200$:   MOVZWL  #SS$_DEVOFFLINE,R0              ; Device offline error  (no PA)
               36   11  03DC          BRB     220$                            ; Exit
        56 14 A1    D0  03DE  210$:   MOVL    SB$L_PBCONNX(R1),R6             ; Get path block
        56 2C A6    D0  03E2          MOVL    PB$L_PDT(R6),R6                 ; Pick up PDT
      0084 C5 56    D0  03E6          MOVL    R6,UCB$L_PDT(R5)                ; Save in UCB
                        03EB          LISTEN  -                               ; Setup a LISTEN
                        03EB                  MSGADR = W^LIS_FORK,-
                        03EB                  ERRADR = W^LIS_ERR,-
                        03EB                  LPRNAM = PROC_NAM,-
                        03EB                  PRINFO = PROC_NAM
            0D 50   E9  0404          BLBC    R0, 220$                        ; If LBC then error
      0090 C5 53    D0  0407          MOVL    R3,UCB$L_LIS_CDT(R5)            ; Save listen CDT
        5C A3 55    D0  040C          MOVL    R5,CDT$L_AUXSTRUC(R3)           ; Set addr of UCB into CDT
        68 A5 01    A8  0410          BISW    #UCB$M_CN_INITED,UCB$W_DEVSTS(R5) ; Indicate device inited
                        0414                                                  ;
   5E 00000070 8F   C0  0414  220$:   ADDL    #SBO_LNG,SP                     ; Restore stack
            53 59   D0  041B          MOVL    R9,R3                           ; Restore IRP addr
                    05  041E          RSB
                        041F
                        041F  LIS_ERR:                                        ; Error on LISTEN CDT
                        041F          DISCONNECT                              ; Put it back to listen
                    05  0425          RSB                                     ; Leave
                        0426
```

```
                          0426                  .SBTTL  SENSEMODE_FDT,  Sense Mode I/O operation FDT routine
                          0426
                          0426          ;++
                          0426          ; SENSEMODE_FDT - Sense Mode FDT routine
                          0426          ;
                          0426          ;  This routine returns information to the caller about the configuration
                          0426          ;  and status of the CI device.  Depending on the function modifier,
                          0426          ;  either the device characteristics or error counters contents are returned.
                          0426          ;
                          0426          ;  The QIO parameters for SENSEMODE are:
                          0426          ;
                          0426          ;       P2 = optional address of buffer descriptor for extended characteristics
                          0426          ;
                          0426          ;
                          0426          ; Inputs:          R3 = IRP address
                          0426          ;                  R4 = PCB address
                          0426          ;                  R5 = UCB address
                          0426          ;                  R6 = CCB address
                          0426          ;                  R7 = Function code
                          0426          ;                  AP = Address of first function-dependent QIO parameter
                          0426          ;
                          0426          ; Outputs:         R0 = status return of sensemode request
                          0426          ;
                          0426          ;                  R3-R5 are preserved.
                          0426          ;
                          0426          ;--
                          0426          SENSE_TABLE:                                            ; Setup list of offset to
                  03A2    0426                  .WORD   SENSE_TABLE - TRIB_PRM_TABLE    ;   parameter tables with using
                  034A    0428                  .WORD   SENSE_TABLE - TRIB_CNT_TABLE    ;   the following 2 bit index:
                  037C    042A                  .WORD   SENSE_TABLE - LINE_PRM_TABLE    ;
                  0338    042C                  .WORD   SENSE_TABLE - LINE_CNT_TABLE    ;       bit 0 set => counters
                          042E                                                          ;       bit 1 set => non-trib
                          042E
                          042E          SENSEMODE_FDT:                                          ; Sensemode FDT I/O processing
              00000080    042E                  SENSE_C_BUF = 128
                          042E
                          042E                  ASSUME  TRIB_PRM_NUM*6  LE  SENSE_C_BUF ; Make sure buffer can hold all
                          042E                  ASSUME  LINE_PRM_NUM*6  LE  SENSE_C_BUF ; info for all cases
                          042E                  ASSUME  TRIB_CNT_NUM*6  LE  SENSE_C_BUF
                          042E                  ASSUME  LINE_CNT_NUM*6  LE  SENSE_C_BUF
                          042E                  ;
                          042E                  ;  Check user buffer.  Get system buffer.  Setup IRP
                          042E                  ;
  3A A3   0080 8F   B0    042E                  MOVW    #SENSE_C_BUF,IRP$L_IOST1+2(R3)  ; Setup buff size needed
             00A2   30    0434                  BSBW    GET_CHAR_RBUF                   ; Setup "read" buff for IOPOST
                          0437                                                          ; - no return on error
     57     20 A3   3C    0437                  MOVZWL  IRP$W_FUNC(R3),R7               ; Get full function code.
  04 A2     3C A3   D0    043B                  MOVL    IRP$L_IOST2(R3),4(R2)           ; Store user buffer virt addr
                          0440                                                          ; in standard place in buffer
        52     62   D0    0440                  MOVL    (R2),R2                         ; Get pointer to data area
                          0443                  ;
                          0443                  ;  Locate parameter/counter table
                          0443                  ;
        56     03   D0    0443                  MOVL    #3,R6                           ; Init SENSE_TABLE index
        58     02   D0    0446                  MOVL    #COUNT_C_ENTRY-2,R8             ; Bias COUNTER table entry size
     09 57     08   E0    0449                  BBS     #IO$V_RD_COUNT,R7,10$           ; If BS, "read counter" request
        56     97         044D                  DECB    R6                              ; Erase "read counter" bit
```

```
          58    0A   D0  044F             MOVL    #PARAM_C_ENTRY-2,R8                  ; Bias PARAM table entry size
       00 57    08   E5  0452             BBCC    #IO$V_RD_COUNT,R7,10$                ; Clear out garbage modifier
          59    55   D0  0456  10$:       MOVL    R5,R9                                ; If IO$V_CTRL, use UCB source
       09 57    09   E0  0459             BBS     #IO$V_CTRL,R7,20$                    ; If BS, not for a tributary
          56    02   8A  045D             BICB    #2,R6                                ; Erase "non-tributary" flag
              0500   30  0460             BSBW    XLATE                                ; Locate CDB, use CDB source
             6C 50   E9  0463             BLBC    R0,100$                              ; If LBC then CDB not found
    50    BC AF46    32  0466  20$:       CVTWL   SENSE_TABLE[R6],R0                   ; Get offset to parameter table
       56    B8 AF   9E  046B             MOVAB   SENSE_TABLE,R6                       ; Get address of base
       56     50     C2  046F             SUBL    R0,R6                                ; Calculate table address
                         0472  30$:       ;
                         0472             ;   Fill buffer with requested information
                         0472             ;
          51    86   B0  0472             MOVW    (R6)+,R1                             ; Get parameter i.d.
                2E   13  0475             BEQL    60$                                  ; If EQL, at end of table
       05 57    08   E0  0477             BBS     #IO$V_RD_COUNT,R7,40$                ; If BS then counter i.d.
    51    F000  8F   AA  047B             BICW    #^C<PRM_M_TYPE>,R1                   ; Else param i.d.,  clear junk
54   66   0A    00   EF  0480  40$:       EXTZV   #OFF_V_VALUE,#OFF_S_VALUE,(R6),R4     ; Get source offset
50   66   06    0A   EF  0485             EXTZV   #OFF_V_WIDTH,#OFF_S_WIDTH,(R6),R0     ; Get source width
                14   13  048A             BEQL    50$                                  ; If EQL, ignore this param
          54    59   C0  048C             ADDL    R9,R4                                ; Calculate source address
          82    51   B0  048F             MOVW    R1,(R2)+                             ; Enter parameter i.d.
82   64   50    00   EF  0492             EXTZV   #0,R0,(R4),(R2)+                     ; Enter parameter value
       05 57    0A   E1  0497             BBC     #IO$V_CLR_COUNT,R7,50$               ; If BC, don't clear source
64   50   00    00   F0  049B             INSV    #0,#0,R0,(R4)                        ; Clear counter
          56    58   C0  04A0  50$:       ADDL    R8,R6                                ; Advance to next entry
          CD    11       04A3             BRB     30$                                  ; Loop
                         04A5  60$:       ;
                         04A5             ;   Setup status and transfer size
                         04A5             ;
      52   2C B3     C2  04A5             SUBL    @IRP$L_SVAPTE(R3),R2                 ; Calculate bytes moved
 00000080 8F    52   D1  04A9             CMPL    R2,#SENSE_C_BUF                      ; Was our buffer large enough ?
                23   1A  04B0             BGTRU   200$                                 ; If GTRU no
      38 A3     01   B0  04B2             MOVW    #SS$_NORMAL,IRP$L_IOST1(R3)          ; Assume success
      52    32 A3    B1  04B6             CMPW    IRP$W_BCNT(R3),R2                    ; User buffer big enough ?
                0A   1E  04BA             BGEQU   80$                                  ; If GEQU then yes
  38 A3   0601 8F    B0  04BC             MOVW    #SS$_BUFFEROVF,IRP$L_IOST1(R3)       ; Show warning
      52    32 A3    B0  04C2             MOVW    IRP$W_BCNT(R3),R2                    ; Shrink xfer size
      3A A3     52   B0  04C6  80$:       MOVW    R2,IRP$L_IOST1+2(R3)                 ; Move xfer size to IOSB image
      32 A3     52   B0  04CA             MOVW    R2,IRP$W_BCNT(R3)                    ; Setup xfer size for IOPOST
      50    38 A3    D0  04CE             MOVL    IRP$L_IOST1(R3),R0                   ; Set length/status
          FDEF       31  04D2  100$:      BRW     FINISH_ERR                           ; Leave setting R0 in IOSB
                         04D5
                         04D5  200$:      BUG_CHECK  INCONSTATE,FATAL                  ; We've corrupted pool
```

