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

FILEID**XIDRIVER

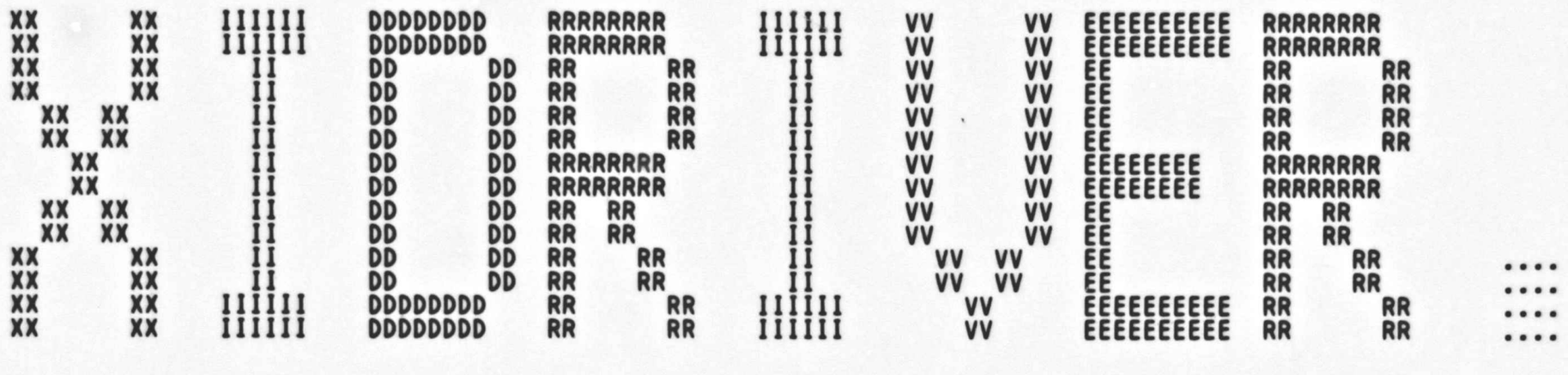


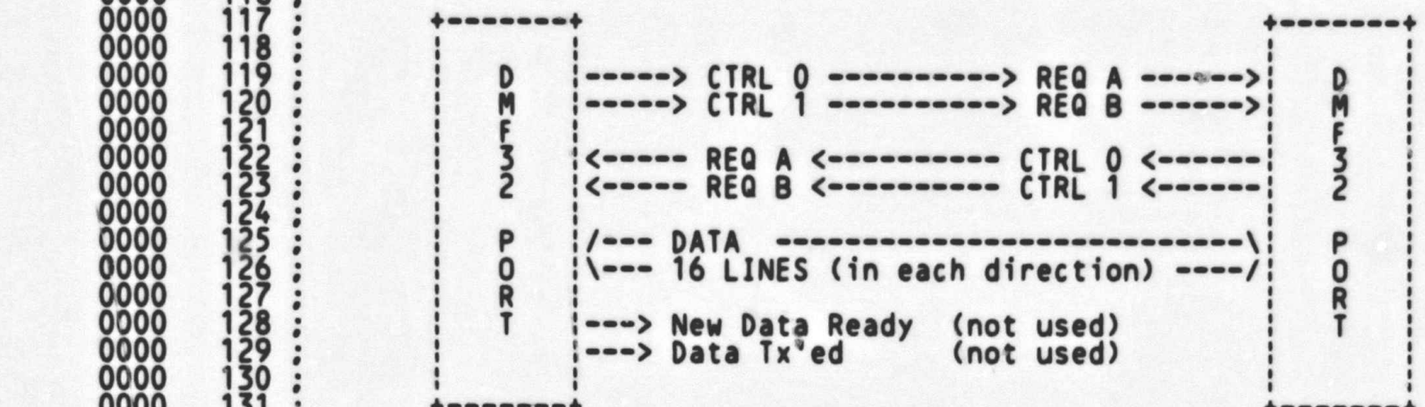
Table of contents

```
0000     1          .TITLE  XIDRIVER - VAX/VMS DMF32 PARALLEL PORT DRIVER
0000     2          .IDENT  'V04-001'
0000     3 ;
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26 ;
0000    27 ;++
0000    28 ;
0000    29 ; FACILITY:
0000    30 ;
0000    31 ;       VAX/VMS Executive, I/O Drivers
0000    32 ;
0000    33 ; ABSTRACT:
0000    34 ;
0000    35 ;       This driver is an example driver for the DMF32 parallel port.
0000    36 ;       This driver implements the DR11C compatibility mode on the device.
0000    37 ;       It does not implement the silo or DMA options, but serves as a
0000    38 ;       template to which such features could be added.
0000    39 ;
0000    40 ;       This module contains the DMF32 PARALLEL PORT driver:
0000    41 ;
0000    42 ;               Tables for loading and dispatching
0000    43 ;               Controller initialization routine
0000    44 ;               FDT routine
0000    45 ;               The start I/O routine
0000    46 ;               The interrupt service routine
0000    47 ;               Device specific Cancel I/O
0000    48 ;
0000    49 ; ENVIRONMENT:
0000    50 ;
0000    51 ;       Kernal Mode, Non-paged
0000    52 ;
0000    53 ; AUTHOR:
0000    54 ;
0000    55 ;       Jake VanNoy     January 1982
0000    56 ;
0000    57 ; MODIFIED BY:
```

```
0000     58 ;
0000     59 ;       V04-001 JLV0396         Jake VanNoy             6-SEP-1984
0000     60 ;               Add AVL to DEVCHAR.
0000     61 ;
0000     62 ;       V03-005 JLV0385         Jake VanNoy             23-JUL-1984
0000     63 ;               Add DPT$M_SVP to DPT.
0000     64 ;
0000     65 ;       V03-004 JLV0341         Jake VanNoy             28-MAR-1984
0000     66 ;               Correct Device IPL.
0000     67 ;
0000     68 ;       V03-003 WHM0002         Bill Matthews           16-Feb-1984
0000     69 ;               Second part of change for edit WHM0001.
0000     70 ;
0000     71 ;       V03-002 WHM0001         Bill Matthews           19-Dec-1983
0000     72 ;               Added code to support new IDB fields IDB$B_COMBO_VECTOR
0000     73 ;               and IDB$B_COMBO_CSR_OFFSET for determining the main CSR
0000     74 ;               address and loading the soft vector for the combo device.
0000     75 ;
0000     76 ;       V03-001 KDM0002         Kathleen D. Morse       28-Jun-1982
0000     77 ;               Added $DCDEF and $DYNDEF.
0000     78 ;
0000     79 ;--
```

```
0000     81          .SBTTL  Description of Interface
0000     82 ;++
0000     83 ;
0000     84 ; The DMF32 Parallel Port interface is a 16 bit parallel port for interfacing
0000     85 ; to a user device. It includes a DR11C compatibility mode (used for word
0000     86 ; mode within this driver), a silo (buffered) mode (not implemented by this
0000     87 ; driver), and a DMA mode (also not implemented by this driver). The interface
0000     88 ; looks like the following:
0000     89 ;
0000     90 ;
0000     91 ;          +--------+                                  +--------+
0000     92 ;          |        |                                  |        |
0000     93 ;          |   D    |---> CTRL 0 ---------->|   U    |
0000     94 ;          |   M    |---> CTRL 1 ---------->|   S    |
0000     95 ;          |   F    |                       |   E    |
0000     96 ;          |   3    |<--- REQ A <-----------|   R    |
0000     97 ;          |   2    |<--- REQ B <-----------|        |
0000     98 ;          |        |                       |   D    |
0000     99 ;          |   P    |/--- DATA  -----------\|   E    |
0000    100 ;          |   O    |\--- 16 LINES --------/|   V    |
0000    101 ;          |   R    |                       |   I    |
0000    102 ;          |   T    |---> New Data Ready -->|   C    |  (pulsed on write to OUTBUF)
0000    103 ;          |        |---> Data Tx'ed ------>|   E    |  (pulsed on read from INBUF)
0000    104 ;          |        |                       |        |
0000    105 ;          +--------+                                  +--------+
0000    106 ;
0000    107 ;
0000    108 ;--
```

```
0000   110 ;++
0000   111 ;
0000   112 ; This driver may be tested using the following configuration of two DMF32's:
0000   113 ; The control lines (CTRL 0 and 1) should be tied into request lines (REQ A
0000   114 ; and B) on the other device. Setting CTRL 0 on the first device causes an
0000   115 ; interrupt on REQ A on the second device (provided interrupt enable A is set).
0000   116 ;
0000   117 ;
0000   118 ;
0000   119 ;
0000   120 ;
0000   121 ;
0000   122 ;
0000   123 ;
0000   124 ;
0000   125 ;
0000   126 ;
0000   127 ;
0000   128 ;
0000   129 ;
0000   130 ;
0000   131 ;
0000   132 ;
0000   133 ;--
```



```
0000   135 .SBTTL  Documentation on interface
0000   136 ;++
0000   137 ;
0000   138 ; The DMF32 parallel port exchanges one 16-bit word at a time.  A single
0000   139 ; QIO request transfers a buffer of data, with an interrupt requested for
0000   140 ; each word.
0000   141 ;
0000   142 ; For each buffer of data transferred, the DMF32 parallel port allows for
0000   143 ; the exchange of additional bits of information: the Control and Status
0000   144 ; Register (CSR) function (CTRL) and status (REQUEST) bits. These bits are
0000   145 ; accessible to an application process through the device driver QIO
0000   146 ; interface.  The CTRL bits are labeled CTRL 0 and CTRL 1. The REQUEST
0000   147 ; bits are labeled REQUEST A and REQUEST B.
