```
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  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
```

FILEID**XDDRIVER

G 2

```
XX      XX  DDDDDDDD    DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
XX      XX  DDDDDDDD    DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
XX      XX  DD      DD  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
XX      XX  DD      DD  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
  XX  XX    DD      DD  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
  XX  XX    DD      DD  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
    XX      DD      DD  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
    XX      DD      DD  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
  XX  XX    DD      DD  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
  XX  XX    DD      DD  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
XX      XX  DD      DD  DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
XX      XX  DD      DD  DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
XX      XX  DDDDDDDD    DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....
XX      XX  DDDDDDDD    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          .NOSHOW  CONDITIONALS
0000
0000          .TITLE  XDDRIVER - VAX/VMS DMP11/DMV11 Device 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 DMP11/DMV11 Device driver
0000    ;
0000    ; ABSTRACT:
0000    ;
0000    ;       This module contains the DMP11/DMV11 driver FDT routines,
0000    ;       interrupt dispatcher, interrupt service and fork routines.
0000    ;
0000    ; AUTHOR:
0000    ;
0000    ;       R. GAMACHE      13-JUL-81
0000    ;
0000    ; MODIFICATION HISTORY:
0000    ;
0000    ;       V03-036 RNG0036         Rod N. Gamache          7-Aug-1984
0000    ;               Make sure requests on the XMIT Pending queue are not aborted
0000    ;               without releasing map registers, when a trib is shutdown.
0000    ;               Set dead threshold from 16 to 3.
0000    ;
0000    ;       V03-035 RNG0035         Rod N. Gamache          4-May-1984
0000    ;               Set DEV$M_NET and DEV$M_AVL bits in device characteristics.
0000    ;
0000    ;       V03-034 RNG0034         Rod N. Gamache          19-Apr-1984
0000    ;               Don't allow transmits larger than 16 KB.
0000    ;
0000    ;       V03-033 RNG0033         Rod N. Gamache          14-Dec-1983
0000    ;               Fix max tribs for DMV11 to be 16 instead of 32.
0000    ;               Fix problem with "streaming tribs" and losing the
```

```
0000    ;               transmit slots.
0000    ;               Get descriptor size as a word rather than a longword.
0000    ;
0000    ;       V03-032 RNG0032         Rod N. Gamache          13-Sep-1983
0000    ;               Add support for the DMV11 in 22-bit mode.
0000    ;
0000    ;       V03-031 RNG0031         Rod N. Gamache          31-Aug-1983
0000    ;               Make all NOOP function request clear BSEL7.
0000    ;               Aborted INPUT CANCEL requests now release the input CSRs.
0000    ;               CANCEL no longer aborts transmits in progress prematurely,
0000    ;               this allows the FORK process to return map registers.
0000    ;
0000    ;       V03-030 RNG0030         Rod N. Gamache          29-Jun-1983
0000    ;               Fix the XMIT and RECV operations to check the ACTIVE
0000    ;               bit rather than the ESTAB bit when performing I/O operations.
0000    ;               Set up the P1 characteristics on a SENSEMODE after finding
0000    ;               the polling state.
0000    ;               Add journaling to file.
0000    ;
0000    ;       V03-029 RNG0029         Rod N. Gamache          23-Jun-1983
0000    ;               Remove internal definition of IRP$Q_STATION (for V4 only)
0000    ;
0000    ;       V03-028 RNG0028         Rod N. Gamache          12-May-1983
0000    ;               Lots of cleanup for MOP mode.
0000    ;
0000    ;       V03-027 RNG0027         Rod N. Gamache          09-May-1983
0000    ;               Wait a little while before issueing a STOP tributary
0000    ;               request. This is in case we just processed a $CANCEL
0000    ;               request and the trib hasn't finished running down.
0000    ;
0000    ;       V03-026 RNG0025         Rod N. Gamache          20-Jan-1983
0000    ;               Initialize the UCB Fork Process address to be the fork
0000    ;               Process routine and not the timeout routine. This is
0000    ;               because in the event a of a device timeout, before the
0000    ;               device is really initialized, then the timeout routine
0000    ;               does not need to have an offset to the timeout routine.
0000    ;               Don't call GET_CDB on HALTT request!
0000    ;               Change default Device IPL to be 21 (instead of 22).
0000    ;
0000    ;       V03-025 RNG0024         Rod N. Gamache          3-Dec-1982
0000    ;               Don't reset the tributary DEVDEPEND field on HALTT request.
0000    ;
0000    ;       V03-024 RNG0023         Rod N. Gamache          4-Nov-1982
0000    ;               Save R4 in Write FDT routine. Setup address of timeout
0000    ;               routine for new fork process in transmit routine.
0000    ;
0000    ;       V03-023 RNG0022         Rod N. Gamache          21-Sep-1982
0000    ;               Add another fork block to UCB to allow a fork on transmit
0000    ;               requests - allows users to transmit any size message up to
0000    ;               16K. Remove the code to pre-allocate one transmit map register.
0000    ;
0000    ;       V03-022 RNG0021         Rod N. Gamache          27-Aug-1982
0000    ;               Fix the CANCEL routine, to only abort I/O on $CANCEL
0000    ;               request and not halt the tributary.
0000    ;
0000    ;--
```

```
          0000          .SBTTL  External and local symbol definitions
          0000
          0000  ;
          0000  ; System definitions
          0000  ;
          0000
          0000          $ACBDEF                                 ; AST control block
          0000          $ADPDEF                                 ; Adapter control block
          0000          $CANDEF                                 ; Define $CANCEL reason codes
          0000          $CRBDEF                                 ; Channel request block
          0000          $CXBDEF                                 ; Define CXB header
          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          $FKBDEF                                 ; Fork block definitions
          0000          $IDBDEF                                 ; Interrupt data block
          0000          $IODEF                                  ; I/O function codes
          0000          $IPLDEF                                 ; Hardware IPL definitions
          0000          $IRPDEF                                 ; I/O request packet
          0000          $JIBDEF                                 ; Job information block defs
          0000          $NMADEF                                 ; Network Management definitions
          0000          $PCBDEF                                 ; Process control block
          0000          $PRDEF                                  ; Processor register definitions
          0000          $PTEDEF                                 ; Define system PTEs
          0000          $SSDEF                                  ; System status codes
          0000          $UBADEF                                 ; UBA definitions
          0000          $UCBDEF                                 ; Unit control block
          0000          $VADEF                                  ; Virtual address defs
          0000          $VECDEF                                 ; Interrupt vector block
          0000          $XMDEF                                  ; XMDRIVER symbols
          0000
          0000  ;
          0000  ; Local symbol definitions
          0000  ;
          0000
00000001  0000  JNX$$$ = 1                                      ; Enables journaling
          0000
          0000  ;
          0000  ; $QIO parameter offsets
          0000  ;
00000000  0000  P1      = 0                                     ; Parameter 1
00000004  0000  P2      = 4                                     ; Parameter 2
00000008  0000  P3      = 8                                     ; Parameter 3
0000000C  0000  P4      = 12                                    ; Parameter 4
00000010  0000  P5      = 16                                    ; Parameter 5
          0000
          0000  ;
          0000  ; Constants
          0000  ;
          0000
00000007  0000  MAX_RCV = 7                                     ; Maximum number outstanding receives
00000007  0000  MAX_XMT = 7                                     ; Maximum number outstanding transmits
00000004  0000  MAX_RCV_UV1 = 4                                 ; Maximum number of receives on u-VAX I
00000002  0000  MAX_XMT_UV1 = 2                                 ; Maximum number of transmits on u-VAX I
00003FFF  0000  MAX_BUFSIZ  = 16383                             ; Maximum DMP/DMV buffer size
```

```
00000400  0000  MAX_BUFSIZ_UV1 = 1024                        ; Maximum size buffer on u-VAX I
00001800  0000  UV1_BUFFER_AREA = <MAX_XMT_UV1*MAX_BUFSIZ_UV1>+- ; Size of u-VAX physically
          0000                    <MAX_RCV_UV1*MAX_BUFSIZ_UV1> ;  contiguous buffer area
000000FF  0000  INF_RCV = 255                                ; "Logical" infinite receives (unlimited)
0000007F  0000  INF_XMT = 127                                ; "Logical" infinite transmits
          0000
00000200  0000  XD_DEF_BUFSIZ   = 512                        ; Default buffer size
00000020  0000  MAX_DMP_TRIB    = 32                         ; Max tributaries on DMP11 device
00000010  0000  MAX_DMV_TRIB    = 16                         ; Max tributaries on DMV11 device
00000004  0000  MAX_XMT_TRB     = 4                          ; Maximum number of xmits per trib
00000004  0000  DSC$A_POINTER   = 4                          ; Descriptor buffer address field
          0000  P1_TRIB_MASK    = XM$M_STS_ACTIVE!-          ; Mask of user "read only" bits
000028FF  0000                    XM$M_STS_RUNNING!^X<FF> ; in CDB device dependent longword
000000FF  0000  P1_LINE_MASK    = ^X<FF>                     ; Mask of user "read only" bits
          0000
          0000  ;
          0000  ; Journal buffer definitions
          0000  ;
00000001  0000  JNL_XMT = 1                                  ; XMIT operation
00000002  0000  JNL_RCV = 2                                  ; RECV operation
00000003  0000  JNL_QIO = 3                                  ; QIO (SETMODE/SENSEMODE) operation
00000004  0000  JNL_QUE = 4                                  ; QUEUED PACKET operation
00000005  0000  JNL_POST = 5                                 ; POSTED PACKET operation
00000006  0000  JNL_XMTA = 6                                 ; Altstart XMIT operation
00000007  0000  JNL_RCVA = 7                                 ; Altstart RECV operation
00000008  0000  JNL_CSRIN = 8                                ; Input to CSRs (same as QUE)
00000009  0000  JNL_CSROUT = 9                               ; Output CSRs
0000000A  0000  JNL_CANC = 10                                ; Cancel request
```

```
0000                                                  ; in UCB device dependent longword
0000    ;
0000    ; Local macros
0000    ;
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  INCC    COUNTER,CONTEXT=L,?L  ; Increment counter
0000                    INC'CONTEXT     COUNTER ; Do Increment
0000                    BCC     L               ; Br if no carry set
0000                    DEC'CONTEXT     COUNTER ; Leave at maximum value
0000            L:
0000            .ENDM   INCC
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            .MACRO  PARAM   TYPE,OFFSET,WIDTH,MIN,MAX,INVALID,BASE
0000    ;
0000    ; Inputs:
0000    ;
0000    ;       TYPE = Parameter type
0000    ;       OFFSET = Offset in UCB/CDB to current value
0000    ;       WIDTH = Width of field in UCB/CDB (B,W,L)
0000    ;       MIN = Minimum value parameter is allowed to take
0000    ;       MAX = Maximum value parameter is allowed to take
0000    ;       INVALID = Invalid flags in status word
0000    ;       BASE = Data base (LINE,TRIB)
0000    ;
0000            .IF BLANK TYPE
0000            .WORD   0
0000            .IF_FALSE
0000            $$$TYP = TYPE & PRM_M_TYPE                      ; Isolate type code
0000            .IIF NOT_BLANK  <MIN>, $$$TYP = $$$TYP!PRM_M_MIN
0000            .IIF NOT_BLANK  <MAX>, $$$TYP = $$$TYP!PRM_M_MAX
0000            .IIF NOT_BLANK  <INVALID>, $$$TYP = $$$TYP!PRM_M_INVALID
0000                    .WORD   $$$TYP
0000            $$$OFF = OFFSET & OFF_M_VALUE                   ; Isolate offset only
0000            $$$WID = 0                                      ; Set null width
0000            .IIF IDN <WIDTH><B>, $$$WID = <1@OFF_V_WIDTH>
0000            .IIF IDN <WIDTH><W>, $$$WID = <2@OFF_V_WIDTH>
0000            .IIF IDN <WIDTH><L>, $$$WID = <3@OFF_V_WIDTH>
0000                    .WORD   $$$OFF!$$$WID
0000            .IIF NOT_BLANK  <MIN>, .WORD    MIN
```

```
0000          .IIF NOT_BLANK  <MAX>, .WORD      MAX
0000          .IIF NOT_BLANK  <INVALID>, .WORD          INVALID
0000          'BASE'_PRM_BUFSIZ = 'BASE'_PRM_BUFSIZ + 6
0000          .ENDC
0000          .ENDM   PARAM
0000
0000          .MACRO  COUNT    TYPE,BITMAP=NO,WIDTH=8,BASE
0000          $$$TYP = TYPE & NMA$M_CNT_TYP
0000          .IIF IDN <BITMAP><YES>, $$$TYP = $$$TYP!<NMA$M_CNT_MAP>
0000          $$$WID = 0                        ; Set reserved mask width
0000          .IIF IDN <WIDTH><8>, $$$WID = <1@NMA$V_CNT_WID>
0000          .IIF IDN <WIDTH><16>, $$$WID = <2@NMA$V_CNT_WID>
0000          .IIF IDN <WIDTH><32>, $$$WID = <3@NMA$V_CNT_WID>
0000          .IIF EQ $$$WID, .ERROR            ; Invalid bit width value
0000          .WORD   NMA$M_CNT_COU!$$$WID!$$$TYP
0000          'BASE'_CNT_BUFSIZ = 'BASE'_CNT_BUFSIZ + 2 + <WIDTH/8>
0000          .IIF IDN <BITMAP><YES>, 'BASE'_CNT_BUFSIZ = 'BASE'_CNT_BUFSIZ + 2
0000          .ENDM   COUNT
0000
0000          .MACRO  SKIP     BIT,LOC,REG,CONTEXT=W,?L          ; SKIP FIELD
0000          BBC     #BIT,LOC,L                ; Br if field not present
0000          TST'CONTEXT      (REG)+           ; Skip next field
0000     L:
0000          .ENDM   SKIP
0000
0000     .MACRO   $DISPATCH,       INDX,VECTOR,TYPE=W,NMODE=S^#,?MN,?MX,?S,?SS,?ZZ
0000     SS:
0000
0000          .MACRO  $DSP1,$DSP1_1
0000                  .IRP     $DSP1_2,$DSP1_1
0000                           $DSP2    $DSP1_2
0000                  .ENDR
0000          .ENDM   $DSP1
0000
0000          .MACRO  $DSP2,$DSP2_1,$DSP2_2
0000                  .=<$DSP2_1-MN>*2 + S
0000                  .WORD    $DSP2_2-S
0000          .ENDM   $DSP2
0000
0000
0000          .MACRO  $BND1,$BND1_1,$BND1_2,$BND1_3
0000                  $BND2    $BND1_1,$BND1_2
0000          .ENDM   $BND1
0000
0000          .MACRO  $BND2,$BND2_1,$BND2_2
0000                  .IIF     $BND2_1,$BND2_2-.,        .=$BND2_2
0000          .ENDM   $BND2
0000
0000          .MACRO  $BND     $BND_1,$BND_2
0000                  .IRP     $BND_3,<$BND_2>
0000                           $BND1    $BND_1,$BND_3
0000                  .ENDR
0000          .ENDM   $BND
0000
0000          .=0
0000     ZZ:
0000          $BND    GT,<VECTOR>
```

```
0000     MX:
0000             $BND    LT,<VECTOR>
0000     MN:
0000             .=SS
0000
0000     CASE'TYPE       INDX,#<MN-ZZ>,NMODE'<MX-MN>
0000     S:
0000             .REPT   MX-MN+1
0000             .WORD   <MX-MN>*2 + 2
0000             .ENDR
0000
0000             .=S
0000
0000             $DSP1   <<VECTOR>>
0000
0000             .=<MX-MN>*2 + S + 2
0000
0000     .ENDM   $DISPATCH
0000
```

```
                  0000   ;
                  0000   ; Overlays of IRP
                  0000   ;
                  0000           $DEFINI IRP
                  0000
        00000021  0000   . = IRP$W_FUNC+1                              ; Overlay Function word
                  0021
                  0021   $DEF      IRP$B_XDFUNC    .BLKB   1           ; DMP driver internal function code
                  0022
        00000041  0022   . = IRP$Q_STATION+1                           ; Overlay Station address
                  0041
                  0041   $DEF      IRP$B_INDEX     .BLKB   1           ; Vector index for CDB
                  0042
        00000042  0042   . = IRP$Q_STATION+2                           ; Overlay Station address
                  0042
                  0042   $DEF      IRP$W_QUOTA     .BLKW   1           ; Number of recieve buffers
                  0044
        00000044  0044   . = IRP$Q_STATION+4                           ; Overlay ditto
                  0044
                  0044   $DEF      IRP$L_CDB       .BLKL   1           ; CDB address
                  0048
        00000042  0048   . = IRP$Q_STATION+2                           ; Overlay ditto
                  0042
                  0042   $DEF      IRP$W_CSTATUS   .BLKW   1           ; Completion status
                  0044
        0000003F  0044   . = IRP$L_MEDIA+7
                  003F
                  003F   $DEF      IRP$B_POLS      .BLKB   1           ; Polling substate
                  0040
                  0040   ;
                  0040   ; Define driver internal function codes stored in IRP$B_XDFUNC of IRP.
                  0040   ; NOTE: These are not really used as bit offsets - but as values.
                  0040   ;
                  0040           _VIELD  XD_FC,0,<-                    ; Internal function codes
                  0040                   <INITT>,-                     ; Establish tributary
                  0040                   <INITC>,-                     ; Init the device
                  0040                   <STOPT>,-                     ; Shutdown tributary
                  0040                   <STOPC>,-                     ; Shutdown controller (device)
                  0040                   <CHGC>,-                      ; Change controller characteristics
                  0040                   <CHGT>,-                      ; Change trib characteristics
                  0040                   <RDMODEM>,-                   ; Read line unit modem register
                  0040                   <RDTERR>,-                    ; Read tributary error counters
                  0040                   <RDGERR>,-                    ; Read global error counters
                  0040                   <RCTERR>,-                    ; Read trib counters and clear
                  0040                   <RCGERR>,-                    ; Read global counters and clear
                  0040                   <RDTSS>,-                     ; Read tributary status slot
                  0040                   <RDGSS>,-                     ; Read global status slot
                  0040                   <WTMODEM>,-                   ; Write line unit modem register
                  0040                   <HALTT>,-                     ; DASSGN request to halt trib
                  0040                   <KILLT>,-                     ; DASSGN request to kill trib
                  0040                   <CANCEL>,-                    ; CANCEL request to halt trib
                  0040                   <XMIT>,-                      ; Transmit request
                  0040                   <RECV>,-                      ; Receive request
                  0040                   <NUMSYNC>,-                   ; Set number of sync chars
                  0040                   <POLSTAT>,-                   ; Latch polling state
                  0040                   <POLSS>,-                     ; Fetch polling sub-state
                  0040                   >                             ;
```

```
0040  392
0040  393       $DEFEND IRP                     ; End of IRP overlays
```

```
          0000    ;
          0000    ; Definitions that follow the standard UCB fields
          0000    ;
          0000
          0000            $DEFINI UCB     GLOBAL                  ; Start of UCB definitions
00000090  0000    . = UCB$C_LENGTH                                ; Position at end of UCB
          0090
          0090    $DEF    UCB$L_XD_JNLBUF  .BLKL  1               ; Pointer to the journal buffer
          0094    $DEF    UCB$Q_XD_QUEUES                         ; Message and I/O request queue heads
          0094    $DEF    UCB$Q_XD_INPUTQ  .BLKQ  1               ; Request awaiting input port
          009C    $DEF    UCB$Q_XD_XMT_REQ .BLKQ  1               ; Transmit I/O requests waiting mapping
          00A4    $DEF    UCB$Q_XD_RCV_PND .BLKQ  1               ; Receive buffers given to device
          00AC    $DEF    UCB$Q_XD_INFOUT  .BLKQ  1               ; Information Out queue
          00B4    $DEF    UCB$Q_XD_POST    .BLKQ  1               ; Requests awaiting posting
          00BC    $DEF    UCB$Q_XD_RCV_BUF .BLKQ  1               ; Free receive buffers (lookaside list)
00000006  00C4    UCB$C_XD_QUEUES = <.-UCB$Q_XD_QUEUES>/8         ; Number of queue heads
          00C4
          00C4    $DEF    UCB$L_XD_RCV_MAP .BLKL  MAX_RCV ; Receive mapping vector
          00E0    $DEF    UCB$L_XD_XMT_MAP .BLKL  MAX_XMT ; Transmit mapping vector
          00FC    $DEF    UCB$L_XD_RCV_PA  .BLKL  MAX_RCV_UV1 ; Receive mapping buffers physical
          010C                                            ; address
          010C    $DEF    UCB$L_XD_XMT_PA  .BLKL  MAX_XMT_UV1 ; Transmit mapping buffers physical
          0114                                            ; address
          0114    $DEF    UCB$L_XD_RCV_VA  .BLKL  MAX_RCV_UV1 ; Receive mapping buffers virtual
          0124                                            ; address
          0124    $DEF    UCB$L_XD_XMT_VA  .BLKL  MAX_XMT_UV1 ; Transmit mapping buffers virtual
          012C                                            ; address
          012C    $DEF    UCB$L_XD_UV1BUF  .BLKL  1               ; u-VAX I buffer area
          0130    $DEF    UCB$B_XD_RCV_MAP .BLKB  1               ; Receive mapping in use flags
          0131    $DEF    UCB$B_XD_XMT_MAP .BLKB  1               ; Transmit mapping in use flags
          0132    $DEF    UCB$B_XD_RCV_MAX .BLKB  1               ; Maximum concurrent receives
          0133    $DEF    UCB$B_XD_XMT_MAX .BLKB  1               ; Maximum concurrent transmits
          0134    $DEF    UCB$B_XD_TRB_CNT .BLKB  1               ; Number of active tributaries
          0135    $DEF    UCB$B_XD_XMT_TRB .BLKB  1               ; Maximum number of xmits per trib
          0136    $DEF    UCB$W_XD_QUOTA   .BLKW  1               ; Receive buffer quota
          0138
          0138    $DEF    UCB$L_XD_VECS                           ; Start of vector areas
          0138            ASSUME  MAX_DMP_TRIB GE MAX_DMV_TRIB ;allocate maximum slots
          0138    $DEF    UCB$B_XD_TRIB_VEC .BLKB MAX_DMP_TRIB ; Tributary address vector
          0158    $DEF    UCB$L_XD_PID_VEC .BLKL  MAX_DMP_TRIB ; PID Lookup vector
          01D8    $DEF    UCB$W_XD_CHAN_VEC .BLKW MAX_DMP_TRIB ; Channel number lookup vector
          0218    $DEF    UCB$L_XD_CDB_VEC .BLKL  MAX_DMP_TRIB ; CDB address vector
00000058  0298    UCB$C_XD_VECS = <.-UCB$L_XD_VECS>/4             ; Number of longwords in vectors
          0298
          0298    $DEF    UCB$L_XD_RUN     .BLKL  1               ; Tribs that entered Run state
          029C    $DEF    UCB$L_XD_HALT    .BLKL  1               ; Tribs that were forced to halt
          02A0    $DEF    UCB$L_XD_PID     .BLKL  1               ; Starter's PID
          02A4    $DEF    UCB$B_XD_MODE    .BLKB  1               ; Mode of operation in DMP11 format
          02A5    $DEF    UCB$B_XD_INTIM   .BLKB  1               ; Input wait timer
          02A6    $DEF    UCB$B_XD_OUTTIM  .BLKB  1               ; Output wait timer
          02A7
          02A7    $DEF    UCB$B_XD_SETPRM                         ; Start of parameter section
          02A7    $DEF    UCB$B_XD_NMS     .BLKB  1               ; Number of sync chars (0=>default)
          02A8    $DEF    UCB$B_XD_PRO     .BLKB  1               ; Protocol selection
          02A9    $DEF    UCB$B_XD_DUP     .BLKB  1               ; Duplex mode
          02AA    $DEF    UCB$B_XD_CON     .BLKB  1               ; Controller mode
          02AB    $DEF    UCB$B_XD_BFN     .BLKB  1               ; Number of receive buffers
```

```
                02AC
                02AC   $DEF     UCB$W_XD_POLLPRM                      ; Start of polling parameter section
                02AC   $DEF     UCB$W_XD_SRT      .BLKW   1           ; Streaming tributary timer
                02AE   $DEF     UCB$W_XD_SLT      .BLKW   1           ; Delta timer
                02B0   $DEF     UCB$W_XD_DDT      .BLKW   1           ; Dead timer
                02B2   $DEF     UCB$W_XD_DLT      .BLKW   1           ; Poll delay time
                02B4   $DEF     UCB$W_XD_RTT      .BLKW   1           ; Retransmit timer
      0000000F  02B6   UCB$C_XD_SETPRM = .-UCB$B_XD_SETPRM            ; Size of parameter section
                02B6
                02B6   $DEF     UCB$B_XD_FKB     .BLKB   FKB$C_LENGTH ; Fork process fork block
                02CE
                02CE   $DEF     UCB$C_XD_LENGTH                       ; Size of XDDRIVER UCB
                02CE
                02CE
      000002C6  02CE   . = UCB$B_XD_FKB+FKB$L_FR3                     ; Position at R3 of fork block
                02C6
                02C6   $DEF     UCB$L_XD_SEL0     .BLKL   1           ; Saved SEL0 and SEL2 from interrupt
                02CA   $DEF     UCB$L_XD_SEL4     .BLKL   1           ; Saved SEL4 and SEL6 from interrupt
                02CE
                02CE   ;
                02CE   ; Define device status bits
                02CE   ;
                02CE            $VIELD  UCB,0,<-                      ; XDDRIVER UCB$W_DEVSTS bits
                02CE                    <XD_INITED,,M>,-              ; Device is initialized
                02CE                    <XD_INITING,,M>,-             ; Device is initing - waiting for Timeout
                02CE                    <XD_PTP,,M>,-                 ; Device in point to point operation
                02CE                    <XD_FORK_PEND,,M>,-           ; Fork process scheduling in progress
                02CE                    <XD_DTRCLR,,M>,-              ; DTR modem signal is cleared
                02CE                    <XD_FORK_XMT,,M>,-            ; Transmit fork block in use
                02CE                    >                             ;
                02CE
                02CE            $DEFEND UCB                           ; End of UCB definitions
```

```
          0000      ;
          0000      ; Device register offsets and bit definitions
          0000      ;
          0000
          0000              $DEFINI XD                          ; Start of status definitions
          0000
          0000
          0000      $DEF    SEL0                                ; 1st word of CSR
          0000      $DEF    BSEL0                               ; Byte select 0 / Control & status
00000001  0000                              .BLKB   1
          0001
          0001      ;
          0001      ; Bit positions for device control/status register
          0001      ;
          0001
          0001              _VIELD  XD_BSEL0,0,<-               ; Control & status bits
          0001              <IEI,,M>,-                          ; Input interrupt enable
          0001              <,3>,-                              ;
          0001              <IEO,,M>,-                          ; Output interrupt enable
          0001              <,2>,-                              ;
          0001              <RQI,,M>,-                          ; Request input
          0001              >
          0001
          0001      $DEF    BSEL1                               ; Byte select 1 / Control & status
00000002  0001                              .BLKB   1
          0002
          0002              _VIELD  XD_BSEL1,0,<-               ; Control & status bits
          0002              <MAINT,,M>,-                        ; Maintenance request
          0002              <,2>,-                              ;
          0002              <LOOPB,,M>,-                        ; Loop back
          0002              <,2>,-                              ;
          0002              <MCLR,,M>,-                         ; Master clear
          0002              <RUN,,M>,-                          ; RUN state
          0002              >
          0002
          0002      $DEF    SEL2                                ; 2nd word of CSR
          0002      $DEF    BSEL2                               ; Byte select 2 / Command register
00000003  0002                              .BLKB   1
          0003
          0003              _VIELD  XD_BSEL2,0,<-               ; Command register fields
          0003              <TYPE,3,M>,-                        ; Command type code
          0003              <Q22,,M>,-                          ; Q-Bus in 22 bit mode
          0003              <RDI,,M>,-                          ; Ready input
          0003              <ERR,,M>,-                          ; ** DRIVER DEFINED error indicator **
          0003              <,1>,-                              ;
          0003              <RDO,,M>,-                          ; Ready output
          0003              >
          0003
          0003              _VIELD  XD_I,0,<-                   ; Define input command codes
          0003              <RCV>,-                             ; Receive in       - 0
          0003              <CONTROL>,-                         ; Control in       - 1
          0003              <MODE_DEF>,-                        ; Mode definition - 2
          0003              <,1>,-                              ; RESERVED
          0003              <XMIT>,-                            ; Xmit in          - 4
          0003              >
          0003
          0003      $DEF    BSEL3               .BLKB   1       ; Byte select 3 / Tributary address
```

```
          0004
          0004  $DEF    SEL4                              ; 3rd word of CSR
          0004  $DEF    BSEL4           .BLKB   1         ; Byte select 4 / Misc.
          0005
          0005  $DEF    BSEL5           .BLKB   1         ; Byte select 5 / Misc.
          0006
          0006  $DEF    SEL6                              ; 4th word of CSR
          0006  $DEF    BSEL6           .BLKB   1         ; TSS/GSS read/write requests
          0007
          0007          _VIELD XD_SEL6,0,<-               ; TSS/GSS control register
          0007          <,5>,-                            ;
          0007          <RTSS,,M>,-                       ; Read TSS
          0007          <RCTS,,M>,-                       ; Read/clear TSS
          0007          <WTSS,,M>,-                       ; Write TSS
          0007          <ECOM,,M>,-                       ; Enable common pool
          0007          <DCOM,,M>,-                       ; Disable common pool
          0007          <,2>,-                            ;
          0007          <UPOL,,M>,-                       ; Unlatch polling
          0007          <LPOL,,M>,-                       ; Latch polling
          0007          <,2>,-                            ;
          0007          >
          0007
          0007          _VIELD  XD_CI,0,<-                ; SEL6 command codes
          0007          <NOOP>,-                          ; NOOP request
          0007          <ESTAB>,-                         ; Establish tributary
          0007          <KILL>,-                          ; Kill tributary
          0007          <ISTRT>,-                         ; Start tributary
          0007          <MAINT>,-                         ; Start trib in MOP mode
          0007          <HALT>,-                          ; Halt tributary
          0007          <,10>,-                           ; RESERVED
          0007          <RDMODEM>,-                       ; Read Modem request
          0007          <WTMODEM>,-                       ; Write Modem request
          0007          <INTDIAG>-                        ; Interface diagnostics
          0007          >
          0007
          0007  $DEF    BSEL7                             ; Byte select 7 / Misc.
00000008  0007                          .BLKB   1
          0008
          0008  $DEF    SEL10           .BLKW   1         ; Byte select 10 - character count
          000A                                            ;    for DMV only
          000A
          000A          $DEFEND XD                        ; End of device register definitions
```

```
          0000  ;
          0000  ; XDDRIVER CDB definitions
          0000  ;
          0000          $DEFINI CDB      GLOBAL
00000000  0000  .=0
          0000  $DEF    CDB_L_DEVDEPEND .BLKL    1      ; Circuit dependent status
          0004  $DEF    CDB_L_AST       .BLKL    1      ; Attention AST list
          0008  $DEF    CDB_W_SIZE      .BLKW    1      ; Size of CDB
          000A  $DEF    CDB_B_TYPE      .BLKB    1      ; Type of structure
          000B  $DEF    CDB_B_POL       .BLKB    1      ; Polling state
          000C  $DEF    CDB_W_STS       .BLKW    1      ; Circuit status
          000E  $DEF    CDB_B_TRB_ADDR  .BLKB    1      ; Tributary address
          000F  $DEF    CDB_B_XMT_CNT   .BLKB    1      ; Number of xmits outstanding
          0010
          0010  $DEF    CDB_Q_QUEUES                    ; Message and I/O request queue heads
          0010  $DEF    CDB_Q_XMT_REQ   .BLKQ    1      ; Transmit requests awaiting RUN
          0018  $DEF    CDB_Q_RCV_REQ   .BLKQ    1      ; Receive I/O requests awaiting message
          0020  $DEF    CDB_Q_RCV_MSG   .BLKQ    1      ; Receive buffers containing messages
          0028  $DEF    CDB_Q_XMT_PND   .BLKQ    1      ; Transmit I/Os given to device
          0030  $DEF    CDB_Q_INFOUT    .BLKQ    1      ; Information Out queue
00000005  0038  CDB_C_QUEUES = <.-CDB_Q_QUEUES>/8      ; Number of queue heads in CDB
          0038
          0038  $DEF    CDB_L_CNTS                      ; Start of counters area of CDB
          0038  $DEF    CDB_L_BRC       .BLKL    1      ; Receive byte count
          003C  $DEF    CDB_L_BSN       .BLKL    1      ; Transmit byte count
          0040  $DEF    CDB_L_DMR       .BLKL    1      ; Receive message count
          0044  $DEF    CDB_L_DMS       .BLKL    1      ; Transmit message count
00000010  0048  CDB_C_CNTS = .-CDB_L_CNTS               ; Size of counter section of CDB
          0048
          0048  $DEF    CDB_L_PID       .BLKL    1      ; Starter's process ID
          004C  $DEF    CDB_W_CHAN      .BLKW    1      ; Channel number associated with this CDB
          004E
          004E  $DEF    CDB_B_SETPRM                    ; Start of settable parameters
          004E  $DEF    CDB_B_MRB       .BLKB    1      ; Receive buffer count
          004F  $DEF    CDB_B_MST       .BLKB    1      ; Maintenance State given by P2 buffer
          0050
          0050  $DEF    CDB_W_POLLPRM                   ; Start of Poll parameter section of CDB
          0050  $DEF    CDB_W_DEL_TIMER .BLKW    1      ; Delay timer
          0052  $DEF    CDB_B_POL_QACT  .BLKB    1      ; Active trib Q value
          0053  $DEF    CDB_B_POL_RACT  .BLKB    1      ; Active trib R value
          0054  $DEF    CDB_B_POL_QIN   .BLKB    1      ; Inactive trib Q value
          0055  $DEF    CDB_B_POL_RIN   .BLKB    1      ; Inactive triv R value
          0056  $DEF    CDB_B_POL_QDYI  .BLKB    1      ; Dying Q value
          0057  $DEF    CDB_B_POL_RDYI  .BLKB    1      ; Dying R value
          0058  $DEF    CDB_B_TO_INACT  .BLKB    1      ; Number of NDM to Inactive
          0059  $DEF    CDB_B_TO_DYING  .BLKB    1      ; Number of no responses to Dying
          005A  $DEF    CDB_B_TO_DEAD   .BLKB    1      ; Number of no responses to Dead
          005B  $DEF    CDB_B_MAX_MSGS  .BLKB    1      ; Maximum number of xmits per selection
          005C  $DEF    CDB_W_SEL_TIMER .BLKW    1      ; Selection timer
          005E  $DEF    CDB_W_BAB_TIMER .BLKW    1      ; Babbling trib timer
00000012  0060  CDB_C_SETPRM = .-CDB_B_SETPRM           ; Size of settable param section of CDB
          0060
          0060  $DEF    CDB_C_LENGTH                    ; Size of XDDRIVER CDB
          0060
          0060  ;
          0060  ; Define status bits used in CDB_B_STS
          0060  ;
```

```
0060
0060        _VIELD  CD_TS,0,<-              ; Tributary status bits for CDB_W_STS
0060                <ESTAB,,M>,-            ; Tributary established
0060                <MOP,,M>,-              ; Tributary is in MOP mode
0060                <CANCEL,,M>,-           ; Cancel I/O in progress
0060                <BUFRET,,M>,-           ; Buffers have been returned
0060                <BUFQUO,,M>,-           ; Buffer quota management needed
0060                <START,,M>,-            ; Tributary is started
0060                >                       ;
0060
0060        $DEFEND CDB
```

```
              0000     ;
              0000     ; Receive buffer definition
              0000     ;
              0000             $DEFINI RCV
    00000000  0000     .=0
              0000     $DEF    RCV_L_LINK       .BLKL   2       ; Forward and backward queue links
              0008     $DEF    RCV_W_BLKSIZE    .BLKW   1       ; Total block size
              000A     $DEF    RCV_B_BLKTYPE    .BLKB   1       ; Block type
              000B     $DEF    RCV_B_MAPSLOT    .BLKB   1       ; Mapping slot number
              000C     $DEF    RCV_L_BACC       .BLKL   1       ; Buffer address / character count
              0010     $DEF    RCV_B_BAHI       .BLKB   1       ; Buffer address hi 6 bits (Q22 mode)
              0048     $DEF    RCV_T_DATA                       ; Receive data
              0048
              0048             $DEFEND RCV
              0000
              0000     ;
              0000     ; P2 buffer header definition
              0000     ;
              0000             $DEFINI P2B
    00000000  0000     .=0
              0000     $DEF    P2B_L_POINTER    .BLKL   1       ; Pointer to start of data
              0004     $DEF    P2B_L_BUFFER     .BLKL   1       ; Address of user's data buffer
              0008     $DEF    P2B_W_SIZE       .BLKW   1       ; Size of P2 buffer
              000A     $DEF    P2B_B_TYPE       .BLKB   1       ; Type of structure
              000B     $DEF    P2B_B_SPARE      .BLKB   1       ; Spare byte
              000C     $DEF    P2B_C_LENGTH                     ; Size of P2 buffer header
              000C     $DEF    P2B_T_DATA                       ; Start of data
              000C
              000C             $DEFEND P2B
```

```
0000        .SBTTL  Standard tables
0000
0000 ;
0000 ; Driver prologue table
0000 ;
0000
0000        DPTAB   -                                         ; DPT-creation macro
0000                END=XD_END,-                              ; End of driver label
0000                ADAPTER=UBA,-                             ; Adapter type
0000                UCBSIZE=<UCB$C_XD_LENGTH>,-               ; Length of UCB
0000                NAME=XDDRIVER,-                           ; Driver name
0000
0038        DPT_STORE INIT                                    ; Start of load
0038                                                          ; initialization table
0038        DPT_STORE UCB,UCB$B_FIPL,B,8                      ; Device fork IPL
003C        DPT_STORE UCB,UCB$B_DIPL,B,21                     ; Device interrupt IPL
0040        DPT_STORE UCB,UCB$L_DEVCHAR,L,<-                  ; Device characteristics:
0040                DEV$M_NET!-                               ;   network device
0040                DEV$M_SHR!-                               ;   shared device
0040                DEV$M_AVL!-                               ;   available device
0040                DEV$M_IDV!-                               ;   input device
0040                DEV$M_ODV>                                ;   output device
0047        DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_SCOM           ; Device class - synch comm
004B        DPT_STORE UCB,UCB$B_DEVTYPE,B,DT$_DMP11           ; Device type
004F
004F        DPT_STORE REINIT                                  ; Start of reload
004F                                                          ; initialization table
004F        DPT_STORE DDB,DDB$L_DDT,D,XD$DDT                  ; Address of DDT
0054        DPT_STORE CRB,-                                   ; Address of device
0054                CRB$L_INTD+VEC$L_UNITINIT,-               ; unit initialization
0054                D,UNIT_INIT                               ; routine
0059        DPT_STORE CRB,-                                   ; Address of CSR RDI
0059                CRB$L_INTD+4,D,RDI_INTRPT                 ; interrupt routine
005E        DPT_STORE CRB,-                                   ; Address of CSR RDO
005E                CRB$L_INTD2+4,D,RDO_INTRPT                ; interrupt routine
0063        DPT_STORE UCB,UCB$L_FPC,D,FORK_PROC               ; Set up timeout vector
0068
0068        DPT_STORE END                                     ; End of initialization
0000                                                          ; tables
0000
0000 ;
0000 ; Driver dispatch table
0000 ;
0000
0000        DDTAB   -                                         ; DDT-creation macro
0000                DEVNAM=XD,-                               ; Name of device
0000                START=STARTIO,-                           ; Start I/O routine
0000                FUNCTB=XD_FUNCTABLE,-                     ; FDT address
0000                CANCEL=CANCEL,-                           ; Cancel I/O routine
0000                REGDMP=REG_DUMP,-                         ; Register dump routine
0000                DIAGBF=<36*12>,-                          ; Diagnostic buffer sizing
0000                ALTSTART=ALT_START                        ; Alternate start I/O
0038
0038 ;
0038 ; Function dispatch table
0038 ;
0038
```

```
0038  XD_FUNCTABLE:                             ; FDT for driver
0038          FUNCTAB ,-                        ; Valid I/O functions:
0038                  <READVBLK,-               ; Read virtual
0038                  READLBLK,-                ; Read logical
0038                  READPBLK,-                ; Read physical
0038                  WRITEVBLK,-               ; Write virtual
0038                  WRITELBLK,-               ; Write logical
0038                  WRITEPBLK,-               ; Write physical
0038                  SETMODE,-                 ; Set device mode
0038                  SENSEMODE,-               ; Sense mode
0038                  SETCHAR>                  ; Set device chars.
0040
0040          FUNCTAB ,-                        ; Buffered functions:
0040                  <READVBLK,-               ; Read virtual
0040                  READLBLK,-                ; Read logical
0040                  READPBLK,-                ; Read physical
0040                  WRITEVBLK,-               ; Write virtual
0040                  WRITELBLK,-               ; Write logical
0040                  WRITEPBLK,-               ; Write physical
0040                  SETMODE,-                 ; Set device mode
0040                  SENSEMODE,-               ; Sense mode
0040                  SETCHAR>                  ; Set device chars.
0048
0048          FUNCTAB RCV_FDT,-                 ; FDT read routine for
0048                  <READVBLK,-               ; read virtual,
0048                  READLBLK,-                ; read logical,
0048                  READPBLK>                 ; and read physical.
0054
0054          FUNCTAB XMT_FDT,-                 ; FDT write routine for
0054                  <WRITEVBLK,-              ; write virtual,
0054                  WRITELBLK,-               ; write logical,
0054                  WRITEPBLK>                ; and write physical.
0060
0060          FUNCTAB SETMODE_FDT,-             ; FDT set mode routine
0060                  <SETMODE,SETCHAR>         ; for set mode, set char.
006C
006C          FUNCTAB SENSEMODE_FDT,-           ; FDT sense mode routine
006C                  <SENSEMODE>               ; for sensemode
```

```
          0078          .SBTTL  P2 buffer verification tables
          0078
          0078  ;
          0078  ; Define P2 buffer verification offsets
          0078  ;
          0078          $DEFINI PARAM
          0000
          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                  <INVALID,1,M>,-                 ; Parameter invalid flags
          0000                  >
          0000
          0000          _VIELD  OFF,0,<-                        ; Offset word fields
          0000                  <VALUE,14,M>,-                  ; Offset value
          0000                  <WIDTH,2,M>,-                   ; Size of field in structure
          0000                  >
          0000
          0000          $DEFEND PARAM
          0078
          0078  ;
          0078  ; Define UCB parameters
          0078  ;
00000000  0078  LINE_PRM_BUFSIZ = 0                             ; Line parameter buffer size
          0078  LINE_PARAM:                                     ; Start of line parameters
          0078
          0078          PARAM   NMA$C_PCLI_PRO,-                ; Protocol selection
          0078                  OFFSET=UCB$B_XD_PRO,WIDTH=B,-
          0078                  MAX=NMA$C_LINPR_DMC,-
          0078                  INVALID=UCB$M_XD_INITED,-       ; Device can't be ON
          0078                  BASE=LINE
          0080
          0080          PARAM   NMA$C_PCLI_DUP,-                ; Duplex mode
          0080                  OFFSET=UCB$B_XD_DUP,WIDTH=B,-
          0080                  MAX=NMA$C_DPX_HAL,-
          0080                  INVALID=UCB$M_XD_INITED,-       ; Device can't be ON
          0080                  BASE=LINE
          0088
          0088          PARAM   NMA$C_PCLI_CON,-                ; Conroller mode
          0088                  OFFSET=UCB$B_XD_CON,WIDTH=B,-
          0088                  MAX=NMA$C_LINCN_LOO,-
          0088                  INVALID=UCB$M_XD_INITED,-       ; Device can't be ON
          0088                  BASE=LINE
          0090
          0090          PARAM   NMA$C_PCLI_BFN,-                ; Number of receives
          0090                  OFFSET=UCB$B_XD_BFN,WIDTH=B,-
          0090                  MIN=1,MAX=255,-
          0090                  INVALID=UCB$M_XD_INITED,-       ; Device can't be ON
          0090                  BASE=LINE
          009A
          009A          PARAM   NMA$C_PCLI_BUS,-                ; Buffer size
          009A                  OFFSET=UCB$W_DEVBUFSIZ,WIDTH=W,-
          009A                  MIN=1,MAX=MAX_BUFSIZ,-
          009A                  INVALID=UCB$M_XD_INITED,-       ; Device can't be ON
          009A                  BASE=LINE
          00A4
```

```
                 00A4        PARAM   NMA$C_PCLI_SLT,-                    ; Delta timer
                 00A4                OFFSET=UCB$W_XD_SLT,WIDTH=W,-
                 00A4                MIN=50,-
                 00A4                BASE=LINE
                 00AA
                 00AA        PARAM   NMA$C_PCLI_DDT,-                    ; Dead timer
                 00AA                OFFSET=UCB$W_XD_DDT,WIDTH=W,-
                 00AA                MIN=50,-
                 00AA                BASE=LINE
                 00B0
                 00B0        PARAM   NMA$C_PCLI_DLT,-                    ; Delay timer
                 00B0                OFFSET=UCB$W_XD_DLT,WIDTH=W,-
                 00B0                BASE=LINE
                 00B4
                 00B4        PARAM   NMA$C_PCLI_SRT,-                    ; Streaming trib timer
                 00B4                OFFSET=UCB$W_XD_SRT,WIDTH=W,-
                 00B4                MIN=50,-
                 00B4                BASE=LINE
                 00BA
                 00BA        PARAM   NMA$C_PCLI_NMS,-                    ; Number of sync chars
                 00BA                OFFSET=UCB$B_XD_NMS,WIDTH=B,-
                 00BA                MIN=8,MAX=255,-
                 00BA                BASE=LINE
                 00C2
                 00C2        PARAM   NMA$C_PCLI_RTT,-                    ; Retransmit timer
                 00C2                OFFSET=UCB$W_XD_RTT,WIDTH=W,-
                 00C2                MIN=50,-
                 00C2                BASE=LINE
                 00C8
                 00C8        PARAM                                       ; End of table
                 00CA  ;
                 00CA  ; Define CDB parameters
                 00CA  ;
        00000000 00CA  TRIB_PRM_BUFSIZ  = 0                              ; Trib parameter buffer size
                 00CA  TRIB_PARAM:                                       ; Start of trib parameters
                 00CA
                 00CA        PARAM   NMA$C_PCCI_TRI,-                    ; Trib address
                 00CA                OFFSET=CDB_B_TRB_ADDR,WIDTH=B,-
                 00CA                MIN=1,-
                 00CA                INVALID=XM$M_STS_ACTIVE,-           ; Trib can't be established
                 00CA                BASE=TRIB
                 00D2
                 00D2        PARAM   NMA$C_PCCI_MRB,-                    ; Maximum receive buffers
                 00D2                OFFSET=CDB_B_MRB,WIDTH=B,-
                 00D2                MIN=1,-
                 00D2                INVALID=XM$M_STS_ACTIVE,-           ; Trib can't be STARTING
                 00D2                BASE=TRIB
                 00DA
                 00DA        ASSUME  NMA$C_STATE_ON EQ 0
                 00DA        PARAM   NMA$C_PCCI_MST,-                    ; Maintenance state
                 00DA                OFFSET=CDB_B_MST,WIDTH=B,-
                 00DA                MAX=NMA$C_STATE_OFF,-
                 00DA                INVALID=XM$M_STS_ACTIVE,-           ; Trib can't be established
                 00DA                BASE=TRIB
                 00E2
                 00E2        PARAM   NMA$C_PCCI_POL,-                    ; Latch polling state
                 00E2                OFFSET=CDB_B_POL,WIDTH=B,-
```

```
00E2                      MAX=NMA$C_CIRPST_DED,-
00E2                      BASE=TRIB
00E8
00E8            PARAM     NMA$C_PCCI_TRT,-                       ; Transmit delay timer
00E8                      OFFSET=CDB_W_DEL_TIMER,WIDTH=W,-
00E8                      BASE=TRIB
00EC
00EC            PARAM     NMA$C_PCCI_ACB,-                       ; Active base
00EC                      OFFSET=CDB_B_POL_QACT,WIDTH=B,-
00EC                      BASE=TRIB
00F0
00F0            PARAM     NMA$C_PCCI_ACI,-                       ; Active increment
00F0                      OFFSET=CDB_B_POL_RACT,WIDTH=B,-
00F0                      BASE=TRIB
00F4
00F4            PARAM     NMA$C_PCCI_IAB,-                       ; Inactive base
00F4                      OFFSET=CDB_B_POL_QIN,WIDTH=B,-
00F4                      BASE=TRIB
00F8
00F8            PARAM     NMA$C_PCCI_IAI,-                       ; Inactive increment
00F8                      OFFSET=CDB_B_POL_RIN,WIDTH=B,-
00F8                      BASE=TRIB
00FC
00FC            PARAM     NMA$C_PCCI_DYB,-                       ; Dying base
00FC                      OFFSET=CDB_B_POL_QDYI,WIDTH=B,-
00FC                      BASE=TRIB
0100
0100            PARAM     NMA$C_PCCI_DYI,-                       ; Dying increment
0100                      OFFSET=CDB_B_POL_RDYI,WIDTH=B,-
0100                      BASE=TRIB
0104
0104            PARAM     NMA$C_PCCI_IAT,-                       ; Inactive threshold
0104                      OFFSET=CDB_B_TO_INACT,WIDTH=B,-
0104                      BASE=TRIB
0108
0108            PARAM     NMA$C_PCCI_DYT,-                       ; Dying threshold
0108                      OFFSET=CDB_B_TO_DYING,WIDTH=B,-
0108                      BASE=TRIB
010C
010C            PARAM     NMA$C_PCCI_DTH,-                       ; Dead threshold
010C                      OFFSET=CDB_B_TO_DEAD,WIDTH=B,-
010C                      BASE=TRIB
0110
0110            PARAM     NMA$C_PCCI_MTR,-                       ; Max abutted xmits
0110                      OFFSET=CDB_B_MAX_MSGS,WIDTH=B,-
0110                      MIN=1,-
0110                      MAX=100,-
0110                      BASE=TRIB
0118
0118            PARAM     NMA$C_PCCI_BBT,-                       ; Babbling trib timer
0118                      OFFSET=CDB_W_BAB_TIMER,WIDTH=W,-
0118                      MIN=50,-
0118                      BASE=TRIB
011E
011E            PARAM     NMA$C_PCCI_RTT,-                       ; Selection timer
011E                      OFFSET=CDB_W_SEL_TIMER,WIDTH=W,-
011E                      MIN=50,-
```

```
          011E              BASE=TRIB
          0124
          0124          PARAM                                   ; End of trib tables
          0126  ;
          0126  ; DEFAULT TRIBUTARY PARAMETERS
          0126  ;
          0126  DEF_TRIB_PARAM::
          0126          ASSUME  CDB_B_MRB EQ CDB_B_SETPRM
          0126          ASSUME  CDB_B_MST EQ CDB_B_MRB+1
          0126          ASSUME  CDB_W_DEL_TIMER EQ CDB_B_MST+1
          0126          ASSUME  CDB_B_POL_QACT EQ CDB_W_DEL_TIMER+2
          0126          ASSUME  CDB_B_POL_RACT EQ CDB_B_POL_QACT+1
          0126          ASSUME  CDB_B_POL_QIN EQ CDB_B_POL_RACT+1
          0126          ASSUME  CDB_B_POL_RIN EQ CDB_B_POL_QIN+1
          0126          ASSUME  CDB_B_POL_QDYI EQ CDB_B_POL_RIN+1
          0126          ASSUME  CDB_B_POL_RDYI EQ CDB_B_POL_QDYI+1
          0126          ASSUME  CDB_B_TO_INACT EQ CDB_B_POL_RDYI+1
          0126          ASSUME  CDB_B_TO_DYING EQ CDB_B_TO_INACT+1
          0126          ASSUME  CDB_B_TO_DEAD EQ CDB_B_TO_DYING+1
          0126          ASSUME  CDB_B_MAX_MSGS EQ CDB_B_TO_DEAD+1
          0126          ASSUME  CDB_W_SEL_TIMER EQ CDB_B_MAX_MSGS+1
          0126          ASSUME  CDB_W_BAB_TIMER EQ CDB_W_SEL_TIMER+2
      FF  0126          .BYTE   INF_RCV                         ; Default # of receive buffers
      01  0127          .BYTE   NMA$C_STATE_OFF                 ; Default maint state is OFF
    0000  0128          .WORD   0                               ; Transmit delay timer
      FF  012A          .BYTE   255                             ; Q for active
      00  012B          .BYTE   0                               ; R for active
      00  012C          .BYTE   0                               ; Q for inactive
      40  012D          .BYTE   64                              ; R for inactive
      00  012E          .BYTE   0                               ; Q for dying
      10  012F          .BYTE   16                              ; R for dying
      08  0130          .BYTE   8                               ; Inactive threshold
      02  0131          .BYTE   2                               ; Dying threshold
      03  0132          .BYTE   3                               ; Dead threshold
      04  0133          .BYTE   4                               ; Maximum abutted messages
    0BB8  0134          .WORD   3000                            ; Selection timer (dummy entry)
    1770  0136          .WORD   6000                            ; Babble timer in Msec
00000012  0138          DEF_TRIB_PARAMSZ = .-DEF_TRIB_PARAM
          0138          ASSUME  CDB_C_SETPRM EQ DEF_TRIB_PARAMSZ
          0138  ;
          0138  ; Default line parameter values
          0138  ;
          0138  DEF_LINE_PARAM::
          0138          ASSUME  UCB$B_XD_NMS EQ UCB$B_XD_SETPRM
          0138          ASSUME  UCB$B_XD_PRO EQ UCB$B_XD_NMS+1
          0138          ASSUME  UCB$B_XD_DUP EQ UCB$B_XD_PRO+1
          0138          ASSUME  UCB$B_XD_CON EQ UCB$B_XD_DUP+1
          0138          ASSUME  UCB$B_XD_BFN EQ UCB$B_XD_CON+1
          0138          ASSUME  UCB$W_XD_SRT EQ UCB$B_XD_BFN+1
          0138          ASSUME  UCB$W_XD_SLT EQ UCB$W_XD_SRT+2
          0138          ASSUME  UCB$W_XD_DDT EQ UCB$W_XD_SLT+2
          0138          ASSUME  UCB$W_XD_DLT EQ UCB$W_XD_DDT+2
          0138          ASSUME  UCB$W_XD_RTT EQ UCB$W_XD_DLT+2
      00  0138          .BYTE   0                               ; Number of sync chars (default)
      00  0139          .BYTE   NMA$C_LINPR_POI                 ; Protocol is point-point
      00  013A          .BYTE   NMA$C_DPX_FUL                   ; Duplex is full
      00  013B          .BYTE   NMA$C_LINCN_NOR                 ; Controller mode is normal
```

```
      00  013C          .BYTE   0                               ; Number of receive buffers
    1770  013D          .WORD   6000                            ; Streaming trib timer in Msec
    0032  013F          .WORD   50                              ; Delta timer in Msec
    2710  0141          .WORD   10000                           ; Dead timer
    0000  0143          .WORD   0                               ; Poll delay timer in Msec
    0BB8  0145          .WORD   3000                            ; Selection timer in Msec
0000000F  0147          DEF_LINE_PARAMSZ = .-DEF_LINE_PARAM
          0147          ASSUME  DCB$C_XD_SETPRM EQ DEF_LINE_PARAMSZ
          0147  ;
          0147  ; Polling sub-state return table
          0147  ;
          0147          ASSUME  NMA$C_CIRPST_ACT NE 0
          0147          ASSUME  NMA$C_CIRPST_INA NE 0
          0147          ASSUME  NMA$C_CIRPST_DIE NE 0
          0147          ASSUME  NMA$C_CIRPST_DED NE 0
          0147  POLSS_TBL:
      02  0147          .BYTE   NMA$C_CIRPST_ACT                ; Active
      03  0148          .BYTE   NMA$C_CIRPST_INA                ; Inactive
      00  0149          .BYTE   0                               ; Unknown
      00  014A          .BYTE   0                               ; Unknown
      04  014B          .BYTE   NMA$C_CIRPST_DIE                ; Dying
      00  014C          .BYTE   0                               ; Unknown
      00  014D          .BYTE   0                               ; Unknown
      05  014E          .BYTE   NMA$C_CIRPST_DED                ; Dead
          014F
          014F  ;
          014F  ;  Maximum buffer size allowed - assumed to be MAX_BUFSIZ (16KB), but
          014F  ;  gets overlaid if the processor is a u-VAX I to be MAX_BUFSIZ_UV1.
          014F  ;
          014F  MAX_W_BUFFER:
    3FFF  014F          .WORD   MAX_BUFSIZ                      ; Assume 16kb buffers
          0151  MAX_W_TRIB:
    0020  0151          .WORD   MAX_DMP_TRIB                    ; Assume maximum tribs for DMP11
          0153  ;
          0153  ;
          0153  ; Line counter type codes
          0153  ;
00000000  0153  LINE_CNT_BUFSIZ = 0                             ; Size of ctrl counter buffer
          0153  LINE_COUNTER:
          0153          COUNT   NMA$C_CTLIN_RPE,-               ; Remote process errors
          0153                  BITMAP=YES,WIDTH=8,BASE=LINE
          0155
          0155          COUNT   NMA$C_CTLIN_LPE,-               ; Local process errors
          0155                  BITMAP=YES,WIDTH=8,BASE=LINE
          0157
          0157  ;
          0157  ; Tributary counter type codes
          0157  ;
          0157  ; NOTE: The first set of trib counters are kept by the driver, since
          0157  ; the device only keeps 16 bit counters for these counters.
          0157  ;
00000000  0157  TRIB_CNT_BUFSIZ = 0                             ; Size of return buffer for trib cou
          0157  TRIB_COUNTER:
          0157          COUNT   NMA$C_CTCIR_BRC,-               ; Bytes received
          0157                  WIDTH=32,BASE=TRIB
          0159
          0159          COUNT   NMA$C_CTCIR_BSN,-               ; Bytes sent
```

```
0159                               WIDTH=32,BASE=TRIB
015B
015B                COUNT          NMA$C_CTCIR_DBR,-                   ; Data blocks received
015B                               WIDTH=32,BASE=TRIB
015D
015D                COUNT          NMA$C_CTCIR_DBS,-                   ; Data blocks sent
015D                               WIDTH=32,BASE=TRIB
015F
015F      TRIB_COUNTER1:
015F                COUNT          NMA$C_CTCIR_SIE,-                   ; Selection intervals elapsed
015F                               WIDTH=16,BASE=TRIB
0161
0161                COUNT          NMA$C_CTCIR_DEO,-                   ; Data errors outbound
0161                               BITMAP=YES,WIDTH=8,BASE=TRIB
0163
0163                COUNT          NMA$C_CTCIR_DEI,-                   ; Data errors inbound
0163                               BITMAP=YES,WIDTH=8,BASE=TRIB
0165
0165                COUNT          NMA$C_CTCIR_LBE,-                   ; Local buffer errors
0165                               BITMAP=YES,WIDTH=8,BASE=TRIB
0167
0167                COUNT          NMA$C_CTCIR_RBE,-                   ; Remote buffer errors
0167                               BITMAP=YES,WIDTH=8,BASE=TRIB
0169
0169                COUNT          NMA$C_CTCIR_SLT,-                   ; Selection timeouts
0169                               BITMAP=YES,WIDTH=8,BASE=TRIB
016B
016B                COUNT          NMA$C_CTCIR_LRT,-                   ; Local reply timeouts
016B                               WIDTH=8,BASE=TRIB
016D
016D                COUNT          NMA$C_CTCIR_RRT,-                   ; Remote reply timeouts
016D                               WIDTH=8,BASE=TRIB
016F
```

```
                              016F          .SBTTL  UNIT_INIT,  Unit initialization routine
                              016F
                              016F  ;++
                              016F  ; UNIT_INIT - Readies unit for I/O operations
                              016F  ;
                              016F  ; Functional description:
                              016F  ;
                              016F  ;       The operating system calls this routine after calling the
                              016F  ;       controller initialization routine:
                              016F  ;
                              016F  ;               at system startup
                              016F  ;               during driver loading
                              016F  ;               during recovery from a power failure
                              016F  ;
                              016F  ;       The unit is put online.  If power has failed, the unit will
                              016F  ;       be forced to timeout.
                              016F  ;
                              016F  ; Inputs:
                              016F  ;
                              016F  ;       R4 = CSR address
                              016F  ;       R5 = UCB address
                              016F  ;
                              016F  ; Outputs:
                              016F  ;
                              016F  ;       R0 destroyed
                              016F  ;
                              016F  ;--
                              016F
                              016F  UNIT_INIT:                                              ; Initialize unit
                    1F   BB   016F          PUSHR   #^M<R0,R1,R2,R3,R4>                     ; Save all registers
              64 A5 10   A8   0171          BISW    #UCB$M_ONLINE,UCB$W_STS(R5)             ; Set unit online
                 0B A5   90   0175          MOVB    UCB$B_FIPL(R5),-                        ; Set the fork block FIPL
               02C1 C5        0178                  UCB$B_XD_FKB+FKB$B_FIPL(R5)             ;
                    08   90   017B          MOVB    #DYN$C_FRK,-                            ; Set structure to FORK BLOCK
               02C0 C5        017D                  UCB$B_XD_FKB+FKB$B_TYPE(R5)             ;
                  226E   30   0180          BSBW    INIT_UCB                                ; Initialize the UCB
                              0183          ;
                              0183          ; For u-VAX I, we will have to allocate a physically contiguous buffer
                              0183          ; area for performing I/O on the DMV.
                              0183          ;
                              0183          CPUDISP <<790,INIT_OTHERS>,-
                              0183                   <780,INIT_OTHERS>,-
                              0183                   <750,INIT_OTHERS>,-
                              0183                   <730,INIT_OTHERS>,-
                              0183                   <UV1,INIT_UV1>>
                              019D
                              019D  INIT_UV1:
