C 8

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

```
                                        F  1
**FILE**ID**XMDRIVER




XX      XX  MM      MM  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
XX      XX  MM      MM  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
XX      XX  MMMM  MMMM  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
XX      XX  MMMM  MMMM  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
  XX  XX    MM  MM  MM  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
  XX  XX    MM  MM  MM  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
    XX      MM      MM  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
    XX      MM      MM  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
  XX  XX    MM      MM  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
  XX  XX    MM      MM  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
XX      XX  MM      MM  DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
XX      XX  MM      MM  DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
XX      XX  MM      MM  DDDDDDDD    RR      RR  IIIIII          VV      EEEEEEEEEE  RR      RR      ....
XX      XX  MM      MM  DDDDDDDD    RR      RR  IIIIII          VV      EEEEEEEEEE  RR      RR      ....

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

Table of contents

```
0000        .TITLE  XMDRIVER - VAX/VMS DMC11/DMR11 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 DMC11/DMR11 Device driver
0000  ;
0000  ; ABSTRACT:
0000  ;
0000  ;       This module contains the DMC11/DMR11 driver FDT routines,
0000  ;       interrupt dispatcher, interrupt service and fork routines.
0000  ;
0000  ; AUTHOR:
0000  ;
0000  ;       R.HEINEN 24-AUG-77
0000  ;
0000  ; MODIFICATION HISTORY:
0000  ;
0000  ;       V03-023 RNG0023         Rod N. Gamache          17-May-1984
0000  ;               Set the DEV$M_AVL bit to make XM units available.
0000  ;
0000  ;       V03-022 RNG0022         Rod N. Gamache          29-Feb-1984
0000  ;               Fix problem with allocation of map registers which causes
0000  ;               too many map registers to be allocated.
0000  ;
0000  ;       V03-021 RNG0021         Rod N. Gamache          29-Oct-1983
0000  ;               Fix broken register useage caused by use of TIMEDWAIT macro.
0000  ;
0000  ;       V03-020 RNG0020         Rod N. Gamache          27-Jul-1983
0000  ;               Changed WAIT10 macro to use system TIMEDWAIT macro.
0000  ;               Change all NOP wait loops to use TIMEDWAIT macro.
0000  ;               Don't do BUG_CHECK if input request was processed by
0000  ;               Interrupt Service Routine.
0000  ;
```

```
0000  ;        V03-019 ROW0169          Ralph O. Weber              3-MAR-1983
0000  ;                Add $IPLDEF.
0000  ;
0000  ;        V03-018 RNG0012          Rod Gamache     28-Jan-1983
0000  ;                Add code to hang up modems on LINE DOWN requests.
0000  ;
0000  ;        V03-017 RNG0011          Rod Gamache     17-Dec-1982
0000  ;                Speed up the startup time for devices that don't run
0000  ;                micro-diagnostics.
0000  ;
0000  ;        V03-016 RNG0010          Rod Gamache     04-Nov-1982
0000  ;                Setup timeout routine offset for new fork process
0000  ;                added to transmit process routine.
0000  ;
0000  ;        V03-015 RNG0009          Rod Gamache     07-Sep-1982
0000  ;                Fix cancel routine to only abort user's I/O on $CANCEL
0000  ;                request and not to shutdown device or abort other users'
0000  ;                I/O. Add another fork block to UCB to allow a fork on tranmit
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  ;        V03-013 RNG0008          Rod Gamache     01-Sep-1982
0000  ;                Reduce startup time required in previous enhancement.
0000  ;                Fix startup problem when running in LOOPBACK mode - fixes
0000  ;                problem found by UETP.
0000  ;
0000  ;        V03-012 RAN0001          R. Newell       08-Jul-1982
0000  ;                Add code to determine whether a DMC has the high-speed or
0000  ;                low-speed microcode chip set and what the mode and interface
0000  ;                switches are set to on a DMR.
0000  ;
0000  ;
0000  ; PREVIOUS MODIFICATIONS:
0000  ;
0000  ;                Al Eldridge, Scott Davis, Len Kawell, Rod Gamache   1979-1982
0000  ;--
```

```
0000    ;
0000    ; System definitions
0000    ;
0000            $ACBDEF                                         ; AST control block
0000            $CANDEF                                         ; Define Cancel reason codes
0000            $CRBDEF                                         ; Controller request block
0000            $CXBDEF                                         ; Define CXB block
0000            $DCDEF                                          ; Device types
0000            $DDBDEF                                         ; Device data block
0000            $DPTDEF                                         ; Driver prologue table
0000            $DYNDEF                                         ; Dynamic data structure types
0000            $FKBDEF                                         ; Fork block definitions
0000            $IDBDEF                                         ; Interrupt data block
0000            $IODEF                                          ; I/O functions
0000            $IPLDEF                                         ; IPL symbolic definitions
0000            $IRPDEF                                         ; I/O packets
0000            $JIBDEF                                         ; Job information block
0000            $NMADEF                                         ; Network management codes
0000            $PCBDEF                                         ; Process control block
0000            $PRDEF                                          ; Processor registers
0000            $SSDEF                                          ; System status codes
0000            $TQEDEF                                         ; Timer Queue Element
0000            $UBADEF                                         ; UNIBUS adapter registers
0000            $UCBDEF                                         ; Unit control block
0000            $VADEF                                          ; Virtual address fields
0000            $VECDEF                                         ; Interrupt vector
0000            $XMDEF                                          ; XMDRIVER symbols
0000
0000    ;
0000    ; Local macros
0000    ;
0000            .MACRO  SETBIT  POS,BAS,?L              ; Set a single bit
0000                    BBSS    POS,BAS,L
0000            L:
0000            .ENDM   SETBIT
0000
0000            ;*******
0000
0000            .MACRO  CLRBIT  POS,BAS,?L              ; Clear a single bit
0000                    BBCC    POS,BAS,L
0000            L:
0000            .ENDM   CLRBIT
0000
0000            ;*******
0000
0000            .MACRO  ADDLC   COUNT,COUNTER,?L; Add to counter
0000                    ADDL    COUNT,COUNTER   ; Increment
0000                    BCC     L               ; Br if no carry
0000                    MNEGL   #1,COUNTER      ; Set to maximum value
0000            L:
0000            .ENDM   ADDLC
0000
0000            ;*******
0000
0000            .MACRO  WAIT10 WTIME,?L1
0000
0000            TIMEDWAIT       TIME=WTIME,-
```

```
          0000                                INS1=<BITB      S^#0,S^#0>,-
          0000                                INS2=<BNEQ      L1>,-
          0000                                DONELBL=L1
          0000
          0000            .ENDM   WAIT10
          0000
          0000            ;*******
          0000
          0000            .MACRO  COUNTER TYPE,BITMAP=NO,WIDTH=8,-
          0000                            BASEOFF1=0,UCBOFF1=DEVCNT,BASEOFF2=0,UCBOFF2=DEVCNT
          0000            $$$TYP = NMA$C_CTCIR_'TYPE' & NMA$M_CNT_TYP
          0000            .IIF IDN <BITMAP><YES>, $$$TYP = $$$TYP!<NMA$M_CNT_MAP>
          0000            $$$WID = 0                          ; Set reserved mask field
          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            .IF  NE BASEOFF1
          0000                              .BYTE BASEOFF1,UCB$B_XM_'UCBOFF1'-UCB$B_XM_DEVCNT
          0000            .IIF NE BASEOFF2, .BYTE BASEOFF2,UCB$B_XM_'UCBOFF2'-UCB$B_XM_DEVCNT
          0000                              .BYTE 0
          0000            .ENDC
          0000            CNT_BUFSIZ = CNT_BUFSIZ + 2 + <WIDTH/8>
          0000            .IIF IDN <BITMAP><YES>, CNT_BUFSIZ = CNT_BUFSIZ + 2
          0000            .ENDM   COUNTER
          0000
          0000    ;
          0000    ; Local symbol definitions
          0000    ;
          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
          0000
00000100  0000    BASETAB_SIZE      = 256                         ; Size of base table
00003FFF  0000    MAX_C_BUFSIZE     = 16383                       ; Maximum transfer size
00000007  0000    MAX_RCV           = 7                           ; Maximum number outstanding receives
00000007  0000    MAX_XMT           = 7                           ; Maximum number outstanding transmits
00000003  0000    DMC_DMR           = 3                           ; DMC or DMR test value
000F4240  0000    SHUT_TIME         = 1000*1000                   ; Shutdown delay time (100 ms)
00002296  0000    UINST_CNF         = ^021226                     ; Microinstruction to get config
0000814D  0000    UINST_RROM        = ^0100515                    ; Microinstruction to read DMC ROM
00000390  0000    LS_UCODE          = ^01620                      ; Contents of addr 0115 in l.s. u-code
0000A40B  0000    DROP_DTR          = ^0122013                    ; Drop DTR on modem
00000082  0000    EXECUTE_UC        = ^0202                       ; Execute in DMC PORT
          0000    ;
          0000    ; XMDRIVER UCB extensions
          0000    ;
          0000            $DEFINI
00000090  0000    . = UCB$C_LENGTH
          0090
          0090    $DEF      UCB$Q_XM_QUEUES                     ; Message and I/O request queue heads
          0090    $DEF      UCB$Q_XM_XMT_REQ .BLKQ  1           ; Transmit I/O requests awaiting start
          0098   $DEF      UCB$Q_XM_RCV_REQ .BLKQ   1          ; Receive I/O requests awaiting message
          00A0   $DEF      UCB$Q_XM_PORT    .BLKQ   1          ; Transmits/receives awaiting the port
          00A8   $DEF      UCB$Q_XM_XMT_PND .BLKQ   1          ; Transmit I/Os given to device
          00B0   $DEF      UCB$Q_XM_RCV_PND .BLKQ   1          ; Receive buffers given to device
          00B8   $DEF      UCB$Q_XM_POST    .BLKQ   1          ; Transmits/receives awaiting posting
          00C0   $DEF      UCB$Q_XM_RCV_BUF .BLKQ   1          ; Free receive buffers
          00C8   $DEF      UCB$Q_XM_RCV_MSG .BLKQ   1          ; Receive buffers containing messages
00000008  00D0   UCB$C_XM_QUEUES = <.-UCB$Q_XM_QUEUES>/8 ; Number of queue heads
          00D0
          00D0   $DEF      UCB$L_XM_RCV_MAP .BLKL   MAX_RCV    ; Receive mapping vector
          00EC   $DEF      UCB$L_XM_XMT_MAP .BLKL   MAX_XMT    ; Transmit mapping vector
          0108   $DEF      UCB$B_XM_RCV_MAP .BLKB   1          ; Receive mapping in use flags
          0109   $DEF      UCB$B_XM_XMT_MAP .BLKB   1          ; Transmit mapping in use flags
          010A   $DEF      UCB$B_XM_RCV_MAX .BLKB   1          ; Maximum concurrent receives
          010B   $DEF      UCB$B_XM_XMT_MAX .BLKB   1          ; Maximum concurrent transmits
          010C
          010C   $DEF      UCB$W_XM_QUOTA   .BLKW   1          ; Starter's byte quota deducted
00000110  010E                              .BLKW   1          ; (spare for alignment)
          0110   $DEF      UCB$L_XM_PID     .BLKL   1          ; Starter's process ID
          0114   $DEF      UCB$L_XM_AST     .BLKL   1          ; Attention AST list
          0118   $DEF      UCB$L_XM_BASETAB .BLKL   1          ; Base table address
          011C   $DEF      UCB$L_XM_BASEMAP .BLKL   1          ; Base table map regiser number/count
          0120
          0120   $DEF      UCB$L_XM_DRVCNT                     ; Driver counters
          0120   $DEF      UCB$L_RCVBYTCNT  .BLKL   1          ; Receive byte count
          0124   $DEF      UCB$L_XMTBYTCNT  .BLKL   1          ; Transmit byte count
          0128   $DEF      UCB$L_RCVMSGCNT  .BLKL   1          ; Receive message count
          012C   $DEF      UCB$L_XMTMSGCNT  .BLKL   1          ; Transmit message count
00000004  0130   UCB$C_XM_DRVCNT = <.-UCB$L_XM_DRVCNT>/4
          0130
          0130   $DEF      UCB$B_XM_DEVCNT                     ; Device counters
          0130   $DEF      UCB$B_XM_NBFR    .BLKB   1          ; NAKs rcvd - no buffer (DMR11)
          0131   $DEF      UCB$B_XM_HCER    .BLKB   1          ; NAKs rcvd - header BCC error (DMR11)
          0132   $DEF      UCB$B_XM_DCER    .BLKB   1          ; NAKs rcvd - data BCC error
          0133   $DEF      UCB$B_XM_NBFS    .BLKB   1          ; NAKs sent - no buffer
          0134   $DEF      UCB$B_XM_HCES    .BLKB   1          ; NAKs sent - header BCC error
          0135   $DEF      UCB$B_XM_DCES    .BLKB   1          ; NAKs sent - data BCC error
          0136   $DEF      UCB$B_XM_REPS    .BLKB   1          ; REPs sent
          0137   $DEF      UCB$B_XM_REPR    .BLKB   1          ; REPs rcvd
00000008  0138   UCB$C_XM_DEVCNT = .-UCB$B_XM_DEVCNT
          0138
          0138   $DEF      UCB$B_XM_FKB     .BLKB   FKB$C_LENGTH ; Fork process fork block
          0150   $DEF      UCB$W_XM_MODSIG  .BLKW   1            ; Modem signals
          0152   $DEF      UCB$C_XM_LENGTH                       ; Size of XMDRIVER UCB
          0152
00000148  0152   . = UCB$B_XM_FKB+FKB$L_FR3
          0148
          0148   $DEF      UCB$L_XM_LSTPRT  .BLKL   1          ; Last port value
          014C   $DEF      UCB$L_XM_LSTCSR  .BLKL   1          ; Last CSR value
          0150
          0150             $VIELD  UCB,0,<-                    ; XMDRIVER UCB$W_DEVSTS bits
          0150                     <XM_FORK_XMT,,M>,-          ; Transmit fork block in use
          0150                     <,2>,-                      ; reserved
          0150                     <XM_INITED,,M>,-            ; Unit initialized
          0150                     <,7>,-                      ; reserved
          0150                     <XM_NOTIF,,M>,-             ; Mailbox notified
          0150                     <XM_LOSTERR,,M>,-           ; Unreported fatal error
          0150                  <XM_FORK_PEND,,M>,-       ; Fork process scheduling in progress
          0150          >
          0150
          0150          $VIELD  MOD,0,-                   ; XMDRIVER UCB$L_DEVDEPEND+3 bits
          0150          <-                                ;         HARDWARE MODE BITS (byte)
          0150                  <XM_HIGH,,M>,-            ; High speed indicator (DMC/DMR)
          0150                  <XM_DMC,,M>,-             ; DMC compatible mode (DMR only)
          0150                  <XM_INTMOD,,M>,-          ; Integral modem (DMR only)
          0150                  <XM_V.35,,M>,-            ; V.35 (DMR only)
          0150                  <XM_RS232,,M>,-           ; RS-232C mode (DMR only)
          0150                  <XM_RS422,,M>,-           ; RS-422 mode (DMR only)
          0150                  <,15,-                    ; RESERVED
          0150                  <XM_BSEL1,,M>,-           ; Indicates that BSEL1 is not locked out
          0150          >                                 ; ..if set, indicates 1st 2 bits are ok
          0150
          0150  ;
          0150  ; DMC11/DMR11 device register definitions
          0150  ;
00000000  0150  . = 0
          0000  $DEF    XM_I_CSR        .BLKW   1         ; Input CSR (SEL 0)
          0002          _VIELD  XM_I,0,<-                 ;
          0002                  <TYPE,2,M>,-              ; Request type
          0002                  <RCV,,M>,-                ; Receive buffer flag
          0002                  <,2>,-                    ; reserved
          0002                  <RQI,,M>,-                ; Request port
          0002                  <IEI,,M>,-                ; Port available interrupt enable
          0002                  <RDI,,M>,-                ; Port available
          0002                  <STEPUP,,M>,-             ; Step microprocessor
          0002                  <ROMI,,M>,-               ; ROM IN
          0002                  <ROMO,,M>,-               ; ROM OUT
          0002                  <LOOPB,,M>,-              ; Internal loopback
          0002                  <,2>,-                    ; Maintenance bits
          0002                  <MCLR,,M>,-               ; Master clear device
          0002                  <RUN,,M>,-                ; Run
          0002          >
          0002  $DEF    XM_O_CSR        .BLKW   1         ; Output CSR (SEL 2)
          0004          _VIELD  XM_O,0,<-                 ;
          0004                  <TYPE,2,M>,-              ; Output type
          0004                  <RCV,,M>,-                ; Receive buffer flag
          0004                  <,3>,-                    ; reserved
          0004                  <IEO,,M>,-                ; Output interrupt enable
          0004                  <RDO,,M>,-                ; Output ready
          0004          >
          0004  $DEF    XM_PORT         .BLKW   1         ; Data port register (SEL 4)
          0006  $DEF    XM_UCODE        .BLKW   1         ; Data/error port register (SEL 6)
          0008          _VIELD  XM_E,0,<-                 ;
          0008                  <DCHK,,M>,-               ; Data check
          0008                  <TIMO,,M>,-               ; Timeout
          0008                  <NOBUF,,M>,-              ; Data overrun
          0008                  <MOP,,M>,-                ; MOP message received
          0008                  <LOST,,M>,-               ; Lost data
          0008                  <TRNER,,M>,-              ; Transfer error
          0008                  <LINEDWN,,M>,-            ; Line down
          0008                  <START,,M>,-              ; Start received
          0008                  <NONEXMEM,,M>,-           ; Non-existent memory
          0008                  <PROCERR,,M>,-            ; Procedure error
          0008                  <POWER,,M>,-              ; System powerfailure (set by driver)
          0008                   <TIMEOUT,,M>,-             ; Transmit timeout (set by driver)
          0008                   >
          0008       ;
          0008       ; Receive buffer definition
          0008       ;
          0008                   $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       $DEF        RCV_L_BACC      .BLKL   1      ; Buffer address / character count
00000048  0010                   .IIF LT .-CXB$C_HEADER, .=CXB$C_HEADER ; (allow for CXB header)
          0048       $DEF        RCV_T_DATA                     ; Receive data
          0048
          0048                   $DEFEND RCV
          0008
          0008       ;
          0008       ; Basetable block definition
          0008       ;
          0008                   $DEFINI BAS
00000000  0000       .=0
          0000       $DEF        BAS_Q_SPARE     .BLKQ   1      ; Spare quadword
          0008       $DEF        BAS_W_SIZE      .BLKW   1      ; Block size
          000A       $DEF        BAS_B_TYPE      .BLKB   1      ; Block type
          000B       $DEF        BAS_B_SPARE     .BLKB   1      ; Spare byte
          000C       $DEF        BAS_T_DATA                     ; Start of real basetable
          000C       $DEF        BAS_C_HEADER                   ; Size of base table header
          000C
          000C                   $DEFEND BAS
```

```
0008            .SBTTL  Standard Driver Tables
0008    ;
0008    ; Driver Prologue Table
0008    ;
0008            DPTAB   -                               ;
0008                    END=XM_END,-                    ; End of driver
0008                    ADAPTER=UBA,-                   ; UNIBUS device
0008                    UCBSIZE=UCB$C_XM_LENGTH,-; UCB size
0008                    NAME=XMDRIVER                   ; Driver name
0038
0038            DPT_STORE INIT                          ; Initialization data
0038            DPT_STORE UCB,UCB$B_FIPL,B,8            ; Fork IPL
003C            DPT_STORE UCB,UCB$B_DIPL,B,21           ; Device IPL
0040            DPT_STORE UCB,UCB$L_DEVCHAR,L,<-; Device characteristics
0040                    DEV$M_NET!DEV$M_AVL!DEV$M_IDV!DEV$M_ODV>
0047            DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_SCOM ; Device class
004B            DPT_STORE UCB,UCB$B_DEVTYPE,B,DT$_DMC11 ; Assume a DMC11
004F            DPT_STORE UCB,UCB$W_DEVBUFSIZ,W,256 ; Default buffer size
0054
0054            DPT_STORE REINIT                        ; Initialization data also for reload
0054            DPT_STORE DDB,DDB$L_DDT,D,XM$DDT; Driver dispatch table
0059            DPT_STORE CRB,CRB$L_INTD+4,D,PORT_INTR ; Port interrupt service routine
005E            DPT_STORE CRB,CRB$L_INTD+VEC$L_UNITINIT,D,UNIT_INIT ; Unit init routine
0063            DPT_STORE CRB,CRB$L_INTD2+4,D,CONTROL_INTR ; Control interrupt service
0068            DPT_STORE END
0008
0008    ;
0008    ; Driver Dispatch Table
0008    ;
0008            DDTAB   DEVNAM=XM,-                     ; Device name
0008                    START=STARTIO,-                 ; Start I/O routine
0008                    FUNCTB=FUNCTABLE,-              ; Function decision table
0008                    CANCEL=CANCEL,-                 ; Cancel I/O routine
0008                    REGDMP=REGDUMP,-                ; Register dump routine
0008                    DIAGBF=<32+36>,-                ; Diagnostic buffer size
0008                    ALTSTART=ALTFDT                 ; Alternate transmit/receive routine
0038    ;
0038    ; Function Decision Table
0038    ;
0038    FUNCTABLE:
0038            FUNCTAB,-                               ; Legal functions
0038                    <WRITEVBLK,WRITELBLK,WRITEPBLK,-; Transmit functions
0038                     READVBLK,READLBLK,READPBLK,-   ; Receive functions
0038                     SETMODE,SETCHAR,-              ; Set mode functions
0038                     SENSEMODE,SENSECHAR>           ; Read and/or clear counters
0040            FUNCTAB,-                               ; Buffered I/O functions
0040                    <READLBLK,READPBLK,READVBLK,-   ;
0040                     SETMODE,SETCHAR>               ;
0048            FUNCTAB XMTFDT,-                        ; Transmit function dispatcher
0048                    <WRITELBLK,WRITEPBLK,WRITEVBLK> ;
0054            FUNCTAB RCVFDT,-                        ; Receive function dispatcher
0054                    <READLBLK,READPBLK,READVBLK>    ;
0060            FUNCTAB SETMODEFDT,-                    ; Set mode function dispatcher
0060                    <SETMODE,SETCHAR>               ;
006C            FUNCTAB SENSEMODEFDT,-                  ; Sense mode function dispatcher
006C                    <SENSEMODE,SENSECHAR>           ;
0078
```

```
                0078    411 ;
                0078    412 ; Counter ID and format table
                0078    413 ;
                0078    414 ;        Note: the order of this table is related to the UCB counters.
                0078    415 ;
       00000000 0078    416 CNT_BUFSIZ = 0
                0078    417 CNTTAB:                                             ; Counters maintained by driver
                0078    418         COUNTER BRC,NO, 32                          ;  Bytes received
                007A    419         COUNTER BSN,NO, 32                          ;  Bytes sent
                007C    420         COUNTER DBR,NO, 32                          ;  Data blocks received
                007E    421         COUNTER DBS,NO, 32                          ;  Data blocks sent
                0080    422                                                     ; Counters maintained by device
                0080    423         COUNTER DEO,YES,8, 4,HCER,5,DCER            ;  Data errors outbound
                0087    424         COUNTER DEI,YES,8, 7,HCES,8,DCES            ;  Data errors inbound
                008E    425         COUNTER LBE,YES,8, 6,NBFS                   ;  Local buffer errors
                0093    426         COUNTER RBE,YES,8, 3,NBFR                   ;  Remote buffer errors
                0098    427         COUNTER LRT,NO, 8, 9,REPS                   ;  Local reply timeouts
                009D    428         COUNTER RRT,NO, 8,10,REPR                   ;  Remote reply timeouts
           0000 00A2    429         .WORD   0
                00A4    430
```