0000   148 ;
0000   149 ; The user device interfaced to the DMF32 parallel port interprets the
0000   150 ; value of the two CTRL bits.  The QIO request that initiates the transfer
0000   151 ; specifies the IO$M_SETFNCT modifer to indicate a change in the value
0000   152 ; for the CTRL bits. The P4 argument of the request specifies this value.
0000   153 ; P4 bits 0 and 1 correspond to CTRL bits 0 and 1 respectively. Bits 2
0000   154 ; through 31 are not used. If required, the CTRL bits must be set for each
0000   155 ; request. The CTRL bits set in the CSR are passed directly to the user
0000   156 ; device.
0000   157 ;
0000   158 ; The device class for the DMF32 parallel port is DC$_REALTIME and the
0000   159 ; device type is DT$_XI_DR11C.  The DMF32 parallel port driver does not
0000   160 ; use the default buffer size field.  The value of this field is set to
0000   161 ; 65,535. This driver defines no device-dependent characteristics.
0000   162 ;
0000   163 ; The DMF32 parallel port can perform logical, virtual, and physical I/O
0000   164 ; operations. The basic I/O functions are read, write, set mode, and set
0000   165 ; characteristics.
0000   166 ;
0000   167 ; +-----------------------+--------------+--------------------+
0000   168 ; !  Function Code and    !   Function   !       Function     !
0000   169 ; !      Arguments        !   Modifiers  !                    !
0000   170 ; +-----------------------+--------------+--------------------+
0000   171 ; !                       !              !                    !
0000   172 ; !  IO$_READLBLK P1,P2,- !  IO$M_SETFNCT !  Read block       !
0000   173 ; !            P3,P4      !  IO$M_RESET  !                    !
0000   174 ; !                       !  IO$M_TIMED  !                    !
0000   175 ; !                       !              !                    !
0000   176 ; !  IO$_WRITELBLK P1,P2,- !  IO$M_SETFNCT !Write logical block!
0000   177 ; !            P3,P4      !  IO$M_RESET  !                    !
0000   178 ; !                       !  IO$M_TIMED  !                    !
0000   179 ; !                       !              !                    !
0000   180 ; !  IO$_SETMODE P1,P3     !  IO$M_ATTNAST !Set PORT charact-  !
0000   181 ; !                       !              !eristics for subse- !
0000   182 ; !                       !              ! quent operations   !
0000   183 ; !                       !              !                    !
0000   184 ; !  IO$_SETCHAR P1,P3     !  IO$M_ATTNAST !Set PORT charact-  !
0000   185 ; !                       !              !eristics for subse- !
0000   186 ; !                       !              ! quent operations   !
0000   187 ; !                       !              !                    !
0000   188 ; +-----------------------+--------------+--------------------+
0000   189 ;
0000   190 ; Not in above table are functions IO$_READPBLK, IO$_READVBLK, WRITEPBLK
0000   191 ; and WRITELBLK. There is no functional difference in these operations.
```

```
0000   192 ; Although the DMF32 parallel port does not differentiate between logical,
0000   193 ; virtual, and physical I/O functions (all are treated identically), the
0000   194 ; user must have the required privilege to issue a request.
0000   195 ;
0000   196 ; The function-dependent arguments for the read and write function codes are:
0000   197 ;
0000   198 ; o      P1 -- the starting virtual address of the buffer that
0000   199 ;         is to receive data in the case of a read operation; or, in
0000   200 ;         the case of a write operation, the virtual address of the
0000   201 ;         buffer that is to send data to the DMF32 parallel port.
0000   202 ;         Modify access to the buffer, rather than read or write
0000   203 ;         access, is checked for all block mode read and write
0000   204 ;         requests.
0000   205 ;
0000   206 ; o      P2 -- the size of the data buffer in bytes, that is, the
0000   207 ;         transfer count.  Since the DMF32 parallel port performs
0000   208 ;         word transfers, the transfer count must be an even value.
0000   209 ;
0000   210 ; o      P3 -- the timeout period for this request (in seconds).
0000   211 ;         The value specified must be equal to or greater than 2.
0000   212 ;         IO$M_TIMED must be specified.  The default timeout value for each
0000   213 ;         request is 10 seconds.
0000   214 ;
0000   215 ; o      P4 -- the value of the DMF32 parallel port Command and Status
0000   216 ;         Register (CSR) function (CTRL) bits to be set.  If
0000   217 ;         IO$M_SETFNCT is specified, the low-order three bits of P4
0000   218 ;         (2:0) are written to CSR CTRL bits 1:0 (respectively) at the
0000   219 ;         time of transfer.
0000   220 ;
0000   221 ; The transfer count specified by the P2 argument must be an even number
0000   222 ; of bytes.  If an odd number or more than 65534 bytes is specifed, an
0000   223 ; error (SS$_BADPARAM) is returned in the I/O status block (IOSB). If the
0000   224 ; transfer count is 0, the driver will transfer no data.  However, if
0000   225 ; IO$M_SETFNCT is specified and P2 is 0, the driver will set the CTRL bits
0000   226 ; in the DMF32 parallel port CSR, and return the current CSR status bit
0000   227 ; values in the IOSB.
0000   228 ;
0000   229 ; The read and write QIO functions can take three function modifiers:
0000   230 ;
0000   231 ; o      IO$M_SETFNCT - set the function (CTRL) bits in the DMF32 parallel
0000   232 ;         port CSR before the data transfer is initiated.  The
0000   233 ;         low-order two bits of the P4 argument specify the CTRL
0000   234 ;         bits.  The user device that interfaces the DMF32 PARALLEL
0000   235 ;         PORT receives the CTRL bits directly and their value is
0000   236 ;         interpreted entirely by the device.
0000   237 ;
0000   238 ; If an unsolicited interrupt is received from the DMF32 parallel port, no
0000   239 ; read or write request is posted, and the next request is for a word mode
0000   240 ; read, the driver will return the word read from the DMF32 parallel port
0000   241 ; INBUF and store it in the first word of the user's buffer. In this case
0000   242 ; the driver does not wait for an interrupt.
0000   243 ;
0000   244 ; o      IO$M_TIMED - set the device timeout interval for the data
0000   245 ;         transfer request. The P3 argument specifies the timeout
0000   246 ;         interval value in seconds.  For consistent results, this
0000   247 ;         value must be equal to or greater than 2.
0000   248 ;
```

```
0000   249 ;  o     IO$M_RESET - perform a device reset to the DMF32 parallel port before
0000   250 ;        any I/O operation is initiated. This function does not
0000   251 ;        affect any other device on the system or on the DMF32.
0000   252 ;
0000   253 ;  The set mode and characteristic function codes are:
0000   254 ;
0000   255 ;  o     IO$_SETMODE
0000   256 ;
0000   257 ;  o     IO$_SETCHAR
0000   258 ;
0000   259 ;  These functions take the following device/function-dependent arguments:
0000   260 ;
0000   261 ;  o     P1 - the virtual address of a quadword characteristics buffer.  If
0000   262 ;        the function modifer IO$M_ATTNAST is specified, P1 is the
0000   263 ;        address the AST service routine. In this case, if P1 is 0,
0000   264 ;        all attention ASTs are disabled.
0000   265 ;
0000   266 ;  o     P3 - the access mode to deliver the AST (maximized with the
0000   267 ;        requestor's access mode).  If IO$M_ATTNAST is not specified, P3
0000   268 ;        is ignored.
0000   269 ;
0000   270 ;        Figure 3-4 shows the quadword P1 characteristics buffer for
0000   271 ;        IO$_SETMODE and IO$_SETCHAR.
0000   272 ;
0000   273 ;        31                    16 15         8 7          0
0000   274 ;        +----------------------+------------+------------+
0000   275 ;        |                      |            |            |
0000   276 ;        |       not used       |    type    |    class   |
0000   277 ;        |                      |            |            |
0000   278 ;        +----------------------+------------+------------+
0000   279 ;        |                                                |
0000   280 ;        |                device characteristics          |
0000   281 ;        |                                                |
0000   282 ;        +------------------------------------------------+
0000   283 ;
0000   284 ;  The IO$_SETMODE and IO$_SETCHAR function codes can take the following function
0000   285 ;  modifier:
0000   286 ;
0000   287 ;  o     IO$M_ATTNAST - enable attention AST
0000   288 ;
0000   289 ;  This function modifier allows the user process to queue an attention AST
0000   290 ;  for delivery when an asynchronous or unsolicited condition is detected
0000   291 ;  by the DMF32 parallel port driver.  Unlike ASTs for other QIO functions,
0000   292 ;  use of this function modifier does not increment the I/O count for the
0000   293 ;  requesting process or lock pages in memory for I/O buffers. There must
0000   294 ;  be an AST quota for each AST.
0000   295 ;
0000   296 ;  Attention ASTs are delivered under the following conditions:
0000   297 ;
0000   298 ;  o     An unsolicited interrupt from the DMF32 parallel port occurs.
0000   299 ;
0000   300 ;  o     An attention AST is queued and a previous unsolicited interrupt
0000   301 ;        has not been acknowledged.
0000   302 ;
0000   303 ;  The $CANCEL system service is used to flush attention ASTs for a specific
0000   304 ;  channel.
0000   305 ;
```

```
0000    306 ; IO$_SETMODE!IO$M_ATTNAST and IO$_SETCHAR!IO$M_ATTNAST are one-time AST
0000    307 ; enables; they must be explicitly re-enabled once the AST has been
0000    308 ; delivered if the user desires notification of the next interrupt.  Use
0000    309 ; of this function modifier does not update the device characteristics.