```
                              04D9            .SBTTL    GET_CHAR_RBUF,  Get P2 characteristics buffer for read
                              04D9            .SBTTL    GET_CHAR_WBUF,  Get P2 characteristics buffer for write
                              04D9
                              04D9  ;++
                              04D9  ; GET_CHAR_RBUF - Get P2 characteristics buffer for read
                              04D9  ; GET_CHAR_WBUF - Get P2 characteristics buffer for write
                              04D9  ;
                              04D9  ; This routine saves the address of P2 buffer for later use by the driver.
                              04D9  ; The P2 buffer address is saved in IRP$L_IOST2 of the IRP, and the size
                              04D9  ; in IRP$L_BCNT.
                              04D9  ;
                              04D9  ; Inputs:           R3 = IRP address
                              04D9  ;                   R4 = PCB address
                              04D9  ;                   R5 = UCB address
                              04D9  ;
                              04D9  ; Outputs:          R0 = Garbage
                              04D9  ;                   R1 = User buffer size
                              04D9  ;                   R3-R5 are preserved.
                              04D9  ;
                              04D9  ;--
                              04D9  GET_CHAR_RBUF:                                              ; Get P2 char buffer for "read"
           2A A3    02    88  04D9            BISB      #IRP$M_FUNC,IRP$W_STS(R3)         ; Mark IRP for "read"
                              04DD  GET_CHAR_WBUF:                                              ; Get P2 char buffer for "write"
                    50    7C  04DD            CLRQ      R0                                ; Setup null user buffer
           52    04 AC    D0  04DF            MOVL      P2(AP),R2                         ; Get address of P2 desc
                    13    13  04E3            BEQL      10$                               ; If EQL, no P2 was specified
50   0B A3    02    00    EF  04E5            EXTZV     #0,#2,IRP$B_RMOD(R3),R0           ; Get access mode
                              04EB            IFNORD    #8,(R2),50$,MODE=R0               ; Br if no read access
              51    62    3C  04F1            MOVZWL    (R2),R1                           ; Get buffer length in bytes
           50    04 A2    D0  04F4            MOVL      4(R2),R0                          ; Get buffer address
           3C A3    50    D0  04F8  10$:      MOVL      R0,IRP$L_IOST2(R3)                ; Save it for later
                    51    B5  04FC            TSTW      R1                                ; Null user buffer ?
                    14    13  04FE            BEQL      30$                               ; If EQL yes, don't probe
                 14'AF    9F  0500            PUSHAB    B^30$                             ; Setup return address
        06 2A A3    01    E1  0503            BBC       #IRP$V_FUNC,IRP$W_STS(R3),20$     ; If BC then "write"
           00000000'GF    17  0508            JMP       G^EXE$READCHK                     ; Check user buffer, setup IRP
           00000000'GF    17  050E  20$:      JMP       G^EXE$WRITECHK                    ; Check user buffer, setup IRP
                              0514                                                          ; - no return on error
        04 2A A3    01    E1  0514  30$:      BBC       #IRP$V_FUNC,IRP$W_STS(R3),40$     ; If BC then "write"
           51    3A A3    3C  0519            MOVZWL    IRP$L_IOST1+2(R3),R1              ; Get required buffer size
                  FC0D    30  051D  40$:      BSBW      GET_BUF                           ; Get buffer
                              0520            SETIPL    UCB$B_FIPL(R5)                    ; Raise IPL
                 06 50    E8  0524            BLBS      R0,60$                            ; Okay if LBS
              50    0C    3C  0527  50$:      MOVZWL    #SS$_ACCVIO,R0                    ; Set error status
                  FD8E    31  052A            BRW       ABORT_REQ                         ; Abort the I/O request
                          05  052D  60$:      RSB
                              052E
                          052E              .SBTTL  START_TRIB,  Start tributary routine
                          052E
                          052E  ;++
                          052E  ; START_TRIB - Start tributary routine
                          052E  ;
                          052E  ;
                          052E  ; This routine is called when a tributary is to be established and started.
                          052E  ; The control parameters are initialized also.
                          052E  ;
                          052E  ;  Inputs:        R3 = IRP address
                          052E  ;                 R5 = UCB address
                          052E  ;                 R9 = CDB address
                          052E  ;
                          052E  ;                 IPL = FIPL.
                          052E  ;
                          052E  ;  Outputs:       R5 is preserved.
                          052E  ;
                          052E  ;
                          052E  ;--
                          052E  START_TRIB:                                                   ; Start tributary
                          052E              ;
                          052E              ; Setup number of receive buffers
                          052E              ;
  51      0479 8F     3C  052E              MOVZWL  #NMA$C_PCCI_MRB,R1                        ; Get param i.d.
             04D6     30  0533              BSBW    UNPACK_P2_BUF                             ; Get param value
            05 50     E8  0536              BLBS    R0,20$                                    ; If LBS then param was there
  52      009F C5     9A  0539              MOVZBL  UCB$B_RCV_CNT(R5),R2                      ; Else, get default
          09   52     91  053E  20$:        CMPB    R2,#RBFMIN                                ; Are there enough buffers to
               03     1A  0541              BGTRU   30$                                       ; reduce datagram loss ?
          52   09     D0  0543              MOVL    #RBFMIN,R2                                ; If not, do user a favor
          52   1F     D1  0546  30$:        CMPL    #RBFMAX,R2                                ; Too many ?
               03     1A  0549              BGTRU   40$                                       ; If GTRU then no
          52   1F     D0  054B              MOVL    #RBFMAX,R2                                ; Use safer minimum
       3C A9   52     90  054E  40$:        MOVB    R2,CDB_B_RCV_CNT(R9)                      ; Setup receive pool accounting
       3D A9   52     90  0552              MOVB    R2,CDB_B_RCV_FQ(R9)                       ; List starts out full
                          0556              ;
                          0556              ; Init CDB state.
                          0556              ;
48 A3  30414150 8F    D0  0556              MOVL    #^A/PAA0/,IRP$B_INDEX+8(R3)               ; Set to connect over local
                          055E                                                                ;  port PAA0
       3F A9   03     90  055E              MOVB    #CDB_C_LSTN,CDB_B_STA(R9)                 ; Assume "listen" state
         009E C5      91  0562              CMPB    UCB$B_CN_PORT(R5),-                       ; Compare our address to
            40 A3         0566                      IRP$B_INDEX(R3)                           ; remote's address
               22     1F  0568              BLSSU   100$                                      ; If LSSU, stay in "listen"
               04     1A  056A              BGTRU   50$                                       ; If GTRU, initiate connect
               52     D4  056C              CLRL    R2                                        ; Else we're talking to
                          056E                                                                ; ourselves -- zero rcv buffers
               04     11  056E              BRB     60$                                       ; CONNECT from "LSTN" state
       3F A9   02     90  0570  50$:        MOVB    #CDB_C_CONN,CDB_B_STA(R9)                 ; Else, go to "connect" state
    14 3A A9   01     E2  0574  60$:        BBSS    #CDB_V_CONN,CDB_W_STS(R9),200$            ; Indicate waiting return from
                          0579                                                                ; CONNECT
               55     DD  0579              PUSHL   R5                                        ; Save UCB address
          55   59     D0  057B              MOVL    R9,R5                                     ; Use CDB for CONNECT context
               11     10  057E              BSBB    CONN                                      ; Post connect request to SCS
               55   8ED0  0580              POPL    R5                                        ; Restore UCB address
    04 3A A9   01     E1  0583              BBC     #CDB_V_CONN,CDB_W_STS(R9),100$            ; If BC, completed synchronously
       54 A9   53     D0  0588              MOVL    R3,CDB_L_CDT(R9)                          ; Set pointer to open CDT
```

```
                      05  058C  100$:   RSB                                          ; Done
                          058D
                          058D  200$:   BUG_CHECK  INCONSTATE,FATAL                  ; Bug if already set
                          0591
     50      3C A9    9A  0591  CONN:   MOVZBL  CDB_B_RCV_CNT(R9),R0                 ; Pick up rcv buffer count
                          0595          CONNECT -                                    ; Request a CONNECT
                          0595                  MSGADR = W^MSG_FORK,-                ; Message address
                          0595                  DGADR  = W^DG_FORK,-                 ; Psuedo interrupt routine
                          0595                  ERRADR = W^CONN_ERR,-                ; Connect errors
                          0595                  RSYSID = 0,-                         ; No remote system specified
                          0595                  RSTADR = IRP$B_INDEX(R3),-           ; Virtual circ to connect over
                          0595                  RPRNAM = PROC_NAM,-                  ; To whom we will speak
                          0595                  LPRNAM = PROC_NAM,-                  ; Our name
                          0595                  INITCR = #1,-                        ; Allow for messages
                          0595                  INITDG = R2,-                        ; Number of receive buffers
                          0595                  CONDAT = CONN_DATA,-                 ; Connect data
                          0595                  AUXSTR = (R5)                        ; Auxiliary structure
                          05CB          ;
                          05CB          ;  Control returns to caller's caller - the JMP G^EXE$QIORETURN.
                          05CB          ;  When the connection completes, the following is called as a fork
                          05CB          ;  process NOT necessarily in the context of process.
                          05CB          ;
                          05CB          ;       R0  =    Status code
                          05CB          ;       R1  =    Reject reason if status = reject
                          05CB          ;       R2 -->   ACCEPT_REQ msg if status = success
                          05CB          ;       R3 -->   Connection CDT
                          05CB          ;       R4 -->   PDT
                          05CB          ;       R5 -->   CDB
                          05CB          ;
                          05CB          CLRBIT  #CDB_V_CONN,CDB_W_STS(R5)            ; No longer awaiting CONN return
             0E 50    E9  05D0          BLBC    R0,20$                               ; If LBS then error
              00F9    30  05D3          BSBW    CHECK_REMOTE                         ; Check remote's connect info
                03    12  05D6          BNEQ    10$                                  ; If NEQ, can't talk to remote
              00B8    31  05D8          BRW     CONN_FIN                             ; Else okay, complete setup
                          05DB  10$:    DISCONNECT                                   ; Break the connection
             54 A5    D4  05E1  20$:    CLRL    CDB_L_CDT(R5)                        ; Forget about CDT, if any
                          05E4
                          05E4  CONN_ABO:
                          05E4          ;
                          05E4          ;  CONNECT or ACCEPT failed.
                          05E4          ;
                          05E4          ;  If we were to return an error immediately every time, the higher
                          05E4          ;  level user (NETACP) would consume too much time trying to restart
                          05E4          ;  the circuit.  This is because the CI, unlike other devices, will
                          05E4          ;  return immediately if the partner is not ready on a CONNECT
                          05E4          ;  attempt.  For all other devices, the connect remains pending
                          05E4          ;  indefinitely.
                          05E4          ;
                          05E4          ;  Therefore, in order to save CPU cycles, simply return and allow the
                          05E4          ;  IO$M_STARTUP $QIO to hang indefinitely.  This forces NETACP to
                          05E4          ;  initiate the subsequent cleanup via a $CANCEL, $DASSGN, or
                          05E4          ;  IO$M_SHUTDOWN.  This should be fixed someday to have CNDRIVER retry
                          05E4          ;  every 5 seconds or so without reporting an error.
                          05E4          ;
                          05E4          ;
                          05E4          ;       *** NOTE:  This logic here has been retained in case it needs
                          05E4          ;                  to be reactivated someday.  However, it has been
```

```
                   05E4       ;                                       found that not returning an error immediately can
                   05E4       ;                                       cause some confusion since it can delay a circuit
                   05E4       ;                                       initialization for 3 minutes or so in some cases.
                   05E4       ;                                       In addition, the time spent by NETACP to continually
                   05E4       ;                                       reinitialize the circuit has been found to be small
                   05E4       ;                                       enough that it presents no real problem.
                   05E4       ;
                   05E4       ;
      3A A5    B5  05E4            TSTW    CDB_W_STS(R5)                      ; All quiet yet ?
         18    12  05E7            BNEQ    100$                               ; If NEQ, just wait
   50 50 A5    D0  05E9            MOVL    CDB_L_UCB(R5),R0                   ; Get UCB address
09 64 A0  05   E0  05ED            BBS     #UCB$V_POWER,UCB$W_STS(R0),50$     ; If BS, powerfial recovery
      5B A5    96  05F2            INCB    CDB_B_RSTCNT(R5)                   ; Another restart attempt
   5B A5  03   93  05F5            BITB    #3,CDB_B_RSTCNT(R5)                ; Is this the 4th phase ?
                   05F9       ;    BEQL    100$                               ; If EQL yes, wait.
               01  05F9            NOP
               01  05FA            NOP
      54  55   D0  05FB  50$:      MOVL    R5,R4                              ; Copy CDB address
        016D   30  05FE            BSBW    ZAP_CDB                            ; Report the error immediately
               05  0601  100$:     RSB                                        ; Wait the gio until contacted
                   0602                                                       ; by user via $CANCEL, etc
                   0602
```