      AC AF   0400 8F    B0   019D          MOVW    #MAX_BUFSIZ_UV1,MAX_W_BUFFER            ; Set maximum size buffer
           AA AF   10    B0   01A3          MOVW    #MAX_DMV_TRIB,MAX_W_TRIB                ; Set maximum number of tribs
           41 A5   17    90   01A7          MOVB    #DT$_DMV11,UCB$B_DEVTYPE(R5)            ; Set device type to be DMV11
               012C C5   D5   01AB          TSTL    UCB$L_XD_UV1BUF(R5)                     ; Is the buffer area allocated?
                    7C   12   01AF          BNEQ    INIT_OTHERS                             ; Br if yes, continue
                    52   D4   01B1          CLRL    R2                                      ; Zero end marker
                    52   DD   01B3  10$:    PUSHL   R2                                      ; Save bad buffer on stack
      51    0000180C 8F  D0   01B5  20$:    MOVL    #UV1_BUFFER_AREA+12,R1                  ; Compute size of buffer area
                              01BC                                                          ; ..that must be contiguous
          00000000'GF    16   01BC          JSB     G^EXE$ALONONPAGED                       ; Allocate buffer
```

```
          5B 50   E9 01C2              BLBC    R0,50$                              ; Br if none available
                     01C5              ASSUME  IRP$W_SIZE LT 12
                     01C5              ASSUME  IRP$B_TYPE EQ IRP$W_SIZE+2
   0013180C 8F    D0 01C5              MOVL    #<DYN$C_BUFIO@16>!<UV1_BUFFER_AREA+12>,-; Set size and type
            08 A2    01CB                      IRP$W_SIZE(R2)                      ; of buffer
                     01CD              ;
                     01CD              ; Make sure buffer area is contiguous!
                     01CD              ;
        50    0C A2   9E 01CD          MOVAB   12(R2),R0                           ; Skip header
50   FFFFFE00 8F   CA 01D1             BICL    #-512,R0                            ; Compute BOFF of buffer area
50   000019FF 8F   C0 01D8             ADDL    #UV1_BUFFER_AREA+511,R0             ; Add in length of area that
                      01DF                                                         ;  must be contiguous & roundup
50   000001FF 8F   CA 01DF             BICL    #511,R0                             ; Round to page boundry
 50     50    F7 8F   78 01E6          ASHL    #-VA$V_VPN,R0,R0                    ; Compute number of pages-1
        51    0C A2   9E 01EB          MOVAB   12(R2),R1                           ; Get address of buffer area
              09      EF 01EF          EXTZV   S^#VA$V_VPN,-                       ; Get system page table number
     54   51  15         01F1                  S^#VA$S_VPN,R1,R4                   ;
53   00000000'GF      D0 01F4          MOVL    G^MMG$GL_SPTBASE,R3                 ; Get system page table base
      54      6344    DE 01FB          MOVAL   (R3)[R4],R4                         ; Compute PTE address
              00      EF 01FF          EXTZV   S^#PTE$V_PFN,-                      ; Get page frame number of
    53   64   15         0201                  S^#PTE$S_PFN,(R4),R3                ;  first page
         54   04      C0 0204          ADDL    #4,R4                               ; Get address of next PTE
              0D      11 0207          BRB     40$                                 ; Get into loop
                         0209  30$:
              00      EF 0209          EXTZV   S^#PTE$V_PFN,-                      ; Get page frame number of
    51   64   15         020B                  S^#PTE$S_PFN,(R4),R1                ;  next page
         54   04      C0 020E          ADDL    #4,R4                               ; Get address of next PTE
         51   53      D1 0211          CMPL    R3,R1                               ; Is buffer contiguous?
              9D      12 0214          BNEQ    10$                                 ; Br if not, try again
                         0216  40$:
              53      D6 0216          INCL    R3                                  ; Compute next page frame number
           EE 50      F5 0218          SOBGTR  R0,30$                              ; Loop if more pages
   012C C5    52      D0 021B          MOVL    R2,UCB$L_XD_UV1BUF(R5)              ; Save buffer area address
                         0220
           50 8ED0       0220  50$:    POPL    R0                                  ; Get next buffer address
              08      13 0223          BEQL    60$                                 ; Br if no more
   00000000'GF        16 0225          JSB     G^COM$DRVDEALMEM                    ; Deallocate the bad buffers
              F3      11 022B          BRB     50$                                 ; Look for more
                         022D  60$:
                         022D
                         022D  INIT_OTHERS:
  15 64 A5    05      E1 022D          BBC     #UCB$V_POWER,UCB$W_STS(R5),90$      ; Branch if not powerfail
              00      E1 0232          BBC     #UCB$V_XD_INITED,-                  ; Br if not inited
       10 68 A5          0234                  UCB$W_DEVSTS(R5),90$                ;
  01 A3    40 8F      90 0237          MOVB    #XD_BSEL1_M_MCLR,BSEL1(R3)          ; Master clear the device
  53    01    15      78 023C          ASHL    #XD_BSEL2_V_ERR+16,#1,R3            ; Indicate error
  54    01    10      78 0240          ASHL    #16,#1,R4                           ; Indicate powerfail
            165C      30 0244          BSBW    SCHED_FORK                          ; Schedule fork process
              1F      BA 0247  90$:    POPR    #^M<R0,R1,R2,R3,R4>                 ; Restore registers
              05         0249          RSB                                         ; Return
```

```
                             024A          .SBTTL  XMT_FDT,  Transmit I/O Operation FDT Routine
                             024A
                             024A   ;++
                             024A   ; XMT_FDT - Transmit I/O Operation FDT Routine
                             024A   ;
                             024A   ; Functional description:
                             024A   ;
                             024A   ; This routine is called by the SYS$QIO system service to dispatch a
                             024A   ; WRITE I/O request.
                             024A   ;
                             024A   ; The QIO parameters for WRITEs are:
                             024A   ;
                             024A   ;         P1 = address of the buffer
                             024A   ;         P2 = size of the buffer
                             024A   ;         All other parameters are unused.
                             024A   ;
                             024A   ; The buffer is validated for access and locked into memory. The I/O operation
                             024A   ; is started if possible else it is blocked temporarily on one of the various
                             024A   ; wait queues of the UCB or CDB.
                             024A   ;
                             024A   ;
                             024A   ; Inputs:
                             024A   ;
                             024A   ;         R3 = IRP address (I/O request packet)
                             024A   ;         R4 = PCB address (process control block)
                             024A   ;         R5 = UCB address (unit control block)
                             024A   ;         R6 = CCB address (channel control block)
                             024A   ;         R7 = bit number of the I/O function code
                             024A   ;
                             024A   ; Outputs:
                             024A   ;
                             024A   ;         R0 = status of transmit request initiation
                             024A   ;
                             024A   ;         R1,R2,R8,R9 are destroyed
                             024A   ;         R3-R7 are preserved.
                             024A   ;
                             024A   ;--
                             024A
                             024A   XMT_FDT:                                          ; Transmit FDT routine
            40 A3     7C     024A           CLRQ    IRP$Q_STATION(R3)                 ; Clear trib address field
            50 14     3C     024D           MOVZWL  S^#SS$_BADPARAM,R0                ; Assume bad parameters
            58 6C     D0     0250           MOVL    P1(AP),R8                         ; Get starting address of user buffer
         59 04 AC     3C     0253           MOVZWL  P2(AP),R9                         ; Get length of user buffer
               7C     13     0257           BEQL    ABORTIO                           ; Br if zero length buffer
            50 58     7D     0259           MOVQ    R8,R0                             ; Retrieve buffer parameters
       00000000'GF    16     025C           JSB     G^EXE$WRITECHK                    ; Check accessibility of user buffer
                             0262                                                     ; (No return on NO ACCESS)
                             0262                                                     ; Returns IRP$W_BCNT
    51 00000048 8F    C0     0262           ADDL2   #CXB$C_HEADER,R1                  ; Calculate length of buffer needed
               38     BB     0269           PUSHR   #^M<R3,R4,R5>                     ; Save registers
       00000000'GF    16     026B           JSB     G^EXE$BUFFRQUOTA                  ; Check if process has sufficient quota
            5F 50     E9     0271           BLBC    R0,20$                            ; Br if quota check failure
       00000000'GF    16     0274           JSB     G^EXE$ALLOCBUF                    ; Allocate CXB buffer for output
            56 50     E9     027A           BLBC    R0,20$                            ; If LBC allocation failure
            53 6E     D0     027D           MOVL    (SP),R3                           ; Retrieve address of IRP
       50 0080 C4     D0     0280           MOVL    PCB$L_JIB(R4),R0                  ; Get JIB address
         20 A0 51     C2     0285           SUBL    R1,JIB$L_BYTCNT(R0)               ; Adjust buffered I/O quota
```

```
     30 A3    51   B0 0289  1265           MOVW    R1,IRP$W_BOFF(R3)           ; Set number of bytes charged to quota
     2C A3    52   D0 028D  1266           MOVL    R2,IRP$L_SVAPTE(R3)         ; Save CXB address in IRP
                      0291  1267
                      0291  1268           ASSUME  CXB$L_FL EQ 0
                      0291  1269           ASSUME  CXB$L_BL EQ CXB$L_FL+4
     82    48 A2   9E 0291  1270           MOVAB   CXB$C_HEADER(R2),(R2)+      ; Set pointer to start of data
              82   D4 0295  1271           CLRL    (R2)+                       ; Clear next link cell
                      0297  1272
                      0297  1273           ASSUME  CXB$W_SIZE EQ CXB$L_BL+4
     82       51   B0 0297  1274           MOVW    R1,(R2)+                    ; Set size of structure
                      029A  1275
                      029A  1276           ASSUME  CXB$B_TYPE EQ CXB$W_SIZE+2
                      029A  1277           ASSUME  CXB$B_CODE EQ CXB$B_TYPE+1
     82       1B   9B 029A  1278           MOVZBW  #DYN$C_CXB,(R2)+            ; Set structure type
                      029D  1279
                      029D  1280           ASSUME  CXB$W_LENGTH EQ CXB$B_CODE+1
     52       3C   C0 029D  1281           ADDL    #CXB$C_HEADER-CXB$W_LENGTH,R2; Get address of data portion of buffer
62   68       59   28 02A0  1282           MOVC3   R9,(R8),(R2)                ; Move data to system buffer
              38   BA 02A4  1283           POPR    #^M<R3,R4,R5>               ; Restore registers
                      02A6  1284
         1C45      30 02A6  1285           BSBW    XLATE                       ; Get CDB address
         29 50     E9 02A9  1286           BLBC    R0,ABORTIO                  ; Br if error
                      02AC  1287           SETIPL  UCB$B_FIPL(R5)              ; Sync acces to UCB
                      02B0  1288
                      02B0  1290           PUSHQ   R6                          ; Save registers
     50       01   90 02B3  1291           MOVB    #JNL_XMT,R0                 ; Set journal type = XMIT
         2214      30 02B6  1292           BSBW    XD_FILL_JNX                 ; Fill the journal buffer
         04 50     E9 02B9  1293           BLBC    R0,10$                      ; Br if error
     86    32 A3   B0 02BC  1294           MOVW    IRP$W_BCNT(R3),(R6)+        ; Store byte count
                      02C0  1295 10$:      POPQ    R6                          ; Restore registers
                      02C3  1297
              54   DD 02C3  1298           PUSHL   R4                          ; Save R4
              20   10 02C5  1299           BSBB    XMT_START                   ; Start transmit operation
           54    8ED0 02C7  1300           POPL    R4                          ; Restore R4
         08 50     E9 02CA  1301           BLBC    R0,ABORTIO                  ; Br if error
   00000000'GF     17 02CD  1302           JMP     G^EXE$QIORETURN             ; Exit QIO service to await completion
                      02D3  1303
              38   BA 02D3  1304 20$:      POPR    #^M<R3,R4,R5>               ; Restore registers
                      02D5  1305
                      02D5  1306 ABORTIO:                                      ; Abort the I/O request
   00000000'GF     17 02D5  1307           JMP     G^EXE$ABORTIO               ; and exit QIO service
```

```
                        02DB          .SBTTL  XMT_START,  Start Transmit Operation
                        02DB
                        02DB  ;++
                        02DB  ; XMT_START - Start Transmit Operation
                        02DB  ;
                        02DB  ; Functional description:
                        02DB  ;
                        02DB  ; This routine is called to start a transmit operation. If the tributary
                        02DB  ; is running and there are not too many transmit requests already pending,
                        02DB  ; then the request is given to the device to transmit immediately if possible.
                        02DB  ; If there are no more mapping register slots available, or a UNIBUS mapping
                        02DB  ; register set cannot be allocated then the request is queued to the UCB xmit
                        02DB  ; wait queue.
                        02DB  ;
                        02DB  ; If the circuit is not running yet or the circuit has too many requests
                        02DB  ; pending, then the request is queued to the CDB xmit wait queue.
                        02DB  ;
                        02DB  ; Inputs:
                        02DB  ;
                        02DB  ;       R3 = IRP address
                        02DB  ;       R5 = UCB address
                        02DB  ;       R9 = CDB address
                        02DB  ;
                        02DB  ;       IPL = FIPL
                        02DB  ;
                        02DB  ; Outputs
                        02DB  ;
                        02DB  ;       R0 = status of transmit request
                        02DB  ;
                        02DB  ;       R3-R7 are preserved.
                        02DB  ;
                        02DB  ;--
                        02DB          .ENABL  LSB
                        02DB  TRIB_INACT:
  50    20D4 8F    3C   02DB          MOVZWL  #SS$_DEVINACT,R0                  ; Trib is not active
              34   11   02E0          BRB     15$                               ; Return error
                        02E2
                        02E2  TOO_BIG:
      50      14   3C   02E2          MOVZWL  #SS$_BADPARAM,R0                  ; Buffer is too large
              2F   11   02E5          BRB     15$                               ; Return error
                        02E7
                        02E7  XMT_START:                                        ; Start transmit operation
              0B   E1   02E7          BBC     #XM$V_STS_ACTIVE,-                ; Br if trib not active
           F0 69        02E9                  CDB_L_DEVDEPEND(R9),TRIB_INACT    ;
  FE5E CF  32 A3   B1   02EB          CMPW    IRP$W_BCNT(R3),MAX_W_BUFFER       ; Is buffer too large?
              EF   1A   02F1          BGTRU   TOO_BIG                           ; Br if buffer is too big
                        02F3  ;
                        02F3  ; Insert entry on CDB transmit wait queue if circuit not running or
                        02F3  ; there are too many transmit requests pending.
                        02F3  ;
     21 A3    11   90   02F3          MOVB    #XD_FC_V_XMIT,IRP$B_XDFUNC(R3)    ; Set function request in IRP
     14 B9    63   0E   02F7          INSQUE  (R3),@CDB_Q_XMT_REQ+4(R9)         ; Insert IRP on CDB xmit wait Q
                        02FB
                        02FB  XMT_ALT_START:                                    ; Alternate start of xmits
                        02FB  ;
                        02FB  ; Inputs:
                        02FB  ;       R9 = CDB address
```

```
                                02FB  ;
                   0D     E1    02FB          BBC     #XM$V_STS_RUNNING,-              ; Br if trib is not running
                1D 69           02FD                  CDB_L_DEVDEPEND(R9),30$          ;
          53    10 B9     0F    02FF          REMQUE  @CDB_Q_XMT_REQ(R9),R3            ; Get oldest xmit request
                   17     1D    0303          BVS     30$                              ; Br if none - process others
                0F A9     97    0305          DECB    CDB_B_XMT_CNT(R9)                ; One less xmit possible for this ci
                   0D     18    0308          BGEQ    20$                              ; Br if we can handle this request
                0F A9     96    030A          INCB    CDB_B_XMT_CNT(R9)                ; Else, set count back to zero
          10 A9    63     0E    030D          INSQUE  (R3),CDB_Q_XMT_REQ(R9)           ; Re-insert on front of queue
                   09     11    0311          BRB     30$                              ; Try to start waiting xmits
                                0313
          50       01     9A    0313  10$:    MOVZBL  S^#SS$_NORMAL,R0                 ; Return success
                          05    0316  15$:    RSB
                                0317
        00A0 D5    63     0E    0317  20$:    INSQUE  (R3),@UCB$Q_XD_XMT_REQ+4(R5)     ; Insert request on UCB wait Q
                                031C  ;
                                031C  ; Find a free mapping register slot.  If none currently available, put the
                                031C  ; I/O request in the CCB wait queue.
                                031C  ;
                   05     E0    031C  30$:    BBS     #UCB$V_XD_FORK_XMT,-             ; Br if XMT fork block in use
             F2 68 A5           031E                  UCB$W_DEVSTS(R5),10$             ;
          54     0133 C5  9A    0321          MOVZBL  UCB$B_XD_XMT_MAX(R5),R4          ; Get max concurrent transmits
 54   0131 C5    54       00 EB 0326          FFC     #0,R4,UCB$B_XD_XMT_MAP(R5),R4    ; Find a free transmit slot
                   E4     13    032D          BEQL    10$                              ; Br if none free
                                032F  ;
                                032F  ; Store IRP info in UCB
                                032F  ;
          53   009C D5    0F    032F          REMQUE  @UCB$Q_XD_XMT_REQ(R5),R3         ; Get oldest xmit request
                   DD     1D    0334          BVS     10$                              ; Br if none, leave
                                0336
                                0336          SETBIT  R4,UCB$B_XD_XMT_MAP(R5)          ; Set mapping slot in use flag
          3C A3    54     90    033C          MOVB    R4,IRP$L_MEDIA+4(R3)             ; Save mapping slot number used
                                0340  ;
                                0340  ; Skip MAP REGISTER useage if u-VAX I.
                                0340  ;
                                0340          CPUDISP <<790,XMT_OTHERS>,-
                                0340                   <780,XMT_OTHERS>,-
                                0340                   <750,XMT_OTHERS>,-
                                0340                   <730,XMT_OTHERS>,-
                                0340                   <UV1,XMT_UV1>>
                                035A
                                035A  XMT_UV1:
    3A A3    32 A3        B0    035A          MOVW    IRP$W_BCNT(R3),IRP$L_MEDIA+2(R3); Save BCNT in device format
     50    010C C544      D0    035F          MOVL    UCB$L_XD_XMT_PA(R5)[R4],R0       ; Get buffer Physical address
        38 A3      50     B0    0365          MOVW    R0,IRP$L_MEDIA(R3)               ; Store BA00-BA15
    50  50      F0 8F     78    0369          ASHL    #-16,R0,R0                       ; Shift down high byte
        3D A3      50     90    036E          MOVB    R0,IRP$L_MEDIA+5(R3)             ; Store BA16-BA21
                                0372          ;
                                0372          ; Copy the data to the contiguous buffer.
                                0372          ;
     51    0124 C544      D0    0372          MOVL    UCB$L_XD_XMT_VA(R5)[R4],R1       ; Get contiguous buffer's VA
        52      2C B3     D0    0378          MOVL    @IRP$L_SVAPTE(R3),R2             ; Get system buffer's VA
                   38     BB    037C          PUSHR   #^M<R3,R4,R5>
    61  62      32 A3     28    037E          MOVC3   IRP$W_BCNT(R3),(R2),(R1)         ; Do the copy
                   38     BA    0383          POPR    #^M<R3,R4,R5>
                 0076     31    0385          BRW     90$                              ; Continue
                                0388
```

```
                        0388  XMT_OTHERS:
    7E A5    32 A3   B0 0388          MOVW    IRP$W_BCNT(R3),UCB$W_BCNT(R5)     ; Set buffer count
       50    2C B3   D0 038D          MOVL    @IRP$L_SVAPTE(R3),R0              ; Get buffer virtual address
          FE00 8F    AB 0391          BICW3   #^C<VA$M_BYTE>,-                  ; Compute BOFF
       7C A5    50      0395                   R0,UCB$W_BOFF(R5)                ;
  50  50    15  09   EF 0398          EXTZV   #VA$V_VPN,#VA$S_VPN,R0,R0         ; Get virtual page number
    51   00000000'GF D0 039D          MOVL    G^MMG$GL_SPTBASE,R1               ; Get the base address of SPTEs
     78 A5    6140   DE 03A4          MOVAL   (R1)[R0],UCB$L_SVAPTE(R5)         ; Set address of the SPTE
                        03A9  ;
                        03A9  ; Allocate UNIBUS map registers
                        03A9  ;
                20   A8 03A9          BISW    #UCB$M_XD_FORK_XMT,-              ; Assume we will have to wait
             68 A5      03AB                  UCB$W_DEVSTS(R5)                  ; ..for fork to run
        000003BB'EF  9F 03AD          PUSHAB  40$                               ; Push address of fork
        00000000'GF  17 03B3          JMP     G^IOC$REQMAPREG                   ; Request some map registers
                        03B9  ;
                        03B9  ; The following code may be executed as a fork process, therefore we have
                        03B9  ; to setup a timeout service routine offset.
                        03B9  ;
             20CB'      03B9          .WORD   TIMEOUT-.                         ; Timeout service routine
                20   AA 03BB  40$:    BICW    #UCB$M_XD_FORK_XMT,-              ; Fork block is no longer in
             68 A5      03BD                  UCB$W_DEVSTS(R5)                  ; ..use
                00   E0 03BF          BBS     #CD_TS_V_ESTAB,-                  ; Br if trib STILL established
          0C 0C A9      03C1                  CDB_W_STS(R9),50$                 ;
                        03C4          RELMPR                                    ; Else, release map registers
          50    2C   3C 03CA          MOVZWL  #SS$_ABORT,R0                     ; Return request in error
              1800   31 03CD          BRW     IO_DONE                           ; Complete the I/O request
                        03D0
                57   DD 03D0  50$:    PUSHL   R7                                ; Save R7
       57    24 A5   D0 03D2          MOVL    UCB$L_CRB(R5),R7                  ; Get CRB address
                        03D6          ASSUME  VEC$W_MAPREG+2 EQ VEC$B_NUMREG
                        03D6          ASSUME  VEC$B_NUMREG+1 EQ VEC$B_DATAPATH
             34 A7   D0 03D6          MOVL    CRB$L_INTD+VEC$W_MAPREG(R7),-     ; Save mapping info
         00E0 C544      03D9                  UCB$L_XD_XMT_MAP(R5)[R4]          ;
                        03DD  ;
                        03DD  ; Map the buffer
                        03DD  ;
                        03DD          ASSUME  UCB$W_BOFF+2 EQ UCB$W_BCNT
    38 A3    7C A5   D0 03DD          MOVL    UCB$W_BOFF(R5),IRP$L_MEDIA(R3)    ; Move byte offset and size
             34 A7   F0 03E2          INSV    CRB$L_INTD+VEC$W_MAPREG(R7),-     ; Insert BA9-BA15
   38 A3    07  09      03E5                  #9,#7,IRP$L_MEDIA(R3)
 50  34 A7  06  07   EF 03E9          EXTZV   #7,#6,CRB$L_INTD+VEC$W_MAPREG(R7),R0 ; Get BA16-BA17
38 A3   02  1E  50   F0 03EF          INSV    R0,#30,#2,IRP$L_MEDIA(R3)         ; Insert BA16-BA17
        00000000'GF  16 03F5          JSB     G^IOC$LOADUBAMAPA                 ; Loadmap registers
              57 8ED0   03FB          POPL    R7                                ; Restore R7
                        03FE
                        03FE  90$:
                        03FE  ;
                        03FE  ; Request and load the port with the buffer address and size, and return
                        03FE  ;
       0098 D5   63  0E 03FE          INSQUE  (R3),@UCB$Q_XD_INPUTQ+4(R5)       ; Insert transmit request at end of
                        0403                                                    ;       UCB input wait queue.
                59   DD 0403          PUSHL   R9                                ; Save CDB address
                        0405          DSBINT  UCB$B_DIPL(R5)                    ; Sync access to device
              0BC1   30 040C          BSBW    LOAD_PORT                         ; Load port
                        040F          ENBINT                                    ; Restore IPL
              59 8ED0   0412          POPL    R9                                ; Restore CDB address
```

```
FEE3   31  0415  1480         BRW     XMT_ALT_START                     ; Lets try it again
           0418  1481         .DSABL  LSB
```

```
                         0418         .SBTTL  RCV_FDT,  Read I/O Operation FDT Routine
                         0418
                         0418   ;++
                         0418   ; RCV_FDT - Read I/O Operation FDT Routine
                         0418   ;
                         0418   ; Functional description:
                         0418   ;
                         0418   ; This routine is called by the SYS$QIO system service to dispatch a
                         0418   ; READ I/O request.
                         0418   ;
                         0418   ; The QIO parameters for READs are:
                         0418   ;
                         0418   ;       P1 = address of the buffer
                         0418   ;       P2 = size of the buffer
                         0418   ;       All other parameters are unused.
                         0418   ;
                         0418   ; The specified buffer is checked for accessibility. The buffer address and
                         0418   ; count are saved in the packet. Then IPL is raised to device fork IPL and if
                         0418   ; a message is available the operation is complete. Otherwise the packet is
                         0418   ; queued onto the waiting receive list of the CDB.
                         0418   ;
                         0418   ; If the function specifies a modifier of IO$M_NOW, if no message is
                         0418   ; available when the test is made, a status of SS$_ENDOFFILE is
                         0418   ; returned.
                         0418   ;
                         0418   ; Inputs:
                         0418   ;
                         0418   ;       R3 = I/O Packet Address
                         0418   ;       R4 = PCB Address
                         0418   ;       R5 = UCB Address
                         0418   ;       R6 = CCB Address
                         0418   ;       R7 = Function Code
                         0418   ;       AP = Address of first operation-specific QIO parameter
                         0418   ;
                         0418   ; Outputs:
                         0418   ;
                         0418   ;       R0 = status of the receive QIO operation
                         0418   ;       R3-R7 - unchanged
                         0418   ;
                         0418   ;--
                         0418
                         0418   RCV_FDT:                                        ; Read FDT process routine
        40 A3      7C    0418           CLRQ    IRP$Q_STATION(R3)               ; Clear tributary address field
                         041B   ;
                         041B   ; Check request parameters
                         041B   ;
     50    14      3C    041B           MOVZWL  S^#SS$_BADPARAM,R0              ; Assume bad parameters
  51    04 AC      3C    041E           MOVZWL  P2(AP),R1                       ; Get length of buffer
           31      13    0422           BEQL    10$                             ; Br if zero - abort I/O
     50    6C      D0    0424           MOVL    P1(AP),R0                       ; Get buffer address
   38 A3   50      D0    0427           MOVL    R0,IRP$L_MEDIA(R3)              ; Save buffer addr in IRP
        30 A3      B4    042B           CLRW    IRP$W_BOFF(R3)                  ; No quota to return during completi
  00000000'GF      16    042E           JSB     G^EXE$READCHK                   ; Check accessibility
                         0434                                                   ; (No return on no access)
         1AB7      30    0434           BSBW    XLATE                           ; Get CDB address
        1B 50      E9    0437           BLBC    R0,10$                          ; Br if error
                         043A           SETIPL  UCB$B_FIPL(R5)                  ; Raise IPL to fork level
```

```
                    043E  1540                               ;  to lock the data base
                    043E  1542          PUSHQ   R6           ; Save registers
   50  02     90    0441  1543          MOVB    #JNL_RCV,R0  ; Set journal type = RECV
     2086     30    0444  1544          BSBW    XD_FILL_JNX  ; Fill the journal buffer
                    0447  1545          POPQ    R6           ; Restore registers
                    044A  1547
       0C     10    044A  1548          BSBB    RCV_START    ; Start receive operation
    06 50     E9    044C  1549          BLBC    R0,10$       ; Br if error
00000000'GF   17    044F  1550          JMP     G^EXE$QIORETURN ; Return to await completion
                    0455  1551
     FE7D     31    0455  1552 10$:     BRW     ABORTIO      ; Abort the I/O request
```

```
                        0458          .SBTTL  RCV_START,  Start Receive Operation
                        0458
                        0458  ;++
                        0458  ; RCV_START - Start Receive Operation
                        0458  ;
                        0458  ; Functional description:
                        0458  ;
                        0458  ; At this point, the buffer is okay and we are now at fork IPL.  Now check
                        0458  ; to make sure the tributary has been established.  If not, we will return
                        0458  ; an SS$_DEVOFFLINE error status.  Note that the circuit need not be turned
                        0458  ; on in order to do a read request.
                        0458  ;
                        0458  ; Inputs:
                        0458  ;
                        0458  ;       R3 = IRP address
                        0458  ;       R5 = UCB address
                        0458  ;       R9 = CDB address
                        0458  ;
                        0458  ;       IPL = FIPL
                        0458  ;
                        0458  ; Outputs:
                        0458  ;
                        0458  ;       R0 = return status of receive request
                        0458  ;       R3-R7 are preserved.
                        0458  ;
                        0458  ;--
                        0458          .ENABL  LSB
                        0458  RCV_START:
 50   20D4 8F      3C   0458          MOVZWL  #SS$_DEVINACT,R0                     ; Assume trib not active
            0B     E1   045D          BBC     #XM$V_STS_ACTIVE,-                   ; Br if trib not active
         4C 69          045F                  CDB_L_DEVDEPEND(R9),40$              ;
   0C A9   04      E1   0461          BBC     #CD_TS_V_BUFQUO,CDB_W_STS(R9),-      ; Br if no buffer management
           26           0465                  RCV_START_ALT                        ;
   21 A3   12      90   0466          MOVB    #XD_FC_V_RECV,IRP$B_XDFUNC(R3)       ; Set function request in IRP
09 20 A3   06      E1   046A          BBC     #IO$V_NOW,IRP$W_FUNC(R3),5$          ; Br if not READ NOW
   52   20 A9      9E   046F          MOVAB   CDB_Q_RCV_MSG(R9),R2                 ; Get address of received msgs
      52   62      D1   0473          CMPL    (R2),R2                              ; Is there a message?
           24      13   0476          BEQL    15$                                  ; Br if no - return EOF
                        0478  5$:     DSBINT  UCB$B_DIPL(R5)                       ; Sync access to device
 0098 D5   63      0E   047F          INSQUE  (R3),@UCB$Q_XD_INPUTQ+4(R5)          ; Insert at end of input Q
         0B49      30   0484          BSBW    LOAD_PORT                            ; Bump quota for trib
                        0487          ENBINT                                       ; Restore IPL
           1E      11   048A          BRB     30$                                  ; Return with success
                        048C
                        048C  RCV_START_ALT:                                       ; Alternate receive startup
                        048C  ;
                        048C  ;  Check to see if message is available
                        048C  ;
   52   20 B9      0F   048C          REMQUE  @CDB_Q_RCV_MSG(R9),R2                ; Dequeue a received message
           05      1D   0490          BVS     10$                                  ; Br if none
                        0492  ;
                        0492  ; Complete the receive with available message
                        0492  ;
         16FE      30   0492          BSBW    FINISH_RCV_IO                        ; Complete the I/O request
           13      11   0495          BRB     30$                                  ; and exit
                        0497  ;
                        0497  ; No message available.  If IO$M_NOW specified, return with SS$_ENDOFFILE
```

```
                        0497  1611 ; status. Otherwise, queue IRP to await arrival of message and continue.
                        0497  1612 ;
 0A 20 A3    06    E1   0497  1613 10$:     BBC     #IO$V_NOW,IRP$W_FUNC(R3),20$     ; Branch if not READ NOW
 50    0870 8F     3C   049C  1614 15$:     MOVZWL  #SS$_ENDOFFILE,R0                ; Set no message status
           172C    30   04A1  1615          BSBW    IO_DONE                          ; Complete the I/O
             04    11   04A4  1616          BRB     30$                              ; and exit
                        04A6  1617 ;
                        04A6  1618 ; Queue the IRP to CDB receive IRP wait queue
                        04A6  1619 ;
    1C B9    63    0E   04A6  1620 20$:     INSQUE  (R3),@CDB_Q_RCV_REQ+4(R9)        ; Queue IRP to await message
       50    01    9A   04AA  1621 30$:     MOVZBL  S^#SS$_NORMAL,R0                 ; Set successful return
                   05   04AD  1622 40$:     RSB                                      ; Return
                        04AE  1623          .DSABL  LSB
```

```
                      04AE        .SBTTL  ALT_START,  Alternate Start I/O Routine
                      04AE
                      04AE  ;++
                      04AE  ; ALT_START - Alternate Start I/O Routine
                      04AE  ;
                      04AE  ; Functional description:
                      04AE  ;
                      04AE  ; This entry point is used by the "internal IRP" interface.  IRP's
                      04AE  ; which enter the driver at this point have not been pre-processed
                      04AE  ; by our FDT routines.  These IRP's have been built by another
                      04AE  ; driver and look exactly like a regular IRP.
                      04AE  ;
                      04AE  ; This routine, therefore, must do the same processing which the FDT
                      04AE  ; routines would do if they had been called as in the case of a normal
                      04AE  ; IRP.
                      04AE  ;
                      04AE  ; NOTE: The PID and CHAN fields of the IRP are sufficient to map to a
                      04AE  ; CDB.
                      04AE  ;
                      04AE  ; Inputs:
                      04AE  ;
                      04AE  ;       R3 - IRP address
                      04AE  ;       R5 - UCB address
                      04AE  ;
                      04AE  ;       All pertinent fields of the IRP are assumed to be valid.
                      04AE  ;
                      04AE  ;       IPL = FIPL
                      04AE  ;
                      04AE  ; Outputs:
                      04AE  ;
                      04AE  ;       R0 = status of the transmit/receive request.
                      04AE  ;
                      04AE  ;       R3-R5 are preserved.
                      04AE  ;
                      04AE  ;--
                      04AE
                      04AE  ALT_START:
                59  DD 04AE         PUSHL   R9                                 ; Save R9
             40 A3  7C 04B0         CLRQ    IRP$Q_STATION(R3)                  ; Clear station field
             28 A3  B5 04B3         TSTW    IRP$W_CHAN(R3)                     ;****temporary****
                05  19 04B6         BLSS    3$                                 ;****temporary****
      28 A3  28 A3  AE 04B8         MNEGW   IRP$W_CHAN(R3),IRP$W_CHAN(R3)      ;****temporary****
       50  02A0 C5  D0 04BD  3$:    MOVL    UCB$L_XD_PID(R5),R0                ; Use starter's PID
              1A2D  30 04C2         BSBW    XLATE_ALT                          ; Get CDB address
             38 50  E9 04C5         BLBC    R0,30$                             ; Br if error
   18 2A A3     01  E0 04C8         BBS     #IRP$V_FUNC,IRP$W_STS(R3),10$      ; Br if read request
                       04CD
                       04CD         PUSHQ   R6                                 ; Save registers
          50    01  90 04D0         MOVB    #JNL_XMT,R0                        ; Set journal type = XMIT
              1FF7  30 04D3         BSBW    XD_FILL_JNX                        ; Fill the journal buffer
             04 50  E9 04D6         BLBC    R0,5$                              ; Br if error
       86    32 A3  B0 04D9         MOVW    IRP$W_BCNT(R3),(R6)+               ; Store byte count
                       04DD  5$:    POPQ    R6                                 ; Restore registers
                       04E0
              FE04  30 04E0         BSBW    XMT_START                          ; Initiate the transmit
                1B  11 04E3         BRB     30$                                ; Return
                       04E5
```

```
                    04E5      10$:
                    04E5               PUSHQ   R6                        ; Save registers
      50    02  90  04E8               MOVB    #JNL_RCV,R0               ; Set journal type = RECV
          1FDF  30  04EB               BSBW    XD_FILL_JNX               ; Fill the journal buffer
                    04EE               POPQ    R6                        ; Restore registers
                    04F1
  52     2C A3  D0  04F1               MOVL    IRP$L_SVAPTE(R3),R2       ; Get address of input buffer
            06  13  04F5               BEQL    20$                       ; Br if none
         2C A3  D4  04F7               CLRL    IRP$L_SVAPTE(R3)          ; Make sure SVAPTE is cleared
          09E3  30  04FA               BSBW    ADDRCVLIST                ; Else, add it to the receive list
          FF58  30  04FD      20$:     BSBW    RCV_START                 ; Initiate the read
         59   8ED0  0500      30$:     POPL    R9                        ; Restore R9
         01 50  E9  0503               BLBC    R0,40$                    ; Branch if error
                05  0506               RSB                               ; Return to caller
                    0507
          16C6  31  0507      40$:     BRW     IO_DONE                   ; Post the I/O request in error
```

```
050A          .SBTTL  SETMODE_FDT,  Set mode I/O operation FDT routine
050A
050A    ;++
050A    ; SETMODE_FDT - Set mode I/O operation FDT routine
050A    ;
050A    ; Functional description:
050A    ;
050A    ; This routine is used to set the configuration of the DMP11 hardware
050A    ; device.  Subfunction modifier bits are used to specify the type of
050A    ; action to be taken.  The two characteristics buffers (P1 and P2)
050A    ; are used to describe specific characteristics.
050A    ;
050A    ; The QIO parameters for SETMODE are:
050A    ;
050A    ;       P1 = Optional address of quadword or longword buffer
050A    ;       P2 = Optional address of buffer descriptor for extended characteristics
050A    ;       P3 = Number of receive buffers to pre-allocate. Required on
050A    ;            controller startup.
050A    ;
050A    ;
050A    ; The subfunction modifiers are as follows:
050A    ;
050A    ;
050A    ;        o STARTUP - establish a tributary - this modifier is used to
050A    ;               establish a tributary so that communication between
050A    ;               a unit and the DMP11 device can occur.  Polling
050A    ;               parameters may also be modified with this command
050A    ;               either at startup time or at any subsequent point
050A    ;               in time.
050A    ;
050A    ;       o SHUTDOWN - shutdown the assigned tributary - this modifier
050A    ;               is used to stop the polling and delete the tributary
050A    ;               by the user of that tributary.  Only the assigned
050A    ;               tributary may be shutdown and this command does not
050A    ;               require privilege.
050A    ;
050A    ;       o ATTNAST - request an attention AST - this modifier is used
050A    ;               to set up an AST to be delivered when a change of
050A    ;               status occurs on this tributary.
050A    ;
050A    ;       o CTRL - perform the request on the Controller not the
050A    ;                tributary.
050A    ;
050A    ;       o SET_MODEM - set line unit's mode register.
050A    ;
050A    ; Inputs:
050A    ;
050A    ;       R3 = IRP address
050A    ;       R4 = PCB address
050A    ;       R5 = UCB address
050A    ;       R6 = CCB address
050A    ;       R7 = Function code
050A    ;       AP = address of first QIO parameter
050A    ;
050A    ; Outputs:
050A    ;
050A    ;       R0 = status of setmode request
```

```
                       050A  ;
                       050A  ;         R3-R5 are preserved.
                       050A  ;
                       050A  ;         R7-R9 = destroyed
                       050A  ;
                       050A  ;--
                       050A
                       050A  SETMODE_FDT:                                          ; Setmode FDT processing
        40 A3    7C    050A          CLRQ    IRP$Q_STATION(R3)                     ; Clean up station field
                       050D
                       050D          PUSHQ   R6                                    ; Save registers
     50    03    90    0510          MOVB    #JNL_QIO,R0                           ; Set journal type = QIO
         1FB7    30    0513          BSBW    XD_FILL_JNX                           ; Fill the journal buffer
        04 50    E9    0516          BLBC    R0,1$                                 ; Br if error
     F8 A6  6C   D0    0519          MOVL    P1(AP),-8(R6)                         ; Store the P1 parameter
                       051D  1$:     POPQ    R6                                    ; Restore registers
                       0520
     57 20 A3    3C    0520          MOVZWL  IRP$W_FUNC(R3),R7                     ; Get entire function code
  04 57    0A    E0    0524          BBS     #IO$V_SET_MODEM,R7,3$                 ; Br if SET_MODEM request
  03 57    09    E1    0528          BBC     #IO$V_CTRL,R7,5$                      ; Br if not controller request
         020A    31    052C  3$:     BRW     SETMODE_CTRL                          ; Process controller request
                       052F  ;
                       052F  ; Perform setmode request on a tributary
                       052F  ;
  36 57    08    E1    052F  5$:     BBC     #IO$V_ATTNAST,R7,30$                  ; Branch if not attention AST
                       0533
                       0533  ;
                       0533  ; User is requesting an attention AST.
                       0533  ;
         19B8    30    0533          BSBW    XLATE                                 ; Get CDB address
        33 50    E9    0536          BLBC    R0,49$                                ; Br if error - abort I/O
     57 04 A9    DE    0539          MOVAL   CDB_L_AST(R9),R7                      ; Get addr of AST list
    00000000'GF  16    053D          JSB     G^COM$SETATTNAST                      ; Set up attention AST
  11 0C A9   00  E1    0543          BBC     #CD_TS_V_ESTAB,CDB_W_STS(R9),20$      ; Br if tributary not estab
     51 20 A9    9E    0548          MOVAB   CDB_Q_RCV_MSG(R9),R1                  ; Check for empty rcv list
        61 51    D1    054C          CMPL    R1,(R1)                               ; Empty?
           08    13    054F          BEQL    20$                                   ; Yes, no need to inform user
           53    DD    0551  10$:    PUSHL   R3                                    ; Else, save IRP address
         1D1A    30    0553          BSBW    POKE_USER                             ; Inform the user
         53    8ED0    0556          POPL    R3                                    ; Restore IRP address
        51 69    D0    0559  20$:    MOVL    CDB_L_DEVDEPEND(R9),R1                ; Get CDB dependent status
     51 44 A5    C8    055C          BISL    UCB$L_DEVDEPEND(R5),R1                ; OR in UCB status
     50    01    3C    0560  23$:    MOVZWL  S^#SS$_NORMAL,R0                      ; Set success
    00000000'GF  17    0563  25$:    JMP     G^EXE$FINISHIO                        ; Complete the I/O
                       0569
                       0569
         02FE    30    0569  30$:    BSBW    GET_CHAR_BUFS                         ; Get P1 and P2 characteristics
        30 50    E9    056C  49$:    BLBC    R0,40$                                ; Br if error - abort I/O
  2F 57    07    E1    056F          BBC     #IO$V_SHUTDOWN,R7,50$                 ; Branch if not trib shutdown
                       0573  ;
                       0573  ; Shutdown tributary modifier specified.
                       0573  ;
                       0573  ; Validate P2 buffer. Try to find CDB, if found then update CDB and stop
                       0573  ; tributary.
                       0573  ;
         1978    30    0573          BSBW    XLATE                                 ; Get CDB address
           52    D4    0576          CLRL    R2                                    ; Assume no CDB present
```

```
          04 50   E9  0578  1819         BLBC    R0,33$                              ; Br if not CDB present
    52    0C A9   3C  057B  1820         MOVZWL  CDB_W_STS(R9),R2                    ; Else, get status
           1B07   30  057F  1821 33$:    BSBW    VALIDATE_P2_CDB                     ; Validate the P2 buffer
          DE 50   E9  0582  1822         BLBC    R0,25$                              ; Br if error
    51    44 A5   D0  0585  1823         MOVL    UCB$L_DEVDEPEND(R5),R1              ; Assume no CDB
             59   D5  0589  1824         TSTL    R9                                  ; Was CDB found?
             D3   13  058B  1825         BEQL    23$                                 ; No - leave now
           1AA9   30  058D  1826         BSBW    CHG_CDB                             ; Change CDB parameters
  C4 0C A9   00   E1  0590  1827         BBC     #CD_TS_V_ESTAB,CDB_W_STS(R9),20$ ; Br if nothing to delete
     21 A3   02   90  0595  1828         MOVB    S^#XD_FC_V_STOPT,IRP$B_XDFUNC(R3) ; Set internal function code
           0522   31  0599  1829         BRW     QUEPKT                              ; Queue packet to driver
                      059C  1830
        53 8ED0       059C  1831 37$:    POPL    R3                                  ; Restore R3
           FD33   31  059F  1832 40$:    BRW     ABORTIO                             ; Abort the I/O request
                      05A2  1833 ;
                      05A2  1834 ; Startup tributary modifier specified or no modifier
                      05A2  1835 ;
           1949   30  05A2  1836 50$:    BSBW    XLATE                               ; Get CDB address
          03 50   E9  05A5  1837         BLBC    R0,55$                              ; Br if CDB address not found
           0166   31  05A8  1838         BRW     150$                                ; Else, change existing trib
                      05AB  1839 ;
                      05AB  1840 ; We are creating a new tributary.
                      05AB  1841 ;
    50  20D4 8F   3C  05AB  1842 55$:    MOVZWL  #SS$_DEVINACT,R0                    ; Assume ctrl not active
             00   E1  05B0  1843         BBC     #UCB$V_XD_INITED,-                  ; Br if ctrl not inited
       EA 68 A5       05B2  1844                 UCB$W_DEVSTS(R5),40$                ;
                      05B5  1845 ;
                      05B5  1846 ; Allow only 1 trib for point to point or tributary mode
                      05B5  1847 ;
  0E 68 A5   02   E1  05B5  1848         BBC     #UCB$V_XD_PTP,UCB$W_DEVSTS(R5),60$ ; Br if not point to point
                      05BA  1849 ;&&&    BBS     #XM$V_CHR_CTRL,-                    ; Br if control station
                      05BA  1850 ;&&&            UCB$L_DEVDEPEND(R5),60$             ;
02A0 C5   0C A3   D1  05BA  1851         CMPL    IRP$L_PID(R3),UCB$L_XD_PID(R5)      ; Make sure same PID
             0C   12  05C0  1852         BNEQ    70$                                 ; Br if not - sneaky devil
        0134 C5   95  05C2  1853         TSTB    UCB$B_XD_TRB_CNT(R5)                ; Any tribs yet?
             06   12  05C6  1854         BNEQ    70$                                 ; Br if trib already exists
                      05C8  1855 ;
                      05C8  1856 ; Find an empty tributary slot in UCB
                      05C8  1857 ;
           18CC   30  05C8  1858 60$:    BSBW    FIND_SLOT                           ; Find an empty slot
          07 50   E8  05CB  1859         BLBS    R0,80$                              ; Br if found one
    50  0850 8F   3C  05CE  1860 70$:    MOVZWL  #SS$_DEVICEFULL,R0                  ; Return error
             CA   11  05D3  1861         BRB     40$                                 ; Abort I/O
                      05D5  1862 ;
                      05D5  1863 ; Found empty slot in UCB translation vector, validate P2 and allocate a CDB
                      05D5  1864 ;
             52   D4  05D5  1865 80$:    CLRL    R2                                  ; No status flags yet
             59   D4  05D7  1866         CLRL    R9                                  ; No CDB address - Yet
           1AAD   30  05D9  1867         BSBW    VALIDATE_P2_CDB                     ; Validate the P2 buffer
          19 50   E9  05DC  1868         BLBC    R0,83$                              ; Br if error
                      05DF  1869 ;
                      05DF  1870 ; Make sure trib address given if startup request.
                      05DF  1871 ; We make this requirement, because this is a new CDB and the tributary
                      05DF  1872 ; address is required for all modes except Point to Point.
                      05DF  1873 ;
     18 57   06   E1  05DF  1874         BBC     #IO$V_STARTUP,R7,85$                ; Br if not startup request
  13 68 A5   02   E0  05E3  1875         BBS     #UCB$V_XD_PTP,UCB$W_DEVSTS(R5),85$ ; Br if point-to-point
```

```
      51   0474 8F   3C 05E8          MOVZWL  #NMA$C_PCCI_TRI,R1                 ; Get trib address
              1C53   30 05ED          BSBW    UNPACK_P2_BUF                      ; From P2 buffer
             08 50   E8 05F0          BLBS    R0,85$                             ; Br if given - Okay
      50   0114 8F   3C 05F3          MOVZWL  #SS$_INSFARG,R0                    ; Insufficient arguments
              FF68   31 05F8  83$:    BRW     25$                                ; Finish the I/O request
                        05FB
      51   0060 8F   3C 05FB  85$:    MOVZWL  #CDB_C_LENGTH,R1                   ; Get size of CDB
                53   DD 0600          PUSHL   R3                                 ; Save IRP address
       00000000'GF   16 0602          JSB     G^EXE$BUFQUOPRC                    ; Check caller's quota
             91 50   E9 0608          BLBC    R0,37$                             ; Br if error
       00000000'GF   16 060B          JSB     G^EXE$ALONONPAGED                  ; Allocate the CDB
             88 50   E9 0611          BLBC    R0,37$                             ; Br if error
             53 8ED0    0614          POPL    R3                                 ; Restore IRP address
      50   0080 C4   D0 0617          MOVL    PCB$L_JIB(R4),R0                   ; Get JIB address
 20 A0   00000060 8F C2 061C          SUBL    #CDB_C_LENGTH,JIB$L_BYTCNT(R0)     ; Charge user for CDB
 24 A0   00000060 8F C2 0624          SUBL    #CDB_C_LENGTH,JIB$L_BYTLM(R0)      ; ..and limit
                        062C  ;
                        062C  ; Zero the CDB
                        062C  ;
             59 52   D0 062C          MOVL    R2,R9                              ; Copy CDB address
                38   BB 062F          PUSHR   #^M<R3,R4,R5>                      ; Save registers
62  0060 8F  00 62   00 2C 0631       MOVC5   #0,(R2),#0,#CDB_C_LENGTH,(R2)      ; Zero the CDB
                38   BA 0639          POPR    #^M<R3,R4,R5>                      ; Restore registers
      48 A9  0C A3   D0 063B          MOVL    IRP$L_PID(R3),CDB_L_PID(R9)        ; Save starter's PID
      4C A9  28 A3   B0 0640          MOVW    IRP$W_CHAN(R3),CDB_W_CHAN(R9)      ; Save associated channel
                        0645          ASSUME  CDB_B_TYPE EQ CDB_W_SIZE+2
                        0645          ASSUME  CDB_B_POL EQ CDB_B_TYPE+1
                        0645  ;
                        0645  ; We will init the Polling state to unlatched. It is assumed that
                        0645  ; a zero polling state means unlatched!
                        0645  ;
       00130060 8F   D0 0645          MOVL    #<<DYN$C_BUFIO@16>!CDB_C_LENGTH>,- ; Set type and size
             08 A9      064B                  CDB_W_SIZE(R9)                     ;
                        064D  ;
                        064D  ; Initialize CDB queues
                        064D  ;
             50 05   9A 064D          MOVZBL  S^#CDB_C_QUEUES,R0                 ; Set number of queue heads
          52 10 A9   9E 0650          MOVAB   CDB_Q_QUEUES(R9),R2                ; Set address of first head
             82 62   9E 0654  90$:    MOVAB   (R2),(R2)+                         ; Init forward link pointer
          82 FC A2   9E 0657          MOVAB   -4(R2),(R2)+                       ; Init backward link pointer
             F6 50   F5 065B          SOBGTR  R0,90$                             ; Loop if more queues
                        065E  ;
                        065E  ; Initialize CDB default parameters
                        065E  ;
             50 12   3C 065E          MOVZWL  #DEF_TRIB_PARAMSZ,R0               ; Set size of parameters in bytes
         51 FAC1 CF  9E 0661          MOVAB   DEF_TRIB_PARAM,R1                  ; Set address of defaults
          52 4E A9   9E 0666          MOVAB   CDB_B_SETPRM(R9),R2                ; Set address of parameters
             82 81   90 066A  100$:   MOVB    (R1)+,(R2)+                        ; Set default
             FA 50   F5 066D          SOBGTR  R0,100$                            ; Loop for all parameters
                        0670
    5C A9  02B4 C5   B0 0670          MOVW    UCB$W_XD_RTT(R5),CDB_W_SEL_TIMER(R9) ; Set default retransmit timer
    0F A9  0135 C5   90 0676          MOVB    UCB$B_XD_XMT_TRB(R5),CDB_B_XMT_CNT(R9) ; Init max xmits allowed
    05 44 A5     06  E0 067C          BBS     #XM$V_CHR_CTRL,UCB$L_DEVDEPEND(R5),110$ ; Br if control mode
    0F A9    7F 8F   90 0681          MOVB    #INF_XMT,CDB_B_XMT_CNT(R9)         ; Else, allow infinite xmits
       58    41 A3   9A 0686  110$:   MOVZBL  IRP$B_INDEX(R3),R8                 ; Get vector index
                        068A          SETIPL  UCB$B_FIPL(R5)                     ; Sync access to UCB
   0218 C548     59  D0 068E          MOVL    R9,UCB$L_XD_CDB_VEC(R5)[R8]        ; Store CDB address in UCB
```

```
 44 A3   0218 C548  DE  0694          MOVAL   UCB$L_XD_CDB_VEC(R5)[R8],-          ; Store address of CDB address
                        069B                  IRP$L_CDB(R3)                       ; ..in CDB
          0C A3     D0  069B          MOVL    IRP$L_PID(R3),-                     ; Save PID in UCB
       0158 C548        069E                  UCB$L_XD_PID_VEC(R5)[R8]            ;
          28 A3     B0  06A2          MOVW    IRP$W_CHAN(R3),-                    ; Save channel index in UCB
       01D8 C548        06A5                  UCB$W_XD_CHAN_VEC(R5)[R8]           ;
          0134 C5   96  06A9          INCB    UCB$B_XD_TRB_CNT(R5)                ; Tally one more tributary
                        06AD ;
                        06AD ; Tributary now exists change its characteristics and set them
                        06AD ; if trib is established.
                        06AD ;
             1989   30  06AD 120$:    BSBW    CHG_CDB                             ; Change CDB parameters
       10 57   06   E0  06B0          BBS     #IO$V_STARTUP,R7,125$               ; Br if startup request
          50   01   3C  06B4          MOVZWL  #SS$_NORMAL,R0                      ; Else, set successful return
          51   69   D0  06B7          MOVL    CDB_L_DEVDEPEND(R9),R1              ; Set IOSB1 return
       51   44 A5   C8  06BA          BISL    UCB$L_DEVDEPEND(R5),R1              ; ..plus UCB status
       00000000'GF  17  06BE 123$:    JMP     G^EXE$FINISHIO                      ; Finish the I/O request
                        06C4 ;
                        06C4 ; Startup tributary requested
                        06C4 ;
       21 A3   00   90  06C4 125$:    MOVB    S^#XD_FC_V_INITT,IRP$B_XDFUNC(R3) ; Set internal function code
  02A0 C5   48 A9   D1  06C8          CMPL    CDB_L_PID(R9),UCB$L_XD_PID(R5)     ; Same PID as starter?
               09   12  06CE          BNEQ    127$                                ; Br if not
       09 69   00   E1  06D0          BBC     #XM$V_CHR_MOP,CDB_L_DEVDEPEND(R9),128$ ; Br if not MOP mode
    04 44 A5   06   E1  06D4          BBC     #XM$V_CHR_CTRL,UCB$L_DEVDEPEND(R5),128$ ; Br if not control station
       4E A9   01   90  06D9 127$:    MOVB    #1,CDB_B_MRB(R9)                    ; Else, set MRB to one
    4E A9   FF 8F   91  06DD 128$:    CMPB    #INF_RCV,CDB_B_MRB(R9)              ; Is MRB at 'unlimited'?
               05   13  06E2          BEQL    129$                                ; Br if yes
                        06E4          SETBIT  #CD_TS_V_BUFQUO,CDB_W_STS(R9)       ; Else, must user buffer manager
    12 68 A5   02   E1  06E9 129$:    BBC     #UCB$V_XD_PTP,UCB$W_DEVSTS(R5),130$ ; Br if not point-point
       0E A9   01   90  06EE          MOVB    #1,CDB_B_TRB_ADDR(R9)               ; Else set trib address
       40 A3   01   90  06F2          MOVB    #1,IRP$Q_STATION(R3)                ; Again in IRP
       50   41 A3   9A  06F6          MOVZBL  IRP$B_INDEX(R3),R0                  ; Get vector index
    0138 C540  01   90  06FA          MOVB    #1,UCB$B_XD_TRIB_VEC(R5)[R0]        ; And again in UCB trib vector
            0E A9   95  0700 130$:    TSTB    CDB_B_TRB_ADDR(R9)                  ; Trib address given?
               31   12  0703          BNEQ    170$                                ; Br if yes - okay
    50   0114 8F    3C  0705          MOVZWL  #SS$_INSFARG,R0                     ; Trib address missing
    51   0474 8F    3C  070A          MOVZWL  #NMA$C_PCCI_TRI,R1                  ; Set bad parameter code
               AD   11  070F          BRB     123$                                ; Finish the I/O
                        0711 ;
                        0711 ; Tributary already exists .. validate P2
                        0711 ;
          52   69   D0  0711 150$:    MOVL    CDB_L_DEVDEPEND(R9),R2              ; Get trib status
             1972   30  0714          BSBW    VALIDATE_P2_CDB                     ; Validate P2
            A4 50   E9  0717          BLBC    R0,123$                             ; Br if error
                        071A          SETIPL  UCB$B_FIPL(R5)                      ; Sync access to CDB
    04 0C A9   00   E0  071E          BBS     #CD_TS_V_ESTAB,CDB_W_STS(R9),160$ ; Br if trib established
                        0723 ;
                        0723 ; Tributary not active .. change CDB parameters and do startup if necessary
                        0723 ;
               69   94  0723          CLRB    CDB_L_DEVDEPEND(R9)                 ; Clear old characteristics
               86   11  0725          BRB     120$                                ; Change CDB and start if needed
                        0727 ;
                        0727 ; Tributary already active .. set CDB and give request to device
                        0727 ;
            3C A3   94  0727 160$:    CLRB    IRP$L_MEDIA+4(R3)                   ; Clear trib characteristics
             190C   30  072A          BSBW    CHG_CDB                             ; Change the CDB params
```

```
      21 A3    05   90 072D  1990        MOVB    S^#XD_FC_V_CHGT,IRP$B_XDFUNC(R3) ; Set new CDB params
   38 A3    FF 8F   90 0731  1991        MOVB    #255,IRP$L_MEDIA(R3)             ; Set 8 parameters
            0385    31 0736  1992 170$:  BRW     QUEPKT                           ; Queue packet to driver
```

```
                          0739          .SBTTL  SETMODE_CTRL, Perform setmode FDT operation on controller
                          0739
                          0739  ;++
                          0739  ; SETMODE_CTRL - Perform setmode FDT operation on controller
                          0739  ;
                          0739  ; Functional description:
                          0739  ;
                          0739  ; This routine performs the SETMODE FDT setup for the controller.
                          0739  ;
                          0739  ; Inputs:
                          0739  ;
                          0739  ;         R3 = IRP address
                          0739  ;         R4 = PCB address
                          0739  ;         R5 = UCB address
                          0739  ;         R7 = IRP function word
                          0739  ;
                          0739  ; Outputs:
                          0739  ;
                          0739  ;         R0 = status of setmode request
                          0739  ;
                          0739  ;         R3-R5 are preserved.
                          0739  ;
                          0739  ;--
                          0739
                          0739  SETMODE_CTRL:                                           ; Perform setmode on controller