```
                        00A4            .SBTTL  UNIT_INIT - Initialize the device unit
                        00A4     ;++
                        00A4     ; UNIT_INIT - Initialize the device unit
                        00A4     ;
                        00A4     ; FUNCTIONAL DESCRIPTION:
                        00A4     ;
                        00A4     ; This routine is called when the driver is loaded and during powerfailure
                        00A4     ; recovery.  It sets the unit status to ONLINE.  Also, if called during
                        00A4     ; powerfail recovery, it shuts down the device.
                        00A4     ;
                        00A4     ; INPUTS:
                        00A4     ;
                        00A4     ;       R3 = CSR address
                        00A4     ;       R4 = CSR address
                        00A4     ;       R5 = UCB address
                        00A4     ;
                        00A4     ; OUTPUTS:
                        00A4     ;
                        00A4     ;       R0,R1,R2,R3,R4,R5 preserved
                        00A4     ;--
                        00A4     UNIT_INIT:                                              ; Initialize the unit
       64 A5    10  A8  00A4            BISW    #UCB$M_ONLINE,UCB$W_STS(R5)             ; Set software status ONLINE
 0143 C5    0B A5   90  00A8            MOVB    UCB$B_FIPL(R5),UCB$B_XM_FKB+-           ; Set FORK BLOCK FORK IPL
                        00AE                    FKB$B_FIPL(R5)                          ;
   63     4000 8F   B0  00AE            MOVW    #XM_I_M_MCLR,(R3)                       ; Master clear the controller
            0E62    30  00B3            BSBW    DISABLE_MODEM                           ; Disable the modem
  14 64 A5      05  E1  00B6            BBC     #UCB$V_POWER,UCB$W_STS(R5),10$         ; Br if not powerfail recovery
                0B  E1  00BB            BBC     #XM$V_STS_ACTIVE,-                      ; Br if not previously active
         0F 44 A5       00BD                    UCB$L_DEVDEPEND(R5),10$                 ;
                1F  BB  00C0            PUSHR   #^M<R0,R1,R2,R3,R4>                     ; Save all registers
   53    01     1A  78  00C2            ASHL    #XM_E_V_POWER+16,#1,R3                  ; Indicate powerfail
   54    01     10  78  00C6            ASHL    #XM_O_V_TYPE+16,#1,R4                   ; Indicate error
              0ABE  30  00CA            BSBW    SCHED_FORK                              ; Schedule fork process
                1F  BA  00CD            POPR    #^M<R0,R1,R2,R3,R4>                     ; Restore registers
                    05  00CF     10$:   RSB                                             ;
                        00D0
                        00D0          .SBTTL XMTFDT - Transmit I/O FDT routine
                        00D0   ;++
                        00D0   ; XMTFDT - Transmit I/O FDT routine
                        00D0   ;
                        00D0   ; Functional description:
                        00D0   ;
                        00D0   ; This routine is called by the SYS$QIO service to dispatch a WRITE I/O
                        00D0   ; request.
                        00D0   ;
                        00D0   ; The QIO parameters for WRITES are:
                        00D0   ;
                        00D0   ;       P1 = address of the buffer
                        00D0   ;       P2 = size of the buffer
                        00D0   ;       P3-P6 = (unused)
                        00D0   ;
                        00D0   ; The buffer is validated for access and locked into the caller's working
                        00D0   ; set, a transmit UNIBUS map register set is allocated, the buffer
                        00D0   ; is mapped, the device input port is requested, the buffer address and size
                        00D0   ; are passed to the device, and finally the I/O request is queued to await
                        00D0   ; the completion of the transmit by the device.
                        00D0   ;
                        00D0   ; If no transmit slot or mapping registers are available, put the I/O request
                        00D0   ; into a wait queue.  When a transmit in progress completes, it will restart
                        00D0   ; the waiting request.  Note - this design depends on having at least one set
                        00D0   ; of map registers pre-allocated.
                        00D0   ;
                        00D0   ; For requests specifing IO$M_ENABLMBX the attention mailbox is enabled.
                        00D0   ;
                        00D0   ; Inputs:
                        00D0   ;
                        00D0   ;       R0-R2 = scratch registers
                        00D0   ;       R3 = I/O packet address
                        00D0   ;       R4 = PCB address
                        00D0   ;       R5 = UCB address
                        00D0   ;       R6 = CCB address
                        00D0   ;       R7 = bit number of the I/O function code
                        00D0   ;       R8 = address of the FDT table entry for this routine
                        00D0   ;       R9-R11 = scratch registers
                        00D0   ;       AP = address of first QIO parameter
                        00D0   ;
                        00D0   ; Outputs:
                        00D0   ;
                        00D0   ;       R0 = status of transmit request initiation
                        00D0   ;
                        00D0   ;       R3,R5 are preserved.
                        00D0   ;--
                        00D0   XMTFDT:                                          ; Transmit FDT routine
        50    14    3C  00D0          MOVZWL  S^#SS$_BADPARAM,R0        ; Assume bad buffer parameters
    51    04 AC    3C  00D3          MOVZWL  P2(AP),R1                 ; Get buffer size
              2F    13  00D7          BEQL    ABORTIO                   ; Br if zero - abort I/O
  3FFF 8F    51    B1  00D9          CMPW    R1,#MAX_C_BUFSIZE         ; Is buffer too big?
              28    1A  00DE          BGTRU   ABORTIO                   ; Br if yes - abort I/O
        50    6C    D0  00E0          MOVL    P1(AP),R0                 ; Get user buffer virtual address
   00000000'GF    16  00E3          JSB     G^EXE$WRITELOCK           ; Check buffer access and lock down
                        00E9                                            ; (no return means no access)
                        00E9          SETIPL  UCB$B_FIPL(R5)            ; Synch access to UCB
  50    0084 8F    3C  00ED          MOVZWL  #SS$_DEVOFFLINE,R0        ; Assume device is not active
```

```
               0B   E1 00F2             BBC     #XM$V_STS_ACTIVE,-      ; Br if not active - abort I/O
          11 44 A5     00F4                     UCB$L_DEVDEPEND(R5),ABORTIO
               07   E1 00F7             BBC     #IO$V_ENABLMBX,-        ; Br if mailbox not to be enabled
          04 20 A3     00F9                     IRP$W_FUNC(R3),5$       ;
        44 A5  10   A8 00FC             BISW    #XM$M_CHR_MBX,UCB$L_DEVDEPEND(R5) ; Enable mailbox
               0C   10 0100     5$:     BSBB    XMT_START               ; Start transmit operation
       00000000'GF  17 0102             JMP     G^EXE$QIORETURN         ; Exit QIO service to await completion
                       0108
                       0108     ABORTIO:                                ; Abort the I/O request
       00000000'GF  17 0108             JMP     G^EXE$ABORTIO           ; and exit QIO service
                       010E
                       010E     ;
                       010E     ; Start transmit operation.
                       010E     ;
                       010E     XMT_START:                              ; Start transmit operation
        0094 D5  63   0E 010E             INSQUE  (R3),@UCB$Q_XM_XMT_REQ+4(R5) ; Insert at end of wait queue
                       0113
                       0113     XMT_START_ALT:                          ; Alternate entry to start xmits
               00   E0 0113             BBS     #UCB$V_XM_FORK_XMT,-    ; Br if XMT fork block in use
          1A 68 A5     0115                     UCB$W_DEVSTS(R5),10$    ;
     53   0090 D5  0F 0118             REMQUE  @UCB$Q_XM_XMT_REQ(R5),R3 ; Remove first entry from queue
               13   1D 011D             BVS     10$                     ; Br if none
                       011F     ;
                       011F     ; Find a free mapping register slot.  If none currently available, put
                       011F     ; the I/O request in the wait queue.
                       011F     ;
     54   010B C5  9A 011F             MOVZBL  UCB$B_XM_XMT_MAX(R5),R4 ; Get max concurrent transmits
54  0109 C5   54  00   EB 0124             FFC     #0,R4,UCB$B_XM_XMT_MAP(R5),R4 ; Find a free transmit slot
               06   12 012B             BNEQ    20$                     ; Br if one free
        0090 C5  63   0E 012D             INSQUE  (R3),UCB$Q_XM_XMT_REQ(R5) ; Re-insert request in wait queue
               05      0132     10$:    RSB                             ; Return to caller
                       0133
                       0133     ;
                       0133     ; Allocate UNIBUS map registers
                       0133     ;
                       0133     20$:    ASSUME  IRP$W_BOFF+2 EQ IRP$W_BCNT
                       0133             ASSUME  UCB$W_BOFF+2 EQ UCB$W_BCNT
   7C A5   30 A3  D0 0133             MOVL    IRP$W_BOFF(R3),UCB$W_BOFF(R5) ; Set buffer offset and count
   78 A5   2C A3  D0 0138             MOVL    IRP$L_SVAPTE(R3),UCB$L_SVAPTE(R5) ; Set buffer PTE address
               01   A8 013D             BISW    #UCB$M_XM_FORK_XMT,-    ; Assume we will have to wait
             68 A5     013F                     UCB$W_DEVSTS(R5)        ; ..for fork block
       0000014F'EF  9F 0141             PUSHAB  30$                     ; Push address of fork process
       00000000'GF  17 0147             JMP     G^IOC$REQMAPREG         ; Request map registers
                       014D     ;
                       014D     ; The following code may be executed as a fork process, therefore
                       014D     ; we must provide for a timeout service routine address.
                       014D     ;
             0BB1'     014D             .WORD   TIMEOUT-.               ; Offset to timeout routine
               01   AA 014F     30$:    BICW    #UCB$M_XM_FORK_XMT,-    ; Fork block is no longer in use
             68 A5     0151                     UCB$W_DEVSTS(R5)        ;
               0B   E0 0153             BBS     #XM$V_STS_ACTIVE,-      ; Br if still active
          0C 44 A5     0155                     UCB$L_DEVDEPEND(R5),40$ ;
                       0158             RELMPR                          ; Else, release the map registers
          50   2C   3C 015E             MOVZWL  #SS$_ABORT,R0           ; Return request in error
             0B0C   31 0161             BRW     IO_DONE                 ; Complete the I/O request
                       0164
               57   DD 0164     40$:    PUSHL   R7                      ; Save R7
```

```
           57    24 A5    D0  0166   583          MOVL    UCB$L_CRB(R5),R7          ; Get CRB address
                              016A   584          ASSUME  VEC$W_MAPREG+2 EQ VEC$B_NUMREG
                              016A   585          ASSUME  VEC$B_NUMREG+1 EQ VEC$B_DATAPATH
                 34 A7    D0  016A   586          MOVL    CRB$L_INTD+VEC$W_MAPREG(R7),- ; Save mapping info
            00EC C544         016D   587                  UCB$L_XM_XMT_MAP(R5)[R4]
                              0171   588  ;
                              0171   589  ; Map the buffer
                              0171   590  ;
                              0171   591          SETBIT  R4,UCB$B_XM_XMT_MAP(R5)            ; Set mapping slot in use flag
           3C A3    54    90  0177   592          MOVB    R4,IRP$L_MEDIA+4(R3)               ; Save mapping slot number used
                              017B   593          ASSUME  IRP$W_BOFF+2 EQ IRP$W_BCNT
        38 A3    30 A3    D0  017B   594          MOVL    IRP$W_BOFF(R3),IRP$L_MEDIA(R3)    ; Move byte offset and size
                 34 A7    F0  0180   595          INSV    CRB$L_INTD+VEC$W_MAPREG(R7),-      ; Insert BA9-BA15
     38 A3    07    09        0183   596                  #9,#7,IRP$L_MEDIA(R3)              ;
50    34 A7    02    07   EF  0187   597          EXTZV   #7,#2,CRB$L_INTD+VEC$W_MAPREG(R7),R0 ; Get BA16-BA17
38 A3    02    1E    50   F0  018D   598          INSV    R0,#30,#2,IRP$L_MEDIA(R3)             ; Insert BA16-BA17
         00000000'GF      16  0193   599          JSB     G^IOC$LOADUBAMAPA                  ; Load map registers
                              0199   600  ;
                              0199   601  ; Request and load the port with the buffer address and size, and return.
                              0199   602  ;
                 57 8ED0      0199   603          POPL    R7                                 ; Restore R7
                              019C   604          DSBINT  UCB$B_DIPL(R5)                     ; Synch access to device
                 07E1     30  01A3   605          BSBW    LOAD_PORT                          ; Load port
                              01A6   606          ENBINT                                     ; Restore IPL
                          05  01A9   607          RSB                                        ; Return to caller to await completion
                              01AA   608
```

```
                        01AA              .SBTTL  RCVFDT - Receive I/O FDT routine
                        01AA     ;++
                        01AA     ; RCVFDT - Receive I/O FDT routine
                        01AA     ;
                        01AA     ; Functional description:
                        01AA     ;
                        01AA     ; This routine is called by the SYS$QIO service to dispatch a READ I/O
                        01AA     ; request.
                        01AA     ;
                        01AA     ; The QIO parameters for READs are:
                        01AA     ;
                        01AA     ;        P1 = address of the buffer
                        01AA     ;        P2 = size of the buffer
                        01AA     ;        P3-P6 = (unused)
                        01AA     ;
                        01AA     ; The specified buffer is checked for accessibility. The buffer address and
                        01AA     ; count are saved in the packet. Then IPL is set to device fork IPL and if
                        01AA     ; a message is available the operation is completed.  Otherwise the packet
                        01AA     ; is queued onto the waiting receive list. The mailbox notified bit is cleared.
                        01AA     ;
                        01AA     ; For requests specifing IO$M_NOW, the I/O is completed with status of
                        01AA     ; SS$_ENDOFILE if no message is available when the test is made.
                        01AA     ;
                        01AA     ; For requests specifing IO$M_DSABLMBX the attention mailbox is disabled.
                        01AA     ;
                        01AA     ; Inputs:
                        01AA     ;
                        01AA     ;        R3 = I/O packet address
                        01AA     ;        R4 = PCB address
                        01AA     ;        R5 = UCB address
                        01AA     ;        R6 = CCB address
                        01AA     ;        R7 = Function code
                        01AA     ;        AP = Address of first I/O request parameter
                        01AA     ;
                        01AA     ; Outputs:
                        01AA     ;
                        01AA     ;        R0 = Status of the receive request
                        01AA     ;
                        01AA     ;        R3-R7 preserved.
                        01AA     ;--
                        01AA     RCVFDT:                                            ; Receive function routine
     50    14      3C   01AA              MOVZWL  S^#SS$_BADPARAM,R0                ; Assume illegal size
  51    04 AC      3C   01AD              MOVZWL  P2(AP),R1                         ; Get size
           2F      13   01B1              BEQL    10$                               ; Br if none specified
     50    6C      D0   01B3              MOVL    P1(AP),R0                         ; Get buffer address
  38 A3    50      D0   01B6              MOVL    R0,IRP$L_MEDIA(R3)                ; Save address
     30 A3         B4   01BA              CLRW    IRP$W_BOFF(R3)                    ; No quota to return during completion
 00000000'GF       16   01BD              JSB     G^EXE$READCHK                     ; Check buffer accessibility
                        01C3                                                        ; (no return on no access)
                        01C3              SETIPL  UCB$B_FIPL(R5)                    ; Synchronize access to the UCB
50     0084 8F     3C   01C7              MOVZWL  #SS$_DEVOFFLINE,R0                ; Assume device not active
           0B      E1   01CC              BBC     #XM$V_STS_ACTIVE,-                ; Br if not active - abort I/O
     11 44 A5           01CE                      UCB$L_DEVDEPEND(R5),10$          ;
           0A      E1   01D1              BBC     #IO$V_DSABLMBX,-                  ; Br if not disabling mailbox
     04 20 A3           01D3                      IRP$W_FUNC(R3),5$                 ;
  44 A5    10      AA   01D6              BICW    #XM$M_CHR_MBX,UCB$L_DEVDEPEND(R5) ; Else, disable mailbox
           09      10   01DA     5$:      BSBB    RCV_START                         ; Start receive operation
```

```
      00000000'GF   17  01DC          JMP     G^EXE$QIORETURN             ; Return to await completion
                        01E2
             FF23   31  01E2  10$:    BRW     ABORTIO                     ; Abort the I/O request
                        01E5  ;
                        01E5  ; Start receive operation.
                        01E5  ;
                        01E5  RCV_START:                                  ; Start receive operation
68 A5     0800 8F   AA  01E5          BICW    #UCB$M_XM_NOTIF,UCB$W_DEVSTS(R5) ; Clear notified status
                        01EB  ;
                        01EB  ; Check for message available and complete receive if it is
                        01EB  ;
     52   00C8 D5   0F  01EB          REMQUE  @UCB$Q_XM_RCV_MSG(R5),R2 ; Dequeue a received message
               03   1D  01F0          BVS     15$                         ; Br if none
             0A43   31  01F2          BRW     FINISH_RCV_IO               ; Complete the I/O and exit
                        01F5  ;
                        01F5  ; Queue the requst for future message arrival unless IO$M_NOW specified.
                        01F5  ;
   06 20 A3    06   E0  01F5  15$:    BBS     #IO$V_NOW,IRP$W_FUNC(R3),20$   ; Br if read NOW
    009C D5    63   0E  01FA          INSQUE  (R3),@UCB$Q_XM_RCV_REQ+4(R5)   ; Queue the I/O packet
               05       01FF          RSB                                    ;
                        0200
     50   0870 8F   3C  0200  20$:    MOVZWL  #SS$_ENDOFFILE,R0           ; Set no message status
             0A68   31  0205          BRW     IO_DONE                     ; Complete the I/O and exit
                        0208
                        0208
```

```
                          0208          .SBTTL  ALTFDT - Alternate Transmit/Receive I/O routine
                          0208   ;++
                          0208   ; ALTFDT - Alternate Transmit/Receive dispatch routine
                          0208   ;
                          0208   ; Functional description:
                          0208   ;
                          0208   ; This routine is called by the other drivers to pass an ''internal'' I/O
                          0208   ; request to the driver.  ''Internal'' IRP's are not built via $QIO.
                          0208   ; The action here is to setup the IRP fields as if the packet had been
                          0208   ; processed by the FDT routines.
                          0208   ;
                          0208   ; In this driver, the alternate entry point is called by the DECnet
                          0208   ; Transport layer driver.
                          0208   ;
                          0208   ; Inputs:
                          0208   ;
                          0208   ;       R3 = I/O packet address
                          0208   ;       R5 = UCB address
                          0208   ;
                          0208   ;       All pertinent fields of the IRP are assumed to be valid.
                          0208   ;
                          0208   ;       IPL = FIPL
                          0208   ;
                          0208   ; Outputs:
                          0208   ;
                          0208   ;       R0 = Success
                          0208   ;
                          0208   ;       R3 and R5 preserved.
                          0208   ;--
                          0208   ALTFDT:                                    ; Alternate FDT routine
 50    0084 8F       3C   0208          MOVZWL  #SS$_DEVOFFLINE,R0         ; Assume device not active
              0B     E0   020D          BBS     #XM$V_STS_ACTIVE,-         ; Br if active
         03 44 A5         020F                  UCB$L_DEVDEPEND(R5),5$     ;
            0A5B     31   0212          BRW     IO_DONE                    ; Post the I/O request in error
                          0215
              01     E0   0215   5$:    BBS     #IRP$V_FUNC,-              ; Br if receive function
         05 2A A3         0217                  IRP$W_STS(R3),10$          ;
            FEF1     30   021A          BSBW    XMT_START                  ; Initiate the transmit
              0E     11   021D          BRB     20$                        ;
                          021F
 52       2C A3      D0   021F   10$:   MOVL    IRP$L_SVAPTE(R3),R2        ; Get address of input buffer
              06     13   0223          BEQL    15$                        ; Br if none
            06A6     30   0225          BSBW    ADDRCVLIST                 ; Add it to the receive list
           2C A3     D4   0228          CLRL    IRP$L_SVAPTE(R3)           ; Buffer now used
              B8     10   022B   15$:   BSBB    RCV_START                  ; Initiate the receive
                          022D
    50    01         3C   022D   20$:   MOVZWL  S^#SS$_NORMAL,R0           ; Always return success
                     05   0230          RSB                                ;
                          0231
                          0231
```

```
0231            .SBTTL  SETMODEFDT - Set mode I/O operation FDT dispatch routine
0231    ;++
0231    ; SETMODEFDT - Set mode FDT processing
0231    ;
0231    ; Functional description:
0231    ;
0231    ; This routine is called by the SYS$QIO service to dispatch a SETMODE/SETCHAR
0231    ; I/O request.
0231    ;
0231    ; The QIO parameters for SETMODE or SETCHAR are:
0231    ;
0231    ;       P1 = address of 8 byte characteristics buffer
0231    ;       P2 = (unused)
0231    ;       P3 = number of receive buffers to pre-allocate (IO$M_STARTUP only)
0231    ;       P4-P6 = (unused)
0231    ;
0231    ;   No modifier -
0231    ;
0231    ;       This function is done in the STARTIO routine. Control is passed to
0231    ;       EXE$SETMODE to validate the new mode buffer and queue the packet.
0231    ;
0231    ;   IO$M_CTRL -
0231    ;
0231    ;       Perform this function on the LINE rather than the circuit. The only
0231    ;       extra action that is done, is that on a STARTUP request the modem
0231    ;       is enabled via a master clear to the DMC. This will re-enable the
0231    ;       DTR signal to the modem. On a SHUTDOWN request, the DTR signal is
0231    ;       inhibited. The STARTUP or SHUTDOWN bit is then cleared and the I/O
0231    ;       request is processed as a regular request for the CIRCUIT.
0231    ;
0231    ;   IO$M_STARTUP -
0231    ;
0231    ;       This function starts the unit and sets the mode.
0231    ;       The action here is to pick up the user buffered i/o quota
0231    ;       and allocate the base table. The base table address is saved in
0231    ;       IRP$L_SVAPTE. The quota is taken from the user is in IRP$W_BOFF.
0231    ;       This value will be the IOSB+2 value at I/O done. This function is
0231    ;       complete when the base table has been given to the unit. The mailbox
0231    ;       is enabled and a receive is started.  This function is done partially
0231    ;       here and the remainder is done in STARTIO.
0231    ;
0231    ;   IO$M_SHUTDOWN -
0231    ;
0231    ;       This function shuts down the unit and optionally resets the mode.
0231    ;       A cancel I/O is preformed, all outstanding I/O is completed, the base
0231    ;       table and message blocks are all returned and the unit is left in an
0231    ;       idle state. This function cannot be done here and the FDT processing is
0231    ;       that of all setmode operations.
0231    ;
0231    ;   IO$M_ATTNAST -
0231    ;
0231    ;       This function sets up a AST to be delivered on one of the following
0231    ;       conditions:
0231    ;
0231    ;               Fatal error that caused shutdown.
0231    ;               Message available to be received.
0231    ;
```

