M 2

```
IIIIIIIII        OOOOOOOOO        SSSSSSSSSSSS  UUU         UUU  PPPPPPPPPPPP
IIIIIIIII        OOOOOOOOO        SSSSSSSSSSSS  UUU         UUU  PPPPPPPPPPPP
IIIIIIIII        OOOOOOOOO        SSSSSSSSSSSS  UUU         UUU  PPPPPPPPPPPP
   III        OOO         OOO  SSS              UUU         UUU  PPP         PPP
   III        OOO         OOO  SSS              UUU         UUU  PPP         PPP
   III        OOO         OOO  SSS              UUU         UUU  PPP         PPP
   III        OOO         OOO  SSS              UUU         UUU  PPP         PPP
   III        OOO         OOO  SSS              UUU         UUU  PPP         PPP
   III        OOO         OOO  SSS              UUU         UUU  PPP         PPP
   III        OOO         OOO     SSSSSSSSS     UUU         UUU  PPPPPPPPPPPP
   III        OOO         OOO     SSSSSSSSS     UUU         UUU  PPPPPPPPPPPP
   III        OOO         OOO     SSSSSSSSS     UUU         UUU  PPPPPPPPPPPP
   III        OOO         OOO              SSS  UUU         UUU  PPP
   III        OOO         OOO              SSS  UUU         UUU  PPP
   III        OOO         OOO              SSS  UUU         UUU  PPP
   III        OOO         OOO              SSS  UUU         UUU  PPP
   III        OOO         OOO              SSS  UUU         UUU  PPP
   III        OOO         OOO              SSS  UUU         UUU  PPP
IIIIIIIII        OOOOOOOOO     SSSSSSSSSSSS     UUUUUUUUUUUUUUU  PPP
IIIIIIIII        OOOOOOOOO     SSSSSSSSSSSS     UUUUUUUUUUUUUUU  PPP
IIIIIIIII        OOOOOOOOO     SSSSSSSSSSSS     UUUUUUUUUUUUUUU  PPP
```

XF9
VO4

FILEID**DRSUP

J 3

```
DDDDDDDD    RRRRRRRR        SSSSSSSS  UU      UU  PPPPPPPP
DDDDDDDD    RRRRRRRR        SSSSSSSS  UU      UU  PPPPPPPP
DD      DD  RR      RR  SS          UU      UU  PP      PP
DD      DD  RR      RR  SS          UU      UU  PP      PP
DD      DD  RR      RR  SS          UU      UU  PP      PP
DD      DD  RR      RR  SS          UU      UU  PP      PP
DD      DD  RRRRRRRR      SSSSSS    UU      UU  PPPPPPPP
DD      DD  RRRRRRRR      SSSSSS    UU      UU  PPPPPPPP
DD      DD  RR  RR              SS  UU      UU  PP
DD      DD  RR  RR              SS  UU      UU  PP
DD      DD  RR    RR            SS  UU      UU  PP              ....
DD      DD  RR    RR            SS  UU      UU  PP              ....
DDDDDDDD    RR      RR  SSSSSSSS    UUUUUUUUUU  PP              ....
DDDDDDDD    RR      RR  SSSSSSSS    UUUUUUUUUU  PP              ....

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
```