```
                        0602             .SBTTL  LIS_FORK, Listen action routine
                        0602
                        0602    ;++
                        0602    ; LIS_FORK - Listen action routine
                        0602    ;
                        0602    ;
                        0602    ; This routine is entered as a fork process activated by the PADRIVER
                        0602    ; when some other process has sent a CONNECT to us.  We can then decide
                        0602    ; to ACCEPT or REJECT the connection.
                        0602    ;
                        0602    ; Inputs:        R2 -->  CONNECT_REQ message
                        0602    ;                R3 -->  Listening CDT
                        0602    ;                R4 -->  PDT
                        0602    ;
                        0602    ;                IPL = FIPL
                        0602    ;
                        0602    ;--
                        0602    LIS_FORK:
    55    5C A3    D0   0602             MOVL    CDT$L_AUXSTRUC(R3),R5               ; Pick up UCB from listen CDT
    51    20 A3    9A   0606             MOVZBL  CDT$B_RSTATION(R3),R1               ; Get other guy's port
55   00A0 C541     D0   060A             MOVL    UCB$L_VEC_CDB(R5)[R1],R5            ; Pick up the CDB
             0B    13   0610             BEQL    REJECT                              ; We don't have one, reject
03    3F A5        91   0612             CMPB    CDB_B_STA(R5),#CDB_C_LSTN           ; Are we listening on this trib
             05    12   0616             BNEQ    REJECT                              ; If NEQ no, reject connection
           00B4    30   0618             BSBW    CHECK_REMOTE                        ; Process connect data
             09    13   061B             BEQL    ACCEPT                              ; If NEQ then from DECnet SYSAP
                        061D    REJECT: ;
                        061D            ;   REJECT the connection.
                        061D            ;
             55    D4   061D             CLRL    R5                                  ; Forget about CDB
       50    01    D0   061F             MOVL    #SS$_NORMAL,R0                      ; Reject reason
                        0622             REJECT                                      ; Yes, reject him - return to
                        0625                                                         ; caller's caller.
                   05   0625             RSB                                         ; Return to SCS (nop)
                        0626
                        0626    ACCEPT: ;
                        0626            ;   ACCEPT the connection.
                        0626            ;
    50    3C A5    9A   0626             MOVZBL  CDB_B_RCV_CNT(R5),R0                ; Pick up rec buf count
   3F A5    04     90   062A             MOVB    #CDB_C_ACPT,CDB_B_STA(R5)           ; Change state to "accept"
                        062E             SETBIT  #CDB_V_ACPT,CDB_W_STS(R5)           ; Inidicate ACCEPT pending
                        0633             ACCEPT  -                                   ; ACCEPT the connection
                        0633                     MSGADR  = W^MSG_FORK,-              ; Message address
                        0633                     DGADR   = W^DG_FORK,-               ; Psuedo interrupt rtn
                        0633                     ERRADR  = W^CONN_ERR,-              ; Error address
                        0633                     INITCR  = #1,-                      ; Allow for messages
                        0633                     INITDG  = R0,-                      ; Receive buffers
                        0633                     CONDAT  = CONN_DATA,-               ; Accept data
                        0633                     AUXSTR  = (R5)                      ; Auxiliary strucure (CDB)
                        065C            ;
                        065C            ; Control returns to caller's caller if this request does not complete
                        065C            ; synchronously.  In that case, when the ACCEPT completes, the
                        065C            ; following is called as a fork process NOT necessarily in the context
                        065C            ; of user's process.
                        065C            ;
                        065C            ;        R0  =   Status
                        065C            ;        R1  =   ?
```

```
                             065C               ;       R2 -->  ACCEPT_RSP message
                             065C               ;       R3 -->  CDT
                             065C               ;       R4 -->  PDT
                             065C               ;       R5 -->  CDB
                             065C               ;
                             065C               CLRBIT  #CDB_V_ACPT,CDB_W_STS(R5)       ; ACCEPT no longer pending
                  13 50  E9  0661               BLBC    R0,50$                          ; If LBC then failed
         50       50 A5  D0  0664               MOVL    CDB_L_UCB(R5),R0                ; Get UCB
                  20 A3  91  0668               CMPB    CDT$B_RSTATION(R3),-            ; Are we talking to ourselves?
               009E C0       066B                       UCB$B_CN_PORT(R0)               ;
                     23  12  066E               BNEQ    CONN_FIN                        ; If NEQ no, complete setup
         0094 C0     53  D0  0670               MOVL    R3,UCB$L_TWIN_CDT(R0)           ; Else setup TWIN CDT
                     20  11  0675               BRB     CONN_FINT                       ; Finish processing without
                             0677                                                       ; storing CDT in CDB
                  53 52  D0  0677      50$:     MOVL    R2,R3                           ; Copy CDT to right register
         50     206C 8F  3C  067A               MOVZWL  #SS$_REMRSRC,R0                 ; REJECT reason
                             067F               SETBIT  #CDB_V_REJECT,CDB_W_STS(R5)     ; Set REJECT in progress
                             0684               REJECT                                  ; Must REJECT on ACCEPT failure
                             0687               CLRBIT  #CDB_V_REJECT,CDB_W_STS(R5)     ; Return is to caller's caller
                             068C                                                       ; - return here after a delay
                             068C                                                       ;   with R5 pointing to CDB
                   FF55  31  068C               BRW     CONN_ABO                        ; Go to common code.
                             068F
                             068F      100$:    BUG_CHECK  INCONSTATE,FATAL
```

```
                          0693
                          0693  ;
                          0693  ; CONNECT (or ACCEPT) succeded
                          0693  ;
                          0693  ; If no status bits are set then enter the "run" state and complete the
                          0693  ; pending IO$M_STARTUP request.  If any status bits are set  -- which can
                          0693  ; happen if we are talking to ourselves since we do both an ACCEPT and a
                          0693  ; CONNECT in that case -- then wait.
                          0693  ;
                          0693  CONN_FIN:
       54 A5     53     D0  0693          MOVL    R3,CDB_L_CDT(R5)                ; Set ptr to CDT
                          0697  CONN_FIN1:
          54     55     D0  0697          MOVL    R5,R4                           ; Copy CDT address
              5B A4     94  069A          CLRB    CDB_B_RSTCNT(R4)                ; Init failed restart counter
              34 A4     D4  069D          CLRL    CDB_L_ABSTIME(R4)               ; Don't inhibit DISCONNECT
       55     50 A4     D0  06A0          MOVL    CDB_L_UCB(R4),R5                ; Restore UCB pointer
       64 A5     20     AA  06A4          BICW    #UCB$M_POWER,UCB$W_STS(R5)      ; Any powerfail recovery is done
              3A A4     B5  06A8          TSTW    CDB_W_STS(R4)                   ; All quiet ?
                 1D     12  06AB          BNEQ    40$                             ; If NEQ no, wait
       53     30 A4     D0  06AD          MOVL    CDB_L_SETMODE(R4),R3            ; Get SETMODE IRP
                 18     13  06B1          BEQL    50$                             ; If EQL then none
              30 A4     D4  06B3          CLRL    CDB_L_SETMODE(R4)               ; Detach IRP from CDB
    10 20 A3     06     E1  06B6          BBC     #IO$V_STARTUP,IRP$W_FUNC(R3),50$; If BC then wrong IRP
       3F A4     01     90  06BB          MOVB    #CDB_C_OPEN,CDB_B_STA(R4)       ; Update current state
                            06BF          SETBIT  #CDB_V_RUN, CDB_W_STS(R4)       ; Allow data message traffic
          50     01     3C  06C4          MOVZWL  #SS$_NORMAL,R0                  ; Setup status
               021A     30  06C7          BSBW    SUC_TRB_IOPOST                  ; Post IRP with "success"
                        05  06CA  40$:    RSB
                            06CB
                            06CB  50$:    BUG_CHECK INCONSTATE,FATAL
                            06CF
                            06CF  CHECK_REMOTE:                                   ; Check remote connect data
                            06CF          ;
                            06CF          ;
                            06CF          ;   0-15(R2)  Contain our process name (who remote is connecting to)
                            06CF          ;  16-31(R2)  Contain remote's process name
                            06CF          ;  32-47(R2)  Contain connect data
                            06CF          ;
                 1D     BB  06CF          PUSHR   #^M<R0,R2,R3,R4>                ; Save some registers
                            06D1                                                  ;
          54     52     D0  06D1          MOVL    R2,R4                           ; Make stable msg pointer
       58 A5     00     B0  06D4          MOVW    #OLD_C_PROT,CDB_W_REMPROT(R5)   ; Assume remote is old protocol
20 A4   FA13 CF   06    29  06D8          CMPC3   #PROC_C_NAM,PROC_NAM,32(R4)     ; Check the connect data
                 05     13  06DF          BEQL    10$                             ; If EQL then old style
       58 A5     20 A4  B0  06E1          MOVW    32(R4),CDB_W_REMPROT(R5)        ; Pickup version + system id's
10 A4   FA05 CF   06    29  06E6  10$:    CMPC3   #PROC_C_NAM,PROC_NAM,16(R4)     ; Check the connect proc nam
                            06ED                                                  ;
                 1D     BA  06ED          POPR    #^M<R0,R2,R3,R4>                ; Restore regs (but save CC's)
                        05  06EF          RSB                                     ; Return condition codes
                            06F0
                            06F0  ;
                            06F0  ;
                            06F0  ; Error after connection established - VC disconnect most likely.
                            06F0  ;
                            06F0  ; If the CDT is the UCB$L_TWIN_CDT then simply do a DISCONNECT.  This CDT is
                            06F0  ; used for receives on connects to ourselves.  SCS will call us again for the
                            06F0  ; other half of that connection with the local CDB's CDT -- at that time, as
                          06F0  1645 ; in all other cases, we will run-down the CDB.
                          06F0  1646 ;
                          06F0  1647 ;  Inputs:
                          06F0  1648 ;       R0  =  Status
                          06F0  1649 ;       R3 --> CDT
                          06F0  1650 ;       R4 --> PDT
                          06F0  1651 ;
                          06F0  1652 CONN_ERR:
      54    5C A3    D0   06F0  1653         MOVL    CDT$L_AUXSTRUC(R3),R4      ; Pick up associated CDB
      55    50 A4    D0   06F4  1654         MOVL    CDB_L_UCB(R4),R5           ; Pick up UCB address
 0094 C5       53    D1   06F8  1655         CMPL    R3,UCB$L_TWIN_CDT(R5)      ; Is this the "local receive" CDT ?
               6F    12   06FD  1656         BNEQ    ZAP_CDB                    ; If NEQ no, ZAP the CDB
         0094 C5     D4   06FF  1657         CLRL    UCB$L_TWIN_CDT(R5)         ; Else, detach it from the UCB
      54    10 A3    D0   0703  1658         MOVL    CDT$L_PDT(R3),R4           ; Recover the PDT
                          0707  1659         DISCONNECT                         ; Tell SCS to cleanup.
                     05   070D  1660         RSB                                ; Done
                          070E  1661
```