   1C 57    0A     E1     0739          BBC     #IO$V_SET_MODEM,R7,10$                  ; Br if not setmodem request
                          073D  ;
                          073D  ; Set modem modifier specified
                          073D  ;
                          073D  ;         P1(AP) = address of modem longword bit settings
                          073D  ;
      50    14     3C     073D          MOVZWL  #SS$_BADPARAM,R0                        ; Assume P1 buffer not given
      52    6C     D0     0740          MOVL    P1(AP),R2                               ; Get address of P1 buffer
            44     13     0743          BEQL    20$                                     ; Br if none specified
      50    0C     3C     0745          MOVZWL  #SS$_ACCVIO,R0                          ; Assume access violation
                          0748          IFNORD  #4,(R2),20$                             ; Br if access violation
   38 A3    62     D0     074E          MOVL    (R2),IRP$L_MEDIA(R3)                    ; Store modem bits in IRP
   21 A3    0D     90     0752          MOVB    S^#XD_FC_V_WTMODEM,IRP$B_XDFUNC(R3) ; Set function request
          0365     31     0756          BRW     QUEPKT                                  ; Queue packet to driver
                          0759
          010E     30     0759  10$:    BSBW    GET_CHAR_BUFS                           ; Get P1 and P2 characteristics
         2A 50     E9     075C          BLBC    R0,20$                                  ; Br if error - Abort I/O
   52    68 A5     3C     075F          MOVZWL  UCB$W_DEVSTS(R5),R2                     ; Get device status
          192F     30     0763          BSBW    VALIDATE_P2_UCB                         ; Validate the P2 buffer
         1A 50     E9     0766          BLBC    R0,17$                                  ; Br if error
                          0769
   1F 57    07     E1     0769          BBC     #IO$V_SHUTDOWN,R7,25$                   ; Br if not shutdown request
                          076D  ;
                          076D  ; Shutdown modifier specified
                          076D  ;
          17F6     30     076D          BSBW    CHG_UCB                                 ; Update the UCB
            00     E1     0770          BBC     #UCB$V_XD_INITED,-                      ; Br if controller not up
      07 68 A5            0772                  UCB$W_DEVSTS(R5),15$                    ;
   21 A3    03     90     0775          MOVB    #XD_FC_V_STOPC,IRP$B_XDFUNC(R3)         ; Set function request
          0342     31     0779          BRW     QUEPKT                                  ; Queue packet to driver
                          077C
   51    44 A5     D0     077C  15$:    MOVL    UCB$L_DEVDEPEND(R5),R1                  ; Get IOSB1 return
```

```
       50  01  9A 0780        MOVZBL  S^#SS$_NORMAL,R0               ; Set success
00000000'GF    17 0783 17$:   JMP     G^EXE$FINISHIO                 ; Complete the I/O request
                  0789
         FB49  31 0789 20$:   BRW     ABORTIO                        ; Abort the I/O request
                  078C ;
                  078C ; Make sure this request is from the starter PID unless device not inited
                  078C ; or device is running in control station mode.
                  078C ;
           00  E1 078C 25$:   BBC     #UCB$V_XD_INITED,-             ; Br if device not inited
     12 68 A5     078E                UCB$W_DEVSTS(R5),30$           ;
           06  E0 0791        BBS     #XM$V_CHR_CTRL,-               ; Br if control station
     0D 44 A5     0793                UCB$L_DEVDEPEND(R5),30$        ;
  50  0850 8F  3C 0796        MOVZWL  #SS$_DEVICEFULL,R0             ; Assume someone has device
02A0 C5  0C A3 D1 079B        CMPL    IRP$L_PID(R3),UCB$L_XD_PID(R5) ; Is this the owner PID?
           E6  12 07A1        BNEQ    20$                            ; Br if no - wrong user
                  07A3
     03 57 06  E0 07A3 30$:   BBS     #IO$V_STARTUP,R7,31$           ; Br if startup request
         007B  31 07A7        BRW     50$                            ; Else, no modifier
                  07AA ;
                  07AA ; Controller startup requested
                  07AA ;
                  07AA ;              P3(AP) = number of receive buffers desired
                  07AA ;
   51   08 AC  3C 07AA 31$:   MOVZWL  P3(AP),R1                      ; Get number of receive buffers
           1B  13 07AE        BEQL    35$                            ; Br if no - ignore it
           00  E1 07B0        BBC     #UCB$V_XD_INITED,-             ; Br if device not inited
     11 68 A5     07B2                UCB$W_DEVSTS(R5),33$           ;
02AB C5    51  91 07B5        CMPB    R1,UCB$B_XD_BFN(R5)            ; Are the buffer numbers the same?
           0A  13 07BA        BEQL    33$                            ; Br if yes
  51  0451 8F  3C 07BC 32$:   MOVZWL  #NMA$C_PCLI_BFN,R1             ; Set IOSB1 return
     50    14  3C 07C1        MOVZWL  #SS$_BADPARAM,R0               ; Set error return
           BD  11 07C4        BRB     17$                            ; Finish the I/O request
02AB C5    51  90 07C6 33$:   MOVB    R1,UCB$B_XD_BFN(R5)            ; Store new receive buffer number
                  07CB ;
                  07CB ; Set new P1, P2 parameters
                  07CB ;
           00  E0 07CB 35$:   BBS     #UCB$V_XD_INITED,-             ; Br if device already inited
     74 68 A5     07CD                UCB$W_DEVSTS(R5),70$           ;
     03 38 A3  E9 07D0        BLBC    IRP$L_MEDIA(R3),37$            ; Br if no P1
        44 A5  94 07D4        CLRB    UCB$L_DEVDEPEND(R5)            ; Clear old characteristics
         178C  30 07D7 37$:   BSBW    CHG_UCB                        ; Change the UCB characteristics
  51  02AB C5  9A 07DA        MOVZBL  UCB$B_XD_BFN(R5),R1            ; Get number of receive buffers
   52   3A A3  3C 07DF        MOVZWL  IRP$L_MEDIA+2(R3),R2           ; Get message size from P1
     04 38 A3  E8 07E3        BLBS    IRP$L_MEDIA(R3),40$            ; Br if P1 buffer valid
   52   42 A5  3C 07E7        MOVZWL  UCB$W_DEVBUFSIZ(R5),R2         ; Else, get buffer size from UCB
     50    14  3C 07EB 40$:   MOVZWL  #SS$_BADPARAM,R0               ; Assume bad parameter
     51    52  C4 07EE        MULL    R2,R1                          ; Compute total needed for buffers
           C9  13 07F1        BEQL    32$                            ; Br if zero - error
     57    51  3C 07F3        MOVZWL  R1,R7                          ; Copy quota
     57    51  D1 07F6        CMPL    R1,R7                          ; Overflow?
           C1  12 07F9        BNEQ    32$                            ; Br if error
           53  DD 07FB        PUSHL   R3                             ; Save R3
00000000'GF    16 07FD        JSB     G^EXE$BUFQUOPRC                ; Check caller's quota
        53   8ED0 0803        POPL    R3                             ; Restore R3
        B3 50  E9 0806        BLBC    R0,32$                         ; Br if error
   42 A3   57  B0 0809        MOVW    R7,IRP$W_QUOTA(R3)             ; Save quota in packet
  50  0080 C4  D0 080D        MOVL    PCB$L_JIB(R4),R0               ; Get JIB address
```

```
       20 A0    57  C2  0812        SUBL    R7,JIB$L_BYTCNT(R0)               ; Charge user for rec. bufs
       24 A0    57  C2  0816        SUBL    R7,JIB$L_BYTLM(R0)                ; ..and limit
       21 A3    01  90  081A        MOVB    S^#XD_FC_V_INITC,IRP$B_XDFUNC(R3) ; Set function request in IRP
       39 A3    01  90  081E        MOVB    S^#XD_FC_V_INITC,IRP$L_MEDIA+1(R3) ; ..Twice for fork process
              0299  31  0822        BRW     QUEPKT                            ; Queue request to driver
                        0825  ;
                        0825  ; No modifier specified - change controller parameters
                        0825  ;
                00  E0  0825  50$:  BBS     #UCB$V_XD_INITED,-                ; Br if already inited
          1A 68 A5      0827                UCB$W_DEVSTS(R5),70$              ;
          03 38 A3  E9  082A        BLBC    IRP$L_MEDIA(R3),53$               ; Br if no P1 buffer
             44 A5  94  082E        CLRB    UCB$L_DEVDEPEND(R5)               ; Clear old UCB characteristics
              1732  30  0831  53$:  BSBW    CHG_UCB                           ; Change UCB parameters
          50    01  3C  0834        MOVZWL  S^#SS$_NORMAL,R0                  ; Else, set success
       51    44 A5  D0  0837        MOVL    UCB$L_DEVDEPEND(R5),R1            ; Set IOSB1 return
       00000000'GF  17  083B  55$:  JMP     G^EXE$FINISHIO                    ; Finish the I/O request
                        0841
              FA91  31  0841  60$:  BRW     ABORTIO                           ; Abort the I/O request
                        0844  ;
                        0844  ; Device already inited - set new parameters and give them to device
                        0844  ;
          0F 38 A3  E9  0844  70$:  BLBC    IRP$L_MEDIA(R3),80$               ; Br if no P1 buffer
    44 A5    3C A3  91  0848        CMPB    IRP$L_MEDIA+4(R3),UCB$L_DEVDEPEND(R5) ; Are characteristics okay?
                08  13  084D        BEQL    80$                               ; Yes - let it go
          50    14  3C  084F        MOVZWL  #SS$_BADPARAM,R0                  ; Return error
          51    01  CE  0852        MNEGL   S^#1,R1                           ; No specific parameter
                E4  11  0855        BRB     55$                               ; Complete the I/O
                        0857
              170C  30  0857  80$:  BSBW    CHG_UCB                           ; Change UCB parameters
       21 A3    04  90  085A        MOVB    S^#XD_FC_V_CHGC,IRP$B_XDFUNC(R3)  ; Set function request
       39 A3    04  90  085E        MOVB    S^#XD_FC_V_CHGC,IRP$L_MEDIA+1(R3) ; ..Twice for fork process
    38 A3    F0 8F  90  0862        MOVB    #<15@4>,IRP$L_MEDIA(R3)           ; Set only four parameters
              0254  31  0867        BRW     QUEPKT                            ; Queue packet to driver
```

```
                    086A          .SBTTL  GET_CHAR_BUFS,  Get P1 and P2 characteristics buffers
                    086A
                    086A  ;++
                    086A  ; GET_CHAR_BUFS - Get P1 and P2 characteristics buffers
                    086A  ;
                    086A  ; Functional description:
                    086A  ;
                    086A  ; This routine saves the P1 and P2 buffers for later use by the driver.
                    086A  ; The P1 buffer is saved in the IRP (IRP$L_MEDIA) as a quadword value.
                    086A  ; The P2 buffer is saved by allocating the appropriate amount of memory from
                    086A  ; non-paged pool.   The user's quota is checked before the allocation is made.
                    086A  ; And the non-paged pool buffer is charged against the user's quota. The P2
                    086A  ; system buffer address is passed in IRP$L_SVAPTE of the IRP.
                    086A  ;
                    086A  ;
                    086A  ; Inputs:
                    086A  ;
                    086A  ;         R3 = IRP address
                    086A  ;         R4 = PCB address
                    086A  ;         R5 = UCB address
                    086A  ;
                    086A  ; Outputs:
                    086A  ;
                    086A  ;         R0 = status of buffers
                    086A  ;
                    086A  ;         R3-R5 are preserved.
                    086A  ;
                    086A  ;--
                    086A
                    086A  GET_CHAR_BUFS:                                  ; Get characteristics buffers
                    086A  ;
                    086A  ; Check access to P1 buffer and save P1 characteristics
                    086A  ;
      38 A3     7C  086A          CLRQ    IRP$L_MEDIA(R3)                ; Reset P1 chars
   52    6C     D0  086D          MOVL    P1(AP),R2                      ; Get address of P1 char buf
         11     13  0870          BEQL    10$                            ; Branch if no P1 buffer
   50    0C     3C  0872          MOVZWL  #SS$_ACCVIO,R0                 ; Assume access violation
                    0875          IFNORD  #8,(R2),20$                    ; Check access
 38 A3   62     7D  087B          MOVQ    (R2),IRP$L_MEDIA(R3)           ; Save P1 characteristics
 38 A3   01     90  087F          MOVB    #1,IRP$L_MEDIA(R3)             ; Indicate valid P1 buffer
                    0883  ;
                    0883  ; Check access to P2 buffer and check process's buffer quota
                    0883  ;
 51    04 AC    D0  0883  10$:    MOVL    P2(AP),R1                      ; Get address P2 char buf desc
          33    13  0887          BEQL    40$                            ; Br if no P2 buffer
          53    DD  0889          PUSHL   R3                             ; Save R3
   00000000'GF  16  088B          JSB     G^EXE$PROBER_DSC               ; Check access to buffer
       0E 50    E9  0891          BLBC    R0,15$                         ; Br if error
   51     51    3C  0894          MOVZWL  R1,R1                          ; Get buffer size as a word
          52    DD  0897          PUSHL   R2                             ; Save R2
   00000000'GF  16  0899          JSB     G^EXE$BUFQUOPRC                ; Check for buffered quota
        52 8ED0     089F          POPL    R2                             ; Restore R2
        53 8ED0     08A2  15$:    POPL    R3                             ; Restore R3
    01 50       E8  08A5          BLBS    R0,30$                         ; Branch if quota ok
            05      08A8  20$:    RSB                                    ; Return
                    08A9
                    08A9  ;
```

```
                        08A9  2199 ; Quota OKAY, allocate buffer and copy info.
                        08A9  2200 ;
           0261   30    08A9  2201 50$:    BSBW    ALLOC_P2BUF              ; Allocate buffer
          10 50   E9    08AC  2202         BLBC    R0,50$                   ; Br if error
     50   2C A3   D0    08AF  2203         MOVL    IRP$L_SVAPTE(R3),R0      ; Get P2 buffer address
             38   BB    08B3  2204         PUSHR   #^M<R3,R4,R5>            ; Save sacred registers
0C A0   62   51   28    08B5  2205         MOVC3   R1,(R2),P2B_T_DATA(R0)   ; Save P2 char buffer
             38   BA    08BA  2206         POPR    #^M<R3,R4,R5>            ; Restore registers
     50      01   9A    08BC  2207 40$:    MOVZBL  S^#SS$_NORMAL,R0         ; Set success
                  05    08BF  2208 50$:    RSB                              ; Return
```

```
                        08C0        .SBTTL  SENSEMODE_FDT,  Sense Mode I/O operation FDT routine
                        08C0
                        08C0  ;++
                        08C0  ; SENSEMODE_FDT - Sense Mode FDT routine
                        08C0  ;
                        08C0  ; Functional Description:
                        08C0  ;
                        08C0  ;  This routine returns information to the caller about the configuration
                        08C0  ;  and status of the DMP11 device.  Depending on the function modifier,
                        08C0  ;  either the device characteristics, error counters or modem register
                        08C0  ;  contents are returned.
                        08C0  ;
                        08C0  ;
                        08C0  ;  The QIO parameters for SENSEMODE are:
                        08C0  ;
                        08C0  ;       P1 = optional address of quadword or longword buffer
                        08C0  ;       P2 = optional address of buffer descriptor for extended characteristics
                        08C0  ;
                        08C0  ;
                        08C0  ; Inputs:
                        08C0  ;
                        08C0  ;       R3 = IRP address
                        08C0  ;       R4 = PCB address
                        08C0  ;       R5 = UCB address
                        08C0  ;       R6 = CCB address
                        08C0  ;       R7 = Function code
                        08C0  ;       AP = Address of first function-dependent QIO parameter
                        08C0  ;
                        08C0  ; Outputs:
                        08C0  ;
                        08C0  ;       R0 = status return of sensemode request
                        08C0  ;
                        08C0  ;       R3-R5 are preserved.
                        08C0  ;
                        08C0  ;--
                        08C0
                        08C0  SENSEMODE_FDT:                                     ; Sensemode FDT I/O processing
        40 A3     7C    08C0          CLRQ    IRP$Q_STATION(R3)                  ; Clear station address field
                        08C3
                        08C3          PUSHQ   R6                                 ; Save registers
     50    03     90    08C6          MOVB    #JNL_QIO,R0                        ; Set journal type = QIO
         1C01     30    08C9          BSBW    XD_FILL_JNX                        ; Fill the journal buffer
        04 50     E9    08CC          BLBC    R0,1$                              ; Br if error
     F8 A6 6C     D0    08CF          MOVL    P1(AP),-8(R6)                      ; Store the P1 parameter
                        08D3  1$:     POPQ    R6                                 ; Restore registers
                        08D6
     57  20 A3    B0    08D6          MOVW    IRP$W_FUNC(R3),R7                  ; Get entire function code
     03 57  07    E1    08DA          BBC     #IO$V_RD_MODEM,R7,3$               ; Br if not read modem request
         01BF     31    08DE          BRW     SENSE_MODEM                        ; Else, process read modem
                        08E1
     03 57  09    E1    08E1  3$:     BBC     #IO$V_CTRL,R7,5$                   ; Br if not controller request
         00EF     31    08E5          BRW     SENSEMODE_CTRL                     ; Else, process controller req.
                        08E8  ;
                        08E8  ; Tributary sensemode request
                        08E8  ;
         1603     30    08E8  5$:     BSBW    XLATE                              ; Get CDB address
        64 50     E9    08EB          BLBC    R0,30$                             ; Br if error
```

```
   31 57     08   E1  08EE           BBC     #IO$V_RD_COUNT,R7,20$             ; Br if not read counters
                      08F2      ;
                      08F2      ; Read tributary counters - modifier RD_COUNT
                      08F2      ;
         01DB     30  08F2           BSBW    CHECK_P2                          ; Check P2 buffer
      50   14     3C  08F5           MOVZWL  #SS$_BADPARAM,R0                  ; Assume zero length buffer
   3A A3   51     B0  08F8           MOVW    R1,IRP$L_MEDIA+2(R3)              ; Save user size of P2 buffer
           54     13  08FC           BEQL    30$                               ; Br if none
      51   3B     3C  08FE           MOVZWL  S^#TRIB_CNT_BUFSIZ,R1             ; Get size of tributary P2 buf
         0209     30  0901           BSBW    ALLOC_P2BUF                       ; Allocate a buffer
        4B 50     E9  0904           BLBC    R0,30$                            ; Br if error
   51   2C A3     D0  0907           MOVL    IRP$L_SVAPTE(R3),R1               ; Get system P2 buffer addr
   04 A1   52     D0  090B           MOVL    R2,P2B_L_BUFFER(R1)               ; Save user's P2 buffer addr
 38 A3   FE 8F    90  090F           MOVB    #254,IRP$L_MEDIA(R3)              ; Get 7 counters
   21 A3   07     90  0914           MOVB    S^#XD_FC_V_RDTERR,IRP$B_XDFUNC(R3) ; Assume Read counts
   04 57   0A     E1  0918           BBC     #IO$V_CLR_COUNT,R7,10$            ; Br if not clear counts
   21 A3   09     90  091C           MOVB    S^#XD_FC_V_RCTERR,IRP$B_XDFUNC(R3) ; Else, Read and Clear
         019B     31  0920      10$: BRW     QUEPKT                            ; Queue packet to driver
                      0923
   2E 57     06   E1  0923      20$: BBC     #IO$V_RD_MEM,R7,40$               ; Br if read tss request
                      0927      ;
                      0927      ; Read TSS address - modifier RD_MEM
                      0927      ;
   51     04 AC   9A  0927           MOVZBL  P2(AP),R1                         ; Get TSS address
      1F   51     91  092B           CMPB    R1,S^#31                          ; Is TSS address too big?
           22     1A  092E           BGTRU   30$                               ; Br if yes - error
   3E A3   51     90  0930           MOVB    R1,IRP$L_MEDIA+6(R3)              ; Save TSS address in IRP
      51   04     9A  0934           MOVZBL  S^#4,R1                           ; Check access to P1 buffer
         01C0     30  0937           BSBW    CHECK_P1                          ; Check P1 buffer
         01D0     30  093A           BSBW    ALLOC_P2BUF                       ; Allocate a P2 buffer
        12 50     E9  093D           BLBC    R0,30$                            ; Br if allocation error
   51   2C A3     D0  0940           MOVL    IRP$L_SVAPTE(R3),R1               ; Get system buffer address
 04 A1   38 A3    D0  0944           MOVL    IRP$L_MEDIA(R3),P2B_L_BUFFER(R1) ; Save user P1 buffer VA
        38 A3     D4  0949           CLRL    IRP$L_MEDIA(R3)                   ; Indicate NOT a counter IRP
   21 A3   0B     90  094C           MOVB    S^#XD_FC_V_RDTSS,IRP$B_XDFUNC(R3) ; Set function request
           CE     11  0950           BRB     10$                               ; Queue packet
                      0952
         F980     31  0952      30$: BRW     ABORTIO                           ; Abort the I/O request
                      0955
                      0955      ;
                      0955      ; Read tributary parameters - no modifier
                      0955      ;
   51      08     3C  0955      40$: MOVZWL  S^#8,R1                           ; Size of P1 buffer if present
         0173     30  0958           BSBW    CHECK_BUFS                        ; Check P1 and P2 buffers
   3C A3   51     B0  095B           MOVW    R1,IRP$L_MEDIA+4(R3)              ; Save user P2 buffer length
           5B     13  095F           BEQL    60$                               ; Br if no P2 buffer present
                      0961
  51   006C 8F    3C  0961           MOVZWL  #TRIB_PRM_BUFSIZ+6,R1             ; Set size of required buffer
         0144     30  0966           BSBW    ALLOC_P2BUF                       ; Allocate a P2 buffer
        E6 50     E9  0969           BLBC    R0,30$                            ; Br if error
   51   2C A3     D0  096C           MOVL    IRP$L_SVAPTE(R3),R1               ; Get system P2 buffer address
   04 A1   52     D0  0970           MOVL    R2,P2B_L_BUFFER(R1)               ; Set user's P2 buffer address
  51    F752 CF   9E  0974           MOVAB   TRIB_PARAM,R1                     ; Get address of return table
      54   59     D0  0979           MOVL    R9,R4                             ; Get CDB address
         186A     30  097C           BSBW    RETURN_P2                         ; Return the P2 parameters
   42 A3   01     B0  097F           MOVW    S^#SS$_NORMAL,IRP$W_CSTATUS(R3) ; Assume success
                      0983      ;
```

```
                         0983        ;       Return polling substate - only if device is inited.
                         0983        ;
              00    E1   0983                BBC     #UCB$V_XD_INITED,-                      ; Br if not inited
        10 68 A5         0985                        UCB$W_DEVSTS(R5),45$                    ;
              06    E1   0988                BBC     #XM$V_CHR_CTRL,-                        ; Br if NOT a control station
         0B 44 A5        098A                        UCB$L_DEVDEPEND(R5),45$                 ;
     3E A3    02    90   098D                MOVB    #2,IRP$L_MEDIA+6(R3)                    ; Set TSS address (as read tss)
     21 A3    15    90   0991                MOVB    #XD_FC_V_POLSS,IRP$B_XDFUNC(R3)         ; Set function request
            0126    31   0995                BRW     QUEPKT                                  ; Else, queue the packet
                         0998
     32 A3    06    A2   0998        45$:    SUBW    #6,IRP$W_BCNT(R3)                       ; No polling substate return
     50    32 A3    B0   099C                MOVW    IRP$W_BCNT(R3),R0                       ; Assume user has good size buffer
     50    3C A3    B1   09A0                CMPW    IRP$L_MEDIA+4(R3),R0                    ; Is user buffer large enough?
              0E    1E   09A4                BGEQU   50$                                     ; Br if yes
   42 A3    0601 8F B0   09A6                MOVW    #SS$_BUFFEROVF,IRP$W_CSTATUS(R3)        ; Return partial success
     50    3C A3    B0   09AC                MOVW    IRP$L_MEDIA+4(R3),R0                    ; Set size of return
     32 A3    50    B0   09B0                MOVW    R0,IRP$W_BCNT(R3)                       ;       ...
  50    50    10    78   09B4        50$:    ASHL    #16,R0,R0                               ; Shift size of buffer return
     50    42 A3    B0   09B8                MOVW    IRP$W_CSTATUS(R3),R0                    ; Set success status
                         09BC
     52    38 A3    D0   09BC        60$:    MOVL    IRP$L_MEDIA(R3),R2                      ; Retrieve P1 buffer address
              08    13   09C0                BEQL    70$                                     ; Br if none
     62    40 A5    7D   09C2                MOVQ    UCB$B_DEVCLASS(R5),(R2)                 ; Else, return characteristics
     04 A2    69    C8   09C6                BISL    CDB_L_DEVDEPEND(R9),4(R2)               ;       ...
        51    69    D0   09CA        70$:    MOVL    CDB_L_DEVDEPEND(R9),R1                  ; Get device dependend info
     51    44 A5    C8   09CD                BISL    UCB$L_DEVDEPEND(R5),R1                  ; ..from UCB also
    00000000'GF     17   09D1                JMP     G^EXE$FINISHIO                          ; Complete the I/O request
```

```
                           09D7           .SBTTL  SENSEMODE_CTRL,  Perform SENSEMODE FDT processing for controller
                           09D7
                           09D7   ;++
                           09D7   ; SENSEMODE_CTRL - Perform SENSEMODE FDT processing for controller
                           09D7   ;
                           09D7   ; Functional description:
                           09D7   ;
                           09D7   ; This routine performs all FDT checking for a controller SENSEMODE request.
                           09D7   ; The P2 buffer if present is check for write access and if okay, a system
                           09D7   ; buffer is allocated for temporarily saving the needed information. The
                           09D7   ; P1 sensemode information is returned through IRP$L_MEDIA and IRP$L_MEDIA+4.
                           09D7   ;
                           09D7   ; Inputs:
                           09D7   ;
                           09D7   ;       R3 = IRP address
                           09D7   ;       R4 = PCB address
                           09D7   ;       R5 = UCB address
                           09D7   ;       R7 = Function code
                           09D7   ;
                           09D7   ; Outputs:
                           09D7   ;
                           09D7   ;       R0 = status return for request
                           09D7   ;
                           09D7   ;       R3-R5 are preserved.
                           09D7   ;
                           09D7   ;--
                           09D7
                           09D7   SENSEMODE_CTRL:                                         ; Process controller sensemode FDT
    31 57    08      E1    09D7           BBC     #IO$V_RD_COUNT,R7,20$                   ; Br if not read counters
                           09DB   ;
                           09DB   ; Read controller counters - modifier RD_COUNT
                           09DB   ;
          00F2       30    09DB           BSBW    CHECK_P2                                ; Check P2 buffer
       50    14      3C    09DE           MOVZWL  #SS$_BADPARAM,R0                        ; Assume zero length buffer
    3A A3    51      B0    09E1           MOVW    R1,IRP$L_MEDIA+2(R3)                    ; Save user P2 buffer size
             54      13    09E5           BEQL    30$                                     ; Br if no P2 buffer
       51    0A      3C    09E7           MOVZWL  S^#LINE_CNT_BUFSIZ,R1                   ; Get size of controller counts
          0120       30    09EA           BSBW    ALLOC_P2BUF                             ; Allocate a buffer
          4B 50      E9    09ED           BLBC    R0,30$                                  ; Br if error
    51    2C A3      D0    09F0           MOVL    IRP$L_SVAPTE(R3),R1                     ; Get system P2 buffer address
    04 A1    52      D0    09F4           MOVL    R2,P2B_L_BUFFER(R1)                     ; Save user P2 buffer address
 38 A3    60 8F      90    09F8           MOVB    #<3@5>,IRP$L_MEDIA(R3)                  ; Get 2 counters
    21 A3    08      90    09FD           MOVB    S^#XD_FC_V_RDGERR,IRP$B_XDFUNC(R3) ; Assume only read counts
    04 57    0A      E1    0A01           BBC     #IO$V_CLR_COUNT,R7,10$                  ; Br if not clear count request
    21 A3    0A      90    0A05           MOVB    S^#XD_FC_V_RCGERR,IRP$B_XDFUNC(R3) ; Else, Read and Clear
          00B2       31    0A09   10$:    BRW     QUEPKT                                  ; Queue packet to driver
                           0A0C
    2E 57    06      E1    0A0C   20$:    BBC     #IO$V_RD_MEM,R7,40$                     ; Br if read tss request
                           0A10   ;
                           0A10   ; Read GSS address - modifier RD_MEM
                           0A10   ;
    51    04 AC      9A    0A10           MOVZBL  P2(AP),R1                               ; Get GSS address
       1F    51      91    0A14           CMPB    R1,S^#31                                ; Is GSS address too big?
             22      1A    0A17           BGTRU   30$                                     ; Br if yes - error
    3E A3    51      90    0A19           MOVB    R1,IRP$L_MEDIA+6(R3)                    ; Save GSS address in IRP
       51    04      9A    0A1D           MOVZBL  S^#4,R1                                 ; Check access to P1 buffer
          00D7       30    0A20           BSBW    CHECK_P1                                ; Check P1 buffer
```

```
           00E7  30  0A23  2411        BSBW    ALLOC_P2BUF                          ; Allocate a P2 buffer
          12 50  E9  0A26  2412        BLBC    R0,30$                               ; Br if error
     51   2C A3  D0  0A29  2413        MOVL    IRP$L_SVAPTE(R3),R1                  ; Get system buffer address
  04 A1   38 A3  D0  0A2D  2414        MOVL    IRP$L_MEDIA(R3),P2B_L_BUFFER(R1)     ; Set user buffer VA
          38 A3  D4  0A32  2415        CLRL    IRP$L_MEDIA(R3)                      ; Indicate NOT a counter IRP
  21 A3      0C  90  0A35  2416        MOVB    S^#XD_FC_V_RDGSS,IRP$B_XDFUNC(R3)    ; Set function request
             CE  11  0A39  2417        BRB     10$                                  ; Queue packet
                     0A3B  2418
           F897  31  0A3B  2419 30$:   BRW     ABORTIO                              ; Abort the I/O request
                     0A3E  2420
                     0A3E  2421 ;
                     0A3E  2422 ; Read controller parameters - no modifier
                     0A3E  2423 ;
     51      08  3C  0A3E  2424 40$:   MOVZWL  S^#8,R1                              ; Size of P1 buffer if present
           008A  30  0A41  2425        BSBW    CHECK_BUFS                           ; Check P1 and P2 buffers
  3C A3      51  D0  0A44  2426        MOVL    R1,IRP$L_MEDIA+4(R3)                 ; Save user P2 buffer length
             42  13  0A48  2427        BEQL    60$                                  ; Br if no P2 buffer
  51    0042 8F  3C  0A4A  2428        MOVZWL  #LINE_PRM_BUFSIZ,R1                  ; Set size of required buffer
           00BB  30  0A4F  2429        BSBW    ALLOC_P2BUF                          ; Allocate a P2 buffer
          E6 50  E9  0A52  2430        BLBC    R0,30$                               ; Br if error
     51   2C A3  D0  0A55  2431        MOVL    IRP$L_SVAPTE(R3),R1                  ; Get system P2 buffer address
  04 A1      52  D0  0A59  2432        MOVL    R2,P2B_L_BUFFER(R1)                  ; Set user's P2 buffer address
     51 F617 CF  9E  0A5D  2433        MOVAB   LINE_PARAM,R1                        ; Get address of return table
     54      55  D0  0A62  2434        MOVL    R5,R4                                ; Get UCB address
           1781  30  0A65  2435        BSBW    RETURN_P2                            ; Return the P2 buffer data
  42 A3      01  B0  0A68  2436        MOVW    S^#SS$_NORMAL,IRP$W_CSTATUS(R3)      ; Assume success
     50   32 A3  B0  0A6C  2437        MOVW    IRP$W_BCNT(R3),R0                    ; Assume user has good size buffer
     50   3C A3  B1  0A70  2438        CMPW    IRP$L_MEDIA+4(R3),R0                 ; Is user buffer large enough?
             0E  1E  0A74  2439        BGEQU   50$                                  ; Br if yes
  42 A3 0601 8F  B0  0A76  2440        MOVW    #SS$_BUFFEROVF,IRP$W_CSTATUS(R3)     ; Return partial success
     50   3C A3  B0  0A7C  2441        MOVW    IRP$L_MEDIA+4(R3),R0                 ; Set size of return
  32 A3      50  B0  0A80  2442        MOVW    R0,IRP$W_BCNT(R3)                    ;       ...
  50     50  10  78  0A84  2443 50$:   ASHL    #16,R0,R0                            ; Shift return buffer size
     50   42 A3  B0  0A88  2444        MOVW    IRP$W_CSTATUS(R3),R0                 ; Set return status
                     0A8C  2445
     52   38 A3  D0  0A8C  2446 60$:   MOVL    IRP$L_MEDIA(R3),R2                   ; Retrieve P1 buffer address
             04  13  0A90  2447        BEQL    70$                                  ; Br if none
     62   40 A5  7D  0A92  2448        MOVQ    UCB$B_DEVCLASS(R5),(R2)              ; Else, return characteristics
     51   44 A5  D0  0A96  2449 70$:   MOVL    UCB$L_DEVDEPEND(R5),R1               ; Get device dependend info
    00000000'GF  17  0A9A  2450        JMP     G^EXE$FINISHIO                       ; Complete the I/O request
```

```
                     0AA0          .SBTTL  SENSE_MODEM,  Perform SENSEMODE READ_MODEM FDT processing
                     0AA0
                     0AA0   ;++
                     0AA0   ; SENSE_MODEM - Perform SENSEMODE READ_MODEM FDT processing
                     0AA0   ;
                     0AA0   ; Functional description:
                     0AA0   ;
                     0AA0   ; This routine performs all FDT checking for a SENSEMODE read modem request.
                     0AA0   ; The P1 buffer is checked for write access and the request is queued to the
                     0AA0   ; driver. When complete the P1 sensemode information is returned in IRP$L_MEDIA
                     0AA0   ; and IRP$L_MEDIA+4.
                     0AA0   ;
                     0AA0   ; Inputs:
                     0AA0   ;
                     0AA0   ;       R3 = IRP address
                     0AA0   ;       R4 = PCB address
                     0AA0   ;       R5 = UCB address
                     0AA0   ;       R7 = Function code
                     0AA0   ;
                     0AA0   ; Outputs:
                     0AA0   ;
                     0AA0   ;       R0 = status return for request
                     0AA0   ;
                     0AA0   ;       R3-R5 are preserved.
                     0AA0   ;
                     0AA0   ;--
                     0AA0
                     0AA0   SENSE_MODEM:                                            ; Process read modem request
     51    04    3C  0AA0          MOVZWL  S^#4,R1                                 ; Size of P1 buffer
           55    10  0AA3          BSBB    CHECK_P1                                ; Check access to P1
                     0AA5                                                          ; (no return on no access)
         0065    30  0AA5          BSBW    ALLOC_P2BUF                             ; Allocate a P2 buffer
        03 50    E8  0AA8          BLBS    R0,10$                                  ; Br if success
         F827    31  0AAB          BRW     ABORTIO                                 ; Else, abort the I/O request
                     0AAE
     51 2C A3    D0  0AAE   10$:   MOVL    IRP$L_SVAPTE(R3),R1                     ; Get system buffer address
  04 A1 38 A3    D0  0AB2          MOVL    IRP$L_MEDIA(R3),P2B_L_BUFFER(R1)        ; Set user buffer VA
        38 A3    D4  0AB7          CLRL    IRP$L_MEDIA(R3)                         ; Indicate NOT a counter IRP
  21 A3    06    90  0ABA          MOVB    S^#XD_FC_V_RDMODEM,IRP$B_XDFUNC(R3)     ; Set function request
                     0ABE          ;BRB    QUEPKT                                  ; Give request to driver
                     0ABE
                     0ABE   ;
                     0ABE   ; I/O Request packet to driver
                     0ABE   ;
                     0ABE   QUEPKT:                                                 ; Queue packet to driver
                     0ABE          SETIPL  UCB$B_FIPL(R5)                          ; Raise IPL to fork IPL
  00000000'GF    16  0AC2          JSB     G^IOC$INITIATE                          ; Initiate I/O request
  00000000'GF    17  0AC8          JMP     G^EXE$QIORETURN                         ; Lower IPL, and RET
```

```
                          0ACE          .SBTTL  CHECK_BUFS, Check P1 and P2 buffers for write access
                          0ACE
                          0ACE  ;++
                          0ACE  ; CHECK_BUFS - Check P1 and P2 buffers for write access
                          0ACE  ;
                          0ACE  ; Functional description:
                          0ACE  ;
                          0ACE  ; This routines checks the P1 and P2 buffers for write access if supplied.
                          0ACE  ;
                          0ACE  ; Inputs:
                          0ACE  ;
                          0ACE  ;       R1 = Size of P1 buffer needed for write access
                          0ACE  ;       R3 = IRP address
                          0ACE  ;       R4 = PCB address
                          0ACE  ;       R5 = UCB address
                          0ACE  ;       R7 = Function code
                          0ACE  ;       R9 = CDB address
                          0ACE  ;
                          0ACE  ; Outputs:
                          0ACE  ;
                          0ACE  ;       R1 = Length of P2 buffer (zero if no P2 buffer)
                          0ACE  ;       R2 = Address of P2 buffer in user's process space
                          0ACE  ;
                          0ACE  ;       R0-R2 are destroyed.
                          0ACE  ;
                          0ACE  ;       No RETURN on NO ACCESS
                          0ACE  ;
                          0ACE  ; Implicit Outputs:
                          0ACE  ;
                          0ACE  ;       IRP$V_FUNC bit set in IRP$W_STS by EXE$READCHK subroutine.
                          0ACE  ;
                          0ACE  ;--
                          0ACE
                          0ACE  CHECK_BUFS:
             2A    10     0ACE          BSBB    CHECK_P1                        ; Check P1 buffer
                          0AD0  CHECK_P2:
             51    D4     0AD0          CLRL    R1                              ; Assume no P2 buffer desc
    52    04 AC    D0     0AD2          MOVL    P2(AP),R2                       ; Get address of P2 desc
             1B    13     0AD6          BEQL    10$                             ; Br if no P2
                          0AD8          IFNORD  #8,(R2),ACCESS                  ; Br if no access
       51    62    3C     0ADE          MOVZWL  (R2),R1                         ; Get length of buffer
       51    01    CA     0AE1          BICL    #1,R1                           ; Must be multiple of 2 bytes
             0D    13     0AE4          BEQL    10$                             ; Br if zero
    50    04 A2    D0     0AE6          MOVL    DSC$A_POINTER(R2),R0            ; Get buffer address
     00000000'GF   16     0AEA          JSB     G^EXE$READCHK                   ; Check write access to buffer
                          0AF0                                                  ; (no return no access)
                          0AF0                                                  ; Also sets IRP$V_FUNC in IRP
       52    50    D0     0AF0          MOVL    R0,R2                           ; Copy buffer address
             05           0AF3  10$:    RSB                                     ; Return to caller
                          0AF4
       50    0C    3C     0AF4  ACCESS: MOVZWL  #SS$_ACCVIO,R0                  ; Return access violation
           F7DB    31     0AF7          BRW     ABORTIO                         ; Abort the I/O request
```

```
                            0AFA        .SBTTL  CHECK_P1,  Check P1 buffer address for write access
                            0AFA
                            0AFA  ;++
                            0AFA  ; CHECK_P1 - Check P1 buffer address for write access
                            0AFA  ;
                            0AFA  ; Functional description:
                            0AFA  ;
                            0AFA  ;  This routine checks the P1 buffer and if okay, the buffer address
                            0AFA  ;  is saved in IRP$L_MEDIA of the IRP.
                            0AFA  ;
                            0AFA  ; Inputs:
                            0AFA  ;
                            0AFA  ;         R1 = Size of buffer for write access
                            0AFA  ;         R3 = IRP address
                            0AFA  ;         R4 = PCB address
                            0AFA  ;         R5 = UCB address
                            0AFA  ;         R7 = Function code
                            0AFA  ;         R9 = CDB address
                            0AFA  ;
                            0AFA  ; Outputs:
                            0AFA  ;
                            0AFA  ;         R0 is destroyed.
                            0AFA  ;
                            0AFA  ;         No RETURN on NO ACCESS.
                            0AFA  ;
                            0AFA  ; Implicit Outputs:
                            0AFA  ;
                            0AFA  ;         IRP$L_MEDIA(R3) = User P1 buffer address.
                            0AFA  ;         IRP$V_FUNC bit set in IRP$W_STS by EXE$READCHK subroutine.
                            0AFA  ;
                            0AFA  ;--
                            0AFA
                            0AFA  CHECK_P1:
          38 A3       D4    0AFA          CLRL    IRP$L_MEDIA(R3)          ; Assume no P1 buffer
       50    6C       D0    0AFD          MOVL    P1(AP),R0                ; Get address of user buffer
             0A       13    0B00          BEQL    10$                      ; Br if none
     00000000'GF      16    0B02          JSB     G^EXE$READCHK            ; Check access to buffer
                            0B08                                           ; (No return - no access)
     38 A3   50       D0    0B08          MOVL    R0,IRP$L_MEDIA(R3)       ; Save P1 buffer address in IRP
                      05    0B0C  10$:    RSB                              ; Retur  o caller
```

```
                        0B0D            .SBTTL  ALLOC_P2BUF,  Allocate a P2 buffer and charge user's quota
                        0B0D
                        0B0D    ;++
                        0B0D    ; ALLOC_P2BUF - Allocate a P2 buffer and charge user's quota
                        0B0D    ;
                        0B0D    ; Functional description:
                        0B0D    ;
                        0B0D    ; This routine allocates a system buffer and returns the address in the IRP at
                        0B0D    ; IRP$L_SVAPTE.  The size of the allocation, including buffer header must
                        0B0D    ; be at least 24 bytes in length.
                        0B0D    ;
                        0B0D    ; Inputs:
                        0B0D    ;
                        0B0D    ;       R1 = Size of allocation desired
                        0B0D    ;       R3 = IRP address
                        0B0D    ;
                        0B0D    ; Outputs:
                        0B0D    ;
                        0B0D    ;       R0 = status of request
                        0B0D    ;
                        0B0D    ;       R1-R5 are preserved.
                        0B0D    ;
                        0B0D    ; Implicit Outputs:
                        0B0D    ;
                        0B0D    ;       IRP$L_SVAPTE(R3) = address of system buffer
                        0B0D    ;       IRP$W_BOFF(R3) = byte count charged to user's process
                        0B0D    ;       IRP$W_BCNT(R3) = original byte count requested
                        0B0D    ;
                        0B0D    ;       All parts of the P2 buffer header are initialized, except for the
                        0B0D    ;       user's P2 buffer address.
                        0B0D    ;
                        0B0D    ;--
                        0B0D
                        0B0D    ALLOC_P2BUF:                                            ; Allocate a non-paged buffer
              51   D5   0B0D            TSTL    R1                                      ; Zero length buffer?
              50   13   0B0F            BEQL    30$                                     ; Br if yes
              0E   BB   0B11            PUSHR   #^M<R1,R2,R3>                           ; Save registers
       32 A3  51   B0   0B13            MOVW    R1,IRP$W_BCNT(R3)                       ; Save original byte count
          0C  51   D1   0B17            CMPL    R1,S^#24-P2B_C_LENGTH                   ; Is buffer big enough?
              03   1A   0B1A            BGTRU   5$                                      ; Br if yes
          51  0C   D0   0B1C            MOVL    S^#24-P2B_C_LENGTH,R1                   ; Else, set size to minimum
          51  0C   C0   0B1F    5$:     ADDL2   S^#P2B_C_LENGTH,R1                      ; Add in size of header
   00000000'GF     16   0B22            JSB     G^EXE$BUFQUOPRC                         ; Check for buffered quota
          0D  50   E9   0B28            BLBC    R0,10$                                  ; Branch if quota bad
                        0B2B    ;
                        0B2B    ; Quota OKAY, allocate buffer and copy info.
                        0B2B    ;
              51   DD   0B2B            PUSHL   R1                                      ; Save size to charge user
   00000000'GF     16   0B2D            JSB     G^EXE$ALONONPAGED                       ; Go allocate a buffer
          05  50   E8   0B33            BLBS    R0,20$                                  ; Br if success
              8E   D5   0B36            TSTL    (SP)+                                   ; Pop saved size
              0E   BA   0B38    10$:    POPR    #^M<R1,R2,R3>                           ; Restore registers
                   05   0B3A            RSB                                             ; Return with error code in R0
                        0B3B    ;
                        0B3B    ; System buffer allocated decrement user's quota
                        0B3B    ;
           53 8ED0      0B3B    20$:    POPL    R3                                      ; Restore user quota charge
```

```
      62    0C A2   9E  0B3E  2652        MOVAB   P2B_T_DATA(R2),P2B_L_POINTER(R2) ; Set address to start of data
      08 A2    53   B0  0B42  2653        MOVW    R3,P2B_W_SIZE(R2)               ; Save buffer size in buffer
      0A A2    13   90  0B46  2654        MOVB    S^#DYN$C_BUFIO,P2B_B_TYPE(R2)   ; Set structure type
         50    52   D0  0B4A  2655        MOVL    R2,R0                           ; Save P2 char buf addr
   52   0080 C4     D0  0B4D  2656        MOVL    PCB$L_JIB(R4),R2                ; Get JIB address
      20 A2    53   C2  0B52  2657        SUBL    R3,JIB$L_BYTCNT(R2)             ; Decrement user's quota
               0E   BA  0B56  2658        POPR    #^M<R1,R2,R3>                   ; Restore registers
      2C A3    50   D0  0B58  2659        MOVL    R0,IRP$L_SVAPTE(R3)             ; Save P2 buffer address in IRP
   30 A3    08 A0   B0  0B5C  2660        MOVW    P2B_W_SIZE(R0),IRP$W_BOFF(R3)   ; Return buffer size in IRP
         50    01   3C  0B61  2661 30$:   MOVZWL  S^#SS$_NORMAL,R0                ; Set success
                    05  0B64  2662        RSB                                     ; Return to caller
```

```
                     0B65          .SBTTL  STARTIO,  Start I/O routine
                     0B65
                     0B65  ;++
                     0B65  ; STARTIO - Start a transmit, receive, or set mode operation
                     0B65  ;
                     0B65  ; Functional description:
                     0B65  ;
                     0B65  ; This routine is called when an IRP is given to EXE$QIODRVPKT.
                     0B65  ; From here, depending on the type of function., the appropriate
                     0B65  ; routine is called.
                     0B65  ;
                     0B65  ; Inputs:
                     0B65  ;
                     0B65  ;        R3 = IRP address (I/O request packet)
                     0B65  ;        R5 = UCB address (unit control block)
                     0B65  ;        R9 = CDB address (only if circuit request)
                     0B65  ;
                     0B65  ;
                     0B65  ;        IPL = FIPL
                     0B65  ;
                     0B65  ; Outputs:
                     0B65  ;
                     0B65  ;        R3,R5 are preserved.
                     0B65  ;
                     0B65  ;        The routine must preserve all registers except R0-R2 and R4.
                     0B65  ;
                     0B65  ;--
                     0B65
                     0B65  STARTIO:                                              ; Process an I/O packet
                     0B65          PUSHQ   R6                                    ; Save registers
      50    04   90  0B68          MOVB    #JNL_QUE,R0                           ; Set journal type = QUE
          195F   30  0B6B          BSBW    XD_FILL_JNX                           ; Fill the journal buffer
                     0B6E          POPQ    R6                                    ; Restore registers
                     0B71
   51    21 A3   9A  0B71          MOVZBL  IRP$B_XDFUNC(R3),R1                   ; Get the function code
      01    51   91  0B75          CMPB    R1,S^#XD_FC_V_INITC                   ; Startup device?
            0A   13  0B78          BEQL    10$                                   ; Branch if so
   50  0084 8F   3C  0B7A          MOVZWL  #SS$_DEVOFFLINE,R0                    ; Assume device not running
            10   E0  0B7F          BBS     #XM$V_ERR_FATAL,-                     ; Br if hung
      42 44 A5       0B81                  UCB$L_DEVDEPEND(R5),DRV_DONE          ;
                     0B84  10$:    $DISPATCH       R1,TYPE=B,-                   ; Dispatch on function request
                     0B84          <-      ;function       action
                     0B84
                     0B84                  <XD_FC_V_INITT  START_TRIB>,-         ; Start tributary
                     0B84                  <XD_FC_V_INITC  START_DEV>,-          ; Start DMP11 controller
                     0B84                  <XD_FC_V_STOPT  TRIB_ACTIVE>,-        ; Stop tributary
                     0B84                  <XD_FC_V_STOPC  DEV_ACTIVE>,-         ; Stop DMP11 controller
                     0B84                  <XD_FC_V_CHGC   DEV_ACTIVE>,-         ; Change controller parameters
                     0B84                  <XD_FC_V_CHGT   TRIB_ACTIVE>,-        ; Change tributary parameters
                     0B84                  <XD_FC_V_RDMODEM DEV_ACTIVE>,-        ; Read modem register
                     0B84                  <XD_FC_V_RDTERR TRIB_ERRS>,-          ; Read tributary error counts
                     0B84                  <XD_FC_V_RDGERR DEV_ACTIVE>,-         ; Read global error counts
                     0B84                  <XD_FC_V_RCTERR TRIB_ERRS>,-          ; Read and clear tributary ctrs
                     0B84                  <XD_FC_V_RCGERR DEV_ACTIVE>,-         ; Read and clear global counts
                     0B84                  <XD_FC_V_RDTSS  TRIB_ACTIVE>,-        ; Read TSS location
                     0B84                  <XD_FC_V_RDGSS  DEV_ACTIVE>,-         ; Read GSS location
                     0B84                  <XD_FC_V_WTMODEM DEV_ACTIVE>,-        ; Write modem request
```

```
                 0B84                 <XD_FC_V_POLSS  TRIB_ACTIVE>,-   ; Read polling sub-state
                 0B84           >
                 0B84   ;
                 0B84   ; Other request types - if we fall thru case instruction.
                 0B84   ;
                 0B84           BUG_CHECK NOBUFPCKT,FATAL              ; Fatal error
                 0BB8   ;
                 0BB8   ; Start up controller
                 0BB8   ;
                 0BB8   START_DEV:                                     ; Startup controller
      0F    10   0BB8           BSBB    STARTUP                        ; Startup device
   01 50    E9   0BBA           BLBC    R0,10$                         ; Br if error
            05   0BBD           RSB                                    ; Return to caller
                 0BBE   ;
                 0BBE   ; Error on startup - shutdown the controller
                 0BBE   ;
      50    DD   0BBE   10$:    PUSHL   R0                             ; Else, save status return
    1179    30   0BC0           BSBW    SHUTDOWN_DEV                   ; Shutdown the device
      50  8ED0   0BC3           POPL    R0                             ; Restore status return
                 0BC6           ;BRB    DRV_DONE                       ; Complete request in error
                 0BC6
                 0BC6   DRV_DONE:                                      ; Driver request complete
    08F8    31   0BC6           BRW     REQ_COM_ERR                    ; Complete the request
```

```
                       0BC9            .SBTTL  STARTUP, Device initialization routine
                       0BC9
                       0BC9    ;++
                       0BC9    ; STARTUP - Device initialization routine
                       0BC9    ;
                       0BC9    ; Functional description:
                       0BC9    ;
                       0BC9    ; The configuration is read from the P1 and P2 buffers and a MODE DEFINITION
                       0BC9    ; command is issued to the DMP11 device.  The buffers which have been set
                       0BC9    ; up for the common receive pool are then given to the device.
                       0BC9    ;
                       0BC9    ; Inputs:
                       0BC9    ;
                       0BC9    ;       R3 = IRP address
                       0BC9    ;       R5 = UCB address
                       0BC9    ;
                       0BC9    ;       IPL = FIPL
                       0BC9    ;
                       0BC9    ; Outputs:
                       0BC9    ;
                       0BC9    ;       R0 = status return for startup operation
                       0BC9    ;
                       0BC9    ;       R3,R5 are preserved
                       0BC9    ;
                       0BC9    ;--
                       0BC9
                       0BC9    STARTUP:                                        ; Controller startup processing
           00     E1   0BC9            BBC     #UCB$V_XD_INITED,-              ; Br if not inited yet
      0C 68 A5         0BCB                    UCB$W_DEVSTS(R5),1$             ;
 50    02C4 8F    3C   0BCE            MOVZWL  #SS$_DEVACTIVE,R0               ; Else, device already active
         08EB     30   0BD3            BSBW    REQ_COM_ERR                     ; Complete request in error
    50     01     9A   0BD6            MOVZBL  S^#SS$_NORMAL,R0                ; DO NOT STOP CONTROLLER!
           05          0BD9            RSB                                     ; Return to caller
                       0BDA
F56F CF    42 A5  B1   0BDA    1$:     CMPW    UCB$W_DEVBUFSIZ(R5),MAX_W_BUFFER; Is buffer too big?
           04     1B   0BE0            BLEQU   2$                              ; Br if not, okay
    50     14     3C   0BE2            MOVZWL  #SS$_BADPARAM,R0                ; Else, set error condition
           05          0BE5            RSB                                     ; Return to caller
                       0BE6
                       0BE6    2$:     ;
                       0BE6            ; For u-VAX I, we must have a contiguous buffer area!
                       0BE6            ;
                       0BE6            CPUDISP <<790,9$>,-
                       0BE6                    <780,9$>,-
                       0BE6                    <750,9$>,-
                       0BE6                    <730,9$>,-
                       0BE6                    <UV1,3$>>
                       0C00    3$:
 52    012C C5    D0   0C00            MOVL    UCB$L_XD_UV1BUF(R5),R2          ; Is the buffer area present?
           06     12   0C05            BNEQ    5$                              ; Br if not, error
 50    0124 8F    3C   0C07            MOVZWL  #SS$_INSFMEM,R0                 ; Else, set error condition
           05          0C0C            RSB                                     ; Return to caller
                       0C0D
                       0C0D    5$:     ;
                       0C0D            ; Compute physical/virtual address of buffers in buffer area.
                       0C0D            ;
  52    0C A2     9E   0C0D            MOVAB   12(R2),R2                       ; Get start of buffer area
```

```
     50  52    15    09  EF  0C11               EXTZV   #VA$V_VPN,#VA$S_VPN,R2,R0            ; Get virtual page number
         51 00000000'GF  D0  0C16               MOVL    G^MMG$GL_SPTBASE,R1                 ; Get the base address of SPTs
              50  6140   D0  0C1D               MOVL    (R1)[R0],R0                         ; Get the PTE contents
  51  52   FFFFFE00 8F   CB  0C21               BICL3   #^C<VA$M_BYTE>,R2,R1                ; Get buffer offset (BA00-BA08)
                             0C29               ASSUME  PTE$S_PFN GE 13
     51  0D    09    50  F0  0C29               INSV    R0,#9,#13,R1                        ; Copy BA09-BA21
                     50  D4  0C2E               CLRL    R0                                  ; Use R0 as receive buffer index
       00FC C540     51  D0  0C30   7$:         MOVL    R1,UCB$L_XD_RCV_PA(R5)[R0]          ; Save RECV Physical address
       0114 C540     52  D0  0C36               MOVL    R2,UCB$L_XD_RCV_VA(R5)[R0]          ; Save RECV Virtual address
     51  00000400   8F   C0  0C3C               ADDL    #MAX_BUFSIZ_UV1,R1                  ; Skip to next buffer
     52  00000400   8F   C0  0C43               ADDL    #MAX_BUFSIZ_UV1,R2                  ;       ...
            E2 50    04  F2  0C4A               AOBLSS  #MAX_RCV_UV1,R0,7$                  ; Loop if more
                     50  D4  0C4E               CLRL    R0                                  ; Use R0 as xmit buffer index
       010C C540     51  D0  0C50   8$:         MOVL    R1,UCB$L_XD_XMT_PA(R5)[R0]          ; Save XMIT Physical address
       0124 C540     52  D0  0C56               MOVL    R2,UCB$L_XD_XMT_VA(R5)[R0]          ; Save XMIT Virtual address
     51  00000400   8F   C0  0C5C               ADDL    #MAX_BUFSIZ_UV1,R1                  ; Skip to next buffer
     52  00000400   8F   C0  0C63               ADDL    #MAX_BUFSIZ_UV1,R2                  ;       ...
            E2 50    02  F2  0C6A               AOBLSS  #MAX_XMT_UV1,R0,8$                  ; Loop if more
                             0C6E
                             0C6E   9$:
  44 A5    18    08  00  F0  0C6E               INSV    #0,#8,#24,UCB$L_DEVDEPEND(R5)       ; Reset status and error flags
     02A0 C5     0C A3   D0  0C74               MOVL    IRP$L_PID(R3),UCB$L_XD_PID(R5)      ; Save starter's PID
                             0C7A               ASSUME  UCB$B_XD_OUTTIM EQ UCB$B_XD_INTIM+1
           02A5 C5       B4  0C7A               CLRW    UCB$B_XD_INTIM(R5)                  ; Init timer cells
                             0C7E               SETBIT  #UCB$V_XD_INITING,UCB$W_DEVSTS(R5) ; Indicate device initing
                             0C83   ;
                             0C83   ; Initialize the buffer and I/O request queue heads
                             0C83   ;
           50    06      3C  0C83               MOVZWL  #UCB$C_XD_QUEUES,R0                 ; Set number of queue heads
     52  0094 C5         9E  0C86               MOVAB   UCB$Q_XD_QUEUES(R5),R2              ; Set address of first head
           82    62      9E  0C8B   10$:        MOVAB   (R2),(R2)+                          ; Init forward link
        82    FC A2      9E  0C8E               MOVAB   -4(R2),(R2)+                        ; Init backward link
           F6 50         F5  0C92               SOBGTR  R0,10$                              ; Br if more queues
                             0C95   ;
                             0C95   ; Initialize the transmit and receive mapping info vector.
                             0C95   ;
           0298 C5       D4  0C95               CLRL    UCB$L_XD_RUN(R5)                    ; Initialize tribs in run
           029C C5       D4  0C99               CLRL    UCB$L_XD_HALT(R5)                   ; Initialize halted tribs
           0134 C5       94  0C9D               CLRB    UCB$B_XD_TRB_CNT(R5)                ; Init number of active tribs
           50    0E      3C  0CA1               MOVZWL  #MAX_RCV+MAX_XMT,R0                 ; Set number of rcv and xmit slots
                             0CA4               ASSUME  UCB$L_XD_RCV_MAP+<4*MAX_RCV> EQ UCB$L_XD_XMT_MAP
     51  00C4 C5         DE  0CA4               MOVAL   UCB$L_XD_RCV_MAP(R5),R1             ; Get map vector address
           81    01      CE  0CA9   15$:        MNEGL   #1,(R1)+                            ; Indicate no map info.
           FA 50         F5  0CAC               SOBGTR  R0,15$                              ; Br if more map slots
                             0CAF   ;
                             0CAF   ; Initialize vector areas of UCB
                             0CAF   ;
     51  0138 C5         DE  0CAF               MOVAL   UCB$L_XD_VECS(R5),R1                ; Get address of start of vectors
     52  0058 8F         3C  0CB4               MOVZWL  #UCB$C_XD_VECS,R2                   ; Get number of longwords
                 81      D4  0CB9   20$:        CLRL    (R1)+                               ; Initialize vector area
           FB 52         F5  0CBB               SOBGTR  R2,20$                              ; Loop if more
                             0CBE
              32 A3      B4  0CBE               CLRW    IRP$W_BCNT(R3)                      ; No read buffer for I/O post
     0136 C5  42 A3      B0  0CC1               MOVW    IRP$W_QUOTA(R3),UCB$W_XD_QUOTA(R5) ; Store receive buffer quota
              42 A3      B4  0CC7               CLRW    IRP$W_QUOTA(R3)                     ; No buffer quota to return
     54       24 A5      D0  0CCA               MOVL    UCB$L_CRB(R5),R4                    ; Get CRB address
                             0CCE               ;
```

```
                              0CCE          ; For uVAX I, the max receives and max transmits are fixed,
                              0CCE          ; and there are no map registers on uVAX I!
                              0CCE          ;
                              0CCE          CPUDISP <<790,30$>,-
                              0CCE                   <780,30$>,-
                              0CCE                   <750,30$>,-
                              0CCE                   <730,30$>,-
                              0CCE                   <UV1,70$>>
                              0CE8
                              0CE8  30$:    ; All except uVAX I