```
                          0231   ;
                          0231   ; Inputs:
                          0231   ;
                          0231   ;           R3 = I/O packet address
                          0231   ;           R4 = PCB address
                          0231   ;           R5 = UCB address
                          0231   ;           R6 = CCB address
                          0231   ;           R7 = Function code
                          0231   ;           AP = Address of first I/O request parameter
                          0231   ;
                          0231   ; Outputs:
                          0231   ;
                          0231   ;           R0 = Status of setmode request
                          0231   ;
                          0231   ;           R3-R5 preserved.
                          0231   ;--
                          0231  SETMODEFDT:                                   ; Set mode FDT processing
          2C A3     D4    0231          CLRL    IRP$L_SVAPTE(R3)              ; Set no buffer
     57   20 A3     B0    0234          MOVW    IRP$W_FUNC(R3),R7             ; Get entire function code
     06 57   09     E1    0238          BBC     #IO$V_CTRL,R7,5$              ; Br if not a LINE request
                          023C   ;
                          023C   ; LINE request
                          023C   ;
          00E2   30       023C          BSBW    SETMODEFDT_LINE               ; Process a LINE request
          3A 50     E8    023F          BLBS    R0,10$                        ; Br if request is complete
                          0242
     40 57   08     E1    0242  5$:     BBC     #IO$V_ATTNAST,R7,20$          ; Br if not AST request
                          0246   ;
                          0246   ; Attention AST request
                          0246   ;
  57   0114 C5      9E    0246          MOVAB   UCB$L_XM_AST(R5),R7           ; Set address of AST block listhead
   00000000'GF      16    024B          JSB     G^COM$SETATTNAST              ; Create AST block
                          0251          DSBINT  UCB$B_FIPL(R5)                ; Synch access to UCB
             54     D4    0258          CLRL    R4                            ; Set Mailbox msg
             0C     E5    025A          BBCC    #UCB$V_XM_LOSTERR,-           ; Br unless unreported fatal errors
        06 68 A5          025C                  UCB$W_DEVSTS(R5),7$           ;
     54     00'8F   9A    025F          MOVZBL  #MSG$_XM_SHUTDN,R4            ; Set message code
             0F     11    0263          BRB     8$
             03     E1    0265  7$:     BBC     #UCB$V_XM_INITED,-            ; Br if device not initialized
        12 68 A5          0267                  UCB$W_DEVSTS(R5),10$          ;
  51     00C8 C5    9E    026A          MOVAB   UCB$Q_XM_RCV_MSG(R5),R1       ; Get address received message queue
        61   51     D1    026F          CMPL    R1,(R1)                       ; Any messages in queue?
             08     13    0272          BEQL    10$                           ; Br if no - nothing to report yet
             53     DD    0274  8$:     PUSHL   R3                            ; Save I/O packet address
          0A41   30       0276          BSBW    POKE_USER                     ; Deliver the AST immediately
          53 8ED0         0279          POPL    R3                            ; Restore register
     51   44 A5     D0    027C  10$:    MOVL    UCB$L_DEVDEPEND(R5),R1        ; Get device characteristics
   00000000'GF      17    0280          JMP     G^EXE$FINISHIO                ; Complete the I/O
                          0286   ;
                          0286   ; Set mode, startup, or shutdown request.  Get the characteristics buffer.
                          0286   ;
          38 A3     7C    0286  20$:    CLRQ    IRP$L_MEDIA(R3)               ; Reset mode data buffer
             53     DD    0289          PUSHL   R3                            ; Save I/O packet address
     52      6C     D0    028B          MOVL    P1(AP),R2                     ; Get address of new characteristics
             11     13    028E          BEQL    30$                           ; Br if none specified
     50      0C     3C    0290          MOVZWL  S^#SS$_ACCVIO,R0              ; Assume no access
                          0293          IFNORD  #8,(R2),45$                   ; Br if no access to buffer
```

```
      38 A3    62    7D  0299              MOVQ    (R2),IRP$L_MEDIA(R3)         ; Save new characteristics in packet
      38 A3    01    90  029D              MOVB    #1,IRP$L_MEDIA(R3)           ; Mark it "valid"
      07 57    06    E0  02A1      30$:    BBS     #IO$V_STARTUP,R7,50$         ; Br if startup function
               53  8ED0  02A5              POPL    R3                           ; Restore packet address
               6B    11  02A8              BRB     90$                          ; Queue the packet
               6F    11  02AA      45$:    BRB     100$                         ;
                         02AC      ;
                         02AC      ; Startup request - check caller's quota and allocate the basetable.
                         02AC      ;
      51       08 AC  9A  02AC     50$:    MOVZBL  P3(AP),R1                    ; Get number of receives to preallocate
               52    D5  02B0              TSTL    R2                           ; Any characteristics specified?
               04    12  02B2              BNEQ    55$                          ; Br if yes
      52       40 A5  9E  02B4             MOVAB   UCB$B_DEVCLASS(R5),R2        ; Else, set addr to current ones
      52       02 A2  3C  02B8     55$:    MOVZWL  2(R2),R2                     ; Get receive buffer size
         50    14    3C  02BC              MOVZWL  S^#SS$_BADPARAM,R0           ; Assume bad parameters
         51    52    C4  02BF              MULL    R2,R1                        ; Compute total needed for buffers
               57    13  02C2              BEQL    100$                         ; Br if somehow in error
    51   0100  8F    A0  02C4              ADDW    #BASETAB_SIZE,R1             ; Add size of base table
         57    51    3C  02C9              MOVZWL  R1,R7                        ; Copy quota
         57    51    D1  02CC              CMPL    R1,R7                        ; Overflow?
               4A    12  02CF              BNEQ    100$                         ; Br if in error
     00000000'GF     16  02D1              JSB     G^EXE$BUFQUOPRC              ; Check caller's quota
            41 50    E9  02D7              BLBC    R0,100$                      ; Br if error
    51   010C  8F    3C  02DA              MOVZWL  #BASETAB_SIZE+BAS_C_HEADER,R1 ; Set size of basetable + header
     00000000'GF     16  02DF              JSB     G^EXE$ALLOCBUF               ; Allocate the table
            33 50    E9  02E5              BLBC    R0,100$                      ; Return if error
               53  8ED0  02E8              POPL    R3                           ; Restore I/O packet address
      30 A3    57    B0  02EB              MOVW    R7,IRP$W_BOFF(R3)            ; Save quota in packet
         57    57    3C  02EF              MOVZWL  R7,R7                        ; Convert to longword
    50   0080  C4    D0  02F2              MOVL    PCB$L_JIB(R4),R0             ; Get job info block address
      20 A0    57    C2  02F7              SUBL    R7,JIB$L_BYTCNT(R0)          ; Adjust byte count quota
      24 A0    57    C2  02FB              SUBL    R7,JIB$L_BYTLM(R0)           ; ..and byte limit quota
      2C A3    52    D0  02FF              MOVL    R2,IRP$L_SVAPTE(R3)          ; Save base table data address
      08 A2    51    D0  0303              MOVL    R1,BAS_W_SIZE(R2)            ; Save size of base table
               38    BB  0307              PUSHR   #^M<R3,R4,R5>                ; Save registers
  00    0C A2  00    2C  0309              MOVC5   #0,BAS_T_DATA(R2),#0,-       ; Zero the base table
  0C A2  00F4  8F        030E                      #BASETAB_SIZE-BAS_T_DATA,BAS_T_DATA(R2) ;
               38    BA  0313              POPR    #^M<R3,R4,R5>                ; Restore registers
     00000000'GF     17  0315      90$:    JMP     G^EXE$QIODRVPKT              ; Queue the I/O packet
                         031B      ;
                         031B      ; Setmode/start error
                         031B      ;
               53  8ED0  031B      100$:   POPL    R3                           ; Restore I/O packet address
             FDE7    31  031E              BRW     ABORTIO                      ; Abort the I/O request
                         0321
                         0321          .SBTTL  SETMODEFDT_LINE - Set mode I/O operation FDT routine for LINE
                         0321   ;++
                         0321   ; SETMODEFDT_LINE - Set mode FDT processing for DMC LINE
                         0321   ;
                         0321   ; Functional description:
                         0321   ;
                         0321   ; This routine is called when normal SETMODE FDT processing has detected that
                         0321   ; the I/O request is on the line.
                         0321   ;
                         0321   ; QIO parameters are the same as for regular SETMODE.
                         0321   ;
                         0321   ; Modifiers:
                         0321   ;
                         0321   ;   IO$M_STARTUP -
                         0321   ;
                         0321   ;       This function forces the DMC/DMR to be master cleared to re-enable
                         0321   ;       the DTR modem signal.
                         0321   ;
                         0321   ;   IO$M_SHUTDOWN -
                         0321   ;
                         0321   ;       This function shuts down the unit's modem, by calling a routine to
                         0321   ;       disable the DTR signal to the modem.
                         0321   ;
                         0321   ; Inputs:
                         0321   ;
                         0321   ;       R3 = I/O packet address
                         0321   ;       R4 = PCB address
                         0321   ;       R5 = UCB address
                         0321   ;       R6 = CCB address
                         0321   ;       R7 = Function code
                         0321   ;       AP = Address of first I/O request parameter
                         0321   ;
                         0321   ;       IPL = IPL$_ASTDEL
                         0321   ;
                         0321   ; Outputs:
                         0321   ;
                         0321   ;       R0 = LBC, if we can continue, else all done with request
                         0321   ;       R1 is destroyed, all other registers are preserved
                         0321   ;
                         0321   ;--
                         0321   SETMODEFDT_LINE:                              ; Set mode FDT processing for DMC LINE
         50    01    9A  0321           MOVZBL  #1,R0                         ; Assume we can't continue
               03    E0  0324           BBS     #UCB$V_XM_INITED,-            ; Br if device initialized
      13 68 A5           0326                   UCB$W_DEVSTS(R5),10$          ;  ignore request, circuit must be
                         0329                                                 ;  off before playing with modem.
      16 57    06    E5  0329           BBCC    #IO$V_STARTUP,R7,20$          ; Br if not startup function
                         032D   ;
                         032D   ;       STARTUP LINE request, enable DTR
                         032D   ;
         51    24 A5  D0  032D          MOVL    UCB$L_CRB(R5),R1              ; Get CRB address
                         0331           ASSUME  IDB$L_CSR EQ 0
         51    2C B1  D0  0331          MOVL    @CRB$L_INTD+VEC$L_IDB(R1),R1 ; Get CSR address
      61    4000 8F  B0  0335           MOVW    #XM_I_M_MCLR,(R1)             ; Master clear controller, resets DTR
               50    D4  033A           CLRL    R0                            ; Allow this function to continue
   20 A3    02C0 8F  AA  033C   10$:    BICW    #IO$M_STARTUP!IO$M_SHUTDOWN!- ; Clear out all processed flags
                         0342                   IO$M_CTRL,IRP$W_FUNC(R3) ;
               05         0342          RSB
```

```
                                  0343
              F5 57    07     E1  0343  20$:    BBC     #IO$V_SHUTDOWN,R7,10$    ; Br if not shutdown function, stop
                                  0347  ;
                                  0347  ;       Disable the modem line.
                                  0347  ;
                    0BCE     30   0347          BSBW    DISABLE_MODEM            ; Disable the modem DTR line
                      55     DD   034A          PUSHL   R5                       ; Save UCB address
                                  034C          ASSUME  IRP$L_ARB+4+TQE$C_LENGTH LE IRP$C_LENGTH
     55    53   00000094 8F  C1   034C          ADDL3   #IRP$C_LENGTH-TQE$C_LENGTH,R3,R5 ; Use end of IRP as TQE
               0A A5    0F   90   0354          MOVB    #DYN$C_TQE,TQE$B_TYPE(R5) ; Set structure type
          0C A5     96'AF    9E   0358          MOVAB   B^30$,TQE$L_FPC(R5)      ; Set wakeup routine address
               0B A5    01   90   035D          MOVB    #TQE$C_SSSNGL,TQE$B_RQTYPE(R5) ; Set the TQE request type
               10 A5    53   D0   0361          MOVL    R3,TQE$L_FR3(R5)         ; Save IRP address in TQE
                                  0365          DSBINT  #IPL$_TIMER              ; Raise IPL
  50    00000000 000F4240 8F  7D  036B          MOVQ    #SHUT_TIME,R0            ; Calculate the delta time
              50   00000000'GF   C0   0376      ADDL    G^EXE$GQ_SYSTIME,R0      ;        ...
              51   00000004'GF   D8   037D      ADWC    G^EXE$GQ_SYSTIME+4,R1    ;        ...
                   00000000'GF   16   0384      JSB     G^EXE$INSTIMQ            ; Insert TQE on timer queue
                                  038A          ENBINT                           ; Restore IPL
                      55   8ED0   038D          POPL    R5                       ; Restore UCB address
                   00000000'GF   17   0390      JMP     G^EXE$QIORETURN          ; Wait for the TQE to complete request
                                  0396  ;
                                  0396  ; TQE wakeup routine
                                  0396  ;
                                  0396  ;       R3 = IRP address
                                  0396  ;       R5 = TQE address at end of IRP
                                  0396  ;
                                  0396  ;       IPL = IPL$_TIMER
                                  0396  ;
                50     01     9A  0396  30$:    MOVZBL  #SS$_NORMAL,R0           ; Return success
                      55     DD   0399          PUSHL   R5                       ; Save TQE address
            55    1C A3     D0    039B          MOVL    IRP$L_UCB(R3),R5         ; Copy UCB address to R5
                    0BCE     30   039F          BSBW    IO_DONE                  ; Complete the I/O request
                      55   8ED0   03A2          POPL    R5                       ; Restore TQE address
                             05   03A5          RSB                              ; Return to caller
                                  03A6
                             03A6         .SBTTL  SENSEMODE - Sense mode I/O operation FDT
                             03A6  ;++
                             03A6  ; SENSEMODE - Sense mode FDT processing
                             03A6  ;
                             03A6  ; This routine is called by the SYS$QIO service to dispatch a SENSEMODE
                             03A6  ; SENSECHAR I/O request.
                             03A6  ;
                             03A6  ; The QIO parameters for SENSEMODE are:
                             03A6  ;
                             03A6  ;         P1 = (unused)
                             03A6  ;         P2 = address of descriptor of buffer to receive counters
                             03A6  ;         P3-P6 = (unused)
                             03A6  ;
                             03A6  ; The error counters are returned to the caller in NICE format in the buffer.
                             03A6  ;
                             03A6  ; Inputs:
                             03A6  ;
                             03A6  ;         R3 = I/O packet address
                             03A6  ;         R4 = PCB address
                             03A6  ;         R5 = UCB address
                             03A6  ;         R6 = CCB address
                             03A6  ;         R7 = Function code
                             03A6  ;         AP = Address of first I/O request parameter
                             03A6  ;
                             03A6  ; Outputs:
                             03A6  ;
                             03A6  ;         R0 = Status of diagnose request
                             03A6  ;
                             03A6  ;         R3-R5 preserved.
                             03A6  ;--
                             03A6  SENSEMODEFDT:                                   ; Sense mode FDT routine
      7D 20 A3    08    E1   03A6          BBC     #IO$V_RD_COUNT,IRP$W_FUNC(R3),80$ ; Br if not returning counters
                             03AB  ;
                             03AB  ; Check the caller's buffer
                             03AB  ;
         50    04 AC    D0   03AB          MOVL    P2(AP),R0               ; Get user buffer descriptor address
                             03AF          IFNORD  #8,(R0),10$             ; Check accessibility
            51    60    3C   03B5          MOVZWL  (R0),R1                 ; Get buffer size
                  0F    13   03B8          BEQL    10$                     ; Br if zero - error
         50    04 A0    D0   03BA          MOVL    4(R0),R0                ; Get buffer address
       00000000'GF      16   03BE          JSB     G^EXE$READCHK           ; Check access to buffer
                             03C4                                          ; (no return on no access)
            32    51    D1   03C4          CMPL    R1,#CNT_BUFSIZ          ; Is buffer long enough?
                  06    1E   03C7          BGEQU   20$                     ; Br if yes
            50    14    7D   03C9  10$:    MOVQ    S^#SS$_BADPARAM,R0      ; Set error status
                FD39    31   03CC          BRW     ABORTIO                 ; Abort the I/O request
                             03CF  ;
                             03CF  ; Move driver maintained counters to caller's buffer
                             03CF  ;
                  18    BB   03CF  20$:    PUSHR   #^M<R3,R4>              ; Save registers
            57    50    D0   03D1          MOVL    R0,R7                   ; Set address of caller's buffer
            50    04    D0   03D4          MOVL    #UCB$C_XM_DRVCNT,R0     ; Get number of driver counters
      51    0120 C5     DE   03D7          MOVAL   UCB$L_XM_DRVCNT(R5),R1  ; Get address of driver counters
      52    FC98 CF     9E   03DC          MOVAB   CNTTAB,R2               ; Get address of ID table
            87    82    B0   03E1  30$:    MOVW    (R2)+,(R7)+             ; Set counter ID
            87    81    D0   03E4          MOVL    (R1)+,(R7)+             ; Set counter value
         F7 50          F5   03E7          SOBGTR  R0,30$                  ; Loop through all driver counters
```

```
                                03EA  ;
                                03EA  ; Move device maintained counters to caller's buffer
                                03EA  ;
       53   0118 C5    D0  03EA          MOVL    UCB$L_XM_BASETAB(R5),R3 ; Get address of basetable
                  58   D4  03EF  40$:    CLRL    R8                      ; Init bitmask
              59  82   B0  03F1          MOVW    (R2)+,R9                ; Get next counter ID
                  30   13  03F4          BEQL    70$                     ; Br if end of table
              87  59   B0  03F6          MOVW    R9,(R7)+                ; Set next ID in buffer
                  87   94  03F9          CLRB    (R7)+                   ; Clear count
           02 59  0C   E1  03FB          BBC     #NMA$V_CNT_MAP,R9,50$   ; Br if not bitmapped
                  87   B4  03FF          CLRW    (R7)+                   ; Clear bitmap
                           0401
              54  82   9A  0401  50$:    MOVZBL  (R2)+,R4                ; Get next basetable counter offset
                  E9   13  0404          BEQL    40$                     ; Br if none - no more with this ID
              56  82   9A  0406          MOVZBL  (R2)+,R6                ; Get next UCB counter offset
                  0B   E1  0409          BBC     #XM$V_STS_ACTIVE,-      ; Br if basetable not active
         F3 44 A5          040B                  UCB$L_DEVDEPEND(R5),50$ ;
  50  6146    6344 81      040E          ADDB3   (R3)[R4],(R1)[R6],R0    ; Add basetable counter to saved value
                  EB   1B  0414          BLEQU   50$                     ; Br if overflow, etc.
           06 59  0C   E1  0416          BBC     #NMA$V_CNT_MAP,R9,60$   ; Br if not bitmapped
                  58   B6  041A          INCW    R8                      ; Increment bitmask
           FD A7  58   A8  041C          BISW    R8,-3(R7)               ; Set bitmap
           FF A7  50   80  0420  60$:    ADDB    R0,-1(R7)               ; Add to count
                  DB   11  0424          BRB     50$                     ; Loop through all device counters
                           0426
                  18   BA  0426  70$:    POPR    #^M<R3,R4>              ; Restore registers
                           0428  ;
                           0428  ; See if counters are to be "cleared".  The controller has its own copy of
                           0428  ; the counters in its RAM, so the basetable copies can't simply be cleared.
                           0428  ; Instead, a negative of the basetable copies will be saved in the UCB and,
                           0428  ; later when the counts are requested, the UCB copies will be added to the
                           0428  ; basetable copies.
                           0428  ;
     24 20 A3     0A   E1  0428  80$:    BBC     #IO$V_CLR_COUNT,IRP$W_FUNC(R3),110$ ; Br if not clearing counters
          0120 C5     7C   042D          CLRQ    UCB$L_RCVBYTCNT(R5)     ; Clear byte counts
          0128 C5     7C   0431          CLRQ    UCB$L_RCVMSGCNT(R5)     ; Clear message counts
     51   0130 C5     9E   0435          MOVAB   UCB$B_XM_DEVCNT(R5),R1  ; Get address of saved counters
  52    0118 C5   03   C1  043A          ADDL3   #3,UCB$L_XM_BASETAB(R5),R2 ; Get address of basetable counters
            59    08   D0  0440          MOVL    #UCB$C_XM_DEVCNT,R9     ; Set number of counters
                  81   94  0443  90$:    CLRB    (R1)+                   ; Clear saved counter
                  0B   E1  0445          BBC     #XM$V_STS_ACTIVE,-      ; Br if basetable not active
         04 44 A5          0447                  UCB$L_DEVDEPEND(R5),100$;
           FF A1  82   8E  044A          MNEGB   (R2)+,-1(R1)            ; Store negative of basetable counter
              F2  59   F5  044E  100$:   SOBGTR  R9,90$                  ; Loop through counters
                           0451
     50    32     10   78  0451  110$:   ASHL    #16,#CNT_BUFSIZ,R0      ; Set returned buffer size
           50     01   B0  0455          MOVW    S^#SS$_NORMAL,R0        ; Success return
    00000000'GF        17  0458          JMP     G^EXE$FINISHIOC         ; Post the I/O
                           045E
```

```
                          045E          .SBTTL  STARTIO - Start setmode I/O operation
                          045E  ;++
                          045E  ; STARTIO - Start setmode operation
                          045E  ;
                          045E  ; Functional description:
                          045E  ;
                          045E  ; This routine is entered to process a setmode request.  All setmode
                          045E  ; requests are queued to single-stream them.
                          045E  ;
                          045E  ; For all functions a change in the characteristics is done.
                          045E  ;
                          045E  ; For startup, the action is to request and set up the UNIBUS
                          045E  ; map for the base table and receives. This data is saved
                          045E  ; after allocation in the UCB.  After this the base table and
                          045E  ; receive buffer addresses are passed to the device, thus starting
                          045E  ; the protocol running.
                          045E  ;
                          045E  ; For shutdown, the device is master cleared and all buffers and
                          045E  ; quotas are returned.
                          045E  ;
                          045E  ; Inputs:
                          045E  ;
                          045E  ;         R3 = I/O packet address
                          045E  ;         R5 = UCB address
                          045E  ;
                          045E  ; Outputs:
                          045E  ;
                          045E  ;         R3 and R5 preserved.
                          045E  ;
                          045E  ;         I/O request completed.
                          045E  ;--
                          045E  STARTIO:                                        ; Start I/O routine
                  06   E1 045E          BBC     #IO$V_STARTUP,-                 ; Br if not startup request
            39 20 A3      0460                  IRP$W_FUNC(R3),10$              ;
                          0463  ;
                          0463  ; Startup request
                          0463  ;
                  0B   E1 0463          BBC     #XM$V_STS_ACTIVE,-              ; Br if it is NOT active
            31 44 A5      0465                  UCB$L_DEVDEPEND(R5),5$          ;
       50    0C A3   3C   0468          MOVZWL  IRP$L_PID(R3),R0                ; Get process index from IRP
  51    00000000'GF  D0   046C          MOVL    G^SCH$GL_PCBVEC,R1              ; Get address of PCB address vector
       50      6140  D0   0473          MOVL    (R1)[R0],R0                     ; Get PCB address
            60 A0    D1   0477          CMPL    PCB$L_PID(R0),-                 ; Still same process?
            0C A3         047A                  IRP$L_PID(R3)                   ;
               11    12   047C          BNEQ    3$                              ; Br if not - forget it
     50    0080 C0   D0   047E          MOVL    PCB$L_JIB(R0),R0                ; Get JIB address
       51    30 A3   3C   0483          MOVZWL  IRP$W_BOFF(R3),R1               ; Convert quota to longword
       20 A0    51   C0   0487          ADDL    R1,JIB$L_BYTCNT(R0)             ; Return byte count quota
       24 A0    51   C0   048B          ADDL    R1,JIB$L_BYTLM(R0)              ; ..and byte limit quota
            30 A3    B4   048F  3$:     CLRW    IRP$W_BOFF(R3)                  ; Reset quota charge
     50    02C4 8F   3C   0492          MOVZWL  #SS$_DEVACTIVE,R0               ; Device already started
               1D    11   0497          BRB     40$                             ; Complete the request
                          0499
             0024    31   0499  5$:     BRW     STARTUP                         ; Start the device
                          049C  ;
                          049C  ; Shutdown request
                          049C  ;
```