```
                    070E                .SBTTL  CANCEL,  Cancel I/O routine
                    070E
                    070E  ;++
                    070E  ; CANCEL, Cancels an I/O operation in progress
                    070E  ;
                    070E  ;
                    070E  ; This routine cancels all I/O on the tributary.
                    070E  ;
                    070E  ; Inputs:         R2 = channel number
                    070E  ;                 R3 = current IRP address
                    070E  ;                 R4 = PCB address
                    070E  ;                 R5 = UCB address
                    070E  ;                 R8 = Cancel reason code:  0 => $CANCEL;  1 => $DASSGN
                    070E  ;
                    070E  ;                 IPL = FIPL
                    070E  ;
                    070E  ; Outputs:        R0-R3 are destroyed.
                    070E  ;
                    070E  ;
                    070E  ;--
                    070E  CANCEL:                                        ; Cancel an I/O operation
       0210 8F  BB  070E                PUSHR   #^M<R4,R9>               ; Save registers
                    0712                                                 ;
       50   52  D0  0712                MOVL    R2,R0                    ; Copy channel number
          025A  30  0715                BSBW    XLATE_CHAN               ; Translate channel
         10 50  E9  0718                BLBC    R0,20$                   ; Br if none
       01   58  D1  071B                CMPL    R8,#1                    ; $DASSGN ?
            09  12  071E                BNEQ    10$                      ; If NEQ then no
    50   3E A9  9A  0720                MOVZBL  CDB_B_TRB_ADDR(R9),R0    ; Pick up trib address
     00E0 C540  B4  0724                CLRW    UCB$W_VEC_CHAN(R5)[R0]   ; Zero channel entry
            40  10  0729  10$:          BSBB    ZAP_CDB_R9               ; Clear all CDB I/O
                    072B                                                 ;
       0210 8F  BA  072B  20$:          POPR    #^M<R4,R9>               ; Restore registers
         5C A5  B5  072F                TSTW    UCB$W_REFC(R5)           ; Last reference to unit?
            01  13  0732                BEQL    CAN_DEV                  ; If EQL yes, shutdown the device
                05  0734                RSB                              ; Return to caller
                    0735
```

```
                       0735              .SBTTL  CAN_DEV,  Device shutdown routine
                       0735
                       0735  ;++
                       0735  ; CAN_DEV -  Device shutdown routine
                       0735  ;
                       0735  ;
                       0735  ; This routine is called to shutdown the CI device.  All tributaries are
                       0735  ; zapped so that they will eventually run-down and be deleted.
                       0735  ;
                       0735  ; Inputs:         R3 = IRP address
                       0735  ;                 R5 = UCB address
                       0735  ;
                       0735  ;                 IPL = FIPL
                       0735  ;
                       0735  ; Outputs:        R0-R2 are clobbered.
                       0735  ;
                       0735  ;
                       0735  ;--
                       0735  CAN_DEV:                                                 ; Shutdown the device
               00  E5  0735            BBCC    #UCB$V_CN_INITED,-                     ;
         30 68 A5      0737                    UCB$W_DEVSTS(R5),50$                   ; Br if dev not inited
               38  BB  073A            PUSHR   #^M<R3,R4,R5>                          ;
                       073C            ;
                       073C            ;  Zap each tributary
                       073C            ;
            53 0F  D0  073C            MOVL    #MAX_TRB-1,R3                          ; Loop counter (zero indexed)
   54   00A0 C543  D0  073F  20$:      MOVL    UCB$L_VEC_CDB(R5)[R3],R4               ; Get next CDB
               02  13  0745            BEQL    30$                                    ; Br if none
               25  10  0747            BSBB    ZAP_CDB                                ; Cancel all I/O on trib
            F3 53  F4  0749  30$:      SOBGEQ  R3,20$                                 ; Loop
                       074C            ;
                       074C            ;  Remove our listener
                       074C            ;
      53   0090 C5 D0  074C            MOVL    UCB$L_LIS_CDT(R5),R3                   ; Pick up listening CDT
               0F  13  0751            BEQL    40$                                    ; None
           0090 C5 D4  0753            CLRL    UCB$L_LIS_CDT(R5)                      ;  and clear any trace
      54   0084 C5 D0  0757            MOVL    UCB$L_PDT(R5),R4                       ; PDT address, just in case
                       075C            DISCONNECT                                     ; Clear our name out of table
                       0762  40$:      ;
                       0762            ;  Clean up the UCB
                       0762            ;
         FFCF 8F   AA  0762            BICW2   #^C<UCB$M_ONLINE!UCB$M_POWER>,-        ;
            64 A5      0766                    UCB$W_STS(R5)                          ; Reset status
                       0768                                                           ;
               38  BA  0768            POPR    #^M<R3,R4,R5>                          ; Restore registers
               05      076A  50$:      RSB                                            ; Return
```

```
                          076B  1748          .SBTTL  ZAP_CDB,  Shutdown the tributary
                          076B  1749
                          076B  1750  ;++
                          076B  1751  ; ZAP_CDB    - Shutdown the tributary
                          076B  1752  ; ZAP_CDB_R9 - Shutdown the tributary, get CDB address from R9
                          076B  1753  ;
                          076B  1754  ;
                          076B  1755  ; This routine is called to abort all I/O pending for this tributary.
                          076B  1756  ;
                          076B  1757  ;     1) Disconnect the Virtual Circuit
                          076B  1758  ;     2) Cancel all outstanding I/O, abort all IRP, deallocate rcv'd buffers.
                          076B  1759  ;     3) Idle the CDB.
                          076B  1760  ;
                          076B  1761  ; Inputs:         R9 = CDB address (ZAP_CDB_R9 only, else not used)
                          076B  1762  ;                 R5 = UCB address
                          076B  1763  ;                 R4 = CDB address (ZAP_CDB only, else garbage)
                          076B  1764  ;
                          076B  1765  ;                 IPL = FIPL
                          076B  1766  ;
                          076B  1767  ; Outputs:        R0-R1 are destroyed.
                          076B  1768  ;
                          076B  1769  ;--
         54    59    D0   076B  1770  ZAP_CDB_R9:     MOVL    R9,R4                           ; Setup proper CDB pointer
                          076E  1771  ZAP_CDB:
                          076E  1772                  ;
                          076E  1773                  ;       If a DISCONNECT is issued on a connection that already has a
                          076E  1774                  ;       DISCONNECT pending, SCS thinks that something is wrong the port
                          076E  1775                  ;       and disconnects all circuits using it.  Therefore, make sure
                          076E  1776                  ;       we do not issue a second DISCONNECT for at least 10 seconds after
                          076E  1777                  ;       the last one was issued.  That should be enough time for normally
                          076E  1778                  ;       functioning circuits to complete a DISCONNECT dialogue.  If the
                          076E  1779                  ;       DISCONNECT is still pending after 10 seconds, its probably okay
                          076E  1780                  ;       to try it again in order to allow the user to run-down all I/O on
                          076E  1781                  ;       this channel.
                          076E  1782                  ;
         54 A4       D5   076E  1783                  TSTL    CDB_L_CDT(R4)                   ; Any CDT connected ?
            0E       13   0771  1784                  BEQL    2$                              ; If EQL, no DISCONNECT needed
         34 A4       C3   0773  1785                  SUBL3   CDB_L_ABSTIME(R4),-             ; Get seconds since last
   50   00000000'GF       0776  1786                          G^EXE$GL_ABSTIM,R0              ; DISCONNECT
       0A      50    D1   077C  1787                  CMPL    R0,#10                          ; At least 10 seconds?
               06    1F   077F  1788                  BLSSU   3$                              ; If LSSU can't DISCONNECT
                          0781  1789                                                          ;
               3C    BB   0781  1790  2$:             PUSHR   #^M<R2,R3,R4,R5>                ; Save regs
               03    10   0783  1791                  BSBB    5$                              ; Use subr call so that SCS's
                          0785  1792                                                          ; DISCONNECT code can return to
                          0785  1793                                                          ; a caller's caller
               3C    BA   0785  1794                  POPR    #^M<R2,R3,R4,R5>                ; Restore regs
                     05   0787  1795  3$:             RSB                                     ; Done
                          0788  1796
                          0788  1797  5$:             ;
                          0788  1798                  ;
                          0788  1799                  ;       DISCONNECT may return to our caller before returning here since SCS
                          0788  1800                  ;       has to enter into a dialogue with the remote node.  Therefore, the
                          0788  1801                  ;       stack must be clear.
                          0788  1802                  ;
                          0788  1803                  ;       FORK immediately after returning from the DISCONNECT in order to
                          0788  1804                  ;       make sure SCS will return all Xmt IRPs it knows about before we
```

```
                    0788  1805        ;  return the ones that are left.  This is not usually necessary since
                    0788  1806        ;  in most cases SCS will queue the DISCONNECT completion to the end of
                    0788  1807        ;  its own fork queue after all Xmt completion notifications have been
                    0788  1808        ;  queued.
                    0788  1809        ;
                    0788  1810        ;  If CDB_V_DISC is already set, then there is a DISCONNECT or FORK
                    0788  1811        ;  already in progress.  Do the DISCONNECT again, in case SCS is stuck,
                    0788  1812        ;  but simply return since the previous DISCONNECT will complete and
                    0788  1813        ;  the processing will continue from there.
                    0788  1814        ;
                    0788  1815        CLRBIT  #CDB_V_RUN,CDB_W_STS(R4)          ; No longer in RUN state
         55     54  D0  078D  1816    MOVL    R4,R5                             ; Save CDB pointer over call
      53     54 A5  D0  0790  1817    MOVL    CDB_L_CDT(R5),R3                  ; Pick up CDT address
             06     12  0794  1818    BNEQ    10$                               ; If NEQ then CDT was there
   25 3A A5  03     E3  0796  1819    BBCS    #CDB_V_DISC,CDB_W_STS(R5),30$     ; FORK to continue
             05         079B  1820    RSB                                       ; Return if already FORKing
                        079C  1821                                              ;
      54     10 A3  D0  079C  1822 10$:  MOVL    CDT$L_PDT(R3),R4               ; Pick up PDT address
34 A5    00000000'GF D0 07A0  1823    MOVL    G^EXE$GL_ABSTIM,CDB_L_ABSTIME(R5) ; Save DISCONNECT start time
   0A 3A A5  03     E3  07A8  1824    BBCS    #CDB_V_DISC,CDB_W_STS(R5),20$     ; Show we are disconnecting
                        07AD  1825    DISCONNECT #0                             ; Tell SCS to do it again
             05         07B6  1826    RSB                                       ; Done
                        07B7  1827                                              ;
                        07B7  1828 20$:  DISCONNECT #0                          ; Do it
   00000000'GF      16  07C0  1829 30$:  JSB     G^EXE$FORK                     ; FORK to synchronize cleanup
         54     55  D0  07C6  1830    MOVL    R5,R4                             ; Recover CDB address
      55     50 A4  D0  07C9  1831    MOVL    CDB_L_UCB(R4),R5                  ; Recover UCB address
                        07CD  1832    SETIPL  UCB$B_FIPL(R5)                    ; Sync with UCB
                        07D1  1833    CLRBIT  #CDB_V_DISC,CDB_W_STS(R4)         ; Show we are back
             04     10  07D6  1834    BSBB    40$                               ; Finish processing
                        07D8  1835    SETIPL  #CDB_C_FIPL                       ; Restore IPL
             05         07DB  1836    RSB
                        07DC  1837
                        07DC  1838 40$:  ;
                        07DC  1839        ;  Complete pending IO$_SETMODE, if any
                        07DC  1840        ;
         54 A4      D4  07DC  1841    CLRL    CDB_L_CDT(R4)                     ; Get rid of any trace
      53     30 A4  D0  07DF  1842    MOVL    CDB_L_SETMODE(R4),R3              ; Recover IRP
             0E     13  07E3  1843    BEQL    50$                               ; None there
         30 A4      D4  07E5  1844    CLRL    CDB_L_SETMODE(R4)                 ; Remove it from the CDB
      50     01     7D  07E8  1845    MOVQ    S^#SS$_NORMAL,R0                  ; Assume IO$V_SHUTDOWN
   09 20 A3  07     E1  07EB  1846    BBC     #IO$V_SHUTDOWN,IRP$W_FUNC(R3),60$ ; If BC then IO$V_STARTUP
           0105     30  07F0  1847    BSBW    IOPOST                            ; Send IRP to IOPOST
                        07F3  1848
                        07F3  1849 50$:  ;
                        07F3  1850        ;  Complete all Receive IRP's
                        07F3  1851        ;
      53     20 B4  0F  07F3  1852    REMQUE  @CDB_Q_RCV_IRP(R4),R3             ; Get next RCV IRP
             05     1D  07F7  1853    BVS     70$                               ; If VS then none
           00E3     30  07F9  1854 60$:  BSBW    ABORT_IRP_POST                 ; Abort the I/O request
             F5     11  07FC  1855    BRB     50$                               ; Get next entry
                        07FE  1856
                        07FE  1857 70$:  ;
                        07FE  1858        ;  Deallocate all Receive buffers
                        07FE  1859        ;
      50     28 B4  0F  07FE  1860    REMQUE  @CDB_Q_RCV_MSG(R4),R0             ; Get next buffer
             04     1D  0802  1861    BVS     80$                               ; If VS then empty
```