        0132 C5    07    90   0CE8          MOVB    #MAX_RCV,UCB$B_XD_RCV_MAX(R5)      ; Set max concurrent receives
        0133 C5    07    90   0CED          MOVB    #MAX_XMT,UCB$B_XD_XMT_MAX(R5)      ; Set max concurrent transmits
                              0CF2  ;
                              0CF2  ; Allocate map registers for recieve buffers.
                              0CF2  ; The unbuffered datapath (DP0) is used for all I/O's due to the fact that
                              0CF2  ; the controller can initiate retransmissions but on the 11/780, the datapath
                              0CF2  ; requires purging before it can be reused.
                              0CF2  ;
       7E A5    42 A5    B0   0CF2          MOVW    UCB$W_DEVBUFSIZ(R5),UCB$W_BCNT(R5) ; Set buffer size
    7C A5    01FF 8F     B0   0CF7          MOVW    #511,UCB$W_BOFF(R5)                ; Set worst case byte offset
                              0CFD          ASSUME  VEC$W_MAPREG+2 EQ VEC$B_NUMREG
                              0CFD          ASSUME  VEC$B_NUMREG+1 EQ VEC$B_DATAPATH
               34 A4     D4   0CFD          CLRL    CRB$L_INTD+VEC$W_MAPREG(R4)        ; Clear map register + datapath
                              0D00          PUSHQ   R6                                 ; Save R6,R7
       56   00C4 C5      DE   0D03          MOVAL   UCB$L_XD_RCV_MAP(R5),R6            ; Get mapping slot address
       57   0132 C5      9A   0D08          MOVZBL  UCB$B_XD_RCV_MAX(R5),R7            ; Get number of rcv map slots
        00000000'GF      16   0D0D  40$:    JSB     G^IOC$ALLOUBAMAP                   ; Allocate a set of map registers
               07 50     E9   0D13          BLBC    R0,50$                             ; Br if unavailable
       86      34 A4     D0   0D16          MOVL    CRB$L_INTD+VEC$W_MAPREG(R4),(R6)+  ; Save receive map info
               F0 57     F5   0D1A          SOBGTR  R7,40$                             ; Br if more slots available
                              0D1D
                              0D1D  50$:    POPQ    R6                                 ; Restore R6,R7
               13 50     E8   0D20          BLBS    R0,90$                             ; Br if okay so far
       50   0344 8F      3C   0D23          MOVZWL  #SS$_INSFMAPREG,R0                 ; Set insufficient map registers
                         05   0D28          RSB                                        ; Return with error
                              0D29
                              0D29  70$:    ; UV1
        0132 C5    04    90   0D29          MOVB    #MAX_RCV_UV1,UCB$B_XD_RCV_MAX(R5)  ; Set max concurrent receives
        0133 C5    02    90   0D2E          MOVB    #MAX_XMT_UV1,UCB$B_XD_XMT_MAX(R5)  ; Set max concurrent transmits
             50    01    9A   0D33          MOVZBL  S^#SS$_NORMAL,R0                   ; Success
                              0D36
                              0D36  90$:    ;
                              0D36          ; Common code path
                              0D36          ;
             42 A5       A7   0D36          DIVW3   UCB$W_DEVBUFSIZ(R5),-              ; Compute quota for rcv buffers
       51   0136 C5           0D39                  UCB$W_XD_QUOTA(R5),R1              ;
        0132 C5    51    91   0D3D          CMPB    R1,UCB$B_XD_RCV_MAX(R5)            ; Is number less than max?
                   05    1E   0D42          BGEQU   99$                                ; Br if not - leave max alone
        0132 C5    51    90   0D44          MOVB    R1,UCB$B_XD_RCV_MAX(R5)            ; Else, reduce max rcvs allowed
                              0D49  99$:
                              0D49
                              0D49  ;  Now the DMP11 device is master cleared.  A timeout is setup to occur
                              0D49  ;  in approximately one to two seconds.  That should give the DMP enough
                              0D49  ;  time to run the micro-diagnostics. After the timeout, we will procede
                              0D49  ;  and initialize the device.
                              0D49  ;
                              0D49  ;  We will use the WFIKPCH macro which will wait for interrupt or timeout,
```

```
                              0D49  ;  but the interrupt should never occur - only the timeout!  It was a handy
                              0D49  ;  way to schedule a 2 second timeout without a lot of fuss and bother.
                              0D49  ;
                              0D49          ASSUME  IDB$L_CSR EQ 0
           54    2C B4    D0  0D49          MOVL    @CRB$[_INTD+VEC$L_IDB(R4),R4       ; Get CSR address
                              0D4D          DSBINT  UCB$B_DIPL(R5)                     ; Disable device interrupts
                              0D54
        01 A4    40 8F    90  0D54          MOVB    #XD_BSEL1_M_MCLR,BSEL1(R4)         ; Set the master clear bit
                              0D59  ;
                              0D59  ; WFIKPCH does not destroy R0, therefore R0 should still have SS$_NORMAL
                              0D59  ; success return status.
                              0D59  ;
                              0D59          WFIKPCH START_CONT,#2                      ; Wait 2 seconds for master clr
```

```
                                 0D63        .SBTTL  START_CONT, Continue after timeout
                                 0D63
                                 0D63  ;++
                                 0D63  ; START_CONT - Continue after timeout
                                 0D63  ;
                                 0D63  ; Functional description:
                                 0D63  ;
                                 0D63  ; This routine is called by the system after the startup timeout was set
                                 0D63  ; when the device was master cleared.
                                 0D63  ;
                                 0D63  ;
                                 0D63  ; Inputs:
                                 0D63  ;
                                 0D63  ;       R3 = IRP address
                                 0D63  ;       R4 = CSR address
                                 0D63  ;       R5 = UCB address
                                 0D63  ;
                                 0D63  ;       IPL = DIPL
                                 0D63  ;
                                 0D63  ; Outputs:
                                 0D63  ;
                                 0D63  ;       R5 is preserved.
                                 0D63  ;
                                 0D63  ;--
                                 0D63
                                 0D63  START_CONT:                                          ; Continue with startup
                                 0D63          CLRBIT  #UCB$V_TIMOUT,UCB$W_STS(R5)          ; Clear timeout status
  0C A5    000018CC'EF     9E    0D68          MOVAB   FORK_PROC,UCB$L_FPC(R5)              ; Set fork process PC
     01 A4    80 8F        93    0D70          BITB    #XD_BSEL1_M_RUN,BSEL1(R4)            ; Is run bit on?
                    07     13    0D75          BEQL    10$                                  ; Br if not - error
     06 A4    C5 8F        91    0D77          CMPB    #^O305,BSEL6(R4)                     ; Successful master clear?
                    15     13    0D7C          BEQL    20$                                  ; Yes, continue
     47 A5    06 A4        90    0D7E  10$:    MOVB    BSEL6(R4),UCB$L_DEVDEPEND+3(R5)      ; Save BSEL6 on error
     00B8 D5       63      0E    0D83          INSQUE  (R3),@UCB$Q_XD_POST+4(R5)            ; Insert IRP on posting Q
     53     01     15      78    0D88          ASHL    #XD_BSEL2_V_ERR+16,#1,R3             ; Indicate fatal error
     54     03     10      78    0D8C          ASHL    #16,#3,R4                            ; Indicate device error
                 0B10      31    0D90          BRW     SCHED_FORK                           ; Request complete
                                 0D93
                                 0D93  ;
                                 0D93  ; The device seems to be running - give mode definition
                                 0D93  ;
     04 A4         1B      91    0D93  20$:    CMPB    #^O33,BSEL4(R4)                      ; Is this a DMV11?
                   09      12    0D97          BNEQ    30$                                  ; Br if not
     41 A5         17      90    0D99          MOVB    #DT$_DMV11,UCB$B_DEVTYPE(R5)         ; Else, set proper device type
   F3AF CF         10      B0    0D9D          MOVW    #MAX_DMV_TRIB,MAX_W_TRIB             ; Set maximum number of tribs
                                 0DA2  30$:    SETBIT  #UCB$V_XD_INITED,UCB$W_DEVSTS(R5)    ; Indicate device inited
                                 0DA7          CLRBIT  #UCB$V_XD_INITING,UCB$W_DEVSTS(R5)   ; No longer initing
        64         10      88    0DAC          BISB    #XD_BSEL0_M_IEO,BSEL0(R4)            ; Enable output interrupts
                   01      E0    0DAF          BBS     #XM$V_CHR_LOOPB,-                    ; Br if loopback
          03 44 A5               0DB1                  UCB$L_DEVDEPEND(R5),35$              ;
                 0083      31    0DB4          BRW     90$                                  ; Continue
                                 0DB7  35$:
                                 0DB7          $DISPATCH UCB$B_DEVTYPE(R5),TYPE=B,- ; Dispatch on device type
                                 0DB7                  <-      ;function       action
                                 0DB7
                                 0DB7                          <DT$_DMV11, 40$>,-    ; DMV11
                                 0DB7                          <DT$_DMP11, 70$>,-    ; DMP11
```

```
                    0DB7  2989                >
                    0DDA  2990
                    0DDA  2991          ; DMV-11
                    0DDA  2992
        41 8F   90  0DDA  2993 40$:     MOVB    #XD_BSEL1_M_MCLR!XD_BSEL1_M_MAINT,- ; Request Maint mode for DMV
        01 A4       0DDD  2994                  BSEL1(R4)
                    0DDF  2995          TIMEWAIT #9,#XD_BSEL2_M_RDO,SEL2(R4),B  ; Wait for MAINT RDY to be set
                    0E04  2996                                                  ; ... same as RDO bit
        2C 50   E9  0E04  2997          BLBC    R0,50$                          ; Br if ERROR
     02 A4  06  B0  0E07  2998          MOVW    #6,SEL2(R4)                     ; Load LOOP request and clear
                    0E0B  2999                                                  ; "maint ready" bit
                    0E0B  3000          TIMEWAIT #9,#XD_BSEL1_M_RUN,BSEL1(R4),B ; Wait for MAINT RDY to be set
        07 50   E8  0E30  3001          BLBS    R0,90$                          ; Br if success
         FF48   31  0E33  3002 50$:     BRW     10$                             ; Else, process error
                    0E36  3003
                    0E36  3004          ; DMP-11
                    0E36  3005
     01 A4  08  88  0E36  3006 70$:     BISB    #XD_BSEL1_M_LOOPB,BSEL1(R4)     ; Else, enable device loopback
                    0E3A  3007
  0094 C5   63  0E  0E3A  3008 90$:     INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)        ; Insert at front of input Q
         018E   31  0E3F  3009          BRW     LOAD_PORT                       ; Request input port
```

```
                              0E42          .SBTTL  START_TRIB,  Start tributary routine
                              0E42
                              0E42   ;++
                              0E42   ; START_TRIB - Start tributary routine
                              0E42   ;
                              0E42   ; Functional description:
                              0E42   ;
                              0E42   ; This routine is called when a tributary is to be established and started.
                              0E42   ; The polling parameters are initialized also.
                              0E42   ;
                              0E42   ;  Inputs:
                              0E42   ;
                              0E42   ;         R3 = IRP address
                              0E42   ;         R5 = UCB address
                              0E42   ;         R9 = CDB address
                              0E42   ;
                              0E42   ;
                              0E42   ;         IPL = FIPL
                              0E42   ;
                              0E42   ;  Outputs:
                              0E42   ;
                              0E42   ;         R5 is preserved.
                              0E42   ;
                              0E42   ;--
                              0E42
                              0E42   START_TRIB:                                             ; Start tributary
08 0C A9      00      E1      0E42           BBC     #CD_TS_V_ESTAB,CDB_W_STS(R9),10$ ; Br if trib not active
   50   02C4 8F       3C      0E47           MOVZWL  #SS$_DEVACTIVE,R0                ; Else, trib already active
           0D81       31      0E4C           BRW     IO_DONE                          ; Request complete
                              0E4F
                              0E4F   10$:    SETIPL  UCB$B_DIPL(R5)                   ; Sync access to device
  0098 D5     63      0E      0E53           INSQUE  (R3),@UCB$Q_XD_INPUTQ+4(R5)      ; Insert at end of input Q
           0175       31      0E58           BRW     LOAD_PORT                        ; Request the CSR
                              0E5B                                                    ; Let return to IOC$INITIATE
                              0E5B                                                    ; ..clean up IPL
```

```
                          0E5B          .SBTTL  DEV_ACTIVE, Device must be active
                          0E5B
                          0E5B  ;++
                          0E5B  ; DEV_ACTIVE - Device must be active
                          0E5B  ;
                          0E5B  ; Functional description:
                          0E5B  ;
                          0E5B  ; This routine is called to queue a request to the device input queue.
                          0E5B  ; The routine must first verify that the device is active, before the
                          0E5B  ; request can be queued, however. If the device is not active, the request
                          0E5B  ; is posted in error.
                          0E5B  ;
                          0E5B  ; Inputs:
                          0E5B  ;
                          0E5B  ;       R3 = IRP address
                          0E5B  ;       R5 = UCB address
                          0E5B  ;
                          0E5B  ;       IPL = FIPL
                          0E5B  ;
                          0E5B  ; Outputs:
                          0E5B  ;
                          0E5B  ;       None.
                          0E5B  ;
                          0E5B  ;--
                          0E5B
                          0E5B  DEV_ACTIVE:                                              ; Check if device is active
08 68 A5    00      E0    0E5B          BBS     #UCB$V_XD_INITED,UCB$W_DEVSTS(R5),10$ ; Br if device inited
50    20D4 8F       3C    0E60          MOVZWL  #SS$_DEVINACT,R0                     ; Else, return error
         0D68       31    0E65          BRW     IO_DONE                              ; Finish I/O
                          0E68
                          0E68  10$:    SETIPL  UCB$B_DIPL(R5)                       ; Sync access to device
0098 D5     63      0E    0E6C          INSQUE  (R3),@UCB$Q_XD_INPUTQ+4(R5)          ; Insert at end of Q
         015C       31    0E71          BRW     LOAD_PORT                            ; Give request to device
                          0E74                                                       ; Let return to IOC$INITIATE
                          0E74                                                       ; ..clean up IPL
```

```
                    0E74          .SBTTL  TRIB_ACTIVE,  Tributary must be active
                    0E74
                    0E74    ;++
                    0E74    ; TRIB_ACTIVE - Tributary must be active
                    0E74    ;
                    0E74    ; Functional description:
                    0E74    ;
                    0E74    ; This routine is called to queue a request to the device for a tributary.
                    0E74    ; The routine must first verify that the trib is active and if so, queue
                    0E74    ; the request, else it must post the request in error.
                    0E74    ;
                    0E74    ; Inputs:
                    0E74    ;
                    0E74    ;         R3 = IRP address
                    0E74    ;         R5 = UCB address
                    0E74    ;         R9 = CDB address
                    0E74    ;
                    0E74    ;         IPL = FIPL
                    0E74    ;
                    0E74    ; Outputs:
                    0E74    ;
                    0E74    ;         None.
                    0E74    ;
                    0E74    ;--
                    0E74
                    0E74    TRIB_ACTIVE:                                          ; Check if trib is active
08 0C A9   00   E0  0E74            BBS     #CD_TS_V_ESTAB,CDB_W_STS(R9),10$      ; Br if trib established
50   20D4 8F   3C  0E79            MOVZWL  #SS$_DEVINACT,R0                      ; Set error return
        0D4F   31  0E7E            BRW     IO_DONE                               ; Finish the request
                    0E81
                    0E81    10$:    SETIPL  UCB$B_DIPL(R5)                        ; Sync access to device
0098 D5    63   0E  0E85            INSQUE  (R3),@UCB$Q_XD_INPUTQ+4(R5)          ; Insert at end of Q
        0143   31  0E8A            BRW     LOAD_PORT                             ; Load CSR's
                    0E8D                                                          ; Let return to IOC$INITIATE
                    0E8D                                                          ; ..clean up IPL
```

```
                          0E8D            .SBTTL  TRIB_ERRS,  Read trib errors routine
                          0E8D
                          0E8D      ;++
                          0E8D      ; TRIB_ERRS - Read trib errors routine
                          0E8D      ;
                          0E8D      ; Functional description:
                          0E8D      ;
                          0E8D      ; This routine is called to read the tributary error counters.  All
                          0E8D      ; counters are read from the device except for data messages and bytes
                          0E8D      ; transmitted and received.  These counters are 32 bits in length and
                          0E8D      ; are kept by the driver itself.
                          0E8D      ;
                          0E8D      ; Inputs:
                          0E8D      ;
                          0E8D      ;         R3 = IRP address
                          0E8D      ;         R5 = UCB address
                          0E8D      ;         R9 = CDB address
                          0E8D      ;
                          0E8D      ;         IPL = FIPL
                          0E8D      ;
                          0E8D      ; Outputs:
                          0E8D      ;
                          0E8D      ;         The error counters are returned and the request is completed.
                          0E8D      ;
                          0E8D      ;--
                          0E8D
                          0E8D      TRIB_ERRS:
 08 0C A9    00     E0    0E8D            BBS     #CD_TS_V_ESTAB,CDB_W_STS(R9),10$ ; Br if trib established
 50   20D4 8F       3C    0E92            MOVZWL  #SS$_DEVINACT,R0                 ; Set error return
        0D36        31    0E97            BRW     IO_DONE                          ; Finish the request
                          0E9A
   52    2C A3      D0    0E9A      10$:  MOVL    IRP$L_SVAPTE(R3),R2              ; Get system P2 address
            2D      13    0E9E            BEQL    20$                              ; Br if none
 50   F2B3 CF       9E    0EA0            MOVAB   TRIB_COUNTER,R0                  ; Get address of counter codes
      51    62      D0    0EA5            MOVL    (R2),R1                          ; Get address of data
                          0EA8            PUSHQ   R2                               ; Save P2 buffer address + IRP
   52    38 A9      DE    0EAB            MOVAL   CDB_L_CNTS(R9),R2                ; Get address of counts
                          0EAF            ASSUME  CDB_C_CNTS EQ 16                 ; Assume 16 bytes of XD counts
      53    04      3C    0EAF            MOVZWL  S^#4,R3                          ; Set number of counters
                          0EB2
      81    80      B0    0EB2      15$:  MOVW    (R0)+,(R1)+                      ; Set next code
      81    82      D0    0EB5            MOVL    (R2)+,(R1)+                      ; Set next value
         F7 53      F5    0EB8            SOBGTR  R3,15$                           ; Loop if more
                          0EBB
                          0EBB            POPQ    R2                               ; Restore P2 buffer address + IRP
      62    51      D0    0EBE            MOVL    R1,(R2)                          ; Reset address of next byte
   21 A3    09      91    0EC1            CMPB    #XD_FC_V_RCTERR,IRP$B_XDFUNC(R3) ; Is this a clear counter?
            06      12    0EC5            BNEQ    20$                              ; Br if not
         38 A9      7C    0EC7            CLRQ    CDB_L_CNTS(R9)                   ; Else, clear counters
         40 A9      7C    0ECA            CLRQ    CDB_L_CNTS+8(R9)                 ;       ...
                          0ECD
                          0ECD      20$:  SETIPL  UCB$B_DIPL(R5)                   ; Sync access to device
 0098 D5    63      0E    0ED1            INSQUE  (R3),@UCB$Q_XD_INPUTQ+4(R5)      ; Insert at end of Q
        00F7        31    0ED6            BRW     LOAD_PORT                        ; Give to device
                          0ED9                                                     ; Let return to IOC$INITIATE
                          0ED9                                                     ; ..clean up IPL
```

```
                        0ED9         .SBTTL  FILLRCVLIST,  Fill receive buffer list
                        0ED9         .SBTTL  ADDRCVLIST,  Add a buffer to the receive list
                        0ED9
                        0ED9  ;++
                        0ED9  ; FILLRCVLIST - Fill receive buffer list
                        0ED9  ; ADDRCVLIST - Add a buffer to the receive list
                        0ED9  ;
                        0ED9  ; Functional description:
                        0ED9  ;
                        0ED9  ; This routine is entered to make sure that the receive buffer pool
                        0ED9  ; is full.  If it is not, buffers are allocated and queued to the
                        0ED9  ; list until it is.
                        0ED9  ;
                        0ED9  ; Inputs:
                        0ED9  ;
                        0ED9  ;       R2 = Buffer address (ADDRCVLIST only)
                        0ED9  ;       R5 = UCB address
                        0ED9  ;
                        0ED9  ;       IPL = FIPL
                        0ED9  ;
                        0ED9  ; Outputs:
                        0ED9  ;
                        0ED9  ;       R0-R2 are destroyed.
                        0ED9  ;
                        0ED9  ;--
                        0ED9         .ENABL  LSB
                        0ED9  FILLRCVLIST:                                    ; Fill receive buffer list
             00   E1    0ED9         BBC     #UCB$V_XD_INITED,-              ; Exit if not inited
       60 68 A5         0EDB                 UCB$W_DEVSTS(R5),40$            ;
             52   D4    0EDE         CLRL    R2                              ; No buffer here
                        0EE0
                        0EE0  ADDRCVLIST:                                     ; Add a buffer to the receive list
                        0EE0         PUSHQ   R3                              ; Save R3, R4
          42 A5   B1    0EE3  5$:    CMPW    UCB$W_DEVBUFSIZ(R5),-           ; Can new block be allocated?
        0136 C5         0EE6                 UCB$W_XD_QUOTA(R5)              ;
             34   1A    0EE9         BGTRU   20$                             ; Br if no - list filled
             51   D4    0EEB         CLRL    R1                              ; Zero size
        004C 8F   A1    0EED         ADDW3   #RCV_T_DATA+CXB$C_TRAILER,-     ; Compute block size needed
       51  42 A5        0EF1                 UCB$W_DEVBUFSIZ(R5),R1          ;
             52   D5    0EF4         TSTL    R2                              ; Buffer already allocated?
             09   12    0EF6         BNEQ    7$                              ; Br if yes
    00000000'GF   16    0EF8         JSB     G^EXE$ALONONPAGED               ; Allocate nonpaged memory
          17 50   E9    0EFE         BLBC    R0,10$                          ; Br if failure
       08 A2 51   B0    0F01  7$:    MOVW    R1,RCV_W_BLKSIZE(R2)            ; Insert block size
       0A A2 17   90    0F05         MOVB    S^#DYN$C_NET,RCV_B_BLKTYPE(R2)  ; Insert block type
     00BC C5 62   0E    0F09         INSQUE  (R2),UCB$Q_XD_RCV_BUF(R5)       ; Insert block on free receive list
          42 A5   A2    0F0E         SUBW    UCB$W_DEVBUFSIZ(R5),-           ; Decrement quota
        0136 C5         0F11                 UCB$W_XD_QUOTA(R5)              ;
             52   D4    0F14         CLRL    R2                              ; No more buffers given
             CB   11    0F16         BRB     5$                              ; Try for more
                        0F18  ;
                        0F18  ; Buffer allocation failure
                        0F18  ;
                        0F18  10$:   SETBIT  #XM$V_STS_BUFFAIL,-             ; Set buffer alloc failure
                        0F18                 UCB$L_DEVDEPEND(R5)             ;
             10   11    0F1D         BRB     30$                             ; And give any receives to device
                        0F1F
```

```
                        0F1F  3233 20$:   CLRBIT  #XM$V_STS_BUFFAIL,-          ; Clear buffer alloc failure
                        0F1F  3234                UCB$L_DEVDEPEND(R5)          ;
          50     52  D0  0F24  3235        MOVL    R2,R0                        ; Get address of buffer
                 06  13  0F27  3236        BEQL    30$                          ; Br if none
    00000000'GF      16  0F29  3237        JSB     G^COM$DRVDEALMEM             ; Deallocate it
                        0F2F  3238
                        0F2F  3239 30$:   DSBINT  UCB$B_DIPL(R5)               ; Sync access to device
                 07  10  0F36  3240        BSBB    START_RECEIVE                ; Start the receives
                        0F38  3241        ENBINT                               ; Restore previous IPL
                        0F3B  3242        POPQ    R3                           ; Restore R3, R4
                     05  0F3E  3243 40$:   RSB                                  ; Return
                        0F3F  3244        .DSABL  LSB
```

```
                              0F3F            .SBTTL  START_RECEIVE,  Start any receive requests pending
                              0F3F
                              0F3F    ;++
                              0F3F    ; START_RECEIVE - Start any receive requests
                              0F3F    ;
                              0F3F    ; Functional description:
                              0F3F    ;
                              0F3F    ; This routine attemptes to start any receives that may be pending.  This
                              0F3F    ; involves dequeueing a free receive buffer, mapping it, and loading it's
                              0F3F    ; address and size into the device.
                              0F3F    ;
                              0F3F    ; Inputs:
                              0F3F    ;
                              0F3F    ;       R5 = UCB address
                              0F3F    ;
                              0F3F    ;       IPL = DIPL
                              0F3F    ;
                              0F3F    ; Outputs:
                              0F3F    ;
                              0F3F    ;       R5 is preserved.
                              0F3F    ;
                              0F3F    ;       R0-R4 are destroyed
                              0F3F    ;
                              0F3F    ;--
                              0F3F
                              0F3F    START_RECEIVE:                                          ; Start receive operation
         51    0132 C5    9A  0F3F            MOVZBL  UCB$B_XD_RCV_MAX(R5),R1                 ; Get max concurrent receives
51   0130 C5    51    00  EB  0F44            FFC     #0,R1,UCB$B_XD_RCV_MAP(R5),R1           ; Get a free mapping slot
                      07  13  0F4B            BEQL    10$                                     ; Br if none - just exit
         53    00BC D5    0F  0F4D            REMQUE  @UCB$Q_XD_RCV_BUF(R5),R3                ; Get a free buffer
                      01  1C  0F52            BVC     20$                                     ; Br if buffer found
                      05      0F54    10$:    RSB                                             ; Return to caller
                              0F55    ;
                              0F55    ; Mark slot in use and create buffer address/character count image
                              0F55    ; in receive buffer and load UNIBUS adapter map registers.
                              0F55    ;
                              0F55    20$:    SETBIT  R1,UCB$B_XD_RCV_MAP(R5)                 ; Mark slot in use
            0B A3    51   90  0F5B            MOVB    R1,RCV_B_MAPSLOT(R3)                    ; Save mapping slot index
               42 A5      B0  0F5F            MOVW    UCB$W_DEVBUFSIZ(R5),-                   ; Insert character count
               0E A3          0F62                    RCV_L_BACC+2(R3)                        ;
                              0F64
                              0F64            CPUDISP <<790,70$>,-
                              0F64                     <780,70$>,-
                              0F64                     <750,70$>,-
                              0F64                     <730,70$>,-
                              0F64                     <UV1,30$>>
                              0F7E
                              0F7E    30$:    ; UV1
       50    00FC C541    D0  0F7E            MOVL    UCB$L_XD_RCV_PA(R5)[R1],R0              ; Get Physical address
            0C A3    50   B0  0F84            MOVW    R0,RCV_L_BACC(R3)                       ; Set BA00-BA15
       50    50    F0 8F  78  0F88            ASHL    #-16,R0,R0                              ; Shift down high byte
            10 A3    50   90  0F8D            MOVB    R0,RCV_B_BAHI(R3)                       ; Set BA16-BA21
                     33   11  0F91            BRB     90$                                     ; Join common code
                              0F93
                              0F93    70$:    ; All except UV1
       54    00C4 C541    DE  0F93            MOVAL   UCB$L_XD_RCV_MAP(R5)[R1],R4             ; Get mapping info slot address
        51    48 A3       9E  0F99            MOVAB   RCV_T_DATA(R3),R1                       ; Get receive buffer data addr
```

```
         0C A3    51    B0  0F9D        MOVW    R1,RCV_L_BACC(R3)                 ; Set BA0-BA8
  0C A3  07   09  64    F0  0FA1        INSV    (R4),#9,#7,RCV_L_BACC(R3)         ; Set BA9-BA15 from map reg
     50  64   06  07    EF  0FA7        EXTZV   #7,#6,(R4),R0                     ; Get BA16-BA17/BA21 also
  0C A3  02   1E  50    F0  0FAC        INSV    R0,#30,#2,RCV_L_BACC(R3)          ; Set BA16-BA17
                  53    DD  0FB2        PUSHL   R3                                ; Save buffer address
           52   02 A4   3C  0FB4        MOVZWL  2(R4),R2                          ; Set number of map registers
              53    64  3C  0FB8        MOVZWL  (R4),R3                           ; Set first map register number
                    54  D4  0FBB        CLRL    R4                                ; Use unbuffered datapath
          00000000'GF   16  0FBD        JSB     G^IOC$LOADUBAMAPN                 ; Load the map registers
                 53 8ED0    0FC3        POPL    R3                                ; Restore buffer address
                            0FC6
                            0FC6  90$:  ;
                            0FC6        ; Common code path
                            0FC6        ;
          0094 C5   63  0E  0FC6        INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)          ; Insert receive buffer at
                            0FCB                                                  ;   FRONT of input queue
                    03  10  0FCB        BSBB    LOAD_PORT                         ; Request port and give request
                 FF6F   31  0FCD        BRW     START_RECEIVE                     ; Let's try it again
```

```
                          0FD0          .SBTTL  LOAD_PORT,  Request CSR routine
                          0FD0
                          0FD0    ;++
                          0FD0    ; LOAD_PORT -  Request CSR port routine
                          0FD0    ;
                          0FD0    ; Functional description:
                          0FD0    ;
                          0FD0    ; Request the controller's input port to give it a request.  Since the
                          0FD0    ; controller doesn't service the input port when it is busy, we may have
                          0FD0    ; to request an interrupt to service the request.  The input request is
                          0FD0    ; assume to be on the input waiting queue.
                          0FD0    ;
                          0FD0    ;
                          0FD0    ; Inputs:
                          0FD0    ;
                          0FD0    ;       R5 = UCB address
                          0FD0    ;
                          0FD0    ;       IPL = DIPL
                          0FD0    ;
                          0FD0    ; Outputs:
                          0FD0    ;
                          0FD0    ;       R0 = Success if port loaded immediately
                          0FD0    ;       R1-R3,R9 = destroyed.
                          0FD0    ;       R4 = CSR address
                          0FD0    ;
                          0FD0    ;--
                          0FD0
                          0FD0    LOAD_PORT:                                         ; Give next request to device
       54     24 A5   D0  0FD0          MOVL    UCB$L_CRB(R5),R4                   ; Get CRB address
                          0FD4          ASSUME  IDB$L_CSR EQ 0
       54     2C B4   D0  0FD4          MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4       ; Get CSR address
       64     80 8F   93  0FD8          BITB    #XD_BSEL0_M_RQI,(R4)               ; Is an input request pending?
                 38   12  0FDC          BNEQ    10$                                ; Br if yes - wait for interrupt
       64     80 8F   88  0FDE          BISB    #XD_BSEL0_M_RQI,(R4)               ; Request input port
          53     05   3C  0FE2          MOVZWL  S^#5,R3                            ; Spin loop count
    02 A4     80 8F   93  0FE5    5$:   BITB    #XD_BSEL2_M_RDO,BSEL2(R4)          ; Is an output interrupt pend?
                 2A   12  0FEA          BNEQ    10$                                ; Br if yes - wait for interrupt
                          0FEC          TIMEWAIT #1,#XD_BSEL2_M_RDI,SEL2(R4),W     ; Wait for RDI to be set-10uSEC
          0C  50      E8  1010          BLBS    R0,LOAD_PORT_AVAIL                 ; Br if true - port is available
          CF  53      F5  1013          SOBGTR  R3,5$                              ; Loop if more time left
                          1016    ;
                          1016    ; Port is not currently available - request an interrupt to process the input.
                          1016    ;
          64     01   88  1016    10$:  BISB    #XD_BSEL0_M_IEI,(R4)               ; Request input interrupt
               13FC   30  1019          BSBW    SET_INTIM                          ; Start input wait timer
                 50   D4  101C          CLRL    R0                                 ; Indicate failure to get port
                      05  101E          RSB                                        ; Return to caller
                          101F
                          101F    ;
                          101F    ; Port is available - load request into CSRs
                          101F    ;
                          101F    LOAD_PORT_AVAIL:                                   ; Load CSR ports
       64     80 8F   8A  101F          BICB    #XD_BSEL0_M_RQI,(R4)               ; Clear request for port
       53   0094 D5   0F  1023          REMQUE  @UCB$Q_XD_INPUTQ(R5),R3            ; Get next request on input Q
                 09   1C  1028          BVC     10$                                ; Br if one there
    06 A4        00   B0  102A          MOVW    #XD_CI_V_NOOP,SEL6(R4)             ; Else, release input port
    02 A4        01   B0  102E          MOVW    #XD_I_V_CONTROL,SEL2(R4)           ; Set request type, clear RDI
```

```
                      05  1032         RSB                                       ; Return to caller
                          1033  ;
                          1033  ; Give request from input queue to device
                          1033  ;
    0A      0A A3     91  1033  10$:   CMPB     IRP$B_TYPE(R3),S^#DYN$C_IRP      ; Is buffer an IRP ?
                71    13  1037         BEQL     30$                              ; Br if yes - dispatch
    17      0A A3     91  1039         CMPB     IRP$B_TYPE(R3),S^#DYN$C_NET      ; Is this a receive msg?
                04    13  103D         BEQL     20$                              ; Br if yes
                          103F         BUG_CHECK NOBUFPCKT,FATAL                 ; Else, fatal error
                          1043  ;
                          1043  ; Load receive request
                          1043  ;
                          1043  20$:
                          1043         PUSHQ    R6                               ; Save registers
              14A5    30  1046         BSBW     XD_BUF_JNX                       ; Get the journal buffer
             14 50    E9  1049         BLBC     R0,25$                           ; Br if error
        86      08    90  104C         MOVB     #JNL_CSRIN,(R6)+                 ; Set journal type = CSRIN
                86    B4  104F         CLRW     (R6)+                            ; No CHAN
  86      1215 8F     B0  1051         MOVW     #<XD_FC_V_RECV@8>!21,(R6)+       ; Enter FUNC
                86    7C  1056         CLRQ     (R6)+                            ; STATION = 0
  86        68 A5     B0  1058         MOVW     UCB$W_DEVSTS(R5),(R6)+           ; Set DEVSTS
  86        44 A5     D0  105C         MOVL     UCB$L_DEVDEPEND(R5),(R6)+        ; Set DEVDEPEND
                          1060  25$:   POPQ     R6                               ; Restore registers
                          1063
  04 A4     0C A3     B0  1063         MOVW     RCV_L_BACC(R3),SEL4(R4)          ; Load buffer address
                          1068
                          1068         $DISPATCH UCB$B_DEVTYPE(R5),TYPE=B,-
                          1068         <-       ;function       action
                          1068                  <DT$_DMV11, 27$>,-               ; DMV-11
                          1068                  <DT$_DMP11, 28$>,-               ; DMP-11
                          1068         >
                          108B
  06 A4     10 A3     90  108B  27$:   MOVB     RCV_B_BAHI(R3),BSEL6(R4)         ;       ...
  08 A4     0E A3     B0  1090         MOVW     RCV_L_BACC+2(R3),SEL10(R4)       ;  and character count
                08    B0  1095         MOVW     #XD_I_V_RCV!XD_BSEL2_M_Q22,-     ; Set request type, clear RDI
             02 A4        1097                  SEL2(R4)                         ;
                09    11  1099         BRB      29$                              ; Continue
                          109B
  06 A4     0E A3     B0  109B  28$:   MOVW     RCV_L_BACC+2(R3),SEL6(R4)        ;  and character count
   02 A4        00    B0  10A0         MOVW     #XD_I_V_RCV,SEL2(R4)             ; Set request type, clear RDI
                          10A4
                          10A4                                                   ;  for trib #0
   00A8 D5      63    0E  10A4  29$:   INSQUE   (R3),@UCB$Q_XD_RCV_PND+4(R5)     ; Insert on end of pending Q
                      05  10A9         RSB                                       ; Return to caller
                          10AA  ;
                          10AA  ; Dispatch to input process handler
                          10AA  ;
                          10AA  30$:
                          10AA         PUSHQ    R6                               ; Save registers
        50      08    90  10AD         MOVB     #JNL_CSRIN,R0                    ; Set journal type = CSRIN
              141A    30  10B0         BSBW     XD_FILL_JNX                      ; Fill the journal buffer
             14 50    E9  10B3         BLBC     R0,50$                           ; Br if error
                66    7C  10B6         CLRQ     (R6)                             ; Clear status
        59  44 A3     D0  10B8         MOVL     IRP$L_CDB(R3),R9                 ; Get address of CDB address
                0C    13  10BC         BEQL     50$                              ; Br if none
        59      69    D0  10BE         MOVL     (R9),R9                          ; Get address of CDB
                07    13  10C1         BEQL     50$                              ; Br if none
```

```
     86   0C A9   B0  10C3          MOVW    CDB_W_STS(R9),(R6)+            ; Enter CDB status
          86  69  D0  10C7          MOVL    CDB_L_DEVDEPEND(R9),(R6)+      ; Enter CDB DEVDEPEND
                      10CA  50$:    POPQ    R6                             ; Restore registers
                      10CD
     51   21 A3   90  10CD          MOVB    IRP$B_XDFUNC(R3),R1            ; Get specific request
             04   10  10D1          BSBB    IN_DISP                        ; Dipatch to input service
     50      01   3C  10D3          MOVZWL  S^#SS$_NORMAL,R0               ; Indicate port success
                  05  10D6          RSB
                      10D7
                      10D7  IN_DISP:                                       ; Dispatch to input service
                      10D7          $DISPATCH       R1,TYPE=B,-
                      10D7          <-      ;function       action
                      10D7
                      10D7                  <XD_FC_V_INITT  IN_INITT>,-    ; Start tributary
                      10D7                  <XD_FC_V_INITC  IN_INITC>,-    ; Start DMP11 controller
                      10D7                  <XD_FC_V_STOPT  IN_STOPT>,-    ; Stop tributary
                      10D7                  <XD_FC_V_STOPC  IN_STOPC>,-    ; Stop DMP11 controller
                      10D7                  <XD_FC_V_CHGC   IN_CHGC>,-     ; Change controller parameters
                      10D7                  <XD_FC_V_CHGT   IN_CHGT>,-     ; Change tributary parameters
                      10D7                  <XD_FC_V_RDMODEM IN_RDMODEM>,- ; Read modem register
                      10D7                  <XD_FC_V_RDTERR IN_RDTERR>,-   ; Read tributary error counts
                      10D7                  <XD_FC_V_RDGERR IN_RDGERR>,-   ; Read global error counts
                      10D7                  <XD_FC_V_RCTERR IN_RCTERR>,-   ; Read and clear tributary ctrs
                      10D7                  <XD_FC_V_RCGERR IN_RCGERR>,-   ; Read and clear global counts
                      10D7                  <XD_FC_V_RDTSS  IN_RDTSS>,-    ; Read TSS location
                      10D7                  <XD_FC_V_RDGSS  IN_RDGSS>,-    ; Read GSS location
                      10D7                  <XD_FC_V_WTMODEM IN_WTMODEM>,- ; Write modem request
                      10D7                  <XD_FC_V_HALTT  IN_HALTT>,-    ; Halt tributary
                      10D7                  <XD_FC_V_KILLT  IN_KILLT>,-    ; Kill tributary
                      10D7                  <XD_FC_V_XMIT   IN_XMIT>,-     ; Xmit request
                      10D7                  <XD_FC_V_CANCEL IN_CANCEL>,-   ; Halt tributary
                      10D7                  <XD_FC_V_RECV   IN_RECV>,-     ; Receive request
                      10D7                  <XD_FC_V_NUMSYNC IN_NUMSYNC>,- ; Set number of sync characters
                      10D7                  <XD_FC_V_POLSTAT IN_POLSTAT>,- ; Latch polling state
                      10D7                  <XD_FC_V_POLSS  IN_RDTSS>,-    ; Read polling sub-state
                      10D7          >
                      1107  ;
                      1107  ; OTHERS: if not one of the above, then error
                      1107  ;
                      1107          BUG_CHECK NOBUFPCKT,FATAL              ; Fatal error
                      110B
                      110B
                      110B  ;+
                      110B  ; Establish tributary
                      110B  ;-
                      110B  IN_INITT:                                      ; Initialize tributary
        0E42      30  110B          BSBW    GET_CDB                        ; Get CDB address
        03 50     E8  110E          BLBS    R0,10$                         ; Br if CDB found
        039D      31  1111          BRW     REQ_ABORT                      ; Else, abort the request
                      1114
  17 0C A9   00   E2  1114  10$:    BBSS    #CD_TS_V_ESTAB,CDB_W_STS(R9),20$ ; Br if trib established
                      1119                                                 ; and indicate trib established
     0101 8F      B0  1119          MOVW    #XD_CI_V_ESTAB!XD_SEL6_M_ECOM,- ; Establish the trib
        06 A4         111D                  SEL6(R4)                       ;
  03 A4   40 A3   90  111F          MOVB    IRP$Q_STATION(R3),BSEL3(R4)    ; Give trib address
     02 A4   01   90  1124          MOVB    #XD_I_V_CONTROL,BSEL2(R4)      ; Give request, clear RDI
  0094 C5    63   0E  1128          INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)       ; Insert at front of input Q
```

```
          FEA0   31  112D           BRW     LOAD_PORT                       ; Request input again
                     1130
                     1130 20$:                                              ; Start tributary
    06 A4   03   B0  1130           MOVW    S^#XD_CI_V_ISTRT,SEL6(R4)       ; ISTRT the trib
    08 69   00   E1  1134           BBC     #XM$V_CHR_MOP,CDB_L_DEVDEPEND(R9),30$ ; Br if not MOP mode
    06 A4   04   B0  1138           MOVW    S^#XD_CI_V_MAINT,SEL6(R4)       ; Else, MOP mode
                     113C           SETBIT  #XM$V_STS_RUNNING,-             ; Indicate trib running
                     113C                   CDB_L_DEVDEPEND(R9)             ;
 06 A4   0100 8F  A8  1140 30$:     BISW    #XD_SEL6_M_ECOM,SEL6(R4)        ; Enable common buffers
 04 A4     4E A9  90  1146          MOVB    CDB_B_MRB(R9),BSEL4(R4)         ; Set buffer quota
 03 A4     40 A3  90  114B          MOVB    IRP$Q_STATION(R3),BSEL3(R4)     ; Set trib address
    02 A4   01   90  1150           MOVB    S^#XD_I_V_CONTROL,BSEL2(R4)     ; Give request, clear RDI
                     1154
                     1154           SETBIT  #XM$V_STS_ACTIVE,-              ; Indicate trib active
                     1154                   CDB_L_DEVDEPEND(R9)             ;
            20   A8  1158           BISW    #CD_TS_M_START,-                ; Trib has been started
         0C A9       115A                   CDB_W_STS(R9)                   ;
 38 A3     FF 8F  90  115C          MOVB    #255,IRP$L_MEDIA(R3)            ; Set 8 parameters
    21 A3   05   90  1161           MOVB    #XD_FC_V_CHGT,IRP$B_XDFUNC(R3)  ; Set new function request
  0094 C5   63   0E  1165           INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)        ; Insert at front of Q
          FE63   31  116A           BRW     LOAD_PORT                       ; Request port again
                     116D
                     116D ;+
                     116D ; Define mode
                     116D ;-
                     116D IN_INITC:                                         ; Give mode definition
 06 A4   02A4 C5  90  116D          MOVB    UCB$B_XD_MODE(R5),BSEL6(R4)     ; Set mode of operation
    02 A4   02   B0  1173           MOVW    #XD_I_V_MODE_DEF,SEL2(R4)       ; Give request, clear RDI
 38 A3     F0 8F  90  1177          MOVB    #<15@4>,IRP$L_MEDIA(R3)         ; Set 4 parameters
    21 A3   04   90  117C           MOVB    #XD_FC_V_CHGC,IRP$B_XDFUNC(R3)  ; Set new function request
  0094 C5   63   0E  1180           INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)        ; Insert at front of Q
          FE48   31  1185           BRW     LOAD_PORT                       ; Request port again
                     1188 ;+
                     1188 ; Stop tributary
                     1188 ;-
                     1188           .ENABL  LSB
                     1188 IN_STOPT:                                         ; Stop tributary
          0DC5   30  1188           BSBW    GET_CDB                         ; Get CDB address
         03 50   E8  118B           BLBS    R0,10$                          ; Br if CDB present
          0320   31  118E 5$:       BRW     REQ_ABORT                       ; Else, abort the request
                     1191
 20 0C A9   03   E4  1191 10$:      BBSC    #CD_TS_V_BUFRET,CDB_W_STS(R9),20$ ; Br if buffers back
            05   E5  1196           BBCC    #CD_TS_V_START,-                ; Br if already halted
      1B 0C A9       1198                   CDB_W_STS(R9),20$               ; ..trib halted itself
    06 A4   05   B0  119B 12$:      MOVW    S^#XD_CI_V_HALT,SEL6(R4)        ; Halt the trib
 03 A4     40 A3  90  119F          MOVB    IRP$Q_STATION(R3),BSEL3(R4)     ; Set trib address
    02 A4   01   90  11A4           MOVB    S^#XD_I_V_CONTROL,BSEL2(R4)     ; Set request, clear RDI
   00002800 8F   CA  11A8           BICL    #XM$M_STS_ACTIVE!XM$M_STS_RUNNING,- ; Indicate trib halted
            69       11AE                   CDB_L_DEVDEPEND(R9)             ;       ...
    34 B9   63   0E  11AF 15$:      INSQUE  (R3),@CDB_Q_INFOUT+4(R9)        ; Wait for buffer returns
          127A   31  11B3           BRW     SET_OUTTIM                      ; Start output wait timer
                     11B6                                                   ; ..and return
                     11B6
            02   E5  11B6 20$:      BBCC    #CD_TS_V_CANCEL,-               ; Br if no CANCEL is pending
      27 0C A9       11B8                   CDB_W_STS(R9),40$               ;
 52      30 A9   9E  11BB           MOVAB   CDB_Q_INFOUT(R9),R2             ; Else, get address of queue
    50      52   D0  11BF           MOVL    R2,R0                           ; Save queue listhead
```

```
           52    62     D0  11C2  30$:    MOVL    (R2),R2                             ; Get next IRP
           50    52     D1  11C5          CMPL    R2,R0                               ; End of queue?
                 E5     13  11C8          BEQL    15$                                 ; Br if yes - insert STOP
        21 A2    10     91  11CA          CMPB    #XD_FC_V_CANCEL,IRP$B_XDFUNC(R2)    ; CANCEL request?
                 F2     12  11CE          BNEQ    30$                                 ; Br if not, keep looking
           52    62     0F  11D0          REMQUE  (R2),R2                             ; Else, remove CANCEL request
      00B8 D5    62     0E  11D3          INSQUE  (R2),@UCB$Q_XD_POST+4(R5)           ; Post the CANCEL
        06 A4    00     B0  11D8          MOVW    #XD_CI_V_NOOP,SEL6(R4)              ; Release input port
        02 A4    01     B0  11DC          MOVW    #XD_I_V_CONTROL,SEL2(R4)            ; Set request type, clear RDI
                 CD     11  11E0          BRB     15$                                 ; Let the STOP assume the role
                            11E2
                 05     E4  11E2  40$:    BBSC    #CD_TS_V_START,-                    ; Br if trib started again!
           B4 0C A9         11E4                  CDB_W_STS(R9),12$                   ;
                            11E7  IN_KILLT:                                           ; Kill trib request
                 00     E1  11E7          BBC     #UCB$V_XD_INITED,-                  ; Br if device halted
              A2 68 A5      11E9                  UCB$W_DEVSTS(R5),5$                 ; ..and abort request
        06 A4    02     B0  11EC          MOVW    S^#XD_CI_V_KILL,SEL6(R4)            ; Set to kill trib
  03 A4    40 A3        90  11F0          MOVB    IRP$Q_STATION(R3),BSEL3(R4)         ; Set trib address
        02 A4    01     90  11F5          MOVB    S^#XD_I_V_CONTROL,BSEL2(R4)         ; Give request, clear RDI
               02C2     31  11F9          BRW     REQ_COM                             ; Request complete
                            11FC          .DSABL  LSB
                            11FC  ;+
                            11FC  ; Stop controller
                            11FC  ;-
                            11FC  IN_STOPC:
     13 68 A5    04     E2  11FC          BBSS    #UCB$V_XD_DTRCLR,UCB$W_DEVSTS(R5),10$ ; Br if DTR is clear
              04 A4     B4  1201          CLRW    SEL4(R4)                            ; Clear DTR
        06 A4    11     90  1204          MOVB    #XD_CI_V_WTMODEM,BSEL6(R4)          ; Set request type
        02 A4    01     B0  1208          MOVW    #XD_I_V_CONTROL,SEL2(R4)            ; Give request, clear RDI
      0094 C5    63     0E  120C          INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)            ; Insert at front of Q
               FDBC     31  1211          BRW     LOAD_PORT
                            1214
                            1214  10$:    CLRBIT  #UCB$V_XD_DTRCLR,UCB$W_DEVSTS(R5)   ; Clear DTR flag
     01 A4    40 8F     90  1219          MOVB    #XD_BSEL1_M_MCLR,BSEL1(R4)          ; Master clear device
                            121E
                            121E  REQ_COM1:                                           ; Long branch to REQ_COM
               029D     31  121E          BRW     REQ_COM                             ; Request complete
                            1221  ;+
                            1221  ; Change controller parameters
                            1221  ;-
                            1221  IN_CHGC:                                            ; Change controller parameters
                 00     E1  1221          BBC     #UCB$V_XD_INITED,-                  ; Br if device not inited
              5F 68 A5      1223                  UCB$W_DEVSTS(R5),DEV_INACT1         ; ..to device offline
  51    38 A3    08     00  EA  1226      FFS     #0,#8,IRP$L_MEDIA(R3),R1            ; Find next bit set
                            122C          CLRBIT  R1,IRP$L_MEDIA(R3)                  ; Indicate parameter set
  04 A4    02A4 C541    B0  1231          MOVW    UCB$W_XD_POLLPRM-<2*4>(R5)[R1],-    ; Set parameter value
                            1238                  SEL4(R4)                            ;   ...
     06 A4    51    18  A1  1238          ADDW3   #24,R1,SEL6(R4)                     ; Set GSS address
     06 A4    80 8F     88  123D          BISB    #XD_SEL6_M_WTSS,BSEL6(R4)           ; Request write of GSS
        02 A4    01     B0  1242          MOVW    S^#XD_I_V_CONTROL,SEL2(R4)          ; Set request, clear RDI
           38 A3        95  1246          TSTB    IRP$L_MEDIA(R3)                     ; All done?
                 08     13  1249          BEQL    20$                                 ; If EQL, yes
      0094 C5    63     0E  124B  10$:    INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)            ; Insert at front of Q
               FD7D     31  1250          BRW     LOAD_PORT                           ; Request port again
                            1253  ;
                            1253  ; Check if number of sync chars given
                            1253  ;
```

```
        02A7 C5   95  1253  20$:    TSTB    UCB$B_XD_NMS(R5)                    ; Is number of sync chars given
             C5   13  1257          BEQL    REQ_COM1                            ; Br if not - all done
       21 A3  13  90  1259          MOVB    #XD_FC_V_NUMSYNC,IRP$B_XDFUNC(R3)   ; Set new function request
             EC   11  125D          BRB     10$                                 ; Give new request to device
                      125F  ;+
                      125F  ; Set number of sync chars
                      125F  ;-
                      125F  IN_NUMSYNC:
             00   E1  125F          BBC     #UCB$V_XD_INITED,-                  ; Br if device not inited
        21 68 A5      1261                  UCB$W_DEVSTS(R5),DEV_INACT1         ; ..to device offline
   04 A4  02A7 C5 90  1264          MOVB    UCB$B_XD_NMS(R5),BSEL4(R4)          ; Set parameter value
   06 A4  009B 8F B0  126A          MOVW    #XD_SEL6_M_WTSS!27,SEL6(R4)         ; Set GSS address + WRITE TSS
      02 A4   01  B0  1270          MOVW    S^#XD_I_V_CONTROL,SEL2(R4)          ; Set request, clear RDI
           0247   31  1274          BRW     REQ_COM                             ; Request complete
                      1277  ;+
                      1277  ; Change tributary
                      1277  ;-
                      1277          .ENABL  LSB
                      1277  IN_CHGT:                                            ; Change controller parameters
           0CD6   30  1277          BSBW    GET_CDB                             ; Get CDB address
          03 50   E8  127A          BLBS    R0,10$                              ; Br if CDB present
           0231   31  127D          BRW     REQ_ABORT                           ; Else, abort the request
                      1280
             00   E0  1280  10$:    BBS     #CD_TS_V_ESTAB,-                    ; Br if trib inited
       03 0C A9       1282                  CDB_W_STS(R9),20$                   ;      ...
                      1285  DEV_INACT1:                                         ; Long branch to inactive
           021A   31  1285          BRW     DEV_INACT                           ; Br to device inactive
                      1288
51   38 A3  08  00 EA 1288  20$:    FFS     #0,#8,IRP$L_MEDIA(R3),R1            ; Find next bit set
                      128E          CLRBIT  R1,IRP$L_MEDIA(R3)                  ; Indicate parameter set
   04 A4  50 A941 B0  1293          MOVW    CDB_W_PO[LPRM(R9)[R1],SEL4(R4)      ; Set parameter value
   06 A4   51  18 A1  1299          ADDW3   #24,R1,SEL6(R4)                     ; Set TSS address
   06 A4   80  8F 88  129E          BISB    #XD_SEL6_M_WTSS,BSEL6(R4)           ; Request write of TSS
   03 A4   40  A3 90  12A3          MOVB    IRP$Q_STATION(R3),BSEL3(R4)         ; Set trib address
      02 A4   01  90  12A8          MOVB    S^#XD_I_V_CONTROL,BSEL2(R4)         ; Set request, clear RDI
         38 A3    95  12AC          TSTB    IRP$L_MEDIA(R3)                     ; All done?
             08   13  12AF          BEQL    40$                                 ; If EQL, yes
      0094 C5 63  0E  12B1  30$:    INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)            ; Insert at front of Q
           FD17   31  12B6          BRW     LOAD_PORT                           ; Request port again
                      12B9  ;
                      12B9  ; Set new polling state - if needed else unlatch poll state
                      12B9  ;
       21 A3  14  90  12B9  40$:    MOVB    #XD_FC_V_POLSTAT,IRP$B_XDFUNC(R3)   ; Set new request
             F2   11  12BD          BRB     30$                                 ; Ask for port again
                      12BF          .DSABL  LSB
                      12BF  ;+
                      12BF  ; Set new Poll State
                      12BF  ;-
                      12BF  IN_POLSTAT:
           0C8E   30  12BF          BSBW    GET_CDB                             ; Get CDB address
          03 50   E8  12C2          BLBS    R0,10$                              ; Br if CDB present
           01E9   31  12C5          BRW     REQ_ABORT                           ; Else, abort the request
                      12C8
             00   E1  12C8  10$:    BBC     #CD_TS_V_ESTAB,-                    ; Br if trib NOT inited
       B8 0C A9       12CA                  CDB_W_STS(R9),DEV_INACT1            ;      ...
                      12CD
                      12CD          $DISPATCH  CDB_B_POL(R9),TYPE=B,<-          ; Dispatch on polling state
```

```
                     12CD                <NMA$C_CIRPST_AUT  20$>,-        ; AUTOMATIC - unlatch
                     12CD                <NMA$C_CIRPST_ACT  30$>,-        ; ACTIVE - latch
                     12CD                <NMA$C_CIRPST_INA  40$>,-        ; INACTIVE - latch
                     12CD                <NMA$C_CIRPST_DIE  50$>,-        ; DYING - latch
                     12CD                <NMA$C_CIRPST_DED  60$>>         ; DEAD - latch
                     12DC
                     12DC                                                 ; All others, then
        04 A4    B4  12DC 20$:   CLRW    SEL4(R4)                         ; Clear SEL4
06 A4   1000 8F  B0  12DF        MOVW    #XD_SEL6_M_UPOL,SEL6(R4)         ; Unlatch polling state
           1B    11  12E5        BRB     80$                              ; Continue
                     12E7
        04 A4    94  12E7 30$:   CLRB    BSEL4(R4)                        ; Latch at ACTIVE
           10    11  12EA        BRB     70$                              ;
     04 A4  01   90  12EC 40$:   MOVB    S^#1,BSEL4(R4)                   ; Latch at INACTIVE
           0A    11  12F0        BRB     70$                              ;
     04 A4  02   90  12F2 50$:   MOVB    S^#2,BSEL4(R4)                   ; Latch at DYING
           04    11  12F6        BRB     70$                              ;
     04 A4  03   90  12F8 60$:   MOVB    S^#3,BSEL4(R4)                   ; Latch at DEAD
06 A4   2000 8F  B0  12FC 70$:   MOVW    #XD_SEL6_M_LPOL,SEL6(R4)         ; Latch the poll state
                     1302
03 A4   40 A3    90  1302 80$:   MOVB    IRP$Q_STATION(R3),BSEL3(R4)      ; Set trib address
02 A4      01    90  1307        MOVB    S^#XD_I_V_CONTROL,BSEL2(R4)      ; Set request, clear RDI
         01B0    31  130B        BRW     REQ_COM                          ; Request complete
                     130E ;+
                     130E ; Read modem request
                     130E ;-
                     130E IN_RDMODEM:                                     ; Read modem register
           00    E0  130E        BBS     #UCB$V_XD_INITED,-               ; Br if device inited
     03 68 A5        1310                UCB$W_DEVSTS(R5),10$             ;
         018C    31  1313        BRW     DEV_INACT                        ; Else, device inactive
                     1316
06 A4      10    B0  1316 10$:   MOVW    S^#XD_CI_V_RDMODEM,SEL6(R4)      ; Set to read modem register
02 A4      01    B0  131A        MOVW    S^#XD_I_V_CONTROL,SEL2(R4)       ; Set request, clear RDI
00B0 D5    63    0E  131E        INSQUE  (R3),@UCB$Q_XD_INFOUT+4(R5)      ; Insert request at end of Q
         110A    31  1323        BRW     SET_OUTTIM                       ; Set output wait timer
                     1326                                                 ; ..and return
                     1326 ;+
                     1326 ; Transmit request
                     1326 ;-
                     1326        .ENABL  LSB
                     1326 IN_XMIT:                                        ; Transmit request
         0C27    30  1326        BSBW    GET_CDB                          ; Get CDB address
        03 50    E8  1329        BLBS    R0,20$                           ; Br if okay
         0182    31  132C 10$:   BRW     REQ_ABORT                        ; And abort the request
                     132F
           0B    E1  132F 20$:   BBC     #XM$V_STS_ACTIVE,-               ; Br if trib not active