```
                07    E1  049C  10$:    BBC     #IO$V_SHUTDOWN,-            ; Br if not shutdown request
          0F 20 A3        049E                  IRP$W_FUNC(R3),20$          ;
     50   0084 8F    3C  04A1           MOVZWL  #SS$_DEVOFFLINE,R0          ; Assume device not started yet
                0B    E1  04A6          BBC     #XM$V_STS_ACTIVE,-          ; Br if not active
          0B 44 A5        04A8                  UCB$L_DEVDEPEND(R5),40$     ;
              08F7    30  04AB          BSBW    SHUTDOWN                    ; Shutdown the device
                03    11  04AE          BRB     30$                         ;
                          04B0  ;
                          04B0  ; Just a change mode request
                          04B0  ;
              03F3    30  04B0  20$:    BSBW    CHANGE_MODE                 ; Change mode and characteristics
                          04B3
          50    01    3C  04B3  30$:    MOVZWL  S^#SS$_NORMAL,R0            ; Set success
     51      44 A5    D0  04B6  40$:    MOVL    UCB$L_DEVDEPEND(R5),R1      ; Set device characteristics
                          04BA          REQCOM                              ; Complete the request
                          04C0
```

```
                               04C0         .SBTTL  STARTUP - Start up controller
                               04C0   ;++
                               04C0   ; STARTUP - Start up controller
                               04C0   ;
                               04C0   ; Functional description:
                               04C0   ;
                               04C0   ; This routine starts the controller running. The action is to allocate
                               04C0   ; the map registers for the base table and receives. Once this is done,
                               04C0   ; the unit is master cleared and the base table and mode are set up.
                               04C0   ; The receive buffer list is filled and the receives started.
                               04C0   ;
                               04C0   ;
                               04C0   ; Inputs:
                               04C0   ;
                               04C0   ;       R3 = I/O packet address
                               04C0   ;       R5 = UCB address
                               04C0   ;
                               04C0   ;       IRP$L_MEDIA(R3) =  New mode buffer
                               04C0   ;       IRP$L_SVAPTE(R3) = Address of allocated base table.
                               04C0   ;       IRP$W_BOFF(R3) = Quota taken from caller.
                               04C0   ;
                               04C0   ; Outputs:
                               04C0   ;
                               04C0   ;       Device started and I/O request completed.
                               04C0   ;
                               04C0   ;       R3,R5 preserved.
                               04C0   ;--
                               04C0   STARTUP:                                           ; Startup controller
  44 A5     18    08    00  F0 04C0         INSV    #0,#8,#24,UCB$L_DEVDEPEND(R5)      ; Reset status and error flags
                      03DD  30 04C6         BSBW    CHANGE_MODE                        ; Set new characteristics
                               04C9   ;
                               04C9   ; Initialize the buffer and I/O request queue heads
                               04C9   ;
                50    08    D0 04C9         MOVL    #UCB$C_XM_QUEUES,R0        ; Set number of queue heads
          52    0090  C5    9E 04CC         MOVAB   UCB$Q_XM_QUEUES(R5),R2     ; Set address of first head
                82    62    9E 04D1   10$:  MOVAB   (R2),(R2)+                 ; Set forward link
           82     FC A2     9E 04D4         MOVAB   -4(R2),(R2)+               ; Set backward link
                   F6 50    F5 04D8         SOBGTR  R0,10$                     ; Loop through all queue heads
                               04DB   ;
                               04DB   ; Initialize the transmit and receive mapping info vectors.
                               04DB   ;
                50    0E    D0 04DB         MOVL    #MAX_RCV+MAX_XMT,R0        ; Set number receive and transmit slots
                               04DE         ASSUME  UCB$L_XM_RCV_MAP+<4*MAX_RCV> EQ UCB$L_XM_XMT_MAP
          51    00D0  C5    DE 04DE         MOVAL   UCB$L_XM_RCV_MAP(R5),R1    ; Get mapping vector address
                81    01    CE 04E3   20$:  MNEGL   #1,(R1)+                   ; Indicate no mapping info
                   FA 50    F5 04E6         SOBGTR  R0,20$                     ; Loop through all mapping slots
         010A C5    07      90 04E9         MOVB    #MAX_RCV,UCB$B_XM_RCV_MAX(R5) ; Set maximum concurrent receives
         010B C5    07      90 04EE         MOVB    #MAX_XMT,UCB$B_XM_XMT_MAX(R5) ; Set maximum concurrent transmits
         011C C5    01      CE 04F3         MNEGL   #1,UCB$L_XM_BASEMAP(R5)  ; Set no mapping for basetable yet
                               04F8
              0100 8F       A3 04F8         SUBW3   #BASETAB_SIZE,-            ; Compute quota for receive buffers
       010C C5    30 A3        04FC                 IRP$W_BOFF(R3),UCB$W_XM_QUOTA(R5)
         51   010C C5       3C 0501         MOVZWL  UCB$W_XM_QUOTA(R5),R1      ; Get buffer quota as longword
           50     42 A5     3C 0506         MOVZWL  UCB$W_DEVBUFSIZ(R5),R0     ; Get buffer size as longword
             51     50      C6 050A         DIVL    R0,R1                      ; Compute maximum number of receive
                               050D                                            ;  buffers based on quota
         010A C5    51      91 050D         CMPB    R1,UCB$B_XM_RCV_MAX(R5) ; Is number less than maximum?
```

```
             05   1E  0512  	       BGEQU   30$                      ; Br if not - value ok
     010A C5   51   90  0514  	       MOVB    R1,UCB$B_XM_RCV_MAX(R5)  ; Else reduce number to quota
         0C A3   D0  0519  30$:        MOVL    IRP$L_PID(R3),-          ; Save starter's process ID
      0110 C5        051C  	               UCB$L_XM_PID(R5)         ;
                     051F  ;
                     051F  ; Save basetable info
                     051F  ;
 54   2C A3   0C   C1  051F  	       ADDL3   #BAS_T_DATA,IRP$L_SVAPTE(R3),R4 ; Get basetable address
     0118 C5   54   D0  0524  	       MOVL    R4,UCB$L_XM_BASETAB(R5)  ; Save in UCB
         2C A3   D4  0529  	       CLRL    IRP$L_SVAPTE(R3)         ; No buffer or quota
         30 A3   B4  052C  	       CLRW    IRP$W_BOFF(R3)           ; for I/O post
            08   A8  052F  	       BISW    #UCB$M_XM_INITED,-       ; Indicate UCB fields now initialized
         68 A5        0531  	               UCB$W_DEVSTS(R5)         ; sufficiently so shutdown can cleanup
                     0533  ;
                     0533  ; Allocate map registers for receive buffers.  The
                     0533  ; unbuffered datapath (DP0) is used for all I/O's due to the fact that
                     0533  ; the controller can initiate retransmissions but on the 11/780, the datapath
                     0533  ; requires purging before it can be reused.
                     0533  ;
         42 A5   B0  0533  	       MOVW    UCB$W_DEVBUFSIZ(R5),-    ; Set buffer size
         7E A5        0536  	               UCB$W_BCNT(R5)
 7C A5   01FF 8F   B0  0538  	       MOVW    #511,UCB$W_BOFF(R5)      ; Set worst case byte offset
     54   24 A5   D0  053E  	       MOVL    UCB$L_CRB(R5),R4         ; Get CRB address
                     0542  	       ASSUME  VEC$W_MAPREG+2 EQ VEC$B_NUMREG
                     0542  	       ASSUME  VEC$B_NUMREG+1 EQ VEC$B_DATAPATH
         34 A4   D4  0542  	       CLRL    CRB$L_INTD+VEC$W_MAPREG(R4) ; Clear map register + datapath
       00C0 8F   BB  0545  	       PUSHR   #^M<R6,R7>               ; Save regs
     56   00D0 C5   DE  0549  	       MOVAL   UCB$L_XM_RCV_MAP(R5),R6  ; Get mapping slot address
     57   010A C5   9A  054E  	       MOVZBL  UCB$B_XM_RCV_MAX(R5),R7  ; Get number of receive slots
    00000000'GF   16  0553  40$:        JSB     G^IOC$ALOUBAMAP          ; Allocate a set of map registers
          07 50   E9  0559  	       BLBC    R0,50$                   ; Br if unavailable
     86   34 A4   D0  055C  	       MOVL    CRB$L_INTD+VEC$W_MAPREG(R4),(R6)+ ; Save map info
          F0 57   F5  0560  	       SOBGTR  R7,40$                   ; Continue until done
                     0563
       00C0 8F   BA  0563  50$:        POPR    #^M<R6,R7>               ; Restore regs
          18 50   E9  0567  	       BLBC    R0,60$                   ; Br if error
                     056A  ;
                     056A  ; Map base table
                     056A  ;
       FE00 8F   AB  056A  	       BICW3   #^C<VA$M_BYTE>,-         ; Get basetable byte offset
 7C A5   0118 C5        056E  	               UCB$L_XM_BASETAB(R5),UCB$W_BOFF(R5)
 7E A5   0100 8F   B0  0573  	       MOVW    #BASETAB_SIZE,UCB$W_BCNT(R5) ; Set basetable size
    00000000'GF   16  0579  	       JSB     G^IOC$ALOUBAMAP          ; Allocate map registers
          08 50   E8  057F  	       BLBS    R0,70$                   ; Br if allocated
 50     0344 8F   3C  0582  60$:        MOVZWL  #SS$_INSFMAPREG,R0       ; Set insufficient map registers
          02C4   31  0587  	       BRW     START_ERROR              ;
                     058A
         34 A4   D0  058A  70$:        MOVL    CRB$L_INTD+VEC$W_MAPREG(R4),- ; Save basetable mapping info
      011C C5        058D  	               UCB$L_XM_BASEMAP(R5)     ;
            09   EF  0590  	       EXTZV   S^#VA$V_VPN,-            ; Get basetable page number
 51   0118 C5   15        0592  	               S^#VA$S_VPN,UCB$L_XM_BASETAB(R5),R1
  50   00000000'GF   D0  0597  	       MOVL    G^MMG$GL_SPTBASE,R0      ; Get SPT address
   78 A5   6041   DE  059E  	       MOVAL   (R0)[R1],UCB$L_SVAPTE(R5) ; Set PTE address
    00000000'GF   16  05A3  	       JSB     G^IOC$LOADUBAMAPA        ; Load the basetable map registers
                     05A9  	       ASSUME  UCB$W_BOFF+2 EQ UCB$W_BCNT
         34 A4   F0  05A9  	       INSV    CRB$L_INTD+VEC$W_MAPREG(R4),- ; Set BA9-BA15
 7C A5    07     09        05AC  	               #9,#7,UCB$W_BOFF(R5)     ;
```

```
              02  07   EF  05B0              EXTZV   #7,#2,-                     ; Get BA16-BA17
          50  34  A4       05B3                      CRB$L_INTD+VEC$W_MAPREG(R4),R0
 7C A5    02  1E  50   F0  05B6              INSV    R0,#30,#2,UCB$W_BOFF(R5) ; Set BA16-BA17
                           05BC       ;
                           05BC       ; Master clear the device and notify it of the address of the base table
                           05BC       ;
          54  2C  B4   D0  05BC  80$:        MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4 ; Get CSR address
                           05C0              DSBINT  UCB$B_DIPL(R5)              ; Disable device interrupts
       03 A4      03   90  05C7              MOVB    #DMC_DMR,XM_O_CSR+1(R4)     ; Set DMC/DMR test value
          64  4000 8F  B0  05CB              MOVW    #XM_I_M_MCLR,(R4)           ; Master clear controller
                           05D0              TIMEWAIT #15,#XM_I_M_RUN,(R4),W     ; Wait for RUN - try 150 usecs
              05  50   E9  05F5              BLBC    R0,85$                      ; Br if device NOT ready
                           05F8              ENBINT                              ; Else, re-enable interrupts
                  20   11  05FB              BRB     95$                         ; And continue
                           05FD
                           05FD  85$:        WFIKPCH 90$,#2                      ; Else, wait about a second for diagnostics
                           0607  90$:        IOFORK                              ; Schedule a fork process
    64 A5    0040 8F   AA  060D              BICW    #UCB$M_TIMOUT,UCB$W_STS(R5) ; Clear timeout status
          64 8000 8F   B3  0613              BITW    #XM_I_M_RUN,(R4)            ; Device running?
                  03   12  0618              BNEQ    95$                         ; Br if yes
                022C   31  061A              BRW     START_CTRL_ERROR            ; Else, error
    0C A5   0B84'CF    9E  061D  95$:        MOVAB   W^FORK_PROC,UCB$L_FPC(R5) ; Set Fork process PC address
       03 A4      03   91  0623              CMPB    #DMC_DMR,XM_O_CSR+1(R4)     ; Device a DMC11?
                  03   12  0627              BNEQ    99$                         ; Br if not
                00B8   31  0629              BRW     120$                        ; Else, must be a DMC11
       41 A5      02   90  062C  99$:        MOVB    #DT$_DMR11,UCB$B_DEVTYPE(R5) ; Indicate a DMR11
                           0630       ;
                           0630       ;      DMR unit - get interface bits, modem signals and configuration bits
                           0630       ;
                           0630       ;             Now, get the interface bits (INTMOD, V.35, RS-232, RS-422)
                           0630       ;
              64  20   A8  0630              BISW    #XM_I_M_RQI,(R4)            ; Assert RQI
                           0633              TIMEWAIT #6,#XM_I_M_RDI,(R4),W      ; Wait for controller to come ready
              17  50   E8  0658              BLBS    R0,105$                     ; Br if port ready
                           065B              DSBINT  UCB$B_DIPL(R5)              ; Else, disable device interrupts
                           0662              WFIKPCH 100$,#2                     ; Wait for about 2 seconds
                           066C  100$:       IOFORK                              ; Create a fork process
          50  07  A4   90  0672  105$:       MOVB    XM_UCODE+1(R4),R0           ; Get interface bits
   51     50  02  03   EF  0676              EXTZV   #3,#2,R0,R1                 ; Get interface bits (INTMOD & V.35)
          51  51  02   78  067B              ASHL    #MOD$V_XM_INTMOD,R1,R1      ; Shift to start of interface bits
          47  A5  51   90  067F              MOVB    R1,UCB$L_DEVDEPEND+3(R5) ; Save in UCB @ DEVDEPEND+3
   51     50  FE  8F   78  0683              ASHL    #MOD$V_XM_RS232-6,R0,R1     ; Shift down next two interface bits
          51  CF  8F   8A  0688              BICB    #^C<MOD$M_XM_RS232!-        ; Remove extraneous bits
                           068C                      MOD$M_XM_RS422>,R1          ;
          47  A5  51   88  068C              BISB    R1,UCB$L_DEVDEPEND+3(R5); Save in UCB
                           0690       ;
                           0690       ;      Now, get the modem signals
                           0690       ;
  0150 C5     04  A4   B0  0690              MOVW    XM_PORT(R4),UCB$W_XM_MODSIG(R5) ; Save modem signals
                  64   B4  0696              CLRW    (R4)                        ; Clear RUN, RDI and RQI bits
                           0698       ;
                           0698       ;      Now, check the BSEL1 lockout switch - and get the config bits if okay
                           0698       ;
          64 8000 8F   B3  0698              BITW    #XM_I_M_RUN,(R4)            ; Did we clear RUN?
                  03   13  069D              BEQL    110$                        ; Br if yes - no BSEL1 lockout
                0115   31  069F              BRW     150$                        ; Else, BSEL1 is locked - skip tests
              80  8F   88  06A2  110$:       BISB    #MOD$M_XM_BSEL1,-           ; Indicate BSEL1 is ok
              47 A5         06A5                  UCB$L_DEVDEPEND+3(R5)      ;
   06 A4    2296 8F    B0   06A7          MOVW    #UINST_CNF,XM_UCODE(R4)    ; Request switch pack bits (config)
      64    0300 8F    A8   06AD          BISW    #XM_I_M_STEPUP!XM_I_M_ROMI,(R4)  ; Step microprocessor
                            06B2          WAIT10  #2                         ; Wait 20 useconds
        50     06 A4   B0   06D5          MOVW    XM_UCODE(R4),R0            ; Get configuration bits
51    50    02     01  EF   06D9          EXTZV   #1,#2,R0,R1                ; Get High Speed & DMC compat mode bits
                            06DE          ASSUME  MOD$V_XM_HIGH EQ 0         ;   No shift needed!
      47 A5     51     88   06DE          BISB    R1,UCB$L_DEVDEPEND+3(R5); Save in UCB
                7A     11   06E2          BRB     130$                       ; Continue in common code
                            06E4   ;
                            06E4   ;      DMC unit
                            06E4   ;
                            06E4   ;          Now, check the BSEL1 lockout - and get configuration if okay
                            06E4   ;
      64    8000 8F    AA   06E4   120$:  BICW    #XM_I_M_RUN,(R4)           ; Clear RUN bit
      64    8000 8F    B3   06E9          BITW    #XM_I_M_RUN,(R4)           ; Did we clear it?
                03     13   06EE          BEQL    125$                       ; Br if YES - BSEL1 is okay
              00C4     31   06F0          BRW     150$                       ; Else, BSEL1 is locked out
                            06F3   125$:  ASSUME  MOD$V_XM_HIGH EQ 0         ; Else, read rom u-code
             80 8F     90   06F3          MOVB    #MOD$M_XM_BSEL1,-          ; Indicate BSEL1 is okay
              47 A5         06F6                  UCB$L_DEVDEPEND+3(R5)      ; ..and assume Low Speed u-code
   06 A4    814D 8F    B0   06F8          MOVW    #UINST_RROM,XM_UCODE(R4)   ; Read the DMC rom
      64    0300 8F    A8   06FE          BISW    #XM_I_M_STEPUP!XM_I_M_ROMI,(R4)  ; Step the microprocessor
                            0703          WAIT10  #2                         ; Wait 20 useconds
      64    0300 8F    AA   0726          BICW    #XM_I_M_ROMI!XM_I_M_STEPUP,(R4)  ; Clear maintenance bits
      64    0400 8F    A8   072B          BISW    #XM_I_M_ROMO,(R4)          ; Set ROMO bit
                            0730          WAIT10  #2                         ; Wait 20 useconds
   06 A4    0390 8F    B1   0753          CMPW    #LS_UCODE,XM_UCODE(R4)     ; Is it low-speed u-code?
                03     13   0759          BEQL    130$                       ; Br if yes - okay
                            075B          ASSUME  MOD$M_XM_HIGH EQ 1
              47 A5    96   075B          INCB    UCB$L_DEVDEPEND+3(R5)      ; Else, indicate high-speed u-code
                            075E   130$:  DSBINT  UCB$B_DIPL(R5)             ; Disable device interrupts
      64    4000 8F    B0   0765          MOVW    #XM_I_M_MCLR,(R4)          ; Master clear controller - again!
                            076A          TIMEWAIT #15,#XM_I_M_RUN,(R4),W    ; Wait for RUN - try 150 usecs
              05 50    E9   078F          BLBC    R0,135$                    ; Br if device NOT ready
                            0792          ENBINT                             ; Else, re-enable interrupts
                20     11   0795          BRB     150$                       ; And continue
                            0797
                            0797   135$:  WFIKPCH 140$,#2                    ; Else, wait about a second
                            07A1   140$:  IOFORK                             ; Schedule a fork process
   64 A5    0040 8F    AA   07A7          BICW    #UCB$M_TIMOUT,UCB$W_STS(R5) ; Clear timeout status
      64    8000 8F    B3   07AD          BITW    #XM_I_M_RUN,(R4)           ; Device running?
                03     12   07B2          BNEQ    150$                       ; Br if yes
              0092     31   07B4          BRW     START_CTRL_ERROR           ; Else, error
   0C A5    0B84'CF    9E   07B7   150$:  MOVAB   W^FORK_PROC,UCB$L_FPC(R5)  ; Set Fork process PC address
                            07BD
                            07BD   ;
                            07BD   ; Set LOOPBACK mode if enabled
                            07BD   ;
                01     E1   07BD          BBC     #XM$V_CHR_LOOPB,-          ; Br if not loopback mode
           05 44 A5         07BF                  UCB$L_DEVDEPEND(R5),180$;
      64    0800 8F    A8   07C2          BISW    #XM_I_M_LOOPB,(R4)         ; Else, set loopback flag
      50        23     90   07C7   180$:  MOVB    #XM_I_M_RQI!3,R0           ; Set command for basetable-in
              0097     30   07CA          BSBW    START_REQ_PORT             ; Request port
   04 A4     7C A5     B0   07CD          MOVW    UCB$W_BOFF(R5),XM_PORT(R4)  ; Set basetable BA0-BA15
   06 A4     7E A5     B0   07D2          MOVW    UCB$W_BCNT(R5),XM_PORT+2(R4) ; Set basetable BA16-BA17
                            07D7          SETIPL                             ; Disable all interrupts
```