XF$
V04

Table of contents

```
0000          .TITLE  XF$DRSUP -- DR32 SUPPORT ROUTINES
0000          .IDENT  'V04-000'
0000
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
0000    ;++
0000    ; FACILITY:      DR32 SUPPORT ROUTINES
0000    ;
0000    ; ABSTRACT:
0000    ;       Provide high-level language interface to DR32
0000    ;
0000    ; ENVIRONMENT:  USER MODE LIBRARY ROUTINES
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-003 TCM0004         Trudy C. Matthews       30-Mar-1983
0000    ;               Correct two bugs introduced in TCM0003 that could cause
0000    ;               user-specified action routines to not be called.
0000    ;
0000    ;       V03-002 TCM0003         Trudy C. Matthews       18-Jun-1982
0000    ;               Change XF$STARTDEV so that it sets the GO bit before exiting.
0000    ;
0000    ;               Correct two problems in XF$PKTBLD -- (1) if an action routine
0000    ;               was not specified, the packet would always be inserted at the
0000    ;               tail of the queue (even if MODES specified "insert at head");
0000    ;               (2) if MODES was defaulted and an action routine specified,
0000    ;               an access violation would occur.
0000    ;
0000    ;       V03-001 SBL3001         Steven B. Lionel        30-Mar-1982
0000    ;               Change module name to XF$DRSUP.  Make PRE_AST, GET_ADDR
0000    ;               and DEVICE_FAB local symbols.
0000    ;
0000    ;       V02-004 PRD0006         Paul R. DeStefano       1-Mar-1982
0000    ;               Correct symbols LIB$GET_VM in ALOCCMD and LIB$FREE_VM in
```

```
0000  ;             XF$CLEANUP.  Symbols were not position independent.
0000  ;
0000  ;     V02-003 TCM0002          Trudy C. Matthews        6-Jul-1981
0000  ;             In XF$GETPKT, correct the instruction that stores the
0000  ;             function code in the user supplied argument to only store
0000  ;             a word instead of a longword.
0000  ;
0000  ;     V02-002 TCM0001          Trudy C. Matthews        15-Jun-1981
0000  ;             In ALOCCMD, correct algorithm that initializes free command
0000  ;             memory pointers.
0000  ;
0000  ;--
```

```
0000          .SBTTL  DECLARATIONS
0000  ;
0000  ; MACROS:
0000  ;
0000          $SSDEF                                  ;define status returns
0000          $XFDEF                                  ;DR32-specific definitions
0000          $$DRDEFS                                ;support routine definitions
0000          $CTXDEF                                 ;offsets into contxt array
0000          $IODEF                                  ;IO status definitions
0000          $SHRDEF                                 ;shared status definitions
0000
0000  ;macro DEFAULT_TEST tests for defaulted FORTRAN-procedure arguments
0000
0000          .MACRO  DEFAULT_TEST    ARGPOS, LABEL1, LABEL2
0000  ;ARGPOS contains the position of an argument in the argument list
0000
0000          CMPL    (AP), #ARGPOS                   ;arg given?
0000          BLSS    LABEL1                          ;argument was not supplied
0000          TSTL    <ARGPOS*4>(AP)                  ;if address = 0
0000          BEQL    LABEL2                          ;argument was defaulted
0000          .ENDM   DEFAULT_TEST
0000
0000  ;macro QRETRY executes an interlocked queue instruction and retries
0000  ;if failure.
0000  ;INPUTS:
0000  ;       OPCODE = opcode name: INSQHI, INSQTI, REMQHI, REMQTI
0000  ;       OPERAND1 = first operand for opcode
0000  ;       OPERAND2 = second operand for opcode
0000  ;       SUCCESS = label to branch to if operation succeeds
0000  ;       ERROR = label to branch to if operation fails
0000  ;OUTPUTS:
0000  ;       R0 is destroyed
0000
0000          .MACRO  QRETRY  OPCODE,OPERAND1,OPERAND2,SUCCESS,ERROR,?LOOP,?OK
0000          CLRL    R0
0000  LOOP:
0000          OPCODE  OPERAND1, OPERAND2
0000          .IF NB  SUCCESS                         ;"C" bit clear <=> success
0000          BCC     SUCCESS
0000          .IFF
0000          BCC     OK
0000          .ENDC
0000          AOBLSS  #RETRY_LIMIT, R0, LOOP  ;queue is interlocked. Retry.
0000          .IF NB ERROR
0000          BRB     ERROR                           ;retry limit exceeded and queue
0000          .ENDC                                   ;is still locked. Assume error.
0000  OK:
0000          .ENDM   QRETRY
0000
0000  ;
0000  ; REGISTER CONVENTIONS:
0000  ;       R6 : address of CONTXT array
0000  ;       R7 : address of current command packet
0000  ;       R10: address of command block
0000  ;
```

```
0000        .SBTTL  XF$SETUP
0000  ;++
0000  ;CONTXT ARRAY:
0000  ;
0000  ;          31                             0
0000  ;          +------------------------------+
0000  ;          ¦                              ¦   :CTX$Q_IOSB
0000  ;          +--     I/O status block     --+
0000  ;          ¦                              ¦
0000  ;          +------------------------------+
0000  ;          ¦  device and command control  ¦   :CTX$L_CONTROL
0000  ;          +------------------------------+
0000  ;          ¦       buffer size            ¦   :CTX$L_BYTECNT
0000  ;          +------------------------------+
0000  ;          ¦       buffer address         ¦   :CTX$L_BFRVA
0000  ;          +------------------------------+
0000  ;          ¦ residual memory byte count   ¦   :CTX$L_MEMCNT
0000  ;          +------------------------------+
0000  ;          ¦  residual DDI byte count     ¦   :CTX$L_DDICNT
0000  ;          +------------------------------+
0000  ;          ¦ DR32 status longword (DSL)   ¦   :CTX$L_DSL
0000  ;          +------------------------------+
0000  ;          ¦    size of command block     ¦   :CTX$L_CMDSIZ    :CTX$B_CMDTBL
0000  ;          +------------------------------+
0000  ;          ¦  address of command block    ¦   :CTX$L_CMDBLK
0000  ;          +------------------------------+
0000  ;          ¦     size of data block       ¦   :CTX$L_DATASIZ
0000  ;          +------------------------------+
0000  ;          ¦    address of data block     ¦   :CTX$L_DATABLK
0000  ;          +------------------------------+
0000  ;          ¦ address of pre- AST routine  ¦   :CTX$L_PRE_AST
0000  ;          +------------------------------+
0000  ;          ¦     pre- AST parameter       ¦   :CTX$L_PRE_PARM
0000  ;          +------------------------------+
0000  ;          ¦               ¦ flags ¦ datart¦   :CTX$B_DATART  CTX$B_FLAGS
0000  ;          +------------------------------+
0000  ;          ¦ addr to receive addr of gobit ¦  :CTX$L_GOBITADR
0000  ;          +------------------------------+
0000  ;          ¦ event flag #  ¦ # of buffers ¦   :CTX$W_NUMBUF  :CTX$W_EFN
0000  ;          +------------------------------+
0000  ;          ¦ address of packet AST routine ¦  :CTX$L_PKTAST
0000  ;          +------------------------------+
0000  ;          ¦   packet AST parameter       ¦   :CTX$L_ASTPARM
0000  ;          +------------------------------+
0000  ;          ¦ size of each buffer in BARRAY ¦  :CTX$L_BUFSIZ
0000  ;          +------------------------------+
0000  ;          ¦  address of IDEVMSG array    ¦   :CTX$L_IDEVMSG
0000  ;          +------------------------------+
0000  ;          ¦  address of ILOGMSG array    ¦   :CTX$L_ILOGMSG
0000  ;          +------------------------------+
0000  ;          ¦size of IDEVMSG¦size of ILOGMSG¦  :CTX$W_ILOGSIZ :CTX$W_IDEVSIZ
0000  ;          +------------------------------+
0000  ;          ¦ address of free memory list  ¦   :CTX$L_FREELIST
0000  ;          +------------------------------+
0000  ;                  note: CONTXT offsets are defined in $CTXDEF
```

```
                    0000  ; FUNCTIONAL DESCRIPTION:
                    0000  ;
                    0000  ;       (1) allocates command area
                    0000  ;       (2) allocates and initializes hardware queue headers
                    0000  ;       (3) initializes free command memory list
                    0000  ;       (4) initializes many fields in the CONTXT array
                    0000  ;
                    0000  ; CALLING SEQUENCE:
                    0000  ;
                    0000  ;       CALLS/G XF$SETUP (contxt, barray, bufsiz, numbuf, [idevmsg], -
                    0000  ;                         [idevsiz], [ilogmsg], [ilogsiz], [cmdsiz], -
                    0000  ;                         [status])
                    0000  ;
                    0000  ; INPUT PARAMETERS:
                    0000  ;   offsets to AP:
          00000004  0000        CONTXT = 4        ;a  50-word array that contains context and
                    0000                          ;status information concerning the current
                    0000                          ;transfer
          00000008  0000        BARRAY = 8        ;base address of data area
          0000000C  0000        BUFSIZ = 12       ;the size in bytes of each buffer in BARRAY
          00000010  0000        NUMBUF = 16       ;the number of buffers in BARRAY
          00000014  0000        IDEVMSG = 20      ;array to receive input device messages
          00000018  0000        IDEVSIZ = 24      ;size in bytes of device message array
          0000001C  0000        ILOGMSG = 28      ;array to receive input log messages
          00000020  0000        ILOGSIZ = 32      ;size in bytes of log message array
          00000024  0000        CMDSIZ = 36       ;size of command area to allocate
                    0000  ;
                    0000  ; OUTPUT PARAMETERS:
                    0000  ;
          00000028  0000        STATUS = 40       ;a longword array to receive status of call
                    0000  ;
                    0000  ; IMPLICIT OUTPUTS:
                    0000  ;
                    0000  ;       fields in CONTXT:       CTX$L_BUFSIZ
                    0000  ;                               CTX$L_CMDBLK
                    0000  ;                               CTX$L_CMDSIZ
                    0000  ;                               CTX$L_DATABLK
                    0000  ;                               CTX$L_DATASIZ
                    0000  ;                               CTX$L_IDEVMSG
                    0000  ;                               CTX$W_IDEVSIZ
                    0000  ;                               CTX$L_ILOGMSG
                    0000  ;                               CTX$W_ILOGSIZ
                    0000  ;                               CTX$W_NUMBUF
                    0000  ;
                    0000  ; COMPLETION CODES:
                    0000  ;
                    0000  ;       (1) SS$_NORMAL          normal successful completion
                    0000  ;       (2) SS$_BADPARAM        invalid input argument
                    0000  ;       (3) error status returns from LIB$GET_VM
                    0000  ;
                    0000  ; SIDE EFFECTS:
                    0000  ;
                    0000  ;       NONE
                    0000  ;
                    0000  ;--
```

```
                  00000000        .PSECT  _XF$CODE         SHR,PIC,EXE,NOWRT
                      0000
              004C    0000        .ENTRY  XF$SETUP         ^M<R2,R3,R6>
                      0002
                      0002  ;store input parameters in CONTXT array
                      0002
        50    14  3C  0002        MOVZWL  #SS$_BADPARAM, R0         ;set R0 for possible error
                      0005                                          ;return
        04    6C  D1  0005        CMPL    (AP), #4                  ;4 obligatory parameters
              03  18  0008        BGEQ    10$
            008C  31  000A        BRW     FINISH                    ;needed parameter defaulted
        56 04 AC  D0  000D  10$:  MOVL    CONTXT(AP), R6            ;R6 contains address of CONTXT
                      0011                                          ;array
  2C A6    08 AC  D0  0011        MOVL    BARRAY(AP), CTX$L_DATABLK(R6)     ;address of buf array
  40 A6    10 BC  B0  0016        MOVW    @NUMBUF(AP), CTX$W_NUMBUF(R6)     ;number of buffers
  4C A6    0C BC  D0  001B        MOVL    @BUFSIZ(AP), CTX$L_BUFSIZ(R6)     ;size of each buffer
                      0020
                      0020  ;determine size of data area, and store in CONTXT
                      0020
     52    40 A6  3C  0020        MOVZWL  CTX$W_NUMBUF(R6), R2      ;R2 <- # of buffers in BARRAY
28 A6   4C A6  52  C5  0024        MULL3    R2, -                           ;number of buffers X
                      002A                CTX$L_BUFSIZ(R6), -               ;size of each buffer
                      002A                CTX$L_DATASIZ(R6)
                      002A
                      002A  ;store addresses and sizes of arrays to receive input messages
                      002A
                      002A  MSG_ARRAYS:
        50 A6     7C  002A        CLRQ    CTX$L_IDEVMSG(R6)         ;zero addresses of device and
                      002D                                          ;log message arrays
        58 A6     D4  002D        CLRL    CTX$W_ILOGSIZ(R6)         ;assume sizes of device and
                      0030                                          ;log message arrays = 0
                      0030
                      0030        DEFAULT_TEST    <IDEVMSG/4>, 10$, 10$
                      003A                                    ;if IDEVMSG defaulted, goto 10$
  50 A6    14 AC  D0  003A        MOVL    IDEVMSG(AP), CTX$L_IDEVMSG(R6)
                      003F                                    ;store addr of IDEVMSG array
                      003F        DEFAULT_TEST    <IDEVSIZ/4>, 10$, 10$
                      0049                                    ;if IDEVSIZ defaulted, goto 10$
  5A A6    18 BC  B0  0049        MOVW    @IDEVSIZ(AP), CTX$W_IDEVSIZ(R6)
                      004E                                    ;store size of IDEVMSG array
                      004E  10$:  DEFAULT_TEST    <ILOGMSG/4>, CMDSIZ_TEST, CMDSIZ_TEST
                      0058                                    ;if ILOGMSG defaulted, goto CMDSIZ_TEST
  54 A6    1C AC  D0  0058        MOVL    ILOGMSG(AP), CTX$L_ILOGMSG(R6)
                      005D                                    ;store addr of ILOGMSG array
                      005D        DEFAULT_TEST    <ILOGSIZ/4>, CMDSIZ_TEST, CMDSIZ_TEST
                      0067                                    ;if ILOGSIZ defaulted, goto CMDSIZ_TEST
  58 A6    20 BC  B0  0067        MOVW    @ILOGSIZ(AP), CTX$W_ILOGSIZ(R6)
                      006C                                    ;store size of ILOGSIZ array
                      006C
```

```
                       006C  CMDSIZ_TEST:
                       006C  ;determine size of command area , and store in CONTXT
                       006C
                       006C          DEFAULT_TEST    <CMDSIZ/4>, COMSIZ, COMSIZ
                       0076                                          ;was size of command block given
                       0076                                          ;if not, goto COMSIZ
                       0076
20 A6   24 BC   18  C1 0076          ADDL3   #24, @CMDSIZ(AP), CTX$L_CMDSIZ(R6)      ;yes, add space
                       007C                                          ;for queue headers,
                       007C                                          ;and store in CONTXT
                14  11 007C          BRB     ALOC
                       007E
                       007E  ;default command size = NUMBUF * (size of fixed portion of command
                       007E  ;packet + idevsiz + ilogsiz) * arbitrary constant ( originally = 3)
                       007E
                       007E  COMSIZ:
                53  D4 007E          CLRL    R3
53    58 A6   5A A6 A1 0080          ADDW3   CTX$W_IDEVSIZ(R6), -     ;this sum will be <= 256
                       0086                  CTX$W_ILOGSIZ(R6), R3
            53  20  C0 0086          ADDL2   #XF$B_PKT_DEVMSG, R3     ;add in fixed portion of packet
     20 A6  53  52  C5 0089          MULL3   R2, R3, CTX$L_CMDSIZ(R6) ;R2 = NUMBUF
        20 A6   03  C4 008E          MULL2   #CMDSIZ_K, CTX$L_CMDSIZ(R6)         ;multiply by constant
                       0092
                       0092  ;intialize the addr of the addr of the go bit in CONTXT now so that
                       0092  ;XF$PKTBLD may be called before XF$STARTDEV.  It will be initialized
                       0092  ;again in XF$STARTDEV; this is a dummy initialization.
                       0092
                       0092  ALOC:
             39 A6  DE 0092          MOVAL   <CTX$B_CMDTBL + XF$B_CMT_FLAGS>(R6),-    ;request go bit
             3C A6     0095                  <CTX$B_CMDTBL + XF$L_CMT_GBITAD>(R6)     ;addr in here
                       0097
                       0097  ;All input parameters have been stored.  Now allocate and initialize
                       0097  ;command area.
                       0097
                0F  10 0097          BSBB    ALOCCMD                      ;allocate command area
                       0099                                               ;and initialize queue heads
                       0099                                               ;status returned in R0
                       0099  FINISH:
                       0099          DEFAULT_TEST    <STATUS/4>, END, END    ;was status arg given?
                       00A3                                               ;if not, branch to END
        28 BC   50  D0 00A3          MOVL    R0, @STATUS(AP)              ;yes, store status return
                       00A7
                    04 00A7  END:    RET
```

```
00A8          .SBTTL  ALOCCMD -- ALLOCATE COMMAND AREA
00A8    ;                        AND INITIALIZE HARDWARE QUEUES
00A8    ;++
00A8    ; FUNCTIONAL DESCRIPTION:
00A8    ;
00A8    ;          This routine is called by XF$SETUP to dynamically allocate the
00A8    ;          virtual memory that will be used as the command block during a
00A8    ;          DR32 data transfer.  If successful, it initializes the first 3
00A8    ;          quadwords as headers of the INPUT, TERMINATION, and FREE queues.
00A8    ;          When this routine exits, command memory looks like:
00A8    ;          31                    0
00A8    ;          +---------------------+
00A8    ;          !     INPTQ FLINK     !     :CMD$L_INPTQ
00A8    ;          +---------------------+
00A8    ;          !     INPTQ BLINK     !
00A8    ;          +---------------------+
00A8    ;          !     TERMQ FLINK     !     :CMD$L_TERMQ
00A8    ;          +---------------------+
00A8    ;          !     TERMQ BLINK     !
00A8    ;          +---------------------+
00A8    ;          !     FREEQ FLINK     !     :CMD$L_FREEQ
00A8    ;          +---------------------+
00A8    ;          !     FREEQ BLINK     !
00A8    ;          +---------------------+
00A8    ;          !          0          !     :first longword of command space
00A8    ;          +---------------------+
00A8    ;          ! size of packet memory !
00A8    ;          +---------------------+
00A8    ;          !          .          !
00A8    ;          !          .          !
00A8    ;          ! memory available to !
00A8    ;          !  build command packets!
00A8    ;          !          .          !
00A8    ;          !          .          !
00A8    ;          +---------------------+
00A8    ;
00A8    ; CALLING SEQUENCE:
00A8    ;
00A8    ;          BSBB    ALOCCMD
00A8    ;          BSBW    ALOCCMD
00A8    ;                        called by XF$SETUP
00A8    ;
```

```
00A8  ; INPUT PARAMETERS:
00A8  ;
00A8  ;       NONE
00A8  ;
00A8  ; IMPLICIT INPUTS:
00A8  ;
00A8  ;       address of CONTXT array in R6
00A8  ;       CONTXT fields used as inputs:   CTX$L_CMDSIZ
00A8  ;
00A8  ; OUTPUT PARAMETERS:
00A8  ;
00A8  ;       NONE
00A8  ;
00A8  ; IMPLICIT OUTPUTS:
00A8  ;
00A8  ;       fields in CONTXT:
00A8  ;               CTX$L_CMDBLK    address of allocated command area
00A8  ;               CTX$L_FREELIST  address of first longword on free list
00A8  ;
00A8  ; COMPLETION CODES:
00A8  ;
00A8  ;       R0 contains status of call to LIB$GET_VM
00A8  ;
00A8  ; SIDE EFFECTS:
00A8  ;
00A8  ;       NONE
00A8  ;
00A8  ;--
00A8
```

```
                                00A8
                                00A8  ALOCCMD:
                     5A   DD    00A8            PUSHL   R10                             ;save register
                                00AA
                                00AA  ;round size of command area up to next page boundary before allocating
                                00AA
     20 A6   000001FF 8F  C0    00AA            ADDL2   #PAGEMASK, CTX$L_CMDSIZ(R6)     ;increase size past
                                00B2                                                    ;next boundary
        20 A6    01FF 8F  AA    00B2            BICW    #PAGEMASK, CTX$L_CMDSIZ(R6)     ;truncate back to
                                00B8                                                    ;multuple
                                00B8
                                00B8  ;allocate command area
                                00B8
                  24 A6   DF    00B8            PUSHAL   CTX$L_CMDBLK(R6)               ;receives address of
                                00BB                                                    ;allocated area
                  20 A6   DF    00BB            PUSHAL  CTX$L_CMDSIZ(R6)                ;size to allocate
       00000000'GF   02   FB    00BE            CALLS   #2, G^LIB$GET_VM                ;get virtual memory
                  1A 50   E9    00C5            BLBC    R0, 10$                         ;error check
                                00C8
                                00C8  ;initialize hardware queues
                                00C8
             5A   24 A6   D0    00C8            MOVL    CTX$L_CMDBLK(R6), R10           ;R10 points to beginning
                                00CC                                                    ;of command block
                     6A   7C    00CC            CLRQ    CMD$L_INPTQ(R10)                ;initialize queue head
                  08 AA   7C    00CE            CLRQ    CMD$L_TERMQ(R10)                ;initialize
                  10 AA   7C    00D1            CLRQ    CMD$L_FREEQ(R10)                ;initialize queue head
                                00D4
                                00D4  ;initialize list of free memory chunks
                                00D4
                  18 AA   DE    00D4            MOVAL   <CMD$L_FREEQ+8>(R10),-  ;FREELIST points to
                  5C A6         00D7                    CTX$L_FREELIST(R6)      ;first available blk of memory
                                00D9
                                00D9  ;The amount of command block memory available for building packets =
                                00D9  ;the size of command area - space reserved for queue heads.
                                00D9
     1C AA    20 A6   18  C3    00D9            SUBL3   #24,CTX$L_CMDSIZ(R6), - ;store size of initially
                                00DF                    <CMD$L_FREEQ+12>(R10)   ;available command memory
                  5C B6   D4    00DF            CLRL    @CTX$L_FREELIST(R6)     ;initialize free block pointer
                0400 8F   BA    00E2  10$:      POPR    #^M<R10>
                     05         00E6            RSB
```

```
                    00E7             .SBTTL  XF$STARTDEV -- START DEVICE
                    00E7     ;++
                    00E7     ; FUNCTIONAL DESCRIPTION:
                    00E7     ;
                    00E7     ;       (1) build command table required by Startdata QIO
                    00E7     ;       (2) assign a channel to the device
                    00E7     ;       (3) issue the Startdata QIO
                    00E7     ;
                    00E7     ; CALLING SEQUENCE:
                    00E7     ;
                    00E7     ;       CALLS/G XF$STARTDEV (CONTXT, DEVNAM,  [PKTAST], [ASTPARM],
                    00E7     ;                           [EFN], [MODES], [DATART], [STATUS])
                    00E7     ;
                    00E7     ; INPUT PARAMETERS:
                    00E7     ;
                    00E7     ; offsets to AP:
           00000004 00E7             CONTXT = 4                  ;address of CONTXT array
           00000008 00E7             DEVNAM = 8                  ;character string; device name of DR32
           0000000C 00E7             PKTAST = 12                 ;address of packet AST
           00000010 00E7             ASTPARM = 16                ;address of AST parameter
           00000014 00E7             EFN = 20                    ;event flag associated with transfer
           00000018 00E7             MODES = 24                  ;contains several switches
           0000001C 00E7             DATART = 28                 ;data rate of transfer
                    00E7     ;
                    00E7     ; IMPLICIT INPUTS:
                    00E7     ;
                    00E7     ;fields in the CONTXT array:
                    00E7     ;       CTX$L_CMDBLK
                    00E7     ;       CTX$L_CMDSIZ
                    00E7     ;       CTX$L_DATABLK
                    00E7     ;       CTX$L_DATASIZ
                    00E7     ;
                    00E7     ; OUTPUT PARAMETERS:
                    00E7     ;
           00000020 00E7             STATUS = 32                 ;optional status return
                    00E7     ;
                    00E7     ; IMPLICIT OUTPUTS:
                    00E7     ;
                    00E7     ;       various fields in the CONTXT array
                    00E7     ;
                    00E7     ; COMPLETION CODES:
                    00E7     ;
                    00E7     ;       (1) SS$_NORMAL          normal successful completion
                    00E7     ;       (2) SS$_BADPARAM        needed parameter defaulted
                    00E7     ;       (3) error returns from:
                    00E7     ;               $CREATE
                    00E7     ;               $QIO
                    00E7     ;
                    00E7     ; SIDE EFFECTS:
                    00E7     ;
                    00E7     ;       NONE
                    00E7     ;
                    00E7     ;--
```

```
                        00E7
                    00000000            .PSECT  _XF$DATA        NOEXE
                        0000  DEVICE_FAB:
                        0000            $FAB    FOP = UFO                       ;User File Open option
                        0050
                        0050
                    000000E7            .PSECT  _XF$CODE        EXE,NOWRT,SHR,PIC
                   004C 00E7            .ENTRY  XF$STARTDEV     ^M<R2,R3,R6>
                        00E9
  56    04 AC   D0      00E9            MOVL    CONTXT(AP), R6          ;R6 <- addr of CONTXT
                        00ED
                        00ED  ;++
                        00ED  ;Two of the device-dependent parameters of the Startdata QIO are the
                        00ED  ;address and the size of a 'command table'.
                        00ED  ;The format of this command table is:
                        00ED  ;              31                                0
                        00ED  ;             +----------------------------------+
                        00ED  ;             !      size of command block       !    :XF$L_CMT_CBLKSIZ
                        00ED  ;             +----------------------------------+
                        00ED  ;             !    address of command block      !    :XF$L_CMT_CBLKAD
                        00ED  ;             +----------------------------------+
                        00ED  ;             !       size of data block         !    :XF$L_CMT_BBLKSIZ
                        00ED  ;             +----------------------------------+
                        00ED  ;             !      address of data block       !    :XF$L_CMT_BBLKAD
                        00ED  ;             +----------------------------------+
                        00ED  ;             ! address of packet AST routine    !    :XF$L_CMT_PASTAD
                        00ED  ;             +----------------------------------+
                        00ED  ;             !     packet AST parameter         !    :XF$L_CMT_PASTPM
                        00ED  ;             +----------------------------------+
                        00ED  ;             !                 ! flags ! datart !    :XF$B_CMT_RATE  :XF$B_CMT_FLAGS
                        00ED  ;             +----------------------------------+
                        00ED  ;             !addr to receive addr of go bit    !    :XF$L_CMT_GBITAD
                        00ED  ;             +----------------------------------+
                        00ED  ;
                        00ED  ;This command table is embedded in the CONTXT array
                        00ED  ; (offset: CTX$B_CMDTBL).  The first 4 longwords have already been
                        00ED  ;initialized by XF$SETUP.  Now build the remainder of the table.
                        00ED  ;--
                        00ED
     30 A6      7C      00ED            CLRQ    <CTX$B_CMDTBL + XF$L_CMT_PASTAD>(R6)    ;zero AST fields
     44 A6      7C      00F0            CLRQ    CTX$L_PKTAST(R6)
     39 A6      94      00F3            CLRB    <CTX$B_CMDTBL + XF$B_CMT_FLAGS>(R6)     ;flags default
  42 A6   15    9B      00F6            MOVZBW  #EFN_DEF, CTX$W_EFN(R6) ;assume event flag # defaulted
     3C A6      DE      00FA            MOVAL   <CTX$B_CMDTBL + XF$L_CMT_GBITAD>(R6),-  ;request go bit
     3C A6              00FD                    <CTX$B_CMDTBL + XF$L_CMT_GBITAD>(R6)    ;addr in here
                        00FF
```

```
                              00FF
                              00FF      ;++
                              00FF      ;Determine if an AST routine is supplied.  If so, store in the
                              00FF      ;command table the address of a pre- AST routine, which is part of the
                              00FF      ;support package.  This pre- AST routine will take the AST, and after
                              00FF      ;some checks call the user AST routine.  The AST parameter in the
                              00FF      ;command table will point to 2 longwords elsewhere in the CONTXT
                              00FF      ;array, which will contain the address of the user AST routine and its
                              00FF      ;parameter.
                              00FF      ;--
                              00FF      PKTAST_TEST:
                              00FF              DEFAULT_TEST    <PKTAST/4>, ASSIGN_CHN, EFN_TEST
         000001D1'EF    DE    0109              MOVAL   PRE_AST,-               ;put pre- AST routine address in
               30 A6          010F                      <CTX$B_CMDTBL + XF$L_CMT_PASTAD>(R6) ;command table
   34 A6       44 A6    DE    0111              MOVAL   CTX$L_PKTAST(R6), -     ;pre- AST parm is a pointer to
                              0116                      <CTX$B_CMDTBL + XF$L_CMT_PASTPM>(R6) ;user AST address
   44 A6       0C AC    D0    0116              MOVL    PKTAST(AP), -           ;put user AST addr into CONTXT
                              011B                      CTX$L_PKTAST(R6)
                              011B              DEFAULT_TEST    <ASTPARM/4>, ASSIGN_CHN, EFN_TEST
   48 A6       10 BC    D0    0125              MOVL    @ASTPARM(AP), -         ;put user AST parm in CONTXT
                              012A                      CTX$L_ASTPARM(R6)
                              012A
                              012A      EFN_TEST:
                              012A              DEFAULT_TEST    <EFN/4>, ASSIGN_CHN, MODE_TEST
   42 A6       14 BC    B0    0134              MOVW    @EFN(AP), CTX$W_EFN(R6) ;put event flag # in CONTXT
                              0139
                              0139      MODE_TEST:
                              0139              DEFAULT_TEST    <MODES/4>, ASSIGN_CHN, DATART_TEST
               18 BC    90    0143              MOVB    @MODES(AP),-            ;put flags into command table
               39 A6          0146                      <CTX$B_CMDTBL + XF$B_CMT_FLAGS>(R6)
                              0148
                              0148      DATART_TEST:
                              0148              DEFAULT_TEST    <DATART/4>, ASSIGN_CHN, ASSIGN_CHN
                              0152                                              ;if < 3 args, goto ASSIGN_CHN
   38 A6       1C BC    90    0152              MOVB    @DATART(AP), -          ;put data rate into cmd table
                              0157                      <CTX$B_CMDTBL + XF$B_CMT_RATE>(R6)
     39 A6       01     88    0157              BISB    #XF$M_CMT_SETRTE, -     ;set data rate bit in FLAGS var
                              015B                      <CTX$B_CMDTBL+XF$B_CMT_FLAGS>(R6)        ;of cmd table
                              015B      ;++
                              015B      ;The command table is now complete.
                              015B      ;Assign a channel to the DR32.  The RMS $CREATE service with the User
                              015B      ;File Open option in the FOP field of the FAB is nothing more than a
                              015B      ;glorified assign channel, but it buys you multiple levels of logical
                              015B      ;name translation.
                              015B      ;--
                              015B
                              015B      ;initialize the FAB with the device name supplied by the caller
                              015B
                              015B      ASSIGN_CHN:
                              015B              DEFAULT_TEST    <DEVNAM/4>, BADPARM, BADPARM
   53    00000000'EF    DE    0165              MOVAL   DEVICE_FAB, R3          ;R3 <- addr of FAB
      52       08 AC    D0    016C              MOVL    DEVNAM(AP), R2          ;R2 <- addr of devnam descriptor
```

```
                              0170  ;++
                              0170  ;The address of the 'ORTRAN character string descriptor is in R2.
                              0170  ;The descriptor look  like:
                              0170  ;
                              0170  ;          +--------------------------------+
                              0170  ;          |   size of char string array    |   :(R2)
                              0170  ;          +--------------------------------+
                              0170  ;          |  address of character string   |
                              0170  ;          +--------------------------------+
                              0170  ;
                              0170  ;If the statically declared size of the array is larger than the actual
                              0170  ;string, the string will be padded with blanks.  Find the true size of
                              0170  ;the character string before assigning the channel.
                              0170  ;--
  2C A3    04 A2        D0    0170          MOVL    4(R2), FAB$L_FNA(R3)       ;move addr of char string to FAB
2C B3   62    20        3A    0175          LOCC    #^040, (R2), @FAB$L_FNA(R3) ;find first blank
  51    2C A3           C2    017A          SUBL2   FAB$L_FNA(R3), R1          ;R1 <- length of char string
  34 A3    51           90    017E          MOVB    R1, FAB$B_FNS(R3)          ;move size of string into FAB
                              0182          $CREATE FAB = DEVICE_FAB           ;returns channel # in STV field
        30 50           E9    018F          BLBC    R0, STAT                   ;store error status
                              0192
                              0192  ;issue QIO specifying evf to be set on every packet interrupt
                              0192
                              0192  10$:    $QIO_S  EFN = CTX$W_EFN(R6), -
                              0192                  CHAN = FAB$L_STV(R3), -
                              0192                  FUNC = #IO$_STARTDATA!IO$M_SETEVF, -
                              0192                  IOSB = CTX$Q_IOSB(R6), -          ;also embedded in CONTXT
                              0192                  ASTADR = @CTX$L_PRE_AST(R6), -    ;packet AST address
                              0192                  ASTPRM = CTX$L_PRE_PARM(R6), -
                              0192                  P1 = CTX$B_CMDTBL(R6), -          ;addr of command table
                              0192                  P2 = #XF$K_CMT_LENGTH             ;size of command table
        09 50           E9    01B6          BLBC    R0,STAT                    ;branch if QIO was unsuccessful
   3C B6    01          90    01B9          MOVB    #1,@<CTX$B_CMDTBL+XF$L_CMT_GBITAD>(R6)  ;set GO bit in case
                              01BD                                             ;there are packets already on INPUTQ
           03           11    01BD          BRB     STAT                       ;R0 contains status of QIO call
                              01BF
                              01BF  BADPARM:
      50   14           3C    01BF          MOVZWL  #SS$_BADPARAM, R0          ;needed argument defaulted
                              01C2  STAT:
                              01C2          DEFAULT_TEST    <STATUS/4>, END_STARTDEV, END_STARTDEV
   20 BC   50           D0    01CC          MOVL    R0, @STATUS(AP)            ;store status
                              01D0  END_STARTDEV:
                04            01D0          RET
```

```
01D1     .SBTTL  PRE_AST -- pre - user AST routine
01D1   ;++
01D1   ; FUNCTIONAL DESCRIPTION:
01D1   ;
01D1   ;       Check if the AST routine is interrupting  critical code in the
01D1   ;       main routine; that is, if it could leave the list of free
01D1   ;       memory in an invalid state.
01D1   ;       If so, turn off AST's, reschedule this AST, and return.
01D1   ;       If not, call the user - specified AST routine.
01D1   ;
01D1   ; CALLING SEQUENCE:
01D1   ;
01D1   ;       CALLS/G PRE_AST (ASTPARM)
01D1   ;
01D1   ; INPUT PARAMETERS:
01D1   ;
01D1   ;       ASTPARM points to two consecutive longwords containing
01D1   ;       the address of the user's AST and the user ASTPARM.
01D1   ;
01D1   ; IMPLICIT INPUTS:
01D1   ;
01D1   ;       NONE
01D1   ;
01D1   ; OUTPUT PARAMETERS:
01D1   ;
01D1   ;       NONE
01D1   ;
01D1   ; IMPLICIT OUTPUTS:
01D1   ;
01D1   ;       NONE
01D1   ;
01D1   ; COMPLETION CODES:
01D1   ;
01D1   ;       NONE
01D1   ;
01D1   ; SIDE EFFECTS:
01D1   ;
01D1   ;       NONE
01D1   ;
01D1   ;--
```

```
                          01D1      PRE_AST:
                    0000  01D1              .WORD   0
 51    04 AC      D0      01D3              MOVL    4(AP), R1               ;R1 <- addr of quadword
                          01D7                                              ;containing addr of PKTAST
                          01D7                                              ;and ASTPARM
 15 51    00      E4      01D7              BBSC    #CRITICAL_BIT, R1, -    ;determine if interrupting
                          01DB                      IMMEDIATE_EXIT          ;critical code; if so, exit
                          01DB      ;++
                          01DB      ;all OK; call user AST-level routine
                          01DB      ;--
                          01DB
       14 AC      DD      01DB              PUSHL   20(AP)                  ;saved PSL
       10 AC      DD      01DE              PUSHL   16(AP)                  ;saved PC
       0C AC      DD      01E1              PUSHL   12(AP)                  ;saved R1
       08 AC      DD      01E4              PUSHL   8(AP)                   ;saved R0
       04 A1      DD      01E7              PUSHL   4(R1)                   ;user AST-level parameter
 00 B1    05      FB      01EA              CALLS   #5, @(R1)               ;call user AST-level routine
          17      11      01EE              BRB     END_PRE_AST
                          01F0      ;++
                          01F0      ;Come here if interrupted main routine during critical code.
                          01F0      ;Disable AST's and reschedule this AST.
                          01F0      ;The main level routine will re-enable AST's when it exits the
                          01F0      ;critical section of code.
                          01F0      ;--
                          01F0      IMMEDIATE_EXIT:
                          01F0              $SETAST_S       #0              ;disable AST's
                          01F9              $DCLAST_S       PRE_AST, R1     ;reschedule this AST
                          0207
                          0207      END_PRE_AST:
                    04    0207              RET
```

```
0208        .SBTTL  XF$PKTBLD
0208  ;++
0208  ; FUNCTIONAL DESCRIPTION:
0208  ;
0208  ;           (1) finds # of bytes needed for command packet
0208  ;           (2) searches freelist to find space for packet and allocates it
0208  ;           (3) builds command packet
0208  ;           (4) puts it on input queue
0208  ;           (5) sets 'go' bit
0208  ;
0208  ;                   format of a command packet:
0208  ;            31                              0
0208  ;           +--------------------------------+
0208  ;           | self - relative forward link   |
0208  ;           +--------------------------------+
0208  ;           | self - relative backward link  |
0208  ;           +--------------------------------+
0208  ;           |pktctl |cmdctl |loglen |msglen  |   :(see below)
0208  ;           +--------------------------------+
0208  ;           |    byte count                  |   :XF$L_PKT_BFRSIZ
0208  ;           +--------------------------------+
0208  ;           |      virtual address of buffer |   :XF$L_PKT_BFRADR
0208  ;           +--------------------------------+
0208  ;           |  residual memory byte count    |   :XF$L_PKT_RMBCNT
0208  ;           +--------------------------------+
0208  ;           |  residual DDI byte count       |   :XF$L_PKT_RDBCNT
0208  ;           +--------------------------------+
0208  ;           |  DR32 Status Longword (DSL)    |   :XF$L_PKT_DSL
0208  ;           +--------------------------------+
0208  ;           |  DR - device message           |   :XF$B_PKT_DEVMSG
0208  ;           |       //      //       //      |
0208  ;           +--------------------------------+
0208  ;           |        log area                |
0208  ;           |       //      //       //      |
0208  ;           +--------------------------------+
0208  ;           |  address of ACTION routine     |
0208  ;           +--------------------------------+
0208  ;           |  address of ACTION parameter   |
0208  ;           +--------------------------------+
0208  ;
0208  ;           The log area and ACTION fields have no symbolic offset because
0208  ;           the length of the device message field is variable.  The third
0208  ;           longword of the command packet looks like this:
0208  ;
0208  ;            8                              0
0208  ;           +-------------------------------+
0208  ;           | length of device message      |   :XF$B_PKT_MSGLEN
0208  ;           +-------------------------------+
0208  ;           |   length of log area          |   :XF$B_PKT_LOGLEN
0208  ;           +-------------------------------+
0208  ;           | command control (function)    |   :XF$B_PKT_CMDCTL
0208  ;           +-------------------------------+
0208  ;           |  packet control byte          |   :XF$B_PKT_PKTCTL
0208  ;           +-------------------------------+
          0208  ; CALLING SEQUENCE:
          0208  ;
          0208  ;       CALLS/G XF$PKTBLD (contxt, func, [index], [difsize], [devmsg],
          0208  ;                   [devsiz], [logsiz], [modes], [action],
          0208  ;                   [actparm], [status])
          0208  ;
          0208  ; INPUT PARAMETERS:
          0208  ;
          0208  ;offsets to AP
          0208
00000004  0208          CONTXT = 4                      ;context array
00000008  0208          FUNC = 8                        ;a word containing a legal DR function
0000000C  0208          INDEX = 12                      ;the index of a buffer in BARRAY
00000010  0208          DIFSIZE = 16                    ;alternate byte count
00000014  0208          DEVMSG = 20                     ;location of a device message
00000018  0208          DEVSIZ = 24                     ;size of device message in bytes
0000001C  0208          LOGSIZ = 28                     ;amt of space to reserve for log msg
00000020  0208          MODES = 32                      ;flags and control bits to go in pkt
00000024  0208          ACTION = 36                     ;address of an ACTION routine
00000028  0208          ACTPARM = 40                    ;address of ACTION routine parameter
          0208  ;
          0208  ; OUTPUT PARAMETERS:
          0208  ;
0000002C  0208          STATUS = 44                     ;optional status returns (see below)
          0208  ;
          0208  ; IMPLICIT OUTPUTS:
          0208  ;
          0208  ;       NONE
          0208  ;
          0208  ; COMPLETION CODES:
          0208  ;
          0208  ;       (1) SS$_NORMAL          normal successful completion
          0208  ;       (2) SS$_BADPARAM        input parameter error
          0208  ;       (3) SS$_BADQUEUEHDR             INPUT queue interlock timeout
          0208  ;       (4) SS$_INSFMEM         not enough space to build packet
          0208  ;       (5) SHR$_NOCMDMEM       command memory not allocated
          0208  ;
          0208  ; SIDE EFFECTS:
          0208  ;
          0208  ;       NONE
          0208  ;
          0208  ;--
```

```
                        0208
                  0FFC  0208              .ENTRY  XF$PKTBLD       ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                        020A
                        020A    ;a command packet is divided into distinct areas
                        020A    ;       1. hardware portion
                        020A    ;               a. fixed length
                        020A    ;               b. variable length
                        020A    ;       2. software portion
                        020A
                        020A    ;both the hardware and the software portions must be allocated command
                        020A    ;space; however, only the size of the hardware portion will be made
                        020A    ;known to the DR32 hardware
                        020A
                        020A    ;the majority of packet information is contained in the hardware-fixed
                        020A    ;portion of the command packet.  The hardware-variable portion has two
                        020A    ;optional variable-length fields -- the device message field
                        020A    ;and the log message field.  These fields can be from 0 - 256
                        020A    ;bytes; however, they must be an integer number of longwords.
                        020A    ;the software portion of the command packet contains the address of the
                        020A    ;ACTION routine(if specified) and the address of its parameter
                        020A    ;ACTPARM (if specified)
                        020A
                        020A    ;in this section of code:
                        020A    ;               R2 will accumulate the total # of bytes for the command packet
                        020A
                        020A    ;compute total size of command packet by determining lengths of
                        020A    ;variable-length and optional fields
                        020A
          52     20  9A     020A          MOVZBL  #32, R2                         ;initialize R2 with # bytes in
                        020D                                                  ;hardware-fixed potion of packet
                 58  7C     020D          CLRQ    R8                              ;initialize device message and
                 53  7C     020F          CLRQ    R3                              ;log area sizes to 0
                        0211
                        0211    ;if < 5 arguments, R2 contains total size of packet--goto BITS
                        0211    ;if DEVSIZ was defaulted, branch to LOGSIZE
                        0211              DEFAULT_TEST    <DEVSIZ/4>, BITS, LOGSIZE
                        021B                                                  ;was size of device msg given?
      53      18 BC  3C     021B          MOVZWL  @DEVSIZ(AP), R3                 ;yes, round DEVSIZ up to
                        021F                                                  ;longword boundary
  58      53     07  C1     021F          ADDL3   #QUADWORD_MASK, R3, R8
          58     07  CA     0223          BICL    #QUADWORD_MASK, R8
  0100 8F        58  B1     0226          CMPW    R8, #256                        ;is size of dev msg > 256?
                 03  1B     022B          BLEQU   10$                             ;no, branch around error
               012E  31     022D          BRW     INVALID_ARG                     ;yes, error
          52     58  C0     0230    10$:  ADDL2   R8, R2                          ; add size to byte count
                        0233    ;R3 contains the actual size of the device message
                        0233    ;R8 contains the size rounded up to the next longword boundary
```

```
                      0233
                      0233  ;if < 6 arguments supplied, R2 contains total size-- goto BITS
                      0233  ;if LOGSIZ was defaulted, branch to see if ACTION routine was given
                      0233  LOGSIZE:
                      0233          DEFAULT_TEST    <LOGSIZ/4>, BITS, ACTION_ROUTINE
                      023D                                          ;was size of log message given?
                      023D
  54    1C BC    3C   023D          MOVZWL  @LOGSIZ(AP), R4         ;yes, round LOGSIZ up to
                      0241                                          ;longword boundary
 59   54    07   C1   0241          ADDL3   #QUADWORD_MASK, R4, R9
      59    07   CA   0245          BICL    #QUADWORD_MASK, R9
0100 8F     59   B1   0248          CMPW    R9, #256                ;is size of log msg > 256 bytes?
            03   1B   024D          BLEQU   GO                      ;no, branch around error path
         010C    31   024F  BR:     BRW     INVALID_ARG             ;yes, error
      52    59   C0   0252  GO:     ADDL2   R9,R2                   ; no, add size to byte count
                      0255  ;R4 contains the actual size of the space reserved for log message
                      0255  ;R9 contains the size rounded up to the next longword boundary
                      0255
                      0255  ACTION_ROUTINE:
                      0255          DEFAULT_TEST    <ACTION/4>, BITS, BITS
                      025F                                          ;was ACTION routine given?
                      025F                                          ;if no, branch to BITS
      52    08   C0   025F          ADDL2   #8, R2                  ;yes,add 4 bytes for ACTION addr
                      0262                                          ; + 4 bytes for ACTPARM address
                      0262
                      0262
                      0262  ;at this point:
                      0262  ;       R2 contains the number of bytes needed for command packet
                      0262
                      0262  BITS:
  56    04 AC    D0   0262          MOVL    CONTXT(AP), R6          ;address of CONTXT array in R6
  5A    24 A6    D0   0266          MOVL    CTX$L_CMDBLK(R6), R10   ;R10 <- addr of command block
            03   12   026A          BNEQ    2$                      ;is command area allocated?
         010B    31   026C          BRW     TRANSFER_HALTED         ;no, return error
  51    5C A6    DE   026F  2$:     MOVAL   CTX$L_FREELIST(R6), R1  ;R1 <- addr of freelist head
                      0273
         0118    30   0273          BSBW    XF$$ALOCPKT             ;input: # bytes in R2
                      0276                                          ;       freelist head in R1
                      0276                                          ;output: addr of pkt in R1
        03 50    E8   0276          BLBS    R0, 5$                  ;test low bit for error
         00F7    31   0279          BRW     NO_MEM                  ;not enough space
                      027C
      57    51   D0   027C  5$:     MOVL    R1, R7                  ;preserve addr of packet from
                      027F                                          ;destruction by MOVC5
      5B    52   D0   027F          MOVL    R2, R11                 ;save size of packet
```

```
                              0282  ;now build the packet
                              0282
                              0282  ;first compute the addresses of and insert the variable-length fields
                              0282  ;         R7:  address of command packet
                              0282  ;         R3:  actual size of device message (in bytes)
                              0282  ;         R8: size of device msg, rounded up to next longword boundary
                              0282  ;         R4: actual size of log area (in bytes)
                              0282  ;         R9: size of log area, rounded up to next longword boundary
                              0282
          08 A7     53    90  0282  NEXT:   MOVB    R3, XF$B_PKT_MSGLEN(R7) ;put size of dev msg
                              0286                                          ;into packet
          09 A7     54    90  0286          MOVB    R4, XF$B_PKT_LOGLEN(R7) ;put in size of log area
          0B A7     00    90  028A          MOVB    #MODES_DEFAULT, XF$B_PKT_PKTCTL(R7)
                              028E                                          ;put default MODES into packet
                              028E
                              028E  ;size of device message is in packet, now put the message itself in
                              028E          DEFAULT_TEST    <DEVMSG/4>, FUNC_FIELD, FUNC_FIELD
                              0298                                          ;if no dev msg, goto FUNC_FIELD
                              0298
                              0298  ;move device message into packet, filling with 0's to next longword
                              0298  ;boundary
 20 A7   58   00   14 BC   53  2C  0298          MOVC5   R3, @DEVMSG(AP), #0, R8, XF$B_PKT_DEVMSG(R7)
                              02A0
                              02A0  ;add the size of the fixed portion of the command packet to the sizes
                              02A0  ;of the device and log message fields to get the byte offset from the
                              02A0  ;beginning of the command packet to the ACTION routine field
                              02A0
                              02A0  FUNC_FIELD:
                              02A0
         58    20 A849    9E  02A0          MOVAB   XF$B_PKT_DEVMSG(R8)[R9], R8 ;R8 <- offset of ACTION
                              02A5
                              02A5  ;insert fixed-length arguments in the order they were supplied
                              02A5
                              02A5          DEFAULT_TEST    <FUNC/4>, INV, INV
                              02AF                                          ;if FUNC defaulted, goto
                              02AF                                          ;INVALID_ARG
          0F     08 BC    B1  02AF          CMPW    @FUNC(AP), #15          ;function codes are from 0:15
                    03    1B  02B3          BLEQU   OK                      ;branch around error path
                  00A6    31  02B5  INV:    BRW     INVALID_ARG             ;invalid function code
       0A A7     08 BC    90  02B8  OK:     MOVB    @FUNC(AP), XF$B_PKT_CMDCTL(R7)
                              02BD                                          ;insert function code
                              02BD                                          ;high bits must be zero
```

```
                            02BD  INDEX_FIELD:
             0C A7     7C   02BD          CLRQ    XF$L_PKT_BFRSIZ(R7)        ;clear byte count & buffer addr
                            02C0                                             ;(assume no data transfer)
                            02C0          DEFAULT_TEST    <INDEX/4>,FIELDS_DONE,ACTION_FIELD
                            02CA                                             ;if < 3 args goto FIELDS_DONE
                            02CA                                             ;else if defaulted go to ACTION_FIELD
         51    0C BC   3C   02CA          MOVZWL  @INDEX(AP), R1             ;R1 <- index of buffer
                  E5   13   02CE          BEQL    INV                        ;index of 0 is invalid
      40 A6       51   B1   02D0          CMPW    R1, CTX$W_NUMBUF(R6)       ;index > number of buffers?
                  DF   1A   02D4          BGTRU   INV                        ;yes, invalid buffer index
                  51   D7   02D6          DECL    R1                         ;R1 <- buffer offset from base
                            02D8                                             ;of buffer array
         51    4C A6   C4   02D8          MULL2   CTX$L_BUFSIZ(R6), R1       ;R1 <- byte offset from base of
                            02DC                                             ;buffer array of this buffer
10 A7    51    2C A6   C1   02DC          ADDL3   CTX$L_DATABLK(R6), R1,-
                            02E2                  XF$L_PKT_BFRADR(R7)        ;put buffer addr into packet
                            02E2          DEFAULT_TEST    <DIFSIZE/4>, 10$, 10$
   0C A7       10 BC   D0   02EC          MOVL    @DIFSIZE(AP), XF$L_PKT_BFRSIZ(R7)
                            02F1                                             ;alternate transfer byte count
                  05   11   02F1          BRB     ACTION_FIELD
   0C A7       4C A6   D0   02F3  10$:    MOVL    CTX$L_BUFSIZ(R6), XF$L_PKT_BFRSIZ(R7)
                            02F8                                             ;standard transfer byte count
                            02F8
                            02F8  ACTION_FIELD:
                            02F8          DEFAULT_TEST    <ACTION/4>,MODES_FIELD,MODES_FIELD
      0B A7       04   88   0302          BISB2   #XF$M_PKT_ACTBIT, -        ;set "ACTION routine given" bit
                            0306                  XF$B_PKT_PKTCTL(R7)        ;in packet control field
                            0306
                            0306  ;R8 contains byte offset from beginning of command packet to ACTION routine
                            0306  ;field of packet
                            0306
            58    57   C0   0306          ADDL2   R7,R8                      ;R8 <- addr of ACTION field
         88    24 AC   D0   0309          MOVL    ACTION(AP),(R8)+           ;put addr of ACTION routine into packet
                            030D          DEFAULT_TEST    <ACTPARM/4>,MODES_FIELD,MODES_FIELD
                            0317                                             ;if ACTPARM defaulted goto MODES_FIELD
         68    28 AC   D0   0317          MOVL    ACTPARM(AP),(R8)           ;put addr of ACTPARM in packet
                            031B
                            031B  MODES_FIELD:
                            031B          DEFAULT_TEST    <MODES/4>,FIELDS_DONE,FIELDS_DONE
                            0325                                             ;if MODES is defaulted, goto
                            0325                                             ;        FIELDS_DONE
      0B A7       00   8A   0325          BICB2   #MODES_DEFAULT, XF$B_PKT_PKTCTL(R7)
                            0329                                             ;clear out default modes settings but
                            0329                                             ;preserve "action routine present" bit
   0B A7       20 BC   88   0329          BISB2   @MODES(AP), XF$B_PKT_PKTCTL(R7)
                            032E                                             ;sets   (1) interrupt control
                            032E                                             ;       (2)length error bit
                            032E                                             ;       (3) pkt control bits
                            032E                                             ;to user-supplied values
                            032E
                            032E  ;the packet is now completely built and ready to be put on the input queue
                            032E
         20 BC    08   E1   032E          BBC     #XF$V_PKT_HT, @MODES(AP),-
                  11        0332                  INSERT_AT_TAIL             ;clear bit <==> tail
                            0333  INSERT_AT_HEAD:
                            0333  ;R10 contains the address of the command block
                            0333
```

```
                      0333                  QRETRY -
                      0333                  INSQHI  (R7), CMD$L_INPTQ(R10) -;attempt insertion
                      0333                          SUCCESS = SET_GO_BIT -
                      0333                          ERROR = Q_FAILURE         ;exceeded retry limit
                      0344
                      0344       FIELDS_DONE:
                      0344       INSERT_AT_TAIL:
                      0344                  QRETRY -
                      0344                  INSQTI  (R7), CMD$L_INPTQ(R10) -;attempt insertion at tail
                      0344                          ERROR = Q_FAILURE         ;exceeded retry limit
                      0355
                      0355       SET_GO_BIT:
                      0355
   3C B6    01    90  0355                  MOVB    #1, @<CTX$B_CMDTBL+XF$L_CMT_GBITAD>(R6)
                      0359                                                    ;notify the Dr that there is a
                      0359                                                    ;packet on the INPUT queue
                      0359
      50    01    3C  0359                  MOVZWL  #SS$_NORMAL, R0           ;success status return
            21    11  035C                  BRB     STORE_STAT                ;branch around error paths
                      035E       INVALID_ARG:
      50    14    3C  035E                  MOVZWL  #SS$_BADPARAM, R0         ;input parameter error
            05    11  0361                  BRB     DEALLOCATE
                      0363       Q_FAILURE:
  50   0394 8F    3C  0363                  MOVZWL  #SS$_BADQUEUEHDR, R0      ;interlocked queue timeout
                      0368       DEALLOCATE:                                  ;inputs to XF$$DEALOCPKT:
   51    5C A6    DE  0368                  MOVAL   CTX$L_FREELIST(R6), R1    ;R1: address of freelist head
      53    5B    D0  036C                  MOVL    R11, R3                   ;R3: size of packet in bytes
                      036F                                                    ;R7: address of packet
            61    10  036F                  BSBB    XF$$DEALOCPKT             ;deallocate the packet
            0C    11  0371                  BRB     STORE_STAT
                      0373       NO_MEM:
  50   0124 8F    3C  0373                  MOVZWL  #SS$_INSFMEM, R0          ;not enough space to build pkt
            05    11  0378                  BRB     STORE_STAT
                      037A       TRANSFER_HALTED:
  50   1278 8F    3C  037A                  MOVZWL  #SHR$_NOCMDMEM, R0        ;command memory not allocated
                      037F       STORE_STAT:
                      037F                  DEFAULT_TEST    <STATUS/4>, END_PKTBLD, END_PKTBLD
                      0389                                                    ;was STATUS arg given?
   2C BC    50    D0  0389                  MOVL    R0, @STATUS(AP) ;yes, store status return
                      038D       END_PKTBLD:
                  04  038D                  RET
```

```
038E    .SBTTL  XF$$ALOCPKT -- ALLOCATE A COMMAND PACKET
                               AND RETURN ITS ADDRESS
038E  ;
038E  ;++
038E  ; FUNCTIONAL DESCRIPTION:
038E  ;
038E  ;       This routine is called by XF$PKTBLD to allocate a command
038E  ;       packet.  It searches the list of free chunks of command
038E  ;       space to find the required amount of memory.
038E  ;
038E  ; CALLING SEQUENCE:
038E  ;
038E  ;       BSBW    XF$$ALOCPKT
038E  ;
038E  ; INPUT PARAMETERS:
038E  ;
038E  ;       NONE
038E  ;
038E  ; IMPLICIT INPUTS:
038E  ;
038E  ;       R1 contains the address of a pointer to the free list
038E  ;       R2 contains the number of bytes needed for packet
038E  ;
038E  ; OUTPUT PARAMETERS:
038E  ;
038E  ;       NONE
038E  ;
038E  ; IMPLICIT OUTPUTS:
038E  ;
038E  ;       R1 contains the address of the allocated packet
038E  ;
038E  ; COMPLETION CODES:
038E  ;
038E  ;       returned in R0 : not enough memory available
038E  ;                        1 = sucess
038E  ;
038E  ; SIDE EFFECTS:
038E  ;
038E  ;       NONE
038E  ;
038E  ;--
```

```
                          038E  XF$$ALOCPKT::                                           ;allocate memory
             0C   BB      038E          PUSHR   #^M<R2,R3>
                          0390
                          0390  ;Since command packets must be quadword aligned, the allocation
                          0390  ;granularity of each packet is 8 bytes.
                          0390
         52  07   C0      0390          ADDL2   #GRANULARITY, R2                ;round size up to next
         52  07   CA      0393          BICL2   #GRANULARITY, R2                ;quadword boundary
         50  51   D0      0396          MOVL    R1, R0                          ;copy address of first free
                          0399                                                  ;block address
      34 A6  01   88      0399          BISB2   #CRITICAL_MASK, -               ;set bit in AST parm to indicate
                          039D                  <CTX$B_CMDTBL+XF$L_CMT_PASTPM>(R6)
                          039D                                                  ;"entering critical code"
                          039D  ;Find a piece of memory large enough for requested allocation.
                          039D
         51  50   D0      039D  10$:    MOVL    R0, R1                          ;save addr of previous free blk
         50  61   D0      03A0          MOVL    (R1), R0                        ;get addr of next free block
             2A   13      03A3          BEQL    END_ALOCPKT                     ;if equal no memory available
      04 A0  52   D1      03A5          CMPL    R2, 4(R0)                       ;free block big enough?
             F2   1A      03A9          BGTRU   10$                             ;no, go try next block
                          03AB
                          03AB  ;free block found
                          03AB
             0E   13      03AB          BEQL    EQUAL                           ;if eql free block is exact size
                          03AD
                          03AD  ;Free block is bigger than requested allocation.  Allocate what was
                          03AD  ;asked for and put remainder of block back on free list.
                          03AD
     53  52  50   C1      03AD          ADDL3   R0, R2, R3                      ;R3 <- addr of new free block
                          03B1
         83  80   D0      03B1          MOVL    (R0)+, (R3)+                    ;copy link to new free block
     63  60  52   C3      03B4          SUBL3   R2, (R0), (R3)                  ;calc size of new free block
         70  73   DE      03B8          MOVAL   -(R3), -(R0)                    ;set link to new free block
                          03BB
                          03BB  ;Remove block from free list.
                          03BB  EQUAL:
         61  60   D0      03BB          MOVL    (R0), (R1)                      ;copy link to new free block
         51  80   9E      03BE          MOVAB   (R0)+, R1                       ;R1 <- addr of allocated blk
                          03C1                                                  ;R0 indicates success
   09 34 A6  00   E4      03C1          BBSC    #CRITICAL_BIT, -                ;did AST interrupt critical code
                          03C6                  <CTX$B_CMDTBL+XF$L_CMT_PASTPM>(R6), -
                          03C6                  END_ALOCPKT                     ;if not, branch to END_ALOCPKT
                          03C6          $SETAST_S       #1                      ;if so, the AST routine disabled
                          03CF                                                  ;AST's and rescheduled itself,so
                          03CF                                                  ;upon exiting critical code,
                          03CF                                                  ;re-enable AST's
                          03CF  END_ALOCPKT:
             0C   BA      03CF          POPR    #^M<R2,R3>
                  05      03D1          RSB
```

```
03D2          .SBTTL  XF$$DEALOCPKT -- DEALLOCATE COMMAND PACKET
03D2   ;++
03D2   ; FUNCTIONAL DESCRIPTION:
03D2   ;
03D2   ;       This routine is called by XF$GETPKT and XF$PKTBLD to return
03D2   ;       the memory used for a command packet.  It searches the list
03D2   ;       of free blocks of memory to find where to return the packet
03D2   ;       memory, and agglomerates the returned memory with adjacent
03D2   ;       blocks if possible.
03D2   ;
03D2   ; CALLING SEQUENCE:
03D2   ;
03D2   ;       NONE
03D2   ;
03D2   ; INPUT PARAMETERS:
03D2   ;
03D2   ;       NONE
03D2   ;
03D2   ; IMPLICIT INPUTS:
03D2   ;
03D2   ;       R1 = address of allocation region listhead
03D2   ;       R3 = size of blocks in bytes
03D2   ;       R7 = address of block to be deallocated
03D2   ;
03D2   ; OUTPUT PARAMETERS:
03D2   ;
03D2   ;       NONE
03D2   ;
03D2   ; IMPLICIT OUTPUTS:
03D2   ;
03D2   ;       NONE
03D2   ;
03D2   ; COMPLETION CODES:
03D2   ;
03D2   ;       NONE
03D2   ;
03D2   ; SIDE EFFECTS:
03D2   ;
03D2   ;       R1, R3, and R7 are destroyed
03D2   ;
03D2   ;--
```

```
                         03D2   XF$$DEALOCPKT::
            52     DD    03D2           PUSHL   R2
        53  07     C0    03D4           ADDL2   #GRANULARITY, R3              ;round size up to next
        53  07     CA    03D7           BICL2   #GRANULARITY, R3              ;quadword boundary
     34 A6  01     88    03DA           BISB2   #CRITICAL_MASK, -             ;set bit in AST parm to indicate
                         03DE                   <CTX$B_CMDTBL+XF$L_CMT_PASTPM>(R6)
                         03DE                                                 ;"entering critical code"
                         03DE
                         03DE   ;Find where in free list to return the memory.
                         03DE
        52  51     D0    03DE   10$:    MOVL    R1, R2                        ;R2 <- addr of prev free block
        51  62     D0    03E1           MOVL    (R2), R1                      ;R1 <- addr of next free block
            05     13    03E4           BEQL    20$                           ;if equal, end of list
        51  57     D1    03E6           CMPL    R7, R1                        ;block logically go here?
            F3     1A    03E9           BGTRU   10$                           ;no, keep looking
                         03EB
                         03EB   ;Determine of returned memory can be agglomerated with the block of
                         03EB   ;memory immediately following it.
                         03EB
        67  51     D0    03EB   20$:    MOVL    R1, (R7)                      ;assume no agglomeration
     7E 53  57     C1    03EE           ADDL3   R7, R3, -(SP)                 ;calculate addr of end of block
        8E  51     D1    03F2           CMPL    R1, (SP)+                     ;end of block = next in list?
            06     12    03F5           BNEQ    30$                           ;if neq do not agglomerate
        67  81     D0    03F7           MOVL    (R1)+, (R7)                   ;move link to block being freed
        53  61     C0    03FA           ADDL2   (R1), R3                      ;R3 <- length of new free block
                         03FD
                         03FD   ;Determine if returned memory can be agglomerated with the block of
                         03FD   ;memory immediately preceeding it.
                         03FD
            52     DD    03FD   30$:    PUSHL   R2                            ;calc end addr of previos block
        82  57     D0    03FF           MOVL    R7, (R2)+                     ;assume no agglomeration
        6E  62     C0    0402           ADDL2   (R2), (SP)                    ;add length to block base addr
        8E  57     D1    0405           CMPL    R7, (SP)+                     ;end addr = block being freed?
            09     12    0408           BNEQ    40$                           ;no, do not agglomerate blocks
        53  62     C0    040A           ADDL2   (R2), R3                      ;R3 <- size of new free block
        72  67     D0    040D           MOVL    (R7), -(R2)                   ;move link to previous free blk
        57  52     D0    0410           MOVL    R2, R7                        ;set addr of new free block
                         0413
     04 A7  53     D0    0413   40$:    MOVL    R3, 4(R7)                     ;set size of free block
  09 34 A6  00     E4    0417           BBSC    #CRITICAL_BIT, -              ;did AST interrupt critical code
                         041C                   <CTX$B_CMDTBL+XF$L_CMT_PASTPM>(R6), -
                         041C                   END_DEALOCPKT
                         041C           $SETAST_S       #1                    ;if so, the AST routine disabled
                         0425                                                 ;AST's and rescheduled itself,so
                         0425                                                 ;upon exiting critical code,
                         0425                                                 ;re-enable AST's
                         0425
                         0425   END_DEALOCPKT:
            04     BA    0425           POPR    #^M<R2>
                   05    0427           RSB
```

```
                 0428          .SBTTL  XF$FREESET -- PUT PACKETS ON FREEQ
                 0428   ;++
                 0428   ; FUNCTIONAL DESCRIPTION:
                 0428   ;
                 0428   ;      Determine the size of the packets to be released onto the FREE
                 0428   ;      queue according to input arguments.  Then build the number of
                 0428   ;      packets specified and release them onto the FREE queue.
                 0428   ;
                 0428   ; CALLING SEQUENCE:
                 0428   ;
                 0428   ;      CALLS/G  XF$FREESET(contxt, [numpkt], [intctrl], [action], -
                 0428   ;                                  [actparm], [status])
                 0428   ;
                 0428   ; INPUT PARAMETERS:
                 0428   ;
                 0428   ;offsets to AP:
        00000004 0428          CONTXT = 4                   ;context array
        00000008 0428          NUMPKT = 8                   ;number of packets to put on FREEQ
        0000000C 0428          INTCTRL = 12                 ;interrupt control bits to put in pkt
        00000010 0428          ACTION = 16                  ;address of ACTION routine
        00000014 0428          ACTPARM = 20                 ;address of ACTION parameter
                 0428   ;
                 0428   ; IMPLICIT INPUTS:
                 0428   ;
                 0428   ;      NONE
                 0428   ;
                 0428   ; OUTPUT PARAMETERS:
                 0428   ;
                 0428   ;offsets to AP:
        00000018 0428          STATUS = 24                  ;status returns (see completion codes)
                 0428   ;
                 0428   ; IMPLICIT OUTPUTS:
                 0428   ;
                 0428   ;      NONE
                 0428   ;
                 0428   ; COMPLETION CODES:
                 0428   ;
                 0428   ;      (1) SS$_NORMAL              normal successful completion
                 0428   ;      (2) SS$_BADQUEUEHDR                INPUT queue interlock timeout
                 0428   ;      (3) SS$_INSFMEM             not enough memory to build packet
                 0428   ;      (4) SHR$_NOCMDMEM           command memory is not allocated
                 0428   ;
                 0428   ; SIDE EFFECTS:
                 0428   ;
                 0428   ;      NONE
                 0428   ;
                 0428   ;--
```

```
                07FC  0428          .ENTRY  XF$FREESET      ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10>
56     04 AC    D0    042A          MOVL    CONTXT(AP), R6          ;R6 <- addr of CONTXT
5A     24 A6    D0    042E          MOVL    CTX$L_CMDBLK(R6), R10   ;R10 <- addr of command area
          03    12    0432          BNEQ    FIND_SIZE               ;if addr of command area = 0,
                      0434                                          ;transfer is halted
        00AD    31    0434          BRW     TRANS_HALTED            ;error path
                      0437
                      0437
                      0437
                      0437  ;++
                      0437  ;determine the size of the packets to be released onto the FREEQ by
                      0437  ;looking at the input arguments
                      0437  ;--
                      0437
                      0437  ;find size of field to reserve for device message
                      0437
                      0437  FIND_SIZE:
52     5A A6    3C    0437          MOVZWL  CTX$W_IDEVSIZ(R6), R2   ;R2 <- size of dev msg
   52     07    C0    043B          ADDL2   #QUADWORD_MASK, R2      ;round up to quadword boundary
   52     07    CA    043E          BICL    #QUADWORD_MASK, R2      ;R2 <- size of devmsg field
   52     20    C0    0441          ADDL2   #32, R2                 ;R2 <- size of command packet
                      0444
                      0444  ;determine if ACTION routine and ACTPARM are to be put in command pkt
                      0444
   57     52    D0    0444          MOVL    R2, R7                  ;R7 <- offset of ACTION routine
          53    7C    0447          CLRQ    R3                      ;assume no ACTION or ACTPARM
                      0449          DEFAULT_TEST    <ACTION/4>, 5$, 5$
                      0453                                          ;if defaulted, goto 5$
53     10 AC    D0    0453          MOVL    ACTION(AP), R3          ;R3 <- addr of ACTION routine
52        08    C0    0457          ADDL2   #8, R2                  ;add sizes of ACTION and ACTPARM
                      045A                                          ;to total packet size
                      045A          DEFAULT_TEST    <ACTPARM/4>, 5$, 5$
                      0464                                          ;if defaulted, goto 5$
54     14 AC    D0    0464          MOVL    ACTPARM(AP), R4 ;R4 <- addr of ACTPARM
                      0468
                      0468  ;find the interrupt control bits to be put in packet
                      0468
   58     00    9A    0468  5$:     MOVZBL  #INT_DEFAULT, R8                ;default interrupt ctrl setting
                      046B          DEFAULT_TEST    <INTCTRL/4>, 10$, 10$
                      0475                                          ;if defaulted goto 10$
58     0C BC    90    0475          MOVB    @INTCTRL(AP), R8        ;R8 <- interrupt control bits
                      0479
                      0479  ;find the number of packets to be put onto the FREEQ
                      0479
   55     01    9A    0479  10$:    MOVZBL  #1, R5                          ;default # of pkts to put on
                      047C                                                  ;FREEQ
                      047C          DEFAULT_TEST    <NUMPKT/4>, ANOTHER_PKT, ANOTHER_PKT
                      0486                                                  ;if defaulted goto ANOTHER_PKT
55     08 BC    3C    0486          MOVZWL  @NUMPKT(AP), R5                 ;R5 <- # of pkts to put on queue
```

```
                                048A  ;++
                                048A  ;build a packet
                                048A  ;--
                                048A  ANOTHER_PKT:
        51    5C A6    DE       048A           MOVAL   CTX$L_FREELIST(R6), R1  ;R1 <- addr of ptr to freelist
                FEFD   30       048E           BSBW    XF$$ALOCPKT             ;input: size of pkt in R2
                                0491                                           ;       freelist ptr in R1
                                0491                                           ;returns addr of pkt in R1
             49 50     E9       0491           BLBC    R0, NOT_MEM             ;not enough memory to build in
                  3E   BB       0494           PUSHR   #^M<R1,R2,R3,R4,R5>     ;preserve from MOVC5
 52    00   00000200'EF  00     2C 0496        MOVC5   #0, DUMMY_ADR, #0, R2, (R1)     ;zero the packet
                  61            049F
                  3E   BA       04A0           POPR    #^M<R1,R2,R3,R4,R5>     ;restore registers
                                04A2
                                04A2  ;put address of ACTION routine and addr of ACTPARM into packet
                                04A2  ;R3 contains addr of ACTION routine, R4 contains addr of parameter
                                04A2  ;R7 contains offset from beginning of packet to ACTION routine field
                                04A2
                  53   D5       04A2           TSTL    R3                      ;addr of ACTION routine
                  0B   13       04A4           BEQL    5$                      ;no ACTION routine if addr = 0
          59    6147   9E       04A6           MOVAB   (R1)[R7], R9            ; R9 <- addr of ACTION field
             69   53   7D       04AA           MOVQ    R3, (R9)                ;put ACTION and ACTPARM in pkt
          0B A1   04   88       04AD           BISB2   #XF$M_PKT_ACTBIT, -     ;set "ACTION routine given" bit
                                04B1                   XF$B_PKT_PKTCTL(R1)     ;in packet control field of pkt
                                04B1
                                04B1  ;insert interrupt control bits into packet
                                04B1
             06   58   F0       04B1  5$:      INSV    R8, #XF$V_PKT_INTCTL,-  ;put interrupt control bits
          0B A1   02            04B4                   #XF$S_PKT_INTCTL, XF$B_PKT_PKTCTL(R1)   ;into packet
                                04B7
                                04B7  ;put size of device message into packet
                                04B7
     08 A1    5A A6    90       04B7           MOVB    CTX$W_IDEVSIZ(R6), XF$B_PKT_MSGLEN(R1)
                                04BC
                                04BC  ;put packet onto FREEQ
                                04BC
                                04BC           QRETRY -
                                04BC           INSQTI  (R1), CMD$L_FREEQ(R10) -;attempt to insert packet
                                04BC                   ERROR = BAD_QUEUE       ;exceeded retry limit
                                04CE
                                04CE  A_OK:                                    ;packet is on queue
                                04CE
             B9 55     F5       04CE           SOBGTR  R5, ANOTHER_PKT         ;go do another packet
                                04D1
                                04D1
```

```
                            04D1  ;++
                            04D1  ;all the packets have been successfully inserted onto the FREEQ
                            04D1  ;--
                            04D1
          50   01   3C      04D1          MOVZWL  #SS$_NORMAL, R0          ;success status return
               13   11      04D4          BRB     END_FREESET
                            04D6
                            04D6  BAD_QUEUE:
     50   0394 8F   3C      04D6          MOVZWL  #SS$_BADQUEUEHDR, R0     ;interlock timeout
               0C   11      04DB          BRB     END_FREESET
                            04DD  NOT_MEM:
     50   0124 8F   3C      04DD          MOVZWL  #SS$_INSFMEM, R0         ;not enough command space
               05   11      04E2          BRB     END_FREESET
                            04E4  TRANS_HALTED:
     50   1278 8F   3C      04E4          MOVZWL  #SHR$_NOCMDMEM, R0       ;transfer halted; command
                            04E9                                           ;space deallocated
                            04E9  END_FREESET:
                            04E9          DEFAULT_TEST    <STATUS/4>, 10$, 10$
       18 BC   50   D0      04F3          MOVL    R0, @STATUS(AP)          ;store status
                    04      04F7  10$:    RET
```

```
                 04F8          .SBTTL  XF$GETPKT -- GET A PACKET
                 04F8  ;                               FROM THE TERMINATION QUEUE
                 04F8  ;++
                 04F8  ; FUNCIONAL DESCRIPTION:
                 04F8  ;
                 04F8  ;       Attempt to remove a packet from the TERMQ. If successful, break
                 04F8  ;the packet up into its various fields and return them to the caller. If
                 04F8  ;an ACTION routine is specified in the packet, call it.  Finally, return
                 04F8  ;the memory that was used to build this packet.
                 04F8  ;
                 04F8  ; CALLING SEQUENCE:
                 04F8  ;
                 04F8  ;       CALLS/G XF$GETPKT (contxt, [waitflg], [func], [index], -
                 04F8  ;                          [devflag], [logflag], [status])
                 04F8  ;
                 04F8  ; INPUT PARAMETERS:
                 04F8  ;
                 04F8  ;offsets to AP:
        00000004 04F8          CONTXT = 4                      ;context array
        00000008 04F8          WAITFLG = 8                     ;wait for event flag/immediate return
                 04F8  ;
                 04F8  ; IMPLICIT INPUTS:
                 04F8  ;
                 04F8  ;fields in the CONTXT array:
                 04F8  ;       CTX$L_DATABLK
                 04F8  ;       CTX$W_NUMBUF
                 04F8  ;       CTX$L_IDEVMSG
                 04F8  ;       CTX$L_IDEVSIZ
                 04F8  ;       CTX$L_ILOGMSG
                 04F8  ;       CTX$L_ILOGSIZ
                 04F8  ;
                 04F8  ; OUTPUT PARAMETERS:
                 04F8  ;
                 04F8  ;offsets to AP:
        0000000C 04F8          FUNC = 12                       ;function specified in packet
        00000010 04F8          INDEX = 16                      ;buffer index specified in packet
        00000014 04F8          DEVFLAG = 20                    ;set if device message in packet
        00000018 04F8          LOGFLAG = 24                    ;set if log message in packet
        0000001C 04F8          STATUS = 28                     ;status return
                 04F8  ;
                 04F8  ; IMPLICIT OUTPUTS:
                 04F8  ;
                 04F8  ;fields in the CONTXT array:
                 04F8  ;       CTX$L_MEMCNT
                 04F8  ;       CTX$L_DDICNT
                 04F8  ;       CTX$L_DSL
                 04F8  ;
```

```
XF$
Sym

$$.
$$.
$$.
$$.
$$.
$$T
ACT
ACT
ACT
ACT
ACT
ALC
ALC
ALC
ANC
ASS
AST
A_C
BAD
BAD
BAR
BIT
BR
BUF
CLE
CLE
CMD
CMD
CMD
CMD
CMD
CMD
COM
CON
CRI
CRI
CTX
CTX
CTX
CTX
CTX
CTX
CTX
CTX
CTX
CTX
CTX
CTX
CTX
CTX
CTX
CTX
CTX
CTX
```

```
04F8   ; COMPLETION CODES:
04F8   ;
04F8   ;         (1) SS$_NORMAL             normal successful completion
04F8   ;         (2) SS$_BADQUEUEHDR              TERM queue interlocked timeout
04F8   ;         (3) SHR$_HALTED            XF$CLEANUP was called
04F8   ;         (4) SHR$_QEMPTY(=0)        no packet, but transfer still going
04F8   ;         (5) SHR$_NOCMDMEM          no command memory was allocated at the
04F8   ;                                    time of the call to this routine
04F8   ;         (6) status of ACTION routine
04F8   ;                 XF$GETPKT's status is an input to the ACTION routine.
04F8   ;                 The ACTION routine may overwrite the status argument
04F8   ;                 with a status return of its own.
04F8   ;
04F8   ; SIDE EFFECTS:
04F8   ;
04F8   ;         If XF$CLEANUP was called, neither the command packets nor the
04F8   ;         queues are any longer accessible.
04F8   ;
04F8   ;--
```

```
                           04F8
                    0FFC   04F8          .ENTRY  XF$GETPKT       ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                           04FA
      56   04 AC     D0    04FA          MOVL    CONTXT(AP), R6           ;R6 <- addr of CONTXT
                           04FE
              52     D4    04FE          CLRL    R2                       ;assume WAITFLG defaulted
                           0500          DEFAULT_TEST    <WAITFLG/4>, 10$, 10$
                           050A                                           ;default is event flag wait
      52   08 BC     9A    050A          MOVZBL  @WAITFLG(AP), R2         ;input to GET_ADDR
            00EA     30    050E  10$:    BSBW    GET_ADDR                 ;returns addr of pkt in R7, if
                           0511                                           ;there is  a pkt on TERMQ
           03 50     E8    0511          BLBS    R0, DISSECT_PKT          ;status returned in R0
            00A1     31    0514          BRW     STORE_STATUS             ;error in removing pkt
                           0517                                           ;from TERMQ
                           0517
                           0517
                           0517  ;++
                           0517  ;come here if successfully removed a packet from TERMQ
                           0517  ;
                           0517  ;give the user as many command packet fields as he supplied variables
                           0517  ;to hold
                           0517  ;--
                           0517  DISSECT_PKT:
                           0517
                           0517  ;++
                           0517  ;compute sizes of device and log message fields
                           0517  ;
                           0517  ;the "actual size" of the device message is the number of bytes
                           0517  ;specified by the DEVSIZ argument in the call to XF$PKTBLD.  This is the
                           0517  ;value stored in the XF$B_PKT_MSGLEN field of the packet.
                           0517  ;the "packet size" of the device message refers to the fact that the
                           0517  ;device message field is 0-filled to the next longword boundary in the
                           0517  ;packet.  A similar situation occurs with the log message field.
                           0517  ;--
                           0517
     58    08 A7     9A    0517          MOVZBL  XF$B_PKT_MSGLEN(R7), R8 ;R8 <- actual size of device msg
  59   58    07      C1    051B          ADDL3   #QUADWORD_MASK, R8, R9  ;round size up to longword bound
       59    07      CA    051F          BICL    #QUADWORD_MASK, R9      ;R9 <- packet size of dev msg
                           0522
     5A    09 A7     9A    0522          MOVZBL  XF$B_PKT_LOGLEN(R7), R10 ;R10 <- actual size of log msg
  5B   5A    07      C1    0526          ADDL3   #QUADWORD_MASK, R10, R11 ;round up to longword boundary
       5B    07      CA    052A          BICL    #QUADWORD_MASK, R11      ;R11 <- packet size of log msg
                           052D
```

```
XF
Ps

PS
--

;
$A
_X
_X

Ph
--
In
Co
Pa
Sy
Pa
Sy
Ps
Cr
As

Th
90
Th
19
34

Ma
--
_$
_$
TO

15

Th

MA
```

```
                      052D          DEFAULT_TEST    <FUNC/4>, TRANSFER_STATUS, INDEX_TEST
                      0537                                      ;if < 3 args goto TRANSFER_STATUS
                      0537                                      ;else if FUNC defaulted goto INDEX_TEST
                      0537
                      0537 ;store function from packet into supplied argument
                      0537
  0C BC   0A A7   9B  0537          MOVZBW  XF$B_PKT_CMDCTL(R7), @FUNC(AP)
                      053C
                      053C
                      053C
                      053C INDEX_TEST:
                      053C          DEFAULT_TEST    <INDEX/4>, TRANSFER_STATUS, DEVFLAG_TEST
                      0546                                      ;if < 4 args goto TRANSFER_STATUS
                      0546                                      ;else if INDEX defaulted goto DEVFLAG
                      0546
                      0546 ;convert buffer address in packet to index
                      0546
     53   10 A7   D0  0546          MOVL    XF$L_PKT_BFRADR(R7),R3  ; was a data buffer transferred?
             0B   13  054A          BEQL    10$                     ; if addr = 0, no
53   53   2C A6   C3  054C          SUBL3   CTX$L_DATABLK(R6),R3,R3 ; yes, R3 <- byte offset from base
                      0551                                          ; of buffer array
     53   4C A6   C6  0551          DIVL2   CTX$L_BUFSIZ(R6), R3    ;R3 <- index offset from base
             53   D6  0555          INCL    R3                      ;R3 <- index of buffer
  10 BC      53   B0  0557 10$:     MOVW    R3, @INDEX(AP)          ;store index
                      055B
                      055B
                      055B
                      055B DEVFLAG_TEST:
                      055B
                      055B ;determine if there is a device message in this packet
                      055B ;R8 contains actual size of device message
                      055B ;The setting of DEVFLAG is a bit convoluted; it stems from the fact
                      055B ;that there are no spare registers left to hold DEVFLAG'S future value
                      055B ;and relies on the fact that MOVC5 clears R2.
                      055B
     52   FF 8F   90  055B          MOVB    #TRUE, R2                       ;R2 is the complement of DEVFLAG
                      055F                                                  ; (assume no device message)
  0E 1C A7   03   E1  055F          BBC     #XF$V_PKT_FREQPK, -             ;was this packet taken from the
                      0564                  XF$L_PKT_DSL(R7), -             ;FREEQ (does it contain
                      0564                  10$                             ;unsolicited input)?
                      0564                                                  ;if not, goto 10$
          50 A6   D5  0564          TSTL    CTX$L_IDEVMSG(R6)               ;was the array to store the
                      0567                                                  ;device message given?
             09   13  0567          BEQL    10$                             ;no, goto 10$
                      0569
                      0569 ;move the device message field from the packet into the array IDEVMSG,
                      0569 ;which was specified in the call to XF$SETUP
                      0569
00   20 A7   58   2C  0569          MOVC5   R8, XF$B_PKT_DEVMSG(R7), #0,-
  50 B6   5A A6       056E                  CTX$W_IDEVSIZ(R6), @CTX$L_IDEVMSG(R6)
                      0572 10$:     DEFAULT_TEST    <DEVFLAG/4>, TRANSFER_STATUS, LOGFLAG_TEST
                      057C                                      ;if < 5 args goto TRANSFER_STATUS
                      057C                                      ;else if DEVFLAG defaulted goto LOGFLAG_
  14 BC      52   92  057C          MCOMB   R2, @DEVFLAG(AP)                ;set DEVFLAG appropriately
```

```
                                 0580
                                 0580  LOGFLAG_TEST:
                                 0580
                                 0580  ;determine if there is a log message in this packet
                                 0580  ;R10 contains actual size of log message
                                 0580  ;the same note on the setting of DEVFLAG applies to LOGFLAG
                                 0580
            52    FF 8F     90   0580          MOVB    #TRUE, R2                   ;R2 is the complement of LOGFLAG
                                 0584                                              ; (assume no log message)
      0F 1C A7      16      E1   0584          BBC     #<XF$V_PKT_DDISTS+XF$V_PKT_LOG>, -       ;is "log msg in"
                                 0589                  XF$L_PKT_DSL(R7), -         ;bit set in the packet?
                                 0589                  10$                         ;branch if not
              54 A6         D5   0589          TSTL    CTX$L_ILOGMSG(R6)           ;was the array to store the
                                 058C                                              ;device message given?
                    0A      13   058C          BEQL    10$                         ;no, goto 10$
                                 058E
                                 058E  ;move the log message field from the packet into the array ILOGMSG,
                                 058E  ;which was specified in the call to XF$SETUP
                                 058E  ;R9 contains packet size of device message
                                 058E
 58 A6   00   20 A749     5A    2C   058E          MOVC5   R10, XF$B_PKT_DEVMSG(R7)[R9], #0, -
              54 B6              0596
                                 0598                  CTX$W_ILOGSIZ(R6), @CTX$L_ILOGMSG(R6)
                                 0598  10$:    DEFAULT_TEST    <LOGFLAG/4>, TRANSFER_STATUS, TRANSFER_STATUS
                                 05A2                                              ;if LOGFLAG defaulted goto TRANSFER_STAT
         18 BC      52      92   05A2          MCOMB   R2, @LOGFLAG(AP)            ;set LOGFLAG appropriately
                                 05A6
                                 05A6  ;++
                                 05A6  ; return the third through the eighth longword of the command packet
                                 05A6  ; to the user by copying them into CONTXT
                                 05A6  ;--
                                 05A6  TRANSFER_STATUS:
      08 A6    08 A7        7D   05A6          MOVQ    XF$B_PKT_MSGLEN(R7), CTX$L_CONTROL(R6)
                                 05AB                                      ;control longword and byte count
      10 A6    10 A7        7D   05AB          MOVQ    XF$L_PKT_BFRADR(R7), CTX$L_BFRVA(R6)
                                 05B0                                      ;buf addr & residual mem byte count
      18 A6    18 A7        7D   05B0          MOVQ    XF$L_PKT_RDBCNT(R7), CTX$L_DDICNT(R6)
                                 05B5                                      ;residual DDI count  and
                                 05B5                                      ;DR32 status longword
            50    01        3C   05B5          MOVZWL  #SS$_NORMAL, R0             ;success status
```

```
                         05B8  ;++
                         05B8  ;store the status of GETPKT now (if a status argument was given),
                         05B8  ;before the call to the ACTION routine.  GETPKT's status is an input
                         05B8  ;to the ACTION routine.
                         05B8  ;--
                         05B8  STORE_STATUS:
                         05B8          DEFAULT_TEST     <STATUS/4>, 10$, 10$
                         05C2                                      ;if STATUS defaulted goto 10$
      1C BC    50    D0  05C2          MOVL    R0, @STATUS(AP) ;store status
         31    50    E9  05C6  10$:    BLBC    R0, END_GETPKT           ;if no packet, goto end
                         05C9
                         05C9  ;determine size of packet through log message field
                         05C9  ;R9 contains packet size of device message
                         05C9  ;R11 contains packet size of log message field
                         05C9
  53     20 A94B     9E  05C9          MOVAB   XF$B_PKT_DEVMSG(R9)[R11], R3
                         05CE                                                      ;R3 <- devmsg size+logmsg size
                         05CE                                                      ; + size of fixed part of pkt
                         05CE                                                      ;(this is an ADDL, not a MOVA)
                         05CE
                         05CE  ;++
                         05CE  ;IF an ACTION routine is associated with this packet
                         05CE  ;        THEN call it
                         05CE  ;the ACTION routine may substitute its status for GETPKT's status
                         05CE  ;--
                         05CE  ACTION_TEST:
    20 0B A7   02    E1  05CE          BBC     #XF$V_PKT_ACTBIT, XF$B_PKT_PKTCTL(R7), RETURN_SPACE
                         05D3                                                      ;if bit is clear, there is no
                         05D3                                                      ;ACTION routine associated with
                         05D3                                                      ;this packet
                         05D3  ;++
                         05D3  ;R3 contains the size of the packet in bytes, up to and including the
                         05D3  ;log message field.  Add this to the base address of the packet to find
                         05D3  ;the addresses of the ACTION routine and the ACTION
                         05D3  ;routine's parameter.  Then add the size of the two addresses to R3, to
                         05D3  ;calculate the total size of the command packet
                         05D3  ;--
                         05D3
     54    6743      9E  05D3          MOVAB   (R7)[R3], R4                    ;R4 <- addr of addr of ACTION
                         05D7                                                  ;routine
     53      08      C0  05D7          ADDL2   #8, R3                          ;R3 <- total size of packet
                         05DA
                         05DA
                         05DA  ;input arguments to ACTION routine
          1C AC      DD  05DA          PUSHL   STATUS(AP)
          10 AC      DD  05DD          PUSHL   INDEX(AP)
          0C AC      DD  05E0          PUSHL   FUNC(AP)
          18 AC      DD  05E3          PUSHL   LOGFLAG(AP)
          14 AC      DD  05E6          PUSHL   DEVFLAG(AP)
          04 A4      DD  05E9          PUSHL   4(R4)                           ;addr of ACTION routine param
          04 AC      DD  05EC          PUSHL   CONTXT(AP)
      00 B4    07    FB  05EF          CALLS   #7, @(R4)                       ;call user-supplied ACTION
                         05F3                                                  ;routine
                         05F3                                                  ;status returned in STATUS arg
```

```
                           05F3   ;++
                           05F3   ;return the memory the command packet was built from
                           05F3   ;--
                           05F3   RETURN_SPACE:
                           05F3                                                   ;inputs to XF$$DEALOCPKT:
          51   5C A6   DE  05F3           MOVAL   CTX$L_FREELIST(R6), R1          ;R1: addr of freelist header
                           05F7                                                   ;R3: size of packet in bytes
                           05F7                                                   ;R7: addr of packet to return
               FDD8    30  05F7           BSBW    XF$$DEALOCPKT                   ;return the packet space
                           05FA                                                   ;(R1, R3, R7 are destroyed)
                           05FA
                           05FA   END_GETPKT:
                       04  05FA           RET
                           05FB
```

```
05FB          .SBTTL  GET_ADDR -- GET PACKET ADDRESS
05FB     ;++
05FB     ; FUNCTIONAL DESCRIPTION:
05FB     ;
05FB     ;        This routine is called by XF$GETPKT to remove a packet from the
05FB     ;        TERMQ and return its address.  The routine, depending on
05FB     ;        conditions,
05FB     ;        (1) returns with address of packet, or
05FB     ;        (2) returns with status "TERMQ empty", or
05FB     ;        (3) determines that this data transaction has completed, and
05FB     ;            calls XF$CLEANUP before returning
05FB     ;
05FB     ; CALLING SEQUENCE:
05FB     ;
05FB     ;        BSBB/W  GET_ADDR
05FB     ;                called by: XF$GETPKT
05FB     ;                calls (under conditions) XF$CLEANUP
05FB     ;
05FB     ; INPUT PARAMETERS:
05FB     ;
05FB     ;        R2 is a switch that determines what action to take when
05FB     ;        TERMQ is empty
05FB     ;                R2 = 0: wait for event flag
05FB     ;                R2 .NE. 0: immediate return with "TERMQ empty" status
05FB     ; IMPLICIT INPUTS:
05FB     ;
05FB     ;        R6 contains the address of the CONTXT array
05FB     ;        fields in CONTXT:
05FB     ;                CTX$L_CMDBLK
05FB     ;                CTX$Q_IOSB
05FB     ;                CTX$W_EFN
05FB     ;
05FB     ; OUTPUT PARAMETERS:
05FB     ;
05FB     ;        R7 contains address of command packet, if one was successfully
05FB     ;        removed from the TERMQ
05FB     ;
05FB     ; IMPLICIT OUTPUTS:
05FB     ;
05FB     ;        NONE
05FB     ;
```

```
05FB   ; COMPLETION CODES:
05FB   ;
05FB   ;        R0 contains status of call
05FB   ;        status returns:
05FB   ;                (1) SS$_NORMAL: normal successful completion
05FB   ;                                packet address is in R7
05FB   ;                (2) SS$_BADQUEUEHDR:    interlocked queue timeout
05FB   ;                (3) SHR$_HALTED:        XF$CLEANUP was called
05FB   ;                (4) SHR$_QEMPTY: no packet, but transfer still going
05FB   ;                (5) SHR$_NOCMDMEM:      command memory not allocated at
05FB   ;                                        the time this routine was called
05FB   ;                (6) error returns from system calls
05FB   ;                        $WAITFR
05FB   ;                        LIB$FREE_VM
05FB   ;                        LIB$DASSGN
05FB   ;
05FB   ; SIDE EFFECTS:
05FB   ;
05FB   ;        If XF$CLEANUP was called (it is called when the TERMQ is empty
05FB   ;        and the transfer is halted), then the command area was
05FB   ;        deallocated and the device's channel deassigned.
05FB   ;
05FB   ;--
```

```
                          05FB  GET_ADDR:
            0406 8F   BB  05FB          PUSHR   #^M<R1,R2,R10>
       5A     24 A6   D0  05FF          MOVL    CTX$L_CMDBLK(R6), R10   ;R10 <- addr of command block
                 63   13  0603          BEQL    CLEANUP_DONE            ;if 0, command area has been
                          0605                                          ;deallocated
                          0605
                          0605  ;++
                          0605  ;attempt to remove packet from head of TERM queue
                          0605  ;if succeed in removing a packet then goto HAVE_PACKET
                          0605  ;this is partly an otpimization to prevent clearing the event flag when
                          0605  ;there is a packet on the TERMQ and partly a test to see if CLEANUP can
                          0605  ;be done .
                          0605  ;--
                          0605  REM_TERMQ:
                 51   D4  0605          CLRL    R1                      ;retry count
       57     08 AA   5E  0607  10$:    REMQHI  CMD$L_TERMQ(R10), R7    ;R7 <- addr of packet
                 69   1C  060B          BVC     HAVE_PACKET             ;removal succeeded
                 0A   1E  060D          BCC     20$                     ;TERMQ empty
 F0 51   0000C350 8F  F3  060F          AOBLEQ  #RETRY_LIMIT, R1, 10$   ;queue locked. retry.
                          0617
                          0617  ;exceeded retry limit and queue is still locked
                          0617  ;assume queue can no longer be valid
                          0617
                 56   11  0617          BRB     QUEUE_ERROR
                          0619
                          0619  ;++
                          0619  ;there is no packet on the TERMQ; if in addition the transfer is
                          0619  ;halted, then clean up.
                          0619  ;--
                 66   B5  0619  20$:    TSTW    CTX$Q_IOSB(R6)          ;test status of transfer
                 38   12  061B          BNEQ    CLEANUP                 ;br if transfer halted
                          061D
                          061D  ;++
                          061D  ;come here if there is no packet on the TERMQ but the transfer is still
                          061D  ;going.  Test R2 to determine whether to immediately return with
                          061D  ;"TERMQ empty" status or whether to wait for the event flag to be set.
                          061D  ;--
                 52   95  061D          TSTB    R2                      ;wait for event flag?
                 5A   12  061F          BNEQ    TERMQ_EMPTY             ;no, immediate return
                          0621  ;++
                          0621  ;come here to wait for an event flag to be set before
                          0621  ;re-attempting to remove an entry from the TERM queue
                          0621  ;--
                          0621  WAIT_FOR_EF:
       52     42 A6   3C  0621          MOVZWL  CTX$W_EFN(R6), R2       ;get event flag number
                          0625          $CLREF_S        EFN = R2        ;clear event flag
```

```
                    51  D4  062E        CLRL    R1                          ;retry count
           57    08 AA  5E  0630  10$:  REMQHI  CMD$L_TERMQ(R10), R7        ;R7 <- addr of packet
                    40  1C  0634        BVC     HAVE_PACKET                 ;removal succeeded
                    0B  1E  0636        BCC     20$                         ;TERMQ empty
  F0 51   0000C350  8F  F3  0638        AOBLEQ  #RETRY_LIMIT, R1, 10$       ;queue locked. retry.
                            0640
                            0640  ;exceeded retry limit and queue is still locked
                            0640  ;assume queue can no longer be valid
                            0640
                  002C  31  0640        BRW     QUEUE_ERROR
                            0643
                    66  B5  0643  20$:  TSTW    CTX$Q_IOSB(R6)              ;has transfer halted?
                    0E  12  0645        BNEQ    CLEANUP                     ;yes, go clean up
                            0647
                            0647        $WAITFR_S       EFN = R2            ;wait for flag to be set
                 B2 50  E8  0650        BLBS    R0, REM_TERMQ               ;re-attempt a packet
                    2B  11  0653        BRB     END_GET_ADDR                ;R0 contains error status from
                            0655                                            ;WAITFR call
                            0655  ;++
                            0655  ;Come here iff there is nothing on TERMQ and transfer is halted.
                            0655  ;--
                            0655  CLEANUP:
                    56  DD  0655        PUSHL   R6                          ;addr of CONTXT array
       00000685'EF  01  FB  0657        CALLS   #1, XF$CLEANUP              ;(1) deallocates command area
                            065E                                            ;(2) deassigns channel
                 1F 50  E9  065E        BLBC    R0, END_GET_ADDR
           50  1270 8F  3C  0661        MOVZWL  #SHR$_HALTED, R0            ;transfer JUST halted
                    18  11  0666        BRB     END_GET_ADDR
                            0668
                            0668
                            0668  ;status paths
                            0668
                            0668  CLEANUP_DONE:
           50  1278 8F  3C  0668        MOVZWL  #SHR$_NOCMDMEM, R0          ;command area deallocated
                    11  11  066D        BRB     END_GET_ADDR
                            066F
                            066F  QUEUE_ERROR:
           50  0394 8F  3C  066F        MOVZWL  #SS$_BADQUEUEHDR, R0        ;interlock timeout occurred
                    0A  11  0674        BRB     END_GET_ADDR
                            0676
                            0676  HAVE_PACKET:
           50       01  3C  0676        MOVZWL  #SS$_NORMAL, R0             ;packet's address is in R7
                    05  11  0679        BRB     END_GET_ADDR
                            067B
                            067B  TERMQ_EMPTY:
           50  1280 8F  3C  067B        MOVZWL  #SHR$_QEMPTY, R0            ;no packet on TERMQ
                            0680
                            0680  END_GET_ADDR:
                            0680
               0406 8F  BA  0680        POPR    #^M<R1,R2,R10>
                        05  0684        RSB
                            0685
```

```
          0685          .SBTTL  XF$CLEANUP
          0685  ;++
          0685  ; FUNCTIONAL DESCRIPTION:
          0685  ;
          0685  ;       (1) deassign channel
          0685  ;       (2) deallocate virtual memory
          0685  ;
          0685  ; CALLING SEQUENCE:
          0685  ;
          0685  ;       CALLS/G  XF$CLEANUP(CONTXT, [STATUS])
          0685  ;
          0685  ; INPUT PARAMETERS:
          0685  ;
00000004  0685          CONTXT = 4
          0685  ;
          0685  ; IMPLICIT INPUTS:
          0685  ;
          0685  ;       fields in CONTXT array:
          0685  ;               CTX$L_CMDBLK
          0685  ;               CTX$L_CMDSIZ
          0685  ;
          0685  ; OUTPUT PARAMETERS:
          0685  ;
00000008  0685          STATUS = 8              ;optional status word
          0685  ;
          0685  ; IMPLICIT OUTPUTS:
          0685  ;
          0685  ;       RO contains status also (used when XF$GETPKT calls XF$CLEANUP)
          0685  ;
          0685  ; COMPLETION CODES:
          0685  ;
          0685  ;       SS$_NORMAL      -- successful completion
          0685  ;       SHR$_NOCMDMEM   -- command memory was not allocated at the time
          0685  ;                          of this call to XF$CLEANUP
          0685  ;       error returns from:
          0685  ;               LIB$FREE_VM
          0685  ;               $DASSGN
          0685  ;
          0685  ; SIDE EFFECTS:
          0685  ;
          0685  ;       NONE
          0685  ;
          0685  ;--
```

```
                          0685
                   0044   0685          .ENTRY  XF$CLEANUP       ^M<R2,R6>
                          0687
       56    04 AC   D0   0687          MOVL    CONTXT(AP), R6           ;R6 <- addr of CONTXT
                          068B
     50   1278 8F    3C   068B          MOVZWL  #SHR$_NOCMDMEM, R0       ;assume cmd memory not allocated
             24 A6   D5   0690          TSTL    CTX$L_CMDBLK(R6)         ;is address non-zero?
                28   13   0693          BEQL    10$                      ;branch if cmd mem not allocated
                          0695
                          0695  ;deassign channel (also cancels any IO still in progress)
                          0695
  52    00000000'EF  DE   0695          MOVAL   DEVICE_FAB, R2           ;channel number still in FAB
                          069C          $DASSGN_S       CHAN = FAB$L_STV(R2)
                          06A7                                           ;deassign the channel
             13 50   E9   06A7          BLBC    R0, 10$                  ;error from $DASSGN
                          06AA
                          06AA  ;deallocate dynamic virtual memory
                          06AA
             24 A6   DF   06AA          PUSHAL  CTX$L_CMDBLK(R6)         ;address of virtual memory
             20 A6   DF   06AD          PUSHAL  CTX$L_CMDSIZ(R6)         ;size of virtual memory block
  00000000'GF   02   FB   06B0          CALLS   #2, G^LIB$FREE_VM        ;return the memory
             03 50   E9   06B7          BLBC    R0, 10$                  ;error return
             24 A6   D4   06BA          CLRL    CTX$L_CMDBLK(R6)         ;signal command mem returned
                          06BD
                          06BD  ;see if STATUS argument supplied
                          06BD
                          06BD  10$:    DEFAULT_TEST     <STATUS/4>, END_CLEANUP, END_CLEANUP
        08 BC    50  D0   06C7          MOVL     R0, @STATUS(AP) ;store status of call
                          06CB
                          06CB  END_CLEANUP:
                     04   06CB          RET
                          06CC
                          06CC          .END
```

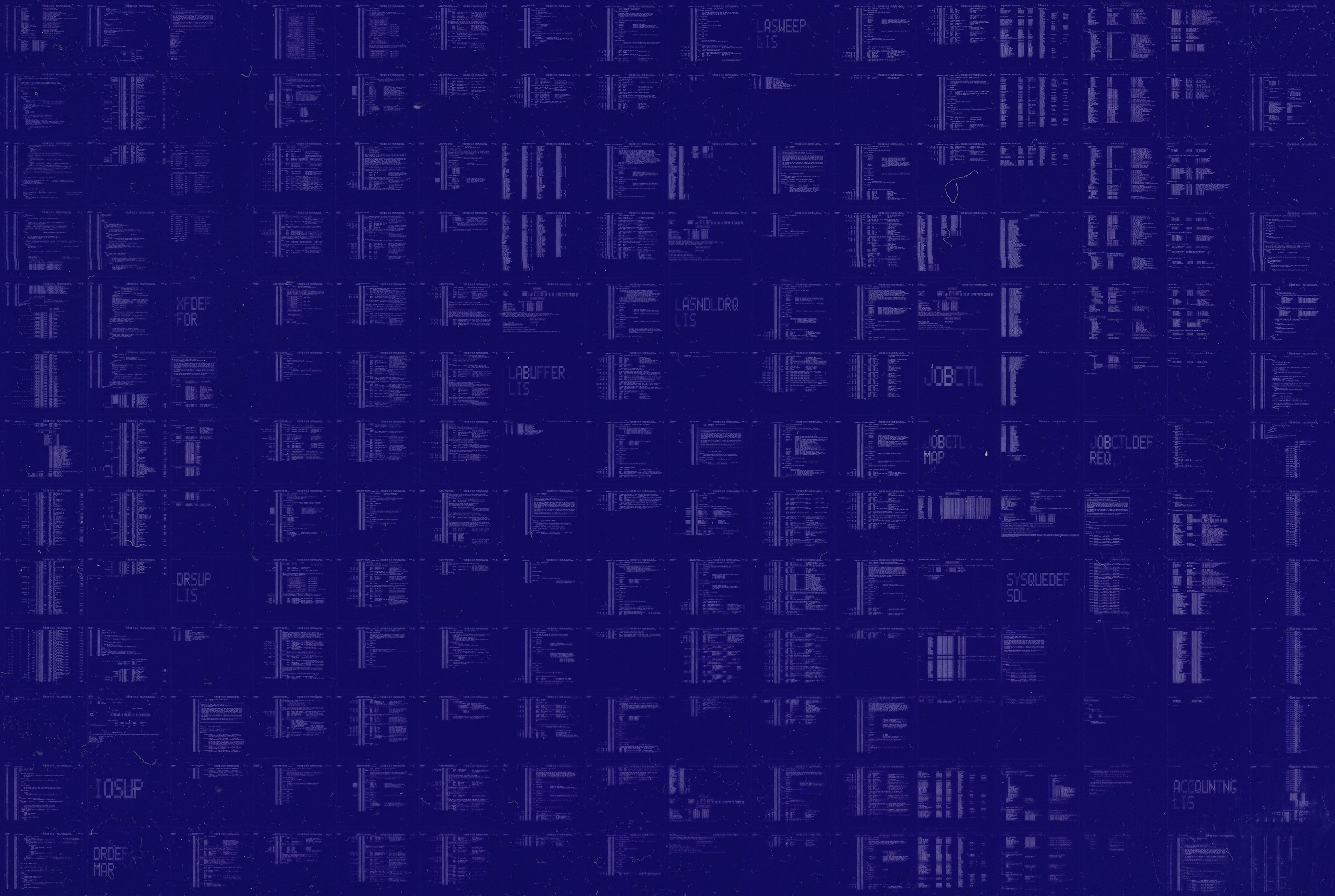
```
$$.TAB                          = 00000000 R     03     CTX$L_PRE_PARM                  00000034
$$.TABEND                       = 00000050 R     03     CTX$Q_IOSB                      00000000
$$.TMP                          = 00000000              CTX$W_EFN                       00000042
$$.TMP1                         = 00000001              CTX$W_IDEVSIZ                   0000005A
$$.TMP2                         = 000000CF              CTX$W_ILOGSIZ                   00000058
$$T1                            = 00000000              CTX$W_NUMBUF                    00000040
ACTION                          = 00000010              DATART                        = 0000001C
ACTION_FIELD                      000002F8 R     02     DATART_DEF                    = 00000000
ACTION_ROUTINE                    00000255 R     02     DATART_TEST                     00000148 R     02
ACTION_TEST                       000005CE R     02     DEALLOCATE                      00000368 R     02
ACTPARM                         = 00000014              DEVFLAG                       = 00000014
ALOC                              00000092 R     02     DEVFLAG_TEST                    0000055B R     02
ALOCCMD                           000000A8 R     02     DEVICE_FAB                      00000000 R     03
ALOCMASK                        = 0000001F              DEVMSG                        = 00000014
ANOTHER_PKT                       0000048A R     02     DEVNAM                        = 00000008
ASSIGN_CHN                        0000015B R     02     DEVSIZ                        = 00000018
ASTPARM                         = 00000010              DIFSIZE                       = 00000010
A_OK                              000004CE R     02     DISSECT_PKT                     00000517 R     02
BADPARM                           000001BF R     02     DUMMY_ADR                     = 00000200
BAD_QUEUE                         000004D6 R     02     EFN                           = 00000014
BARRAY                          = 00000008              EFN_DEF                       = 00000015
BITS                              00000262 R     02     EFN_TEST                        0000012A R     02
BR                                0000024F R     02     END                             000000A7 R     02
BUFSIZ                          = 0000000C              END_ALOCPKT                     000003CF R     02
CLEANUP                           00000655 R     02     END_CLEANUP                     000006CB R     02
CLEANUP_DONE                      00000668 R     02     END_DEALOCPKT                   00000425 R     02
CMD$L_FREEQ                     = 00000010              END_FREESET                     000004E9 R     02
CMD$L_INPTQ                     = 00000000              END_GETPKT                      000005FA R     02
CMD$L_TERMQ                     = 00000008              END_GET_ADDR                    00000680 R     02
CMDSIZ                          = 00000024              END_PKTBLD                      0000038D R     02
CMDSIZ_K                        = 00000003              END_PRE_AST                     00000207 R     02
CMDSIZ_TEST                       0000006C R     02     END_STARTDEV                    000001D0 R     02
COMSIZ                            0000007E R     02     EQUAL                           000003BB R     02
CONTXT                          = 00000004              FAB$B_FNS                     = 00000034
CRITICAL_BIT                    = 00000000              FAB$C_BID                     = 00000003
CRITICAL_MASK                   = 00000001              FAB$C_BLN                     = 00000050
CTX$B_CMDTBL                    = 00000020              FAB$C_SEQ                     = 00000000
CTX$B_CMTFLAGS                    00000039              FAB$C_VAR                     = 00000002
CTX$B_DATART                      00000038              FAB$L_ALQ                     = 00000010
CTX$L_ASTPARM                     00000048              FAB$L_FNA                     = 0000002C
CTX$L_BFRVA                       00000010              FAB$L_FOP                     = 00000004
CTX$L_BUFSIZ                      0000004C              FAB$L_STV                     = 0000000C
CTX$L_BYTECNT                     0000000C              FAB$V_CHAN_MODE               = 00000002
CTX$L_CMDBLK                      00000024              FAB$V_FILE_MODE               = 00000004
CTX$L_CMDSIZ                      00000020              FAB$V_LNM_MODE                = 00000000
CTX$L_CONTROL                     00000008              FAB$V_UFO                     = 00000011
CTX$L_DATABLK                     0000002C              FAB$W_GBC                     = 00000048
CTX$L_DATASIZ                     00000028              FIELDS_DONE                     00000344 R     02
CTX$L_DDICNT                      00000018              FIND_SIZE                       00000437 R     02
CTX$L_DSL                         0000001C              FINISH                          00000099 R     02
CTX$L_FREELIST                    0000005C              FUNC                          = 0000000C
CTX$L_GOBITADR                    0000003C              FUNC_FIELD                      000002A0 R     02
CTX$L_IDEVMSG                     00000050              GET_ADDR                        000005FB R     02
CTX$L_ILOGMSG                     00000054              GO                              00000252 R     02
CTX$L_MEMCNT                      00000014              GRANULARITY                   = 00000007
CTX$L_PKTAST                      00000044              HAVE_PACKET                     00000676 R     02
CTX$L_PRE_AST                     00000030              IDEVMSG                       = 00000014
```

```
                                                        E 7
XF$DRSUP                        -- DR32 SUPPORT ROUTINES            16-SEP-1984 01:45:18  VAX/VMS Macro V04-00        Page  46
Symbol table                                                         5-SEP-1984 01:32:02  [IOSUP.SRC]DRSUP.MAR;1          (50)

IDEVSIZ                         = 00000018                    SYS$DCLAST                         ********   GX    02
ILOGMSG                         = 0000001C                    SYS$QIO                            ********   GX    02
ILOGSIZ                         = 00000020                    SYS$SETAST                         ********   GX    02
IMMEDIATE_EXIT                    000001F0 R       02         SYS$WAITFR                         ********   GX    02
INDEX                           = 00000010                    TERMQ_EMPTY                        0000067B R       02
INDEX_FIELD                       000002BD R       02         TRANSFER_HALTED                    0000037A R       02
INDEX_TEST                        0000053C R       02         TRANSFER_STATUS                    000005A6 R       02
INSERT_AT_HEAD                    00000333 R       02         TRANS_HALTED                       000004E4 R       02
INSERT_AT_TAIL                    00000344 R       02         TRUE                             = 000000FF
INTCTRL                         = 0000000C                    WAITFLG                          = 00000008
INT_DEFAULT                     = 00000000                    WAIT_FOR_EF                        00000621 R       02
INV                               000002B5 R       02         XF$$ALOCPKT                        0000038E RG      02
INVALID_ARG                       0000035E R       02         XF$$DEALOCPKT                      000003D2 RG      02
IO$M_SETEVF                     = 00000040                    XF$B_CMT_FLAGS                   = 00000019
IO$_STARTDATA                   = 00000038                    XF$B_CMT_RATE                    = 00000018
LIB$FREE_VM                       ********   X     02         XF$B_PKT_CMDCTL                  = 0000000A
LIB$GET_VM                        ********   X     02         XF$B_PKT_DEVMSG                  = 00000020
LOGFLAG                         = 00000018                    XF$B_PKT_LOGLEN                  = 00000009
LOGFLAG_TEST                      00000580 R       02         XF$B_PKT_MSGLEN                  = 00000008
LOGSIZ                          = 0000001C                    XF$B_PKT_PKTCTL                  = 0000000B
LOGSIZE                           00000233 R       02         XF$CLEANUP                         00000685 RG      02
MODES                           = 00000020                    XF$FREESET                         00000428 RG      02
MODES_DEFAULT                   = 00000000                    XF$GETPKT                          000004F8 RG      02
MODES_FIELD                       0000031B R       02         XF$K_CMT_LENGTH                  = 00000020
MODE_TEST                         00000139 R       02         XF$L_CMT_GBITAD                  = 0000001C
MSG_ARRAYS                        0000002A R       02         XF$L_CMT_PASTAD                  = 00000010
NEXT                              00000282 R       02         XF$L_CMT_PASTPM                  = 00000014
NOT_MEM                           000004DD R       02         XF$L_PKT_BFRADR                  = 00000010
NO_MEM                            00000373 R       02         XF$L_PKT_BFRSIZ                  = 0000000C
NUMBUF                          = 00000010                    XF$L_PKT_DSL                     = 0000001C
NUMPKT                          = 00000008                    XF$L_PKT_RDBCNT                  = 00000018
OK                                000002B8 R       02         XF$M_CMT_SETRTE                  = 00000001
PAGEMASK                        = 000001FF                    XF$M_PKT_ACTBIT                  = 00000004
PKTAST                          = 0000000C                    XF$PKTBLD                          00000208 RG      02
PKTAST_TEST                       000000FF R       02         XF$SETUP                           00000000 RG      02
PRE_AST                           000001D1 R       02         XF$STARTDEV                        000000E7 RG      02
QUADWORD_MASK                   = 00000007                    XF$S_PKT_INTCTL                  = 00000002
QUEUE_ERROR                       0000066F R       02         XF$V_PKT_ACTBIT                  = 00000002
Q_FAILURE                         00000363 R       02         XF$V_PKT_DDISTS                  = 00000010
REM_TERMQ                         00000605 R       02         XF$V_PKT_FREQPK                  = 00000003
RETRY_LIMIT                     = 0000C350                    XF$V_PKT_HT                      = 00000008
RETURN_SPACE                      000005F3 R       02         XF$V_PKT_INTCTL                  = 00000006
SET_GO_BIT                        00000355 R       02         XF$V_PKT_LOG                     = 00000006
SHR$_HALTED                     = 00001270
SHR$_NOCMDMEM                   = 00001278
SHR$_QEMPTY                     = 00001280
SS$_BADPARAM                    = 00000014
SS$_BADQUEUEHDR                 = 00000394
SS$_INSFMEM                     = 00000124
SS$_NORMAL                      = 00000001
STAT                              000001C2 R       02
STATUS                          = 00000008
STORE_STAT                        0000037F R       02
STORE_STATUS                      000005B8 R       02
SYS$CLREF                         ********   GX    02
SYS$CREATE                        ********   GX    02
SYS$DASSGN                        ********   GX    02
```

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

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                           00000060  (   96.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
_XF$CODE                        000006CC  ( 1740.)  02 (  2.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD  NOWRT NOVEC BYTE
_XF$DATA                        00000050  (   80.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD    WRT NOVEC BYTE

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

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     34    00:00:00.10    00:00:00.52
Command processing                157    00:00:00.58    00:00:01.96
Pass 1                            396    00:00:15.45    00:00:31.25
Symbol table sort                   0    00:00:02.10    00:00:04.31
Pass 2                            327    00:00:04.87    00:00:09.06
Symbol table output                27    00:00:00.20    00:00:01.01
Psect synopsis output               2    00:00:00.03    00:00:00.03
Cross-reference output              0    00:00:00.00    00:00:00.00
Assembler run totals              945    00:00:23.33    00:00:48.15

The working set limit was 1950 pages.
90665 bytes (178 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1377 non-local and 33 local symbols.
1917 source lines were read in Pass 1, producing 31 object records in Pass 2.
34 pages of virtual memory were used to define 29 macros.

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

Macro library name                         Macros defined
------------------                         --------------
_$255$DUA28:[IOSUP.SRC]DRDEF.MLB;1                   2
_$255$DUA28:[SYSLIB]STARLET.MLB;2                   22
TOTALS (all libraries)                              24

1597 GETS were required to define 24 macros.

There were no errors, warnings or information messages.

MACRO/DISABLE=TRACE/LIS=LIS$:DRSUP/OBJ=OBJ$:DRSUP MSRC$:DRSUP/UPDATE=(ENH$:DRSUP)+SRC$:DRDEF/LIB
```

0190 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY


LASWEEP
LIS

XFDEF
FOR

LASNDLDRQ
LIS

LABUFFER
LIS

JOBCTL

JOBCTL
MAP

JOBCTLDEF
REQ

DRSUP
LIS

SYSQUEDEF
SDL

IOSUP

ACCOUNTNG
LIS

DRDEF
MAR