        F9 69        1331                CDB_L_DEVDEPEND(R9),10$          ;
04 A4   38 A3    B0  1333        MOVW    IRP$L_MEDIA(R3),SEL4(R4)         ; Load buffer address
03 A4   40 A3    90  1338        MOVB    IRP$Q_STATION(R3),BSEL3(R4)      ; Set trib address
                     133D
                     133D        $DISPATCH UCB$B_DEVTYPE(R5),TYPE=B,-
                     133D        <-      ;function        action
                     133D                <DT$_DMV11, 40$>,-               ; DMV-11
                     133D                <DT$_DMP11, 60$>,-               ; DMP-11
                     133D        >
                     1360
06 A4   3D A3    90  1360 40$:   MOVB    IRP$L_MEDIA+5(R3),BSEL6(R4)      ;       ...
```

```
08 A4   3A A3  B0  1365       MOVW    IRP$L_MEDIA+2(R3),SEL10(R4)           ;   and character count
          0C   90  136A       MOVB    S^#XD_I_V_XMIT!XD_BSEL2_M_Q22,-       ; Set request, clear RDI
     02 A4         136C               BSEL2(R4)
          09   11  136E       BRB     90$                                   ; Continue in common code
                   1370
06 A4   3A A3  B0  1370 60$:  MOVW    IRP$L_MEDIA+2(R3),SEL6(R4)            ;   and character count
   02 A4   04  90  1375       MOVB    S^#XD_I_V_XMIT,BSEL2(R4)              ; Set request, clear RDI
                   1379
   2C B9   63  0E  1379 90$:  INSQUE  (R3),@CDB_Q_XMT_PND+4(R9)             ; Insert at end of pending Q
               05  137D       RSB                                           ;
                   137E ;+
                   137E ; Receive request
                   137E ;-
                   137E IN_RECV:                                            ; Bump receive buffer quota
        0BCF   30  137E       BSBW    GET_CDB                               ; Get CDB address
       A8 50   E9  1381       BLBC    R0,10$                                ; Br if error
          0B   E1  1384       BBC     #XM$V_STS_ACTIVE,-                    ; Br if trib not active
       A4 69       1386               CDB_L_DEVDEPEND(R9),10$               ;
   04 A4   01  90  1388       MOVB    S^#1,BSEL4(R4)                        ; Else, bump quota by 1 buffer
06 A4 0100 8F  B0  138C       MOVW    #XD_SEL6_M_ECOM,SEL6(R4)              ; Set request type
03 A4   40 A3  90  1392       MOVB    IRP$Q_STATION(R3),BSEL3(R4)           ; Set trib address
   02 A4   01  90  1397       MOVB    S^#XD_I_V_CONTROL,BSEL2(R4)           ; Set request, clear RDI
        0120   31  139B       BRW     REQ_COM
                   139E       .DSABL  LSB
                   139E ;+
                   139E ; Read selected tss address
                   139E ;-
                   139E IN_RDTSS:                                           ; Read tss address
        0BAF   30  139E       BSBW    GET_CDB                               ; Get CDB address
       03 50   E8  13A1       BLBS    R0,10$                                ; Br if got it
        010A   31  13A4       BRW     REQ_ABORT                             ; Else, abort the request
                   13A7
06 A4   3E A3  90  13A7 10$:  MOVB    IRP$L_MEDIA+6(R3),BSEL6(R4)           ; Set tss address
   06 A4   20  88  13AC       BISB    #XD_SEL6_M_RTSS,BSEL6(R4)             ; Read the tss address
          2A   11  13B0       BRB     IN_READT                              ; Join common code
                   13B2 ;+
                   13B2 ; Read tributary counters
                   13B2 ;-
                   13B2       .ENABL  LSB
                   13B2 IN_RDTERR:                                          ; Read tributary counters
      7E   20  B0  13B2       MOVW    S^#XD_SEL6_M_RTSS,-(SP)               ; Set to read TSS
           05  11  13B5       BRB     10$                                   ; Continue
                   13B7
                   13B7 ;+
                   13B7 ; Read and clear tributary counters
                   13B7 ;-
                   13B7 IN_RCTERR:                                          ; Read and clear trib counters
  7E   0040 8F  B0 13B7       MOVW    #XD_SEL6_M_RCTS,-(SP)                 ; Set to read and clear TSS
                   13BC
        0B91   30  13BC 10$:  BSBW    GET_CDB                               ; Get CDB address
       03 50   E8  13BF       BLBS    R0,15$                                ; Br if got one
        00EC   31  13C2       BRW     REQ_ABORT                             ; Else, abort the request
                   13C5
          00   E0  13C5 15$:  BBS     #CD_TS_V_ESTAB,-                      ; Br if trib established
    03 0C A9       13C7               CDB_W_STS(R9),20$                     ;
        00D3   31  13CA       BRW     DEV_INACT_SP                          ; Else, device inactive
                   13CD
```

```
 51   38 A3    08   00   EA 13CD  20$:    FFS     #0,#8,IRP$L_MEDIA(R3),R1           ; Get next counter index
      06 A4    51   08   A1 13D3          ADDW3   #8,R1,SEL6(R4)                     ; Calculate TSS address
           06 A4    8E   A8 13D8          BISW    (SP)+,SEL6(R4)                     ; Set read mode
                            13DC          .DSABL  LSB
                            13DC  IN_READT:                                          ; Finish read tss request
      03 A4    40 A3     90 13DC          MOVB    IRP$Q_STATION(R3),BSEL3(R4)        ; Set tributary address
         02 A4    01     90 13E1          MOVB    S^#XD_I_V_CONTROL,BSEL2(R4)        ; Set request, clear RDI
         34 B9    63     0E 13E5          INSQUE  (R3),@CDB_Q_INFOUT+4(R9)           ; Insert request at end of Q
               1044      31 13E9          BRW     SET_OUTTIM                         ; Start output wait timer
                            13EC
                            13EC  ;+
                            13EC  ; Write modem request
                            13EC  ;-
                            13EC  IN_WTMODEM:                                        ; Write modem register
         06 A4    11     90 13EC          MOVB    S^#XD_CI_V_WTMODEM,BSEL6(R4)       ; Write modem request
      04 A4    38 A3     B0 13F0          MOVW    IRP$L_MEDIA(R3),SEL4(R4)           ; Set new modem values
         02 A4    01     B0 13F5          MOVW    S^#XD_I_V_CONTROL,SEL2(R4)         ; Set request, clear RDI
               00C2      31 13F9          BRW     REQ_COM                            ; Request complete
                            13FC                                                     ; ..and return
                            13FC  ;+
                            13FC  ; Halt trib
                            13FC  ;
                            13FC  ;       On a SYS$CANCEL request we will halt the tributary and then re-start
                            13FC  ;       the tributary. This will get back all I/O from the hardware, but still
                            13FC  ;       allow the user to continue using the tributary.
                            13FC  ;-
                            13FC          .ENABL  LSB
                            13FC  IN_CANCEL:                                         ; Halt trib - CANCEL request
               0B51      30 13FC          BSBW    GET_CDB                            ; Get CDB address
              03 50      E8 13FF          BLBS    R0,20$                             ; Br if CDB found
               00AC      31 1402  10$:    BRW     REQ_ABORT                          ; Else, abort the request
                            1405
      16 0C A9    03     E4 1405  20$:    BBSC    #CD_TS_V_BUFRET,CDB_W_STS(R9),30$ ; Br if buffers returned
                  05     E1 140A          BBC     #CD_TS_V_START,-                   ; Br if already halted
           F3 0C A9         140C                  CDB_W_STS(R9),10$                  ; ..must be second request
                  47     10 140F          BSBB    100$                               ; Else, perform halt on trib
                  20     AA 1411          BICW    #CD_TS_M_START,-                   ; Indicate trib has
              0C A9         1413                  CDB_W_STS(R9)                      ;  been halted
         0C A9    04     A8 1415          BISW    #CD_TS_M_CANCEL,CDB_W_STS(R9)      ; Indicate that we are waiting
                            1419                                                     ;  for $CANCEL to complete
         34 B9    63     0E 1419          INSQUE  (R3),@CDB_Q_INFOUT+4(R9)           ; Wait for buffer returns
               1010      31 141D          BRW     SET_OUTTIM                         ; Start output wait timer
                            1420
                  0C     AA 1420  30$:    BICW    #CD_TS_M_CANCEL!CD_TS_M_BUFRET,-; Indicate that $CANCEL
              0C A9         1422                  CDB_W_STS(R9)                      ;  completed
         06 A4    03     B0 1424          MOVW    S^#XD_CI_V_ISTRT,SEL6(R4)          ; ISTRT the trib
      08 69       00     E1 1428          BBC     #XM$V_CHR_MOP,CDB_L_DEVDEPEND(R9),40$ ; Br if not MOP mode
         06 A4    04     B0 142C          MOVW    S^#XD_CI_V_MAINT,SEL6(R4)          ; Else, MOP mode
                            1430          SETBIT  #XM$V_STS_RUNNING,-                ; Indicate trib running
                            1430                  CDB_L_DEVDEPEND(R9)                ;
   06 A4    0100 8F      A8 1434  40$:    BISW    #XD_SEL6_M_ECOM,SEL6(R4)           ; Enable common buffers
      04 A4    4E A9     90 143A          MOVB    CDB_B_MRB(R9),BSEL4(R4)            ; Set buffer quota
      03 A4    40 A3     90 143F          MOVB    IRP$Q_STATION(R3),BSEL3(R4)        ; Set trib address
         02 A4    01     90 1444          MOVB    S^#XD_I_V_CONTROL,BSEL2(R4)        ; Give request, clear RDI
                            1448
                            1448          SETBIT  #XM$V_STS_ACTIVE,-                 ; Indicate trib active
                            1448                  CDB_L_DEVDEPEND(R9)                ;
```

```
              20 A8 144C            BISW    #CD_TS_M_START,-              ; set Internal flag also
           0C A9    144E                    CDB_W_STS(R9)                 ;
            006B 31 1450    60$:    BRW     REQ_COM                       ; Request complete
                    1453                                                  ; ..and return
                    1453    IN_HALTT:                                     ; Halt trib - no status checks
              03 10 1453            BSBB    100$                          ; Perform halt on trib
            0066 31 1455            BRW     REQ_COM                       ; Request complete
                    1458
              00 E1 1458    100$:   BBC     #UCB$V_XD_INITED,-            ; Br if device halted
        54 68 A5    145A                    UCB$W_DEVSTS(R5),REQ_ABORT    ; ..and abort the request
     06 A4    05 90 145D            MOVB    S^#XD_CI_V_HALT,BSEL6(R4)     ; Halt trib request
  03 A4    40 A3 90 1461            MOVB    IRP$Q_STATION(R3),BSEL3(R4)   ; Set trib address
     02 A4    01 90 1466            MOVB    S^#XD_I_V_CONTROL,SEL2(R4)    ; Set request, clear RDI
                 05 146A            RSB                                   ; Return to caller
                    146B            .DSABL  LSB
                    146B    ;+
                    146B    ; Read selected gss address
                    146B    ;-
                    146B    IN_RDGSS:                                     ; Read gss address
  06 A4    3E A3 90 146B            MOVB    IRP$L_MEDIA+6(R3),BSEL6(R4)   ; Set gss address
     06 A4    20 88 1470            BISB    S^#XD_SEL6_M_RTSS,BSEL6(R4)   ; Request read of gss address
              1E 11 1474            BRB     IN_READG                      ; Finish read gss
                    1476    ;+
                    1476    ; Read global counters
                    1476    ;-
                    1476            .ENABL  LSB
                    1476    IN_RDGERR:                                    ; Read global counters
           7E 20 B0 1476            MOVW    S^#XD_SEL6_M_RTSS,-(SP)       ; Set to read GSS
              05 11 1479            BRB     10$                           ; Continue
                    147B
                    147B    ;+
                    147B    ; Read and clear global counters
                    147B    ;-
                    147B    IN_RCGERR:                                    ; Read global counters
      7E  0040 8F B0 147B           MOVW    #XD_SEL6_M_RCTS,-(SP)         ; Set to read and clear GSS
                    1480
              00 E1 1480    10$:    BBC     #UCB$V_XD_INITED,-            ; Br if device not inited
        1B 68 A5    1482                    UCB$W_DEVSTS(R5),DEV_INACT_SP ; ..to device inactive
51   38 A3 08 00 EA 1485            FFS     #0,#8,IRP$L_MEDIA(R3),R1      ; Get next counter index
  06 A4    51 08 A1 148B            ADDW3   #8,R1,SEL6(R4)                ; Calculate GSS address
     06 A4    8E A8 1490            BISW    (SP)+,SEL6(R4)                ; Set read mode
                    1494            .DSABL  LSB
                    1494    IN_READG:                                     ; Finish read gss request
     02 A4    01 B0 1494            MOVW    S^#XD_I_V_CONTROL,SEL2(R4)    ; Set request, clear RDI
                    1498                                                  ;  for trib #0
   00B0 D5    63 0E 1498            INSQUE  (R3),@UCB$Q_XD_INFOUT+4(R5)   ; Insert request at end of Q
            0F90 31 149D            BRW     SET_OUTTIM                    ; Start output wait timer
                    14A0                                                  ; ..and return
                    14A0    ;+
                    14A0    ; Device inactive
                    14A0    ;-
                    14A0    DEV_INACT_SP:                                 ; Device inactive and pop stack
              8E B5 14A0            TSTW    (SP)+                         ; Pop one word from stack
                    14A2    DEV_INACT:                                    ; Device inactive
                    14A2            ; Release the input CSR's
     06 A4    00 B0 14A2            MOVW    S^#XD_CI_V_NOOP,SEL6(R4)      ; Set noop request
     02 A4    01 B0 14A6            MOVW    S^#XD_I_V_CONTROL,SEL2(R4)    ; Set request type, clear RDI
```

```
50    20D4 8F    B0  14AA  3896          MOVW    #SS$_DEVINACT,R0                 ; Set error return code
             10  11  14AF  3897          BRB     REQ_COM_ERR                      ; Complete request
                     14B1  3898  ;+
                     14B1  3899  ; Complete request
                     14B1  3900  ;-
                     14B1  3901  REQ_ABORT:                                       ; Abort request
                     14B1  3902          ; Release the input CSR's
    06 A4    00  B0  14B1  3903          MOVW    S^#XD_CI_V_NOOP,SEL6(R4)         ; Set noop request
    02 A4    01  B0  14B5  3904          MOVW    S^#XD_I_V_CONTROL,SEL2(R4)       ; Set request type, clear RDI
       50    2C  B0  14B9  3905          MOVW    #SS$_ABORT,R0                    ; Set status return
             03  11  14BC  3906          BRB     REQ_COM_ERR                      ; Complete request in error
                     14BE  3907  REQ_COM:                                         ; Request complete
       50    01  B0  14BE  3908          MOVW    S^#SS$_NORMAL,R0                 ; Return success
                     14C1  3909  REQ_COM_ERR:
    42 A3    50  B0  14C1  3910          MOVW    R0,IRP$W_CSTATUS(R3)             ; Set completion status
  00B8 D5    63  0E  14C5  3911          INSQUE  (R3),@UCB$Q_XD_POST+4(R5)        ; Insert IRP on completion Q
       03D4      31  14CA  3912          BRW     SCHED_FORKC                      ; Schedule a fork process
```

```
                        14CD          .SBTTL  RDI_INTRPT,  CSR Ready Input Interrupt Service Routine
                        14CD
                        14CD    ;++
                        14CD    ; RDI_INTRPT  -  CSR Ready Input Interrupt Service Routine
                        14CD    ;
                        14CD    ; Functional description:
                        14CD    ;
                        14CD    ; This routine is called when an interrupt occurs to give an input waiting
                        14CD    ; request to the controller.  Prior to execution of this routine, a request
                        14CD    ; was made to LOAD_PORT, but the port was unavailable at that time.
                        14CD    ;
                        14CD    ;
                        14CD    ; Inputs:
                        14CD    ;
                        14CD    ;         00(SP) = address of IDB address
                        14CD    ;         04(SP) - 1C(SP) = R0 - R5
                        14CD    ;
                        14CD    ;         IPL = DIPL
                        14CD    ;
                        14CD    ; Outputs:
                        14CD    ;
                        14CD    ;         The request is loaded and if there are more requests pending,
                        14CD    ;         they are given until there are no more.  Finally, the interrupt
                        14CD    ;         is dismissed.
                        14CD    ;
                        14CD    ;--
                        14CD
                        14CD    RDI_INTRPT:
         54   9E   D0   14CD            MOVL    @(SP)+,R4                       ; Get IDB address
              59   DD   14D0            PUSHL   R9                              ; Save R9
     55   18  A4   D0   14D2            MOVL    IDB$L_UCBLST(R4),R5             ; Get UCB address
         54   64   D0   14D6            MOVL    (R4),R4                         ; Get CSR address
     64   81  8F   8A   14D9            BICB    #XD_BSEL0_M_IEI!XD_BSEL0_M_RQI,(R4) ; Clear input request flags
              00   E1   14DD            BBC     #UCB$V_XD_INITED,-              ; Exit if controller not inited
         32 68 A5       14DF                    UCB$W_DEVSTS(R5),INTERR
       02A5  C5    94   14E2            CLRB    UCB$B_XD_INTIM(R5)              ; Reset input wait timer
                        14E6    ;
                        14E6    ; Find CDB address
                        14E6    ;
     53   0094 D5  0F   14E6            REMQUE  @UCB$Q_XD_INPUTQ(R5),R3         ; Get next request on input Q
              27   1D   14EB            BVS     INTERR                          ; Br if none - error
   0094 C5    63   0E   14ED            INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)        ; Re-insert packet on Q
            FB2A   30   14F2            BSBW    LOAD_PORT_AVAIL                 ; Load CSRs
                        14F5
     53   0094 D5  0F   14F5    10$:    REMQUE  @UCB$Q_XD_INPUTQ(R5),R3         ; Get next request on input Q
              0B   1D   14FA            BVS     INTEXIT                         ; Br if none - exit
   0094 C5    63   0E   14FC            INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)        ; Re-insert packet on Q
            FACC   30   1501            BSBW    LOAD_PORT                       ; Load CSRs
           EE 50   E8   1504            BLBS    R0,10$                          ; Br if port was available
                        1507    ;
                        1507    ; Exit interrupt
                        1507    ;
                        1507    INTEXIT:                                        ; Exit interrupt
          59 8ED0       1507            POPL    R9                              ; Restore R9
         50   8E   7D   150A            MOVQ    (SP)+,R0                        ; Restore registers
         52   8E   7D   150D            MOVQ    (SP)+,R2                        ;
         54   8E   7D   1510            MOVQ    (SP)+,R4                        ;
```

```
            02  1513            REI                                     ; Return from interrupt.
                1514    ;
                1514    ; An input interrupt occurred with nothing on input queue. Set a noop
                1514    ; request to device and hope a mode definition has been given previously.
                1514    ;
                1514    INTERR:
06 A4   00  90  1514            MOVB    S^#XD_CI_V_NOOP,BSEL6(R4)       ; Set noop request
02 A4   01  B0  1518            MOVW    S^#XD_I_V_CONTROL,SEL2(R4)      ; Set request type, clear RDI
        E9  11  151C            BRB     INTEXIT                         ; Exit the interrupt
```

```
                       151E         .SBTTL  RDO_INTRPT,  CSR Ready Output Interrupt Service Routine
                       151E
                       151E   ;++
                       151E   ; RDO_INTRPT - CSR Ready Output Interrupt Service Routine
                       151E   ;
                       151E   ; Functional description:
                       151E   ;
                       151E   ; This routine is called when the DMP11 device generates an output
                       151E   ; interrupt after setting the RDO bit in the BSEL2.  This means that
                       151E   ; the device has some type of information for the driver, solicited
                       151E   ; or otherwise.  This may be buffers being returned from transmits
                       151E   ; or receives, error counters or polling parameters being read, or
                       151E   ; error conditions.
                       151E   ;
                       151E   ;
                       151E   ; Inputs:
                       151E   ;
                       151E   ;       00(SP) = address of IDB address
                       151E   ;       04(SP) - 1C(SP) = R0 - R5
                       151E   ;
                       151E   ;       IPL = DIPL
                       151E   ;
                       151E   ; Outputs:
                       151E   ;
                       151E   ;       Interrupt is dismissed.
                       151E   ;
                       151E   ;--
                       151E
                       151E   RDO_INTRPT:
        54   9E   D0   151E           MOVL    @(SP)+,R4                       ; Get IDB address
   55   18   A4   D0   1521           MOVL    IDB$L_UCBLST(R4),R5             ; Get UCB address
        52   64   D0   1525           MOVL    (R4),R2                         ; Get CSR address
             59   DD   1528           PUSHL   R9                              ; Save R9
             00   E0   152A           BBS     #UCB$V_XD_INITED,-              ; Br if inited
     07 68   A5        152C                   UCB$W_DEVSTS(R5),5$             ;
02 A2   80   8F   8A   152F   3$:     BICB    #XD_BSEL2_M_RDO,BSEL2(R2)       ; Else, release CSRs
             D1   11   1534           BRB     INTEXIT                         ; And exit interrupt
                       1536
   53   02   A2   B0   1536   5$:     MOVW    SEL2(R2),R3                     ; Get SEL2 in R3
        51   53   D0   153A           MOVL    R3,R1                           ; Save type code
   53   53   10   78   153D           ASHL    #16,R3,R3                       ; Shift to high word of R3
        53   62   B0   1541           MOVW    SEL0(R2),R3                     ; Get SEL0 in low
                       1544                                                   ;  word of R3
   54   06   A2   B0   1544           MOVW    SEL6(R2),R4                     ; Get SEL6 in high
54 54        10   78   1548           ASHL    #16,R4,R4                       ;  word of R4
   54   04   A2   B0   154C           MOVW    SEL4(R2),R4                     ; Get SEL4 in low
                       1550                                                   ;  word of R4
                       1550           PUSHQ   R6                              ; Save registers
           0F98   30   1553           BSBW    XD_BUF_JNX                      ; Get the journal buffer
        09   50   E9   1556           BLBC    R0,7$                           ; Br if error
        86   09   90   1559           MOVB    #JNL_CSROUT,(R6)+               ; Set journal type = CSROUT
        86   53   D0   155C           MOVL    R3,(R6)+                        ; Store the CSRs
        86   54   D0   155F           MOVL    R4,(R6)+                        ;       ...
                       1562   7$:     POPQ    R6                              ; Restore registers
                       1565
   50   03   A2   90   1565           MOVB    BSEL3(R2),R0                    ; Get trib address
             59   D4   1569           CLRL    R9                              ; Assume no CDB
```

```
             50  95  156B        TSTB    R0                                    ; Trib #0?
             06  13  156D        BEQL    10$                                   ; Br if yes - no CDB
           0952  30  156F        BSBW    FIND_TRIB                             ; Find CDB from trib address
          BA 50  E9  1572        BLBC    R0,3$                                 ; Br if trib not found - exit
 51  FFFFFFF8 8F  CA  1575  10$: BICL    #^C<XD_BSEL2_M_TYPE>,R1               ; Clear all but type code
                     157C        CASE    R1,TYPE=B,<OUT_RCV,OUT_CTL,OUT_INFO,OUT_RCV_ERR,- ;
                     157C                OUT_XMIT,3$,OUT_XMIT_ERR,OUT_XMIT_ERR> ; Case on the type code
                     1590  ;+
                     1590  ; Process receives completed by the DMP11. After the receive buffer is
                     1590  ; posted for completion, any receive buffers that are need are given to
                     1590  ; the device.
                     1590  ;-
                     1590        .ENABL  LSB
                     1590  OUT_RCV:                                            ; Process receive complete
          51  52  D0  1590       MOVL    R2,R1                                 ; Save CSR address
     52  00A4 D5  0F  1593       REMQUE  @UCB$Q_XD_RCV_PND(R5),R2              ; Get oldest pending receive
             12  1D  1598        BVS     15$                                   ; Error if none.
  54  C0000000 8F  CA  159A      BICL    #^XC0000000,R4                        ; Clear BA16 and BA17 from BA/CC
      0C A2   54  B1  15A1       CMPW    R4,RCV_L_BACC(R2)                     ; Buffer address match?
             08  13  15A5        BEQL    20$                                   ; Br if yes - ok
   00A4 C5   62  0E  15A7  10$:  INSQUE  (R2),UCB$Q_XD_RCV_PND(R5)             ; Requeue the receive buffer
           FF58  31  15AC  15$:  BRW     INTEXIT                               ; Exit the interrupt
      50   0B A2  9A  15AF  20$: MOVZBL  RCV_B_MAPSLOT(R2),R0                  ; Get mapping slot number
  EE 0130 C5  50  E5  15B3       BBCC    R0,UCB$B_XD_RCV_MAP(R5),10$           ; Mark as free - error if not
                     15B9
                     15B9        $DISPATCH UCB$B_DEVTYPE(R5),TYPE=B,-
                     15B9        <-      ;function       action
                     15B9                <DT$_DMV11, 23$>,-                    ; DMV-11
                     15B9                <DT$_DMP11, 26$>,-                    ; DMP-11
                     15B9        >
                     15DC
   0E A2   08 A1  B0  15DC  23$: MOVW    SEL10(R1),RCV_L_BACC+2(R2)            ; Save byte count for recv
             04  11  15E1        BRB     30$                                   ; Continue in common code
                     15E3
      0C A2   54  D0  15E3  26$: MOVL    R4,RCV_L_BACC(R2)                     ; Save byte count
                     15E7
   02 A1   80 8F  8A  15E7  30$: BICB    #XD_BSEL2_M_RDO,SEL2(R1)              ; Release CSRs
      51   0E A2  B0  15EC       MOVW    RCV_L_BACC+2(R2),R1                   ; Get byte count
                     15F0        CPUDISP <<790,37$>,-
                     15F0                 <780,37$>,-
                     15F0                 <750,37$>,-
                     15F0                 <730,37$>,-
                     15F0                 <UV1,33$>>
                     160A
   50    0114 C540  D0  160A  33$: MOVL  UCB$L_XD_RCV_VA(R5)[R0],R0            ; Get buffer VA
             3C  BB  1610        PUSHR   #^M<R2,R3,R4,R5>                      ; Save registers
   48 A2   60  51  28  1612      MOVC3   R1,(R0),RCV_T_DATA(R2)                ; Copy the data
             3C  BA  1617        POPR    #^M<R2,R3,R4,R5>                      ; Restore registers
  53   53   08  18  EF  1619  37$: EXTZV #24,#8,R3,R3                          ; Get trib address
             06  13  161E        BEQL    45$                                   ; Br if trib zero????
      0C A2   53  90  1620       MOVB    R3,RCV_L_BACC(R2)                     ; Return trib address
             59  11  1624        BRB     80$                                   ; Complete the rcv in fork
                     1626  ;
                     1626  ; Bogus trib address
                     1626  ;
   00BC C5   62  0E  1626  45$:  INSQUE  (R2),UCB$Q_XD_RCV_BUF(R5)             ; Requeue buffer to device
           FED9  31  162B        BRW     INTEXIT                               ; Exit the interrupt
```

```
                             162E  ;+
                             162E  ; Process a transmit buffer complete.
                             162E  ;-
                             162E  OUT_XMIT:                                                       ; Process xmit complete
     02 A2    80 8F     8A   162E              BICB    #XD_BSEL2_M_RDO,SEL2(R2)                    ; Release CSRs
                  59    D5   1633              TSTL    R9                                          ; CDB address given?
                  17    13   1635              BEQL    50$                                         ; Br if not - exit interrupt
        52    28 B9     0F   1637              REMQUE  @CDB_Q_XMT_PND(R9),R2                       ; Get next xmit for trib
                  11    1D   163B              BVS     50$                                         ; Error if none
  54     C0000000 8F    CA   163D              BICL    #^XC0000000,R4                              ; Clear BA16 and BA17 from BA/CC
     38 A2        54    B1   1644              CMPW    R4,IRP$L_MEDIA(R2)                          ; Buffer address match?
                  07    13   1648              BEQL    60$                                         ; Br if yes - ok
     28 A9        62    0E   164A              INSQUE  (R2),CDB_Q_XMT_PND(R9)                      ; Else, requeue the request
                FEB6    31   164E  50$:        BRW     INTEXIT                                     ; Exit the interrupt
                             1651
     38 A2        54    D0   1651  60$:        MOVL    R4,IRP$L_IOST1(R2)                          ; Save byte count
     42 A2        01    B0   1655              MOVW    S^#SS$_NORMAL,IRP$W_CSTATUS(R2)             ; Set success
                  24    11   1659              BRB     80$                                         ; Queue the xmit for completion
                             165B
                             165B  OUT_XMIT_ERR:                                                   ; Xmit not sent
     02 A2    80 8F     8A   165B              BICB    #XD_BSEL2_M_RDO,SEL2(R2)                    ; Release CSRs
                  59    D5   1660              TSTL    R9                                          ; CDB address present
                  EA    13   1662              BEQL    50$                                         ; Br if not - exit
        52    28 B9     0F   1664              REMQUE  @CDB_Q_XMT_PND(R9),R2                       ; Get next xmit pending
                  E4    1D   1668              BVS     50$                                         ; Br if none
                  0F    11   166A              BRB     70$                                         ;
                             166C
                             166C  OUT_RCV_ERR:                                                    ; Receive buffer unused
     02 A2    80 8F     8A   166C              BICB    #XD_BSEL2_M_RDO,SEL2(R2)                    ; Release CSRs
     52     00A4 D5     0F   1671              REMQUE  @UCB$Q_XD_RCV_PND(R5),R2                    ; Get oldest pending receive
                  D6    1D   1676              BVS     50$                                         ; Br if none
           0C A2        94   1678              CLRB    RCV_L_BACC(R2)                              ; Set receive error return
     42 A2        2C    B0   167B  70$:        MOVW    #SS$_ABORT,IRP$W_CSTATUS(R2)                ; Set error return
                             167F
     00B8 D5      62    0E   167F  80$:        INSQUE  (R2),@UCB$Q_XD_POST+4(R5)                   ; Queue request for posting
                  03    12   1684              BNEQ    90$                                         ; Br if not first entry
                0218    30   1686              BSBW    SCHED_FORKC                                 ; Schedule fork process
                F8B3    30   1689  90$:        BSBW    START_RECEIVE                               ; Start any receives
                FE78    31   168C              BRW     INTEXIT                                     ; Exit interrupt
                             168F              .DSABL  LSB
                             168F
                             168F  ;+
                             168F  ; This routine is called when the device interrupts with a INFORMATION OUT
                             168F  ; command being returned to the driver.  This command results from an
                             168F  ; associated CONTROL IN command, requesting some information from the device.
                             168F  ; Since the information may be of some global type, such as global error
                             168F  ; counters, or the modem register contents; the DMP11 device may not return
                             168F  ; a tributary address on OUTPUT. In this case all global requests are
                             168F  ; assumed to be processed in a FIFO order and the global requests are
                             168F  ; queued to the UCB INFO OUT QUEUE (and not the CDB INFO OUT QUEUE).
                             168F  ;-
                             168F  OUT_INFO:                                                       ; Process information out
     02 A2    80 8F     8A   168F              BICB    #XD_BSEL2_M_RDO,SEL2(R2)                    ; Release CSRs
  52    54    08    10  EF   1694              EXTZV   #16,#8,R4,R2                                ; Get BSEL6
        52    60 8F     93   1699              BITB    #<XD_SEL6_M_RTSS!XD_SEL6_M_RCTS>,R2 ; Was this a read request?
                  05    12   169D              BNEQ    5$                                          ; Br if yes
        52        08    91   169F              CMPB    #8,R2                                       ; Was it a read modem request?
```

```
            06  12 16A2          BNEQ    10$                             ; Br if not
          0123  30 16A4  5$:     BSBW    OUT_STATUS                      ; Else, process status
          FE5D  31 16A7          BRW     INTEXIT                         ; Exit interrupt
                   16AA
      52    10  91 16AA  10$:    CMPB    #16,R2                          ; Is it a BUFFER RETURN COMPLETE?
            32  12 16AD          BNEQ    30$                             ; Br if not - exit interrupt
            59  D5 16AF          TSTL    R9                              ; CDB address found?
            2E  13 16B1          BEQL    30$                             ; Br if no - exit interrupt
                   16B3          SETBIT  #CD_TS_V_BUFRET,CDB_W_STS(R9)   ; Note buffers returned
                   16B8  ;
                   16B8  ; We will scan the information out queue for the stop request
                   16B8  ;
      53 30 A9  9E 16B8          MOVAB   CDB_Q_INFOUT(R9),R3             ; Get address
         50 53  D0 16BC          MOVL    R3,R0                           ; Save queue listhead address
         53 63  D0 16BF  20$:    MOVL    (R3),R3                         ; Get next IRP
         50 53  D1 16C2          CMPL    R3,R0                           ; End of queue?
            1A  13 16C5          BEQL    30$                             ; Br if yes - exit interrupt
   21 A3    10  91 16C7          CMPB    #XD_FC_V_CANCEL,IRP$B_XDFUNC(R3) ; Is this a CANCEL request?
            06  13 16CB          BEQL    25$                             ; Br if yes - complete it
   21 A3    02  91 16CD          CMPB    #XD_FC_V_STOPT,IRP$B_XDFUNC(R3) ; Is this the stop request?
            EC  12 16D1          BNEQ    20$                             ; Br if no - keep looking
      53    63  0F 16D3  25$:    REMQUE  IRP$L_IOQFL(R3),R3              ; Else, get the IRP
          0D57  30 16D6          BSBW    SET_OUTTIM                      ; Start output wait timer
  0094 C5   63  0E 16D9          INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)        ; Insert stop at front of Q
          F8EF  30 16DE          BSBW    LOAD_PORT                       ; Give request to device
          FE23  31 16E1  30$:    BRW     INTEXIT                         ; Else, exit interrupt
                   16E4
                   16E4  ;+
                   16E4  ; This routine is called when the DMP11 device interrupts with a CONTROL
                   16E4  ; OUT command.  Some type of error has occurred and the driver must notify
                   16E4  ; the user and also must take any appropriate steps to handle the error.
                   16E4  ;-
                   16E4  OUT_CTL:                                        ; Process control out
  02 A2  80 8F  8A 16E4          BICB    #XD_BSEL2_M_RDO,SEL2(R2)        ; Release CSRs
51  54   08 10  EF 16E9          EXTZV   #16,#8,R4,R1                    ; Get BSEL6
    51   C0 8F  91 16EE          CMPB    #^O<300>,R1                     ; Is this buffer too small?
            1C  13 16F2          BEQL    PROCEDURE_ERR                   ; Br if yes - procedure error
    51   C4 8F  91 16F4          CMPB    #^O<304>,R1                     ; Is this a disconnect?
            05  12 16F8          BNEQ    10$                             ; Br if not
                   16FA          SETBIT  #XM$V_STS_DISC,UCB$L_DEVDEPEND(R5) ; Set error flag
52  51   02 06  EF 16FF  10$:    EXTZV   #6,#2,R1,R2                     ; Else, get severity code
                   1704          CASE    R2,TYPE=B,<-                    ; Process by severity
                   1704                  SOFT_ERR,-                      ;   Soft error
                   1704                  PROCEDURE_ERR,-                 ;   Procedure error
                   1704                  PROCEDURE_ERR,-                 ;   Lots of procedure errors
                   1704                  FATAL_ERR>                      ;   Fatal error
                   1710  ;+
                   1710  ; PROCEDURE_ERR - Process procedure error from device
                   1710  ;-
                   1710  PROCEDURE_ERR:                                  ; Process procedure error
            59  D5 1710          TSTL    R9                              ; CDB address found?
            66  13 1712          BEQL    NOOP                            ; Br if no - exit now
                   1714          SETBIT  #XM$V_ERR_TRIB,CDB_L_DEVDEPEND(R9) ; Indicate trib error
                   1718          ;BRB    FORCE_HALT                      ; Force trib to shutdown
                   1718
                   1718  FORCE_HALT:                                     ; Force trib to shutdown
   0C A9    20  AA 1718          BICW    #CD_TS_M_START,CDB_W_STS(R9)    ; Clear started flag
```

```
                                 171C  FORCE_HALT1:
      53   54   08   10   EF     171C           EXTZV   #16,#8,R4,R3                        ; Get BSEL6 value
            03 A9   53   90      1721           MOVB    R3,CDB_L_DEVDEPEND+3(R9)            ; Save BSEL6 in CDB
          00002800 8F    CA      1725           BICL    #XM$M_STS_RUNNING!XM$M_STS_ACTIVE,- ; Tributary is halted
                    69           172B                   CDB_L_DEVDEPEND(R9)                 ;
      53      029C C5    DE      172C           MOVAL   UCB$L_XD_HALT(R5),R3                ; Set address of halted trib list
                                 1731  SET_INDEX:                                           ; Set trib index in list
            52   0E A9   9A      1731           MOVZBL  CDB_B_TRB_ADDR(R9),R2               ; Get trib address from CDB
 0138 C5   EA17 CF  52   3A      1735           LOCC    R2,MAX_W_TRIB,UCB$B_XD_TRIB_VEC(R5)      ; Find trib address
                    3B   13      173D           BEQL    NOOP                                ; Exit if not found ???
      52      0138 C5    9E      173F           MOVAB   UCB$B_XD_TRIB_VEC(R5),R2            ; Else, get address of vector
            51      52   C2      1744           SUBL2   R2,R1                               ; Calculate index
                                 1747           SETBIT  R1,(R3)                             ; Set trib index
                                 174B  SFORK_EXIT:                                          ; Exit with a fork process
                  0153   30      174B           BSBW    SCHED_FORKC                         ; Schedule a fork process
                    2A   11      174E           BRB     NOOP                                ; Exit the interrupt
                                 1750
                                 1750  ;+
                                 1750  ; SOFT_ERR - Process soft error from device
                                 1750  ;-
                                 1750  SOFT_ERR:                                            ; Process soft error
                    59   D5      1750           TSTL    R9                                  ; CDB address found?
                    26   13      1752           BEQL    NOOP                                ; Br if no - exit now
      52   51   06   00   EF     1754           EXTZV   #0,#6,R1,R2                         ; Get error type
        52   52   FF 8F   78     1759           ASHL    #-1,R2,R2                           ; Convert type to case code
                                 175E           CASE    R2,TYPE=B,LIMIT=#1,<-               ; Process by type
                                 175E                   THRESH,-                            ;   Receive threshold error
                                 175E                   THRESH,-                            ;   Transmit threshold error
                                 175E                   THRESH,-                            ;   Select threshold error
                                 175E                   START_ERR,-                         ;   Start when running
                                 175E                   MAINT_ERR,-                         ;   MOP when running
                                 175E                   NOOP,-                              ;   MOP when halted
                                 175E                   START_ERR1,-                        ;   Start when in MOP
                                 175E                   NOOP,-                              ;   Reserved
                                 175E                   DEAD_ERR,-                          ;   Dead tributary
                                 175E                   RUNNING_NOTIF,-                     ;   Running tributary
                                 175E                   PROCEDURE_ERR,-                     ;   Babbling trib
                                 175E                   PROCEDURE_ERR>                      ;   Streaming trib
                                 177A  ;
                                 177A  ; All others
                                 177A  ;
                                 177A  NOOP:                                                ; No special operation
                  FD8A   31      177A           BRW     INTEXIT                             ; Just exit the interrupt
                                 177D
                                 177D  THRESH:                                              ; Threshold errors
                                 177D           SETBIT  #XM$V_ERR_THRESH,CDB_L_DEVDEPEND(R9) ; Indicate warning
                    F7   11      1781           BRB     NOOP                                ; And exit interrupt
                                 1783
                                 1783  START_ERR:                                           ; Start received in RUN
                                 1783           SETBIT  #XM$V_ERR_START,CDB_L_DEVDEPEND(R9) ; Indicate cause of error
                    8F   11      1787           BRB     FORCE_HALT                          ; Force trib to shutdown
                                 1789
                                 1789  START_ERR1:                                          ; Start received in MOP
                                 1789           SETBIT  #XM$V_ERR_START,CDB_L_DEVDEPEND(R9) ; Indicate start received
                    EB   11      178D           BRB     NOOP                                ; Exit the interrupt
                                 178F
                                 178F  MAINT_ERR:                                           ; MOP received while running
```

```
                       178F  4268           SETBIT  #XM$V_ERR_MAINT,CDB_L_DEVDEPEND(R9) ; Indicate cause of error
        0C A9   20  A8 1793  4269           BISW    #CD_TS_M_START,CDB_Q_STS(R9)      ; Trib starts up by itself in MOP
                83  11 1797  4270           BRB     FORCE_HALT1                       ; Force trib shutdown
                       1799  4271
                       1799  4272 DEAD_ERR:                                           ; Dead tributary
                       1799  4273           CLRBIT  #XM$V_STS_RUNNING,CDB_L_DEVDEPEND(R9) ; No longer running
                DB  11 179D  4274           BRB     NOOP                              ; Exit the interrupt
                       179F  4275
                       179F  4276 RUNNING_NOTIF:                                      ; Running tributary
                       179F  4277           SETBIT  #XM$V_STS_RUNNING,-               ; Indicate trib is running
                       179F  4278                   CDB_L_DEVDEPEND(R9)               ;
     53    0298 C5  DE 17A3  4279           MOVAL   UCB$L_XD_RUN(R5),R3               ; Set address of run list
              FF86  31 17A8  4280           BRW     SET_INDEX                         ; Set trib index in run list
                       17AB  4281
                       17AB  4282 ;+
                       17AB  4283 ; FATAL_ERR - Process a fatal device error
                       17AB  4284 ;-
                       17AB  4285 FATAL_ERR:                                          ; Process a fatal device error
     52      24 A5  D0 17AB  4286           MOVL    UCB$L_CRB(R5),R2                  ; Get CRB address
                       17AF  4287           ASSUME  IDB$L_CSR EQ 0
     52      2C B2  D0 17AF  4288           MOVL    @CRB$L_INTD+VEC$L_IDB(R2),R2      ; Get CSR address
  01 A2      40 8F  90 17B3  4289           MOVB    #XD_BSEL1_M_MCLR,BSEL1(R2)        ; Master clear the device
53   54      08 10  EF 17B8  4290           EXTZV   #16,#8,R4,R3                      ; Get BSEL6 value
     47 A5      53  90 17BD  4291           MOVB    R3,UCB$L_DEVDEPEND+3(R5)          ; Save error code
     53      01 15  78 17C1  4292           ASHL    #XD_BSEL2_V_ERR+16,#1,R3          ; Note fatal error
              00DB  30 17C5  4293           BSBW    SCHED_FORK                        ; Schedule a fork process
                B0  11 17C8  4294           BRB     NOOP                              ; And exit
```

```
                          17CA            .SBTTL  OUT_STATUS,  Process Information Out from device
                          17CA
                          17CA      ;++
                          17CA      ; OUT_STATUS - Process Information Out from device
                          17CA      ;
                          17CA      ; Functional description:
                          17CA      ;
                          17CA      ; This routine get the next request from the information out waiting queue
                          17CA      ; and returns the information from the CSRs.
                          17CA      ;
                          17CA      ; Inputs:
                          17CA      ;
                          17CA      ;         R3 = SEL0 and SEL2
                          17CA      ;         R4 = SEL4 and SEL6
                          17CA      ;         R5 = UCB address
                          17CA      ;         R9 = CDB address (only if tributary information)
                          17CA      ;
                          17CA      ;         IPL = DIPL
                          17CA      ;
                          17CA      ; Outputs:
                          17CA      ;
                          17CA      ;         R0-R3 = Destroyed
                          17CA      ;--
                          17CA
                          17CA      OUT_STATUS:                                              ; Process information out
  51   53   08   18   EF  17CA              EXTZV   #24,#8,R3,R1                             ; Get trib address
                 55   13  17CF              BEQL    40$                                      ; Br if global request
                          17D1      ;
                          17D1      ; Read from TSS
                          17D1      ;
       52   30 A9   9E    17D1              MOVAB   CDB_Q_INFOUT(R9),R2                      ; Get address
            50   52   D0  17D5              MOVL    R2,R0                                    ; Save queue listhead address
            52   62   D0  17D8      5$:     MOVL    (R2),R2                                  ; Get next IRP
            50   52   D1  17DB              CMPL    R2,R0                                    ; End of queue?
                 3E   13  17DE              BEQL    20$                                      ; Br if yes - exit
                          17E0      ;
                          17E0      ; Only a read TSS, read counter, CANCEL and a stop can be on the INFOUT queue.
                          17E0      ; Find the first non-stop request.
                          17E0      ;
       21 A2    02   91   17E0              CMPB    #XD_FC_V_STOPT,IRP$B_XDFUNC(R2)          ; Is this a stop request?
                F2   13   17E4              BEQL    5$                                       ; Br if yes - keep looking
       21 A2    10   91   17E6              CMPB    #XD_FC_V_CANCEL,IRP$B_XDFUNC(R2)         ; Is this a CANCEL request?
                EC   13   17EA              BEQL    5$                                       ; Br if yes - keep looking
          52    62   0F   17EC              REMQUE  IRP$L_IOQFL(R2),R2                       ; Else get read request
          0C3E       30   17EF              BSBW    SET_OUTTIM                               ; Start output wait timer
       21 A2    15   91   17F2              CMPB    #XD_FC_V_POLSS,IRP$B_XDFUNC(R2)          ; Is this the polling substate
                0B   12   17F6              BNEQ    7$                                       ; Br if not
  54   54    F8 8F   78   17F8              ASHL    #-8,R4,R4                                ; Shift down high byte
       3F A2    54   90   17FD              MOVB    R4,IRP$B_POLS(R2)                        ; Save return info
                0D   11   1801              BRB     15$                                      ; Continue
  51      2C A2      D0   1803      7$:     MOVL    IRP$L_SVAPTE(R2),R1                      ; Get system buffer address
          38 A2      D5   1807              TSTL    IRP$L_MEDIA(R2)                          ; Is this a counter request?
                13   12   180A              BNEQ    30$                                      ; Br if yes
       0C A1    54   3C   180C      10$:    MOVZWL  R4,P2B_T_DATA(R1)                        ; Else, store return info.
       42 A2    01   B0   1810      15$:    MOVW    #SS$_NORMAL,IRP$W_CSTATUS(R2)            ; Set success return
     00B8 D5    62   0E   1814              INSQUE  (R2),@UCB$Q_XD_POST+4(R5)                ; Queue request for posting
                03   12   1819              BNEQ    20$                                      ; Br if not first entry
```

```
               0083  30  181B         BSBW    SCHED_FORKC                   ; Schedule fork process
                     05  181E  20$:   RSB                                   ; Return to caller
                         181F  ;
                         181F  ; Trib counters request
                         181F  ;
       50   E93A CF  9E  181F  30$:   MOVAB   TRIB_COUNTER1-2,R0            ; Get address of counter types
                 18  11  1824         BRB     50$                           ; Complete the request
                         1826  ;
                         1826  ; Global request
                         1826  ;
                         1826  ; Only a read GSS or read global counters can be on INFOUT queue.
                         1826  ;
       52   00AC D5  0F  1826  40$:   REMQUE  @UCB$Q_XD_INFOUT(R5),R2       ; Get next global request
                 F1  1D  182B         BVS     20$                           ; Br if none waiting
               0C00  30  182D         BSBW    SET_OUTTIM                    ; Start output wait timer
         51   2C A2  D0  1830         MOVL    IRP$L_SVAPTE(R2),R1           ; Get system buffer address
              38 A2  D5  1834         TSTL    IRP$L_MEDIA(R2)               ; Is this a counters request?
                 D3  13  1837         BEQL    10$                           ; Br if no - read of GSS
                         1839  ;
                         1839  ; Global counters request
                         1839  ;
       50   E90C CF  9E  1839         MOVAB   LINE_COUNTER-<2*5>,R0         ; Get address of counter types
                         183E
                         183E  ;
                         183E  ; Handle DMP error counter types.
                         183E  ;
53   38 A2   08   00  EA  183E  50$:   FFS     #0,#8,IRP$L_MEDIA(R2),R3      ; Get counter index
                         1844         CLRBIT  R3,IRP$L_MEDIA(R2)            ; Indicate counter complete
            51   61  D0  1849         MOVL    (R1),R1                       ; Get next data pointer
                 52  DD  184C         PUSHL   R2                            ; Save IRP address
          81   8043  B0  184E         MOVW    (R0)+[R3],(R1)+               ; Set next counter type
52   FE A1   03   0C  EF  1852         EXTZV   #NMA$V_CNT_MAP,#3,-2(R1),R2   ; Get width + bit map
                         1858         CASE    R2,TYPE=B,LIMIT=#2,<-         ; Handle only DMP counter types
                         1858                 60$,-                         ;   8 bit counter (2 per read)
                         1858                 70$,-                         ;   8 bit counter, bit map
                         1858                 80$>                          ;   16 bit counter
                         1862
            81   54  90  1862  60$:   MOVB    R4,(R1)+                      ; Store 1st 8 bit counter
          81   6043  B0  1865         MOVW    (R0)[R3],(R1)+                ; Set next counter type
     54   54   F8 8F  78  1869         ASHL    #-8,R4,R4                     ; Shift down next counter
            81   54  90  186E         MOVB    R4,(R1)+                      ; Store 2nd 8 bit counter
                 10  11  1871         BRB     90$                           ; Check if all done
                         1873
  52   54    08   08  EF  1873  70$:   EXTZV   #8,#8,R4,R2                   ; Get bit map
            81   52  9B  1878         MOVZBW  R2,(R1)+                      ; Store bit map
            81   54  90  187B         MOVB    R4,(R1)+                      ; Store counter value
                 03  11  187E         BRB     90$                           ; Check if all done
                         1880
            81   54  B0  1880  80$:   MOVW    R4,(R1)+                      ; Store 16 bit counter
                         1883
               52 8ED0   1883  90$:   POPL    R2                            ; Restore IRP address
         53   2C A2  D0  1886         MOVL    IRP$L_SVAPTE(R2),R3           ; Get system PTE address
            63   51  D0  188A         MOVL    R1,(R3)                       ; Reset next data pointer
              38 A2  95  188D         TSTB    IRP$L_MEDIA(R2)               ; All done?
                 07  12  1890         BNEQ    100$                          ; Br if not
         63   0C A3  9E  1892         MOVAB   P2B_T_DATA(R3),(R3)           ; Else, reset start of data
               FF77  31  1896         BRW     15$                           ; And complete request
```

```
                 1899 4410
0094 C5  62  0E  1899 4411 100$:   INSQUE  (R2),UCB$Q_XD_INPUTQ(R5)        ; Insert IRP on input Q
        F72F 31  189E 4412         BRW     LOAD_PORT                       ; Request input
```

```
                          18A1          .SBTTL  SCHED_FORK,  Schedule the fork process
                          18A1          .SBTTL  SCHED_FORKC, Schedule the fork process with R3 clear
                          18A1
                          18A1  ;++
                          18A1  ; SCHED_FORK - Schedule the fork process
                          18A1  ; SCHED_FORKC - Schedule the fork process with R3 clear
                          18A1  ;
                          18A1  ; Functional description:
                          18A1  ;
                          18A1  ; This routine is called to schedule the error and I/O completion fork process.
                          18A1  ; The last controller port and CSR values are saved for examination.  If the
                          18A1  ; fork process is already pending, only the last CSR values are saved if there
                          18A1  ; was an error.
                          18A1  ;
                          18A1  ; Inputs:
                          18A1  ;
                          18A1  ;         R3 = Last SEL0 and SEL2 values
                          18A1  ;         R4 = Last SEL4 and SEL6 values
                          18A1  ;         R5 = UCB address
                          18A1  ;
                          18A1  ;         IPL = DIPL or higher
                          18A1  ;
                          18A1  ; Outputs:
                          18A1  ;
                          18A1  ;         R3 is cleared if SCHED_FORKC entry.
                          18A1  ;         R5 = UCB address
                          18A1  ;
                          18A1  ; If XD_BSEL2_V_ERR is set in BSEL2 the following is returned:
                          18A1  ;
                          18A1  ;         UCB$L_XD_SEL0(R5) = SEL0 and SEL2 values
                          18A1  ;         UCB$L_XD_SEL4(R5) = SEL4 and SEL6 values
                          18A1  ;
                          18A1  ;--
                          18A1
                          18A1  SCHED_FORKC:                                                    ; Schedule fork process, clr R3
                53   D4   18A1          CLRL    R3                                              ; No device error
                          18A3  SCHED_FORK:                                                     ; Schedule fork process
       18 68 A5 03   E2   18A3          BBSS    #UCB$V_XD_FORK_PEND,UCB$W_DEVSTS(R5),10$ ; Br if fork pending
                55   DD   18A8          PUSHL   R5                                              ; Save R5
                04   10   18AA          BSBB    5$                                              ; Setup fork process
                55 8ED0   18AC          POPL    R5                                              ; Restore R5
                     05   18AF          RSB                                                     ; Return to caller
                          18B0
 55  000002B6 8F     C0   18B0  5$:     ADDL    #UCB$B_XD_FKB,R5                                ; Point to fork block in UCB
           CC'AF     9F   18B7          PUSHAB  B^FORK_PROC                                     ; Set address of fork process
     00000000'GF     17   18BA          JMP     G^EXE$FORK                                      ; Schedule the fork and return
                          18C0
      05 53 15       E1   18C0  10$:    BBC     #XD_BSEL2_V_ERR+16,R3,20$                       ; Br if not error
     02C6 C5 53      7D   18C4          MOVQ    R3,UCB$L_XD_SEL0(R5)                            ; Save last CSR values
                     05   18C9  20$:    RSB                                                     ; Return to caller
```

```
                        18CA          .SBTTL  FORK_PROC, Error and completion fork process handling
                        18CA
                        18CA  ;++
                        18CA  ; FORK_PROC - Error and completion fork processing
                        18CA  ;
                        18CA  ; Functional description:
                        18CA  ;
                        18CA  ; This routine is called as a fork process to handle errors and all completions
                        18CA  ; pending.
                        18CA  ;
                        18CA  ; Inputs:
                        18CA  ;
                        18CA  ;       R3 = Last SEL0 and SEL2 values
                        18CA  ;       R4 = Last SEL4 and SEL6 values
                        18CA  ;       R5 = UCB address
                        18CA  ;
                        18CA  ;       IPL = FIPL
                        18CA  ;
                        18CA  ; Outputs:
                        18CA  ;
                        18CA  ;       R5 is preserved.
                        18CA  ;
                        18CA  ;--
                 0BBA'  18CA          .WORD   TIMEOUT-.                       ; Offset to timeout routine
                        18CC          .ENABL  LSB
                        18CC  FORK_PROC:                                      ; Error/completion fork process
 55   000002B6 8F   C2  18CC          SUBL    #UCB$B_XD_FKB,R5                ; Point to UCB
                        18D3          CLRBIT  #UCB$V_XD_FORK_PEND,UCB$W_DEVSTS(R5) ; Clear fork process flag
    0C 53   15      E1  18D8          BBC     #XD_BSEL2_V_ERR+16,R3,5$        ; Br if not an error
                        18DC          SETBIT  #XM$V_ERR_FATAL,UCB$L_DEVDEPEND(R5) ; Indicate fatal error
       0082 C5      B6  18E1          INCW    UCB$W_ERRCNT(R5)                ; Bump error counter
          0454      31  18E5          BRW     SHUTDOWN_DEV                    ; Shutdown the device
                        18E8  ;
                        18E8  ; Complete any requests on completion queue
                        18E8  ;
            59      DD  18E8  5$:     PUSHL   R9                              ; Save R9
 52     00B4 D5     0F  18EA  10$:    REMQUE  @UCB$Q_XD_POST(R5),R2           ; Get next completed block
            0A      1C  18EF          BVC     20$                             ; Br if got one
          0326      30  18F1          BSBW    HALT_TRIBS                      ; Halt all tribs with errors
          0342      30  18F4          BSBW    START_XMITS                     ; Start xmits for tribs in run
         59 8ED0        18F7          POPL    R9                              ; Restore R9
                    05  18FA          RSB                                     ; Return to caller
                        18FB  ;
                        18FB  ; Found a completed block - finish it
                        18FB  ;
 17       0A A2     91  18FB  20$:    CMPB    IRP$B_TYPE(R2),S^#DYN$C_NET     ; Was it a receive?
            0A      13  18FF          BEQL    30$                             ; Br if yes - complete it
 0A       0A A2     91  1901          CMPB    IRP$B_TYPE(R2),S^#DYN$C_IRP     ; Was it an IRP?
            4F      13  1905          BEQL    50$                             ; Br if yes - complete it
                        1907          BUG_CHECK NOBUFPCKT,FATAL               ; Else, fatal error
                        190B  ;
                        190B  ; Receive complete - if there is a pending receive I/O request, complete it.
                        190B  ; Otherwise, queue the buffer and, if enabled, deliver attention AST.
                        190B  ;
 50       0C A2     9A  190B  30$:    MOVZBL  RCV_L_BACC(R2),R0               ; Get trib address
            11      12  190F          BNEQ    32$                             ; Br if no error