```
          3F   10  0804             BSBB    DEALLMEM                       ; Get rid of it
          F6   11  0806             BRB     70$                            ; Get next entry
                   0808
                   0808       80$:  ;
                   0808             ;  Complete all Transmit IPR's
                   0808             ;
  53   18 B4   0F  0808             REMQUE  @CDB_Q_XMT_IRP(R4),R3          ; Get next IRP
          05   1D  080C             BVS     90$                            ; If VS then none
        00CE   30  080E             BSBW    ABORT_IRP_POST                 ; Abort the I/O request
          F5   11  0811             BRB     80$                            ; Loop
                   0813
                   0813       90$:  ;
                   0813             ;  Idle the CDB
                   0813             ;
  3F A4    00  90  0813             MOVB    #CDB_C_IDLE,CDB_B_STA(R4)      ; Reinit CDB state
               05  0817             RSB
                   0818
        0818          .SBTTL  MSG_FORK,  Fork process for receipt of Seq Messages
        0818
        0818     ;++
        0818     ; MSG_FORK - Process received MSG
        0818     ;
        0818     ; Inputs:
        0818     ;
        0818     ;        R1  =   Bytes send/received
        0818     ;        R2 -->  Start of user data
        0818     ;        R3 -->  CDT
        0818     ;        R4 -->  PDT
        0818     ;
        0818     ;        IPL =   FIPL
        0818     ;
        0818     ; Outputs:
        0818     ;
        0818     ;--
        0818     MSG_FORK:
        0818          DEALLOC_MSG_BUF_REG                        ; Deallocate the message buffer
    05  081B          RSB
        081C
```

```
                       081C                  .SBTTL  DG_FORK,  Fork process for receipt of DG
                       081C
                       081C        ;++
                       081C        ; DG_FORK - Process received DG
                       081C        ;
                       081C        ; Inputs:
                       081C        ;
                       081C        ;         R0  =   0 - Received a DG
                       081C        ;             =   1 - Transmit finished
                       081C        ;         R1  =   Bytes send/received
                       081C        ;         R2 -->  Start of user data
                       081C        ;         R3 -->  CDT
                       081C        ;         R4 -->  PDT
                       081C        ;
                       081C        ;         IPL =   FIPL
                       081C        ;
                       081C        ; Outputs:
                       081C        ;
                       081C        ;--
                       081C        DG_FORK:
   54    5C A3    D0   081C                  MOVL    CDT$L_AUXSTRUC(R3),R4             ; Pick up pointer to CDB
            20    13   0820                  BEQL    EMPTY                              ; Closed CDB, discard
53    52    01    C3   0822                  SUBL3   #1,R2,R3                           ; Make a biased copy of msg ptr
      52    20    C2   0826                  SUBL    #32,R2                             ; Reset R2 to head of PPD buffer
   55    08 A2    32   0829                  CVTWL   8(R2),R5                           ; Get offset to CXB header
            0F    18   082D                  BGEQ    20$                                ; If GEQ then bug
      52    55    C0   082F                  ADDL    R5,R2                              ; Reset R2 to head of CXB buffer
                       0832                                                             ;
            53    D6   0832        10$:      INCL    R3                                 ; Advance to next byte
   63    FF 8F    91   0834                  CMPB    #-1,(R3)                           ; Pad byte ?
            11    12   0838                  BNEQ    DG                                 ; If NEQ not pad byte
            51    D7   083A                  DECL    R1                                 ; Reduce count
            F4    14   083C                  BGTR    10$                                ; If LEQ then no data
                       083E
                       083E        20$:      BUG_CHECK  INCONSTATE,FATAL                ; Illegal offset
                       0842                                                             ;
      50    52    D0   0842        EMPTY:    MOVL    R2,R0                              ; Pick up buffer
                       0845        DEALLMEM:                                            ;
00000000'GF       17   0845                  JMP     G^COM$DRVDEALMEM                   ; Deallocate buffer
                       084B                                                             ;
                       084B        DG:       ;
                       084B                  ;  Update counters
                       084B                  ;
                       084B                  ASSUME  CDB_L_BSN EQ 4+CDB_L_BRC
                       084B                  ASSUME  CDB_L_DBR EQ 4+CDB_L_BSN
                       084B                  ASSUME  CDB_L_DBS EQ 4+CDB_L_DBR
                       084B
   55    40 A4    9E   084B                  MOVAB   CDB_L_BRC(R4),R5                   ; Point to receive counter base
         03 50    E9   084F                  BLBC    R0,5$                              ; If LBC, then rcv
      55    04    C0   0852                  ADDL    #4,R5                   ; Adance to xmt counter base
      65    51    C0   0855        5$:       ADDL    R1,(R5)                            ; Update byte count
            03    1E   0858                  BCC     10$                                ; Br if no overflow
      65    01    CE   085A                  MNEGL   #1,(R5)                            ; Else, latch it
         08 A5    D6   085D        10$:      INCL    8(R5)                              ; Update message count
            04    1E   0860                  BCC     20$                                ; If CC, no overflow
   08 A5    01    CE   0862                  MNEGL   #1,8(R5)                           ; Else, latch it
   55    50 A4    D0   0866        20$:      MOVL    CDB_L_UCB(R4),R5                   ; Pick up ptr to UCB
```

```
          1D 50   E8  086A  1959          BLBS    R0,SEND_FORK                    ; If LBS then xmt complete
                      086D  1960          ;
                      086D  1961          ;   RECEIVE complete - if there is a pending receive I/O request,
                      086D  1962          ;   complete it.  Otherwise, queue the buffer.
                      086D  1963          ;
                      086D  1964          ASSUME  CDB_V_RUN  EQ  0                ;
       D1 3A A4   E9  086D  1965          BLBC    CDB_W_STS(R4),EMPTY             ; Br if trib not in RUN state
          3D A4   97  0871  1966          DECB    CDB_B_RCV_FQ(R4)                ; Dec the buffer count
 0E A2   53   52  A3  0874  1967          SUBW3   R2,R3,CXB$W_OFFSET(R2)          ; Store offset to message
    0C A2     51  B0  0879  1968          MOVW    R1,CXB$W_LENGTH(R2)             ; Set size
    53     20 B4  0F  087D  1969          REMQUE  @CDB_Q_RCV_IRP(R4),R3           ; Remove waiting IRP
              2C  1C  0881  1970          BVC     FINISH_RCV_IO                   ; If VC then gone one, finish
                      0883  1971                                                  ;  the I/O & exit
       2C B4  62  0E  0883  1972          INSQUE  (R2),@CDB_Q_RCV_MSG+4(R4)       ; Queue receive msg for late
            0086  31  0887  1973          BRW     FILLRCVLIST                     ; Fill the receive buffer pool
                      088A  1974
                      088A  1975  SEND_FORK:
                      088A  1976          ;
                      088A  1977          ;   TRANSMIT completed.  Locate and deque XMIT IRP and post it.
                      088A  1978          ;
                      088A  1979          ;   NOTE: the IRP's may be returned out of sequence on a power fail.
                      088A  1980          ;
 50     51    10  9C  088A  1981          ROTL    #16,R1,R0                       ; Size in R0 high word
        50    01  B0  088E  1982          MOVW    #SS$_NORMAL,R0                  ; Status in low word
   51      18 A4  9E  0891  1983          MOVAB   CDB_Q_XMT_IRP(R4),R1            ; Address queue header
        53    51  D0  0895  1984          MOVL    R1,R3                           ; Make a copy
        53    63  D0  0898  1985  20$:    MOVL    (R3),R3                         ; Get next IRP
        51    53  D1  089B  1986          CMPL    R3,R1                           ; Back to head of queue?
              0B  13  089E  1987          BEQL    50$                             ; If EQL then yes, bugcheck
     2C A3    52  D1  08A0  1988          CMPL    R2,IRP$L_SVAPTE(R3)             ; Buffer address match ?
              F2  12  08A4  1989          BNEQ    20$                             ; If NEQ no, try again
        53    63  0F  08A6  1990          REMQUE  (R3),R3                         ; Remove IRP from queue
              39  11  08A9  1991          BRB     SUC_TRB_IOPOST                  ; Complete the I/O with trib
                      08AB  1992                                                  ; info stuffed into IOST2
                      08AB  1993
                      08AB  1994  50$:    BUG_CHECK  INCONSTATE,FATAL
                      08AF  1995
```