```
               64 A5    05    E0  07DA              BBS     #UCB$V_POWER,UCB$W_STS(R5),- ; Br if power failed
                        6A        07DE                      START_CTRL_ERROR            ;
                  64    20    AA  07DF              BICW    #XM_I_M_RQI,(R4)            ; Release port
                                  07E2              SETIPL  UCB$B_FIPL(R5)              ; Restore IPL
                      0081    30  07E6              BSBW    START_WAIT_PORT             ; Wait for controller ready
                                  07E9      ;
                                  07E9      ; Set the device mode and enable interrupts
                                  07E9      ;
                  50    21    90  07E9              MOVB    #XM_I_M_RQI!1,R0            ; Set command for control-in
                        76    10  07EC              BSBB    START_REQ_PORT              ; Request port
                     04 A4    B4  07EE              CLRW    XM_PORT(R4)                 ; Clear port (?)
                              AB  07F1              BICW3   #^C<<XM$M_CHR_MOP!-         ; Set mode bits
                                  07F2                            XM$M_CHR_HDPLX!-      ;
                                  07F2                            XM$M_CHR_SLAVE>@8>,-
   06 A4    43 A5    F2FF 8F      07F2                      UCB$L_DEVDEPEND-1(R5),XM_PORT+2(R4)
                  64    20    8A  07F9              BICB    #XM_I_M_RQI,(R4)            ; Free port
                                  07FC
            014C C5     64    B0  07FC              MOVW    XM_I_CSR(R4),UCB$L_XM_LSTCSR(R5); Save CSR values
           014E C5   02 A4    B0  0801              MOVW    XM_O_CSR(R4),UCB$L_XM_LSTCSR+2(R5)
           0148 C5   04 A4    B0  0807              MOVW    XM_PORT(R4),UCB$L_XM_LSTPRT(R5) ; Save port values
           014A C5   06 A4    B0  080D              MOVW    XM_PORT+2(R4),UCB$L_XM_LSTPRT+2(R5)
           0C A5    0B84'CF   9E  0813              MOVAB   W^FORK_PROC,UCB$L_FPC(R5)         ; Set normal fork process
           02 A4     40 8F    90  0819              MOVB    #XM_O_M_IEO,XM_O_CSR(R4)          ; Enable output interrupts
           02 A4     40 8F    90  081E              MOVB    #XM_O_M_IEO,XM_O_CSR(R4)          ; (again)
                                  0823              SETIPL                                    ; Disable all interrupts
               64 A5    05    E0  0826              BBS     #UCB$V_POWER,UCB$W_STS(R5),-      ; Br if power failed
                        1E        082A                      START_CTRL_ERROR                  ;
                   0800 8F    A8  082B              BISW    #XM$M_STS_ACTIVE,-
                     44 A5        082F                      UCB$L_DEVDEPEND(R5)               ; Set controller now active
                                  0831              SETIPL  UCB$B_FIPL(R5)                    ; Restore IPL
                                  0835      ;
                                  0835      ; Start receives and complete the request
                                  0835      ;
                      008E    30  0835              BSBW    FILLRCVLIST                 ; Fill receive buffer list
                   0100 8F    A1  0838              ADDW3   #BASETAB_SIZE,-             ; Set quota as bytecount in I/O status
               50   010C C5       083C                      UCB$W_XM_QUOTA(R5),R0       ;
               50   50  10    78  0840              ASHL    #16,R0,R0                   ; Shift into place
                  50    01    B0  0844              MOVW    S^#SS$_NORMAL,R0            ; Set success
                        0D    11  0847              BRB     START_COMPLETE              ;
                                  0849      ;
                                  0849      ; Error during startup - shutdown and complete I/O request
                                  0849      ;
                                  0849      START_CTRL_ERROR:                           ; Controller error during startup
               50    0054 8F  3C  0849              MOVZWL  #SS$_CTRLERR,R0             ;
                                  084E      START_ERROR:                                ; Error during startup
                        50    DD  084E              PUSHL   R0                          ; Save failure status
                      0552    30  0850              BSBW    SHUTDOWN                    ; Shutdown in case partly started
                        50  8ED0  0853              POPL    R0                          ; Restore status
                                  0856      START_COMPLETE:                             ; Complete startup request
               51    44 A5    D0  0856              MOVL    UCB$L_DEVDEPEND(R5),R1      ; Get device dependent longword
               53    58 A5    D0  085A              MOVL    UCB$L_IRP(R5),R3            ; Get I/O packet address
                                  085E              REQCOM                              ; Complete I/O request
                                  0864
                                  0864      ;++
                                  0864      ; START_REQ_PORT - Startup sequence request port
                                  0864      ; START_WAIT_PORT - Startup sequence wait for port
                                  0864      ;
                   0864  ; Inputs:
                   0864  ;
                   0864  ;          R0 = Command (REQ_PORT only)
                   0864  ;          R4 = CSR address
                   0864  ;          R5 = UCB address
                   0864  ;          00(SP) = Return address
                   0864  ;
                   0864  ; Outputs:
                   0864  ;
                   0864  ;          If unsuccessful, exits to START_CTRL_ERROR.
                   0864  ;--
                   0864  START_REQ_PORT:                                  ; Set function and wait
   64    50   88   0864           BISB     R0,(R4)                        ; Set command in CSR
   64    50   88   0867           BISB     R0,(R4)                        ; (again)
                   086A  START_WAIT_PORT:                                 ; Wait for controller ready
                   086A           SETIPL   UCB$B_FIPL(R5)                 ; Lower IPL
                   086E           TIMEDWAIT        TIME=#25,-
                   086E                    INS1=<BICB3  #^C<XM_I_M_RDI!XM_I_M_RQI>,(R4),R2>,- ; Get flags
                   086E                    INS2=<BEQL   20$>,-      ; Br if both clear -done
                   086E                    INS3=<CMPB   #XM_I_M_RDI!XM_I_M_RQI,R2>,- ; Check if both set
                   086E                    INS4=<BEQL   20$>,-           ; Br if both set - done
                   086E                    DONELBL=20$
 05    50   E8     0899           BLBS     R0,40$                         ; Br if success
 5E    04   C0     089C           ADDL     #4,SP                          ; Else, Pop return address
       A8   11     089F           BRB      START_CTRL_ERROR               ; Exit
                   08A1
                   08A1  40$:     SETIPL   UCB$B_DIPL(R5)                 ; Raise IPL again
            05     08A5           RSB                                     ;
                   08A6
```

```
                          08A6          .SBTTL  CHANGE_MODE - Change mode and characteristics
                          08A6    ;++
                          08A6    ; CHANGE_MODE - Change mode and characteristics
                          08A6    ;
                          08A6    ; Functional description:
                          08A6    ;
                          08A6    ; This routine is entered for changing the mode and characteristics on an idle
                          08A6    ; or active unit:
                          08A6    ;
                          08A6    ; Inputs:
                          08A6    ;
                          08A6    ;         R3 = I/O packet address
                          08A6    ;         R5 = UCB address
                          08A6    ;
                          08A6    ;         IRP$L_MEDIA(R3) = Receive buffer size
                          08A6    ;         IRP$L_MEDIA+4(R3) = New device dependent characteristics
                          08A6    ;
                          08A6    ;         The device dependent longword is defined by $XMDEF:
                          08A6    ;
                          08A6    ;         +----------------+----------------+----------------+----------------+
                          08A6    ;         :    not used    :  error status  :     status     : characteristics :
                          08A6    ;         +----------------+----------------+----------------+----------------+
                          08A6    ;
                          08A6    ; Outputs:
                          08A6    ;
                          08A6    ;         UCB$W_DEVBUFFSIZ(R5) = Receive buffer size
                          08A6    ;         UCB$L_DEVDEPEND(R5) = Device dependent characteristics
                          08A6    ;--
                          08A6   CHANGE_MODE:
           38 A3    97    08A6            DECB    IRP$L_MEDIA(R3)               ; Valid data buffer?
              1A    12    08A9            BNEQ    10$                           ; Br if not
           3A A3    B0    08AB            MOVW    IRP$L_MEDIA+2(R3),-           ; Set new buffer size
           42 A5          08AE                    UCB$W_DEVBUFSIZ(R5)           ;
     FFFFF7FF 8F    CA    08B0            BICL    #^C<XM$M_STS_ACTIVE>,-        ; Clear all but active flag
           44 A5          08B6                    UCB$L_DEVDEPEND(R5)           ;
     00000800 8F    CA    08B8            BICL    #<XM$M_STS_ACTIVE>,-          ; Clear active flag
           3C A3          08BE                    IRP$L_MEDIA+4(R3)             ;
           3C A3    C8    08C0            BISL    IRP$L_MEDIA+4(R3),-           ; Set new characteristics
           44 A5          08C3                    UCB$L_DEVDEPEND(R5)           ;
                    05    08C5   10$:     RSB
                          08C6
```

```
                                   08C6          .SBTTL  FILLRCVLIST - Fill receive buffer list
                                   08C6   ;++
                                   08C6   ; FILLRCVLIST - Fill receive buffer list
                                   08C6   ; ADDRCVLIST - Add a buffer to receive list
                                   08C6   ;
                                   08C6   ; Functional description:
                                   08C6   ;
                                   08C6   ; This routine fills the receive buffer free list up to the quota specified
                                   08C6   ; at device startup.
                                   08C6   ;
                                   08C6   ; Inputs:
                                   08C6   ;
                                   08C6   ;       R2 = Buffer address (ADDRCVLIST only)
                                   08C6   ;       R5 = UCB address
                                   08C6   ;
                                   08C6   ;       IPL = FIPL
                                   08C6   ;
                                   08C6   ; Outputs:
                                   08C6   ;
                                   08C6   ;       R5 = UCB address
                                   08C6   ;       R1,R2,R4 destroyed.
                                   08C6   ;--
                                   08C6   FILLRCVLIST:                                                  ; Fill receive buffer list
                      52      D4   08C6           CLRL    R2                                            ; Clear buffer address
                      0B      E0   08C8           BBS     #XM$V_STS_ACTIVE,-                            ; Continue if device active
             01 44 A5              08CA                   UCB$L_DEVDEPEND(R5),ADDRCVLIST                ;
                              05   08CD           RSB
                                   08CE   ADDRCVLIST:                                                   ; Add to receive buffer list
                      09      BB   08CE           PUSHR   #^M<R0,R3>                                    ; Save registers
                   42 A5      B1   08D0   5$:     CMPW    UCB$W_DEVBUFSIZ(R5),-                         ; Can new block be allocated ?
                 010C C5           08D3                   UCB$W_XM_QUOTA(R5)                            ;
                      34      1A   08D6           BGTRU   20$                                           ; Br if no - list filled
                      51      D4   08D8           CLRL    R1                                            ; Zero size
                 004C 8F      A1   08DA           ADDW3   #RCV_T_DATA+CXB$C_TRAILER,-                   ; Compute needed block size
             51    42 A5           08DE                   UCB$W_DEVBUFSIZ(R5),R1                        ;
                      52      D5   08E1           TSTL    R2                                            ; Buffer allocated already?
                      09      12   08E3           BNEQ    7$                                            ; Br if yes
            00000000'GF       16   08E5           JSB     G^EXE$ALONONPAGED                             ; Allocate nonpaged memory
                   17 50      E9   08EB           BLBC    R0,10$                                        ; Br if failure
             08 A2    51      B0   08EE   7$:     MOVW    R1,RCV_W_BLKSIZE(R2)                          ; Insert block size
             0A A2    17      90   08F2           MOVB    S^#DYN$C_NET,RCV_B_BLKTYPE(R2)                ; Insert block type
           00C0 C5    62      0E   08F6           INSQUE  (R2),UCB$Q_XM_RCV_BUF(R5)                     ; Insert block on list
                   42 A5      A2   08FB           SUBW    UCB$W_DEVBUFSIZ(R5),-                         ; Decrement quota
                 010C C5           08FE                   UCB$W_XM_QUOTA(R5)
                      52      D4   0901           CLRL    R2                                            ; Clear buffer pointer
                      CB      11   0903           BRB     5$                                            ;
                                   0905
                                   0905   10$:    SETBIT  #XM$V_STS_BUFFAIL,-                           ; Set buffer alloc failure
                                   0905                   UCB$L_DEVDEPEND(R5)                           ;
                      10      11   090A           BRB     30$                                           ;
                                   090C
                                   090C   20$:    CLRBIT  #XM$V_STS_BUFFAIL,-                           ; Clear buffer alloc failure
                                   090C                   UCB$L_DEVDEPEND(R5)
                   50 52      D0   0911           MOVL    R2,R0                                         ; Set address of buffer
                      06      13   0914           BEQL    30$                                           ; Br if none
            00000000'GF       16   0916           JSB     G^COM$DRVDEALMEM                              ; Deallocate it
                                   091C
```

```
              091C  30$:    DSBINT  UCB$B_DIPL(R5)                      ; Synch access to device
    06    10  0923          BSBB    START_RECEIVE                       ; Start the receives
              0925          ENBINT                                      ; Restore IPL
    09    BA  0928          POPR    #^M<R0,R3>                          ; Restore registers
          05  092A  40$:    RSB
              092B
```

```
                               092B          .SBTTL  START_RECEIVE - Start any receives
                               092B   ;++
                               092B   ; START_RECEIVE - Start receives
                               092B   ;
                               092B   ; Functional description:
                               092B   ;
                               092B   ; This routine attempts to start any receives that may be pending.  This
                               092B   ; involves dequeueing a free receive buffer, mapping, and loading its address
                               092B   ; and size into the device.
                               092B   ;
                               092B   ; Inputs:
                               092B   ;
                               092B   ;          R5 = UCB address
                               092B   ;
                               092B   ;          IPL = DIPL
                               092B   ;
                               092B   ; Outputs:
                               092B   ;
                               092B   ;          R5 preserved.
                               092B   ;
                               092B   ;          R0 - R4 destroyed
                               092B   ;--
                               092B   START_RECEIVE:                                        ; Start receive operation
           51   010A C5    9A  092B           MOVZBL  UCB$B_XM_RCV_MAX(R5),R1               ; Get max concurrent receives
  51  0108 C5     51   00  EB  0930           FFC     #0,R1,UCB$B_XM_RCV_MAP(R5),R1         ; Get free mapping slot
                       07  13  0937           BEQL    10$                                   ; Br if none
           53   00C0 D5    0F  0939           REMQUE  @UCB$Q_XM_RCV_BUF(R5),R3              ; Get a free buffer
                       01  1C  093E           BVC     20$                                   ; Br if buffer
                           05  0940   10$:    RSB                                           ;
                               0941   ;
                               0941   ; Mark slot in use and create buffer address / character count image,
                               0941   ; and load UNIBUS adapter map registers.
                               0941   ;
                               0941   20$:    SETBIT  R1,UCB$B_XM_RCV_MAP(R5)               ; Mark slot in use
           0B A3      51   90  0947           MOVB    R1,RCV_B_MAPSLOT(R3)                  ; Save mapping slot number used
       54   00D0 C541      DE  094B           MOVAL   UCB$L_XM_RCV_MAP(R5)[R1],R4           ; Get mapping info slot address
           51     48 A3    9E  0951           MOVAB   RCV_T_DATA(R3),R1                     ; Get receive buffer data addr
           0C A3      51   B0  0955           MOVW    R1,RCV_L_BACC(R3)                     ; Set BA0-BA8
                  42 A5    B0  0959           MOVW    UCB$W_DEVBUFSIZ(R5),-                 ; Insert character count
                  0E A3        095C                   RCV_L_BACC+2(R3)                      ;
  0C A3     07    09  64   F0  095E           INSV    (R4),#9,#7,RCV_L_BACC(R3)             ; Set BA9-BA15 from map reg
      50    64    02  07   EF  0964           EXTZV   #7,#2,(R4),R0                         ; Get BA16-BA17 also
  0C A3     02    1E  50   F0  0969           INSV    R0,#30,#2,RCV_L_BACC(R3)              ; Set BA16-BA17
                               096F
                      53   DD  096F           PUSHL   R3                                    ; Save buffer address
           52     02 A4    3C  0971           MOVZWL  2(R4),R2                              ; Set number of map registers
               53     64   3C  0975           MOVZWL  (R4),R3                               ; Set first map register number
                      54   D4  0978           CLRL    R4                                    ; Use unbuffered datapath
           00000000'GF     16  097A           JSB     G^IOC$LOADUBAMAPN                     ; Load the map registers
                  53 8ED0      0980           POPL    R3                                    ; Restore buffer address
                      02   10  0983           BSBB    LOAD_PORT                             ; Load buffer into port
                      A4   11  0985           BRB     START_RECEIVE                         ; That was fun - try another
                               0987
```

```
                          0987              .SBTTL  LOAD_PORT - Load controller input port
                          0987       ;++
                          0987       ; LOAD_PORT - Load controller input port
                          0987       ;
                          0987       ; Functional description:
                          0987       ;
                          0987       ; Request the controller's input port to start an I/O request.  Since the controller
                          0987       ; doesn't service input requests when it is busy, it may not be attainable
                          0987       ; in a reasonable amount of time.  In this case, the driver will just have to
                          0987       ; request an interrupt.
                          0987       ;
                          0987       ; Inputs:
                          0987       ;
                          0987       ;       R3 = Transmit I/O packet or receive buffer
                          0987       ;       R5 = UCB address
                          0987       ;
                          0987       ;       IPL = DIPL
                          0987       ;
                          0987       ; Outputs:
                          0987       ;
                          0987       ;       R0 = Success if port loaded immediately
                          0987       ;       R4 = CSR address
                          0987       ;       R5 = UCB address
                          0987       ;
                          0987       ;       R0-R1 destroyed.
                          0987       ;--
                          0987       LOAD_PORT:                                      ; Load buffer address/size into port
                          0987       ;
                          0987       ; Receive buffers go to head of queue to get initiated first.
                          0987       ; This prevents the link from shutting down due to receive buffer
                          0987       ; starvation.
                          0987       ;
                          0987       ; Note that receive buffers can go onto queue in any order since, they are
                          0987       ; merely empty buckets and one is exactly the same as another.  However,
                          0987       ; transmit buffers contain information and their order must be preserved.
                          0987       ;
      50   00A4 D5   9E   0987              MOVAB   @UCB$Q_XM_PORT+4(R5),R0 ; Assume request goes at tail of queue
       0A    0A A3   91   098C              CMPB    IRP$B_TYPE(R3),S^#DYN$C_IRP ; Is buffer a transmit?
                05   13   0990              BEQL    10$                     ; Br if yes
      50   00A0 C5   9E   0992              MOVAB   UCB$Q_XM_PORT(R5),R0    ; Else, get address of head of queue
          60    63   0E   0997       10$:   INSQUE  (R3),(R0)               ; Insert request in queue
                          099A
                          099A       LOAD_PORT_ALT:                          ; Entry from PORT_INTR routine, order
                          099A                                               ; of entries on port queue is preserved
      54     24 A5   D0   099A              MOVL    UCB$L_CRB(R5),R4        ; Get CRB address
      54     2C B4   D0   099E              MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4 ; Get CSR address
          64    20   B3   09A2              BITW    #XM_I_M_RQI,(R4)        ; Is a request already pending ?
          65    12   12   09A5              BNEQ    10$                     ; Br if yes - leave
                          09A7              TIMEWAIT #5,#XM_I_M_RDI,(R4),W,EQL ; Wait for controller to release port
          3D 50      E9   09CC              BLBC    R0,10$                  ; Br if failure - wait for an interrupt
          64    20   90   09CF              MOVB    #XM_I_M_RQI,(R4)        ; Request input port
 02 A4   0080 8F     B3   09D2              BITW    #XM_O_M_RDO,XM_O_CSR(R4); Is control out pending?
                28   12   09D8              BNEQ    5$                      ; Br if yes - request interrupt
                          09DA              TIMEWAIT #5,#XM_I_M_RDI,(R4),W  ; Wait for controller to come ready
          0D 50      E8   09FF              BLBS    R0,20$                  ; Br if success - port now available
                          0A02       ;
                          0A02       ; Port is not currently available - request an interrupt and wait
```

```
                                 0A02  ; until the interrupt occurs.
                                 0A02  ;
        64   0060 8F      A8     0A02  5$:     BISW    #XM_I_M_RQI!XM_I_M_IEI,(R4) ; Request interrupt
        64   0060 8F      A8     0A07          BISW    #XM_I_M_RQI!XM_I_M_IEI,(R4) ; (again)
                   50     D4     0A0C  10$:    CLRL    R0                          ; Set failure to load
                   05     0A0E                 RSB                                 ;
                                 0A0F  ;
                                 0A0F  ; Port is available - load the buffer address and size into the port
                                 0A0F  ;
        53   00A0 D5      0F     0A0F  20$:    REMQUE  @UCB$Q_XM_PORT(R5),R3       ; Get first entry on port queue
                   57     1D     0A14          BVS     INPUT_DONE                  ; Br if none, assume interrupt processed
                                 0A16                                              ;  the request.
                                 0A16
                                 0A16  LOAD_PORT_AVAIL:                            ; Load port - it's available
         0A   0A A3       91     0A16          CMPB    IRP$B_TYPE(R3),S^#DYN$C_IRP ; Is buffer a transmit?
                   0A     13     0A1A          BEQL    10$                         ; Br if yes
         17   0A A3       91     0A1C          CMPB    IRP$B_TYPE(R3),S^#DYN$C_NET ; Is buffer a receive buffer?
                   26     13     0A20          BEQL    20$                         ; Br if yes
                                 0A22          BUG_CHECK NOBUFPCKT,FATAL           ; Else, fatal error
                                 0A26  ;
                                 0A26  ; Load transmit
                                 0A26  ;
       00AC D5     63     0E     0A26  10$:    INSQUE  (R3),@UCB$Q_XM_XMT_PND+4(R5)      ; Store on pending queue
      04 A4     38 A3     B0     0A2B          MOVW    IRP$L_MEDIA(R3),XM_PORT(R4)       ; Load buffer address and
      06 A4     3A A3     B0     0A30          MOVW    IRP$L_MEDIA+2(R3),XM_PORT+2(R4)   ; character count
 00000000'GF   000000FF 8F  C1   0A35          ADDL3   #255,G^EXE$GL_ABSTIM,-  ; Set 255 second timer
                   6C A5         0A40                  UCB$L_DUETIM(R5)
                   03     A8     0A42          BISW    #UCB$M_TIM!UCB$M_INT,-  ; Enable timer
                64 A5            0A44                  UCB$W_STS(R5)
                   12     11     0A46          BRB     30$                     ;
                                 0A48  ;
                                 0A48  ; Load receive
                                 0A48  ;
       00B4 D5     63     0E     0A48  20$:    INSQUE  (R3),@UCB$Q_XM_RCV_PND+4(R5)      ; Store on pending queue
      04 A4     0C A3     B0     0A4D          MOVW    RCV_L_BACC(R3),XM_PORT(R4)        ; Load buffer address and
      06 A4     0E A3     B0     0A52          MOVW    RCV_L_BACC+2(R3),XM_PORT+2(R4)    ; character count
            64     04     A8     0A57          BISW    #XM_I_M_RCV,(R4)                  ; Set receive buffer type
                                 0A5A
                                 0A5A  30$:    DSBINT                                    ; Disable all interrupts
   05 64 A5        05     E0     0A60          BBS     #UCB$V_POWER,UCB$W_STS(R5),40$    ; Br if powerfailed - forget it
        64   0060 8F      AA     0A65          BICW    #XM_I_M_RQI!XM_I_M_IEI,(R4)       ; Release port, start transfer
                                 0A6A  40$:    ENBINT                                    ; Re-enable interrupts
                                 0A6D
                                 0A6D  INPUT_DONE:
            50     01     3C     0A6D          MOVZWL  S^#SS$_NORMAL,R0                  ; Set success loading
                          05     0A70          RSB
                                 0A71
```