0000    310 ;
0000    311 ; After the AST is delivered, the QIO astprm parameter contains the
0000    312 ; contents of the DMF32 parallel port CSR in the low two bytes and the
0000    313 ; value read from the DMF32 parallel port INBUF in the high two bytes.
0000    314 ;
0000    315 ; On completion of each read or write request, the I/O status block
0000    316 ; is filled with system and DMF32 parallel port status information.
0000    317 ;
0000    318 ;                                +2                          IOSB
0000    319 ;                 +-------------------------+-------------------------+
0000    320 ;                 |                         |                         |
0000    321 ;                 |       byte count        |         status          |
0000    322 ;                 |                         |                         |
0000    323 ;                 +-------------------------+-------------------------+
0000    324 ;                 |                         |                         |
0000    325 ;                 |         unused          |        PORT CSR         |
0000    326 ;                 |                         |                         |
0000    327 ;                 +-------------------------+-------------------------+
0000    328 ;
0000    329 ;
0000    330 ;--
0000    331
```

```
                                        .SBTTL  External and local symbol definitions

                                ; External symbols

                                        $ACBDEF                         ; AST control block
                                        $CRBDEF                         ; Channel request block
                                        $DCDEF                          ; Device types
                                        $DDBDEF                         ; Device data block
                                        $DPTDEF                         ; Driver prolog table
                                        $DYNDEF                         ; Dynamic data structure types
                                        $IDBDEF                         ; Interrupt data block
                                        $IODEF                          ; I/O function codes
                                        $IPLDEF                         ; Hardware IPL definitions
                                        $IRPDEF                         ; I/O request packet
                                        $PRDEF                          ; Internal processor registers
                                        $PRIDEF                         ; Scheduler priority increments
                                        $SSDEF                          ; System messages
                                        $UCBDEF                         ; Unit control block
                                        $VECDEF                         ; Interrupt vector block

                                ; Local symbols

                                ; Argument list (AP) offsets for device-dependent QIO parameters

00000000                        P1      = 0                             ; First QIO parameter
00000004                        P2      = 4                             ; Second QIO parameter
00000008                        P3      = 8                             ; Third QIO parameter
0000000C                        P4      = 12                            ; Fourth QIO parameter
00000010                        P5      = 16                            ; Fifth QIO parameter
00000014                        P6      = 20                            ; Sixth QIO parameter

                                ; Other constants

0000000A                        XI_DEF_TIMEOUT  = 10                    ; 10 second default device timeout
0000FFFF                        XI_DEF_BUFSIZ   = 65535                 ; Default buffer size
00000002                        XI$K_VEC_OFFSET = 2                     ; Vector offset

                                ;
                                ; Macros
                                ;

                                ;
                                ; The SETCTRL macro sets the CTRL 0 and 1 lines as they have been
                                ; specified in P4 in a read or write QIO. They are cleared and a wait
                                ; occurs before being set. This is because testing for this example
                                ; driver was done between two DMF32's in word mode, and the delay is so the
                                ; microcode on the DMF32 can see the control line changes.
                                ;

                                .MACRO  SETCTRL
                                        BICW    #XI_CSR$M_CTRL0!XI_CSR$M_CTRL1,XI_CSR(R4)
                                        CLRL    -(SP)
                                        TIMEWAIT -
                                                TIME = #2,-
                                                BITVAL = #1,-
                                                SOURCE = (SP),-
```

```
0000   390                      CONTEXT = L,-
0000   391                      SENSE = .TRUE.
0000   392              TSTL    (SP)+
0000   393              BISW    IRP$L_SEGVBN(R3),XI_CSR(R4)
0000   394 .ENDM       SETCTRL
```

```
                     0000   396
                     0000   397 ; UCB_XI definitions that follow the standard UCB fields
                     0000   398
                     0000   399          $DEFINI UCB
                     0000   400
000000A0             0000   401          .=UCB$L_DPC+4   ;
                     00A0   402
                     00A0   403 $DEF     UCB$L_XI_ATTN
000000A4             00A0   404                          .BLKL   1       ; Attention AST queue
                     00A4   405
                     00A4   406 $DEF     UCB$L_XI_DPR
000000A8             00A4   407                          .BLKL   1       ; Word count?
                     00A8   408
                     00A8   409 $DEF     UCB$W_XI_INBUF
000000AA             00A8   410                          .BLKW   1       ; Input buffer temporary
                     00AA   411
                     00AA   412 $DEF     UCB$W_XI_CSR
000000AC             00AA   413                          .BLKW   1       ; CSR temporary
                     00AC   414
                     00AC   415 ; Bit positions for device-dependent status field in UCB (UCB$W_DEVSTS)
                     00AC   416
                     00AC   417          $VIELD  UCB,0,<-                ; UCB device specific bit definitions
                     00AC   418                  <ATTNAST,,M>,-
                     00AC   419                  <UNEXPT,,M>-
                     00AC   420                  >
                     00AC   421
000000AC             00AC   422 UCB$K_SIZE=.
                     00AC   423          $DEFEND UCB
```

```
                                                J 15
00000060   0000   452 XI_CSR$M_IEAB   = <XI_CSR$M_INTENB_A>!<XI_CSR$M_INTENB_B> ; Interrupt enable mask
00000002   0000   454                          .BLKW   1
           0002   455 $DEF    XI_OUTBUF                       ; Output buffer Register
00000004   0002   456                          .BLKW   1
           0004   457
           0004   458 ; Note that XI_INBUF and XI_MISC are at the same offset
           0004   459
           0004   460 $DEF    XI_INBUF                        ; Input buffer Register (when read)
           0004   461 $DEF    XI_MISC                         ; Miscellaneous Register (when written)
           0004   462
           0004   463 ; Bit positions for miscellaneous register
           0004   464
           0004   465         $VIELD  XI_MISC,0,<-            ; Miscellaneous register
           0004   466                 <MODE,4,M>,-            ; Mode
           0004   467                 <,10,M>,-               ; unused
           0004   468                 <SECBUF,,M>-            ; Secondary Buffer Address, Bit 17
           0004   469                 <PRIBUF,,M>-            ; Primary Buffer Address, Bit 17
           0004   470         >
00000006   0004   471                          .BLKW   1
           0006   472 $DEF    XI_IND                          ; Indirect Register
00000008   0006   473                          .BLKW   1
           0008   474
           0008   475         $DEFEND XI                      ; End of PORT CSR definitions
```

```
                        .SBTTL  Device Driver Tables

; Driver prologue table

        DPTAB   -                                       ; DPT-creation macro
                END=XI_END,-                            ; End of driver label
                ADAPTER=UBA,-                           ; Adapter type
                FLAGS=DPT$M_SVP,-                       ; Allocate system page table
                UCBSIZE=UCB$K_SIZE,-                    ; UCB size
                NAME=XIDRIVER                           ; Driver name

        DPT_STORE INIT                                  ; Start of load
                                                        ; initialization table
        DPT_STORE UCB,UCB$B_FIPL,B,8                    ; Device fork IPL
        DPT_STORE UCB,UCB$B_DIPL,B,21                   ; Device interrupt IPL
        DPT_STORE UCB,UCB$L_DEVCHAR,L,<-                ; Device characteristics
                DEV$M_AVL!-                             ; Available
                DEV$M_RTM!-                             ; Real Time device
                DEV$M_IDV!-                             ;   input device
                DEV$M_ODV>                              ;   output device
        DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_REALTIME     ; Device class
        DPT_STORE UCB,UCB$B_DEVTYPE,B,DT$_XI_DR11C      ; Device Type
        DPT_STORE UCB,UCB$W_DEVBUFSIZ,W,-               ; Default buffer size
                XI_DEF_BUFSIZ
        DPT_STORE REINIT                                ; Start of reload
                                                        ; initialization table
        DPT_STORE DDB,DDB$L_DDT,D,XI$DDT                ; Address of DDT
        DPT_STORE CRB,CRB$L_INTD+4,D,-                  ; Address of interrupt
                XI_INTERRUPT                            ; service routine
        DPT_STORE CRB,CRB$L_INTD2+4,D,-                 ; Address of interrupt
                XI_INTERRUPT                            ; service routine
        DPT_STORE CRB,CRB$L_INTD+VEC$L_INITIAL,-        ; Address of controller
                D,XI_CONTROL_INIT                       ; initialization routine
        DPT_STORE END                                   ; End of initialization
                                                        ; tables

; Driver dispatch table

        DDTAB   -                                       ; DDT-creation macro
                DEVNAM=XI,-                             ; Name of device
                START=XI_STARTIO,-                      ; Start I/O routine
                FUNCTB=XI_FUNCTABLE,-                   ; FDT address
                CANCEL=XI_CANCEL                        ; Cancel I/O routine
```

```
0038   521 ;
0038   522 ; Function dispatch table
0038   523 ;
0038   524 XI_FUNCTABLE:                                        ; FDT for driver
0038   525
0038   526         ; Valid I/O functions
0038   527
0038   528         FUNCTAB ,-
0038   529                 <READPBLK,READLBLK,READVBLK, -
0038   530                 WRITEPBLK,WRITELBLK,WRITEVBLK,-
0038   531                 SETMODE,SETCHAR,SENSEMODE,SENSECHAR>
0040   532
0040   533         FUNCTAB ,                               ; No buffered functions
0048   534
0048   535         FUNCTAB XI_READ_WRITE,-                 ; Device-specific FDT
0048   536                 <READPBLK,READLBLK,READVBLK, -
0048   537                 WRITEPBLK,WRITELBLK,WRITEVBLK>
0054   538         FUNCTAB +EXE$READ,<READPBLK,READLBLK,READVBLK>
0060   539         FUNCTAB +EXE$WRITE,<WRITEPBLK,WRITELBLK,WRITEVBLK>
006C   540         FUNCTAB XI_SETMODE,<SETMODE,SETCHAR>
0078   541         FUNCTAB +EXE$SENSEMODE,<SENSEMODE,SENSECHAR>
```

```
                     0084   543          .SBTTL  XI_CONTROL_INIT, Controller initialization
                     0084   544
                     0084   545 ;++
                     0084   546 ; XI_CONTROL_INIT, Called when driver is loaded, system is booted, or
                     0084   547 ; power failure recovery.
                     0084   548 ;
                     0084   549 ; Functional Description:
                     0084   550 ;
                     0084   551 ;       1) Allocates the direct data path permanently
                     0084   552 ;       2) Assigns the controller data channel permanently
                     0084   553 ;       3) Clears the Control and Status Register
                     0084   554 ;       4) If power recovery, requests device time-out
                     0084   555 ;
                     0084   556 ; Inputs:
                     0084   557 ;
                     0084   558 ;       R4 = address of CSR
                     0084   559 ;       R5 = address of IDB
                     0084   560 ;       R6 = address of DDB
                     0084   561 ;       R8 = address of CRB
                     0084   562 ;
                     0084   563 ; Outputs:
                     0084   564 ;
                     0084   565 ;       VEC$V_PATHLOCK bit set in CRB$L_INTD+VEC$B_DATAPATH
                     0084   566 ;       UCB address placed into IDB$L_OWNER
                     0084   567 ;
                     0084   568 ;
                     0084   569 ;--
                     0084   570
                     0084   571 XI_CONTROL_INIT:
                     0084   572
     50    18 A5    D0   0084   573          MOVL    IDB$L_UCBLST(R5),R0      ; Address of UCB
     04 A5    50    D0   0088   574          MOVL    R0,IDB$L_OWNER(R5)       ; Make permanent controller owner
     64 A0    10    A8   008C   575          BISW    #UCB$M_ONLINE, -
                           0090   576                  UCB$W_STS(R0)            ; Set device status "on-line"
                           0090   577
                           0090   578 ; If powerfail has occured and device was active, force device time-out.
                           0090   579 ; The user can set his own time-out interval for each request.  Time-
                           0090   580 ; out is forced so a very long time-out period will be short circuited.
                           0090   581
05 64 A0    05    E0   0090   582          BBS     #UCB$V_POWER, -
                           0095   583                  UCB$W_STS(R0),10$        ; Branch if powerfail
37 A8    80 8F    88   0095   584          BISB    #VEC$M_PATHLOCK, -
                           009A   585                  CRB$L_INTD+VEC$B_DATAPATH(R8) ; Permanently allocate direct datapath
                           009A   586 10$:
                           009A   587
     50    0F A5    98   009A   588          CVTBL   IDB$B_COMBO_CSR_OFFSET(R5),R0   ; GET OFFSET TO MAIN DMF CSR
           10 A5    83   009E   589          SUBB3   IDB$B_COMBO_VECTOR_OFFSET(R5),- ; CALCULATE AND LOAD THE
   6440    0B A5         00A1   590                  IDB$B_VECTOR(R5),(R4)[R0]       ;       VECTOR ADDRESS
            030D    30   00A5   591          BSBW    XI_DEV_RESET             ; Reset port
                    05   00A8   592          RSB                              ; Done
```

```
                                00A9   594          .SBTTL  XI_READ_WRITE,   Data transfer FDT
                                00A9   595
                                00A9   596 ;++
                                00A9   597 ; XI_READ_WRITE, FDT for READLBLK,READVBLK,READPBLK,WRITELBLK,WRITEVBLK,
                                00A9   598 ;                        WRITEPBLK
                                00A9   599 ;
                                00A9   600 ; Functional description:
                                00A9   601 ;
                                00A9   602 ;       1) Rejects QUEUE I/O's with odd transfer count
                                00A9   603 ;       2) Rejects QUEUE I/O's for DMA request to UBA Direct Data
                                00A9   604 ;          PATH on odd byte boundary
                                00A9   605 ;       3) Stores request time-out count specified in P3 into IRP
                                00A9   606 ;       4) Stores CTRL bits specified in P4 into IRP
                                00A9   607 ;       6) Checks block mode transfers for memory modify access
                                00A9   608 ;
                                00A9   609 ; Inputs:
                                00A9   610 ;
                                00A9   611 ;       R3 = Address of IRP
                                00A9   612 ;       R4 = Address of PCB
                                00A9   613 ;       R5 = Address of UCB
                                00A9   614 ;       R6 = Address of CCB
                                00A9   615 ;       AP = Address of P1
                                00A9   616 ;               P1 = Buffer Address
                                00A9   617 ;               P2 = Buffer size in bytes
                                00A9   618 ;               P3 = Request time-out period (conditional on IO$M_TIMED)
                                00A9   619 ;               P4 = Value for CSR CTRL bits (conditional on IO$M_SETFNCT)
                                00A9   620 ;               P5 = 0 for Request A, 1 for Request B (DMA)
                                00A9   621 ;
                                00A9   622 ; Outputs:
                                00A9   623 ;
                                00A9   624 ;       R0 = Error status if odd transfer count
                                00A9   625 ;       IRP$L_MEDIA = Time-out count for this request
                                00A9   626 ;       IRP$L_SEGVBN = CTRL bits for PORT CSR
                                00A9   627 ;
                                00A9   628 ;--
                                00A9   629
                                00A9   630 XI_READ_WRITE:
                                00A9   631
      09 04 AC    E9            00A9   632          BLBC    P2(AP),20$                      ; Branch if transfer count even
        50    14    3C          00AD   633 10$:     MOVZWL  #SS$_BADPARAM,R0                ; Set error status code
 00000000'GF    17               00B0   634          JMP     G^EXE$ABORTIO                   ; Abort request
                                 00B6   635
        51    20 A3    3C       00B6   636 20$:     MOVZWL  IRP$W_FUNC(R3),R1               ; Fetch I/O Function code
   38 A3    08 AC    D0          00BA   637          MOVL    P3(AP),IRP$L_MEDIA(R3)          ; Set request specific time-out count
      04 51    07    E0          00BF   638          BBS     #IO$V_TIMED,R1,30$              ; Branch if time-out specified
      38 A3    0A    3C          00C3   639          MOVZWL  #XI_DEF_TIMEOUT, -
                                 00C7   640                  IRP$L_MEDIA(R3)                 ; Else set default timeout value
OC AC    02    00    EF          00C7   641 30$:     EXTZV   #0,#2,P4(AP),-
          48 A3                  00CC   642                  IRP$L_SEGVBN(R3)                ; Get value for CTRL bits
                    05           00CE   643          RSB                                     ; Return
```

```
                                00CF   645          .SBTTL  XI_SETMODE,      Set Mode, Set Char FDT
                                00CF   646
                                00CF   647 ;++
                                00CF   648 ; XI_SETMODE, FDT routine to process SET MODE and SET CHARACTERISTICS
                                00CF   649 ;
                                00CF   650 ; Functional description:
                                00CF   651 ;
                                00CF   652 ;       If IO$M_ATTNAST modifier is set, queue attention AST for device
                                00CF   653 ;       Else, finish I/O.
                                00CF   654 ;
                                00CF   655 ; Inputs:
                                00CF   656 ;
                                00CF   657 ;       R3 = I/O packet address
                                00CF   658 ;       R4 = PCB address
                                00CF   659 ;       R5 = UCB address
                                00CF   660 ;       R6 = CCB address
                                00CF   661 ;       R7 = Function code
                                00CF   662 ;       AP = QIO Paramater list address
                                00CF   663 ;
                                00CF   664 ; Outputs:
                                00CF   665 ;
                                00CF   666 ;       If IO$M_ATTNAST is specified, queue AST on UCB attention AST list.
                                00CF   667 ;       Else, use exec routine to update device characteristics
                                00CF   668 ;
                                00CF   669 ;--
                                00CF   670
                                00CF   671 XI_SETMODE:
                                00CF   672
            50   20 A3    3C  00CF   673          MOVZWL  IRP$W_FUNC(R3),R0       ; Get entire function code
            28 50    08    E1  00D3   674          BBC     #IO$V_ATTNAST,R0,20$    ; Branch if not an ATTN AST
                                00D7   675
                                00D7   676 ; Attention AST request
                                00D7   677
                0090 8F     BB  00D7   678          PUSHR   #^M<R4,R7>
            57  00A0 C5     9E  00DB   679          MOVAB   UCB$L_XI_ATTN(R5),R7    ; Address of ATTN AST control block list
             00000000'GF    16  00E0   680          JSB     G^COM$SETATTNAST        ; Set up attention AST
                0090 8F     BA  00E6   681          POPR    #^M<R4,R7>
                  18 50     E9  00EA   682          BLBC    R0,30$                  ; Branch if error
              68 A5   01    A8  00ED   683          BISW    #UCB$M_ATTNAST, -
                                00F1   684                  UCB$W_DEVSTS(R5)        ; Flag ATTN AST expected.
        03 68 A5    01    E1  00F1   685          BBC     #UCB$V_UNEXPT, -
                                00F6   686                  UCB$W_DEVSTS(R5),10$    ; Deliver AST if unsolicited interrupt
                    026D    30  00F6   687          BSBW    XI_DE[_ATTNAST
             00000000'GF    17  00F9   688 10$:     JMP     G^EXE$FINISHIO          ; Thats all for now
                                00FF   689
             00000000'GF    17  00FF   690 20$:     JMP     G^EXE$SETCHAR           ; Set device characteristics
                                0105   691
                      51    D4  0105   692 30$:     CLRL    R1                      ; zero R1
             00000000'GF    17  0107   693          JMP     G^EXE$ABORTIO           ; Abort I/O with R0 as status
                                010D   694
```

```
                     010D   696          .SBTTL  XI_START,        Start I/O routines
                     010D   697 ;++
                     010D   698 ; XI_START - Start a data transfer, set characteristics, enable ATTN AST.
                     010D   699 ;
                     010D   700 ; Functional Description:
                     010D   701 ;
                     010D   702 ;       This routine has one major function:
                     010D   703 ;
                     010D   704 ;       1) Start an I/O transfer. The CTRL bits in the port CSR are set.  If
                     010D   705 ;          the transfer count is zero, the STATUS bits in the PORT CSR
                     010D   706 ;          are read and the request completed.
                     010D   707 ;
                     010D   708 ; Inputs:
                     010D   709 ;
                     010D   710 ;       R3 = Address of the I/O request packet
                     010D   711 ;       R5 = Address of the UCB
                     010D   712 ;
                     010D   713 ; Outputs:
                     010D   714 ;
                     010D   715 ;       R0 = final status and number of bytes transferred
                     010D   716 ;       R1 = value of CSR STATUS bits
                     010D   717 ;
                     010D   718 ;--
                     010D   719
                     010D   720 XI_START:
                     010D   721
                     010D   722 ; Retrieve the address of the device CSR
                     010D   723
                     010D   724          ASSUME  IDB$L_CSR EQ 0
          54   24 A5  D0  010D   725          MOVL    UCB$L_CRB(R5),R4        ; Address of CRB
          54   2C B4  D0  0111   726          MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4
                     0115   727                                                  ; Address of CSR
                     0115   728
                     0115   729 ; Fetch the I/O function code
                     0115   730
          51   20 A3  3C  0115   731          MOVZWL  IRP$W_FUNC(R3),R1       ; Get entire function code
        009A C5   51  B0  0119   732          MOVW    R1,UCB$W_FUNC(R5)       ; Save FUNC in UCB
    52  51   06  00  EF  011E   733          EXTZV   #IO$V_FCODE, -
                     0123   734                  #IO$S_FCODE,R1,R2       ; Extract function field
                     0123   735
                     0123   736 ; If subfunction modifier for device reset is set, do one here
                     0123   737
          03 51   0B  E1  0123   738          BBC     S^#IO$V_RESET,R1,40$    ; Branch if not device reset
                028B  30  0127   739          BSBW    XI_DEV_RESET            ; Reset port
                     012A   740
                     012A   741 ; This must be a data transfer function - i.e. READ OR WRITE
                     012A   742 ; Check to see if this is a zero length transfer.
                     012A   743 ; If so, only set CSR CTRL bits and return STATUS from CSR
                     012A   744
             7E A5   B5  012A   745 40$:     TSTW    UCB$W_BCNT(R5)          ; Is transfer count zero?
                50   12  012D   746          BNEQ    100$                    ; No, continue with data transfer
          3C 51   09  E1  012F   747          BBC     S^#IO$V_SETFNCT,R1,60$  ; Set CSR CTRL specified?
                     0133   748          DSBINT                          ; Disable Interrupts
                     0139   749          SETCTRL                         ; Set CTRL bits in CSR
             51   64  3C  0167   750          MOVZWL  XI_CSR(R4),R1           ; Save CSR
                     016A   751          ENBINT                          ; Enable Interrupts
                02   11  016D   752          BRB     70$                     ; Skip clearing of R1
```

```
                                016F   753
                          51    D4  016F   754 60$:     CLRL    R1                          ; Clear R1
                    0060  8F    A8  0171   755 70$:     BISW    #XI_CSR$M_IEAB,-
                          64        0175   756                 XI_CSR(R4)                  ; Enable device interrupts (A & B)
                    50    01    3C  0176   757         MOVZWL  #SS$_NORMAL,R0              ; Set success
                                    0179   758         REQCOM                              ; Request done
                                    017F   759
                                    017F   760 ;
                                    017F   761 ; Do the read or the write
                                    017F   762 ;
                                    017F   763
                                    017F   764 100$:
                    50    7E    A5  3C  017F   765         MOVZWL  UCB$W_BCNT(R5),R0           ; Get byte count
              00A4  C5    50    FF  8F  78  0183   766         ASHL    #-1,R0,UCB$L_WI_DPR(R5) ; Make byte count into word count
                                    018A   767
                                    018A   768         .SBTTL  - word mode tranfer
                                    018A   769 ;++
                                    018A   770 ; WORD MODE -- Process word mode (interrupt per word) transfer
                                    018A   771 ;
                                    018A   772 ; FUNCTIONAL DESCRIPTION:
                                    018A   773 ;
                                    018A   774 ;       Data is transferred one word at a time with an interrupt for each word.
                                    018A   775 ;       The request is handled separately for a write (from memory to port
                                    018A   776 ;       and a read (from port to memory).
                                    018A   777 ;       For a write, data is fetched from memory, loaded into the ODR of the
                                    018A   778 ;       port and the system waits for an interrupt.  For a read, the system
                                    018A   779 ;       waits for a port interrupt and the INBUF is transferred into memory.
                                    018A   780 ;       If the unsolicited interrupt flag is set, the first word is transferred
                                    018A   781 ;       directly into memory withou waiting for an interrupt.
                                    018A   782 ;--
                                    018A   783
                                    018A   784 WORD_MODE:
                                    018A   785
                                    018A   786 ; Dispatch to separate loops on READ or WRITE
                                    018A   787
                                    018A   788 10$:
                    52    0C    91  018A   789         CMPB    #IO$_READPBLK,R2            ; Check for read function
                          7D    13  018D   790         BEQL    WORD_MODE_READ
```

```
                     018F    792 ;++
                     018F    793 ; WORD MODE WRITE -- Write (output) in word mode
                     018F    794 ;
                     018F    795 ; FUNCTIONAL DESCRIPTION:
                     018F    796 ;
                     018F    797 ;       Transfer the requested number of words from user memory to
                     018F    798 ;       the port OUTBUF one word at a time, wait for interrupt for each
                     018F    799 ;       word.
                     018F    800 ;--
                     018F    801
                     018F    802 WORD_MODE_WRITE:
                     018F    803 10$:
             0110  30 018F    804         BSBW    MOVFRUSER                       ; Get two bytes from user buffer
                     0192    805         DSBINT                                  ; Lock out interrupts
                     0198    806                                                 ; Flag interrupt expected
       02 A4  51  B0 0198    807         MOVW    R1,XI_OUTBUF(R4)                ; Move data to port
  64   0060 8F  A8 019C    808         BISW    #XI_CSR$M_IEAB,-
                     01A1    809                 XI_CSR(R4)                      ; Set Interrupt Enable (A & B)
                     01A1    810         SETCTRL                                 ; Clear and set CTRL bits
                     01CF    811
                     01CF    812 ; Wait for interrupt, powerfail, or device time-out
                     01CF    813
                     01CF    814         WFIKPCH XI_TIME_OUTW,IRP$L_MEDIA(R3)
                     01DA    815
                     01DA    816 ; Decrement transfer count, and loop until done
                     01DA    817
                     01DA    818         IOFORK                                  ; Fork to lower IPL
         00A4 C5  B7 01E0    819         DECW    UCB$L_XI_DPR(R5)                ; All words transferred?
              A9  12 01E4    820         BNEQ    10$                             ; No, loop until finished.
                     01E6    821
                     01E6    822 ; Transfer is done, clear interrupt expected flag and FORK
                     01E6    823 ; All words read or written in WORD MODE.  Finish I/O.
                     01E6    824
                     01E6    825 RETURN_STATUS:
                     01E6    826
              50  01  3C 01E6    827         MOVZWL  #SS$_NORMAL,R0                  ; Complete success status
51   00A4 C5  02  A5 01E9    828         MULW3   #2,UCB$L_XI_DPR(R5),R1          ; Calculate actual bytes xfered
  51   7E A5  51  A3 01EF    829         SUBW3   R1,UCB$W_BCNT(R5),R1            ; From requested number of bytes
50   10   10  51  F0 01F4    830         INSV    R1,#16,#16,R0                   ; And place in high word of R0
     51   00AA C5  3C 01F9    831         MOVZWL  UCB$W_XI_CSR(R5),R1             ; Return CSR status
                  AA 01FE    832         BICW    #<XI_CSR$M_CTRL0! -
                     01FF    833                   XI_CSR$M_CTRL1>,-
              64  03 01FF    834                 XI_CSR(R4)                      ; Clear CTRL bits
              0060 8F  A8 0201    835         BISW    #XI_CSR$M_IEAB,-
                  64 0205    836                 XI_CSR(R4)                      ; Enable device interrupts (A & B)
                     0206    837         REQCOM                                  ; Finish request in exec
```

```
                     020C   839 ;++
                     020C   840 ; WORD MODE READ -- Read (input) in word mode
                     020C   841 ;
                     020C   842 ; FUNCTIONAL DESCRIPTION:
                     020C   843 ;
                     020C   844 ;       Transfer the requested number of words from the port INBUF into
                     020C   845 ;       user memory one word at a time, wait for interrupt for each word.
                     020C   846 ;       If the unexpected (unsolicited) interrupt bit is set, transfer the
                     020C   847 ;       first (last received) word to memory without waiting for an
                     020C   848 ;       interrupt.
                     020C   849 ;--
                     020C   850
                     020C   851 WORD_MODE_READ:
                     020C   852         SETIPL  UCB$B_DIPL(R5)          ; Lock out interrupts
                     0210   853
                     0210   854 ; If an unexpected (unsolicited) interrupt has occured, assume it
                     0210   855 ; is for this READ request and return value to user buffer without
                     0210   856 ; waiting for an interrupt.
                     0210   857
4A 68 A5    01   E4 0210   858         BBSC    #UCB$V_UNEXPT, -
                     0215   859                 UCB$W_DEVSTS(R5),20$    ; Branch if unexpected interrupt
                     0215   860         DSBINT
 64    0060 8F    A8 021B   861 10$:    BISW    #XI_CSR$M_IEAB, -
                     0220   862                 XI_CSR(R4)              ; Set Interrupt Enable (A & B)
                     0220   863         SETCTRL                         ; Clear and set CTRL bits
                     024E   864
                     024E   865 ; Wait for interrupt, powerfail, or device time-out
                     024E   866
                     024E   867         WFIKPCH XI_TIME_OUTW,IRP$L_MEDIA(R3)
                     0259   868
                     0259   869 ; Decrement transfer count, and loop until done
                     0259   870
                     0259   871         IOFORK                          ; Fork to lower IPL
                     025F   872 20$:
      0051   30     025F   873         BSBW    MOVTOUSER               ; Store two bytes into user buffer
                     0262   874
                     0262   875 ; Send interrupt back to sender.  Acknowledge we got last word.
                     0262   876
                     0262   877         DSBINT
00A4 C5    B7     0268   878         DECW    UCB$L_XI_DPR(R5)        ; Decrement transfer count
     AD    12     026C   879         BNEQ    10$                     ; Loop until all words transferred
                     026E   880         SETCTRL
                     029C   881         ENBINT
      FF44   31     029F   882         BRW     RETURN_STATUS           ; Finish request in common code
```

```
                        02A2    884 ;
                        02A2    885 ; MOVFRUSER - Routine to fetch two bytes from user buffer.
                        02A2    886 ;
                        02A2    887 ; INPUTS:
                        02A2    888 ;
                        02A2    889 ;       R5 = UCB address
                        02A2    890 ;
                        02A2    891 ; OUTPUTS:
                        02A2    892 ;
                        02A2    893 ;       R1 = Two bytes of data from users buffer
                        02A2    894 ;       Buffer descriptor in UCB is updated.
                        02A2    895 ;
                        02A2    896         .ENABL  LSB
                        02A2    897 MOVFRUSER:
     51    7E    DE     02A2    898         MOVAL   -(SP),R1                ; Address of temporary stack loc
     52    02    9A     02A5    899         MOVZBL  #2,R2                   ; Fetch two bytes
00000000'GF    16     02A8    900         JSB     G^IOC$MOVFRUSER         ; Call exec routine to do the deed
     51    8E    D0     02AE    901         MOVL    (SP)+,R1                ; Retrieve the bytes
           0E    11     02B1    902         BRB     20$                     ; Update UCB buffer pointers
                        02B3    903
                        02B3    904 ;
                        02B3    905 ; MOVTOUSER - Routine to store two bytes into users buffer.
                        02B3    906 ;
                        02B3    907 ; INPUTS:
                        02B3    908 ;
                        02B3    909 ;       R5 = UCB address
                        02B3    910 ;       UCB$W_XI_INBUF(R5) = Location where two bytes are saved
                        02B3    911 ;
                        02B3    912 ; OUTPUTS:
                        02B3    913 ;
                        02B3    914 ;       Two bytes are stored in user buffer and buffer descriptor in
                        02B3    915 ;       UCB is updated.
                        02B3    916 ;
                        02B3    917
                        02B3    918 MOVTOUSER:
     51    00A8 C5    9E     02B3    919         MOVAB   UCB$W_XI_INBUF(R5),R1   ; Address of internal buffer
          52    02    9A     02B8    920         MOVZBL  #2,R2
    00000000'GF    16     02BB    921         JSB     G^IOC$MOVTOUSER         ; Call exec
                        02C1    922 20$:                                    ; Update buffer pointers in UCB
      7C A5    02    A0     02C1    923         ADDW    #2,UCB$W_BOFF(R5)       ; Add two to buffer descriptor
7C A5    FE00 8F    AA     02C5    924         BICW    #^C<^X01FF>,UCB$W_BOFF(R5) ; Modulo the page size
           04    12     02CB    925         BNEQ    30$                     ; If NEQ, no page boundary crossed
      78 A5    04    C0     02CD    926         ADDL    #4,UCB$L_SVAPTE(R5)     ; Point to next page
                        02D1    927 30$:
                 05     02D1    928         RSB
                        02D2    929         .DSABL  LSB
```

```
                        02D2   931          .SBTTL  XI_TIME_OUTW,  Device time-out routine
                        02D2   932 ;++
                        02D2   933 ; Device TIME-OUT
                        02D2   934 ;
                        02D2   935 ; Clear port CSR
                        02D2   936 ; Return error status
                        02D2   937 ;
                        02D2   938 ; Power failure will appear as a device time-out
                        02D2   939 ;--
                        02D2   940
                        02D2   941 XI_TIME_OUTW:                            ; Time-out for WORD mode transfer
                        02D2   942
          00E0    30    02D2   943          BSBW    XI_DEV_RESET             ; Reset controller
     50   022C 8F 3C    02D5   944          MOVZWL  #SS$_TIMEOUT,R0          ; Error status
          51      D4    02DA   945          CLRL    R1
        68 A5     B4    02DC   946          CLRW    UCB$W_DEVSTS(R5)         ; Clear ATTN AST flags
                  AA    02DF   947          BICW    #<UCB$M_TIM     ! -
                        02E0   948                    UCB$M_INT     ! -
                        02E0   949                    UCB$M_TIMOUT  ! -
                        02E0   950                    UCB$M_CANCEL  ! -
                        02E0   951                    UCB$M_POWER>,-
     64 A5   006B 8F    02E0   952                  UCB$W_STS(R5)            ; Clear unit status flags
                        02E5   953          REQCOM                           ; Complete I/O in exec
```

```
                        02EB   955          .SBTTL  XI_INTERRUPT,    Interrupt service routine for PORT
                        02EB   956 ;++
                        02EB   957 ; XI_INTERRUPT, Handles interrupts generated by port
                        02EB   958 ;
                        02EB   959 ; Functional description:
                        02EB   960 ;
                        02EB   961 ;       This routine is entered whenever an interrupt is generated
                        02EB   962 ;       by the port.  It checks that an interrupt was expected.
                        02EB   963 ;       If not, it sets the unexpected (unsolicited) interrupt flag.
                        02EB   964 ;       All device registers are read and stored into the UCB.
                        02EB   965 ;       If an interrupt was expected, it calls the driver back at its Wait
                        02EB   966 ;       For Interrupt point.
                        02EB   967 ;       Deliver ATTN AST's if unexpected interrupt.
                        02EB   968 ;
                        02EB   969 ; Inputs:
                        02EB   970 ;
                        02EB   971 ;       00(SP) = Pointer to address of the device IDB
                        02EB   972 ;       04(SP) = saved R0
                        02EB   973 ;       08(SP) = saved R1
                        02EB   974 ;       12(SP) = saved R2
                        02EB   975 ;       16(SP) = saved R3
                        02EB   976 ;       20(SP) = saved R4
                        02EB   977 ;       24(SP) = saved R5
                        02EB   978 ;       28(SP) = saved PSL
                        02EB   979 ;       32(SP) = saved PC
                        02EB   980 ;
                        02EB   981 ; Outputs:
                        02EB   982 ;
                        02EB   983 ;       The driver is called at its Wait For Interrupt point if an
                        02EB   984 ;       interrupt was expected.
                        02EB   985 ;       The current value of the port CSR's are stored in the UCB.
                        02EB   986 ;
                        02EB   987 ;--
                        02EB   988 XI_INTERRUPT:                            ; Interrupt service for PORT
                        02EB   989
                  54    9E   D0  02EB   990          MOVL    @(SP)+,R4      ; Address of IDB and pop SP
                  54    64   7D  02EE   991          MOVQ    (R4),R4        ; CSR and UCB address from IDB
                        02F1   992 ;
                        02F1   993 ; Read INBUF and CSR
                        02F1   994 ;
              00A8 C5     04 A4  B0  02F1   995          MOVW    XI_INBUF(R4), -
                        02F7   996                          UCB$W_XI_INBUF(R5)  ; Read input data
                        64     B0  02F7   997          MOVW    XI_CSR(R4),-
                  00AA C5          02F9   998                  UCB$W_XI_CSR(R5)    ; Read CSR
                        02FC   999
                        02FC  1000 ; Check to see if device transfer request active or not
                        02FC  1001 ; If so, call driver back at Wait for Interrupt point and
                        02FC  1002 ; Clear unexpected interrupt flag.
                        02FC  1003
           0D 64 A5     01   E5  02FC  1004          BBCC    #UCB$V_INT, -
                        0301  1005                          UCB$W_STS(R5),10$   ; If clear, no interrupt expected
                        0301  1006
                        0301  1007 ; Interrupt expected, clear unexpected interrupt flag and call driver
                        0301  1008 ; back.
                        0301  1009
              68 A5     02   AA  0301  1010          BICW    #UCB$M_UNEXPT, -
                        0305  1011                          UCB$W_DEVSTS(R5)    ; Clear unexpected interrupt flag
```

```
                                                     J 16
53   10 A5   D0   0305  1012          MOVL    UCB$L_FR3(R5),R3        ; Restore drivers R3
     0C B5   16   0309  1013          JSB     @UCB$L_FPC(R5)          ; Call driver back after WFIKPCH
        0C   11   030C  1014          BRB     20$                     ; Exit
                  030E  1015
                  030E  1016 ; Deliver ATTN AST's if no interrupt expected and set unexpected
                  030E  1017 ; interrupt flag.
                  030E  1018
                  030E  1019 10$:
68 A5   02   A8   030E  1020          BISW    #UCB$M_UNEXPT,-
                  0312  1021                  UCB$W_DEVSTS(R5)        ; Set unexpected interrupt flag
      0051   30   0312  1022          BSBW    XI_DE[_ATTNAST          ; Deliver ATTN AST's
   0060 8F   A8   0315  1023          BISW    #XI_CSR$M_IEAB,-
        64        0319  1024                  XI_CSR(R4)              ; Enable device interrupts (A & B)
                  031A  1025
                  031A  1026 ; Restore registers and return from interrupt
                  031A  1027
                  031A  1028 20$:
        3F   BA   031A  1029          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers
             02   031C  1030          REI                             ; Return from interrupt
```

```
                     031D  1032          .SBTTL  XI_CANCEL,       Cancel I/O routine
                     031D  1033 ;++
                     031D  1034 ; XI_CANCEL, Cancels an I/O operation in progress
                     031D  1035 ;
                     031D  1036 ; Functional description:
                     031D  1037 ;
                     031D  1038 ;       Flushes Attention AST queue for the user.
                     031D  1039 ;       If transfer in progress, do a device reset to port
                     031D  1040 ;       and finish the request.
                     031D  1041 ;       Clear interrupt expected flag.
                     031D  1042 ;
                     031D  1043 ; Inputs:
                     031D  1044 ;
                     031D  1045 ;       R2 = negated value of channel index
                     031D  1046 ;       R3 = address of current IRP
                     031D  1047 ;       R4 = address of the PCB requesting the cancel
                     031D  1048 ;       R5 = address of the device's UCB
                     031D  1049 ;
                     031D  1050 ; Outputs:
                     031D  1051 ;
                     031D  1052 ;--
                     031D  1053
                     031D  1054 XI_CANCEL:                                       ; Cancel I/O
                     031D  1055
1A 68 A5    00   E5  031D  1056          BBCC    #UCB$V_ATTNAST, -
                     0322  1057                  UCB$W_DEVSTS(R5),20$    ; ATTN AST enabled?
                     0322  1058
                     0322  1059 ; Finish all ATTN AST's for this process.
                     0322  1060
     00C4 8F    BB   0322  1061          PUSHR   #^M<R2,R6,R7>
     56   52    D0   0326  1062          MOVL    R2,R6                   ; Set up channel number
57   00A0 C5    9E   0329  1063          MOVAB   UCB$L_XI_ATTN(R5),R7    ; Address of listhead
 00000000'GF    16   032E  1064          JSB     G^COM$FLUSHATTNS        ; Flush ATTN AST's for process
     00C4 8F    BA   0334  1065          POPR    #^M<R2,R6,R7>
  68 A5    02    AA   0338  1066          BICW    #UCB$M_UNEXPT, -
                     033C  1067                  UCB$W_DEVSTS(R5)        ; Clear unexpected interrupt flag
                     033C  1068
                     033C  1069 ; Check to see if a data transfer request is in progress
                     033C  1070 ; for this process on this channel
                     033C  1071
                     033C  1072 20$:
                     033C  1073          SETIPL  UCB$B_DIPL(R5)          ; Lock out device interrupts
 00000000'GF    16   0340  1074          JSB     G^IOC$CANCELIO          ; Check if transfer going
16 64 A5    03   E1  0346  1075          BBC     #UCB$V_CANCEL, -
                     034B  1076                  UCB$W_STS(R5),30$       ; Branch if not for this guy
                     034B  1077
 50   0830 8F    3C   034B  1078          MOVZWL  #SS$_CANCEL,R0          ; Status is request canceled
           51    D4   0350  1079          CLRL    R1
        68 A5    B4   0352  1080          CLRW    UCB$W_DEVSTS(R5)        ; Clear unexpected interrupt flag
                  AA   0355  1081          BICW    #<UCB$M_TIM     ! -
                     0356  1082                    UCB$M_BSY     ! -
                     0356  1083                    UCB$M_CANCEL  ! -
                     0356  1084                    UCB$M_INT     ! -
                     0356  1085                    UCB$M_TIMOUT>,-
64 A5   014B 8F      0356  1086                  UCB$W_STS(R5)           ; Clear unit status flags
                     035B  1087          REQCOM                          ; Jump to exec to finish I/O
                     0361  1088
```

```
                0361  1089 30$:
                0361  1090          SETIPL  UCB$B_FIPL(R5)          ; Lower to FORK IPL
 05             0365  1091          RSB                             ; Return
```

```
                                                M 16
                     0366  1093          .SBTTL  XI_DEL_ATTNAST,  Deliver ATTN AST's
                     0366  1094 ;++
                     0366  1095 ; XI_DEL_ATTNAST, Deliver all outstanding ATTN AST's
                     0366  1096 ;
                     0366  1097 ; Functional description:
                     0366  1098 ;
                     0366  1099 ;       This routine is used by the port driver to deliver all of the
                     0366  1100 ;       outstanding attention AST's.  It is copied from COM$DELATTNAST in
                     0366  1101 ;       the exec.  In addition, it places the saved value of the port CSR
                     0366  1102 ;       and Input Data Buffer Register in the AST paramater.
                     0366  1103 ;
                     0366  1104 ; Inputs:
                     0366  1105 ;
                     0366  1106 ;       R5 = UCB of unit
                     0366  1107 ;
                     0366  1108 ; Outputs:
                     0366  1109 ;
                     0366  1110 ;       R0,R1,R2 Destroyed
                     0366  1111 ;       R3,R4,R5 Preserved
                     0366  1112 ;--
                     0366  1113 XI_DEL_ATTNAST:
 49 68 A5    00   E5 0366  1114          BBCC    #UCB$V_ATTNAST, -
                     036B  1115                  UCB$W_DEVSTS(R5),30$    ; Any ATTN AST's expected?
          38   BB  036B  1116          PUSHR   #^M<R3,R4,R5>           ; Save R3,R4,R5
    51    08   AE  D0  036D  1117 10$:    MOVL    8(SP),R1                ; Get address of UCB
 52    00A0  C1   9E  0371  1118          MOVAB   UCB$L_XI_ATTN(R1),R2    ; Address of ATTN AST listhead
       55    62   D0  0376  1119          MOVL    (R2),R5                 ; Address of next entry on list
             37   13  0379  1120          BEQL    20$                     ; No next entry, end of loop
    68 A1    02   AA  037B  1121          BICW    #UCB$M_UNEXPT, -
                     037F  1122                  UCB$W_DEVSTS(R1)        ; Clear unexpected interrupt flag
          62   65   D0  037F  1123          MOVL    (R5),(R2)               ; Close list
1E A5     00A8  C1   B0  0382  1124          MOVW    UCB$W_XI_INBUF(R1), -
                     0388  1125                  ACB$L_KAST+6(R5)        ; Store INBUF in AST paramater
1C A5     00AA  C1   B0  0388  1126          MOVW    UCB$W_XI_CSR(R1), -
                     038E  1127                  ACB$L_KAST+4(R5)        ; Store CSR in AST paramater
             DC AF   9F  038E  1128          PUSHAB  B^10$                   ; Set return address for FORK
                     0391  1129                                          ;   so that it loops through all AST's
                     0391  1130          FORK                            ; FORK for this AST
                     0397  1131
                     0397  1132 ; AST fork procedure
                     0397  1133
10 A5     18 A5     7D  0397  1134          MOVQ    ACB$L_KAST(R5),ACB$L_AST(R5)
                     039C  1135                                          ; Re-arrange entries
0B A5     20 A5     90  039C  1136          MOVB    ACB$L_KAST+8(R5),ACB$B_RMOD(R5)
0C A5     24 A5     D0  03A1  1137          MOVL    ACB$L_KAST+12(R5),ACB$L_PID(R5)
          18 A5     D4  03A6  1138          CLRL    ACB$L_KAST(R5)
       52    01     9A  03A9  1139          MOVZBL  #PRI$_IOCOM,R2          ; Set up priority increment
 00000000'GF       17  03AC  1140          JMP     G^SCH$QAST              ; Queue the AST
                     03B2  1141
          38   BA  03B2  1142 20$:    POPR    #^M<R3,R4,R5>           ; Restore registers
               05  03B4  1143 30$:    RSB                             ; Return
```

```
                                                B 1
                     03B5  1145          .SBTTL  XI_DEV_RESET,    Device reset routine
                     03B5  1146 ;++
                     03B5  1147 ; XI_DEV_RESET - Device reset routine
                     03B5  1148 ;
                     03B5  1149 ; This routine raises IPL to device IPL, performs a device reset to
                     03B5  1150 ; the required controler, and re-enables device interrupts.
                     03B5  1151 ;
                     03B5  1152 ; Inputs:
                     03B5  1153 ;
                     03B5  1154 ;       R4 - Address of Control and Status Register
                     03B5  1155 ;       R5 - Address of UCB
                     03B5  1156 ;
                     03B5  1157 ; Outputs:
                     03B5  1158 ;
                     03B5  1159 ;       Controller is reset, controller interrupts are enabled
                     03B5  1160 ;
                     03B5  1161 ;--
                     03B5  1162
                     03B5  1163 XI_DEV_RESET:
                     03B5  1164
                     03B5  1165          DSBINT                          ; Raise IPL to lock all interrupts
                     03BB  1166
4000 8F    AB        03BB  1167          BISW    #XI_CSR$M_RESET,-
     64              03BF  1168                  XI_CSR(R4)              ; Reset device
                     03C0  1169
                     03C0  1170          TIMEWAIT -                      ; Timewait to allow reset
                     03C0  1171                  TIME = #500,-
                     03C0  1172                  BITVAL = #XI_CSR$M_RESET,-
                     03C0  1173                  SOURCE = XI_CSR(R4),-
                     03C0  1174                  CONTEXT = W,-
                     03C0  1175                  SENSE = .FALSE.
                     03E9  1176
0060 8F    AB        03E9  1177          BISW    #XI_CSR$M_IEAB,-
     64              03ED  1178                  XI_CSR(R4)              ; Enable device interrupts (A & B)
                     03EE  1179
                     03EE  1180          ENBINT                          ; Restore IPL
          05         03F1  1181          RSB
                     03F2  1182
                     03F2  1183 XI_END:                                  ; End of driver label
                     03F2  1184          .END
```

```
$$$                             = 00000020 R      02
$$OP                            = 00000002
ACB$B_RMOD                      = 0000000B
ACB$L_AST                       = 00000010
ACB$L_KAST                      = 00000018
ACB$L_PID                       = 0000000C
AT$_UBA                         = 00000001
COM$FLUSHATTNS                    ********    X   03
COM$SETATTNAST                    ********    X   03
CRB$L_INTD                      = 00000024
CRB$L_INTD2                     = 00000048
DC$_REALTIME                    = 00000060
DDB$L_DDT                       = 0000000C
DEV$M_AVL                         ********    X   02
DEV$M_IDV                         ********    X   02
DEV$M_ODV                         ********    X   02
DEV$M_RTM                         ********    X   02
DPT$C_LENGTH                    = 00000038
DPT$C_VERSION                   = 00000004
DPT$INITAB                        00000038 R      02
DPT$M_SVP                       = 00000002
DPT$REINITAB                      00000054 R      02
DPT$TAB                           00000000 R      02
DT$_XI_DR11C                    = 0000000D
DYN$C_CRB                       = 00000005
DYN$C_DDB                       = 00000006
DYN$C_DPT                       = 0000001E
DYN$C_UCB                       = 00000010
EXE$ABORTIO                       ********    X   03
EXE$FINISHIO                      ********    X   03
EXE$FORK                          ********    X   03
EXE$GL_TENUSEC                    ********    X   03
EXE$GL_UBDELAY                    ********    X   03
EXE$IOFORK                        ********    X   03
EXE$READ                          ********    X   03
EXE$SENSEMODE                     ********    X   03
EXE$SETCHAR                       ********    X   03
EXE$WRITE                         ********    X   03
FUNCTAB_LEN                     = 0000004C
IDB$B_COMBO_CSR_OFFSET          = 0000000F
IDB$B_COMBO_VECTOR_OFFSET       = 00000010
IDB$B_VECTOR                    = 0000000B
IDB$L_CSR                       = 00000000
IDB$L_OWNER                     = 00000004
IDB$L_UCBLST                    = 00000018
IO$S_FCODE                      = 00000006
IO$V_ATTNAST                    = 00000008
IO$V_FCODE                      = 00000000
IO$V_RESET                      = 0000000B
IO$V_SETFNCT                    = 00000009
IO$V_TIMED                      = 00000007
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$_WRITEVBLK                   = 00000030
IOC$CANCELIO                      ********    X   03
IOC$MNTVER                        ********    X   03
IOC$MOVFRUSER                     ********    X   03
IOC$MOVTOUSER                     ********    X   03
IOC$REQCOM                        ********    X   03
IOC$RETURN                        ********    X   03
IOC$WFIKPCH                       ********    X   03
IRP$L_MEDIA                     = 00000038
IRP$L_SEGVBN                    = 00000048
IRP$W_FUNC                      = 00000020
MASKH                           = 00000080
MASKL                           = 08000000
MOVFRUSER                         000002A2 R      03
MOVTOUSER                         000002B3 R      03
P1                              = 00000000
P2                              = 00000004
P3                              = 00000008
P4                              = 0000000C
P5                              = 00000010
P6                              = 00000014
PR$_IPL                         = 00000012
PRI$_IOCOM                      = 00000001
RETURN_STATUS                     000001E6 R      03
SCH$QAST                          ********    X   03
SIZ...                          = 00000001
SS$_BADPARAM                    = 00000014
SS$_CANCEL                      = 00000830
SS$_NORMAL                      = 00000001
SS$_TIMEOUT                     = 0000022C
UCB$B_DEVCLASS                  = 00000040
UCB$B_DEVTYPE                   = 00000041
UCB$B_DIPL                      = 0000005E
UCB$B_FIPL                      = 0000000B
UCB$K_SIZE                      = 000000AC
UCB$L_CRB                       = 00000024
UCB$L_DEVCHAR                   = 00000038
UCB$L_DPC                       = 0000009C
UCB$L_FPC                       = 0000000C
UCB$L_FR3                       = 00000010
UCB$L_SVAPTE                    = 00000078
UCB$L_XI_ATTN                     000000A0
UCB$L_XI_DPR                      000000A4
UCB$M_ATTNAST                   = 00000001
UCB$M_BSY                       = 00000100
UCB$M_CANCEL                    = 00000008
UCB$M_INT                       = 00000002
UCB$M_ONLINE                    = 00000010
UCB$M_POWER                     = 00000020
UCB$M_TIM                       = 00000001
UCB$M_TIMOUT                    = 00000040
UCB$M_UNEXPT                    = 00000002
UCB$V_ATTNAST                   = 00000000
```

```
UCB$V_CANCEL                    = 00000003
UCB$V_INT                       = 00000001
UCB$V_POWER                     = 00000005
UCB$V_UNEXPT                    = 00000001
UCB$W_BCNT                      = 0000007E
UCB$W_BOFF                      = 0000007C
UCB$W_DEVBUFSIZ                 = 00000042
UCB$W_DEVSTS                    = 00000068
UCB$W_FUNC                      = 0000009A
UCB$W_STS                       = 00000064
UCB$W_XI_CSR                      000000AA
UCB$W_XI_INBUF                    000000A8
VEC$B_DATAPATH                  = 00000013
VEC$L_IDB                       = 00000008
VEC$L_INITIAL                   = 0000000C
VEC$M_PATHLOCK                  = 00000080
WORD_MODE                         0000018A R      03
WORD_MODE_READ                    0000020C R      03
WORD_MODE_WRITE                   0000018F R      03
XI$DDT                            00000000 RG     03
XI$K_VEC_OFFSET                 = 00000002
XI_CANCEL                         0000031D R      03
XI_CONTROL_INIT                   00000084 R      03
XI_CSR                            00000000
XI_CSR$M_CTRL0                  = 00000001
XI_CSR$M_CTRL1                  = 00000002
XI_CSR$M_IEAB                   = 00000060
XI_CSR$M_INTENB_A               = 00000020
XI_CSR$M_INTENB_B               = 00000040
XI_CSR$M_RESET                  = 00004000
XI_DEF_BUFSIZ                   = 0000FFFF
XI_DEF_TIMEOUT                  = 0000000A
XI_DEL_ATTNAST                    00000366 R      03
XI_DEV_RESET                      000003B5 R      03
XI_END                            000003F2 R      03
XI_FUNCTABLE                      00000038 R      03
XI_INBUF                          00000004
XI_IND                            00000006
XI_INTERRUPT                      000002EB R      03
XI_MISC                           00000004
XI_OUTBUF                         00000002
XI_READ_WRITE                     000000A9 R      03
XI_SETMODE                        000000CF R      03
XI_START                          0000010D R      03
XI_TIME_OUTW                      000002D2 R      03
```

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

```
PSECT name                      Allocation              PSECT No.  Attributes
----------                      ----------              ---------  ----------
.  ABS  .                       00000000  (     0.)     00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                           000000AC  (   172.)     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                   000003F2  ( 1010.)     03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC LONG
```

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.04	00:00:00.71
Command processing	106	00:00:00.39	00:00:02.80
Pass 1	496	00:00:14.23	00:00:51.93
Symbol table sort	0	00:00:02.05	00:00:06.46
Pass 2	211	00:00:03.10	00:00:10.70
Symbol table output	20	00:00:00.12	00:00:00.59
Psect synopsis output	1	00:00:00.01	00:00:00.10
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	866	00:00:19.94	00:01:13.30

The working set limit was 1950 pages.
118674 bytes (232 pages) of virtual memory were used to buffer the intermediate code.
There were 110 pages of symbol table space allocated to hold 1953 non-local and 39 local symbols.
1184 source lines were read in Pass 1, producing 18 object records in Pass 2.
40 pages of virtual memory were used to define 37 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYS.OBJ]LIB.MLB;1	24
_255DUA28:[SYSLIB]STARLET.MLB;2	9
TOTALS (all libraries)	33

2206 GETS were required to define 33 macros.

There were no errors, warnings or information messages.

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

XGDRIVER
LIS

XIDRIVER
LIS

XQDRIVER
LIS

XMDRIVER
LIS