         42 A5      A0  1911          ADDW    UCB$W_DEVBUFSIZ(R5),-           ; Adjust receive buffer quota
```

```
     0136 C5       1914                UCB$W_XD_QUOTA(R5)                     ;
     50   52  D0   1917         MOVL   R2,R0                                  ; Get address of buffer
00000000'GF   16   191A         JSB    G^COM$DRVDEALMEM                       ; Deallocate buffer
          C8  11   1920         BRB    10$                                    ; Get next completion
                   1922
        059F  30   1922  32$:   BSBW   FIND_TRIB                              ; Get CDB address
       04 50  E8   1925         BLBS   R0,33$                                 ; Br if got CDB
                   1928         BUG_CHECK NOBUFPCKT,FATAL                     ; Else, fatal error
                   192C
51    0E A2   3C   192C  33$:   MOVZWL RCV_L_BACC+2(R2),R1                    ; Get the byte count
38 A9    51   C0   1930         ADDL   R1,CDB_L_BRC(R9)                       ; Update byte count
         04   1E   1934         BCC    35$                                    ; Br if no overflow
38 A9    01   CE   1936         MNEGL  #1,CDB_L_BRC(R9)                       ; Else, latch it
                   193A  35$:   INCC   CDB_L_DMR(R9)                          ; Update message count
53    18 B9   0F   1942         REMQUE @CDB_Q_RCV_REQ(R9),R3                  ; Remove waiting IRP
         05   1D   1946         BVS    40$                                    ; Br if none - queue for later
       0248   30   1948         BSBW   FINISH_RCV_IO                          ; Else, finish the I/O
         9D   11   194B         BRB    10$                                    ; Look for next completion
                   194D
24 B9    62   0E   194D  40$:   INSQUE (R2),@CDB_Q_RCV_MSG+4(R9)              ; Queue receive msg for later
       091C   30   1951         BSBW   POKE_USER                              ; Deliver ASTs
         94   11   1954         BRB    10$                                    ; Look for more completions
                   1956  ;
                   1956  ; IRP packet completion - branch to proper completion handling routine.
                   1956  ;
   53    52   D0   1956  50$:   MOVL   R2,R3                                  ; Copy IRP address
52    21 A3   9A   1959         MOVZBL IRP$B_XDFUNC(R3),R2                    ; Get function type
                   195D         $DISPATCH        R2,TYPE=B,-                  ; Dispatch on request type
                   195D         <-
                   195D                 <XD_FC_V_INITT  FORK_PKT>,-           ; Start tributary
                   195D                 <XD_FC_V_INITC  FORK_INITC>,-         ; Start DMP11 controller
                   195D                 <XD_FC_V_STOPT  FORK_STOPT>,-         ; Stop tributary
                   195D                 <XD_FC_V_STOPC  FORK_STOPC>,-         ; Stop DMP11 controller
                   195D                 <XD_FC_V_CHGC   FORK_INITC>,-         ; Change controller parameters
                   195D                 <XD_FC_V_CHGT   FORK_PKT>,-           ; Change tributary parameters
                   195D                 <XD_FC_V_RDMODEM FORK_SSLOT>,-        ; Read modem register
                   195D                 <XD_FC_V_RDTERR FORK_COUNTER>,-       ; Read tributary error counts
                   195D                 <XD_FC_V_RDGERR FORK_COUNTER>,-       ; Read global error counts
                   195D                 <XD_FC_V_RCTERR FORK_COUNTER>,-       ; Read and clear tributary ctrs
                   195D                 <XD_FC_V_RCGERR FORK_COUNTER>,-       ; Read and clear global counts
                   195D                 <XD_FC_V_RDTSS  FORK_SSLOT>,-         ; Read TSS location
                   195D                 <XD_FC_V_RDGSS  FORK_SSLOT>,-         ; Read GSS location
                   195D                 <XD_FC_V_WTMODEM FORK_PKT>,-          ; Write modem request
                   195D                 <XD_FC_V_HALTT  FORK_HALTT>,-         ; Halt tributary
                   195D                 <XD_FC_V_KILLT  FORK_KILLT>,-         ; Kill tributary
                   195D                 <XD_FC_V_CANCEL FORK_CANCEL>,-        ; CANCEL request to halt trib
                   195D                 <XD_FC_V_XMIT   FORK_XMIT>,-          ; Xmit request
                   195D                 <XD_FC_V_RECV   FORK_RECV>,-          ; Receive request
                   195D                 <XD_FC_V_NUMSYNC FORK_INITC>,-        ; Set number of sync characters
                   195D                 <XD_FC_V_POLSTAT FORK_PKT>,-          ; Latch polling state
                   195D                 <XD_FC_V_POLSS  FORK_POLSS>,-         ; Read polling sub-state
                   195D         >
                   198D  ;
                   198D  ; Otherwise if not a valid IRP
                   198D  ;
                   198D         BUG_CHECK NOBUFPCKT,FATAL                     ; Fatal driver error
                   1991
```

```
                            1991 FORK_PKT:                                         ; Process completion on packet
          50    42 A3  3C   1991          MOVZWL  IRP$W_CSTATUS(R3),R0             ; Get status of completion
                 0238  30   1995          BSBW    IO_DONE                          ; Complete the I/O
                 FF4F  31   1998          BRW     10$                              ; Get next packet
                            199B
                            199B FORK_INITC:                                       ; Process ctrl init completion
       39 A3       01  91   199B          CMPB    S^#XD_FC_V_INITC,IRP$L_MEDIA+1(R3) ; Was this an init controller?
                   F0  12   199F          BNEQ    FORK_PKT                         ; Br if not
                42 A3  B5   19A1          TSTW    IRP$W_QUOTA(R3)                  ; Any quota to return?
                   27  13   19A4          BEQL    55$                              ; Br if no
          50    0C A3  3C   19A6          MOVZWL  IRP$L_PID(R3),R0                 ; Get process index from IRP
     51   00000000'GF  D0   19AA          MOVL    G^SCH$GL_PCBVEC,R1               ; Get address of PCB addr vec
          50     6140  D0   19B1          MOVL    (R1)[R0],R0                      ; Get PCB address
                60 A0  D1   19B5          CMPL    PCB$L_PID(R0),-                  ; Still same process?
                0C A3       19B8                  IRP$L_PID(R3)                    ;
                   11  12   19BA          BNEQ    55$                              ; Br if not - forget it
          50     0080 C0  D0 19BC         MOVL    PCB$L_JIB(R0),R0                 ; Get JIB address
           51   42 A3  3C   19C1          MOVZWL  IRP$W_QUOTA(R3),R1               ; Convert quota to longword
           20 A0   51  C0   19C5          ADDL    R1,JIB$L_BYTCNT(R0)              ; Return byte count quota
           24 A0   51  C0   19C9          ADDL    R1,JIB$L_BYTLM(R0)               ; ..and byte limit quota
                            19CD
          50    42 A3  3C   19CD 55$:     MOVZWL  IRP$W_CSTATUS(R3),R0             ; Get completion status
                   50  DD   19D1          PUSHL   R0                               ; Save completion status
                 01FA  30   19D3          BSBW    IO_DONE                          ; Complete the I/O request
                50 8ED0     19D6          POPL    R0                               ; Restore status
                03 50  E9   19D9          BLBC    R0,57$                           ; Br ir error on startup
                 F4FA  30   19DC          BSBW    FILLRCVLIST                      ; Start the receives on success
                 FF08  31   19DF 57$:     BRW     10$                              ; Get next packet
                            19E2
                            19E2 FORK_STOPT:                                       ; Process stop of tributary
                 056B  30   19E2          BSBW    GET_CDB                          ; Get CDB address
                24 50  E9   19E5          BLBC    R0,58$                           ; Br if no CDB
                 026D  30   19E8          BSBW    SHUTDOWN_TRIB1                   ; Abort all I/O
          50       01  3C   19EB          MOVZWL  S^#SS$_NORMAL,R0                 ; Set completion status
                            19EE ;
                            19EE ; Cleanup CDB in case we use it again
                            19EE ;
                0C A9  B4   19EE          CLRW    CDB_W_STS(R9)                    ; Clean up status word
  69     18    08  00  F0   19F1          INSV    #0,#8,#24,CDB_L_DEVDEPEND(R9)    ; Clean up device status
                38 A9  7C   19F6          CLRQ    CDB_L_CNTS(R9)                   ; Clean up counts
                40 A9  7C   19F9          CLRQ    CDB_L_CNTS+8(R9)                 ;       ...
       0F A9     0135 C5  90 19FC         MOVB    UCB$B_XD_XMT_TRB(R5),CDB_B_XMT_CNT(R9) ; Init max xmits allowed
       05 44 A5    06  E0   1A02          BBS     #XM$V_CHR_CTRL,UCB$L_DEVDEPEND(R5),58$ ; Br if control station
       0F A9    7F 8F  90   1A07          MOVB    #INF_XMT,CDB_B_XMT_CNT(R9)       ; Else, allow infinite xmits
                 01C1  30   1A0C 58$:     BSBW    IO_DONE                          ; Complete the I/O request
                 FED8  31   1A0F          BRW     10$                              ; Get next packet
                            1A12
                            1A12 FORK_STOPC:                                       ; Process stop of controller
                 0327  30   1A12          BSBW    SHUTDOWN_DEV                     ; Abort all I/O on all tribs
          50       01  3C   1A15          MOVZWL  S^#SS$_NORMAL,R0                 ; Set completion status
                 01B5  30   1A18          BSBW    IO_DONE                          ; Complete the I/O request
                 FECC  31   1A1B          BRW     10$                              ; Get next packet
                            1A1E
                            1A1E FORK_COUNTER:                                     ; Process counter done
          50    32 A3  3C   1A1E          MOVZWL  IRP$W_BCNT(R3),R0                ; Get size of P2 buffer
                32 A3  B4   1A22          CLRW    IRP$W_BCNT(R3)                   ; Assume error - no data return
             18 42 A3  E9   1A25          BLBC    IRP$W_CSTATUS(R3),70$            ; Br if err when getting counts
```

```
      32 A3    50  B0  1A29             MOVW    R0,IRP$W_BCNT(R3)              ; Else, reset size of data
      50    3A A3  B1  1A2D             CMPW    IRP$L_MEDIA+2(R3),R0           ; Is user buffer larger than P2?
               0A  1E  1A31             BGEQU   60$                            ; Br if yes - okay
      50    3A A3  B0  1A33             MOVW    IRP$L_MEDIA+2(R3),R0           ; Else, set user return size
  42 A3    0601 8F  B0  1A37            MOVW    #SS$_BUFFEROVF,IRP$W_CSTATUS(R3) ; Set error return
  50    50     10  78  1A3D     60$:    ASHL    #16,R0,R0                      ; Move count to high word
      50    42 A3  B0  1A41     70$:    MOVW    IRP$W_CSTATUS(R3),R0           ; Set return status
             0188  30  1A45             BSBW    IO_DONE                        ; Complete the I/O request
             FE9F  31  1A48             BRW     10$                            ; Get next packet
                       1A4B
                       1A4B     FORK_SSLOT:                                    ; Process status slot read
      50    42 A3  3C  1A4B             MOVZWL  IRP$W_CSTATUS(R3),R0           ; Get completion status
            03 50  E8  1A4F             BLBS    R0,75$                         ; Br if success
            32 A3  94  1A52             CLRB    IRP$W_BCNT(R3)                 ; Else, no data to return
             0178  30  1A55     75$:    BSBW    IO_DONE                        ; Complete the I/O request
             FE8F  31  1A58             BRW     10$                            ; Get next packet
                       1A5B
                       1A5B     FORK_XMIT:                                     ; Process transmit done
             04F2  30  1A5B             BSBW    GET_CDB                        ; Get CDB address
            06 50  E8  1A5E             BLBS    R0,78$                         ; Br if CDB present
      42 A3    50  B0  1A61             MOVW    R0,IRP$W_CSTATUS(R3)           ; Else, set error return
               1D  11  1A65             BRB     90$                            ; Return map registers
                       1A67
            0F A9  96  1A67     78$:    INCB    CDB_B_XMT_CNT(R9)              ; Indicate another xmit complete
      16 42 A3     E9  1A6A             BLBC    IRP$W_CSTATUS(R3),90$          ; Br if xmit error
      51    32 A3  3C  1A6E             MOVZWL  IRP$W_BCNT(R3),R1              ; Get size of transmit
      3C A9    51  C0  1A72             ADDL    R1,CDB_L_BSN(R9)               ; Update byte count
               04  1E  1A76             BCC     80$                            ; Br if no overflow
      3C A9    01  CE  1A78             MNEGL   #1,CDB_L_BSN(R9)               ; Else, latch it
                       1A7C     80$:    INCC    CDB_L_DMS(R9)                  ; Update message count
      51    3C A3  9A  1A84     90$:    MOVZBL  IRP$L_MEDIA+4(R3),R1           ; Get mapping slot number used
                       1A88             CLRBIT  R1,UCB$B_XD_XMT_MAP(R5)        ; Clear in use flag
                       1A8E     ;
                       1A8E     ; For all except uVAX I, we must release the UBA map registers.
                       1A8E     ;
                       1A8E             CPUDISP <<790,93$>,-
                       1A8E                     <780,93$>,-
                       1A8E                     <750,93$>,-
                       1A8E                     <730,93$>,-
                       1A8E                     <UV1,96$>>
                       1AA8
      52    24 A5  D0  1AA8     93$:    MOVL    UCB$L_CRB(R5),R2               ; Get CRB address
  34 A2   00E0 C541  D0  1AAC           MOVL    UCB$L_XD_XMT_MAP(R5)[R1],-     ; Setup map register data
                       1AB3                     CRB$L_INTD+VEC$W_MAPREG(R2)    ;
  00E0 C541    01  CE  1AB3             MNEGL   #1,UCB$L_XD_XMT_MAP(R5)[R1]    ; Set mapping data not allocated
                       1AB9             RELMPR                                 ; Release the map registers
                       1ABF
                       1ABF     96$:
      50    42 A3  3C  1ABF             MOVZWL  IRP$W_CSTATUS(R3),R0           ; Get return status
            0B 50  E9  1AC3             BLBC    R0,100$                        ; Br if error
      50    32 A3  B0  1AC6             MOVW    IRP$W_BCNT(R3),R0              ; Get count of bytes transmitted
  50    50     10  78  1ACA             ASHL    #16,R0,R0                      ; Shift into place
      50       01  B0  1ACE             MOVW    S^#SS$_NORMAL,R0               ; Set success
             00FC  30  1AD1     100$:   BSBW    IO_DONE                        ; Complete the I/O
             E824  30  1AD4             BSBW    XMT_ALT_START                  ; Give other xmits to device
             FE10  31  1AD7             BRW     10$                            ; Get next packet
                       1ADA
```

```
                              1ADA  FORK_RECV:                                          ; Process receive request
                0473   30    1ADA           BSBW    GET_CDB                             ; Get CDB address
               0C 50   E9    1ADD           BLBC    R0,110$                             ; Br if error
       50      42 A3   3C    1AE0           MOVZWL  IRP$W_CSTATUS(R3),R0                ; Get completion status
               05 50   E9    1AE4           BLBC    R0,110$                             ; Br if error
                E9A2   30    1AE7           BSBW    RCV_START_ALT                       ; Get any receives waiting
                  03   11    1AEA           BRB     120$                                ; Continue
                             1AEC
                00E1   30    1AEC  110$:    BSBW    IO_DONE                             ; Complete the I/O request
                FDF8   31    1AEF  120$:    BRW     10$                                 ; Get next packet
                             1AF2
                             1AF2  FORK_HALTT:
            2E 42 A3   E9    1AF2           BLBC    IRP$W_CSTATUS(R3),FORK_KILLT        ; Br if error
         21 A3    0F   90    1AF6           MOVB    #XD_FC_V_KILLT,IRP$B_XDFUNC(R3)     ; Set to kill trib
                             1AFA           DSBINT  UCB$B_DIPL(R5)                      ; Sync access to device
     0098 D5      63   0E    1B01           INSQUE  (R3),@UCB$Q_XD_INPUTQ+4(R5)         ; Insert at end of input Q
                F4C7   30    1B06           BSBW    LOAD_PORT                           ; Give request to device
                             1B09           ENBINT                                      ; Restore IPL
                FDDB   31    1B0C           BRW     10$                                 ; Get next packet
                             1B0F
                             1B0F  FORK_CANCEL:                                         ; Process completion of $CANCEL
                043E   30    1B0F           BSBW    GET_CDB                             ; Get CDB address
               0F 50   E9    1B12           BLBC    R0,FORK_KILLT                       ; Br if error
                  0D   E1    1B15           BBC     #XM$V_STS_RUNNING,-                 ; Br if not in
               0B 69         1B17                   CDB_L_DEVDEPEND(R9),FORK_KILLT      ;   run state
             0228 8F   BB    1B19           PUSHR   #^M<R3,R5,R9>                       ; Save registers
                E7DB   30    1B1D           BSBW    XMT_ALT_START                       ; Start up any pending xmits
             0228 8F   BA    1B20           POPR    #^M<R3,R5,R9>                       ; Restore registers
                             1B24
                             1B24  FORK_KILLT:                                          ; Process completion of kill
                00DA   30    1B24           BSBW    POST_IT                             ; Deallocate CDB from IOPOST
                FDC0   31    1B27           BRW     10$                                 ; Get next packet
                             1B2A
                             1B2A  FORK_POLSS:                                          ; Process Poll substate
            22 42 A3   E9    1B2A           BLBC    IRP$W_CSTATUS(R3),130$              ; Br if error
                  06   E1    1B2E           BBC     #XM$V_CHR_CTRL,-                    ; Br if not control station
            1D 44 A5         1B30                    UCB$L_DEVDEPEND(R5),130$           ;    ...
   50   3F A3   08   00   EA    1B33           FFS     #0,#8,IRP$B_POLS(R3),R0             ; Find state
                  15   13    1B39           BEQL    130$                                ; Br if no-state????
                             1B3B           ASSUME  P2B_L_POINTER EQ 0
       51      2C B3   D0    1B3B           MOVL    @IRP$L_SVAPTE(R3),R1                ; Get P2 buffer address
   66 A1     03F3 8F   B0    1B3F           MOVW    #NMA$C_PCCI_PLS,TRIB_PRM_BUFSIZ(R1) ; Stuff parameter code
   68 A1   E5FD CF40   9A    1B45           MOVZBL  POLSS_TBL[R0],TRIB_PRM_BUFSIZ+2(R1) ; Return parameter value
                  02   13    1B4C           BEQL    130$                                ; Br if UNKNOWN
                  04   11    1B4E           BRB     140$                                ; Continue
         32 A3    06   A2    1B50  130$:    SUBW    #6,IRP$W_BCNT(R3)                   ; No polling substate return
         50    32 A3   B0    1B54  140$:    MOVW    IRP$W_BCNT(R3),R0                   ; Assume user has good size buffer
         50    3C A3   B1    1B58           CMPW    IRP$L_MEDIA+4(R3),R0                ; Is user buffer large enough?
                  0E   1E    1B5C           BGEQU   150$                                ; Br if yes
   42 A3     0601 8F   B0    1B5E           MOVW    #SS$_BUFFEROVF,IRP$W_CSTATUS(R3)    ; Return partial success
         50    3C A3   B0    1B64           MOVW    IRP$L_MEDIA+4(R3),R0                ; Set size of return
         32 A3    50   B0    1B68           MOVW    R0,IRP$W_BCNT(R3)                   ;    ...
   50      50     10   78    1B6C  150$:    ASHL    #16,R0,R0                           ; Shift size of buffer return
         50    42 A3   B0    1B70           MOVW    IRP$W_CSTATUS(R3),R0                ; Set success status
                  50   DD    1B74           PUSHL   R0                                  ; Save return status
         52    38 A3   D0    1B76           MOVL    IRP$L_MEDIA(R3),R2                  ; Retrieve P1 buffer address
                  0E   13    1B7A           BEQL    170$                                ; Br if none
```

```
  62   40 A5     7D  1B7C  4750         MOVQ    UCB$B_DEVCLASS(R5),(R2)       ; Else, return characteristics
         03CD    30  1B80  4751         BSBW    GET_CDB                       ; Get CDB address
        04 50    E9  1B83  4752         BLBC    R0,170$                       ; Br if error
  04 A2    69    C8  1B86  4753         BISL    CDB_L_DEVDEPEND(R9),4(R2)     ;       ...
           50  8ED0  1B8A  4754 170$:   POPL    R0                            ; Restore return status
         0040    30  1B8D  4755         BSBW    IO_DONE                       ; Complete the I/O request
         FD57    31  1B90  4756         BRW     10$                           ; Get next packet
                     1B93  4757         .DSABL  LSB
```

```
                          1B93          .SBTTL  FINISH_RCV_IO,  Finish receive I/O processing
                          1B93
                          1B93  ;++
                          1B93  ; FINISH_RCV_IO - Finish receive I/O processing
                          1B93  ;
                          1B93  ; Functional description:
                          1B93  ;
                          1B93  ; This routine completes a receive operation that has been matched with a
                          1B93  ; message block. After the receive has been completed the message free list
                          1B93  ; is filled and a receive is started if needed.
                          1B93  ;
                          1B93  ;  Inputs:
                          1B93  ;
                          1B93  ;         R2 = message buffer address
                          1B93  ;         R3 = I/O packet address
                          1B93  ;         R5 = UCB address
                          1B93  ;         R9 = CDB address
                          1B93  ;
                          1B93  ;         IPL = FIPL
                          1B93  ;
                          1B93  ;  Outputs:
                          1B93  ;
                          1B93  ;         R5 is preserved.
                          1B93  ;
                          1B93  ;         The request is completed via I/O post.
                          1B93  ;--
                          1B93
                          1B93  FINISH_RCV_IO:                                           ; Finish recieve I/O request
   2C A3      52     D0   1B93          MOVL    R2,IRP$L_SVAPTE(R3)                      ; Save block address
   62      48 A2     9E   1B97          MOVAB   RCV_T_DATA(R2),(R2)                      ; Set address of received data
04 A2      38 A3     D0   1B9B          MOVL    IRP$L_MEDIA(R3),4(R2)                    ; Set address of user buffer
           42 A5     A0   1BA0          ADDW    UCB$W_DEVBUFSIZ(R5),-                    ; Adjust receive buffer quota
         0136 C5          1BA3                  UCB$W_XD_QUOTA(R5)
   51      0E A2     B0   1BA6          MOVW    RCV_L_BACC+2(R2),R1                      ; Get size of transfer
   32 A3      51     B1   1BAA          CMPW    R1,IRP$W_BCNT(R3)                        ; Request larger than actual?
              04     1B   1BAE          BLEQU   10$                                      ; Br if no
   51      32 A3     3C   1BB0          MOVZWL  IRP$W_BCNT(R3),R1                        ; Set size to minimum of two
   32 A3      51     B0   1BB4  10$:    MOVW    R1,IRP$W_BCNT(R3)                        ; Set size to transfer
   50      51 10     78   1BB8          ASHL    #16,R1,R0                                ; Set buffer size in status
              07     12   1BBC          BNEQ    20$                                      ; Br if success
   50    0054 8F     B0   1BBE          MOVW    #SS$_CTRLERR,R0                          ; Set data path error
              03     11   1BC3          BRB     30$                                      ; Fill receive list
                          1BC5
      50      01     B0   1BC5  20$:    MOVW    S^#SS$_NORMAL,R0                         ; Set success
              06     10   1BC8  30$:    BSBB    IO_DONE                                  ; Post the I/O request
            F30C     31   1BCA          BRW     FILLRCVLIST                              ; Fill up the receive buffers
                          1BCD  ;
                          1BCD  ; Complete an I/O request packet
                          1BCD  ;
                          1BCD  ABORT_PKT:                                               ; Abort the I/O request
      50      2C     3C   1BCD          MOVZWL  #SS$_ABORT,R0                            ; Return aborted status
                          1BD0  IO_DONE:                                                 ; Complete a transfer I/O request
   38 A3      50     D0   1BD0          MOVL    R0,IRP$L_IOST1(R3)                       ; Set status return and size
                          1BD4  IO_DONE1:                                                ; Alternate entry point
3C A3      44 A5     D0   1BD4          MOVL    UCB$L_DEVDEPEND(R5),IRP$L_IOST2(R3) ; Set other info
   50      44 A3     D0   1BD9          MOVL    IRP$L_CDB(R3),R0                         ; Get address of CDB address
              09     13   1BDD          BEQL    10$                                      ; Br if none
```

```
                            1BDF
             50      60  D0 1BDF              MOVL    (R0),R0                          ; Get CDB address
                     04  13 1BE2              BEQL    10$                              ; Br if none
          3C A3      60  C8 1BE4              BISL    CDB_L_DEVDEPEND(R0),IRP$L_IOST2(R3) ; Set circuit bits
       14 2A A3      07  E1 1BE8      10$:    BBC     #IRP$V_DIAGBUF,IRP$W_STS(R3),20$ ; Br if no diagnostic buffer
    50    4C B3      08  C1 1BED              ADDL3   #8,@IRP$L_DIAGBUF(R3),R0         ; Address buffer past start time
   80    00000000'GF     7D 1BF2              MOVQ    G^EXE$GQ_SYSTIME,(R0)+           ; Insert stop time
          80    0082 C5  3C 1BF9              MOVZWL  UCB$W_ERRCNT(R5),(R0)+           ; Insert error counter
                    0239 30 1BFE              BSBW    REG_DUMP                         ; Dump registers
                            1C01      20$:
                            1C01      POST_IT:                                         ; Post IRP for completion
                            1C01              PUSHQ   R6                               ; Save registers
             50      05  90 1C04              MOVB    #JNL_POST,R0                     ; Set journal type = POST
                   08C3  30 1C07              BSBW    XD_FILL_JNX                      ; Fill the journal buffer
                  04 50  E9 1C0A              BLBC    R0,50$                           ; Br if error
          66      38 A3  7D 1C0D              MOVQ    IRP$L_IOST1(R3),(R6)             ; Store the IOSB
                            1C11      50$:    POPQ    R6                               ; Restore registers
                            1C14
          00000000'GF    17 1C14              JMP     G^COM$POST                       ; Post the I/O and return
```

```
                                 1C1A         .SBTTL  HALT_TRIBS,  Halt all tribs that were forced to shutdown
                                 1C1A
                                 1C1A  ;++
                                 1C1A  ; HALT_TRIBS - Halt all tribs that were forced to shutdown
                                 1C1A  ;
                                 1C1A  ; Functional description:
                                 1C1A  ;
                                 1C1A  ; This routine scans the UCB halt longword for any trib that were recently
                                 1C1A  ; shutdown due to device errors.
                                 1C1A  ;
                                 1C1A  ; Inputs:
                                 1C1A  ;
                                 1C1A  ;         R5 = UCB address
                                 1C1A  ;
                                 1C1A  ;         IPL = FIPL
                                 1C1A  ;
                                 1C1A  ; Outputs:
                                 1C1A  ;
                                 1C1A  ;         R0-R2,R9 are destroyed.
                                 1C1A  ;
                                 1C1A  ;--
                                 1C1A
                                 1C1A  HALT_TRIBS:                                            ; Shutdown all tribs that are halted
 52  029C C5   E532 CF   00   EA 1C1A          FFS     #0,MAX_W_TRIB,UCB$L_XD_HALT(R5),R2 ; Get index of next halted trib
                         13   13 1C23          BEQL    10$                                ; Br if none
                                 1C25          CLRBIT  R2,UCB$L_XD_HALT(R5)               ; Else, indicate trib shutdown
            59   0218 C542   D0 1C2B          MOVL    UCB$L_XD_CDB_VEC(R5)[R2],R9        ; Get CDB address
                         E7   13 1C31          BEQL    HALT_TRIBS                         ; Br if no CDB address
                       0039   30 1C33          BSBW    SHUTDOWN_TRIB                      ; Shutdown the tributary
                         E2   11 1C36          BRB     HALT_TRIBS                         ; Look for next halted trib
                              05 1C38  10$:    RSB                                        ; Return to caller
```

```
                                    1C39          .SBTTL  START_XMITS,  Start xmits for tribs in run state
                                    1C39
                                    1C39   ;++
                                    1C39   ; START_XMITS - Start xmits for tribs in run state
                                    1C39   ;
                                    1C39   ; Functional description:
                                    1C39   ;
                                    1C39   ; This routine scans the UCB run longword for any tribs that have recently
                                    1C39   ; entered the run state, to start the xmit requests for that tributary.
                                    1C39   ;
                                    1C39   ;  Inputs:
                                    1C39   ;
                                    1C39   ;       R5 = UCB address
                                    1C39   ;
                                    1C39   ;       IPL = FIPL
                                    1C39   ;
                                    1C39   ;  Outputs:
                                    1C39   ;
                                    1C39   ;       R0-R2,R9 are destroyed.
                                    1C39   ;
                                    1C39   ;--
                                    1C39
                                    1C39   START_XMITS:                                            ; Start xmits on running tribs
   52   0298 C5   E513 CF   00   EA  1C39           FFS     #0,MAX_W_TRIB,UCB$L_XD_RUN(R5),R2 ; Get index of next running trib
                            13   13  1C42           BEQL    10$                             ; Br if none running
                                     1C44           CLRBIT  R2,UCB$L_XD_RUN(R5)             ; Else, indicate xmits started
           59   0218 C542        D0  1C4A           MOVL    UCB$L_XD_CDB_VEC(R5)[R2],R9     ; Get CDB address
                            E7   13  1C50           BEQL    START_XMITS                     ; Br if no CDB address
                          E6A6   30  1C52           BSBW    XMT_ALT_START                   ; Start xmits for this trib
                            E2   11  1C55           BRB     START_XMITS                     ; Get next trib
                                 05  1C57   10$:    RSB                                     ; Return to caller
```

```
                        1C58         .SBTTL  SHUTDOWN_TRIB,  Shutdown the tributary
                        1C58
                        1C58  ;++
                        1C58  ; SHUTDOWN_TRIB - Shutdown the tributary
                        1C58  ;
                        1C58  ; Functional description:
                        1C58  ;
                        1C58  ; This routine is called to abort all I/O pending for this tributary.
                        1C58  ;
                        1C58  ; Inputs:
                        1C58  ;
                        1C58  ;         R5 = UCB address
                        1C58  ;         R9 = CDB address
                        1C58  ;
                        1C58  ;         IPL = FIPL
                        1C58  ;
                        1C58  ; Outputs:
                        1C58  ;
                        1C58  ;         R0-R2 are destroyed.
                        1C58  ;
                        1C58  ;--
                        1C58
                        1C58  SHUTDOWN_TRIB1:
                        1C58          PUSHQ   R3                              ; Save R3, R4
             12     10  1C5B          BSBB    SHUTDOWN_TRIB                   ; Shutdown the tributary
     53   30 B9     0F  1C5D  20$:    REMQUE  @CDB_Q_INFOUT(R9),R3            ; Get any leftover IRPs
             05     1D  1C61          BVS     50$                             ; Br if none
           FF67     30  1C63          BSBW    ABORT_PKT                       ; Else, abort the IRP
             F5     11  1C66          BRB     20$                             ; Look for more
                        1C68  50$:    POPQ    R3                              ; Restore R3, R4
           07C2     30  1C6B          BSBW    SET_OUTTIM                      ; Reset output wait timer
                    05  1C6E          RSB
                        1C6F
                        1C6F  SHUTDOWN_TRIB:                                  ; Shutdown a tributary
  01 0C A9   00     E0  1C6F          BBS     #CD_TS_V_ESTAB,CDB_W_STS(R9),10$ ; Br if trib established
                    05  1C74          RSB
                        1C75  ;
                        1C75  ; Flush all attention ASTs
                        1C75  ;
        00D8 8F     BB  1C75  10$:    PUSHR   #^M<R3,R4,R6,R7>                ; Save registers
     57   04 A9     9E  1C79  20$:    MOVAB   CDB_L_AST(R9),R7                ; Get address of AST listhead
        50   67     D0  1C7D          MOVL    (R7),R0                         ; Anything in list?
             1B     13  1C80          BEQL    30$                             ; Br if not
     56   22 A0     3C  1C82          MOVZWL  ACB$L_KAST+10(R0),R6            ; Force channel match
     52   24 A0     3C  1C86          MOVZWL  ACB$L_KAST+12(R0),R2            ; Get process index
 54    00000000'GF  D0  1C8A          MOVL    G^SCH$GL_PCBVEC,R4              ; Get PCB address vector address
        54   6442   D0  1C91          MOVL    (R4)[R2],R4                     ; Get PCB address
       00000000'GF  16  1C95          JSB     G^COM$FLUSHATTNS                ; Flush AST
             DC     11  1C9B          BRB     20$                             ; Continue flushing ASTs
                        1C9D  ;
                        1C9D  ; Complete all IRPs for this tributary
                        1C9D  ;
    00002800 8F     CA  1C9D  30$:    BICL    #XM$M_STS_ACTIVE!XM$M_STS_RUNNING,- ; Clear the running
             65         1CA3                  CDB_L_DEVDEPEND(R5)             ;  and active flags
     56   10 A9     9E  1CA4          MOVAB   CDB_Q_QUEUES(R9),R6             ; Get address of first queue listhea
                        1CA8          ASSUME  CDB_Q_INFOUT EQ CDB_Q_QUEUES+<8*<CDB_C_QUEUES-1>>
                        1CA8          ASSUME  CDB_Q_XMT_PND EQ CDB_Q_QUEUES+<8*<CDB_C_QUEUES-2>>
```

```
        57   03   3C  1CA8  4959          MOVZWL  #CDB_C_QUEUES-2,R7              ; Get number of queues
     53   00 B6   0F  1CAB  4960 40$:     REMQUE  @(R6),R3                        ; Get next IRP/buffer
             23   1D  1CAF  4961          BVS     80$                             ; Br if none - queue empty
     0A   0A A3   91  1CB1  4962          CMPB    IRP$B_TYPE(R3),S^#DYN$C_IRP     ; Is this an IRP?
             0A   13  1CB5  4963          BEQL    50$                             ; Br if yes
     17   0A A3   91  1CB7  4964          CMPB    IRP$B_TYPE(R3),S^#DYN$C_NET     ; Is it a receive buffer?
             09   13  1CBB  4965          BEQL    70$                             ; Br if yes
                      1CBD  4966          BUG_CHECK NOBUFPCKT,FATAL               ; Else, fatal error
                      1CC1  4967 ;
                      1CC1  4968 ; IRP
                      1CC1  4969 ;
           FF09   30  1CC1  4970 50$:     BSBW    ABORT_PKT                       ; Abort the I/O request
             E5   11  1CC4  4971          BRB     40$                             ; Get next entry
                      1CC6  4972 ;
                      1CC6  4973 ; Receive buffer
                      1CC6  4974 ;
        52   53   D0  1CC6  4975 70$:     MOVL    R3,R2                           ; Set receive buffer address
          42 A5   A0  1CC9  4976          ADDW    UCB$W_DEVBUFSIZ(R5),-           ; Adjust receive buffer quota
        0136 C5       1CCC  4977                  UCB$W_XD_QUOTA(R5)              ;
           F20E   30  1CCF  4978          BSBW    ADDRCVLIST                      ; Try to add to receive list
             D7   11  1CD2  4979          BRB     40$                             ; Get next entry
                      1CD4  4980 ;
                      1CD4  4981 ; Loop to next queue
                      1CD4  4982 ;
        56   08   C0  1CD4  4983 80$:     ADDL    #8,R6                           ; Skip to next queue listhead
          D1 57   F5  1CD7  4984          SOBGTR  R7,40$                          ; Loop if more queues
                      1CDA  4985 ;
                      1CDA  4986 ; Abort the transmit requests in progress and return the map registers
                      1CDA  4987 ; and transmit slots!
                      1CDA  4988 ;
     53   28 B9   0F  1CDA  4989 90$:     REMQUE  @CDB_Q_XMT_PND(R9),R3           ; Get next xmit request
             26   1D  1CDE  4990          BVS     100$                            ; Br if none
     51   3C A3   9A  1CE0  4991          MOVZBL  IRP$L_MEDIA+4(R3),R1            ; Get mapping slot number used
                      1CE4  4992          CLRBIT  R1,UCB$B_XD_XMT_MAP(R5)         ; Clear in use flag
     52   24 A5   D0  1CEA  4993          MOVL    UCB$L_CRB(R5),R2                ; Get CRB address
34 A2   00E0 C541 D0  1CEE  4994          MOVL    UCB$L_XD_XMT_MAP(R5)[R1],-      ; Setup map register data
                      1CF5  4995                  CRB$L_INTD+VEC$W_MAPREG(R2)
 00E0 C541   01   CE  1CF5  4996          MNEGL   #1,UCB$L_XD_XMT_MAP(R5)[R1]     ; Set mapping data not allocated
                      1CFB  4997          RELMPR                                  ; Release the map registers
           FEC9   30  1D01  4998          BSBW    ABORT_PKT                       ; Abort the I/O request
             D4   11  1D04  4999          BRB     90$                             ; Loop til done
                      1D06  5000 100$:
                      1D06  5001 ;
                      1D06  5002 ; Take special care not to abort any shutdown requests that are pending
                      1D06  5003 ;
     56   30 A9   9E  1D06  5004          MOVAB   CDB_Q_INFOUT(R9),R6             ; Get address of infout queue
        53   66   D0  1D0A  5005 140$:    MOVL    (R6),R3                         ; Get address of next in queue
        56   53   D1  1D0D  5006 150$:    CMPL    R3,R6                           ; At end of queue?
             1D   13  1D10  5007          BEQL    180$                            ; Br if yes, all done
     21 A3   02   91  1D12  5008          CMPB    #XD_FC_V_STOPT,IRP$B_XDFUNC(R3) ; Is this a STOPT request?
             12   13  1D16  5009          BEQL    170$                            ; Br if yes
     21 A3   10   91  1D18  5010          CMPB    #XD_FC_V_CANCEL,IRP$B_XDFUNC(R3) ; Is this a CANCEL request?
             04   12  1D1C  5011          BNEQ    160$                            ; Br if no
     0C A9   04   AA  1D1E  5012          BICW    #CD_TS_M_CANCEL,CDB_W_STS(R9)   ; Indicate that CANCEL is done
        53   63   0F  1D22  5013 160$:    REMQUE  (R3),R3                         ; Else, remove from queue
           FEA5   30  1D25  5014          BSBW    ABORT_PKT                       ; And abort the I/O request
             E0   11  1D28  5015          BRB     140$                            ; Look for more
```

```
            53    63   D0  1D2A  5016 170$:   MOVL    (R3),R3                                   ; Move down list
                  DE   11  1D2D  5017         BRB     150$                                      ; Continue
                           1D2F  5018 ;
                           1D2F  5019 ; Restart output timers if not a fatal device error
                           1D2F  5020 ;
       03 44 A5    10   E0  1D2F  5021 180$:   BBS     #XM$V_ERR_FATAL,UCB$L_DEVDEPEND(R5),190$ ; Br if fatal error
                06F9   30  1D34  5022         BSBW    SET_OUTTIM                                ; Reset output timer
           00D8 8F     BA  1D37  5023 190$:   POPR    #^M<R3,R4,R6,R7>                          ; Restore registers
                       05  1D3B  5024         RSB                                               ; Return to caller
```

```
                   1D3C        .SBTTL  SHUTDOWN_DEV, Shutdown the controller
                   1D3C
                   1D3C  ;++
                   1D3C  ; SHUTDOWN_DEV - Shutdown the controller
                   1D3C  ;
                   1D3C  ; Functional description:
                   1D3C  ;
                   1D3C  ; This routine is called to abort all I/O pending on all tributaries.
                   1D3C  ;
                   1D3C  ; Inputs:
                   1D3C  ;
                   1D3C  ;       R5 = UCB address
                   1D3C  ;
                   1D3C  ;       IPL = FIPL
                   1D3C  ;
                   1D3C  ; Outputs:
                   1D3C  ;
                   1D3C  ;       R0-R2 are destroyed.
                   1D3C  ;
                   1D3C  ;--
                   1D3C
                   1D3C  SHUTDOWN_DEV:                                         ; Shutdown controller
0A 64 A5   04  E1  1D3C          BBC     #UCB$V_ONLINE,UCB$W_STS(R5),10$      ; Br if not online
06 68 A5   00  E0  1D41          BBS     #UCB$V_XD_INITED,UCB$W_DEVSTS(R5),20$ ; Br if device inited
01 68 A5   01  E0  1D46          BBS     #UCB$V_XD_INITING,UCB$W_DEVSTS(R5),20$ ; Br if device initing
           05      1D4B  10$:    RSB
                   1D4C  ;
                   1D4C  ; Abort all I/O for all tribs and deallocate CDBs
                   1D4C  ;
    02D8 8F    BB  1D4C  20$:    PUSHR   #^M<R3,R4,R6,R7,R9>                   ; Save registers
 54    24 A5   D0  1D50          MOVL    UCB$L_CRB(R5),R4                      ; Get CRB address
                   1D54          ASSUME  IDB$L_CSR EQ 0
 54    2C B4   D0  1D54          MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4          ; Get CSR address
01 A4  40 8F   90  1D58          MOVB    #XD_BSEL1_M_MCLR,BSEL1(R4)            ; Master Clear device
                   1D5D          ASSUME  UCB$B_XD_OUTTIM EQ UCB$B_XD_INTIM+1
    02A5 C5    B4  1D5D          CLRW    UCB$B_XD_INTIM(R5)                    ; Stop all timers
                   1D61          ASSUME  UCB$B_XD_DUP EQ UCB$B_XD_PRO+1
                   1D61          ASSUME  UCB$B_XD_CON EQ UCB$B_XD_DUP+1
                   1D61          ASSUME  UCB$B_XD_BFN EQ UCB$B_XD_CON+1
    02A8 C5    D4  1D61          CLRL    UCB$B_XD_PRO(R5)                      ; Reset all line characteristics
    57   1F    9A  1D65          MOVZBL  S^#<MAX_DMP_TRIB-1>,R7                ; Set number of tributaries
59  0218 C547  D0  1D68  30$:    MOVL    UCB$L_XD_CDB_VEC(R5)[R7],R9           ; Get next CDB address
         1A    13  1D6E          BEQL    40$                                   ; Br if none
    0218 C547  D4  1D70          CLRL    UCB$L_XD_CDB_VEC(R5)[R7]              ; Remove CDB from vector
    0158 C547  D4  1D75          CLRL    UCB$L_XD_PID_VEC(R5)[R7]              ; Zero PID entry
    01D8 C547  B4  1D7A          CLRW    UCB$W_XD_CHAN_VEC(R5)[R7]             ; Zero channel entry
    0138 C547  94  1D7F          CLRB    UCB$B_XD_TRIB_VEC(R5)[R7]             ; Reset trib address
       FED1    30  1D84          BSBW    SHUTDOWN_TRIBT                        ; Abort all I/O on CDB
       00D6    30  1D87          BSBW    CLEAR_CDB                             ; Deallocate CDB and restore quota
      DB 57    F4  1D8A  40$:    SOBGEQ  R7,30$                                ; Loop if more tribs
                   1D8D  ;
                   1D8D  ; Clear device status flags
                   1D8D  ;
    0134 C5    94  1D8D          CLRB    UCB$B_XD_TRB_CNT(R5)                  ; No tributaries
                   1D91          DSBINT  UCB$B_DIPL(R5)                        ; Sync access to UCB
         22    AA  1D98          BICW    #UCB$M_INT!UCB$M_POWER,-              ; Reset device status flags
      64 A5        1D9A                  UCB$W_STS(R5)                         ;
```

```
      FFF7 8F   AA  1D9C           BICW    #^C<UCB$M_XD_FORK_PEND>,-              ; Reset all but fork pending
         68 A5      1DA0                   UCB$W_DEVSTS(R5)                       ; ..bit.
                    1DA2           ENBINT                                         ; Restore IPL
                    1DA5   ;
                    1DA5   ; Release the receive and transmit buffer map registers
                    1DA5   ;
   54    24 A5  D0  1DA5           MOVL    UCB$L_CRB(R5),R4                       ; Get CRB address
            57  D4  1DA9           CLRL    R7                                     ; Init slot number
                    1DAB           ASSUME  UCB$L_XD_RCV_MAP+<4*MAX_RCV> EQ UCB$L_XD_XMT_MAP
   56  00C4 C5  9E  1DAB           MOVAB   UCB$L_XD_RCV_MAP(R5),R6                ; Get address of mapping slots
   34 A4    86  D0  1DB0   50$:    MOVL    (R6)+,CRB$L_INTD+VEC$W_MAPREG(R4)      ; Set mapping info.
            0A  19  1DB4           BLSS    60$                                    ; Br if none allocated
                    1DB6           RELMPR                                         ; Release the map registers
   FC A6    01  CE  1DBC           MNEGL   #1,-4(R6)                              ; Reset mapping info
                    1DC0   60$:    CLRBIT  R7,UCB$B_XD_RCV_MAP(R5)                ; Clear mapping slot flag
   E6 57    0E  F2  1DC6           AOBLSS  #MAX_RCV+MAX_XMT,R7,50$                ; Loop if more map registers
                    1DCA   ;
                    1DCA   ; Deallocate all receive buffers and complete all I/O request packets
                    1DCA   ;
   56  0094 C5  9E  1DCA   70$:    MOVAB   UCB$Q_XD_QUEUES(R5),R6                 ; Get address of first queue listhea
      57    06  3C  1DCF           MOVZWL  #UCB$C_XD_QUEUES,R7                    ; Set number of queues
   53    00 B6  0F  1DD2   80$:    REMQUE  @(R6),R3                               ; Get next IRP/buffer
            29  1D  1DD6           BVS     120$                                   ; Br if none - queue empty
   0A    0A A3  91  1DD8           CMPB    IRP$B_TYPE(R3),S^#DYN$C_IRP            ; Is this an IRP?
            0A  13  1DDC           BEQL    90$                                    ; Br if yes
   17    0A A3  91  1DDE           CMPB    IRP$B_TYPE(R3),S^#DYN$C_NET            ; Is it a receive buffer?
            0C  13  1DE2           BEQL    110$                                   ; Br if yes
                    1DE4           BUG_CHECK NOBUFPCKT,FATAL                      ; Else, fatal error
                    1DE8   ;
                    1DE8   ; IRP
                    1DE8   ;
         44 A3  D4  1DE8   90$:    CLRL    IRP$L_CDB(R3)                          ; Definitely no CDB
          FDDF  30  1DEB           BSBW    ABORT_PKT                              ; Abort the I/O request
            E2  11  1DEE           BRB     80$                                    ; Get next entry
                    1DF0   ;
                    1DF0   ; Receive buffer
                    1DF0   ;
      50    53  D0  1DF0   110$:   MOVL    R3,R0                                  ; Set receive buffer address
         42 A5  A0  1DF3           ADDW    UCB$W_DEVBUFSIZ(R5),-                  ; Restore quota
       0136 C5      1DF6                   UCB$W_XD_QUOTA(R5)                     ;
   00000000'GF  16  1DF9           JSB     G^COM$DRVDEALMEM                       ; Deallocate the receive buffer
            D1  11  1DFF           BRB     80$                                    ; Get next entry
                    1E01   ;
                    1E01   ; Loop to next queue
                    1E01   ;
      56    08  C0  1E01   120$:   ADDL    #8,R6                                  ; Skip to next queue listhead
         CB 57  F5  1E04           SOBGTR  R7,80$                                 ; Loop if more queues
                    1E07   ;
                    1E07   ; Restore the buffered I/O quota to the starter
                    1E07   ;
   50  02A0 C5  3C  1E07           MOVZWL  UCB$L_XD_PID(R5),R0                    ; Get process index of starter
51   00000000'GF  D0  1E0C         MOVL    G^SCH$GL_PCBVEC,R1                     ; Get address of PCB address vector
      50  6140  D0  1E13           MOVL    (R1)[R0],R0                            ; Get PCB address of starter
02A0 C5   60 A0  D1  1E17          CMPL    PCB$L_PID(R0),UCB$L_XD_PID(R5)         ; Still the same process?
            16  12  1E1D           BNEQ    130$                                   ; Br if not
   50  0080 C0  D0  1E1F           MOVL    PCB$L_JIB(R0),R0                       ; Get JIB address
   51  0136 C5  3C  1E24           MOVZWL  UCB$W_XD_QUOTA(R5),R1                  ; Convert quota to longword
```

```
  20 A0    51   C0  1E29  5140         ADDL   R1,JIB$L_BYTCNT(R0)       ; Return byte count quota
  24 A0    51   C0  1E2D  5141         ADDL   R1,JIB$L_BYTLM(R0)        ; ..and limit
     0136  C5   B4  1E31  5142         CLRW   UCB$W_XD_QUOTA(R5)        ; Reset quota
     02D8  8F   BA  1E35  5143 130$:   POPR   #^M<R3,R4,R6,R7,R9>       ; Restore registers
                05  1E39  5144         RSB                              ; Return to caller
```

```
                        1E3A  5146          .SBTTL  REG_DUMP,  Device register dump routine
                        1E3A  5147
                        1E3A  5148 ;++
                        1E3A  5149 ; REG_DUMP, Dumps the contents of device registers to a buffer
                        1E3A  5150 ;
                        1E3A  5151 ; Functional description:
                        1E3A  5152 ;
                        1E3A  5153 ; Writes the number of longwords returned, and the contents of the
                        1E3A  5154 ; device registers into a diagnostic or error buffer.
                        1E3A  5155 ;
                        1E3A  5156 ; Inputs:
                        1E3A  5157 ;
                        1E3A  5158 ;       R0 = Diagnostic buffer address
                        1E3A  5159 ;       R5 = UCB address
                        1E3A  5160 ;
                        1E3A  5161 ;       IPL = FIPL
                        1E3A  5162 ;
                        1E3A  5163 ; Outputs:
                        1E3A  5164 ;
                        1E3A  5165 ;       R0 is destroyed.
                        1E3A  5166 ;
                        1E3A  5167 ;       R5 is preserved.
                        1E3A  5168 ;
                        1E3A  5169 ;--
                        1E3A  5170
                        1E3A  5171 REG_DUMP:                                              ; Dump device registers
       80    03    9A   1E3A  5172          MOVZBL  #3,(R0)+                              ; Store # of longwords returned
                        1E3D  5173          ASSUME  UCB$L_XD_SEL4 EQ UCB$L_XD_SEL0+4
  80   02C6 C5     7D   1E3D  5174          MOVQ    UCB$L_XD_SEL0(R5),(R0)+               ; Store all CSR registers
    80   44 A5     D0   1E42  5175          MOVL    UCB$L_DEVDEPEND(R5),(R0)+             ; Store device dependent longword
                   05   1E46  5176          RSB                                           ; Return
```

```
                              1E47          .SBTTL  RETURN_IRP,  Deallocate IRP
                              1E47          .SBTTL  RETURN_CDB,  Deallocate CDB
                              1E47          .SBTTL  CLEAR_CDB,  Deallocate CDB
                              1E47
                              1E47  ;++
                              1E47  ; RETURN_IRP - Deallocate IRP (Called from IOPOST)
                              1E47  ; RETURN_CDB - Deallocate CDB (Called from IOPOST)
                              1E47  ; CLEAR_CDB - Deallocate CDB (Called from CANCEL or the Fork Process)
                              1E47  ;
                              1E47  ; Functional description:
                              1E47  ;
                              1E47  ; These subroutines are called to deallocate the CDB and restore the byte
                              1E47  ; count quota.
                              1E47  ;
                              1E47  ; Inputs:
                              1E47  ;
                              1E47  ;       R5 = IRP address (RETURN_IRP entry only)
                              1E47  ;       R5 = CDB address (RETURN_CDB entry only)
                              1E47  ;       R9 = CDB address (CLEAR_CDB entry only)
                              1E47  ;
                              1E47  ;       IPL = FIPL
                              1E47  ;
                              1E47  ; Outputs:
                              1E47  ;
                              1E47  ;       R0-R2 are destroyed.
                              1E47  ;       R9 destroyed if CLEAR_CDB entry.
                              1E47  ;
                              1E47  ;--
                              1E47
                              1E47  RETURN_IRP:                                           ; Deallocate the IRP
              50    55   D0   1E47          MOVL    R5,R0                                 ; Copy IRP address
          00000000'GF    17   1E4A          JMP     G^COM$DRVDEALMEM                      ; Deallocate the IRP
                              1E50
                              1E50  RETURN_CDB:                                           ; Deallocate the CDB
                    59   DD   1E50          PUSHL   R9                                    ; Save R9
              59    55   D0   1E52          MOVL    R5,R9                                 ; Copy CDB address
       48 A5    38 A5    D0   1E55          MOVL    IRP$L_MEDIA(R5),CDB_L_PID(R5)         ; Move PID to proper place
                    04   10   1E5A          BSBB    CLEAR_CDB                             ; Deallocate CDB
                 59 8ED0      1E5C          POPL    R9                                    ; Restore R9
                         05   1E5F          RSB                                           ; Return to caller (IOPOST)
                              1E60
                              1E60  CLEAR_CDB:                                            ; Deallocate CDB
              50    59   D0   1E60          MOVL    R9,R0                                 ; Copy CDB address
           59    48 A0   D0   1E63          MOVL    CDB_L_PID(R0),R9                      ; Get PID
          00000000'GF    16   1E67          JSB     G^COM$DRVDEALMEM                      ; Deallocate the CDB
              50    59   3C   1E6D          MOVZWL  R9,R0                                 ; Get process index
    51    00000000'GF    D0   1E70          MOVL    G^SCH$GL_PCBVEC,R1                    ; Get address of PCB vector
             51   6140   D0   1E77          MOVL    (R1)[R0],R1                           ; Get PCB address
           59    60 A1   D1   1E7B          CMPL    PCB$L_PID(R1),R9                      ; Is this the same PID?
                    15   12   1E7F          BNEQ    10$                                   ; Br if not - just leave
           51   0080 C1  D0   1E81          MOVL    PCB$L_JIB(R1),R1                      ; Get JIB address
 20 A1    00000060 8F    C0   1E86          ADDL    #CDB_C_LENGTH,JIB$L_BYTCNT(R1)        ; Restore byte count quota
 24 A1    00000060 8F    C0   1E8E          ADDL    #CDB_C_LENGTH,JIB$L_BYTLM(R1)         ; ..and limit
                         05   1E96  10$:    RSB                                           ; Return to caller
```

```
                                 1E97        .SBTTL  FIND_SLOT,  Find an empty slot in translation vector
                                 1E97
                                 1E97  ;++
                                 1E97  ; FIND_SLOT - Find an empty slot in the translation vector
                                 1E97  ;
                                 1E97  ; Functional Description:
                                 1E97  ;
                                 1E97  ; This routine is called to return the index of the first available
                                 1E97  ; slot in the translation vector starting from the hash point.
                                 1E97  ;
                                 1E97  ; Inputs:
                                 1E97  ;
                                 1E97  ;          R3 = IRP address
                                 1E97  ;          R5 = UCB address
                                 1E97  ;
                                 1E97  ; Outputs:
                                 1E97  ;
                                 1E97  ;          R0 = status return
                                 1E97  ;          R1,R2 = destroyed.
                                 1E97  ;
                                 1E97  ; If success:
                                 1E97  ;
                                 1E97  ;          IRP$B_INDEX(R3) = Index of empty slot in translation vector.
                                 1E97  ;
                                 1E97  ;--
                                 1E97
                                 1E97  FIND_SLOT:                                                ; Find empty slot in vector
 51    0C A3    05    00    EF   1E97          EXTZV   #0,#5,IRP$L_PID(R3),R1                    ; Get start hash
       52     E2B0 CF       3C   1E9D          MOVZWL  W^MAX_W_TRIB,R2                           ; Get size of vector
       50     E2AB CF       3C   1EA2          MOVZWL  MAX_W_TRIB,R0                             ; Get number of tribs
          50        50      D2   1EA7          MCOML   R0,R0                                     ; Get complement for modulo
                                 1EAA
           0158 C541        D5   1EAA  10$:    TSTL    UCB$L_XD_PID_VEC(R5)[R1]                  ; Empty slot?
                    0B      13   1EAF          BEQL    20$                                       ; Br if yes
                    51      D6   1EB1          INCL    R1                                        ; Set to next slot
              51    50      CA   1EB3          BICL    R0,R1                                     ; Modulo the trib count
                 F1 52      F5   1EB6          SOBGTR  R2,10$                                    ; Br if more
                    50      D4   1EB9          CLRL    R0                                        ; Indicate error
                    05           1EBB          RSB                                               ; Return in error
                                 1EBC  ;
                                 1EBC  ; Empty slot found - save in IRP
                                 1EBC  ;
           41 A3    51      90   1EBC  20$:    MOVB    R1,IRP$B_INDEX(R3)                        ; Save index
              50    01      3C   1EC0          MOVZWL  S^#SS$_NORMAL,R0                          ; Set success
                    05           1EC3          RSB                                               ; Return to caller
```

```
                                    1EC4          .SBTTL  FIND_TRIB,  Find CDB from trib address
                                    1EC4
                                    1EC4  ;++
                                    1EC4  ; FIND_TRIB - Find CDB from trib address
                                    1EC4  ;
                                    1EC4  ; Functional Description:
                                    1EC4  ;
                                    1EC4  ; This routine is called to return the CDB address for a given tributary
                                    1EC4  ; address. The tributary address vector is scanned to make the comparison.
                                    1EC4  ;
                                    1EC4  ; Inputs:
                                    1EC4  ;
                                    1EC4  ;         R0 = tributary address
                                    1EC4  ;         R5 = UCB address
                                    1EC4  ;
                                    1EC4  ; Outputs:
                                    1EC4  ;
                                    1EC4  ;         R0 = status return
                                    1EC4  ;
                                    1EC4  ; If Success:
                                    1EC4  ;
                                    1EC4  ;         R9 = CDB address
                                    1EC4  ;
                                    1EC4  ;--
                                    1EC4
                                    1EC4  FIND_TRIB:                                              ; Find CDB from trib address
                                    1EC4          PUSHQ   R1                                      ; Save R1, R2
                  52      50    D0  1EC7          MOVL    R0,R2                                   ; Copy trib address
 0138 C5    E282 CF       52    3A  1ECA          LOCC    R2,MAX_W_TRIB,UCB$B_XD_TRIB_VEC(R5) ; Find trib in vector
                          16    13  1ED2          BEQL    10$                                     ; Leave if not found (R0 = 0)
                                    1ED4
             52    0138 C5      9E  1ED4          MOVAB   UCB$B_XD_TRIB_VEC(R5),R2                ; Get address of trib vector
                  51      52    C2  1ED9          SUBL2   R2,R1                                   ; Calculate index
        50   0218 C541          D0  1EDC          MOVL    UCB$L_XD_CDB_VEC(R5)[R1],R0             ; Get CDB address
                          06    13  1EE2          BEQL    10$                                     ; Leave if none
                  59      50    D0  1EE4          MOVL    R0,R9                                   ; Return CDB address
                  50      01    3C  1EE7          MOVZWL  S^#SS$_NORMAL,R0                        ; Set success
                                    1EEA  10$:    POPQ    R1                                      ; Restore R1, R2
                                05  1EED          RSB                                             ; Return to caller
```

```
                       1EEE          .SBTTL  XLATE,  Translate PID and Channel to CDB address
                       1EEE          .SBTTL  XLATE_ALT,  Translate PID and Channel to CDB address
                       1EEE
                       1EEE   ;++
                       1EEE   ; XLATE - Translate PID and Channel to CDB address
                       1EEE   ; XLATE_ALT - Translate PID and Channel to CDB address
                       1EEE   ;
                       1EEE   ; Functional Description:
                       1EEE   ;
                       1EEE   ; This routine is called to return the CDB address for a particular
                       1EEE   ; channel. If the controller is running in point to point mode or
                       1EEE   ; the controller is a tributary then it is assumed that the first
                       1EEE   ; hash into the CDB address vector has the one and only CDB address.
                       1EEE   ;
                       1EEE   ; This routine hashes the low order 5 bits of the PID to start the
                       1EEE   ; search. This should optimize scanning of the PID vector only if
                       1EEE   ; the CDB already exists.
                       1EEE   ;
                       1EEE   ; Inputs:
                       1EEE   ;
                       1EEE   ;       R0 = PID (XLATE_ALT only)
                       1EEE   ;       R3 = IRP address
                       1EEE   ;       R5 = UCB address
                       1EEE   ;
                       1EEE   ; Implicit Inputs:
                       1EEE   ;
                       1EEE   ;       IRP$L_PID(R3) = PID of requestor (XLATE only)
                       1EEE   ;       IRP$W_CHAN(R3) = Channel of requestor
                       1EEE   ;
                       1EEE   ; Outputs:
                       1EEE   ;
                       1EEE   ;       R0 = status return for success of call.
                       1EEE   ;       R9 = CDB address (if success, else zero)
                       1EEE   ;
                       1EEE   ;       R1,R2 are destroyed.
                       1EEE   ;
                       1EEE   ;--
                       1EEE
                       1EEE  XLATE:                                       ; Translate IRP into CDB address
      50   0C A3   D0  1EEE          MOVL    IRP$L_PID(R3),R0             ; Get PID
                       1EF2  XLATE_ALT:                                   ; Alternate translation entry
              53   DD  1EF2          PUSHL   R3                           ; Save IRP address
      53   28 A3   3C  1EF4          MOVZWL  IRP$W_CHAN(R3),R3            ; Get channel
              17   10  1EF8          BSBB    XLATE_PID_CHAN               ; Translate PID and channel
              53 8ED0  1EFA          POPL    R3                           ; Restore IRP address
           10 50   E9  1EFD          BLBC    R0,10$                       ; Br if error
                       1F00   ;
                       1F00   ; Found PID and Channel in translation vector, get CDB address
                       1F00   ;
   41 A3      51   90  1F00          MOVB    R1,IRP$B_INDEX(R3)           ; Save index in IRP
44 A3  0218 C541   DE  1F04          MOVAL   UCB$L_XD_CDB_VEC(R5)[R1],-   ; Store address of CDB address
                       1F0B                  IRP$L_CDB(R3)                ;
           0E A9   90  1F0B          MOVB    CDB_B_TRB_ADDR(R9),-         ; Store trib address also
           40 A3       1F0E                  IRP$Q_STATION(R3)            ;
                   05  1F10  10$:    RSB                                  ; Return to caller
                       1F11   ;+
                       1F11   ; The Following entry point is called by the CANCEL routine also.
```

```
                                1F11  5376 ;
                                1F11  5377 ; Inputs:
                                1F11  5378 ;
                                1F11  5379 ;         R0 = PID
                                1F11  5380 ;         R3 = Channel for request
                                1F11  5381 ;
                                1F11  5382 ; Outputs:
                                1F11  5383 ;
                                1F11  5384 ;         R0 = Status return for request
                                1F11  5385 ;         R9 = CDB address if success, else zero
                                1F11  5386 ;-
                                1F11  5387 XLATE_PID_CHAN:                                ; Translate PID & CHAN into CDB addr
51    50    05    00    EF      1F11  5388         EXTZV   #0,#5,R0,R1                    ; Hash PID into 32 indices
   29 68 A5    02    E0      1F16  5389         BBS     #UCB$V_XD_PTP,UCB$W_DEVSTS(R5),40$ ; Br if point to point
                                1F1B  5390 ;        BBC     #XM$V_CHR_CTRL,UCB$L_DEVDEPEND(R5),40$ ; Br if not control station
          52    51    D0      1F1B  5391         MOVL    R1,R2                          ; Save start hash
  0158 C541    50    D1      1F1E  5392 10$:    CMPL    R0,UCB$L_XD_PID_VEC(R5)[R1]    ; PIDs match?
                08    12      1F24  5393         BNEQ    20$                            ; Br if not
  01D8 C541    53    B1      1F26  5394         CMPW    R3,UCB$W_XD_CHAN_VEC(R5)[R1]   ; Channels match?
                16    13      1F2C  5395         BEQL    40$                            ; Br if yes - got it
                51    D6      1F2E  5396 20$:    INCL    R1                             ; Else, set to next entry
51    FFFFFFE0 8F    CA      1F30  5397         BICL    #^C<31>,R1                     ; Modulo 32
          52    51    D1      1F37  5398         CMPL    R1,R2                          ; Back to start?
                E2    12      1F3A  5399         BNEQ    10$                            ; Br if more to check
    50    0084 8F    3C      1F3C  5400 30$:    MOVZWL  #SS$_DEVOFFLINE,R0             ; Return channel offline
                59    D4      1F41  5401         CLRL    R9                             ; Return zero in R9
                05            1F43  5402         RSB                                    ; Return to caller
                                1F44  5403 ;
                                1F44  5404 ; Found match on PID and Channel
                                1F44  5405 ;
   59    0218 C541    D0      1F44  5406 40$:    MOVL    UCB$L_XD_CDB_VEC(R5)[R1],R9   ; Get CDB address
                F0    13      1F4A  5407         BEQL    30$                            ; Br if no CDB address - error
          50    01    3C      1F4C  5408         MOVZWL  S^#SS$_NORMAL,R0               ; Set successful return status
                05            1F4F  5409         RSB                                    ; Return
```

```
                          1F50            .SBTTL  GET_CDB,  Get CDB address from IRP
                          1F50
                          1F50  ;++
                          1F50  ; GET_CDB - Get CDB address from IRP
                          1F50  ;
                          1F50  ; Inputs:
                          1F50  ;
                          1F50  ;         R3 = IRP address
                          1F50  ;         R5 = UCB address
                          1F50  ;
                          1F50  ; Implicit Inputs:
                          1F50  ;
                          1F50  ;         IRP$L_CDB(R3) = address of CDB address (pointer into UCB vector)
                          1F50  ;
                          1F50  ; Outputs:
                          1F50  ;
                          1F50  ;         R0 = status of request
                          1F50  ;         R9 = CDB address if present in UCB vector
                          1F50  ;
                          1F50  ;--
                          1F50
                          1F50  GET_CDB:                                            ; Get CDB address from IRP
     59    44 A3    D0    1F50            MOVL    IRP$L_CDB(R3),R9                  ; Get address of CDB address
              04    12    1F54            BNEQ    10$                               ; Br if present
                          1F56            BUG_CHECK NOBUFPCKT,FATAL                 ; Else, fatal error
                          1F5A
        50    01    3C    1F5A  10$:      MOVZWL  S^#SS$_NORMAL,R0                  ; Assume good CDB address
        59    69    D0    1F5D            MOVL    (R9),R9                           ; Get CDB address
              03    12    1F60            BNEQ    20$                               ; Br if good
        50    2C    3C    1F62            MOVZWL  #SS$_ABORT,R0                     ; Else, abort the request
              05          1F65  20$:      RSB                                       ; Return to caller
```

```
                               1F66          .SBTTL  CHG_UCB,  Change UCB parameter values
                               1F66
                               1F66  ;++
                               1F66  ; CHG_UCB - Change UCB parameter values
                               1F66  ;
                               1F66  ; Functional description:
                               1F66  ;
                               1F66  ; This routine is called to initialize the UCB with new P1 and P2 buffer
                               1F66  ; characteristics. It is assumed here that the parameters have already
                               1F66  ; been validated.
                               1F66  ;
                               1F66  ; Inputs:
                               1F66  ;
                               1F66  ;        R3 = IRP address
                               1F66  ;        R5 = UCB address
                               1F66  ;
                               1F66  ;        IPL = FIPL
                               1F66  ;
                               1F66  ; Outputs:
                               1F66  ;
                               1F66  ;        R0-R2 = destroyed.
                               1F66  ;
                               1F66  ;--
                               1F66
                               1F66  CHG_UCB:                                              ; Change UCB parameters
                   54    DD    1F66          PUSHL   R4                                    ; Save R4
            51 38  A3    E9    1F68          BLBC    IRP$L_MEDIA(R3),10$                   ; Br if no P1 buffer
                               1F6C  ;
                               1F6C  ; Set new P1 buffer characteristics
                               1F6C  ;
         05 68 A5    00  E0    1F6C          BBS     #UCB$V_XD_INITED,UCB$W_DEVSTS(R5),5$ ; Br if device inited
         42 A5    3A A3  B0    1F71          MOVW    IRP$L_MEDIA+2(R3),UCB$W_DEVBUFSIZ(R5) ; Set new buffer size
                               1F76  5$:     ASSUME  P1_LINE_MASK EQ <^XFF>
 44 A5    18    08    00  F0    1F76          INSV    #0,#8,#24,UCB$L_DEVDEPEND(R5)        ; Reset all Read/Write flags
         44 A5    3C A3  C8    1F7C          BISL    IRP$L_MEDIA+4(R3),UCB$L_DEVDEPEND(R5) ; Set new characteristics
                               1F81  ;
                               1F81  ; Now update UCB based on P1 buffer
                               1F81  ;
                               1F81          ASSUME  NMA$C_LINPR_POI EQ 0
         02A8 C5    94         1F81          CLRB    UCB$B_XD_PRO(R5)                      ; Assume point to point mode
      44 A5    E0 8F    93     1F85          BITB    #XM$M_CHR_DMC!XM$M_CHR_TRIB!-         ; Is this point to point mode?
                               1F8A                  XM$M_CHR_CTRL,UCB$L_DEVDEPEND(R5)     ;
                17    13       1F8A          BEQL    6$                                    ; Br if yes
                               1F8C          ASSUME  NMA$C_LINPR_CON EQ 1
         02A8 C5    96         1F8C          INCB    UCB$B_XD_PRO(R5)                      ; Assume control station
                06    E0       1F90          BBS     #XM$V_CHR_CTRL,-                      ; Br if control mode
         0E 44 A5              1F92                  UCB$L_DEVDEPEND(R5),6$                ;
                               1F95          ASSUME  NMA$C_LINPR_TRI EQ 2
         02A8 C5    96         1F95          INCB    UCB$B_XD_PRO(R5)                      ; Assume trib station
                07    E0       1F99          BBS     #XM$V_CHR_TRIB,-                      ; Br if trib station
         05 44 A5              1F9B                  UCB$L_DEVDEPEND(R5),6$                ;
    02A8 C5    04    90        1F9E          MOVB    #NMA$C_LINPR_DMC,UCB$B_XD_PRO(R5)     ; Must be DMC compatible mode
                               1FA3  6$:     ASSUME  NMA$C_LINCN_NOR EQ 0
         02AA C5    94         1FA3          CLRB    UCB$B_XD_CON(R5)                      ; Assume normal mode
                01    E1       1FA7          BBC     #XM$V_CHR_LOOPB,-                     ; Br if not loopback mode
         04 44 A5              1FA9                  UCB$L_DEVDEPEND(R5),7$                ;
                               1FAC          ASSUME  NMA$C_LINCN_LOO EQ 1
```

```
    02AA C5   96  1FAC          INCB     UCB$B_XD_CON(R5)                     ; Must be loopback mode
                  1FB0   7$:    ASSUME   NMA$C_DPX_FUL EQ 0
    02A9 C5   94  1FB0          CLRB     UCB$B_XD_DUP(R5)                     ; Assume full duplex
         02   E1  1FB4          BBC      #XM$V_CHR_HDPLX,-                    ; Br if not half duplex
   04 44 A5       1FB6                   UCB$L_DEVDEPEND(R5),10$              ;
                  1FB9          ASSUME   NMA$C_DPX_HAL EQ 1
    02A9 C5   96  1FB9          INCB     UCB$B_XD_DUP(R5)                     ; Must be half duplex
                  1FBD
                  1FBD   ;
                  1FBD   ; Set new P2 buffer characteristics
                  1FBD   ;
      54 55   D0  1FBD   10$:   MOVL     R5,R4                                ; Copy UCB address
  51 E0B4 CF  9E  1FC0          MOVAB    LINE_PARAM,R1                        ; Get address of verification table
       01B0   30  1FC5          BSBW     UPDATE_P2                            ; Update the UCB
                  1FC8   ;
                  1FC8   ; Set device characteristics and device mode definition
                  1FC8   ;
      51 01   9A  1FC8          MOVZBL   S^#1,R1                              ; Assume point to point mode
      44 A5   94  1FCB          CLRB     UCB$L_DEVDEPEND(R5)                  ; Reset all characteristics
                  1FCE          ASSUME   NMA$C_LINPR_POI EQ 0
  52 02A8 C5  9A  1FCE          MOVZBL   UCB$B_XD_PRO(R5),R2                  ; Get protocol mode
         21   13  1FD3          BEQL     30$                                  ; Br if point to point mode
                  1FD5
                  1FD5          ASSUME   NMA$C_LINPR_CON EQ 1
         51   D6  1FD5          INCL     R1                                   ; Assume control station mode
         52   D7  1FD7          DECL     R2                                   ; Control station mode?
         07   12  1FD9          BNEQ     15$                                  ; Br if not
                  1FDB          SETBIT   #XM$V_CHR_CTRL,UCB$L_DEVDEPEND(R5) ; Indicate control station
         14   11  1FE0          BRB      30$                                  ;
                  1FE2
                  1FE2   15$:   ASSUME   NMA$C_LINPR_TRI EQ 2
         52   D7  1FE2          DECL     R2                                   ; Tributary station protocol?
         09   12  1FE4          BNEQ     20$                                  ; Br if no
         51   D6  1FE6          INCL     R1                                   ; Set mode to trib station
                  1FE8          SETBIT   #XM$V_CHR_TRIB,UCB$L_DEVDEPEND(R5) ; Indicate tributary station
         07   11  1FED          BRB      30$                                  ; And continue
                  1FEF
                  1FEF   20$:   SETBIT   #XM$V_CHR_DMC,UCB$L_DEVDEPEND(R5)  ; Must be DMC compatible mode
         51   D4  1FF4          CLRL     R1                                   ; Set mode to DMC compatible
                  1FF6
  51  51  01  78  1FF6   30$:   ASHL     #1,R1,R1                             ; Shift for mode definition code
                  1FFA          SETBIT   #XM$V_CHR_HDPLX,UCB$L_DEVDEPEND(R5) ; Assume half duplex mode
                  1FFF          ASSUME   NMA$C_DPX_FUL EQ 0
    02A9 C5   95  1FFF          TSTB     UCB$B_XD_DUP(R5)                     ; Full duplex mode?
         07   12  2003          BNEQ     40$                                  ; Br if no - okay
         51   D6  2005          INCL     R1                                   ; Set new device mode
                  2007          CLRBIT   #XM$V_CHR_HDPLX,UCB$L_DEVDEPEND(R5) ; Set to full duplex mode
 02A4 C5  51  90  200C   40$:   MOVB     R1,UCB$B_XD_MODE(R5)                 ; Save device mode (DMP format)
                  2011          CLRBIT   #XM$V_CHR_LOOPB,UCB$L_DEVDEPEND(R5) ; Clear loopback bit
                  2016          ASSUME   NMA$C_LINCN_NOR EQ 0
    02AA C5   95  2016          TSTB     UCB$B_XD_CON(R5)                     ; Is line in normal mode?
         05   13  201A          BEQL     50$                                  ; Br if yes
                  201C          SETBIT   #XM$V_CHR_LOOPB,UCB$L_DEVDEPEND(R5) ; Set loopback mode
                  2021   50$:   CLRBIT   #UCB$V_XD_PTP,UCB$W_DEVSTS(R5)     ; Assume not pt-pt
 000000C0 8F  D3  2026          BITL     #XM$M_CHR_CTRL!XM$M_CHR_TRIB,-     ; Is this trib or ctrl station?
      44 A5       202C                   UCB$L_DEVDEPEND(R5)                  ;
         05   12  202E          BNEQ     60$                                  ; Br if yes
```

```
             2030  5557          SETBIT  #UCB$V_XD_PTP,UCB$W_DEVSTS(R5)  ; Else, indicate pt-pt
     54 8ED0 2035  5558 60$:     POPL    R4                              ; Restore R4
          05 2038  5559          RSB                                     ; Return to caller
```

```
                          2039          .SBTTL  CHG_CDB,  Change CDB parameter values
                          2039
                          2039  ;++
                          2039  ; CHG_CDB - Change CDB parameter values
                          2039  ;
                          2039  ; Functional description:
                          2039  ;
                          2039  ; This routine is called to initialize the CDB with new P1 and P2 buffer
                          2039  ; characteristics. It is assumed here that the parameters have already
                          2039  ; been validated.
                          2039  ;
                          2039  ; Inputs:
                          2039  ;
                          2039  ;       R3 = IRP address
                          2039  ;       R5 = UCB address
                          2039  ;       R9 = CDB address
                          2039  ;
                          2039  ;       IPL = FIPL
                          2039  ;
                          2039  ; Outputs:
                          2039  ;
                          2039  ;       R1,R2 = destroyed.
                          2039  ;--
                          2039
                          2039  CHG_CDB:                                                ; Validate P2 buffer
                54    DD  2039          PUSHL   R4                                      ; Save R4
          1D 38 A3    E9  203B          BLBC    IRP$L_MEDIA(R3),10$                     ; Br if no P1 buffer
                          203F  ;
                          203F  ; Set new P1 buffer characteristics
                          203F  ;
       FFFFD700 8F    CA  203F          BICL    #^C<P1_TRIB_MASK>,-                     ; Clear all but special
                69        2045                  CDB_L_DEVDEPEND(R9)                     ; ..fields in CDB
       00002800 8F    CA  2046          BICL    #XM$M_STS_ACTIVE!XM$M_STS_RUNNING,-     ; Clear special fields
             3C A3        204C                  IRP$L_MEDIA+4(R3)                       ; ..from IRP
       69    3C A3    C8  204E          BISL    IRP$L_MEDIA+4(R3),CDB_L_DEVDEPEND(R9)   ; Set new characteristics
                          2052          ASSUME  NMA$C_STATE_ON EQ 0
                          2052          ASSUME  NMA$C_STATE_OFF EQ 1
             4F A9    94  2052          CLRB    CDB_B_MST(R9)                           ; Assume MOP
       03 69    00    E0  2055          BBS     #XM$V_CHR_MOP,CDB_L_DEVDEPEND(R9),10$   ; Br if true
             4F A9    96  2059          INCB    CDB_B_MST(R9)                           ; Else, NORMAL mode
                          205C  ;
                          205C  ; Set new P2 buffer characteristics
                          205C  ;
          54    59    D0  205C  10$:    MOVL    R9,R4                                   ; Copy CDB address
    51    E067 CF     9E  205F          MOVAB   TRIB_PARAM,R1                           ; Get address of verification table
              0111    30  2064          BSBW    UPDATE_P2                               ; Update CDB
       51    0E A9    9A  2067          MOVZBL  CDB_B_TRB_ADDR(R9),R1                   ; Get trib address
       40 A3    51    90  206B          MOVB    R1,IRP$Q_STATION(R3)                    ; Return trib address
       52    41 A3    9A  206F          MOVZBL  IRP$B_INDEX(R3),R2                      ; Get vector index
     0138 C542  51    90  2073          MOVB    R1,UCB$B_XD_TRIB_VEC(R5)[R2]            ; Set new trib address
                          2079          CLRBIT  #XM$V_CHR_MOP,CDB_L_DEVDEPEND(R9)       ; Assume not MOP mode
                          207D          ASSUME  NMA$C_STATE_ON EQ 0
                          207D          ASSUME  NMA$C_STATE_OFF EQ 1
          04 4F A9    E8  207D          BLBS    CDB_B_MST(R9),20$                       ; Br if MAINT state is OFF
                          2081          SETBIT  #XM$V_CHR_MOP,CDB_L_DEVDEPEND(R9)       ; Else, indicate MOP
             54 8ED0      2085  20$:    POPL    R4                                      ; Restore R4
                05        2088          RSB                                             ; Return to caller
```

```
                    2089          .SBTTL  VALIDATE_P2,  Validate P2 buffer parameters
                    2089          .SBTTL  VALIDATE_P2_CDB,  Validate P2 buffer with CDB
                    2089          .SBTTL  VALIDATE_P2_UCB,  Validate P2 buffer with UCB
                    2089
                    2089  ;++
                    2089  ; VALIDATE_P2 - Validate P2 buffer parameters
                    2089  ; VALIDATE_P2_CDB - Validate P2 buffer with CDB
                    2089  ; VALIDATE_P2_UCB - Validate P2 buffer with UCB
                    2089  ;
                    2089  ; This routine is called to validate the P2 buffer parameters. The parameters
                    2089  ; are checked against a parameter table which verifies that the minimum value
                    2089  ; and maximum value is not violated, and that invalid status flags are not set.
                    2089  ;
                    2089  ; Inputs:
                    2089  ;
                    2089  ;       R1 = Address of parameter verification table (VALIDATE_P2 entry only)
                    2089  ;       R2 = Status word from UCB or CDB
                    2089  ;       R3 = IRP address
                    2089  ;       R4 = UCB or CDB address for value checking (VALIDATE_P2 entry only)
                    2089  ;       R5 = UCB address
                    2089  ;       R9 = CDB address (VALIDATE_P2_CDB entry only)
                    2089  ;
                    2089  ;       IPL = FIPL
                    2089  ;
                    2089  ; Outputs:
                    2089  ;
                    2089  ;       R0 = status return of parameters
                    2089  ;
                    2089  ; If no error:
                    2089  ;       R1 = Address of parameter verification table
                    2089  ; If error:
                    2089  ;       R1 = Bad parameter value
                    2089  ;
                    2089  ;       All other registers are preserved.
                    2089  ;
                    2089  ;--
                    2089          .ENABL  LSB
                    2089  VALIDATE_P2_CDB:                                  ; Validate P2 buffer with CDB
            54  DD  2089          PUSHL   R4                                ; Save R4
51   E03B CF    9E  208B          MOVAB   TRIB_PARAM,R1                     ; Get address of verification table
     54   59    D0  2090          MOVL    R9,R4                             ; Copy CDB address
          0A    11  2093          BRB     10$                               ;
                    2095  VALIDATE_P2_UCB:                                  ; Validate P2 buffer with UCB
            54  DD  2095          PUSHL   R4                                ; Save R4
51   DFDD CF    9E  2097          MOVAB   LINE_PARAM,R1                     ; Get address of verification table
     54   55    D0  209C          MOVL    R5,R4                             ; Copy UCB address
          04    10  209F  10$:    BSBB    VALIDATE_P2                       ; Do the validation
        54 8ED0     20A1          POPL    R4                                ; Restore R4
            05      20A4          RSB                                       ; Return to caller
                    20A5          .DSABL  LSB
                    20A5
                    20A5  VALIDATE_P2:                                      ; Validate P2 buffer parameters
     03EE 8F    BB  20A5          PUSHR   #^M<R1,R2,R3,R5,R6,R7,R8,R9>      ; Save registers
                    20A9                                                    ; NB:R1 must be on top of stack
56     2C A3    D0  20A9          MOVL    IRP$L_SVAPTE(R3),R6               ; Get system P2 buffer address
          03    12  20AD          BNEQ    10$                               ; Br if a system buffer
        00B6    31  20AF          BRW     150$                              ; Else, leave
```

```
                              20B2  ;
                              20B2  ; Make sure that if trib address given, that it is unique, or belongs to this
                              20B2  ; circuit.
                              20B2  ;
              59   52   D0    20B2  10$:    MOVL    R2,R9                           ; Save status word
         51  0474  8F   3C    20B5          MOVZWL  #NMA$C_PCCI_TRI,R1              ; Get trib address
                 0186   30    20BA          BSBW    UNPACK_P2_BUF                   ; From P2 buffer
              20   50   E9    20BD          BLBC    R0,20$                          ; Br if no trib address
 0138 C5   E08C CF   52   3A    20C0          LOCC    R2,MAX_W_TRIB,UCB$B_XD_TRIB_VEC(R5) ; Try to find same trib address
                   16   13    20C8          BEQL    20$                             ; Br if unique
         58  0138  C5   9E    20CA          MOVAB   UCB$B_XD_TRIB_VEC(R5),R8        ; Else, get address of trib vec
              51   58   C2    20CF          SUBL2   R8,R1                           ; Calculate index
         51   41   A3   91    20D2          CMPB    IRP$B_INDEX(R3),R1              ; If not unique, must be same CEB
                   08   13    20D6          BEQL    20$                             ; Br if same circuit
         50  0474  8F   3C    20D8          MOVZWL  #NMA$C_PCCI_TRI,R0              ; Return bad parameter
                 008D   31    20DD          BRW     170$                            ; Else, error
                              20E0
              52   59   D0    20E0  20$:    MOVL    R9,R2                           ; Restore status word
         58   32   A3   3C    20E3          MOVZWL  IRP$W_BCNT(R3),R8               ; Get size of P2 buffer
              58   06   C6    20E7          DIVL    #6,R8                           ; Calculate number of parameters
              56   66   D0    20EA          MOVL    P2B_L_POINTER(R6),R6            ; Point to start of P2 data
                              20ED  ;
                              20ED  ; Loop to check next parameter in P2 buffer
                              20ED  ;
              50   86   3C    20ED  30$:    MOVZWL  (R6)+,R0                        ; Get parameter type from P2
              55   86   D0    20F0          MOVL    (R6)+,R5                        ; Get parameter value from P2
              57   6E   D0    20F3          MOVL    (SP),R7                         ; Get parameter table address
                              20F6  ;
                              20F6  ; Loop to check P2 buffer parameter to Line parameter table
                              20F6  ;
              59   87   B0    20F6  40$:    MOVW    (R7)+,R9                        ; Get parameter + flags
                   72   13    20F9          BEQL    170$                            ; Br if end of verify table
    50   59   0C   00   ED    20FB          CMPZV   #PRM_V_TYPE,#PRM_S_TYPE,R9,R0   ; Parameters match?
                   16   13    2100          BEQL    50$                             ; Br if yes
                   87   B5    2102          TSTW    (R7)+                           ; Skip offset word
                              2104          SKIP    PRM_V_MIN,R9,R7                 ; Skip minimum value
                              210A          SKIP    PRM_V_MAX,R9,R7                 ; Skip maximum value
                              2110          SKIP    PRM_V_INVALID,R9,R7             ; Skip invalid flags
                   DE   11    2116          BRB     40$                             ; Try next parameter
                              2118  ;
                              2118  ; Match found - nullify if same value & check min,max,valid,invalid
                              2118  ;
              51   87   B0    2118  50$:    MOVW    (R7)+,R1                        ; Get offset + width
                   54   D5    211B          TSTL    R4                              ; Is data structure present?
                   2B   13    211D          BEQL    100$                            ; Br if not - check values
    53   51   02   0E   EF    211F          EXTZV   #OFF_V_WIDTH,#OFF_S_WIDTH,R1,R3 ; Get width only
    51   51   0E   00   EF    2124          EXTZV   #OFF_V_VALUE,#OFF_S_VALUE,R1,R1 ; Get offset only
              51   54   C0    2129          ADDL    R4,R1                           ; Calculate address of datum
                              212C          CASE    R3,TYPE=B,LIMIT=#1,<-           ; Br to handler
                              212C                  60$,-                           ;   Byte value
                              212C                  70$,-                           ;   Word value
                              212C                  80$>                            ;   Longword value
                              2136  ;
                              2136  ; Byte value in structure
                              2136  ;
              61   55   91    2136  60$:    CMPB    R5,(R1)                         ; Is this the same?
                   08   11    2139          BRB     90$                             ; Check result
```

```
                   213B  ;
                   213B  ; Word value
                   213B  ;
   61   55   B1    213B  70$:     CMPW    R5,(R1)                     ; Is this the same?
        03   11    213E           BRB     90$                         ; Check result
                   2140
   61   55   D1    2140  80$:     CMPL    R5,(R1)                     ; Is this the same?
        05   12    2143  90$:     BNEQ    100$                        ; Br if no - continue checks
     FA A6   B4    2145           CLRW    -6(R6)                      ; Nullify the parameter code
        1B   11    2148           BRB     140$                        ; Try next parameter - skip checks
                   214A
05 59   0C   E1    214A  100$:    BBC     #PRM_V_MIN,R9,110$          ; Br if no minimum value
   87   55   B1    214E           CMPW    R5,(R7)+                    ; Is the value too small?
        1A   1F    2151           BLSSU   170$                        ; Br if yes - error
05 59   0D   E1    2153  110$:    BBC     #PRM_V_MAX,R9,130$          ; Br if no maximum value
   87   55   B1    2157           CMPW    R5,(R7)+                    ; Is the value too big?
        11   1A    215A           BGTRU   170$                        ; Br if yes - error
05 59   0E   E1    215C  130$:    BBC     #PRM_V_INVALID,R9,140$      ; Br if no invalid flags
   52   87   B3    2160           BITW    (R7)+,R2                    ; Check invalid bits
        08   12    2163           BNEQ    170$                        ; Br if on - error
     85 58   F5    2165  140$:    SOBGTR  R8,30$                      ; Loop if more parameters
                   2168
   50   01   3C    2168  150$:    MOVZWL  S^#SS$_NORMAL,R0            ; Set success return
        06   11    216B           BRB     180$                        ; And return
                   216D
   6E   50   3C    216D  170$:    MOVZWL  R0,(SP)                     ; Return bad parameter code
                   2170                                               ; * R1 Must be on top of stack
   50   14   3C    2170           MOVZWL  #SS$_BADPARAM,R0            ; Set error return
 03EE   8F   BA    2173  180$:    POPR    #^M<R1,R2,R3,R5,R6,R7,R8,R9> ; Restore registers
             05    2177           RSB                                 ; Return to caller
```

```
                             2178        .SBTTL  UPDATE_P2, Update UCB/CDB based on P2 buffer parameters
                             2178
                             2178  ;++
                             2178  ; UPDATE_P2 - Update UCB/CDB with P2 buffer parameters
                             2178  ;
                             2178  ; This routine is called to update the UCB/CDB with the P2 buffer parameters.
                             2178  ; The parameters are stored in the appropriate cells of the UCB/CDB.
                             2178  ;
                             2178  ; Inputs:
                             2178  ;
                             2178  ;       R1 = Address of parameter verification table
                             2178  ;       R3 = IRP address
                             2178  ;       R4 = UCB or CDB address for storing
                             2178  ;       R5 = UCB address
                             2178  ;
                             2178  ;       IPL = FIPL
                             2178  ;
                             2178  ; Outputs:
                             2178  ;
                             2178  ;       R0 = destroyed.
                             2178  ;       All other registers are preserved.
                             2178  ;
                             2178  ;--
                             2178
                             2178  UPDATE_P2:                                                  ; Update the UCB/CDB
          03E6 8F      BB    2178          PUSHR   #^M<R1,R2,R5,R6,R7,R8,R9>                   ; Save registers
                             217C                                                              ; NB:R1 must be on top of stack
      56    2C A3      D0    217C          MOVL    IRP$L_SVAPTE(R3),R6                         ; Get system P2 buffer address
               62      13    2180          BEQL    80$                                         ; Br if no system buffer
      58    32 A3      3C    2182          MOVZWL  IRP$W_BCNT(R3),R8                           ; Get size of P2 buffer
         58    06      C6    2186          DIVL    #6,R8                                       ; Calculate number of parameters
         56    66      D0    2189          MOVL    P2B_L_POINTER(R6),R6                        ; Point to start of data
                             218C  ;
                             218C  ; Loop to get next parameter from P2 buffer
                             218C  ;
         50    86      3C    218C  10$:    MOVZWL  (R6)+,R0                                    ; Get parameter type from P2
         55    86      D0    218F          MOVL    (R6)+,R5                                    ; Get parameter value from P2
         57    6E      D0    2192          MOVL    (SP),R7                                     ; Get parameter table address
                             2195  ;
                             2195  ; Loop to store buffer parameter in UCB/CDB
                             2195  ;
         59    87      B0    2195  20$:    MOVW    (R7)+,R9                                    ; Get parameter + flags
               47      13    2198          BEQL    70$                                         ; Br if end of verify table
 51   59    0C    00   EF    219A          EXTZV   #PRM_V_TYPE,#PRM_S_TYPE,R9,R1               ; Get type field
         51    50      B1    219F          CMPW    R0,R1                                       ; Parameters match?
               16      13    21A2          BEQL    30$                                         ; Br if yes
               87      B5    21A4          TSTW    (R7)+                                       ; Skip offset word
                             21A6          SKIP    PRM_V_MIN,R9,R7                             ; Skip minimum value
                             21AC          SKIP    PRM_V_MAX,R9,R7                             ; Skip maximum value
                             21B2          SKIP    PRM_V_INVALID,R9,R7                         ; Skip invalid flags
               DB      11    21B8          BRB     20$                                         ; Try next parameter
                             21BA  ;
                             21BA  ; Match found - nullify if same value & check min,max,valid,invalid
                             21BA  ;
         51    87      B0    21BA  30$:    MOVW    (R7)+,R1                                    ; Get offset + width
 52   51    02    0E   EF    21BD          EXTZV   #OFF_V_WIDTH,#OFF_S_WIDTH,R1,R2             ; Get width only
 51   51    0E    00   EF    21C2          EXTZV   #OFF_V_VALUE,#OFF_S_VALUE,R1,R1             ; Get offset only
```

```
51   54   C0  21C7          ADDL    R4,R1                          ; Calculate address of datum
              21CA          CASE    R2,TYPE=B,LIMIT=#1,<-          ; Br to handler
              21CA                  40$,-                          ;   Byte value
              21CA                  50$,-                          ;   Word value
              21CA                  60$>                           ;   Longword value
              21D4    ;
              21D4    ; Byte, word, longword value in structure
              21D4    ;
61   55   90  21D4    40$:  MOVB    R5,(R1)                        ; Store byte value
     08   11  21D7          BRB     70$                            ;
61   55   B0  21D9    50$:  MOVW    R5,(R1)                        ; Store word value
     03   11  21DC          BRB     70$                            ;
61   55   D0  21DE    60$:  MOVL    R5,(R1)                        ; Store longword value
  A8 58   F5  21E1    70$:  SOBGTR  R8,10$                         ; Br if more parameters
              21E4
03E6 8F   BA  21E4    80$:  POPR    #^M<R1,R2,R5,R6,R7,R8,R9>      ; Restore registers
          05  21E8          RSB                                    ; Return to caller
```

```
                          21E9         .SBTTL  RETURN_P2,  Return UCB/CDB buffer parameters
                          21E9
                          21E9   ;++
                          21E9   ; RETURN_P2 - Return P2 buffer parameters
                          21E9   ;
                          21E9   ; This routine is called to return the UCB/CDB buffer parameters.
                          21E9   ;
                          21E9   ; Inputs:
                          21E9   ;
                          21E9   ;       R1 = Address of return table (same format as verification table)
                          21E9   ;       R3 = IRP address
                          21E9   ;       R4 = UCB or CDB address for storing
                          21E9   ;       R5 = UCB address
                          21E9   ;
                          21E9   ; Outputs:
                          21E9   ;
                          21E9   ;       R0 = destroyed.
                          21E9   ;       All other registers are preserved.
                          21E9   ;
                          21E9   ;--
                          21E9
                          21E9   RETURN_P2:                                                      ; Return P2 buffer parameters
             00E6 8F  BB  21E9           PUSHR   #^M<R1,R2,R5,R6,R7>                             ; Save registers
          56   2C A3  D0  21ED           MOVL    IRP$L_SVAPTE(R3),R6                             ; Get system P2 buffer address
                  4B  13  21F1           BEQL    60$                                             ; Br if no system buffer
             56   66  D0  21F3           MOVL    P2B_L_POINTER(R6),R6                            ; Point to start of output buf
                          21F6   ;
                          21F6   ; Loop to return next parameter
                          21F6   ;
             55   81  B0  21F6   10$:    MOVW    (R1)+,R5                                        ; Get parameter + flags
                  43  13  21F9           BEQL    60$                                             ; Br if end of verify table
   57    55   0C  00  EF  21FB           EXTZV   #PRM_V_TYPE,#PRM_S_TYPE,R5,R7                   ; Get type field
                  86  57  B0  2200       MOVW    R7,(R6)+                                        ; Return parameter
                  57  81  B0  2203       MOVW    (R1)+,R7                                        ; Get offset + width
   52    57   02  0E  EF  2206           EXTZV   #OFF_V_WIDTH,#OFF_S_WIDTH,R7,R2                 ; Get width only
   57    57   0E  00  EF  220B           EXTZV   #OFF_V_VALUE,#OFF_S_VALUE,R7,R7                 ; Get offset only
             57   54  C0  2210           ADDL    R4,R7                                           ; Calculate address of datum
                          2213           CASE    R2,TYPE=B,LIMIT=#1,<-                           ; Br to handler
                          2213                   20$,-                                           ;   Byte value
                          2213                   30$,-                                           ;   Word value
                          2213                   40$>                                            ;   Longword value
                          221D   ;
                          221D   ; Byte, word, longword value in structure
                          221D   ;
             86   67  9A  221D   20$:    MOVZBL  (R7),(R6)+                                      ; Store byte value
                  08  11  2220           BRB     50$                                             ;
             86   67  3C  2222   30$:    MOVZWL  (R7),(R6)+                                      ; Store word value
                  03  11  2225           BRB     50$                                             ;
             86   67  D0  2227   40$:    MOVL    (R7),(R6)+                                      ; Store longword value
                          222A   50$:    SKIP    PRM_V_MIN,R5,R1                                 ; Skip minimum value
                          2230           SKIP    PRM_V_MAX,R5,R1                                 ; Skip maximum value
                          2236           SKIP    PRM_V_INVALID,R5,R1                             ; Skip invalid flags
                  B8  11  223C           BRB     10$                                             ; Try for more parameters
                          223E
             00E6 8F  BA  223E   60$:    POPR    #^M<R1,R2,R5,R6,R7>                             ; Restore registers
                      05  2242           RSB                                                     ; Return to caller
```

```
                        2243        .SBTTL  UNPACK_P2_BUF,  Unpack a P2 parameter from P2 buffer
                        2243
                        2243  ;++
                        2243  ; UNPACK_P2_BUF - Unpack a P2 parameter from P2 buffer
                        2243  ;
                        2243  ; Functional description:
                        2243  ;
                        2243  ; This routine is called to get a P2 parameter from the P2 buffer.
                        2243  ;
                        2243  ; Inputs:
                        2243  ;
                        2243  ;       R1 = Parameter type code
                        2243  ;       R3 = IRP address
                        2243  ;       R5 = UCB address
                        2243  ;
                        2243  ; Outputs:
                        2243  ;
                        2243  ;       R0 = Low bit clear if specified Parameter type code is found in P2
                        2243  ;       R2 = Parameter value if success else destroyed
                        2243  ;
                        2243  ;       All other registers are preserved.
                        2243  ;
                        2243  ;--
                        2243
                        2243  UNPACK_P2_BUF:                                  ; Unpack P2 buffer
      00E0 8F      BB   2243          POSHR   #^M<R5,R6,R7>                 ; Save registers
56    2C A3        D0   2247          MOVL    IRP$L_SVAPTE(R3),R6           ; Get system P2 buffer address
         1C        13   224B          BEQL    20$                           ; Br if none
57    32 A3        3C   224D          MOVZWL  IRP$W_BCNT(R3),R7             ; Get size of P2 buffer
   57    06        C6   2251          DIVL    #6,R7                         ; Caculate number of parameters
56    0C A6        9E   2254          MOVAB   P2B_T_DATA(R6),R6             ; Point to start of data
   50    01        3C   2258          MOVZWL  S^#SS$_NORMAL,R0              ; Assume success
                        225B  ;
                        225B  ; Loop to check next parameter in P2 buffer
                        225B  ;
   55    86        3C   225B  10$:    MOVZWL  (R6)+,R5                      ; Get parameter type from P2
   52    86        D0   225E          MOVL    (R6)+,R2                      ; Get parameter value from P2
   55    51        B1   2261          CMPW    R1,R5                         ; Parameters match?
         05        13   2264          BEQL    30$                           ; Br if yes
      F2 57        F5   2266          SOBGTR  R7,10$                        ; Br if more parameters
                        2269
         50        D4   2269  20$:    CLRL    R0                            ; Return error
      00E0 8F      BA   226B  30$:    POPR    #^M<R5,R6,R7>                 ; Restore registers
                   05   226F          RSB                                   ; Return to caller
```

```
                            2270          .SBTTL  POKE_USER,  Deliver attention AST
                            2270
                            2270   ;++
                            2270   ; POKE_USER, Deliver attention AST
                            2270   ;
                            2270   ; Functional description:
                            2270   ;
                            2270   ;      This routine is used to deliver an attention AST if one has been
                            2270   ;      requested.
                            2270   ;
                            2270   ; Inputs:
                            2270   ;
                            2270   ;      R5 = UCB address
                            2270   ;      R9 = CDB address
                            2270   ;
                            2270   ; Outputs:
                            2270   ;
                            2270   ;      R0 = Low bit clear only if user is not notified
                            2270   ;      R1-R4 = destroyed.
                            2270   ;
                            2270   ;--
                            2270
                            2270   POKE_USER:                                         ; Poke user process
                            2270          DSBINT  UCB$B_FIPL(R5)                     ; Sync access to CDB
                    7E  D4  2277          CLRL    -(SP)                              ; Assume failure
           51    04 A9  9E  2279          MOVAB   CDB_L_AST(R9),R1                   ; Get AST listhead
                    61  D5  227D          TSTL    (R1)                               ; Empty?
                    20  13  227F          BEQL    30$                                ; Branch if yes
                    6E  D6  2281          INCL    (SP)                               ; Indicate success
                    54  DD  2283          PUSHL   R4                                 ; Save R4
              54    51  D0  2285          MOVL    R1,R4                              ; Copy listhead address
              51    61  D0  2288   10$:   MOVL    (R1),R1                            ; Address a block
                    0B  13  228B          BEQL    20$                                ; Branch if done
                 44 A5  D0  228D          MOVL    UCB$L_DEVDEPEND(R5),-              ; Change parameter
                 1C A1      2290                  ACB$L_KAST+4(R1)                   ;    return status
                    69  C8  2292          BISL    CDB_L_DEVDEPEND(R9),-              ; OR in circuit status
                 1C A1      2294                  ACB$L_KAST+4(R1)                   ;
                    F0  11  2296          BRB     10$                                ; Continue thru AST blocks
         00000000'GF    16  2298   20$:   JSB     G^COM$DELATTNAST                   ; Deliver the AST's
                 54 8ED0    229E          POPL    R4                                 ; Restore R4
                            22A1
                 50 8ED0    22A1   30$:   POPL    R0                                 ; Return success indicator
                            22A4          ENBINT                                     ; Restore IPL
                    05      22A7          RSB                                        ; Return to caller
```

```
                    22A8          .SBTTL  CANCEL,  Cancel I/O routine
                    22A8
                    22A8  ;++
                    22A8  ; CANCEL, Cancels an I/O operation in progress
                    22A8  ;
                    22A8  ; Functional description:
                    22A8  ;
                    22A8  ;       This routine cancels all I/O on the tributary.
                    22A8  ;
                    22A8  ; Inputs:
                    22A8  ;
                    22A8  ;       R2 = channel number
                    22A8  ;       R3 = current IRP address
                    22A8  ;       R4 = PCB address
                    22A8  ;       R5 = UCB address
                    22A8  ;       R8 = Cancel reason code (zero is for vanila flavored cancel).
                    22A8  ;
                    22A8  ;       IPL = FIPL
                    22A8  ;
                    22A8  ; Outputs:
                    22A8  ;
                    22A8  ;       R0-R3 are destroyed.
                    22A8  ;
                    22A8  ;--
                    22A8
                    22A8  CANCEL:                                          ; Cancel an I/O operation
                    22A8          PUSHQ   R6                               ; Save registers
        0240   30   22AB          BSBW    XD_BUF_JNX                       ; Get a journal buffer
       09 50   E9   22AE          BLBC    R0,1$                            ; Br if error
   86    0A    90   22B1          MOVB    #JNL_CANC,(R6)+                  ; Set journal type = CANCEL
   86    52    B0   22B4          MOVW    R2,(R6)+                         ; Enter channel number
   86    58    B0   22B7          MOVW    R8,(R6)+                         ; Enter reason code
                    22BA  1$:     POPQ    R6                               ; Restore registers
                    22BD
                    22BD          ASSUME  CAN$C_CANCEL EQ 0
                    22BD          ASSUME  CAN$C_DASSGN EQ 1
                    22BD          PUSHQ   R8                               ; Save R8, R9
       5C A5   B5   22C0          TSTW    UCB$W_REFC(R5)                   ; Last reference to unit?
          0D   12   22C3          BNEQ    5$                               ; Br if no
                    22C5  ;
                    22C5  ; Last deassign on unit
                    22C5  ;
  53   24 A5   D0   22C5          MOVL    UCB$L_CRB(R5),R3                 ; Get CRB address
        FA70   30   22C9          BSBW    SHUTDOWN_DEV                     ; Shutdown the controller
        0122   30   22CC          BSBW    INIT_UCB                         ; Re-initialize the UCB
        00B8   31   22CF          BRW     20$                              ; And leave
                    22D2
  50   60 A4   D0   22D2  5$:     MOVL    PCB$L_PID(R4),R0                 ; Get PID
     53   52   D0   22D6          MOVL    R2,R3                            ; Copy channel number
        FC35   30   22D9          BSBW    XLATE_PID_CHAN                   ; Translate PID and channel
       59 50   E9   22DC          BLBC    R0,8$                            ; Br if none
          58   D5   22DF          TSTL    R8                               ; Is this a plain cancel?
          57   13   22E1          BEQL    10$                              ; Br if yes
                    22E3  ;
                    22E3  ; Deassign channel request
                    22E3  ;
   0218 C541   D4   22E3          CLRL    UCB$L_XD_CDB_VEC(R5)[R1]         ; Remove CDB address from vector
```

```
         0158 C541  D4 22E8          CLRL    UCB$L_XD_PID_VEC(R5)[R1]            ; Zero PID entry
         01D8 C541  B4 22ED          CLRW    UCB$W_XD_CHAN_VEC(R5)[R1]           ; Zero channel entry
         0138 C541  94 22F2          CLRB    UCB$B_XD_TRIB_VEC(R5)[R1]           ; Reset trib address
            0134 C5  97 22F7         DECB    UCB$B_XD_TRB_CNT(R5)                ; One less tributary
     06 0C A9    00  E0 22FB         BBS     #CD_TS_V_ESTAB,CDB_W_STS(R9),7$     ; Br if trib established
            FB5D  30 2300            BSBW    CLEAR_CDB                           ; Clean up CDB
            0084  31 2303            BRW     20$                                 ; Pop registers and return
                     2306
            0085  30 2306    7$:     BSBW    DO_CANCEL                           ; Abort UCB xmit I/O
            F94C  30 2309            BSBW    SHUTDOWN_TRIB1                      ; Clear all CDB I/O
                     230C    ;
                     230C    ; Turn CDB into IRP to halt/kill trib
                     230C    ;
      50    48 A9  D0 230C           MOVL    CDB_L_PID(R9),R0                    ; Save PID
      51    0E A9  90 2310           MOVB    CDB_B_TRB_ADDR(R9),R1               ; Save trib address
         52    59  D0 2314           MOVL    R9,R2                               ; Copy pointer to IRP
            00B6  30 2317            BSBW    BLD_IRP                             ; Build the IRP
      38 A3    50  D0 231A           MOVL    R0,IRP$L_MEDIA(R3)                  ; Save PID in IRP
      40 A3    51  90 231E           MOVB    R1,IRP$Q_STATION(R3)                ; Store trib address
      21 A3    0E  90 2322           MOVB    #XD_FC_V_HALTT,IRP$B_XDFUNC(R3)     ; Set to halt tributary
                     2326            DSBINT  UCB$B_DIPL(R5)                      ; Sync access to device
   0094 C5    63   0E 232D           INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)            ; Insert at front of input Q
            EC9B  30 2332            BSBW    LOAD_PORT                           ; Give request to device
                     2335            ENBINT                                      ; Restore IPL
              50  11 2338    8$:     BRB     20$                                 ;
                     233A    ;
                     233A    ; Cancel request
                     233A    ;
   58    0218 C541  DE 233A  10$:    MOVAL   UCB$L_XD_CDB_VEC(R5)[R1],R8         ; Save address of trib address
     45 0C A9    00  E1 2340         BBC     #CD_TS_V_ESTAB,CDB_W_STS(R9),20$ ; Br if trib not established
      51    00C4 8F  3C 2345         MOVZWL  #IRP$C_LENGTH,R1                    ; Set size of IRP
                     234A    ;
                     234A    ; NOTE - We must use EXE$ALONONPAGED because the EXE$ALLOCIRP resets the
                     234A    ;        IPL to ASTDEL.
                     234A    ;
        00000000'GF  16 234A         JSB     G^EXE$ALONONPAGED                   ; Allocate an IRP
             09 50  E8 2350          BLBS    R0,15$                              ; Br if success
      50    0124 8F  3C 2353         MOVZWL  #SS$_INSFMEM,R0                     ; Else, return error
                     2358            SETIPL  #0                                  ; Enable interrupts
                  04 235B            RET                                         ; RETURN to caller
                     235C
            0071  30 235C    15$:    BSBW    BLD_IRP                             ; Build the IRP
      44 A3    58  D0 235F           MOVL    R8,IRP$L_CDB(R3)                    ; Save address of CDB address
   0C A3    FAE0 CF  9E 2363         MOVAB   W^RETURN_IRP,IRP$L_PID(R3)          ; Set return address in IRP
   40 A3    0E A9  90 2369           MOVB    CDB_B_TRB_ADDR(R9),IRP$Q_STATION(R3) ; Store trib address
      21 A3    10  90 236E           MOVB    #XD_FC_V_CANCEL,IRP$B_XDFUNC(R3)  ; Set function request
                     2372            DSBINT  UCB$B_DIPL(R5)                      ; Sync access to device
   0094 C5    63   0E 2379           INSQUE  (R3),UCB$Q_XD_INPUTQ(R5)            ; Insert at front of input Q
      0C A9    08  AA 237E           BICW    #CD_TS_M_BUFRET,CDB_W_STS(R9)       ; Clear BUFRET flag
            EC4B  30 2382            BSBW    LOAD_PORT                           ; Give request to device
                     2385            ENBINT                                      ; Restore IPL
              04  10 2388            BSBB    DO_CANCEL                           ; Abort all IRPs
                     238A    20$:    POPQ    R8                                  ; Restore R8, R9
                  05 238D            RSB                                         ; Return to caller
                     238E
                     238E    DO_CANCEL:                                          ; Do the cancel on waiting I/O
                     238E                                                        ; ..pending I/O is canceled
```

```
                              238E                                                    ; ..when trib is halted.
        52   009C C5    DE    238E         MOVAL   UCB$Q_XD_XMT_REQ(R5),R2            ; Get address of xmit request Q
                  0A    10    2393         BSBB    CHECK_PKT                          ; Run down the queue
        52     18 A9    DE    2395         MOVAL   CDB_Q_RCV_REQ(R9),R2               ; Get address of recv IRP Q
                  04    10    2399         BSBB    CHECK_PKT                          ; Run down the queue
        52     10 A9    DE    239B         MOVAL   CDB_Q_XMT_REQ(R9),R2               ; Get address of xmit IRP Q
                              239F                                                    ; Run down the queue
                              239F CHECK_PKT:
                              239F ;
                              239F ; NOTE: The xmit request queue on the UCB must only be accessed at FIPL
                              239F ;
        51     52       D0    239F         MOVL    R2,R1                              ; Copy listhead address
                              23A2         ASSUME  IRP$L_IOQFL EQ 0
        51     61       D0    23A2 10$:    MOVL    (R1),R1                            ; Get next in list
        51     52       D1    23A5         CMPL    R2,R1                              ; End of queue?
               25       13    23A8         BEQL    20$                                ; Br ir yes
            0C A1       D5    23AA         TSTL    IRP$L_PID(R1)                      ; Is this an Internal IRP?
               0A       14    23AD         BGTR    15$                                ; Br if NO - PID must match
     48 A9   02A0 C5    D1    23AF         CMPL    UCB$L_XD_PID(R5),CDB_L_PID(R9)     ; Else, ABORT all the Starter's
               EB       12    23B5         BNEQ    10$                                ;  IRPs - but only on a match
               07       11    23B7         BRB     17$                                ; Check the CHAN as well
     48 A9     0C A1    D1    23B9 15$:    CMPL    IRP$L_PID(R1),CDB_L_PID(R9)        ; Do the PIDs match?
               E2       12    23BE         BNEQ    10$                                ; Br if no - try next packet
     4C A9     28 A1    B1    23C0 17$:    CMPW    IRP$W_CHAN(R1),CDB_W_CHAN(R9)      ; Do the Channels match?
               DB       12    23C5         BNEQ    10$                                ; Br if no - try next packet
                              23C7         ASSUME  IRP$L_IOQFL EQ 0
        53     61       0F    23C7         REMQUE  (R1),R3                            ; Remove packet from queue
             F800       30    23CA         BSBW    ABORT_PKT                          ; Abort the IRP
               D0       11    23CD         BRB     CHECK_PKT                          ; Check entire queue
               05             23CF 20$:    RSB                                        ; Return
                              23D0
                              23D0 ;++
                              23D0 ; BLD_IRP - Build an IRP
                              23D0 ;
                              23D0 ; Inputs:
                              23D0 ;
                              23D0 ;       R2 = address of IRP
                              23D0 ;       R5 = UCB address
                              23D0 ;
                              23D0 ; Outputs:
                              23D0 ;
                              23D0 ;       R2 = Address of IRP$Q_STATION in IRP
                              23D0 ;       R3 = Original address of IRP
                              23D0 ;--
                              23D0
                              23D0 BLD_IRP:                                           ; Build an IRP
        53     82       7E    23D0         MOVAQ   (R2)+,R3                           ; Save IRP address, skip to size fie
                              23D3         ASSUME  IRP$W_SIZE EQ 8
                              23D3         ASSUME  IRP$B_TYPE EQ IRP$W_SIZE+2
                              23D3         ASSUME  IRP$B_RMOD EQ IRP$B_TYPE+1
   82   000A0060 8F     D0    23D3         MOVL    #<<DYN$C_IRP@16>!CDB_C_LENGTH>,(R2)+ ; Make it look like an IRP
                              23DA         ASSUME  IRP$L_PID EQ IRP$B_RMOD+1
        82   FA72 CF    9E    23DA         MOVAB   W^RETURN_CDB,(R2)+                 ; Store return address from IOPOST
                              23DF         ASSUME  IRP$L_AST EQ IRP$L_PID+4
                              23DF         ASSUME  IRP$L_ASTPRM EQ IRP$L_AST+4
               82       7C    23DF         CLRQ    (R2)+                              ; Clear AST, ASTPRM
                              23E1         ASSUME  IRP$L_WIND EQ IRP$L_ASTPRM+4
```

```
         82  D4  23E1        CLRL    (R2)+                              ; Clear WIND
                 23E3        ASSUME  IRP$L_UCB EQ IRP$L_WIND+4
     82  55  D0  23E3        MOVL    R5,(R2)+                           ; Store UCB address
                 23E6        ASSUME  IRP$W_FUNC EQ IRP$L_UCB+4
                 23E6        ASSUME  IRP$B_EFN EQ IRP$W_FUNC+2
                 23E6        ASSUME  IRP$B_PRI EQ IRP$B_EFN+1
                 23E6        ASSUME  IRP$L_IOSB EQ IRP$B_PRI+1
         82  7C  23E6        CLRQ    (R2)+                              ; Clear FUNC, EFN, PRI, IOSB
                 23E8        ASSUME  IRP$W_CHAN EQ IRP$L_IOSB+4
                 23E8        ASSUME  IRP$W_STS EQ IRP$W_CHAN+2
                 23E8        ASSUME  IRP$L_SVAPTE EQ IRP$W_STS+2
         82  7C  23E8        CLRQ    (R2)+                              ; Clear CHAN, STS, SVAPTE
                 23EA        ASSUME  IRP$W_BOFF EQ IRP$L_SVAPTE+4
                 23EA        ASSUME  IRP$W_BCNT EQ IRP$W_BOFF+2
                 23EA        ASSUME  IRP$L_MEDIA EQ IRP$Q_BCNT+6        ; BCNT is really a longword
         82  7C  23EA        CLRQ    (R2)+                              ; Clear BOFF, BCNT
         82  7C  23EC        CLRQ    (R2)+                              ; Clear MEDIA
                 23EE        ASSUME  IRP$Q_STATION EQ IRP$L_MEDIA+8
         82  7C  23EE        CLRQ    (R2)+                              ; Clear STATION
             05  23F0        RSB                                        ; Return to caller
```

XDD
VAX

Mac

_$2
_$2
_$2
TOT

385

The

MAC

```
                          23F1  6180          .SBTTL  INIT_UCB,  Initialize UCB parameters
                          23F1  6181
                          23F1  6182  ;++
                          23F1  6183  ; INIT_UCB - Initialize UCB parameters
                          23F1  6184  ;
                          23F1  6185  ; Functional description:
                          23F1  6186  ;
                          23F1  6187  ; This routine is called to reset the UCB default parameters when
                          23F1  6188  ; the last channel is deassigned or when the unit is initialized.
                          23F1  6189  ;
                          23F1  6190  ; Inputs:
                          23F1  6191  ;
                          23F1  6192  ;         R5 = UCB address
                          23F1  6193  ;
                          23F1  6194  ;         IRP = FIPL or higher
                          23F1  6195  ;
                          23F1  6196  ; Outputs:
                          23F1  6197  ;
                          23F1  6198  ;         R0-R2 are destroyed.
                          23F1  6199  ;
                          23F1  6200  ;--
                          23F1  6201
                          23F1  6202  INIT_UCB:                                               ; Initialize the UCB
                          23F1  6203          ASSUME  UCB$B_XD_OUTTIM EQ UCB$B_XD_INTIM+1
         02A5 C5      B4  23F1  6204          CLRW    UCB$B_XD_INTIM(R5)                      ; Stop all timers
                          23F5  6205          ASSUME  UCB$B_XD_DUP EQ UCB$B_XD_PRO+1
                          23F5  6206          ASSUME  UCB$B_XD_CON EQ UCB$B_XD_DUP+1
                          23F5  6207          ASSUME  UCB$B_XD_BFN EQ UCB$B_XD_CON+1
         02A8 C5      D4  23F5  6208          CLRL    UCB$B_XD_PRO(R5)                        ; Reset all line characteristics
   42 A5 0200 8F      B0  23F9  6209          MOVW    #XD_DEF_BUFSIZ,UCB$W_DEVBUFSIZ(R5) ; Set default buffer size
              04      90  23FF  6210          MOVB    #MAX_XMT_TRB,-                          ; Set maximum number of xmits
         0135 C5          2401  6211                  UCB$B_XD_XMT_TRB(R5)                    ; ..per tributary
         50   0F      3C  2404  6212          MOVZWL  #DEF_LINE_PARAMSZ,R0                    ; Set size of defaults in bytes
   51    DD2D CF      9E  2407  6213          MOVAB   DEF_LINE_PARAM,R1                       ; Set address of defaults
   52    02A7 C5      9E  240C  6214          MOVAB   UCB$B_XD_SETPRM(R5),R2                  ; Set address of parameters
         82   81      90  2411  6215  10$:    MOVB    (R1)+,(R2)+                             ; Set next default
      FA 50           F5  2414  6216          SOBGTR  R0,10$                                  ; Loop on all parameters
              05          2417  6217          RSB
```

```
                                 2418          .SBTTL  Timer setup routines
                                 2418
                                 2418  ;++
                                 2418  ; Timer setup routines
                                 2418  ;
                                 2418  ; Functional description:
                                 2418  ;
                                 2418  ; These routines set/reset the input and output wait timers.
                                 2418  ;
                                 2418  ; Inputs:
                                 2418  ;
                                 2418  ;       R5 = UCB address
                                 2418  ;
                                 2418  ; Outputs:
                                 2418  ;
                                 2418  ;       R0,R1 are destroyed.
                                 2418  ;
                                 2418  ;--
                                 2418
                                 2418  SET_INTIM:                                          ; Start input timer
       51     0094 C5      9E    2418          MOVAB   UCB$Q_XD_INPUTQ(R5),R1              ; Get address of input queue
                51   61    D1    241D          CMPL    (R1),R1                             ; Anything on queue?
                     01    12    2420          BNEQ    20$                                 ; Br if yes - reset timer
                     05          2422  10$:    RSB                                         ; Else, exit
                                 2423
            02A5 C5        95    2423  20$:    TSTB    UCB$B_XD_INTIM(R5)                  ; Timer already going?
                     F9    12    2427          BNEQ    10$                                 ; Br if yes - exit
            02A5 C5   03   90    2429          MOVB    #3,UCB$B_XD_INTIM(R5)               ; Reset timeout cell
                     41    11    242E          BRB     START_TIMER                         ; Start timer if needed
                                 2430
                                 2430  SET_OUTTIM:                                         ; Start output timer
                                 2430          DSBINT  UCB$B_DIPL(R5)                      ; Disable device interrupts
            02A6 C5        94    2437          CLRB    UCB$B_XD_OUTTIM(R5)                 ; Clear output timer
       51     00AC C5      9E    243B          MOVAB   UCB$Q_XD_INFOUT(R5),R1              ; Get address of UCB output queue
                51   61    D1    2440          CMPL    (R1),R1                             ; Anything on queue?
                     24    12    2443          BNEQ    30$                                 ; Br if yes - reset timer
       50     DD08 CF      3C    2445          MOVZWL  MAX_W_TRIB,R0                       ; Get maximum number of CDBs
                     50    D7    244A          DECL    R0                                  ;   Minus 1
      51     0218 C540     D0    244C  10$:    MOVL    UCB$L_XD_CDB_VEC(R5)[R0],R1         ; Get address of next CDB
                     0E    13    2452          BEQL    20$                                 ; Br if none
       09 0C A1      00    E1    2454          BBC     #CD_TS_V_ESTAB,CDB_W_STS(R1),20$ ; Br if trib not established
          51   30 A1       9E    2459          MOVAB   CDB_Q_INFOUT(R1),R1                 ; Get address of output queue
                51   61    D1    245D          CMPL    (R1),R1                             ; Anything on queue?
                     07    12    2460          BNEQ    30$                                 ; Br if yes - reset timer
             E7   50       F4    2462  20$:    SOBGEQ  R0,10$                              ; Loop on all CDBs
                                 2465          ENBINT                                      ; Restore IPL
                     05          2468          RSB                                         ; Else, exit
                                 2469
                                 2469  30$:    ENBINT                                      ; Restore IPL
            02A6 C5   0A   90    246C          MOVB    #10,UCB$B_XD_OUTTIM(R5)             ; Reset timeout cell
                                 2471          ;BRB    START_TIMER                         ; Start timer
                                 2471
                                 2471  START_TIMER:                                        ; Startup timer
          0D 64 A5    00   E0    2471          BBS     #UCB$V_TIM,UCB$W_STS(R5),10$        ; Br if timer going
  6C A5   00000000'GF     02  C1 2476          ADDL3   #2,G^EXE$GL_ABSTIM,UCB$L_DUETIM(R5) ; Set 2 second timer
             64 A5    03   A8    247F          BISW    #UCB$M_TIM!UCB$M_INT,UCB$W_STS(R5) ; Enable timer
                      05         2483  10$:    RSB
```

```
                    2484           .SBTTL  TIMEOUT,  Timeout Routine
                    2484
                    2484  ;++
                    2484  ; TIMEOUT - Timeout Routine
                    2484  ;
                    2484  ; Functional description:
                    2484  ;
                    2484  ; This routine is entered on tributary timeout.  The action is to set
                    2484  ; the error status and branch to the error routine.
                    2484  ;
                    2484  ; Inputs:
                    2484  ;
                    2484  ;       R5 = UCB address
                    2484  ;
                    2484  ; Outputs:
                    2484  ;
                    2484  ;       R3,R4 are destroyed.
                    2484  ;       R5 is preserved.
                    2484  ;
                    2484  ;--
                    2484
                    2484  TIMEOUT:
   64 A5    03  AA  2484           BICW    #UCB$M_TIM!UCB$M_INT,UCB$W_STS(R5) ; Disable timer
3F 68 A5    00  E1  2488           BBC     #UCB$V_XD_INITED,UCB$W_DEVSTS(R5),40$ ; Br if not inited
   53       01  15  78  248D       ASHL    #XD_BSEL2_V_ERR+16,#1,R3           ; Set error indicator
   54       01  10  78  2491       ASHL    #16,#1,R4                          ; Assume powerfail
2F 64 A5    05  E0  2495           BBS     #UCB$V_POWER,UCB$W_STS(R5),30$     ; Br if powerfail
   02A5 C5      95  249A           TSTB    UCB$B_XD_INTIM(R5)                 ; Input timer going?
        09      13  249E           BEQL    10$                                ; Br if not
   02A5 C5      97  24A0           DECB    UCB$B_XD_INTIM(R5)                 ; Input wait timeout?
        0F      13  24A4           BEQL    20$                                ; Br if yes - error
     C8 AF      9F  24A6           PUSHAB  B^START_TIMER                      ; Else, remember to restart timer
   02A6 C5      95  24A9  10$:     TSTB    UCB$B_XD_OUTTIM(R5)                ; Output timer going?
        1D      13  24AD           BEQL    40$                                ; Br if not - exit
   02A6 C5      97  24AF           DECB    UCB$B_XD_OUTTIM(R5)                ; Output wait timeout?
        BC      12  24B3           BNEQ    START_TIMER                        ; Br it not - restart timer
   54    24 A5  D0  24B5  20$:     MOVL    UCB$L_CRB(R5),R4                   ; Get CRB address
                    24B9           ASSUME  IDB$L_CSR EQ 0
   54    2C B4  D0  24B9           MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4       ; Get CSR address
01 A4    40 8F  90  24BD           MOVB    #XD_BSEL1_M_MCLR,BSEL1(R4)         ; Stop the device
           54   D4  24C2           CLRL    R4                                 ; Indicate timeout
                    24C4           SETBIT  #XM$V_STS_TIMO,UCB$L_DEVDEPEND(R5) ; Set error status
      F3D7      30  24C9  30$:     BSBW    SCHED_FORK                         ; Schedule a fork process
                05  24CC  40$:     RSB                                        ; Return to caller
```

```
                          24CD          .SBTTL  XD_FILL_JNX,  Fill a journal record
                          24CD          .SBTTL  XD_BUF_JNX,   Get a journal record
                          24CD
                          24CD  ;++
                          24CD  ; XD_FILL_JNX - Fill a journal record
                          24CD  ; XD_BUF_JNX - Get a journal record
                          24CD  ;
                          24CD  ; Functional description:
                          24CD  ;
                          24CD  ; This routine finds a journal record and stores the IRP information
                          24CD  ; into the journal record.
                          24CD  ;
                          24CD  ; Inputs:
                          24CD  ;
                          24CD  ;       R0 = Journal record type
                          24CD  ;       R3 = IRP address (XD_FILL_JNX only)
                          24CD  ;       R5 = UCB address
                          24CD  ;
                          24CD  ; Outputs:
                          24CD  ;       R0 = success indicator
                          24CD  ;       R6,R7 are destroyed.
                          24CD  ;
                          24CD  ;--
             00000020     24CD          JNL_REC_SIZ = 32
                          24CD
                          24CD  XD_FILL_JNX::                                   ; Fill the journal record
                 50  DD   24CD          PUSHL   R0                              ; Save record type
                 1D  10   24CF          BSBB    XD_BUF_JNX                      ; Get a journal record
              16 50  E9   24D1          BLBC    R0,90$                          ; Br if error
              50 8ED0     24D4          POPL    R0                              ; Restore record type
              86 50  90   24D7          MOVB    R0,(R6)+                        ; Enter record type
         86  28 A3  B0    24DA          MOVW    IRP$W_CHAN(R3),(R6)+            ; Enter CHAN
         86  20 A3  B0    24DE          MOVW    IRP$W_FUNC(R3),(R6)+            ; Enter FUNC
         86  40 A3  7D    24E2          MOVQ    IRP$Q_STATION(R3),(R6)+         ; Enter STATION
              50 01  9A   24E6          MOVZBL  #1,R0                           ; Return with success
                     05   24E9          RSB
                          24EA
              5E 04  C0   24EA  90$:    ADDL    #4,SP                           ; Pop stack
                     05   24ED          RSB                                     ; Return
                          24EE
                          24EE  XD_BUF_JNX::                                    ; Get a journal record
                          24EE          DSBINT  UCB$B_DIPL(R5)                  ; Sync access to buffers
         57  0090 C5  D0  24F5          MOVL    UCB$L_XD_JNLBUF(R5),R7          ; Get the journal buffer
                 06  13   24FA          BEQL    100$                            ; Br if none
          06 A7  20  B1   24FC          CMPW    #JNL_REC_SIZ,6(R7)              ; Is there enough space?
                 06  1F   2500          BLSSU   200$                            ; Br if no
                          2502  100$:   ENBINT                                  ; Re-enable interrupts
                 50  D4   2505          CLRL    R0                              ; Return error
                     05   2507          RSB
                          2508
          06 A7  20  A2   2508  200$:   SUBW    #JNL_REC_SIZ,6(R7)              ; Acquire space to be used
              56 67  D0   250C          MOVL    (R7),R6                         ; Get output pointer
              67 20  C0   250F          ADDL    #JNL_REC_SIZ,(R7)               ; Adjust output pointer
                          2512          ENBINT                                  ; Re-enable interrupts
              57 56  D0   2515          MOVL    R6,R7                           ; Copy buffer pointer
     86  00000000'GF  7D  2518          MOVQ    G^EXE$GQ_SYSTIME,(R6)+          ; Enter timestamp
              50 01  9A   251F          MOVZBL  #1,R0                           ; Return with success
```

```
05  2522  6380           RSB                               ; Return to caller
    2523  6381
    2523  6383
    2523  6384
    2523  6385
    2523  6386 XD_END:                                     ; Last location in driver
    2523  6387           .END
```

```
$$$                  = 00000020 R      02     CDB_L_BRC                00000038  G
$$$OFF               = 0000005C               CDB_L_BSN                0000003C  G
$$$TYP               = 00000406               CDB_L_CNTS               00000038  G
$$$WID               = 00002000               CDB_L_DEVDEPEND          00000000  G
$$BASE               = 00000001               CDB_L_DMR                00000040  G
$$DISPL              = 00000008               CDB_L_DMS                00000044  G
$$GENSW              = 00000001               CDB_L_PID                00000048  G
$$HIGH               = 00000007               CDB_Q_INFOUT             00000030  G
$$LIMIT              = 00000006               CDB_Q_QUEUES             00000010  G
$$LOW                = 00000001               CDB_Q_RCV_MSG            00000020  G
$$MNSW               = 00000001               CDB_Q_RCV_REQ            00000018  G
$$MXSW               = 00000001               CDB_Q_XMT_PND            00000028  G
$$OP                 = 00000002               CDB_Q_XMT_REQ            00000010  G
ABORTIO                000002D5 R      03     CDB_W_BAB_TIMER          0000005E  G
ABORT_PKT              00001BCD R      03     CDB_W_CHAN               0000004C  G
ACB$L_KAST           = 00000018               CDB_W_DEL_TIMER          00000050  G
ACCESS                 00000AF4 R      03     CDB_W_POLLPRM            00000050  G
ADDRCVLIST             00000EE0 R      03     CDB_W_SEL_TIMER          0000005C  G
ALLOC_P2BUF            00000B0D R      03     CDB_W_SIZE               00000008  G
ALT_START              000004AE R      03     CDB_W_STS                0000000C  G
AT$_UBA              = 00000001               CD_TS_M_BUFQUO         = 00000010  G
BLD_IRP                000023D0 R      03     CD_TS_M_BUFRET         = 00000008  G
BSEL0                  00000000               CD_TS_M_CANCEL         = 00000004  G
BSEL1                  00000001               CD_TS_M_ESTAB          = 00000001  G
BSEL2                  00000002               CD_TS_M_MOP            = 00000002  G
BSEL3                  00000003               CD_TS_M_START          = 00000020  G
BSEL4                  00000004               CD_TS_V_BUFQUO         = 00000004  G
BSEL5                  00000005               CD_TS_V_BUFRET         = 00000003  G
BSEL6                  00000006               CD_TS_V_CANCEL         = 00000002  G
BSEL7                  00000007               CD_TS_V_ESTAB          = 00000000  G
BUG$_NOBUFPCKT         ********   X    03     CD_TS_V_MOP            = 00000001  G
BUG$_UNSUPRTCPU        ********   X    03     CD_TS_V_START          = 00000005  G
CAN$C_CANCEL         = 00000000               CHECK_BUFS               00000ACE R      03
CAN$C_DASSGN         = 00000001               CHECK_P1                 00000AFA R      03
CANCEL                 000022A8 R      03     CHECK_P2                 00000AD0 R      03
CDB_B_MAX_MSGS         0000005B  G            CHECK_PKT                0000239F R      03
CDB_B_MRB              0000004E  G            CHG_CDB                  00002039 R      03
CDB_B_MST              0000004F  G            CHG_UCB                  00001F66 R      03
CDB_B_POL              0000000B  G            CLEAR_CDB                00001E60 R      03
CDB_B_POL_QACT         00000052  G            COM$DELATTNAST           ********   X    03
CDB_B_POL_QDYI         00000056  G            COM$DRVDEALMEM           ********   X    03
CDB_B_POL_QIN          00000054  G            COM$FLUSHATTNS           ********   X    03
CDB_B_POL_RACT         00000053  G            COM$POST                 ********   X    03
CDB_B_POL_RDYI         00000057  G            COM$SETATTNAST           ********   X    03
CDB_B_POL_RIN          00000055  G            CRB$L_INTD             = 00000024
CDB_B_SETPRM           0000004E  G            CRB$L_INTD2            = 00000048
CDB_B_TO_DEAD          0000005A  G            CXB$B_CODE             = 0000000B
CDB_B_TO_DYING         00000059  G            CXB$B_TYPE             = 0000000A
CDB_B_TO_INACT         00000058  G            CXB$C_HEADER           = 00000048
CDB_B_TRB_ADDR         0000000E  G            CXB$C_TRAILER          = 00000004
CDB_B_TYPE             0000000A  G            CXB$L_BL               = 00000004
CDB_B_XMT_CNT          0000000F  G            CXB$L_FL               = 00000000
CDB_C_CNTS           = 00000010               CXB$W_LENGTH           = 0000000C
CDB_C_LENGTH           00000060  G            CXB$W_SIZE             = 00000008
CDB_C_QUEUES         = 00000005               DC$_SCOM               = 00000020
CDB_C_SETPRM         = 00000012               DDB$L_DDT              = 0000000C
CDB_L_AST              00000004  G            DEAD_ERR                 00001799 R      03
```

```
DEF_LINE_PARAM                 00000138 RG     03      FORCE_HALT                       00001718 R      03
DEF_LINE_PARAMSZ             = 0000000F                FORCE_HALT1                      0000171C R      03
DEF_TRIB_PARAM                 00000126 RG     03      FORK_CANCEL                      00001B0F R      03
DEF_TRIB_PARAMSZ             = 00000012                FORK_COUNTER                     00001A1E R      03
DEV$M_AVL                    = 00040000                FORK_HALTT                       00001AF2 R      03
DEV$M_IDV                    = 04000000                FORK_INITC                       0000199B R      03
DEV$M_NET                    = 00002000                FORK_KILLT                       00001B24 R      03
DEV$M_ODV                    = 08000000                FORK_PKT                         00001991 R      03
DEV$M_SHR                    = 00010000                FORK_POLSS                       00001B2A R      03
DEV_ACTIVE                     00000E5B R      03      FORK_PROC                        000018CC R      03
DEV_INACT                      000014A2 R      03      FORK_RECV                        00001ADA R      03
DEV_INACT1                     00001285 R      03      FORK_SSLOT                       00001A4B R      03
DEV_INACT_SP                   000014A0 R      03      FORK_STOPC                       00001A12 R      03
DO_CANCEL                      0000238E R      03      FORK_STOPT                       000019E2 R      03
DPT$C_LENGTH                 = 00000038                FORK_XMIT                        00001A5B R      03
DPT$C_VERSION                = 00000004                FUNCTAB_LEN                    = 00000040
DPT$INITAB                     00000038 R      02      GET_CDB                          00001F50 R      03
DPT$REINITAB                   0000004F R      02      GET_CHAR_BUFS                    0000086A R      03
DPT$TAB                        00000000 R      02      HALT_TRIBS                       00001C1A R      03
DRV_DONE                       00000BC6 R      03      IDB$L_CSR                      = 00000000
DSC$A_POINTER                = 00000004                IDB$L_UCBLST                   = 00000018
DT$_DMP11                    = 00000009                INF_RCV                        = 000000FF
DT$_DMV11                    = 00000017                INF_XMT                        = 0000007F
DYN$C_BUFIO                  = 00000013                INIT_OTHERS                      0000022D R      03
DYN$C_CRB                    = 00000005                INIT_UCB                         000023F1 R      03
DYN$C_CXB                    = 0000001B                INIT_UV1                         0000019D R      03
DYN$C_DDB                    = 00000006                INTERR                           00001514 R      03
DYN$C_DPT                    = 0000001E                INTEXIT                          00001507 R      03
DYN$C_FRK                    = 00000008                IN_CANCEL                        000013FC R      03
DYN$C_IRP                    = 0000000A                IN_CHGC                          00001221 R      03
DYN$C_NET                    = 00000017                IN_CHGT                          00001277 R      03
DYN$C_UCB                    = 00000010                IN_DISP                          000010D7 R      03
EXE$ABORTIO                    ********    X   03      IN_HALTT                         00001453 R      03
EXE$ALLOCBUF                   ********    X   03      IN_INITC                         0000116D R      03
EXE$ALONONPAGED                ********    X   03      IN_INITT                         0000110B R      03
EXE$BUFFRQUOTA                 ********    X   03      IN_KILLT                         000011E7 R      03
EXE$BUFQUOPRC                  ********    X   03      IN_NUMSYNC                       0000125F R      03
EXE$FINISHIO                   ********    X   03      IN_POLSTAT                       000012BF R      03
EXE$FORK                       ********    X   03      IN_RCGERR                        0000147B R      03
EXE$GB_CPUTYPE                 ********    X   03      IN_RCTERR                        000013B7 R      03
EXE$GL_ABSTIM                  ********    X   03      IN_RDGERR                        00001476 R      03
EXE$GL_TENUSEC                 ********    X   03      IN_RDGSS                         0000146B R      03
EXE$GL_UBDELAY                 ********    X   03      IN_RDMODEM                       0000130E R      03
EXE$GQ_SYSTIME                 ********    X   03      IN_RDTERR                        000013B2 R      03
EXE$PROBER_DSC                 ********    X   03      IN_RDTSS                         0000139E R      03
EXE$QIORETURN                  ********    X   03      IN_READG                         00001494 R      03
EXE$READCHK                    ********    X   03      IN_READT                         000013DC R      03
EXE$WRITECHK                   ********    X   03      IN_RECV                          0000137E R      03
FATAL_ERR                      000017AB R      03      IN_STOPC                         000011FC R      03
FILLRCVLIST                    00000ED9 R      03      IN_STOPT                         00001188 R      03
FIND_SLOT                      00001E97 R      03      IN_WTMODEM                       000013EC R      03
FIND_TRIB                      00001EC4 R      03      IN_XMIT                          00001326 R      03
FINISH_RCV_IO                  00001B93 R      03      IO$V_ATTNAST                   = 00000008
FKB$B_FIPL                   = 0000000B                IO$V_CLR_COUNT                 = 0000000A
FKB$B_TYPE                   = 0000000A                IO$V_CTRL                      = 00000009
FKB$C_LENGTH                 = 00000018                IO$V_NOW                       = 00000006
FKB$L_FR3                    = 00000010                IO$V_RD_COUNT                  = 00000008
```

```
IO$V_RD_MEM                 = 00000006
IO$V_RD_MODEM               = 00000007
IO$V_SET_MODEM              = 0000000A
IO$V_SHUTDOWN               = 00000007
IO$V_STARTUP                = 00000006
IO$_READLBLK                = 00000021
IO$_READPBLK                = 0000000C
IO$_READVBLK                = 00000031
IO$_SENSEMODE               = 00000027
IO$_SETCHAR                 = 0000001A
IO$_SETMODE                 = 00000023
IO$_VIRTUAL                 = 0000003F
IO$_WRITELBLK               = 00000020
IO$_WRITEPBLK               = 0000000B
IO$_WRITEVBLK               = 00000030
IOC$ALOUBAMAP                 ********   X    03
IOC$INITIATE                  ********   X    03
IOC$LOADUBAMAPA               ********   X    03
IOC$LOADUBAMAPN               ********   X    03
IOC$MNTVER                    ********   X    03
IOC$RELMAPREG                 ********   X    03
IOC$REQMAPREG                 ********   X    03
IOC$RETURN                    ********   X    03
IOC$WFIKPCH                   ********   X    03
IO_DONE                       00001BD0 R      03
IO_DONE1                      00001BD4 R      03
IRP$B_EFN                   = 00000022
IRP$B_INDEX                   00000041
IRP$B_POLS                    0000003F
IRP$B_PRI                   = 00000023
IRP$B_RMOD                  = 0000000B
IRP$B_TYPE                  = 0000000A
IRP$B_XDFUNC                  00000021
IRP$C_LENGTH                = 000000C4
IRP$L_AST                   = 00000010
IRP$L_ASTPRM                = 00000014
IRP$L_CDB                     00000044
IRP$L_DIAGBUF               = 0000004C
IRP$L_IOQFL                 = 00000000
IRP$L_IOSB                  = 00000024
IRP$L_IOST1                 = 00000038
IRP$L_IOST2                 = 0000003C
IRP$L_MEDIA                 = 00000038
IRP$L_PID                   = 0000000C
IRP$L_SVAPTE                = 0000002C
IRP$L_UCB                   = 0000001C
IRP$L_WIND                  = 00000018
IRP$Q_STATION               = 00000040
IRP$V_DIAGBUF               = 00000007
IRP$V_FUNC                  = 00000001
IRP$W_BCNT                  = 00000032
IRP$W_BOFF                  = 00000030
IRP$W_CHAN                  = 00000028
IRP$W_CSTATUS                 00000042
IRP$W_FUNC                  = 00000020
IRP$W_QUOTA                   00000042
IRP$W_SIZE                  = 00000008
IRP$W_STS                   = 0000002A
JIB$L_BYTCNT                = 00000020
JIB$L_BYTLM                 = 00000024
JNL_CANC                    = 0000000A
JNL_CSRIN                   = 00000008
JNL_CSROUT                  = 00000009
JNL_POST                    = 00000005
JNL_QIO                     = 00000003
JNL_QUE                     = 00000004
JNL_RCV                     = 00000002
JNL_RCVA                    = 00000007
JNL_REC_SIZ                 = 00000020
JNL_XMT                     = 00000001
JNL_XMTA                    = 00000006
JNX$$$                      = 00000001
LINE_CNT_BUFSIZ             = 0000000A
LINE_COUNTER                  00000153 R      03
LINE_PARAM                    00000078 R      03
LINE_PRM_BUFSIZ             = 00000042
LOAD_PORT                     00000FD0 R      03
LOAD_PORT_AVAIL               0000101F R      03
MAINT_ERR                     0000178F R      03
MASKH                       = 00000080
MASKL                       = 00000000
MAX_BUFSIZ                  = 00003FFF
MAX_BUFSIZ_UV1              = 00000400
MAX_DMP_TRIB                = 00000020
MAX_DMV_TRIB                = 00000010
MAX_RCV                     = 00000007
MAX_RCV_UV1                 = 00000004
MAX_W_BUFFER                  0000014F R      03
MAX_W_TRIB                    00000151 R      03
MAX_XMT                     = 00000007
MAX_XMT_TRB                 = 00000004
MAX_XMT_UV1                 = 00000002
MMG$GL_SPTBASE                ********   X    03
NMA$C_CIRPST_ACT            = 00000002
NMA$C_CIRPST_AUT            = 00000001
NMA$C_CIRPST_DED            = 00000005
NMA$C_CIRPST_DIE            = 00000004
NMA$C_CIRPST_INA            = 00000003
NMA$C_CTCIR_BRC             = 000003E8
NMA$C_CTCIR_BSN             = 000003E9
NMA$C_CTCIR_DBR             = 000003F2
NMA$C_CTCIR_DBS             = 000003F3
NMA$C_CTCIR_DEI             = 000003FC
NMA$C_CTCIR_DEO             = 000003FD
NMA$C_CTCIR_LBE             = 00000411
NMA$C_CTCIR_LRT             = 00000407
NMA$C_CTCIR_RBE             = 00000410
NMA$C_CTCIR_RRT             = 00000406
NMA$C_CTCIR_SIE             = 0000041A
NMA$C_CTCIR_SLT             = 0000041B
NMA$C_CTLIN_LPE             = 0000044D
NMA$C_CTLIN_RPE             = 0000044C
NMA$C_DPX_FUL               = 00000000
NMA$C_DPX_HAL               = 00000001
```

```
NMA$C_LINCN_LOO                 = 00000001
NMA$C_LINCN_NOR                 = 00000000
NMA$C_LINPR_CON                 = 00000001
NMA$C_LINPR_DMC                 = 00000004
NMA$C_LINPR_POI                 = 00000000
NMA$C_LINPR_TRI                 = 00000002
NMA$C_PCCI_ACB                  = 0000047E
NMA$C_PCCI_ACI                  = 0000047F
NMA$C_PCCI_BBT                  = 00000475
NMA$C_PCCI_DTH                  = 00000486
NMA$C_PCCI_DYB                  = 00000483
NMA$C_PCCI_DYI                  = 00000484
NMA$C_PCCI_DYT                  = 00000485
NMA$C_PCCI_IAB                  = 00000480
NMA$C_PCCI_IAI                  = 00000481
NMA$C_PCCI_IAT                  = 00000482
NMA$C_PCCI_MRB                  = 00000479
NMA$C_PCCI_MST                  = 00000AFA
NMA$C_PCCI_MTR                  = 0000047A
NMA$C_PCCI_PLS                  = 000003F3
NMA$C_PCCI_POL                  = 000003F2
NMA$C_PCCI_RTT                  = 00000477
NMA$C_PCCI_TRI                  = 00000474
NMA$C_PCCI_TRT                  = 00000476
NMA$C_PCLI_BFN                  = 00000451
NMA$C_PCLI_BUS                  = 00000AF1
NMA$C_PCLI_CON                  = 00000456
NMA$C_PCLI_DDT                  = 0000047F
NMA$C_PCLI_DLT                  = 00000480
NMA$C_PCLI_DUP                  = 00000457
NMA$C_PCLI_NMS                  = 00000AFA
NMA$C_PCLI_PRO                  = 00000458
NMA$C_PCLI_RTT                  = 00000461
NMA$C_PCLI_SLT                  = 0000047E
NMA$C_PCLI_SRT                  = 00000481
NMA$C_STATE_OFF                 = 00000001
NMA$C_STATE_ON                  = 00000000
NMA$M_CNT_COU                   = 00008000
NMA$M_CNT_MAP                   = 00001000
NMA$M_CNT_TYP                   = 00000FFF
NMA$V_CNT_MAP                   = 0000000C
NMA$V_CNT_WID                   = 0000000D
NOOP                              0000177A R      03
OFF_M_VALUE                     = 00003FFF
OFF_S_VALUE                     = 0000000E
OFF_S_WIDTH                     = 00000002
OFF_V_VALUE                     = 00000000
OFF_V_WIDTH                     = 0000000E
OUT_CTL                           000016E4 R      03
OUT_INFO                          0000168F R      03
OUT_RCV                           00001590 R      03
OUT_RCV_ERR                       0000166C R      03
OUT_STATUS                        000017CA R      03
OUT_XMIT                          0000162E R      03
OUT_XMIT_ERR                      0000165B R      03
P1                              = 00000000
P1_LINE_MASK                    = 000000FF
P1_TRIB_MASK                    = 000028FF
P2                              = 00000004
P2B_B_SPARE                       0000000B
P2B_B_TYPE                        0000000A
P2B_C_LENGTH                      0000000C
P2B_L_BUFFER                      00000004
P2B_L_POINTER                     00000000
P2B_T_DATA                        0000000C
P2B_W_SIZE                        00000008
P3                              = 00000008
P4                              = 0000000C
P5                              = 00000010
PCB$L_JIB                       = 00000080
PCB$L_PID                       = 00000060
POKE_USER                         00002270 R      03
POLS$_TBL                         00000147 R      03
POST_IT                           00001C01 R      03
PR$_IPL                         = 00000012
PR$_SID_TYP730                  = 00000003
PR$_SID_TYP750                  = 00000002
PR$_SID_TYP780                  = 00000001
PR$_SID_TYP790                  = 00000004
PR$_SID_TYPUV1                  = 00000007
PRM_M_INVALID                   = 00004000
PRM_M_MAX                       = 00002000
PRM_M_MIN                       = 00001000
PRM_M_TYPE                      = 00000FFF
PRM_S_TYPE                      = 0000000C
PRM_V_INVALID                   = 0000000E
PRM_V_MAX                       = 0000000D
PRM_V_MIN                       = 0000000C
PRM_V_TYPE                      = 00000000
PROCEDURE_ERR                     00001710 R      03
PTE$S_PFN                       = 00000015
PTE$V_PFN                       = 00000000
QUEPKT                            00000ABE R      03
RCV_B_BAHI                        00000010
RCV_B_BLKTYPE                     0000000A
RCV_B_MAPSLOT                     0000000B
RCV_FDT                           00000418 R      03
RCV_L_BACC                        0000000C
RCV_L_LINK                        00000000
RCV_START                         00000458 R      03
RCV_START_ALT                     0000048C R      03
RCV_T_DATA                        00000048
RCV_W_BLKSIZE                     00000008
RDI_INTRPT                        000014CD R      03
RDO_INTRPT                        0000151E R      03
REG_DUMP                          00001E3A R      03
REQ_ABORT                         000014B1 R      03
REQ_COM                           000014BE R      03
REQ_COM1                          0000121E R      03
REQ_COM_ERR                       000014C1 R      03
RETURN_CDB                        00001E50 R      03
RETURN_IRP                        00001E47 R      03
RETURN_P2                         000021E9 R      03
RUNNING_NOTIF                     0000179F R      03
```

```
SCH$GL_PCBVEC                  ********  X   03
SCHED_FORK                     000018A3 R    03
SCHED_FORKC                    000018A1 R    03
SEL0                           00000000
SEL10                          00000008
SEL2                           00000002
SEL4                           00000004
SEL6                           00000006
SENSEMODE_CTRL                 000009D7 R    03
SENSEMODE_FDT                  000008C0 R    03
SENSE_MODEM                    00000AA0 R    03
SETMODE_CTRL                   00000739 R    03
SETMODE_FDT                    0000050A R    03
SET_INDEX                      00001731 R    03
SET_INTIM                      00002418 R    03
SET_OUTTIM                     00002430 R    03
SFORK_EXIT                     0000174B R    03
SHUTDOWN_DEV                   00001D3C R    03
SHUTDOWN_TRIB                  00001C6F R    03
SHUTDOWN_TRIB1                 00001C58 R    03
SIZ...                       = 00000002
SOFT_ERR                       00001750 R    03
SS$_ABORT                    = 0000002C
SS$_ACCVIO                   = 0000000C
SS$_BADPARAM                 = 00000014
SS$_BUFFEROVF                = 00000601
SS$_CTRLERR                  = 00000054
SS$_DEVACTIVE                = 000002C4
SS$_DEVICEFULL               = 00000850
SS$_DEVINACT                 = 000020D4
SS$_DEVOFFLINE               = 00000084
SS$_ENDOFFILE                = 00000870
SS$_INSFARG                  = 00000114
SS$_INSFMAPREG               = 00000344
SS$_INSFMEM                  = 00000124
SS$_NORMAL                   = 00000001
STARTIO                        00000B65 R    03
STARTUP                        00000BC9 R    03
START_CONT                     00000D63 R    03
START_DEV                      00000BB8 R    03
START_ERR                      00001783 R    03
START_ERR1                     00001789 R    03
START_RECEIVE                  00000F3F R    03
START_TIMER                    00002471 R    03
START_TRIB                     00000E42 R    03
START_XMITS                    00001C39 R    03
THRESH                         0000177D R    03
TIMEOUT                        00002484 R    03
TOO_BIG                        000002E2 R    03
TRIB_ACTIVE                    00000E74 R    03
TRIB_CNT_BUFSIZ              = 0000003B
TRIB_COUNTER                   00000157 R    03
TRIB_COUNTER1                  0000015F R    03
TRIB_ERRS                      00000E8D R    03
TRIB_INACT                     000002DB R    03
TRIB_PARAM                     000000CA R    03
TRIB_PRM_BUFSIZ              = 00000066
UCB$B_DEVCLASS               = 00000040
UCB$B_DEVTYPE                = 00000041
UCB$B_DIPL                   = 0000005E
UCB$B_FIPL                   = 0000000B
UCB$B_XD_BFN                   000002AB   G
UCB$B_XD_CON                   000002AA   G
UCB$B_XD_DUP                   000002A9   G
UCB$B_XD_FKB                   000002B6   G
UCB$B_XD_INTIM                 000002A5   G
UCB$B_XD_MODE                  000002A4   G
UCB$B_XD_NMS                   000002A7   G
UCB$B_XD_OUTTIM                000002A6   G
UCB$B_XD_PRO                   000002A8   G
UCB$B_XD_RCV_MAP               00000130   G
UCB$B_XD_RCV_MAX               00000132   G
UCB$B_XD_SETPRM                000002A7   G
UCB$B_XD_TRB_CNT               00000134   G
UCB$B_XD_TRIB_VEC              00000138   G
UCB$B_XD_XMT_MAP               00000131   G
UCB$B_XD_XMT_MAX               00000133   G
UCB$B_XD_XMT_TRB               00000135   G
UCB$C_LENGTH                 = 00000090
UCB$C_XD_LENGTH                000002CE   G
UCB$C_XD_QUEUES              = 00000006
UCB$C_XD_SETPRM              = 0000000F
UCB$C_XD_VECS                = 00000058
UCB$L_CRB                    = 00000024
UCB$L_DEVCHAR                = 00000038
UCB$L_DEVDEPEND              = 00000044
UCB$L_DUETIM                 = 0000006C
UCB$L_FPC                    = 0000000C
UCB$L_SVAPTE                 = 00000078
UCB$L_XD_CDB_VEC               00000218   G
UCB$L_XD_HALT                  0000029C   G
UCB$L_XD_JNLBUF                00000090   G
UCB$L_XD_PID                   000002A0   G
UCB$L_XD_PID_VEC               00000158   G
UCB$L_XD_RCV_MAP               000000C4   G
UCB$L_XD_RCV_PA                000000FC   G
UCB$L_XD_RCV_VA                00000114   G
UCB$L_XD_RUN                   00000298   G
UCB$L_XD_SEL0                  000002C6   G
UCB$L_XD_SEL4                  000002CA   G
UCB$L_XD_UV1BUF                0000012C   G
UCB$L_XD_VECS                  00000138   G
UCB$L_XD_XMT_MAP               000000E0   G
UCB$L_XD_XMT_PA                0000010C   G
UCB$L_XD_XMT_VA                00000124   G
UCB$M_INT                    = 00000002
UCB$M_ONLINE                 = 00000010
UCB$M_POWER                  = 00000020
UCB$M_TIM                    = 00000001
UCB$M_XD_DTRCLR              = 00000010   G
UCB$M_XD_FORK_PEND           = 00000008   G
UCB$M_XD_FORK_XMT            = 00000020   G
UCB$M_XD_INITED              = 00000001   G
UCB$M_XD_INITING             = 00000002   G
```

```
UCB$M_XD_PTP               = 00000004   G
UCB$Q_XD_INFOUT              000000AC   G
UCB$Q_XD_INPUTQ              00000094   G
UCB$Q_XD_POST                000000B4   G
UCB$Q_XD_QUEUES              00000094   G
UCB$Q_XD_RCV_BUF             000000BC   G
UCB$Q_XD_RCV_PND             000000A4   G
UCB$Q_XD_XMT_REQ             0000009C   G
UCB$V_ONLINE               = 00000004
UCB$V_POWER                = 00000005
UCB$V_TIM                  = 00000000
UCB$V_TIMOUT               = 00000006
UCB$V_XD_DTRCLR            = 00000004   G
UCB$V_XD_FORK_PEND         = 00000003   G
UCB$V_XD_FORK_XMT          = 00000005   G
UCB$V_XD_INITED            = 00000000   G
UCB$V_XD_INITING           = 00000001   G
UCB$V_XD_PTP               = 00000002   G
UCB$W_BCNT                 = 0000007E
UCB$W_BOFF                 = 0000007C
UCB$W_DEVBUFSIZ            = 00000042
UCB$W_DEVSTS               = 00000068
UCB$W_ERRCNT               = 00000082
UCB$W_REFC                 = 0000005C
UCB$W_STS                  = 00000064
UCB$W_XD_CHAN_VEC            000001D8   G
UCB$W_XD_DDT                 000002B0   G
UCB$W_XD_DLT                 000002B2   G
UCB$W_XD_POLLPRM             000002AC   G
UCB$W_XD_QUOTA               00000136   G
UCB$W_XD_RTT                 000002B4   G
UCB$W_XD_SLT                 000002AE   G
UCB$W_XD_SRT                 000002AC   G
UNIT_INIT                    0000016F R      03
UNPACK_P2_BUF                00002243 R      03
UPDATE_P2                    00002178 R      03
UV1_BUFFER_AREA            = 00001800
VA$M_BYTE                  = 000001FF
VA$S_VPN                   = 00000015
VA$V_VPN                   = 00000009
VALIDATE_P2                  000020A5 R      03
VALIDATE_P2_CDB              00002089 R      03
VALIDATE_P2_UCB              00002095 R      03
VEC$B_DATAPATH             = 00000013
VEC$B_NUMREG               = 00000012
VEC$L_IDB                  = 00000008
VEC$L_UNITINIT             = 00000018
VEC$W_MAPREG               = 00000010
XD$DDT                       00000000 RG     03
XD_BSEL0_M_IEI             = 00000001
XD_BSEL0_M_IEO             = 00000010
XD_BSEL0_M_RQI             = 00000080
XD_BSEL1_M_LOOPB           = 00000008
XD_BSEL1_M_MAINT           = 00000001
XD_BSEL1_M_MCLR            = 00000040
XD_BSEL1_M_RUN             = 00000080
XD_BSEL2_M_Q22             = 00000008
XD_BSEL2_M_RDI             = 00000010
XD_BSEL2_M_RDO             = 00000080
XD_BSEL2_M_TYPE            = 00000007
XD_BSEL2_V_ERR             = 00000005
XD_BUF_JNX                   000024EE RG     03
XD_CI_V_ESTAB              = 00000001
XD_CI_V_HALT               = 00000005
XD_CI_V_ISTRT              = 00000003
XD_CI_V_KILL               = 00000002
XD_CI_V_MAINT              = 00000004
XD_CI_V_NOOP               = 00000000
XD_CI_V_RDMODEM            = 00000010
XD_CI_V_WTMODEM            = 00000011
XD_DEF_BUFSIZ              = 00000200
XD_END                       00002523 R      03
XD_FC_V_CANCEL             = 00000010
XD_FC_V_CHGC               = 00000004
XD_FC_V_CHGT               = 00000005
XD_FC_V_HALTT              = 0000000E
XD_FC_V_INITC              = 00000001
XD_FC_V_INITT              = 00000000
XD_FC_V_KILLT              = 0000000F
XD_FC_V_NUMSYNC            = 00000013
XD_FC_V_POLSS              = 00000015
XD_FC_V_POLSTAT            = 00000014
XD_FC_V_RCGERR             = 0000000A
XD_FC_V_RCTERR             = 00000009
XD_FC_V_RDGERR             = 00000008
XD_FC_V_RDGSS              = 0000000C
XD_FC_V_RDMODEM            = 00000006
XD_FC_V_RDTERR             = 00000007
XD_FC_V_RDTSS              = 0000000B
XD_FC_V_RECV               = 00000012
XD_FC_V_STOPC              = 00000003
XD_FC_V_STOPT              = 00000002
XD_FC_V_WTMODEM            = 0000000D
XD_FC_V_XMIT               = 00000011
XD_FILL_JNX                  000024CD RG     03
XD_FUNCTABLE                 00000038 R      03
XD_I_V_CONTROL             = 00000001
XD_I_V_MODE_DEF            = 00000002
XD_I_V_RCV                 = 00000000
XD_I_V_XMIT                = 00000004
XD_SEL6_M_ECOM             = 00000100
XD_SEL6_M_LPOL             = 00002000
XD_SEL6_M_RCTS             = 00000040
XD_SEL6_M_RTSS             = 00000020
XD_SEL6_M_UPOL             = 00001000
XD_SEL6_M_WTSS             = 00000080
XLATE                        00001EEE R      03
XLATE_ALT                    00001EF2 R      03
XLATE_PID_CHAN               00001F11 R      03
XM$M_CHR_CTRL              = 00000040
XM$M_CHR_DMC               = 00000020
XM$M_CHR_TRIB              = 00000080
XM$M_STS_ACTIVE            = 00000800
XM$M_STS_RUNNING           = 00002000
```

```
XM$V_CHR_CTRL                    = 00000006
XM$V_CHR_DMC                     = 00000005
XM$V_CHR_HDPLX                   = 00000002
XM$V_CHR_LOOPB                   = 00000001
XM$V_CHR_MOP                     = 00000000
XM$V_CHR_TRIB                    = 00000007
XM$V_ERR_FATAL                   = 00000010
XM$V_ERR_MAINT                   = 00000013
XM$V_ERR_START                   = 00000017
XM$V_ERR_THRESH                  = 00000015
XM$V_ERR_TRIB                    = 00000016
XM$V_STS_ACTIVE                  = 0000000B
XM$V_STS_BUFFAIL                 = 0000000C
XM$V_STS_DISC                    = 0000000E
XM$V_STS_RUNNING                 = 0000000D
XM$V_STS_TIMO                    = 00000009
XMT_ALT_START                      000002FB R     03
XMT_FDT                            0000024A R     03
XMT_OTHERS                         00000388 R     03
XMT_START                          000002E7 R     03
XMT_UV1                            0000035A 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
$ABS$              000002CE (  718.)   01 (  1.)  NOPIC  USR  CON  ABS  LCL NOSHR   EXE   RD    WRT NOVEC BYTE
$$$105_PROLOGUE    00000069 (  105.)   02 (  2.)  NOPIC  USR  CON  REL  LCL NOSHR   EXE   RD    WRT NOVEC BYTE
$$$115_DRIVER      00002523 ( 9507.)   03 (  3.)  NOPIC  USR  CON  REL  LCL NOSHR   EXE   RD    WRT NOVEC LONG

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

Phase                     Page faults   CPU Time      Elapsed Time
-----                     -----------   --------      ------------
Initialization                     36   00:00:00.05   00:00:01.58
Command processing                131   00:00:00.50   00:00:03.36
Pass 1                           1655   00:00:47.63   00:03:09.77
Symbol table sort                   0   00:00:04.50   00:00:18.32
Pass 2                            470   00:00:12.28   00:00:47.09
Symbol table output                 2   00:00:00.39   00:00:01.75
Psect synopsis output               2   00:00:00.02   00:00:00.02
Cross-reference output              0   00:00:00.00   00:00:00.00
Assembler run totals             2299   00:01:05.37   00:04:21.90

The working set limit was 3450 pages.
437790 bytes (856 pages) of virtual memory were used to buffer the intermediate code.
There were 230 pages of symbol table space allocated to hold 3844 non-local and 564 local symbols.
6387 source lines were read in Pass 1, producing 50 object records in Pass 2.
83 pages of virtual memory were used to define 74 macros.
```

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

Macro library name                          Macros defined
------------------                          --------------
_$255$DUA28:[SHRLIB]NMALIBRY.MLB;1                       1
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                          35
_$255$DUA28:[SYSLIB]STARLET.MLB;2                       12
TOTALS (all libraries)                                  48

3853 GETS were required to define 48 macros.

There were no errors, warnings or information messages.

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

0118 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

XDDRIVER
LIS

XDDRIVER
LIS