```
                        0A71          .SBTTL  PORT_INTR - Input port ready interrupt service routine
                        0A71    ;++
                        0A71    ; PORT_INTR - Input port ready interrupt service routine
                        0A71    ;
                        0A71    ; Functional description:
                        0A71    ;
                        0A71    ; This interrupt occurs when the port is ready for the driver to pass a
                        0A71    ; buffer address and buffer size to the controller.  Prior to this, a request
                        0A71    ; for the port was made to LOAD_PORT, but the port wasn't available in a
                        0A71    ; short enough amount of time.
                        0A71    ;
                        0A71    ;
                        0A71    ; Inputs:
                        0A71    ;
                        0A71    ;       0(SP) = Address of the unit IDB address
                        0A71    ;       4(SP) - 20(SP) = R1 - R4
                        0A71    ;
                        0A71    ; Outputs:
                        0A71    ;
                        0A71    ;       A receive or transmit is loaded, a check is made for any other
                        0A71    ;       buffers waiting to be loaded and if there are, another request for
                        0A71    ;       the port is made.  Finally, the interrupt is dismissed.
                        0A71    ;
                        0A71    ;       If the interrupt was unexpected, that is no receives or transmits were
                        0A71    ;       pending, the controller is assumed to be in error and is shutdown.
                        0A71    ;--
                        0A71    PORT_INTR:                                              ; Input port ready interrupt
           54    9E  D0 0A71            MOVL    @(SP)+,R4                               ; Get IDB address
        55    18 A4  D0 0A74            MOVL    IDB$L_UCBLST(R4),R5                     ; Get UCB address
              0B     E1 0A78            BBC     #XM$V_STS_ACTIVE,-                      ; Exit if controller not active
           25 44 A5     0A7A                    UCB$L_DEVDEPEND(R5),INTEXIT
           54    64  D0 0A7D            MOVL    (R4),R4                                 ; Get CSR address
 50   00A0 8F    64  AB 0A80            BICW3   (R4),#XM_I_M_RDI!XM_I_M_RQI,R0          ; Is a request really pending?
              1A     12 0A86            BNEQ    INTEXIT                                 ; Br if not - exit
                        0A88
      53  00A0 D5    0F 0A88            REMQUE  @UCB$Q_XM_PORT(R5),R3                   ; Get a waiting buffer/IRP
              1D     1D 0A8D            BVS     INTERR                                  ; If VS then none - error
            FF84     30 0A8F            BSBW    LOAD_PORT_AVAIL                         ; Load and free the port
                        0A92
      50  00A0 C5    9E 0A92    10$:    MOVAB   UCB$Q_XM_PORT(R5),R0                    ; Get address of port queue
           60    50  D1 0A97            CMPL    R0,(R0)                                 ; Any more on queue?
              06     13 0A9A            BEQL    INTEXIT                                 ; Br if no - exit interrupt
            FEFB     30 0A9C            BSBW    LOAD_PORT_ALT                           ; Attempt to load the port
           F0 50     E8 0A9F            BLBS    R0,10$                                  ; Try another
                        0AA2    ;
                        0AA2    ; Exit interrupt
                        0AA2    ;
                        0AA2    INTEXIT:                                                ; Exit interrupt
           50    8E  7D 0AA2            MOVQ    (SP)+,R0                                ; Restore registers
           52    8E  7D 0AA5            MOVQ    (SP)+,R2
           54    8E  7D 0AA8            MOVQ    (SP)+,R4
                     02 0AAB            REI
                        0AAC    ;
                        0AAC    ; An unexpected interrupt occured.  Since there is no NOP function to initiate,
                        0AAC    ; the controller must be shutdown.
                        0AAC    ;
                        0AAC    INTERR:                                                 ;
```

```
                      024F   30  0AAC  1816         BSBW    TIMEOUT                        ; Fake a timeout error
                        F1   11  0AAF  1817         BRB     INTEXIT                        ;
                                 0AB1  1818
```

```
                                0AB1            .SBTTL  CONTROL_INTR - Control out interrupt service routine
                                0AB1    ;++
                                0AB1    ; CONTROL_INTR - Control out interrupt service routine
                                0AB1    ;
                                0AB1    ; FUNCTIONAL DESCRIPTION:
                                0AB1    ;
                                0AB1    ; This routine is the control out interupt service routine.  These interrupts
                                0AB1    ; signal receive or transmit buffer done or errors.
                                0AB1    ;
                                0AB1    ; INPUTS:
                                0AB1    ;
                                0AB1    ;       0(SP) = IDB address
                                0AB1    ;       4(SP) - 20(SP) = R1-R5
                                0AB1    ;
                                0AB1    ; OUTPUTS:
                                0AB1    ;
                                0AB1    ; IMPLICIT OUTPUTS:
                                0AB1    ;
                                0AB1    ;       If the interrupt signals an error,
                                0AB1    ;               the port is held and the fork process is scheduled to process
                                0AB1    ;               the error.
                                0AB1    ;
                                0AB1    ;       If the interrupt signals receive done,
                                0AB1    ;               the port is freed;
                                0AB1    ;               the fork process is scheduled to complete any pending I/O;
                                0AB1    ;               the next receive is started if possible.
                                0AB1    ;
                                0AB1    ;       If the interrupt signals transmit done,
                                0AB1    ;               the port is freed;
                                0AB1    ;               the fork process is scheduled to complete the transmit I/O.
                                0AB1    ;--
                                0AB1    CONTROL_INTR:                                   ; Control out interrupt
            54      9E      D0  0AB1            MOVL    @(SP)+,R4                       ; Get IDB address
        55      18  A4      D0  0AB4            MOVL    IDB$L_UCBLST(R4),R5             ; Get UCB address
            52      64      D0  0AB8            MOVL    (R4),R2                         ; Get CSR address
            0B              E1  0ABB            BBC     #XM$V_STS_ACTIVE,-              ; Br if not active
        E2  44      A5          0ABD                    UCB$L_DEVDEPEND(R5),INTEXIT
        54      02  A2      B0  0AC0            MOVW    XM_O_CSR(R2),R4                 ; Get output CSR,
    54      54      10      78  0AC4            ASHL    #16,R4,R4                       ; shift, and
            54      62      B0  0AC8            MOVW    XM_I_CSR(R2),R4                 ; get input CSR
        53      06  A2      B0  0ACB            MOVW    XM_PORT+2(R2),R3                ; Get port high word,
    53      53      10      78  0ACF            ASHL    #16,R3,R3                       ; shift, and
        53      04  A2      B0  0AD3            MOVW    XM_PORT(R2),R3                  ; get port low word
        04  54      10      E1  0AD7            BBC     #XM_O_V_TYPE+16,R4,10$          ; Br if not error
                    7E      10  0ADB            BSBB    SCHED_FORK                      ; Schedule fork process to report error
                    C3      11  0ADD            BRB     INTEXIT                         ;
                                0ADF
    02  A2      0080  8F    AA  0ADF    10$:    BICW    #XM_O_M_RDO,XM_O_CSR(R2); Release output port
  53    C0000000  8F        CA  0AE5            BICL    #^XC0000000,R3                  ; Clear BA16 and BA17 from BA/CC
                                0AEC                                                    ; (not always correct anyway)
        24  54      12      E1  0AEC            BBC     #XM_O_V_RCV+16,R4,40$           ; Br if transmit complete
                                0AF0    ;
                                0AF0    ; Receive completed.  Get the next receive buffer and schedule the fork
                                0AF0    ; process.
                                0AF0    ;
        52      00B0  D5    0F  0AF0            REMQUE  @UCB$Q_XM_RCV_PND(R5),R2 ; Get oldest pending receive
                    B5      1D  0AF5            BVS     INTERR                          ; Error if none
```

```
         0C A2    53   B1  0AF7            CMPW    R3,RCV_L_BACC(R2)          ; Buffer address match?
                  07   13  0AFB            BEQL    30$                        ; Br if yes - ok
    00B0 C5       62   0E  0AFD   20$:     INSQUE  (R2),UCB$Q_XM_RCV_PND(R5)  ; Requeue the receive buffer
                  A8   11  0B02            BRB     INTERR                     ; Shutdown the controller
                           0B04
      50    0B A2      9A  0B04   30$:     MOVZBL  RCV_B_MAPSLOT(R2),R0       ; Get mapping slot number used
  EF 0108 C5      50   E5  0B08            BBCC    R0,UCB$B_XM_RCV_MAP(R5),20$; Mark the slot free
         0C A2    53   D0  0B0E            MOVL    R3,RCV_L_BACC(R2)          ; Save byte count
                  1F   11  0B12            BRB     100$                       ;
                           0B14      ;
                           0B14      ; Transmit completed.  Get the next transmit I/O packet and schedule fork
                           0B14      ; process to complete the I/O request.
                           0B14      ;
      52    00A8 D5    0F  0B14   40$:     REMQUE  @UCB$Q_XM_XMT_PND(R5),R2 ; Get pending transmit I/O packet
                  91   1D  0B19            BVS     INTERR                     ; Error if none
                  04   12  0B1B            BNEQ    45$                        ; Br if not last one
                  03   AA  0B1D            BICW    #UCB$M_INT!UCB$M_TIM,-     ; Disable timer
               64 A5       0B1F                    UCB$W_STS(R5)
         38 A2    53   B1  0B21   45$:     CMPW    R3,IRP$L_MEDIA(R2)         ; Buffer address match?
                  08   13  0B25            BEQL    60$                        ; Br if yes - ok
    00A8 C5       62   0E  0B27   50$:     INSQUE  (R2),UCB$Q_XM_XMT_PND(R5)  ; Requeue the I/O packet
                FF7D   31  0B2C            BRW     INTERR                     ; Shutdown the controller
                           0B2F
         38 A2    53   D0  0B2F   60$:     MOVL    R3,IRP$L_IOST1(R2)         ; Save byte count
                           0B33
    00BC D5       62   0E  0B33   100$:    INSQUE  (R2),@UCB$Q_XM_POST+4(R5) ; Queue receive buffer or I/O packet
                  03   12  0B38            BNEQ    110$                       ; Br if not first entry
            001E       30  0B3A            BSBW    SCHED_FORK                 ; Schedule fork process
                           0B3D      ;
                           0B3D      ; An input buffer may be waiting to be loaded, but for some reason, the
                           0B3D      ; port was unable to be requested.  Check for this condition and if occuring,
                           0B3D      ; attempt to load the port.  Also, since we may have freed-up a receive slot,
                           0B3D      ; it may be possible to load another receive.
                           0B3D      ;
      10 54       05   E0  0B3D   110$:    BBS     #XM_I_V_RQI,R4,120$        ; Br if input request already pending
      50    00A0 C5    9E  0B41   115$:    MOVAB   UCB$Q_XM_PORT(R5),R0       ; Get address of input request queue
            60    50   D1  0B46            CMPL    R0,(R0)                    ; Anything on queue?
                  06   13  0B49            BEQL    120$                       ; Br if no - start receives
                FE4C   30  0B4B            BSBW    LOAD_PORT_ALT              ; Load and free the port
               F0 50   E8  0B4E            BLBS    R0,115$                    ; Br if success - try for another
      03 54       12   E1  0B51   120$:    BBC     #XM_O_V_RCV+16,R4,130$     ; Br if last transfer wasn't receive
                FDD3   30  0B55            BSBW    START_RECEIVE              ; Start any receives
                FF47   31  0B58   130$:    BRW     INTEXIT                    ; Exit
                           0B5B
```

```
                          0B5B        .SBTTL  SCHED_FORK - Schedule the fork process
                          0B5B  ;++
                          0B5B  ; SCHED_FORK - Schedule the fork process
                          0B5B  ;
                          0B5B  ; Functional description:
                          0B5B  ;
                          0B5B  ; This routine is called to schedule the error and I/O completion fork process.
                          0B5B  ; The last controller port and CSR values are saved for examination.
                          0B5B  ; If the process's execution is already pending, the last port and CSR values
                          0B5B  ; are just saved.
                          0B5B  ;
                          0B5B  ; Inputs:
                          0B5B  ;
                          0B5B  ;       R3 = Last port values
                          0B5B  ;       R4 = Last CSR values
                          0B5B  ;       R5 = UCB address
                          0B5B  ;
                          0B5B  ;       IPL = DIPL or higher
                          0B5B  ;
                          0B5B  ; Outputs:
                          0B5B  ;
                          0B5B  ;       R5 = UCB address
                          0B5B  ;
                          0B5B  ;       UCB$L_XM_LSTPRT(R5) = Last port values
                          0B5B  ;       UCB$L_XM_LSTCSR(R5) = Last CSR values
                          0B5B  ;--
                          0B5B  SCHED_FORK:                                   ; Schedule fork process for execution
            0D      E2    0B5B          BBSS    #UCB$V_XM_FORK_PEND,-         ; Br if fork process scheduling pending
       18 68 A5           0B5D                  UCB$W_DEVSTS(R5),10$
            55      DD    0B60          PUSHL   R5                            ; Save R5
            04      10    0B62          BSBB    5$                            ; Setup fork process
            55    8ED0    0B64          POPL    R5                            ; Restore R5
                    05    0B67          RSB                                   ; Return to caller
                          0B68
55   00000138 8F    C0    0B68  5$:     ADDL    #UCB$B_XM_FKB,R5              ; Point to fork block
           84'AF    9F    0B6F          PUSHAB  B^FORK_PROC                   ; Set address of fork process
     00000000'GF    17    0B72          JMP     G^EXE$FORK                    ; Schedule FORK and return to caller
                          0B78
      05 54   10    E1    0B78  10$:    BBC     #XM_O_V_TYPE+16,R4,20$        ; Br if not an error to handle
    0148 C5   53    7D    0B7C          MOVQ    R3,UCB$L_XM_LSTPRT(R5)        ; Save last port and CSR values
                    05    0B81  20$:    RSB                                   ;
                          0B82
```

```
                                0B82           .SBTTL  FORK_PROC - Error and I/O completion fork process
                                0B82  ;++
                                0B82  ; FORK_PROC - Error and I/O completion fork process
                                0B82  ;
                                0B82  ; Functional description:
                                0B82  ;
                                0B82  ; This routine is called as a fork process to handle errors and I/O
                                0B82  ; completions.
                                0B82  ;
                                0B82  ; Inputs:
                                0B82  ;
                                0B82  ;       R3 = Last port values
                                0B82  ;       R4 = Last CSR values
                                0B82  ;       R5 = UCB address at FORK BLOCK
                                0B82  ;
                                0B82  ;       IPL = FIPL
                                0B82  ;
                                0B82  ; Outputs:
                                0B82  ;
                                0B82  ;       R5 preserved.
                                0B82  ;--
                         017C'  0B82           .WORD   TIMEOUT-.                  ; Offset to timeout routine
                                0B84  FORK_PROC:                                  ; Error/completion fork process
                                0B84           CLRBIT  #UCB$V_XM_FORK_PEND,-      ; Clear fork process scheduling pending
                                0B84                   UCB$W_DEVSTS-UCB$B_XM_FKB(R5)
     55   00000138 8F    C2     0B8A           SUBL    #UCB$B_XM_FKB,R5           ; Point to UCB
            03 54  10    E1     0B91           BBC     #XM_O_V_TYPE+16,R4,20$     ; Br if not error
              0180       30     0B95           BSBW    DEVICE_ERROR               ; Handle the error
                                0B98  ;
                                0B98  ; Complete any transmits or receives
                                0B98  ;
     52    00B8 D5       0F     0B98  20$:     REMQUE  @UCB$Q_XM_POST(R5),R2      ; Get next completed block
                01       1C     0B9D           BVC     23$                        ; Br if one
                05              0B9F           RSB                                ; Else, return
     0A      0A A2       91     0BA0  23$:     CMPB    IRP$B_TYPE(R2),S^#DYN$C_IRP ; Was it a transmit I/O?
                47       13     0BA4           BEQL    50$                        ; Br if yes - complete it
     17      0A A2       91     0BA6           CMPB    IRP$B_TYPE(R2),S^#DYN$C_NET ; Was it a receive?
                04       13     0BAA           BEQL    24$                        ; Br if yes
                                0BAC           BUG_CHECK NOBUFPCKT,FATAL          ; Else, fatal error
                                0BB0  ;
                                0BB0  ; Receive completed - if there is a pending receive I/O request, complete it.
                                0BB0  ; Otherwise, queue the buffer and, if enabled, send a message to mailbox.
                                0BB0  ;
     51      0E A2       3C     0BB0  24$:     MOVZWL  RCV_L_BACC+2(R2),R1        ; Get the byte count
                                0BB4           ADDLC   R1,UCB$L_RCVBYTCNT(R5)     ; Update byte count
           0128 C5       D6     0BC0           INCL    UCB$L_RCVMSGCNT(R5)        ; Update message count
     53    0098 D5       0F     0BC4           REMQUE  @UCB$Q_XM_RCV_REQ(R5),R3   ; Remove waiting receive I/O request
                04       1D     0BC9           BVS     25$                        ; Br if none - queue for later
                6B       10     0BCB           BSBB    FINISH_RCV_IO              ; Else, finish the I/O
                C9       11     0BCD           BRB     20$
                                0BCF
  00CC D5       62       0E     0BCF  25$:     INSQUE  (R2),@UCB$Q_XM_RCV_MSG+4(R5); Else, queue message buffer
                54       D4     0BD4           CLRL    R4                         ; Set no mailbox
                0B       E0     0BD6           BBS     #UCB$V_XM_NOTIF,-          ; Br if already notified
           04 68 A5             0BD8                   UCB$W_DEVSTS(R5),30$       ;
     54      00'8F       9A     0BDB           MOVZBL  #MSG$_XM_DATAVL,R4         ; Set message type
              00D8       30     0BDF  30$:     BSBW    POKE_USER                  ; Poke the user
            06 50   E9  0BE2          BLBC    R0,40$                           ; If low clear then not sent
          0800 8F   A8  0BE5          BISW    #UCB$M_XM_NOTIF,-                ; Set notified
            68 A5       0BE9                  UCB$W_DEVSTS(R5)                 ;
               AB   11  0BEB  40$:    BRB     20$                              ;
                        0BED
                        0BED  ;
                        0BED  ; Transmit completed - deallocate the map registers and complete the I/O
                        0BED  ; request.  If there is a transmit request waiting for mapping resources,
                        0BED  ; restart it.
                        0BED  ;
            53 52   D0  0BED  50$:    MOVL    R2,R3                            ; Get I/O packet address
       51  32 A3    3C  0BF0          MOVZWL  IRP$W_BCNT(R3),R1                ; Get byte count
                        0BF4          ADDLC   R1,UCB$L_XMTBYTCNT(R5)           ; Update byte count
          012C C5   D6  0C00          INCL    UCB$L_XMTMSGCNT(R5)              ; Update message count
       51  3C A3    9A  0C04          MOVZBL  IRP$L_MEDIA+4(R3),R1             ; Get mapping slot number used
                        0C08          CLRBIT  R1,UCB$B_XM_XMT_MAP(R5)          ; Clear in use flag
       52  24 A5    D0  0C0E          MOVL    UCB$L_CRB(R5),R2                 ; Get CRB address
 34 A2  00EC C541   D0  0C12          MOVL    UCB$L_XM_XMT_MAP(R5)[R1],- ; Setup map register data
                        0C19                  CRB$L_INTD+VEC$W_MAPREG(R2)
    00EC C541   01  CE  0C19          MNEGL   #1,UCB$L_XM_XMT_MAP(R5)[R1] ; Set mapping data not allocated
                        0C1F          RELMPR                                   ; Release the map registers
       50  32 A3    B0  0C25          MOVW    IRP$W_BCNT(R3),R0                ; Get count of bytes transmitted
   50  50   10      78  0C29          ASHL    #16,R0,R0                        ; Shift into place
       50   01      B0  0C2D          MOVW    S^#SS$_NORMAL,R0                 ; Set success
            3E      10  0C30          BSBB    IO_DONE                          ; Post the I/O
                        0C32
          F4DE      30  0C32          BSBW    XMT_START_ALT                    ; Continue any waiting requests
          FF60      31  0C35  70$:    BRW     20$                              ;
                        0C38
```

```
                              0C38          .SBTTL  FINISH_RCV_IO - Finish receive I/O processing
                              0C38    ;++
                              0C38    ; FINISH_RCV_IO - Finish receive I/O processing
                              0C38    ;
                              0C38    ; FUNCTIONAL DESCRIPTION:
                              0C38    ;
                              0C38    ; This routine completes a receive operation that has been matched with a
                              0C38    ; message block. After the receive has been completed the message free list is
                              0C38    ; filled and a receive is started if needed.
                              0C38    ;
                              0C38    ; INPUTS:
                              0C38    ;
                              0C38    ;       R2 = message buffer address
                              0C38    ;       R3 = I/O packet address
                              0C38    ;       R5 = UCB address
                              0C38    ;
                              0C38    ;       IPL = FIPL
                              0C38    ;
                              0C38    ; OUTPUTS:
                              0C38    ;
                              0C38    ;       R5 = UCB address
                              0C38    ;
                              0C38    ;       The request is completed via I/O post.
                              0C38    ;--
                              0C38    FINISH_RCV_IO:                                  ; Finish receive I/O request
         2C A3     52    D0   0C38            MOVL    R2,IRP$L_SVAPTE(R3)             ; Save block address
         62     48 A2    9E   0C3C            MOVAB   RCV_T_DATA(R2),(R2)             ; Set address of received data
      04 A2     38 A3    D0   0C40            MOVL    IRP$L_MEDIA(R3),4(R2)           ; Set address of user buffer
                42 A5    A0   0C45            ADDW    UCB$W_DEVBUFSIZ(R5),-           ; Adjust receive buffer quota
              010C C5         0C48                    UCB$W_XM_QUOTA(R5)              ;
         51     0E A2    B0   0C4B            MOVW    RCV_L_BACC+2(R2),R1             ; Get size of transfer
         32 A3     51    B1   0C4F            CMPW    R1,IRP$W_BCNT(R3)               ; Request larger than actual?
                   04    1B   0C53            BLEQU   20$                             ; Br if no
         51     32 A3    3C   0C55            MOVZWL  IRP$W_BCNT(R3),R1               ; Set size to minimum of two sizes
         32 A3     51    B0   0C59    20$:    MOVW    R1,IRP$W_BCNT(R3)               ; Set size to transfer
      50     51    10    78   0C5D            ASHL    #16,R1,R0                       ; Set up status
                   07    12   0C61            BNEQ    25$                             ; Br if success
         50   0054 8F    B0   0C63            MOVW    #SS$_CTRLERR,R0                 ; Set data path error
                   03    11   0C68            BRB     30$                             ;
            50     01    B0   0C6A    25$:    MOVW    S^#SS$_NORMAL,R0                ; Set success
                 FC56    30   0C6D    30$:    BSBW    FILLRCVLIST                     ; Load another receive
                              0C70    ;
                              0C70    ; Complete a transfer I/O request
                              0C70    ;
                              0C70    IO_DONE:                                        ; Complete a transfer I/O request
         38 A3     50    D0   0C70            MOVL    R0,IRP$L_IOST1(R3)              ; Set status and size
      3C A3     44 A5    D0   0C74            MOVL    UCB$L_DEVDEPEND(R5),IRP$L_IOST2(R3) ; Set other info
      13 2A A3     07    E1   0C79            BBC     #IRP$V_DIAGBUF,IRP$W_STS(R3),10$ ; Br if no diagnostic buffer
   50     4C B3    08    C1   0C7E            ADDL3   #8,@IRP$L_DIAGBUF(R3),R0 ; Address buffer past start time
   80    00000000'GF     7D   0C83            MOVQ    G^EXE$GQ_SYSTIME,(R0)+    ; Insert stop time
      80    0082 C5      3C   0C8A            MOVZWL  UCB$W_ERRCNT(R5),(R0)+    ; Insert error counter
                   06    10   0C8F            BSBB    REGDUMP                   ; Dump registers
       00000000'GF       17   0C91    10$:    JMP     G^COM$POST                ; Post the I/O and return
                              0C97
```