```
                           08AF         .SBTTL  FINISH_RCV_IO,  Finish receive I/O processing
                           08AF
                           08AF   ;++
                           08AF   ; FINISH_RCV_IO - Finish receive I/O processing
                           08AF   ;
                           08AF   ;
                           08AF   ; This routine finishes receive processing and sends the IRP back to IOPOST.
                           08AF   ; The receive free list is filled and a receive is started if needed.
                           08AF   ;
                           08AF   ;  Inputs:        R2 = message buffer address
                           08AF   ;                 R3 = IRP address
                           08AF   ;                 R4 = CDB address
                           08AF   ;                 R5 = UCB address
                           08AF   ;
                           08AF   ;                 IPL = FIPL
                           08AF   ;
                           08AF   ;  Outputs:       R0-R4 are clobbered.  All other registers are preserved.
                           08AF   ;
                           08AF   ;
                           08AF   ;--
                           08AF   FINISH_RCV_IO:                                        ; Finish recieve I/O request
     2C A3     52     D0   08AF           MOVL      R2,IRP$L_SVAPTE(R3)                 ; Save block address
     62    0E  A2     3C   08B3           MOVZWL    CXB$W_OFFSET(R2),(R2)               ; Store offset to message
           62  52     C0   08B7           ADDL      R2,(R2)                             ; Make it a pointer
  04 A2    3C  A3     D0   08BA           MOVL      IRP$L_IOST2(R3),4(R2)               ; Set address of user buffer
     51    0C  A2     B0   08BF           MOVW      CXB$W_LENGTH(R2),R1                 ; Get size of transfer
     32 A3     51     B1   08C3           CMPW      R1,IRP$W_BCNT(R3)                   ; Request larger than actual?
               04     1A   08C7           BGTRU     10$                                 ; Br GTRU then yes
     32 A3     51     B0   08C9           MOVW      R1,IRP$W_BCNT(R3)                   ; Set size to transfer
     50    30  A3     D0   08CD   10$:    MOVL      IRP$W_BCNT-2(R3),R0                 ; Setup size of xfer in high word
           50  01     B0   08D1           MOVW      #SS$_NORMAL,R0                      ; Setup status in low word
               0E     12   08D4           BNEQ      SUC_TRB_IOPOST                      ; Br if success
   50    0054  8F     3C   08D6           MOVZWL    #SS$_CTRLERR,R0                     ; Set data path error
               51     D4   08DB           CLRL      R1                                  ; Init second longword
               19     11   08DD           BRB       IOPOST                              ; Post it
                           08DF
                           08DF   ABORT_IRP_POST:
           50  2C     7D   08DF           MOVQ      S^#SS$_ABORT,R0                     ; Setup IOSB image
               14     11   08E2           BRB       IOPOST                              ; Finish up
                           08E4
                           08E4
                           08E4   SUC_TRB_IOPOST:                                       ; Successful Trib I/O completion
51    00002800 8F     D0   08E4           MOVL      #XM$M_STS_ACTIVE!-                  ; Set device dependent bits to indicate
                           08EB                      XM$M_STS_RUNNING,R1                ; that the circuit is running
     3D A4     06     91   08EB           CMPB      #RBFTHR,CDB_B_RCV_FQ(R4);           ; Receive queue under threshold ?
               07     1B   08EF           BLEQU     IOPOST                              ; If LEQU then no
51    00001000 8F     C8   08F1           BISL      #XM$M_STS_BUFFAIL,R1                ; Signal buffer threshold problems
      38 A3    50     7D   08F8   IOPOST: MOVQ      R0,IRP$L_IOST1(R3)                  ; Store IOSB image
    00000000'GF       17   08FC           JMP       G^COM$POST                          ; Post IRP
                           0902
```

```
                        0902                   .SBTTL  FILLRCVLIST,  Fill receive buffer list
                        0902                   .SBTTL  ADDRCVLIST,   Move IRP buffer to free list
                        0902       ;++
                        0902       ; FILLRCVLIST - Add to the receive buffer list
                        0902       ; ADDRCVLIST  - Add IRP buffer to free list
                        0902       ;
                        0902       ; This routine is entered to make sure that the receive buffer pool is full.
                        0902       ; If it is not, buffers are allocated and queued to the list until it is.
                        0902       ;
                        0902       ; For ADDRCVLIST, any buffer attached to the IRP is added to the free list
                        0902       ; even if the list is already filled.
                        0902       ;
                        0902       ; Inputs:        R3 - IRP address (ADDRCVLIST only)
                        0902       ;                R4 - CDB address
                        0902       ;                R5 - UCB address
                        0902       ;
                        0902       ; Outputs:       Only R0-R2 are clobbered.
                        0902       ;
                        0902       ;--
                        0902                             .ENABL  LSB
                        0902       ADDRCVLIST:                                                 ; Add IRP buffer to free list
       52    2C A3  D0  0902               MOVL    IRP$L_SVAPTE(R3),R2                         ; Get buffer, if any
                08  13  0906               BEQL    FILLRCVLIST                                 ; If none, fill rcv list if needed
             2C A3  D4  0908               CLRL    IRP$L_SVAPTE(R3)                            ; Detach the buffer
                        090B               PUSHQ   R3                                          ; Save regs
                1E  11  090E               BRB     20$                                         ; Add buffer to free list
                        0910
                        0910       FILLRCVLIST:
                        0910               PUSHQ   R3                                          ; Save regs
    3C A4    3D A4  91  0913       10$:    CMPB    CDB_B_RCV_FQ(R4),CDB_B_RCV_CNT(R4) ; Should new block be added?
                45  1E  0918               BGEQU   50$                                         ; If GEQU no - list filled
51   38 A4  004C 8F A1  091A               ADDW3   #CXB$C_OVERHEAD,CDB_W_BUFSIZ(R4),R1; Compute block size need
       00000000'GF  16  0921               JSB     G^EXE$ALONONPAGED                           ; Allocate nonpaged memory
             35 50  E9  0927               BLBC    R0,50$                                      ; If LBC then failure
       08 A2    51  B0  092A               MOVW    R1,IRP$W_SIZE(R2)                           ; Insert block size
                        092E       20$:    ;
                        092E               ;  Give SCS receive datagram
                        092E               ;
       53    54 A4  D0  092E               MOVL    CDB_L_CDT(R4),R3                            ; Pick up CDT address
    20 A3  009E C5  91  0932               CMPB    UCB$B_CN_PORT(R5),CDT$B_RSTATION(R3) ; Talking to ourselves?
                05  12  0938               BNEQ    30$                                         ; If NEQ, no
       53  0094 C5  D0  093A               MOVL    UCB$L_TWIN_CDT(R5),R3                       ; Yes, use other CDT
       54  0084 C5  D0  093F       30$:    MOVL    UCB$L_PDT(R5),R4                            ;  and PDT address
       0A A2    1B  9B  0944               MOVZBW  S^#DYN$C_CXB,IRP$B_TYPE(R2)                 ; Insert block type
                        0948               QUEUE_DG_BUF                                        ; Put the block on the free que
             09 50  E9  094B               BLBC    R0,40$                                      ; Br if failure
       54    04 AE  D0  094E               MOVL    4(SP),R4                                    ; Pick up CDB pointer
             3D A4  96  0952               INCB    CDB_B_RCV_FQ(R4)                            ; Bump free que count
                BC  11  0955               BRB     10$                                         ; Try for more
       50       52  D0  0957       40$:    MOVL    R2,R0                                       ; Pick up the buffer
                03  13  095A               BEQL    50$                                         ; There is none
              FEE6  30  095C               BSBW    DEALLMEM                                    ; Get rid of the buffer
                        095F                                                                   ;
                        095F       50$:    POPQ    R3                                          ; Restore regs
                    05  0962               RSB                                                 ; Return
                        0963
                        0963                             .DSABL  LSB
```

```
                       0963   2106             .SBTTL  XLATE,  Translate Channel to CDB address
                       0963   2107
                       0963   2108 ;++
                       0963   2109 ; XLATE - Translate Channel to CDB address
                       0963   2110 ;
                       0963   2111 ; This routine is called to return the CDB address for a particular
                       0963   2112 ; channel.
                       0963   2113 ;
                       0963   2114 ;
                       0963   2115 ; Inputs:          R3 = IRP address
                       0963   2116 ;                  R5 = UCB address
                       0963   2117 ;
                       0963   2118 ; Outputs:         R0 - status return for success of call.
                       0963   2119 ;
                       0963   2120 ;                  R9 = CDB address if successful
                       0963   2121 ;                       1 otherwise
                       0963   2122 ;
                       0963   2123 ;                  R1,R2 are clobberd, all other registers are preserved.
                       0963   2124 ;
                       0963   2125 ;--
                       0963   2126 XLATE:                                            ; Translate CHAN into CDB address
      50    28 A3   3C 0963   2127             MOVZWL  IRP$W_CHAN(R3),R0                 ; Get channel
              09    10 0967   2128             BSBB    XLATE_CHAN                        ; Translate channel
      40 A3   51    90 0969   2129             MOVB    R1,IRP$B_INDEX(R3)                ; Save index in IRP
      54 A3   59    D0 096D   2130             MOVL    R9,IRP$L_CDB(R3)                  ; Store CDB address in IRP
                    05 0971   2131             RSB                                       ; Return to caller
                       0972   2132
                       0972   2133 XLATE_CHAN:
         51   0F    9A 0972   2134             MOVZBL  #MAX_TRB-1,R1                     ; Setup loop counter (zero indexed)
 00E0 C541    50    B1 0975   2135 10$:        CMPW    R0,UCB$W_VEC_CHAN(R5)[R1] ; Channels match?
              0C    13 097B   2136             BEQL    40$                               ; Br if yes - got it
         F5   51    F4 097D   2137             SOBGEQ  R1,10$                            ; Loop
   50    20D4 8F    3C 0980   2138 30$:        MOVZWL  #SS$_DEVINACT,R0                  ; Return channel offline
         59   01    D0 0985   2139             MOVL    #1,R9                             ; Setup "R9 invalid" flag
                    05 0988   2140             RSB                                       ; And leave
                       0989   2141
                       0989   2142 40$:        ;
                       0989   2143             ; Found match on channel
                       0989   2144             ;
 59    00A0 C541    D0 0989   2145             MOVL    UCB$L_VEC_CDB(R5)[R1],R9; Get CDB address
              04    13 098F   2146             BEQL    50$                               ; Br if no CDB address - error
      50      01    3C 0991   2147             MOVZWL  S^#SS$_NORMAL,R0                  ; Set successful return status
                    05 0994   2148             RSB                                       ; Return
                       0995   2149
                       0995   2150 50$:        BUG_CHECK  INCONSTATE,FATAL
                       0999   2151
```

```
                              0999               .SBTTL  VALIDATE_P2,  Validate P2 buffer parameters
                              0999
                              0999      ;++
                              0999      ; VALIDATE_P2 - Validate P2 buffer parameters
                              0999      ;
                              0999      ; This routine is called to validate the P2 buffer parameters. The parameters
                              0999      ; are checked against a parameter table which verifies that the minimum value
                              0999      ; and maximum value is not violated, and that status flags are set or clear
                              0999      ; as required.
                              0999      ;
                              0999      ; The way in this routine is written, the require word of the verification
                              0999      ; table can only have 1 bit set at a time.
                              0999      ;
                              0999      ; Inputs:         R1 = Address of parameter verification table
                              0999      ;                 R2 = Status word from UCB or CDB
                              0999      ;                 R3 = IRP address
                              0999      ;                 R5 = UCB address
                              0999      ;                 R9 = If low bit clear then ptr to context block (CDB or UCB)
                              0999      ;                      If low bit set then no context block exists
                              0999      ;
                              0999      ;                 IPL = FIPL or ASTDEL
                              0999      ;
                              0999      ; Outputs:        R0 = status return of parameters
                              0999      ;                 R1 = i.d. of parameter causing problem on error
                              0999      ;
                              0999      ;                 All other registers are preserved.
                              0999      ;
                              0999      ;--
                              0999      VALIDATE_P2:                                    ; Validate P2 buffer parameters
              01EA 8F    BB   0999               PUSHR   #^M<R1,R3,R5,R6,R7,R8>         ; Save registers
                              099D                                                      ; NB:R1 must be on top of stack
          56    2C B3    D0   099D               MOVL    @IRP$L_SVAPTE(R3),R6           ; Get system P2 buffer address
          58    32 A3    3C   09A1               MOVZWL  IRP$W_BCNT(R3),R8              ; Get size of P2 buffer
             58    06    C6   09A5               DIVL    #6,R8                          ; Get number of params in P2
                   4F    11   09A8               BRB     40$                            ; Treat as none if too few bytes
                              09AA      10$:     ;
                              09AA               ;  Loop to check next parameter in P2 buffer
                              09AA               ;
             50    86    3C   09AA               MOVZWL  (R6)+,R0                       ; Get parameter type from P2
             55    86    D0   09AD               MOVL    (R6)+,R5                       ; Get parameter value from P2
             57    6E    D0   09B0               MOVL    (SP),R7                        ; Get parameter table address
                              09B3      20$:     ;
                              09B3               ;  Loop to check P2 buffer parameter to circuit parameter table
                              09B3               ;
   51    87   F000 8F    AB   09B3               BICW3   #^C<PRM_M_TYPE>,(R7)+,R1; Get next param i.d.
                   46    13   09B9               BEQL    50$                            ; If EQL, at end of table
             51    50    B1   09BB               CMPW    R0,R1                          ; Parameters match?
                   05    13   09BE               BEQL    30$                            ; Br if yes
             57    0A    C0   09C0               ADDL2   #PARAM_C_ENTRY-2,R7            ; Else, skip to next parameter
                   EE    11   09C3               BRB     20$                            ; Try next parameter
                              09C5      30$:     ;
                              09C5               ;  Match found - check min,max,valid,invalid
                              09C5               ;
             53    87    3C   09C5               MOVZWL  (R7)+,R3                       ; Get offset/width
                15 59    E8   09C8               BLBS    R9,33$                         ; If LBS then no current block
   50    53    0A    00  EF   09CB               EXTZV   #OFF_V_VALUE,#OFF_S_VALUE,R3,R0 ; Get offset
   53    53    06    0A  EF   09D0               EXTZV   #OFF_V_WIDTH,#OFF_S_WIDTH,R3,R3 ; Get width
```

```
  53   6940   53   00   EF   09D5              EXTZV   #0,R3,(R9)[R0],R3         ; Get current value
              53   55   B1   09DB              CMPW    R5,R3                     ; Value change ?
                   19   13   09DE              BEQL    40$                       ; If EQL no, try next param
              87   55   B1   09E0      33$:    CMPW    R5,(R7)+                  ; Is the value too small?
                   1C   1F   09E3              BLSSU   50$                       ; Br if yes - error
              87   55   B1   09E5              CMPW    R5,(R7)+                  ; Is the value too big?
                   17   1A   09E8              BGTRU   50$                       ; Br if yes - error
              55   87   B0   09EA              MOVW    (R7)+,R5                  ; Pick up required
                   05   13   09ED              BEQL    35$                       ; None
              52   55   B3   09EF              BITW    R5,R2                     ; Check required bit
                   0D   13   09F2              BEQL    50$                       ; Br if not on - error
              52   87   B3   09F4      35$:    BITW    (R7)+,R2                  ; Check invalid bits
                   08   12   09F7              BNEQ    50$                       ; Br if on - error
              AE   58   F5   09F9      40$:    SOBGTR  R8,10$                    ; Br if more parameters
              50   01   3C   09FC              MOVZWL  S^#SS$_NORMAL,R0          ; Set success return
                   06   11   09FF              BRB     60$                       ; And return
                             0A01                                                ;
              6E   51   3C   0A01      50$:    MOVZWL  R1,(SP)                   ; Return bad parameter type
              50   14   D0   0A04              MOVL    #SS$_BADPARAM,R0          ; Set error return
            01EA   8F   BA   0A07      60$:    POPR    #^M<R1,R3,R5,R6,R7,R8>    ; Restore registers
                        05   0A0B              RSB                               ; Return to caller
```

```
                      0A0C           .SBTTL  UNPACK_P2_BUF,  Unpack a P2 parameter from P2 buffer
                      0A0C
                      0A0C   ;++
                      0A0C   ; UNPACK_P2_BUF - Unpack a P2 parameter from P2 buffer
                      0A0C   ;
                      0A0C   ;
                      0A0C   ; This routine is called to get a P2 parameter from the P2 buffer.
                      0A0C   ;
                      0A0C   ; Inputs:         R1 = Parameter type code
                      0A0C   ;                 R3 = IRP address
                      0A0C   ;                 R5 = UCB address
                      0A0C   ;
                      0A0C   ;                 IPL = IPL$_ASTDEL to allow user paging.
                      0A0C   ;
                      0A0C   ; Outputs:        R0 = SS$_NORMAL  if successful
                      0A0C   ;                      SS$_INSFARG otherwise
                      0A0C   ;                 R2 = Parameter value if success else destroyed
                      0A0C   ;
                      0A0C   ;                 All other registers are preserved.
                      0A0C   ;
                      0A0C   ;--
                      0A0C   UNPACK_P2_BUF:                                  ; Unpack P2 buffer
        00E0 8F   BB  0A0C           PUSHR   #^M<R5,R6,R7>                   ; Save registers
   56     2C B3   D0  0A10           MOVL    @IRP$L_SVAPTE(R3),R6            ; Get system P2 buffer address
   57     32 A3   3C  0A14           MOVZWL  IRP$W_BCNT(R3),R7               ; Get size of P2 buffer
        57   06   C6  0A18           DIVL    #6,R7                           ; Get number of params in P2
             11   13  0A1B           BEQL    20$                             ; Treat as none if too few bytes
        50   01   3C  0A1D           MOVZWL  S^#SS$_NORMAL,R0                ; Assume success
                      0A20   10$:    ;
                      0A20           ;  Loop to check next parameter in P2 buffer
                      0A20           ;
        55   86   3C  0A20           MOVZWL  (R6)+,R5                        ; Get parameter type from P2
        52   86   D0  0A23           MOVL    (R6)+,R2                        ; Get parameter value from P2
        55   51   B1  0A26           CMPW    R1,R5                           ; Parameters match?
             08   13  0A29           BEQL    30$                             ; Br if yes
          F2 57   F5  0A2B           SOBGTR  R7,10$                          ; Br if more parameters
                      0A2E                                                   ;
  50    0114 8F   3C  0A2E   20$:    MOVZWL  #SS$_INSFARG,R0                 ; Return error
        00E0 8F   BA  0A33   30$:    POPR    #^M<R5,R6,R7>                   ; Restore registers
                  05  0A37           RSB                                     ; Return to caller
                  0A38             .SBTTL  CN_END,  End of driver
                  0A38
        00000A40  0A38             . = <.+15>&<-16>
                  0A40
        00000018  0A40   PATCH:: .LONG   32-8
        00000A48' 0A44             .LONG   PATCH+8
        00000A60  0A48             .BLKB   32-8
                  0A60
                  0A60   ;++
                  0A60   ; Label that marks the end of the driver
                  0A60   ;--
                  0A60
                  0A60   CN_END:                                  ; Last location in driver
                  0A60
                  0A60             .END
```

```
$$$                             = 00000020 R    02     CDB_W_STS                         0000003A
$$$NUM                          = 00000000             CDT$B_RSTATION                  = 00000020
$$$OFF                          = 0000004C             CDT$L_AUXSTRUC                  = 0000005C
$$$TYP                          = 000063F3             CDT$L_PDT                       = 00000010
$$OP                            = 00000002             CHECK_REMOTE                      000006CF R    03
ABORT_IRP_POST                    000008DF R    03     CLR_IRP                           00000115 R    03
ABORT_REQ                         000002BB R    03     CN$DDT                            00000000 RG   03
ABORT_REQ1                        000002B8 R    03     CN_END                            00000A60 R    03
ABORT_START                       00000226 R    03     CN_FUNCTABLE                      00000038 R    03
ACCEPT                            00000626 R    03     COM$DRVDEALMEM                    ********    X  03
ADDRCVLIST                        00000902 R    03     COM$POST                          ********    X  03
ALT_START                         000001B8 R    03     CONN                              00000591 R    03
AT$_NULL                        = 00000005             CONN_ABO                          000005E4 R    03
BUG$_INCONSTATE                   ********    X  03    CONN_DATA                         00000100 R    03
CANCEL                            0000070E R    03     CONN_ERR                          000006F0 R    03
CAN_DEV                           00000735 R    03     CONN_FIN                          00000693 R    03
CDB_B_DUMMY                       0000005A             CONN_FIN1                         00000697 R    03
CDB_B_FIPL                        0000000B             COUNT_C_ENTRY                   = 00000004
CDB_B_RCV_CNT                     0000003C             CRB$L_INTD                      = 00000024
CDB_B_RCV_FQ                      0000003D             CXB$C_HEADER                    = 00000048
CDB_B_REMSYS                      00000059             CXB$C_OVERHEAD                  = 0000004C
CDB_B_REMVER                      00000058             CXB$W_LENGTH                    = 0000000C
CDB_B_RSTCNT                      0000005B             CXB$W_OFFSET                    = 0000000E
CDB_B_STA                         0000003F             DC$_SCOM                        = 00000020
CDB_B_TRB_ADDR                    0000003E             DDB$L_DDT                       = 0000000C
CDB_B_TYPE                        0000000A             DEALLMEM                          00000845 R    03
CDB_C_ACPT                      = 00000004             DEV$M_IDV                       = 04000000
CDB_C_CONN                      = 00000002             DEV$M_NET                       = 00002000
CDB_C_FIPL                      = 00000006             DEV$M_ODV                       = 08000000
CDB_C_IDLE                      = 00000000             DEV$M_REC                       = 00000001
CDB_C_LENGTH                    = 00000060             DG                                0000084B R    03
CDB_C_LSTN                      = 00000003             DG_FORK                           0000081C R    03
CDB_C_OPEN                      = 00000001             DPT$C_LENGTH                    = 00000038
CDB_L_ABSTIME                     00000034             DPT$C_VERSION                   = 00000004
CDB_L_BRC                         00000040             DPT$INITAB                        00000038 R    02
CDB_L_BSN                         00000044             DPT$M_SCS                       = 00000008
CDB_L_CDT                         00000054             DPT$REINITAB                      00000062 R    02
CDB_L_DBR                         00000048             DPT$TAB                           00000000 R    02
CDB_L_DBS                         0000004C             DYN$C_CIDG                      = 0000003B
CDB_L_FPC                         0000000C             DYN$C_CRB                       = 00000005
CDB_L_FR3                         00000010             DYN$C_CXB                       = 0000001B
CDB_L_FR4                         00000014             DYN$C_DDB                       = 00000006
CDB_L_SETMODE                     00000030             DYN$C_DPT                       = 0000001E
CDB_L_UCB                         00000050             DYN$C_NET                       = 00000017
CDB_M_RUN                       = 00000001             DYN$C_ORB                       = 00000049
CDB_Q_FORK                        00000000             DYN$C_UCB                       = 00000010
CDB_Q_RCV_IRP                     00000020             EMPTY                             00000842 R    03
CDB_Q_RCV_MSG                     00000028             EXE$ABORTIO                       ********    X  03
CDB_Q_XMT_IRP                     00000018             EXE$ALONONPAGED                   ********    X  03
CDB_V_ACPT                      = 00000002             EXE$BUFFRQUOTA                    ********    X  03
CDB_V_CONN                      = 00000001             EXE$FINISHIO                      ********    X  03
CDB_V_DISC                      = 00000003             EXE$FORK                          ********    X  03
CDB_V_REJECT                    = 00000004             EXE$GL_ABSTIM                     ********    X  03
CDB_V_RUN                       = 00000000             EXE$QIORETURN                     ********    X  03
CDB_W_BUFSIZ                      00000038             EXE$READCHK                       ********    X  03
CDB_W_REMPROT                   = 00000058             EXE$WRITECHK                      ********    X  03
CDB_W_SIZE                        00000008             FILLRCVLIST                       00000910 R    03
```

```
FINISH_ERR                         000002C4 R        03      NMA$C_CTCIR_DBR                  = 000003F2
FINISH_RCV_IO                      000008AF R        03      NMA$C_CTCIR_DBS                  = 000003F3
FINISH_REQ                         000002D5 R        03      NMA$C_DPX_FUL                    = 00000000
FINISH_SUC                         000002C1 R        03      NMA$C_DPX_HAL                    = 00000001
FUNCTAB_LEN                      = 0000004C                  NMA$C_LINCN_LOO                  = 00000001
GET_BUF                            0000012D R        03      NMA$C_LINCN_NOR                  = 00000000
GET_CHAR_RBUF                      000004D9 R        03      NMA$C_PCCI_MRB                   = 00000479
GET_CHAR_WBUF                      000004DD R        03      NMA$C_PCCI_MST                   = 00000AFA
IO$V_CLR_COUNT                   = 0000000A                  NMA$C_PCCI_TRI                   = 00000474
IO$V_CTRL                        = 00000009                  NMA$C_PCLI_BFN                   = 00000451
IO$V_RD_COUNT                    = 00000008                  NMA$C_PCLI_BUS                   = 00000AF1
IO$V_SHUTDOWN                    = 00000007                  NMA$C_PCLI_CON                   = 00000456
IO$V_STARTUP                     = 00000006                  NMA$C_PCLI_DUP                   = 00000457
IO$_READLBLK                     = 00000021                  NMA$C_STATE_OFF                  = 00000001
IO$_SENSEMODE                    = 00000027                  NMA$C_STATE_ON                   = 00000000
IO$_SETCHAR                      = 0000001A                  NMA$M_CNT_COU                    = 00008000
IO$_SETMODE                      = 00000023                  NMA$V_CNT_WID                    = 0000000D
IO$_VIRTUAL                      = 0000003F                  OFF_M_VALUE                      = 000003FF
IO$_WRITELBLK                    = 00000020                  OFF_S_VALUE                      = 0000000A
IOC$MNTVER                         ********      X   03      OFF_S_WIDTH                      = 00000006
IOC$RETURN                         ********      X   03      OFF_V_VALUE                      = 00000000
IOPOST                             000008F8 R        03      OFF_V_WIDTH                      = 0000000A
IRP$B_INDEX                        00000040                  OLD_C_PROT                       = 00000000
IRP$B_RMOD                       = 0000000B                  ORB$B_FLAGS                      = 0000000B
IRP$B_TYPE                       = 0000000A                  ORB$L_OWNER                      = 00000000
IRP$L_BCNT                       = 00000032                  ORB$M_PROT_16                    = 00000001
IRP$L_CDB                          00000054                  ORB$W_PROT                       = 00000018
IRP$L_EXTEND                     = 00000054                  P1                               = 00000000
IRP$L_IOST1                      = 00000038                  P2                               = 00000004
IRP$L_IOST2                      = 0000003C                  PARAM_C_ENTRY                    = 0000000C
IRP$L_SEGVBN                     = 00000048                  PATCH                              00000A40 RG       03
IRP$L_SVAPTE                     = 0000002C                  PB$L_PDT                         = 0000002C
IRP$M_FUNC                       = 00000002                  PCB$L_JIB                        = 00000080
IRP$Q_NT_PRVMSK                  = 00000040                  PDT$L_DEALRGMSG                  = 00000024
IRP$V_FUNC                       = 00000001                  PDT$L_QUEUEDG                    = 0000003C
IRP$W_BCNT                       = 00000032                  PDT$L_REJECT                     = 0000004C
IRP$W_BOFF                       = 00000030                  PDT$L_SENDRGDG                   = 0000007C
IRP$W_CHAN                       = 00000028                  PR$_IPL                          = 00000012
IRP$W_FUNC                       = 00000020                  PRM_M_INVALID                    = 00008000
IRP$W_SIZE                       = 00000008                  PRM_M_MAX                        = 00002000
IRP$W_STS                        = 0000002A                  PRM_M_MIN                        = 00001000
JIB$L_BYTCNT                     = 00000020                  PRM_M_REQUIRE                    = 00004000
LINE_CNT_NUM                     = 00000000                  PRM_M_TYPE                       = 00000FFF
LINE_CNT_TABLE                   = 000000EE R        03      PROC_C_NAM                       = 00000006
LINE_PRM_NUM                     = 00000004                  PROC_NAM                           000000F0 R        03
LINE_PRM_TABLE                   = 000000AA R        03      QIORET                             000002B2 R        03
LISTEN                             000003AF R        03      RBFMAX                           = 0000001F
LIS_ERR                            0000041F R        03      RBFMIN                           = 00000009
LIS_FORK                           00000602 R        03      RBFTHR                           = 00000006
MASKH                            = 00000080                  RCV_FDT                            00000184 R        03
MASKL                            = 00000000                  RCV_START                          00000229 R        03
MAX_TRB                          = 00000010                  REJECT                             0000061D R        03
MSG_FORK                           00000818 R        03      SB$L_PBCONNX                     = 00000014
NEW_CDB                            0000030A R        03      SBC$B_RSTATION1                  = 0000003C
NEW_TRIB                           000002DB R        03      SBO$C_LENGTH                     = 00000050
NMA$C_CTCIR_BRC                  = 000003E8                  SBO_LNG                          = 00000070
NMA$C_CTCIR_BSN                  = 000003E9                  SCS$ACCEPT                         ********      GX   03
```

```
SCS$CONFIG_SYS              ********  X    03    VALIDATE_P2                   00000999 R    03
SCS$CONNECT                 ********  X    03    VEC$L_UNITINIT              = 00000018
SCS$DISCONNECT              ********  X    03    XLATE                         00000963 R    03
SCS$GB_SYSTEMID             ********  X    03    XLATE_CHAN                    00000972 R    03
SCS$GW_MAXDG                ********  X    02    XM$M_STS_ACTIVE             = 00000800
SCS$LISTEN                  ******** GX    03    XM$M_STS_BUFFAIL            = 00001000
SEND_FORK                   0000088A R     03    XM$M_STS_RUNNING            = 00002000
SENSEMODE_FDT               0000042E R     03    XMT_FDT                       00000125 R    03
SENSE_C_BUF               = 00000080             XMT_RCV_FDT_CO                00000190 R    03
SENSE_TABLE                 00000426 R     03    XMT_START                     000001D2 R    03
SETMODE_CTRL                0000035E R     03    ZAP_CDB                       0000076E R    03
SETMODE_FDT                 0000023D R     03    ZAP_CDB_R9                    0000076B R    03
SIZ...                    = 00000001
SS$_ABORT                 = 0000002C
SS$_ACCVIO                = 0000000C
SS$_BADPARAM              = 00000014
SS$_BUFFEROVF             = 00000601
SS$_CTRLERR               = 00000054
SS$_DEVACTIVE             = 000002C4
SS$_DEVALRALLOC           = 00000641
SS$_DEVINACT              = 000020D4
SS$_DEVOFFLINE            = 00000084
SS$_INSFARG               = 00000114
SS$_NORMAL                = 00000001
SS$_REMRSRC               = 0000206C
START_TRIB                  0000052E R     03
SUC_TRB_IOPOST              000008E4 R     03
TRIB_CNT_NUM              = 00000004
TRIB_CNT_TABLE            = 000000DC R     03
TRIB_PRM_NUM              = 00000003
TRIB_PRM_TABLE            = 00000084 R     03
UCB$B_CN_PORT               0000009E
UCB$B_DEVCLASS            = 00000040
UCB$B_DIPL                = 0000005E
UCB$B_FIPL                = 0000000B
UCB$B_RCV_CNT               0000009F
UCB$C_CN_LENGTH           = 00000100
UCB$C_LENGTH              = 00000090
UCB$L_DEVCHAR             = 00000038
UCB$L_DGHDRSZ               00000098
UCB$L_LIS_CDT               00000090
UCB$L_PDT                 = 00000084
UCB$L_TWIN_CDT              00000094
UCB$L_VEC_CDB               000000A0
UCB$M_CN_INITED           = 00000001
UCB$M_ONLINE              = 00000010
UCB$M_POWER               = 00000020
UCB$V_CN_INITED           = 00000000
UCB$V_POWER               = 00000005
UCB$W_DEVBUFSIZ           = 00000042
UCB$W_DEVSTS              = 00000068
UCB$W_DUMMY                 0000009C
UCB$W_REFC                = 0000005C
UCB$W_STS                 = 00000064
UCB$W_VEC_CHAN              000000E0
UNIT_INIT                   00000110 R     03
UNPACK_P2_BUF               00000A0C R     03
```

```
                                         +----------------+
                                         ! Psect synopsis !
                                         +----------------+

PSECT name                       Allocation           PSECT No.  Attributes
----------                       ----------           ---------  ----------
.  ABS  .                        00000000  (     0.)  00 (   0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                            00000100  (   256.)  01 (   1.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
$$$105_PROLOGUE                  0000006D  (   109.)  02 (   2.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
$$$115_DRIVER                    00000A60  (  2656.)  03 (   3.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC LONG

                                      +------------------------+
                                      ! Performance indicators !
                                      +------------------------+

Phase                       Page faults    CPU Time       Elapsed Time
-----                       -----------    --------       ------------
Initialization                       29    00:00:00.07    00:00:00.85
Command processing                  113    00:00:00.39    00:00:04.55
Pass 1                              811    00:00:26.09    00:01:43.79
Symbol table sort                     0    00:00:03.73    00:00:16.44
Pass 2                              417    00:00:05.69    00:00:23.01
Symbol table output                  15    00:00:00.20    00:00:00.82
Psect synopsis output                 0    00:00:00.02    00:00:00.06
Cross-reference output                0    00:00:00.00    00:00:00.00
Assembler run totals               1387    00:00:36.20    00:02:29.55

The working set limit was 2400 pages.
217048 bytes (424 pages) of virtual memory were used to buffer the intermediate code.
There were 200 pages of symbol table space allocated to hold 3593 non-local and 117 local symbols.
2287 source lines were read in Pass 1, producing 22 object records in Pass 2.
68 pages of virtual memory were used to define 62 macros.

                                     +--------------------------+
                                     ! Macro library statistics !
                                     +--------------------------+

Macro library name                            Macros defined
------------------                            --------------
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                         36
_$255$DUA28:[SYSLIB]STARLET.MLB;2                      14
TOTALS (all libraries)                                 50

3866 GETS were required to define 50 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:CNDRIVER/OBJ=OBJ$:CNDRIVER MSRC$:CNDRIVER/UPDATE=(ENH$:CNDRIVER)+EXECML$/LIB
```

0107 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

XADRIVER
MAR

CONINTERR
LIS

XIDRIVER
MAR

PAMAC
MAR

ENDRIVER
LIS