```
                         0C97          .SBTTL  REGDUMP - Error log and diagnostics register dump
                         0C97  ;++
                         0C97  ; REGDUMP - Error log and diagnostics register dump routine
                         0C97  ;
                         0C97  ; Functional description:
                         0C97  ;
                         0C97  ; This routine is used to return the controller error counters if a diagnostic
                         0C97  ; buffer was specified for an I/O request.
                         0C97  ;
                         0C97  ; Inputs:
                         0C97  ;
                         0C97  ;        R0 = Diagnostic buffer address
                         0C97  ;        R5 = UCB address
                         0C97  ;
                         0C97  ; Outputs:
                         0C97  ;
                         0C97  ;        R5 = UCB address
                         0C97  ;
                         0C97  ;        R0-R1 destroyed.
                         0C97  ;--
                         0C97  REGDUMP:                                          ; Dump registers and counters
        80    08    9A   0C97          MOVZBL  #8,(R0)+                          ; Insert number longwords returned
  80  014C C5       D0   0C9A          MOVL    UCB$L_XM_LSTCSR(R5),(R0)+ ; Insert last CSR value
  80  0148 C5       D0   0C9F          MOVL    UCB$L_XM_LSTPRT(R5),(R0)+ ; Insert last port value
              80    7C   0CA4          CLRQ    (R0)+                             ; Zero error counters
              0B    E1   0CA6          BBC     #XM$V_STS_ACTIVE,-                ; Br if not active
        0A 44 A5         0CA8                  UCB$L_DEVDEPEND(R5),10$           ;
  51  0118 C5       D0   0CAB          MOVL    UCB$L_XM_BASETAB(R5),R1           ; Get address of base table
                         0CB0          ASSUME  UCB$C_XM_DEVCNT EQ 8
 F8 A0    03 A1     7D   0CB0          MOVQ    3(R1),-8(R0)                      ; Return error counters
              80    7C   0CB5  10$:    CLRQ    (R0)+                             ; Clear other counters
              60    7C   0CB7          CLRQ    (R0)                              ;
              05         0CB9          RSB
                         0CBA
```

```
                           0CBA          .SBTTL  POKE_USER - Poke user process on attention condition
                           0CBA   ;++
                           0CBA   ; POKE_USER - Poke user process on attention condition
                           0CBA   ;
                           0CBA   ; Functional description:
                           0CBA   ;
                           0CBA   ; This routine is used when data is avaliable or a controller error occurs.
                           0CBA   ; the action is to deliver any attention AST's and send a message to the
                           0CBA   ; associated mailbox.
                           0CBA   ;
                           0CBA   ; Inputs:
                           0CBA   ;
                           0CBA   ;       R4 = Mailbox message type
                           0CBA   ;          = Zero if none
                           0CBA   ;       R5 = UCB address
                           0CBA   ;
                           0CBA   ; Outputs:
                           0CBA   ;
                           0CBA   ;       R0 = Low bit clear only if user is not notified
                           0CBA   ;       R5 = UCB address
                           0CBA   ;--
                           0CBA   POKE_USER:                                        ; Poke user process
                 7E   D4   0CBA           CLRL    -(SP)                             ; Assume failure
                 54   DD   0CBC           PUSHL   R4                                ; Save message type
      51   0114 C5    9E   0CBE           MOVAB   UCB$L_XM_AST(R5),R1               ; Get AST listhead
                 61   D5   0CC3           TSTL    (R1)                              ; Empty ?
                 18   13   0CC5           BEQL    17$                               ; If so, branch
              04 AE   D6   0CC7           INCL    4(SP)                             ; Indicate success
              54 51   D0   0CCA           MOVL    R1,R4                             ; Copy listhead address
              51 61   D0   0CCD   10$:    MOVL    (R1),R1                           ; Get address of next block
                 07   13   0CD0           BEQL    15$                               ; Br if none - done
              44 A5   D0   0CD2           MOVL    UCB$L_DEVDEPEND(R5),-             ; Save status as new AST parameter
              1C A1        0CD5                   ACB$L_KAST+4(R1)                  ;
                 F4   11   0CD7           BRB     10$
        00000000'GF   16   0CD9   15$:    JSB     G^COM$DELATTNAST                  ; Deliver the AST's
                           0CDF
            54 8ED0        0CDF   17$:    POPL    R4                                ; Get mailbox message type
                 16   13   0CE2           BEQL    30$                               ; Br if none - no mailbox message
         53   60 A5   D0   0CE4           MOVL    UCB$L_AMB(R5),R3                  ; Get mailbox message address
                 10   13   0CE8           BEQL    30$                               ; Br if none
       0B 44 A5   04  E1   0CEA           BBC     #XM$V_CHR_MBX,UCB$L_DEVDEPEND(R5),30$ ; Br if disabled
        00000000'GF   16   0CEF           JSB     G^EXE$SNDEVMSG                    ; Send the mailbox message
              05 8E   E9   0CF5           BLBC    (SP)+,35$                         ; If AST failed, keep R0
                 01   DD   0CF8           PUSHL   #1                                ; Else force success
            50 8ED0        0CFA   30$:    POPL    R0                                ; Set status
                      05   0CFD   35$:    RSB                                       ;
                           0CFE
```

```
                          0CFE          .SBTTL  TIMEOUT - Transmit timeout handler
                          0CFE  ;++
                          0CFE  ; TIMEOUT - Transmit timeout handler
                          0CFE  ;
                          0CFE  ; Functional description:
                          0CFE  ;
                          0CFE  ; This routine is called by the system clock routine to handle a timed-out
                          0CFE  ; unit.  Transmits are the only I/O that is timed for this device.  If it
                          0CFE  ; has timed-out, the error handling fork process is scheduled.
                          0CFE  ;
                          0CFE  ; Inputs:
                          0CFE  ;
                          0CFE  ;       R5 = UCB address
                          0CFE  ;
                          0CFE  ; Outputs:
                          0CFE  ;
                          0CFE  ;       R5 is preserved.
                          0CFE  ;--
                          0CFE  TIMEOUT:                                        ; Timeout handler
              0B    E1    0CFE          BBC     #XM$V_STS_ACTIVE,-              ; Br if controller inactive
        14 44 A5          0D00                   UCB$L_DEVDEPEND(R5),20$
  53    01    1B    78    0D03          ASHL    #XM_E_V_TIMEOUT+16,#1,R3 ; Set timeout flag
04 64 A5    05    E1      0D07          BBC     #UCB$V_POWER,UCB$W_STS(R5),10$  ; Br if not powerfail
                          0D0C          SETBIT  #XM_E_V_POWER+16,R3             ; Set powerfail flag too
  54    01    10    78    0D10  10$:    ASHL    #XM_O_V_TYPE+16,#1,R4           ; Set error flag
            FE44    30    0D14          BSBW    SCHED_FORK                      ; Schedule the fork process
                    05    0D17  20$:    RSB
                          0D18
```

```
                         0D18          .SBTTL  DEVICE_ERROR - Device error handler
                         0D18   ;++
                         0D18   ; DEVICE_ERROR - Device error handler
                         0D18   ;
                         0D18   ; Functional description:
                         0D18   ;
                         0D18   ; This procedure is called to handle device errors.  If the error is non-fatal,
                         0D18   ; the action is simply to, if enabled, send a mailbox message to the device
                         0D18   ; owner.  If the error is fatal, the fatal error status is saved away in
                         0D18   ; the UCB, if enabled, a mailbox message is sent to the device owner, and the
                         0D18   ; device is shutdown.
                         0D18   ;
                         0D18   ; Inputs:
                         0D18   ;
                         0D18   ;       R3 = Last port values
                         0D18   ;       R4 = Last CSR values
                         0D18   ;       R5 = UCB address
                         0D18   ;
                         0D18   ;       IPL = FIPL
                         0D18   ;
                         0D18   ; Outputs:
                         0D18   ;
                         0D18   ;       R5 preserved.
                         0D18   ;--
                         0D18   DEVICE_ERROR:                                         ; Device error handler
        50    24 A5  D0  0D18          MOVL    UCB$L_CRB(R5),R0                      ; Get CRB address
        50    2C B0  D0  0D1C          MOVL    @CRB$L_INTD+VEC$L_IDB(R0),R0 ; Get CSR address
  02 A0     0080 8F  AA  0D20          BICW    #XM_O_M_RDO,XM_O_CSR(R0) ; Free the port
  53    53    F0 8F  78  0D26          ASHL    #-16,R3,R3                            ; Get last port error value
            0082 C5  B6  0D2B          INCW    UCB$W_ERRCNT(R5)                      ; Increment error count
     53     0F98 8F  B3  0D2F          BITW  #<XM_E_M_PROCERR!-                      ; Was error a fatal error?
                         0D34                  XM_E_M_NONEXMEM!-
                         0D34                  XM_E_M_START!-
                         0D34                  XM_E_M_LOST!-
                         0D34                  XM_E_M_POWER!-
                         0D34                  XM_E_M_TIMEOUT!-
                         0D34                  XM_E_M_MOP>,R3
                  0B  12 0D34          BNEQ    20$                                   ; Br if yes
       45 A5      53  88 0D36          BISB    R3,UCB$L_DEVDEPEND+1(R5); Save error status
     54        00'8F  9A 0D3A          MOVZBL  #MSG$_XM_ATTN,R4                      ; Set mailbox message type
                FF79  31 0D3E          BRW     POKE_USER                             ; If enabled, send mailbox message
                         0D41                                                        ; and return
                         0D41   ;
                         0D41   ; Fatal error - device must be shutdown
                         0D41   ;
            0800 8F  AA  0D41   20$:   BICW    #XM$M_STS_ACTIVE,-                    ; Clear active flag
              44 A5      0D45                  UCB$L_DEVDEPEND(R5)                   ;
                         0D47          ASSUME  <XM_E_M_MOP!XM_E_M_LOST!XM_E_M_START> LE <^XFF>
  46 A5    53   67 8F  8B  0D47          BICB3   #^C<XM_E_M_MOP!-                      ; Save MOP, lost, and start flags
                         0D4D                    XM_E_M_LOST!-                       ;
                         0D4D                    XM_E_M_START>,R3,-                  ;
                         0D4D                  UCB$L_DEVDEPEND+2(R5)                 ;
       14 53      09  E0 0D4D          BBS     #XM_E_V_PROCERR,R3,40$                ; Br if procedure error - don't notify
                         0D51          ASSUME  <XM$M_ERR_FATAL@-16> LE <^XFF>
                  01  88 0D51          BISB    #XM$M_ERR_FATAL@-16,-                 ; Set fatal error flag
               46 A5     0D53                  UCB$L_DEVDEPEND+2(R5)                 ;
     54        00'8F  9A 0D55   30$:   MOVZBL  #MSG$_XM_SHUTDN,R4                    ; Set mailbox message type
```

```
                  FF5E   30  0D59  2276          BSBW    POKE_USER                    ; If enabled, send mailbox message
               06 50     E8  0D5C  2277          BLBS    R0,40$                       ; Br if successful
             1000 8F     A8  0D5F  2278          BISW    #UCB$M_XM_LOSTERR,-          ; Else, remember lost error
               68 A5         0D63  2279                  UCB$W_DEVSTS(R5)             ;
                  3E     11  0D65  2280 40$:     BRB     SHUTDOWN                     ; Shutdown device and return
                             0D67  2281
```

```
                          0D67          .SBTTL SHUTDOWN - Shut down device
                          0D67          .SBTTL CANCEL - Cancel I/O and Deassign Routine
                          0D67  ;++
                          0D67  ; SHUTDOWN - Shut down device
                          0D67  ; CANCEL - Cancel I/O and Deassign Routine
                          0D67  ;
                          0D67  ; Functional description:
                          0D67  ;
                          0D67  ; This routine is used to shut down the device unit as a result of a
                          0D67  ; SETMODE and SHUTDOWN request, a $CANCEL, or a fatal error. The action is
                          0D67  ; to halt the device, deallocate the basetable, deallocate receive
                          0D67  ; buffers, deallocate all map registers, abort all transmit and receive
                          0D67  ; I/O requests, and restore the quotas to the starting process.
                          0D67  ;
                          0D67  ; Inputs:
                          0D67  ;
                          0D67  ;       R5 = UCB address
                          0D67  ;       R8 = Cancel reason code (zero if $CANCEL else $DASSGN)
                          0D67  ;
                          0D67  ;       IPL = FIPL
                          0D67  ;
                          0D67  ; Outputs:
                          0D67  ;
                          0D67  ;       R0-R3 are destroyed.
                          0D67  ;--
                          0D67
                          0D67  CANCEL:                                         ; Cancel I/O routine
          5C A5     B5    0D67          TSTW    UCB$W_REFC(R5)                  ; Is this the last $DASSGN or $CANCEL?
             31     13    0D6A          BEQL    100$                            ; Br if yes
                          0D6C  ;
                          0D6C  ; NOT the last $CANCEL or last $DASSGN
                          0D6C  ;
                          0D6C  ; Perform only a selective $CANCEL (same for $DASSGN)
                          0D6C  ;
             03     E1    0D6C          BBC     #UCB$V_XM_INITED,-              ; Br if unit NOT inited
       2B 68 A5           0D6E                  UCB$W_DEVSTS(R5),10$            ;
                          0D71  ;
                          0D71  ; Flush all attention ASTs for this CHANNEL
                          0D71  ;
        00D4 8F     BB    0D71          PUSHR   #^M<R2,R4,R6,R7>                ; Save registers
   57   0114 C5     9E    0D75          MOVAB   UCB$L_XM_AST(R5),R7             ; Get address of AST listhead
       56    52     3C    0D7A          MOVZWL  R2,R6                           ; Get channel number
     00000000'GF    16    0D7D          JSB     G^COM$FLUSHATTNS                ; Flush all AST for this channel
        00D4 8F     BA    0D83          POPR    #^M<R2,R4,R6,R7>                ; Restore registers
                          0D87  ;
                          0D87  ; Complete all associated receive IRPs
                          0D87  ;
             56     DD    0D87          PUSHL   R6                              ; Save R6
   56   0098 C5     9E    0D89          MOVAB   UCB$Q_XM_RCV_REQ(R5),R6         ; Get address of receive IRPs
           0153     30    0D8E          BSBW    DO_CANCEL                       ; Do the cancel
   56   0090 C5     9E    0D91          MOVAB   UCB$Q_XM_XMT_REQ(R5),R6         ; Get address of XMIT IRPs
           014B     30    0D96          BSBW    DO_CANCEL                       ; Do the cancel
          56 8ED0         0D99          POPL    R6                              ; Restore R6
                    05    0D9C  10$:    RSB                                     ; Return to caller
                          0D9D
                          0D9D  ;
                          0D9D  ; Last $CANCEL or last $DASSGN request
```

```
                         0D9D     ;
        51   01   9A     0D9D     100$:   MOVZBL  #1,R1                   ; Assume last $DASSGN system service
                         0DA0                                             ; modem is cleared only on last $DASSGN
        58   01   91     0DA0             CMPB    #CAN$C_DASSGN,R8        ; Is this a $DASSGN?
             02   13     0DA3             BEQL    SHUTDOWN_ALT            ; Br if yes, shutdown the modem
                         0DA5
                         0DA5     ;
                         0DA5     ; Shutdown request on unit
                         0DA5     ;
                         0DA5     SHUTDOWN:                               ; Shut down unit
             51   D4     0DA5             CLRL    R1                      ; Do not shutdown the modem
                         0DA7
                         0DA7     SHUTDOWN_ALT:
             04   E1     0DA7             BBC     #UCB$V_ONLINE,-         ; Br if not online
      0B 64  A5          0DA9                     UCB$W_STS(R5),10$       ;
             03   E0     0DAC             BBS     #UCB$V_XM_INITED,-      ; Br if UCB initialized
      07 68  A5          0DAE                     UCB$W_DEVSTS(R5),15$    ;
         03  51   E9     0DB1             BLBC    R1,10$                  ; Br if not to clear DTR
           0161   30     0DB4             BSBW    DISABLE_MODEM           ; Else, disable the modem
             05          0DB7     10$:    RSB                             ; Exit
                         0DB8     ;
                         0DB8     ; Clear device and device status
                         0DB8     ;
      00D0 8F     BB     0DB8     15$:    PUSHR   #^M<R4,R6,R7>           ; Save registers
   54    24 A5    D0     0DBC             MOVL    UCB$L_CRB(R5),R4        ; Get CRB address
                         0DC0             ASSUME  IDB$L_CSR EQ 0
   54    2C B4    D0     0DC0             MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4 ; Get CSR address
                         0DC4             DSBINT  UCB$B_DIPL(R5)          ; Synch access to status flags
 64    4000 8F    B0     0DCB             MOVW    #XM_I_M_MCLR,(R4)       ; Master clear the unit
 64 A5      23    AA     0DD0             BICW    #UCB$M_INT!UCB$M_TIM!-  ; Reset device status flags
                         0DD4                     UCB$M_POWER,UCB$W_STS(R5) ;
      0800 8F     AA     0DD4             BICW    #XM$M_STS_ACTIVE,-      ; Reset active flag
        44 A5            0DD8                     UCB$L_DEVDEPEND(R5)     ;
                  AA     0DDA             BICW    #^C<UCB$M_XM_LOSTERR! - ; Clear all but lost error bit,
                         0DDB                         UCB$M_XM_FORK_PEND>,-; and fork process pending
 68 A5    CFFF 8F        0DDB                     UCB$W_DEVSTS(R5)        ;
         03  51   E9     0DE0             BLBC    R1,17$                  ; Br if not to clear DTR
           0132   30     0DE3             BSBW    DISABLE_MODEM           ; Disable the modem
                         0DE6     17$:    ENBINT                          ; Restore IPL
                         0DE9     ;
                         0DE9     ; Deallocate all the attention AST control blocks
                         0DE9     ;
   57    0114 C5  9E     0DE9     20$:    MOVAB   UCB$L_XM_AST(R5),R7     ; Get address of AST listhead
      50      67  D0     0DEE             MOVL    (R7),R0                 ; Anything in the list?
              1B  13     0DF1             BEQL    25$                     ; Br if not
   56      22 A0  3C     0DF3             MOVZWL  ACB$L_KAST+10(R0),R6    ; Force channel match
   52      24 A0  3C     0DF7             MOVZWL  ACB$L_KAST+12(R0),R2    ; Get process index
54    00000000'GF D0     0DFB             MOVL    G^SCH$GL_PCBVEC,R4      ; Get PCB address vector address
      54    6442  D0     0E02             MOVL    (R4)[R2],R4             ; Get PCB address
   00000000'GF    16     0E06             JSB     G^COM$FLUSHATTNS        ; Flush AST
              DB  11     0E0C             BRB     20$                     ; Continue until all flushed
                         0E0E     ;
                         0E0E     ; Release the base table map registers, save the error counters, and
                         0E0E     ; deallocate the base table.
                         0E0E     ;
   54      24 A5  D0     0E0E     25$:    MOVL    UCB$L_CRB(R5),R4        ; Get CRB address
      011C C5     D0     0E12             MOVL    UCB$L_XM_BASEMAP(R5),-  ; Set mapping info
```

```
        34 A4          0E16                 CRB$L_INTD+VEC$W_MAPREG(R4)
           0B   19     0E18         BLSS    27$                             ; Br if none
                       0E1A         RELMPR                                  ; Release the map registers
  011C C5  01   CE     0E20         MNEGL   #1,UCB$L_XM_BASEMAP(R5)         ; Reset mapping info
                       0E25
  50   0118 C5  D0     0E25 27$:    MOVL    UCB$L_XM_BASETAB(R5),R0         ; Get address of base table
           27   13     0E2A         BEQL    30$                             ; Br if none
  51   50  03   C1     0E2C         ADDL3   #3,R0,R1                        ; Set address of error counters
       52  08   D0     0E30         MOVL    #UCB$C_XM_DEVCNT,R2             ; Set number of counters
  53   0130 C5  9E     0E33         MOVAB   UCB$B_XM_DEVCNT(R5),R3          ; Get address of saved counters
       83  81   80     0E38 28$:    ADDB    (R1)+,(R3)+                     ; Add counter to saved counter
        FA 52   F5     0E3B         SOBGTR  R2,28$                          ; Loop through counters
                       0E3E
       0118 C5  D4     0E3E         CLRL    UCB$L_XM_BASETAB(R5)            ; Reset state to no table
  50    F4 A0   9E     0E42         MOVAB   -BAS_T_DATA(R0),R0              ; Reset pointer to start of block
010C C5  0100 8F  A0   0E46         ADDW    #BASETAB_SIZE,UCB$W_XM_QUOTA(R5); Restore quota
   00000000'GF  16     0E4D         JSB     G^COM$DRVDEALMEM                ; Deallocate the base table
                       0E53 ;
                       0E53 ; Release the receive and transmit buffer map registers
                       0E53 ;
           57   D4     0E53 30$:    CLRL    R7                              ; Init slot number
                       0E55         ASSUME  UCB$L_XM_RCV_MAP+<4*MAX_RCV> EQ UCB$L_XM_XMT_MAP
  56   00D0 C5  9E     0E55         MOVAB   UCB$L_XM_RCV_MAP(R5),R6         ; Get address of mapping slots
  34 A4    86   D0     0E5A 50$:    MOVL    (R6)+,CRB$L_INTD+VEC$W_MAPREG(R4) ; Set mapping info
           0A   19     0E5E         BLSS    60$                             ; Br if none allocated
                       0E60         RELMPR                                  ; Release the map registers
     FC A6  01  CE     0E66         MNEGL   #1,-4(R6)                       ; Reset mapping info
                       0E6A 60$:    CLRBIT  R7,UCB$B_XM_RCV_MAP(R5)         ; Clear mapping slot flag
  E6 57    0E   F2     0E70         AOBLSS  #MAX_RCV+MAX_XMT,R7,50$         ; Loop through all mapping slots
                       0E74 ;
                       0E74 ; Deallocate all receive buffers and abort all I/O requests
                       0E74 ;
  56   0090 C5  9E     0E74 90$:    MOVAB   UCB$Q_XM_QUEUES(R5),R6          ; Get address of first queue listhead
       57  08   D0     0E79         MOVL    #UCB$C_XM_QUEUES,R7             ; Get number of queues
  50   00 B6    0F     0E7C 95$:    REMQUE  @(R6),R0                        ; Get next I/O packet/buffer
           29   1D     0E80         BVS     110$                            ; Br if none - queue empty
  0A   0A A0    91     0E82         CMPB    IRP$B_TYPE(R0),S^#DYN$C_IRP     ; Is it an I/O packet?
           0A   13     0E86         BEQL    97$                             ; Br if yes
  17   0A A0    91     0E88         CMPB    IRP$B_TYPE(R0),S^#DYN$C_NET     ; Is it a receive buffer?
           0F   13     0E8C         BEQL    100$                            ; Br if yes
                       0E8E         BUG_CHECK NOBUFPCKT,FATAL               ; Else, fatal error
                       0E92
       53  50   D0     0E92 97$:    MOVL    R0,R3                           ; Set I/O packet address
       50  2C   3C     0E95         MOVZWL  #SS$_ABORT,R0                   ; Set I/O status
         FDD5   30     0E98         BSBW    IO_DONE                         ; Abort the I/O request
           DF   11     0E9B         BRB     95$                             ;
                       0E9D
        42 A5   A0     0E9D 100$:   ADDW    UCB$W_DEVBUFSIZ(R5),-           ; Restore quota
      010C C5          0EA0                 UCB$W_XM_QUOTA(R5)              ;
   00000000'GF  16     0EA3         JSB     G^COM$DRVDEALMEM                ; Deallocate the receive buffer
           D1   11     0EA9         BRB     95$                             ;
                       0EAB
       56  08   C0     0EAB 110$:   ADDL    #8,R6                           ; Increment queue listhead pointer
        CB 57   F5     0EAE         SOBGTR  R7,95$                          ; Loop through queues
                       0EB1 ;
                       0EB1 ; Restore the buffered I/O quota to the starter
                       0EB1 ;
        50   0110 C5   3C  0EB1              MOVZWL  UCB$L_XM_PID(R5),R0           ; Get process index of last starter
   51   00000000'GF    D0  0EB6              MOVL    G^SCH$GL_PCBVEC,R1            ; Get address of PCB address vector
           50   6140   D0  0EBD              MOVL    (R1)[R0],R0                   ; Get PCB address of starter
              60 A0    D1  0EC1              CMPL    PCB$L_PID(R0),-               ; Still same process?
            0110 C5        0EC4                      UCB$L_XM_PID(R5)              ;
                 16    12  0EC7              BNEQ    140$                          ; Br if not - forget it
        50   0080 C0   D0  0EC9              MOVL    PCB$L_JIB(R0),R0              ; Get JIB address
        51   010C C5   3C  0ECE              MOVZWL  UCB$W_XM_QUOTA(R5),R1         ; Convert to longword
        20 A0     51   C0  0ED3              ADDL    R1,JIB$L_BYTCNT(R0)           ; Return byte count quota
        24 A0     51   C0  0ED7              ADDL    R1,JIB$L_BYTLM(R0)            ; ..and byte limit quota
            010C C5    B4  0EDB              CLRW    UCB$W_XM_QUOTA(R5)            ; Reset quota
            00D0 8F    BA  0EDF 140$:        POPR    #^M<R4,R6,R7>                 ; Restore registers
                 05        0EE3              RSB                                   ;
                           0EE4
                           0EE4 DO_CANCEL:                                         ; Cancel the I/O
            53   56    D0  0EE4              MOVL    R6,R3                         ; Copy listhead address
            53   63    D0  0EE7 10$:         MOVL    (R3),R3                       ; Get next entry
            56   53    D1  0EEA              CMPL    R3,R6                         ; At start of list?
                 0F    13  0EED              BEQL    20$                           ; Br if yes
                 0E    10  0EEF              BSBB    CHECK_PKT                     ; Check for match
                 F4    12  0EF1              BNEQ    10$                           ; Br if no match
            53   63    0F  0EF3              REMQUE  (R3),R3                       ; Remove IRP from list
            50   2C    9A  0EF6              MOVZBL  S^#SS$_ABORT,R0               ; Else, set the I/O status return
               FD74    30  0EF9              BSBW    IO_DONE                       ; Abort the I/O request
                 E6    11  0EFC              BRB     DO_CANCEL                     ; Continue from start of list - again
                 05        0EFE 20$:         RSB                                   ; Return to caller
                           0EFF
                           0EFF CHECK_PKT:
        28 A3    52    B1  0EFF              CMPW    R2,IRP$W_CHAN(R3)             ; Channel match?
                 12    12  0F03              BNEQ    20$                           ; Br if no
              0C A3    D5  0F05              TSTL    IRP$L_PID(R3)                 ; Is this an Internal IRP?
                 08    14  0F08              BGTR    10$                           ; Br if NO - PID must match
   0110 C5    60 A4    D1  0F0A              CMPL    PCB$L_PID(R4),UCB$L_XM_PID(R5) ; Starter's PID?
                 05    11  0F10              BRB     20$                           ; Done
     0C A3    60 A4    D1  0F12 10$:         CMPL    PCB$L_PID(R4),IRP$L_PID(R3) ; PIDs match?
                 05        0F17 20$:         RSB
                           0F18
```

```
                        0F18         .SBTTL  DISABLE_MODEM - DISABLE THE MODEM LINE DTR
                        0F18   ;++
                        0F18   ; DISABLE_MODEM - DISABLE THE MODEM
                        0F18   ;
                        0F18   ; Functional description:
                        0F18   ;
                        0F18   ;       This routine will clear the DTR line to the modem to hang up
                        0F18   ;       any phone connection still active.
                        0F18   ;
                        0F18   ;
                        0F18   ; Inputs:
                        0F18   ;
                        0F18   ;       R5 = UCB ADDRESS
                        0F18   ;
                        0F18   ; Outputs:
                        0F18   ;
                        0F18   ;       NONE.
                        0F18   ;
                        0F18   ;--
                        0F18
                        0F18   DISABLE_MODEM:                                          ; Disable the modem line (DTR)
              51    DD  0F18           PUSHL   R1                                      ; Save R1
     51    24 A5    D0  0F1A           MOVL    UCB$L_CRB(R5),R1                        ; Get CRB address
                        0F1E           ASSUME  IDB$L_CSR EQ 0
     51    2C B1    D0  0F1E           MOVL    @CRB$L_INTD+VEC$L_IDB(R1),R1 ; Get CSR address
   61    4000 8F    B0  0F22           MOVW    #XM_I_M_MCLR,(R1)                       ; Master clear the unit
06 A1    A40B 8F    B0  0F27           MOVW    #DROP_DTR,XM_PORT+2(R1)                 ; Load micro-instruction to drop DTR
 01 A1     82 8F    90  0F2D           MOVB    #EXECUTE_UC,T(R1)                       ; Tell controller to execute instruction
              51 8ED0   0F32           POPL    R1                                      ; Restore R1
                    05  0F35           RSB                                             ; Return to caller
                        0F36
                        0F36
                        0F36   XM_END:
                        0F36           .END
```

```
$$$                              = 00000020 R      02
$$$TYP                           = 00000406
$$$WID                           = 00002000
$$OP                             = 00000002
ABORTIO                            00000108 R      03
ACB$L_KAST                       = 00000018
ADDRCVLIST                         000008CE R      03
ALTFDT                             00000208 R      03
AT$_UBA                          = 00000001
BASETAB_SIZE                     = 00000100
BAS_B_SPARE                        0000000B
BAS_B_TYPE                         0000000A
BAS_C_HEADER                       0000000C
BAS_Q_SPARE                        00000000
BAS_T_DATA                         0000000C
BAS_W_SIZE                         00000008
BUG$_NOBUFPCKT                     ********    X   03
CAN$C_DASSGN                     = 00000001
CANCEL                             00000D67 R      03
CHANGE_MODE                        000008A6 R      03
CHECK_PKT                          00000EFF R      03
CNTTAB                             00000078 R      03
CNT_BUFSIZ                       = 00000032
COM$DELATTNAST                     ********    X   03
COM$DRVDEALMEM                     ********    X   03
COM$FLUSHATTNS                     ********    X   03
COM$POST                           ********    X   03
COM$SETATTNAST                     ********    X   03
CONTROL_INTR                       00000AB1 R      03
CRB$L_INTD                       = 00000024
CRB$L_INTD2                      = 00000048
CXB$C_HEADER                     = 00000048
CXB$C_TRAILER                    = 00000004
DC$_SCOM                         = 00000020
DDB$L_DDT                        = 0000000C
DEV$M_AVL                          ********    X   02
DEV$M_IDV                          ********    X   02
DEV$M_NET                          ********    X   02
DEV$M_ODV                          ********    X   02
DEVICE_ERROR                       00000D18 R      03
DISABLE_MODEM                      00000F18 R      03
DMC_DMR                          = 00000003
DO_CANCEL                          00000EE4 R      03
DPT$C_LENGTH                     = 00000038
DPT$C_VERSION                    = 00000004
DPT$INITAB                         00000038 R      02
DPT$REINITAB                       00000054 R      02
DPT$TAB                            00000000 R      02
DROP_DTR                         = 0000A40B
DT$_DMC11                        = 00000001
DT$_DMR11                        = 00000002
DYN$C_CRB                        = 00000005
DYN$C_DDB                        = 00000006
DYN$C_DPT                        = 0000001E
DYN$C_IRP                        = 0000000A
DYN$C_NET                        = 00000017
DYN$C_TQE                        = 0000000F
```

```
DYN$C_UCB                        = 00000010
EXE$ABORTIO                        ********    X   03
EXE$ALLOCBUF                       ********    X   03
EXE$ALONONPAGED                    ********    X   03
EXE$BUFQUOPRC                      ********    X   03
EXE$FINISHIO                       ********    X   03
EXE$FINISHIOC                      ********    X   03
EXE$FORK                           ********    X   03
EXE$GL_ABSTIM                      ********    X   03
EXE$GL_TENUSEC                     ********    X   03
EXE$GL_UBDELAY                     ********    X   03
EXE$GQ_SYSTIME                     ********    X   03
EXE$INSTIMQ                        ********    X   03
EXE$IOFORK                         ********    X   03
EXE$QIODRVPKT                      ********    X   03
EXE$QIORETURN                      ********    X   03
EXE$READCHK                        ********    X   03
EXE$SNDEVMSG                       ********    X   03
EXE$WRITELOCK                      ********    X   03
EXECUTE_UC                       = 00000082
FILLRCVLIST                        000008C6 R      03
FINISH_RCV_IO                      00000C38 R      03
FKB$B_FIPL                       = 0000000B
FKB$C_LENGTH                     = 00000018
FKB$L_FR3                        = 00000010
FORK_PROC                          00000B84 R      03
FUNCTABLE                          00000038 R      03
FUNCTAB_LEN                      = 00000040
IDB$L_CSR                        = 00000000
IDB$L_UCBLST                     = 00000018
INPUT_DONE                         00000A6D R      03
INTERR                             00000AAC R      03
INTEXIT                            00000AA2 R      03
IO$M_CTRL                        = 00000200
IO$M_SHUTDOWN                    = 00000080
IO$M_STARTUP                     = 00000040
IO$V_ATTNAST                     = 00000008
IO$V_CLR_COUNT                   = 0000000A
IO$V_CTRL                        = 00000009
IO$V_DSABLMBX                    = 0000000A
IO$V_ENABLMBX                    = 00000007
IO$V_NOW                         = 00000006
IO$V_RD_COUNT                    = 00000008
IO$V_SHUTDOWN                    = 00000007
IO$V_STARTUP                     = 00000006
IO$_READLBLK                     = 00000021
IO$_READPBLK                     = 0000000C
IO$_READVBLK                     = 00000031
IO$_SENSECHAR                    = 0000001B
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$LOADUBAMAPA                   ********   X    03        NMA$C_CTCIR_RBE                    = 00000410
IOC$LOADUBAMAPN                   ********   X    03        NMA$C_CTCIR_RRT                    = 00000406
IOC$MNTVER                        ********   X    03        NMA$M_CNT_COU                      = 00008000
IOC$RELMAPREG                     ********   X    03        NMA$M_CNT_MAP                      = 00001000
IOC$REQCOM                        ********   X    03        NMA$M_CNT_TYP                      = 00000FFF
IOC$REQMAPREG                     ********   X    03        NMA$V_CNT_MAP                      = 0000000C
IOC$RETURN                        ********   X    03        NMA$V_CNT_WID                      = 0000000D
IOC$WFIKPCH                       ********   X    03        P1                                 = 00000000
IO_DONE                           00000C70 R      03        P2                                 = 00000004
IPL$_TIMER                      = 00000008                  P3                                 = 00000008
IRP$B_TYPE                      = 0000000A                  PCB$L_JIB                          = 00000080
IRP$C_LENGTH                    = 000000C4                  PCB$L_PID                          = 00000060
IRP$L_ARB                       = 00000058                  POKE_USER                            00000CBA R      03
IRP$L_DIAGBUF                   = 0000004C                  PORT_INTR                            00000A71 R      03
IRP$L_IOST1                     = 00000038                  PR$_IPL                            = 00000012
IRP$L_IOST2                     = 0000003C                  RCVFDT                               000001AA R      03
IRP$L_MEDIA                     = 00000038                  RCV_B_BLKTYPE                        0000000A
IRP$L_PID                       = 0000000C                  RCV_B_MAPSLOT                        0000000B
IRP$L_SVAPTE                    = 0000002C                  RCV_L_BACC                           0000000C
IRP$L_UCB                       = 0000001C                  RCV_L_LINK                           00000000
IRP$V_DIAGBUF                   = 00000007                  RCV_START                            000001E5 R      03
IRP$V_FUNC                      = 00000001                  RCV_T_DATA                           00000048
IRP$W_BCNT                      = 00000032                  RCV_W_BLKSIZE                        00000008
IRP$W_BOFF                      = 00000030                  REGDUMP                              00000C97 R      03
IRP$W_CHAN                      = 00000028                  SCH$GL_PCBVEC                        ********   X    03
IRP$W_FUNC                      = 00000020                  SCHED_FORK                           00000B5B R      03
IRP$W_STS                       = 0000002A                  SENSEMODEFDT                         000003A6 R      03
JIB$L_BYTCNT                    = 00000020                  SETMODEFDT                           00000231 R      03
JIB$L_BYTLM                     = 00000024                  SETMODEFDT_LINE                      00000321 R      03
LOAD_PORT                         00000987 R      03        SHUTDOWN                             00000DA5 R      03
LOAD_PORT_ALT                     0000099A R      03        SHUTDOWN_ALT                         00000DA7 R      03
LOAD_PORT_AVAIL                   00000A16 R      03        SHUT_TIME                          = 000F4240
LS_UCODE                        = 00000300                  SIZ...                             = 00000001
MASKH                           = 00000080                  SS$_ABORT                          = 0000002C
MASKL                           = 08000000                  SS$_ACCVIO                         = 0000000C
MAX_C_BUFSIZE                   = 00003FFF                  SS$_BADPARAM                       = 00000014
MAX_RCV                         = 00000007                  SS$_CTRLERR                        = 00000054
MAX_XMT                         = 00000007                  SS$_DEVACTIVE                      = 000002C4
MMG$GL_SPTBASE                    ********   X    03        SS$_DEVOFFLINE                     = 00000084
MOD$M_XM_BSEL1                  = 00000080                  SS$_ENDOFFILE                      = 00000870
MOD$M_XM_HIGH                   = 00000001                  SS$_INSFMAPREG                     = 00000344
MOD$M_XM_RS232                  = 00000010                  SS$_NORMAL                         = 00000001
MOD$M_XM_RS422                  = 00000020                  STARTIO                              0000045E R      03
MOD$V_XM_HIGH                   = 00000000                  STARTUP                              000004C0 R      03
MOD$V_XM_INTMOD                 = 00000002                  START_COMPLETE                       00000856 R      03
MOD$V_XM_RS232                  = 00000004                  START_CTRL_ERROR                     00000849 R      03
MSG$_XM_ATTN                      ********   X    03        START_ERROR                          0000084E R      03
MSG$_XM_DATAVL                    ********   X    03        START_RECEIVE                        0000092B R      03
MSG$_XM_SHUTDN                    ********   X    03        START_REQ_PORT                       00000864 R      03
NMA$C_CTCIR_BRC                 = 000003E8                  START_WAIT_PORT                      0000086A R      03
NMA$C_CTCIR_BSN                 = 000003E9                  TIMEOUT                              00000CFE R      03
NMA$C_CTCIR_DBR                 = 000003F2                  TQE$B_RQTYPE                       = 0000000B
NMA$C_CTCIR_DBS                 = 000003F3                  TQE$B_TYPE                         = 0000000A
NMA$C_CTCIR_DEI                 = 000003FC                  TQE$C_LENGTH                       = 00000030
NMA$C_CTCIR_DEO                 = 000003FD                  TQE$C_SSSNGL                       = 00000001
NMA$C_CTCIR_LBE                 = 00000411                  TQE$L_FPC                          = 0000000C
NMA$C_CTCIR_LRT                 = 00000407                  TQE$L_FR3                          = 00000010
```

```
                                                              M 5
XMDRIVER                         - VAX/VMS DMC11/DMR11 Device Driver      16-SEP-1984 00:26:05  VAX/VMS Macro V04-00        Page  58
Symbol table                                                               5-SEP-1984 00:20:43  [DRIVER.SRC]XMDRIVER.MAR;1        (27)

UCB$B_DEVCLASS                  = 00000040          UCB$Q_XM_RCV_BUF                  000000C0
UCB$B_DEVTYPE                   = 00000041          UCB$Q_XM_RCV_MSG                  000000C8
UCB$B_DIPL                      = 0000005E          UCB$Q_XM_RCV_PND                  000000B0
UCB$B_FIPL                      = 0000000B          UCB$Q_XM_RCV_REQ                  00000098
UCB$B_XM_DCER                     00000132          UCB$Q_XM_XMT_PND                  000000A8
UCB$B_XM_DCES                     00000135          UCB$Q_XM_XMT_REQ                  00000090
UCB$B_XM_DEVCNT                   00000130          UCB$V_ONLINE                    = 00000004
UCB$B_XM_FKB                      00000138          UCB$V_POWER                     = 00000005
UCB$B_XM_HCER                     00000131          UCB$V_XM_FORK_PEND              = 0000000D
UCB$B_XM_HCES                     00000134          UCB$V_XM_FORK_XMT               = 00000000
UCB$B_XM_NBFR                     00000130          UCB$V_XM_INITED                 = 00000003
UCB$B_XM_NBFS                     00000133          UCB$V_XM_LOSTERR                = 0000000C
UCB$B_XM_RCV_MAP                  00000108          UCB$V_XM_NOTIF                  = 0000000B
UCB$B_XM_RCV_MAX                  0000010A          UCB$W_BCNT                      = 0000007E
UCB$B_XM_REPR                     00000137          UCB$W_BOFF                      = 0000007C
UCB$B_XM_REPS                     00000136          UCB$W_DEVBUFSIZ                 = 00000042
UCB$B_XM_XMT_MAP                  00000109          UCB$W_DEVSTS                    = 00000068
UCB$B_XM_XMT_MAX                  0000010B          UCB$W_ERRCNT                    = 00000082
UCB$C_LENGTH                    = 00000090          UCB$W_REFC                      = 0000005C
UCB$C_XM_DEVCNT                 = 00000008          UCB$W_STS                       = 00000064
UCB$C_XM_DRVCNT                 = 00000004          UCB$W_XM_MODSIG                   00000150
UCB$C_XM_LENGTH                   00000152          UCB$W_XM_QUOTA                    0000010C
UCB$C_XM_QUEUES                 = 00000008          UINST_CNF                       = 00002296
UCB$L_AMB                       = 00000060          UINST_RROM                      = 0000814D
UCB$L_CRB                       = 00000024          UNIT_INIT                         000000A4 R      03
UCB$L_DEVCHAR                   = 00000038          VA$M_BYTE                       = 000001FF
UCB$L_DEVDEPEND                 = 00000044          VA$S_VPN                        = 00000015
UCB$L_DUETIM                    = 0000006C          VA$V_VPN                        = 00000009
UCB$L_FPC                       = 0000000C          VEC$B_DATAPATH                  = 00000013
UCB$L_IRP                       = 00000058          VEC$B_NUMREG                    = 00000012
UCB$L_RCVBYTCNT                   00000120          VEC$L_IDB                       = 00000008
UCB$L_RCVMSGCNT                   00000128          VEC$L_UNITINIT                  = 00000018
UCB$L_SVAPTE                    = 00000078          VEC$W_MAPREG                    = 00000010
UCB$L_XMTBYTCNT                   00000124          XM$DDT                            00000000 RG     03
UCB$L_XMTMSGCNT                   0000012C          XM$M_CHR_HDPLX                  = 00000004
UCB$L_XM_AST                      00000114          XM$M_CHR_MBX                    = 00000010
UCB$L_XM_BASEMAP                  0000011C          XM$M_CHR_MOP                    = 00000001
UCB$L_XM_BASETAB                  00000118          XM$M_CHR_SLAVE                  = 00000008
UCB$L_XM_DRVCNT                   00000120          XM$M_ERR_FATAL                  = 00010000
UCB$L_XM_LSTCSR                   0000014C          XM$M_STS_ACTIVE                 = 00000800
UCB$L_XM_LSTPRT                   00000148          XM$V_CHR_LOOPB                  = 00000001
UCB$L_XM_PID                      00000110          XM$V_CHR_MBX                    = 00000004
UCB$L_XM_RCV_MAP                  000000D0          XM$V_STS_ACTIVE                 = 0000000B
UCB$L_XM_XMT_MAP                  000000EC          XM$V_STS_BUFFAIL                = 0000000C
UCB$M_INT                       = 00000002          XMTFDT                            000000D0 R      03
UCB$M_ONLINE                    = 00000010          XMT_START                         0000010E R      03
UCB$M_POWER                     = 00000020          XMT_START_ALT                     00000113 R      03
UCB$M_TIM                       = 00000001          XM_END                            00000F36 R      03
UCB$M_TIMOUT                    = 00000040          XM_E_M_LOST                     = 00000010
UCB$M_XM_FORK_PEND              = 00002000          XM_E_M_MOP                      = 00000008
UCB$M_XM_FORK_XMT               = 00000001          XM_E_M_NONEXMEM                 = 00000100
UCB$M_XM_INITED                 = 00000008          XM_E_M_POWER                    = 00000400
UCB$M_XM_LOSTERR                = 00001000          XM_E_M_PROCERR                  = 00000200
UCB$M_XM_NOTIF                  = 00000800          XM_E_M_START                    = 00000080
UCB$Q_XM_PORT                     000000A0          XM_E_M_TIMEOUT                  = 00000800
UCB$Q_XM_POST                     000000B8          XM_E_V_POWER                    = 0000000A
UCB$Q_XM_QUEUES                   00000090          XM_E_V_PROCERR                  = 00000009
```

```
                                                              N 5
XMDRIVER                          - VAX/VMS DMC11/DMR11 Device Driver      16-SEP-1984 00:26:05  VAX/VMS Macro V04-00          Page  59
Symbol table                                                                5-SEP-1984 00:20:43  [DRIVER.SRC]XMDRIVER.MAR;1            (27)

XM_E_V_TIMEOUT                    = 0000000B
XM_I_CSR                            00000000
XM_I_M_IEI                        = 00000040
XM_I_M_LOOPB                      = 00000800
XM_I_M_MCLR                       = 00004000
XM_I_M_RCV                        = 00000004
XM_I_M_RDI                        = 00000080
XM_I_M_ROMI                       = 00000200
XM_I_M_ROMO                       = 00000400
XM_I_M_RQI                        = 00000020
XM_I_M_RUN                        = 00008000
XM_I_M_STEPUP                     = 00000100
XM_I_V_RQI                        = 00000005
XM_O_CSR                            00000002
XM_O_M_IEO                        = 00000040
XM_O_M_RDO                        = 00000080
XM_O_V_RCV                        = 00000002
XM_O_V_TYPE                       = 00000000
XM_PORT                             00000004
XM_UCODE                            00000006

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

PSECT name                        Allocation           PSECT No.  Attributes
----------                        ----------           ---------  ----------
.  ABS  .                         00000000  (     0.)  00 (  0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                             00000152  (   338.)  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                     00000F36  (  3894.)  03 (  3.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC LONG

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

Phase                       Page faults    CPU Time       Elapsed Time
-----                       -----------    --------       ------------
Initialization                       29    00:00:00.04    00:00:01.11
Command processing                  120    00:00:00.44    00:00:05.02
Pass 1                              828    00:00:26.80    00:01:32.21
Symbol table sort                     0    00:00:03.68    00:00:12.70
Pass 2                              505    00:00:06.16    00:00:24.08
Symbol table output                   1    00:00:00.23    00:00:02.57
Psect synopsis output                 0    00:00:00.04    00:00:00.51
Cross-reference output                0    00:00:00.00    00:00:00.00
Assembler run totals               1485    00:00:37.40    00:02:18.21

The working set limit was 2400 pages.
223463 bytes (437 pages) of virtual memory were used to buffer the intermediate code.
There were 190 pages of symbol table space allocated to hold 3446 non-local and 183 local symbols.
2525 source lines were read in Pass 1, producing 27 object records in Pass 2.
59 pages of virtual memory were used to define 55 macros.
```

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

Macro library name                Macros defined
-------------------               --------------
_$255$DUA28:[SHRLIB]NMALIBRY.MLB;1              1
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                 34
_$255$DUA28:[SYSLIB]STARLET.MLB;2              11
TOTALS (all libraries)                         46

3627 GETS were required to define 46 macros.

There were no errors, warnings or information messages.

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

0121 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY