N 3

```
EEEEEEEEEE  XX      XX    AAAAAA    MM      MM  PPPPPPPP    LL          EEEEEEEEEE    SSSSSSSS
EEEEEEEEEE  XX      XX    AAAAAA    MM      MM  PPPPPPPP    LL          EEEEEEEEEE    SSSSSSSS
EEEEEEEEEE  XX      XX    AAAAAA    MM      MM  PPPPPPPP    LL          EEEEEEEEEE    SSSSSSSS
EE          XX      XX  AA      AA  MMMM  MMMM  PP      PP  LL          EE            SS
EE          XX      XX  AA      AA  MMMM  MMMM  PP      PP  LL          EE            SS
EE          XX      XX  AA      AA  MMMM  MMMM  PP      PP  LL          EE            SS
EE            XX  XX    AA      AA  MM  MM  MM  PP      PP  LL          EE            SS
EE            XX  XX    AA      AA  MM  MM  MM  PP      PP  LL          EE            SS
EE            XX  XX    AA      AA  MM  MM  MM  PP      PP  LL          EE            SS
EEEEEEEE        XX      AA      AA  MM      MM  PPPPPPPP    LL          EEEEEEEE        SSSSSS
EEEEEEEE        XX      AA      AA  MM      MM  PPPPPPPP    LL          EEEEEEEE        SSSSSS
EEEEEEEE        XX      AA      AA  MM      MM  PPPPPPPP    LL          EEEEEEEE        SSSSSS
EE            XX  XX    AAAAAAAAAA  MM      MM  PP          LL          EE                    SS
EE            XX  XX    AAAAAAAAAA  MM      MM  PP          LL          EE                    SS
EE            XX  XX    AAAAAAAAAA  MM      MM  PP          LL          EE                    SS
EE          XX      XX  AA      AA  MM      MM  PP          LL          EE                    SS
EE          XX      XX  AA      AA  MM      MM  PP          LL          EE                    SS
EE          XX      XX  AA      AA  MM      MM  PP          LL          EE                    SS
EEEEEEEEEE  XX      XX  AA      AA  MM      MM  PP          LLLLLLLLLL  EEEEEEEEEE  SSSSSSSS
EEEEEEEEEE  XX      XX  AA      AA  MM      MM  PP          LLLLLLLLLL  EEEEEEEEEE  SSSSSSSS
EEEEEEEEEE  XX      XX  AA      AA  MM      MM  PP          LLLLLLLLLL  EEEEEEEEEE  SSSSSSSS
```

LPN

```
**FILE**ID**DRMASTER                                    C  4
```

```
DDDDDDDD    RRRRRRRR    MM      MM    AAAAAA      SSSSSSSS  TTTTTTTTTT  EEEEEEEEEE  RRRRRRRR
DDDDDDDD    RRRRRRRR    MM      MM    AAAAAA      SSSSSSSS  TTTTTTTTTT  EEEEEEEEEE  RRRRRRRR
DD       DD RR      RR  MMMM  MMMM  AA      AA  SS              TT      EE          RR      RR
DD       DD RR      RR  MMMM  MMMM  AA      AA  SS              TT      EE          RR      RR
DD       DD RR      RR  MM  MM  MM  AA      AA  SS              TT      EE          RR      RR
DD       DD RR      RR  MM  MM  MM  AA      AA  SS              TT      EE          RR      RR
DD       DD RRRRRRRR    MM      MM  AA      AA    SSSSSS        TT      EEEEEEEE    RRRRRRRR
DD       DD RRRRRRRR    MM      MM  AA      AA    SSSSSS        TT      EEEEEEEE    RRRRRRRR
DD       DD RR  RR      MM      MM  AAAAAAAAAA          SS      TT      EE          RR  RR
DD       DD RR  RR      MM      MM  AAAAAAAAAA          SS      TT      EE          RR  RR
DD       DD RR    RR    MM      MM  AA      AA          SS      TT      EE          RR    RR        ....
DD       DD RR    RR    MM      MM  AA      AA          SS      TT      EE          RR    RR        ....
DDDDDDDD    RR      RR  MM      MM  AA      AA  SSSSSSSS        TT      EEEEEEEEEE  RR      RR      ....
DDDDDDDD    RR      RR  MM      MM  AA      AA  SSSSSSSS        TT      EEEEEEEEEE  RR      RR      ....

FFFFFFFFFF    OOOOOO    RRRRRRRR
FFFFFFFFFF    OOOOOO    RRRRRRRR
FF           OO      OO  RR      RR
FF           OO      OO  RR      RR
FF           OO      OO  RR      RR
FF           OO      OO  RR      RR
FFFFFFFF     OO      OO  RRRRRRRR
FFFFFFFF     OO      OO  RRRRRRRR
FF           OO      OO  RR  RR
FF           OO      OO  RR  RR
FF           OO      OO  RR    RR
FF           OO      OO  RR    RR
FF            OOOOOO    RR      RR
FF            OOOOOO    RR      RR
```

```
C       DRMASTER
C       Version 'V04-000'
C
C
C*****************************************************************************
C*                                                                           *
C*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
C*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
C*  ALL RIGHTS RESERVED.                                                     *
C*                                                                           *
C*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
C*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
C*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
C*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
C*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
C*  TRANSFERRED.                                                             *
C*                                                                           *
C*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
C*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
C*  CORPORATION.                                                             *
C*                                                                           *
C*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
C*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
C*                                                                           *
C*                                                                           *
C*****************************************************************************
C

C++
C FACILITY:      DRCOPY -- EXAMPLE PROGRAM FOR DR32
C
C ABSTRACT:
C       This set of routines constitutes the Master portion of the
C       DRCOPY file transfer example program.
C       For more information on the DR32 and how it is supported by
C       VAX/VMS, see Chapter 11 of the VAX/VMS I/O Users' Guide.
C
C ENVIRONMENT:
C       These programs run in User mode; no privileges are needed.
C
C AUTHOR:  Steve Beckhardt,     CREATION DATE: July, 1979
C
C MODIFIED BY:
C
C
C       , : VERSION
C 01    -
C--
```

```
C ****************************************************************************
C       TO RUN DRCOPY:
C ****************************************************************************
C
C       DRCOPY requires two CPUs and two DR32s; the DR32s form the
C communications path between the two CPUs.
C       To run DRCOPY, type the following commands on BOTH CPUs:
C
C       $ SET DEFAULT SYS$SYSDISK:[SYSHELP.EXAMPLES]
C       $ @DRCOPYBLD     ! if necessary, to create image file.
C       $ RUN DRCOPY
C
C       A prompt, "DRCOPY>", should appear.  To get a description of the valid
C DRCOPY file transfer commands, type "HELP" in response to the prompt.
C       In order to use DRCOPY, both CPUs must be running the DRCOPY program
C (i.e. a terminal on each CPU should be waiting at the "DRCOPY>" prompt).
C
C ****************************************************************************
C       THE FOLLOWING SECTIONS ARE INCLUDED AS AN AID TO UNDERSTANDING THE
C       IMPLEMENTATION OF THE DRCOPY PROGRAM.
C ****************************************************************************
C
C       This set of routines is used to implement a CPU - to - CPU file
C transfer protocol using the DR32.  The goals are to implement the
C protocol (excluding the data source and data sink routines) in
C FORTRAN, using the library of high-level support routines provided
C in VMS Release 2 for the DR780.**** Please read Chapter 11 of the VAX/VMS
C I/O User's Guide before trying to understand this material. ****
C
C
C               THE MASTER ROUTINES AND THE SLAVE ROUTINES
C
C       In DRCOPY's model of the world, there exists a Master program
C in one CPU and a Slave program in the other.  The Master always
C initiates file transfers, and the direction of the file transfers are
C defined from the Master's point of view (i.e. a 'read' or a 'get'
C operation means 'transfer a file from the Slave to the Master', while
C a 'put' or a 'write' operation transfers a file from the Master to the
C Slave).
C       While it is convenient to think of one CPU as the "Master", and the
C other as the "Slave", in reality both images of DRCOPY contain a set of
C routines that are collectively called the Master routines and a set of
C routines called the Slave routines.  During discussions of a transfer, the
C Master CPU is the CPU currently executing the Master routines; the Slave CPU
C is currently executing the Slave routines.  But since both images contain
C both sets of routines, either CPU can potentially be the Master or the Slave;
C in fact, both CPUs can be Master and Slave simultaneously.
C
C       DRCOPY on CPU A                         DRCOPY on CPU B
C       +---------------+                       +---------------+
C       |   MASTER      |---------    ----------|   MASTER      |
C       |   routines    |         \/            |   routines    |
C       |---------------|         /\            |---------------|
C       |   SLAVE       |        /  \           |   SLAVE       |
C       |   routines    |<--------    --------->|   routines    |
C       +---------------+                       +---------------+
C
C       The files being tranferred are assumed to be on disk, and too big
C (or too something) to be locked down into memory during the transfer, so
C they are buffered in main memory.  The 'source' side of any given transfer
C (the Master during a PUT and the Slave during a GET) is involved in two
C asynchronous processes: (1) filling its ring of buffers from disk; and
C (2) shipping filled buffers via the DR32 to the far-end CPU.
C       The receiving side is in turn: (1) obtaining buffers full of data from
C the far-end CPU and (2) emptying the buffers back out to its own disk.  The
C transfer of data between disk and memory will be called the RMS process;
C the transfer of data from one CPU's memory to the other's will be called the
C DR-transfer process.
C
C
C                       THE MAIN ROUTINES
C
C
C       PARSE     \
C       GET_TOKEN - command interface routines
C       HELP      /
C
C       DO_PUT  -Top level Master routine for copying a file from local CPU
C                to remote CPU.
C
C       DO_GET  -Top level Master routine for copying a file from remote CPU
C                to local CPU.
C
C
C                       THE AST ROUTINES
C
C       MRMS_AST -Master RMS AST completion routine
C       SLV_BUFDONE -Slave RMS AST completion routine
C
C       PKT_AST -Called when a completed DR32 command packet is placed on an
C                empty termination queue.  Call XF$GETPKT until TERMQ is empty.
C                XF$GETPKT will call the action routine associated with each
C                packet as it removes that packet from TERMQ.
C
C
C                       THE ACTION ROUTINES
C
C       When building command packets, the Master routines specify different
C       action routines depending on the function this command packet will
C       perform. The Master also specifies a special action routine for
C       command packets that it loads on the free queue.
C
C       ACT_NOPPKT -Called when a command packet specifying a NOP function
C               completes.
C       ACT_RWPKT -Called when a read or write packet completes.
C
C       ACT_FREQUE -This is the action routine address built into packets
C               released onto the free queue; it is called after the DR32
C               stores a command/device message from the far-end device
C               into a packet from the free queue and then inserts that
C               packet onto the termination queue.
```

```
C       According to the protocol defined for DRCOPY, the first longword of all
C       device messages is a type code.  ACT_FREQUE dispatches to the routine
C       associated with each type code.  The type codes fall into two main
C       categories: those whose names begin with MS_MSG are messages from the
C       Master to the Slave;  those whose names begin with SM_MSG are messages
C       from the Slave to the Master.  For instance,  the type code
C       MS_MSG_STARTPUT is a message from the Master informing the Slave that
C       a PUT operation is to be initiated.
C
C       Slave routines are only invoked in response to device messages from the
C       Master side.  (There is one exception to this: after a message from the
C       Master starts up the Slave's RMS process, that process proceeds without
C       coordination from the Master while there are buffers available to it.)
C       The Slave routines respond to device messages from the far-end Master
C       routines, but require the local Master routines to remove the packet
C       (containing the far-end device message) from the termination queue
C       and to call the appropriate Slave routine according to the type of
C       device message.
C
C
C
C                       SLAVE FREE QUEUE ROUTINES (SFQ_)
C
C       SFQ_STARTGET -Called when the Slave receives an MS_MSG_STARTGET
C               message;  Slave opens the file and sends its attributes
C               back to the Master.  Slave also sends the addresses of
C               its buffers.
C       SFQ_GOGET -Called when Master signals that he received file attributes,
C               opened the file, and is ready to accept data; Slave issues
C               an RMS read to get things going.
C       SFQ_STARTPUT -Called when the Slave receives an MS_MSG_STARTPUT
C               message; Slave creates the file, sends back the addresses
C               of its buffers, and waits for data from Master.
C       SFQ_PNXTBFR -Called when the Slave receives a ''process your next
C               buffer'' message.
C       SFQ_PLSTBFR -This message is only sent during a PUT operation; it means
C               the last buffer to be written to disk has arrived.
C
C
C
C                       MASTER FREE QUEUE ROUTINES (MFQ_)
C
C       MFQ_BFRADS -Process list of buffer addresses sent by Slave.
C       MFQ_FILEATTR -Copy attributes of file opened by Slave.
C       MFQ_PNXTBFR -Called when Slave sends message that it has processed
C               another buffer; this means another buffer is available
C               to the Master.
C       MFQ_PLSTBFR -Called when Slave sends a message that it has processed
C               its last buffer;  if GET, read last buffer; if PUT,
C               transfer is complete - wake main level.
C       MFQ_ERROR -Called when Slave sends error message.
```

```
C
C DRMASTER -- the Master portion of the DRCOPY example program
C
        INCLUDE 'SYS$LIBRARY:XFDEF.FOR/NOLIST'     ! DR32 definitions
        INCLUDE 'DRCOPY.PRM'                       ! Parameters

C
C       Local Variables
C
        INTEGER*4 STATUS

C
C       Common variables and areas
C
        CHARACTER*80  INPLINE                      ! Input line
        CHARACTER*64  LOC_FNAME                    ! Local file name
        CHARACTER*64  REM_FNAME                    ! Remote file name

        COMMON  /CHARS/  INPLINE,LOC_FNAME,REM_FNAME

        INTEGER*2  LOC_FNSIZE                      ! Local file name size
        INTEGER*2  REM_FNSIZE                      ! Remote file name size
        INTEGER*2  SPOS                            ! Starting token pos.
        INTEGER*2  EPOS                            ! Ending token pos.

        COMMON  /SIZES/  LOC_FNSIZE,REM_FNSIZE,SPOS,EPOS

        INTEGER*4 XFDATA(30)                       ! Context array
        BYTE MBFRS(BUFSIZ,NUM_MBFRS)               ! Master buffers
        BYTE SBFRS(BUFSIZ,NUM_SBFRS)               ! Slave buffers

        COMMON /MS_SHARE/  XFDATA,MBFRS,SBFRS

        INTEGER*4 IDEVMSG(32)                      ! Incoming device messages
        INTEGER*4 ODEVMSG(32)                      ! Outgoing device messages
        INTEGER*4 REM_BFRADS(25)                   ! Remote buffer addresses
        INTEGER*4 FILEATTR(6)                      ! File attributes
        INTEGER*4 CSTATUS                          ! Common status
        INTEGER*4 LASTBFRSIZ                       ! Last buffer size
        INTEGER*4 DDIDIS                           ! DDI disable
        INTEGER*2 MRMS_CNT                         ! Master RMS count
        INTEGER*2 MRMS_IDX                         ! Master RMS index
        INTEGER*2 QPKT_CNT                         ! Queue packet count
        INTEGER*2 QPKT_IDX                         ! Queue packet index
        INTEGER*2 REM_CNT                          ! Remote buffer count
        INTEGER*2 REM_IDX                          ! Remote buffer index
        INTEGER*2 NUMREM_BFRS                      ! Number of remote buffers
        LOGICAL*1 GPFLAG                           ! Get/put flag
        LOGICAL*1 LASTBFR                          ! Last buffer flag
        LOGICAL*1 EOFFLAG                          ! End of file flag
        LOGICAL*1 ERRFLAG                          ! Error flag
        LOGICAL*1 REMFLAG                          ! Remote error flag

        COMMON /MDATA/  IDEVMSG,ODEVMSG,REM_BFRADS,FILEATTR,CSTATUS,
        1               LASTBFRSIZ,DDIDIS,MRMS_CNT,MRMS_IDX,QPKT_CNT,
        2               QPKT_IDX,REM_CNT,REM_IDX,NUMREM_BFRS,GPFLAG,
        3               LASTBFR,EOFFLAG,ERRFLAG,REMFLAG

        INTEGER*4 SYS$CLREF
        INTEGER*4 SYS$SETEF
        INTEGER*4 SYS$WAITFR

        EXTERNAL ACT_FREQUE
        EXTERNAL ACT_NOPPKT
        EXTERNAL PKT_AST


C
C       Set event flag 2.  This is used by the slave half to indicate
C       when it is active so that we don't exit while the slave is active.
C
        STATUS = SYS$SETEF(%VAL(5))
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)

C
C       Start the DR32.  This involves three calls.  One to set
C       up everything, one to initialize the free queue with empty
C       packets, and one to actually start the DR32.
C
        CALL XF$SETUP(XFDATA,                   ! Context array
        1             MBFRS,                    ! Data buffers
        2             BUFSIZ,                   ! Data buffer size
        3             NUM_MBFRS + NUM_SBFRS,    ! Number of data buffers
        4             IDEVMSG,128,              ! Incoming device msg array and size
        5             ,,,                       ! No log area
        6             STATUS)                   ! Status
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)

        CALL XF$FREESET(XFDATA,                 ! Context array
        1               NUM_MBFRS + NUM_SBFRS,  ! Number of Free Q packets
        2               1,                      ! AST if Term Q empty
        3               ACT_FREQUE,,            ! Action routine, no param.
        4               STATUS)                 ! Status
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)

        CALL XF$STARTDEV(XFDATA,                ! Context array
        1                'XFA0:',               ! Device name
        2                PKT_AST,,,,            ! AST routine
        3                DATARATE,              ! Data rate
        4                STATUS)                ! Status
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)

C
C       Enable random access mode (device initiated transfers)
C
        STATUS = SYS$CLREF(%VAL(1))             ! Clear event flag
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)

        CALL XF$PKTBLD(XFDATA,
        1              XF$K_PKT_SETRND,         ! Set random enable
        2              .....
        3               64+256,                 ! Ins @ head, Int. if empty
        4               ACT_NOPPKT,,            ! Action routine, parm
        5               STATUS)                 ! Status
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)
        STATUS = SYS$WAITFR(%VAL(1))            ! Wait for packet
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)

C
C       Get the command
C
500     WRITE(6,600)                            ! Prompt for input
600     FORMAT(' DRCOPY> ',$)
        READ(5,650,END=8000)INPLINE             ! Get input line
650     FORMAT(A80)
        CALL PARSE(GPFLAG)                      ! Parse it
        IF(.NOT. GPFLAG) GO TO 500              ! No command, repeat
D       WRITE(6,700)GPFLAG,LOC_FNAME(1:LOC_FNSIZE),REM_FNAME(1:REM_FNSIZE)
D 700   FORMAT(1X,'GPFLAG = ',I1,',  LOCAL FILE NAME = ',A,
D       1  ',  REMOTE FILE NAME = ',A)

C
C       Do the requested operation
C
        IF (GPFLAG .EQ. 1) THEN
            CALL DO_GET
        ELSE
            CALL DO_PUT
        END IF
        GO TO 500

C
C       Wait until slave half finishes before exiting
C
8000    STATUS = SYS$WAITFR(%VAL(5))
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)

9000    END
```

```
        SUBROUTINE PARSE(GPFLAG)

C       This subroutine parses the input line into a command,
C       a local filename, and a remote filename.

C
C       Local Variables
C
        INTEGER*4 GPFLAG                                ! Get/Put flag

C
C       Common variables and areas
C
        CHARACTER*80  INPLINE                           ! Input line
        CHARACTER*64  LOC_FNAME                         ! Local file name
        CHARACTER*64  REM_FNAME                         ! Remote file name

        COMMON  /CHARS/  INPLINE,LOC_FNAME,REM_FNAME

        INTEGER*2  LOC_FNSIZE                           ! Local file name size
        INTEGER*2  REM_FNSIZE                           ! Remote file name size
        INTEGER*2  SPOS                                 ! Starting token pos.
        INTEGER*2  EPOS                                 ! Ending token pos.

        COMMON  /SIZES/  LOC_FNSIZE,REM_FNSIZE,SPOS,EPOS


C
C       Raise lowercase characters to uppercase
C
        DO 1000 I = 1,80
        J = ICHAR(INPLINE(I:I))                         ! Get next character
        IF (J.GE.'61'X .AND. J.LE.'7A'X) THEN           ! If its between a and z
            J = J - '20'X                               ! make it between A and Z
            INPLINE(I:I) = CHAR(J)                      ! Replace it in input line
        END IF
1000    CONTINUE

C
C       Get command
C
        SPOS = 1                                        ! Starting position
        CALL GET_TOKEN                                  ! Get next token
        IF (SPOS .LT. 0) GO TO 8000                     ! Nothing on line
        IF(INPLINE(SPOS:EPOS-1) .EQ. 'GET') THEN
            GPFLAG = 1
        ELSE IF (INPLINE(SPOS:EPOS-1) .EQ. 'PUT') THEN
            GPFLAG = 3
        ELSE IF (INPLINE(SPOS:EPOS-1) .EQ. 'HELP') THEN
            CALL HELP
            GO TO 8000
        ELSE
            GO TO 7000                                  ! Syntax error
        END IF
C
C       Get local filename
```

```
C
        SPOS = EPOS
        CALL GET_TOKEN
        IF (SPOS .LT. 0) GO TO 7000                     ! Syntax error
        LOC_FNAME = INPLINE(SPOS:EPOS-1)                ! Extract filename
        LOC_FNSIZE = EPOS - SPOS                        ! and get size
C
C       Process 'TO' or 'FROM'
C
        SPOS = EPOS
        CALL GET_TOKEN                                  ! Get next token
        IF (SPOS .LT. 0) GO TO 4000                     ! Remote filename defaulted
        IF (GPFLAG .EQ. 1  .AND.  INPLINE(SPOS:EPOS-1) .NE. 'FROM')
        1   GO TO 7000                                  ! Syntax error
        IF (GPFLAG .EQ. 3  .AND.  INPLINE(SPOS:EPOS-1) .NE. 'TO')
        1   GO TO 7000                                  ! Syntax error

C
C       Get remote filename
C
        SPOS = EPOS
        CALL GET_TOKEN                                  ! Get next token
        IF (SPOS .LT. 0) GO TO 7000                     ! Syntax error
        REM_FNAME = INPLINE(SPOS:EPOS-1)                ! Extract filename
        REM_FNSIZE = EPOS - SPOS                        ! and get size

C
C       Make sure rest of line is empty
C
        SPOS = EPOS
        CALL GET_TOKEN
        IF(SPOS .GE. 0) GO TO 7000                      ! Syntax error

C
C       If either filename is '*', use the other name
C
        IF (REM_FNAME(1:REM_FNSIZE) .EQ. '*') GO TO 4000
        IF (LOC_FNAME(1:LOC_FNSIZE) .NE. '*') GO TO 9000

        LOC_FNAME = REM_FNAME                           ! Local filename = *
        LOC_FNSIZE = REM_FNSIZE
        GO TO 9000

4000    IF (LOC_FNAME(1:LOC_FNSIZE) .EQ. '*') GO TO 7000
        REM_FNAME = LOC_FNAME                           ! Remote filename = *
        REM_FNSIZE = LOC_FNSIZE
        GO TO 9000

C
C       Syntax error
C
7000    WRITE(6,7100)
7100    FORMAT(1X,'%DRCOPY-E-SYNTAX, syntax error on command line')

8000    GPFLAG = 0
```

```
9000    RETURN
        END
```

```
DRM

C
C
C
C

C
C
C
C
C
C

400
450

C
C
C
C
C
C
C
C
500

C
```

```
        SUBROUTINE GET_TOKEN
C
C       This subroutine gets the next token on the input line.
C
C       Inputs:
C               SPOS - Starting character position
C
C       Outputs:
C               SPOS - Starting position of token
C               EPOS - One character after end of token
C
C       If there are no more tokens on the line SPOS is set to -1
C

C
C       Common variables and areas
C
        CHARACTER*80  INPLINE                   ! Input line
        CHARACTER*64  LOC_FNAME                 ! Local file name
        CHARACTER*64  REM_FNAME                 ! Remote file name

        COMMON  /CHARS/  INPLINE,LOC_FNAME,REM_FNAME

        INTEGER*2  LOC_FNSIZE                   ! Local file name size
        INTEGER*2  REM_FNSIZE                   ! Remote file name size
        INTEGER*2  SPOS                         ! Starting token pos.
        INTEGER*2  EPOS                         ! Ending token pos.

        COMMON  /SIZES/  LOC_FNSIZE,REM_FNSIZE,SPOS,EPOS

C
C       Return immediately if SPOS is past end of line
C
        IF (SPOS .GT. 80) GO TO 400

C
C       Skip leading blanks
C
        DO 100 SPOS = SPOS,80
        IF (INPLINE(SPOS:SPOS) .NE. ' ')GO TO 200
100     CONTINUE
        GO TO 400                               ! No more tokens

C
C       SPOS points to start of token.  Now find first blank after token
C
200     DO 300 EPOS = SPOS,80
        IF (INPLINE(EPOS:EPOS) .EQ. ' ') GO TO 500
300     CONTINUE
        GO TO 500

400     SPOS = -1                               ! No more tokens

500     RETURN
```

DRM

```
C
C
C
600

C
C
C
C
C
C
700
```

```
      END
```

```
C
C
C
C
C

C
C
C

C
C
C

C
C
```

```
        SUBROUTINE HELP
C
C       This subroutine prints out the HELP message
C

        WRITE(6,100)
        WRITE(6,200)
        WRITE(6,300)
        WRITE(6,400)
100     FORMAT('0','The commands to DRCOPY are:')
200     FORMAT('0','     GET filespec1 [FROM filespec2]'/
        1  ' ','     PUT filespec1 [TO filespec2]')
300     FORMAT('0','     filespec1 is always the local filename'/
        1  ' ','     filespec2 is always the remote filename')
400     FORMAT('0','If either filespec is specified as *, the other ',
        1  'filespec is used for both.'/' If the second half of the ',
        2  'command is omitted, filespec1 is used for filespec2.'/)

        RETURN
        END
        SUBROUTINE DO_PUT

C
C       This routine is the top level routine for copying a file
C       from the local cpu to the remote cpu.
C

        INCLUDE 'SYS$LIBRARY:XFDEF.FOR/NOLIST'    ! DR32 definitions
        INCLUDE 'DRCOPY.PRM/NOLIST'               ! Parameters

C
C       Local Variables
C
        INTEGER*4 STATUS                          ! Local status

C
C       Common variables and areas
C
        CHARACTER*80  INPLINE                     ! Input line
        CHARACTER*64  LOC_FNAME                   ! Local file name
        CHARACTER*64  REM_FNAME                   ! Remote file name

        COMMON  /CHARS/  INPLINE,LOC_FNAME,REM_FNAME

        INTEGER*2  LOC_FNSIZE                     ! Local file name size
        INTEGER*2  REM_FNSIZE                     ! Remote file name size
        INTEGER*2  SPOS                           ! Starting token pos.
        INTEGER*2  EPOS                           ! Ending token pos.

        COMMON  /SIZES/  LOC_FNSIZE,REM_FNSIZE,SPOS,EPOS

        INTEGER*4 XFDATA(30)                      ! Context array
        BYTE MBFRS(BUFSIZ,NUM_MBFRS)              ! Master buffers
        BYTE SBFRS(BUFSIZ,NUM_SBFRS)              ! Slave buffers

        COMMON /MS_SHARE/  XFDATA,MBFRS,SBFRS

        INTEGER*4 IDEVMSG(32)                     ! Incoming device messages
        INTEGER*4 ODEVMSG(32)                     ! Outgoing device messages
        INTEGER*4 REM_BFRADS(25)                  ! Remote buffer addresses
        INTEGER*4 FILEATTR(6)                     ! File attributes
        INTEGER*4 CSTATUS                         ! Common status
        INTEGER*4 LASTBFRSIZ                      ! Last buffer size
        INTEGER*4 DDIDIS                          ! DDI disable
        INTEGER*2 MRMS_CNT                        ! Master RMS count
        INTEGER*2 MRMS_IDX                        ! Master RMS index
        INTEGER*2 QPKT_CNT                        ! Queue packet count
        INTEGER*2 QPKT_IDX                        ! Queue packet index
        INTEGER*2 REM_CNT                         ! Remote buffer count
        INTEGER*2 REM_IDX                         ! Remote buffer index
        INTEGER*2 NUMREM_BFRS                     ! Number of remote buffers
        LOGICAL*1 GPFLAG                          ! Get/put flag
        LOGICAL*1 LASTBFR                         ! Last buffer flag
        LOGICAL*1 EOFFLAG                         ! End of file flag
        LOGICAL*1 ERRFLAG                         ! Error flag
        LOGICAL*1 REMFLAG                         ! Remote error flag
        COMMON /MDATA/  IDEVMSG,ODEVMSG,REM_BFRADS,FILEATTR,CSTATUS,
        1               LASTBFRSIZ,DDIDIS,MRMS_CNT,MRMS_IDX,QPKT_CNT,
        2               QPKT_IDX,REM_CNT,REM_IDX,NUMREM_BFRS,GPFLAG,
        3               LASTBFR,EOFFLAG,ERRFLAG,REMFLAG

        CHARACTER*64 OFNA
        BYTE ODEVMSGB(128)

        EQUIVALENCE (ODEVMSGB,ODEVMSG)
        EQUIVALENCE (OFNA,ODEVMSGB(33))

        INTEGER*4 SYS$CLREF
        INTEGER*4 SYS$WAITFR

C
C       Initialize flags
C
        LASTBFR = .FALSE.                       ! Last buffer flag
        EOFFLAG = .FALSE.                       ! End of file flag
        ERRFLAG = .FALSE.                       ! Error flag
        REMFLAG = .FALSE.                       ! Remote error flag

C
C       Queue a NOP packet to the DR32.  The purpose of this
C       is to examine the DDI disable bit in the DSL to determine
C       if the DR32 at the other end is ready to go.
C
        CALL QUEUE_NOP(STATUS)
        IF (.NOT. STATUS) RETURN

C
C       Open local file and copy file attributes into device
C       message array.
C
        CALL OPEN_FILE(LOC_FNAME,LOC_FNSIZE,ODEVMSG(3),STATUS)
        IF (.NOT. STATUS) THEN
            CALL ERROR(STATUS,.FALSE.)
            RETURN
        END IF

C
C       Finish building device message and send it.
C
        ODEVMSG(1) = 1                          ! Packet type
        ODEVMSG(2) = BUFSIZ                     ! Buffer size
        ODEVMSGB(32) = REM_FNSIZE               ! Remote filename size
        OFNA = REM_FNAME                        ! Remote filename
        STATUS = SYS$CLREF(%VAL(1))             ! Clear event flag
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)
        CALL XF$PKTBLD(XFDATA,                  ! Context array
        1               XF$K_PKT_WRTCM,         ! Function = write ctrl msg
        2               ,,                      ! No index or size
        3               ODEVMSG,                ! Device message
        4               96,                     ! Device message size
        5               ,                               ! No log area
        6               64+256,                         ! Ins. @ head, int. if Q empty
        7               ,,                              ! No action routine or parm
        8               STATUS)                         ! Status
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)
        STATUS = SYS$WAITFR(%VAL(1))                    ! Wait for event flag
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)
        IF (.NOT. CSTATUS) THEN                         ! Error from remote system
            CALL ERROR(CSTATUS,REMFLAG)
            GO TO 8000
        END IF

C
C       Set up buffer counters and indexes
C
        MRMS_CNT = NUM_MBFRS - 1                        ! # of avl RMS buffers
        MRMS_IDX = 2                                    ! Next RMS buffer
        QPKT_CNT = 0                                    ! # of buffers to be queued
        QPKT_IDX = 1                                    ! Next buffer to be queued
        REM_CNT = NUMREM_BFRS                           ! # of remote buffers
        REM_IDX = 1                                     ! Next remote buffer to use

C
C       Start the transfer going by starting an RMS read and then
C       wait until it completes.
C
        STATUS = SYS$CLREF(%VAL(3))                     ! Clear event flag
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)
        CALL START_RMS(MBFRS(1,1),BUFSIZ,GPFLAG)! Start RMS
        STATUS = SYS$WAITFR(%VAL(3))                    ! Wait for event flag
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)
        IF (.NOT. CSTATUS) CALL ERROR(CSTATUS,REMFLAG)

8000    CALL CLOSE_FILE(STATUS)
        IF (.NOT. STATUS) CALL ERROR(STATUS,.FALSE.)

        RETURN
        END
        SUBROUTINE DO_GET

C
C       This routine is the top level routine for copying a file
C       from the remote cpu to the local cpu.
C

        INCLUDE 'SYS$LIBRARY:XFDEF.FOR/NOLIST'  ! DR32 definitions
        INCLUDE 'DRCOPY.PRM/NOLIST'             ! Parameters

C
C       Local Variables
C
        INTEGER*4 STATUS                        ! Local status

C
C       Common variables and areas
C
        CHARACTER*80  INPLINE                   ! Input line
        CHARACTER*64  LOC_FNAME                 ! Local file name
        CHARACTER*64  REM_FNAME                 ! Remote file name

        COMMON  /CHARS/  INPLINE,LOC_FNAME,REM_FNAME

        INTEGER*2  LOC_FNSIZE                   ! Local file name size
        INTEGER*2  REM_FNSIZE                   ! Remote file name size
        INTEGER*2  SPOS                         ! Starting token pos.
        INTEGER*2  EPOS                         ! Ending token pos.

        COMMON  /SIZES/  LOC_FNSIZE,REM_FNSIZE,SPOS,EPOS

        INTEGER*4 XFDATA(30)                    ! Context array
        BYTE MBFRS(BUFSIZ,NUM_MBFRS)            ! Master buffers
        BYTE SBFRS(BUFSIZ,NUM_SBFRS)            ! Slave buffers

        COMMON /MS_SHARE/  XFDATA,MBFRS,SBFRS

        INTEGER*4 IDEVMSG(32)                   ! Incoming device messages
        INTEGER*4 ODEVMSG(32)                   ! Outgoing device messages
        INTEGER*4 REM_BFRADS(25)                ! Remote buffer addresses
        INTEGER*4 FILEATTR(6)                   ! File attributes
        INTEGER*4 CSTATUS                       ! Common status
        INTEGER*4 LASTBFRSIZ                    ! Last buffer size
        INTEGER*4 DDIDIS                        ! DDI disable
        INTEGER*2 MRMS_CNT                      ! Master RMS count
        INTEGER*2 MRMS_IDX                      ! Master RMS index
        INTEGER*2 QPKT_CNT                      ! Queue packet count
        INTEGER*2 QPKT_IDX                      ! Queue packet index
        INTEGER*2 REM_CNT                       ! Remote buffer count
        INTEGER*2 REM_IDX                       ! Remote buffer index
        INTEGER*2 NUMREM_BFRS                   ! Number of remote buffers
        LOGICAL*1 GPFLAG                        ! Get/put flag
        LOGICAL*1 LASTBFR                       ! Last buffer flag
        LOGICAL*1 EOFFLAG                       ! End of file flag
        LOGICAL*1 ERRFLAG                       ! Error flag
        LOGICAL*1 REMFLAG                       ! Remote error flag
```

```
      COMMON /MDATA/  IDEVMSG,ODEVMSG,REM_BFRADS,FILEATTR,CSTATUS,
     1                LASTBFRSIZ,DDIDIS,MRMS_CNT,MRMS_IDX,QPKT_CNT,
     2                QPKT_IDX,REM_CNT,REM_IDX,NUMREM_BFRS,GPFLAG,
     3                LASTBFR,EOFFLAG,ERRFLAG,REMFLAG

      CHARACTER*64 OFNA
      BYTE ODEVMSGB(128)

      EQUIVALENCE (ODEVMSGB,ODEVMSG)
      EQUIVALENCE (OFNA,ODEVMSGB(33))

      INTEGER*4 SYS$CLREF
      INTEGER*4 SYS$WAITFR
      INTEGER*4 SYS$WFLAND

C
C     Initialize flags
C
      LASTBFR = .FALSE.                         ! Last buffer flag
      EOFFLAG = .FALSE.                         ! End of file flag
      ERRFLAG = .FALSE.                         ! Error flag
      REMFLAG = .FALSE.                         ! Remote error flag

C
C     Queue a NOP packet to the DR32.  The purpose of this is to
C     examine the DDI disable bit in the DSL to determine if
C     the DR32 at the other end is ready to go.
C
      CALL QUEUE_NOP(STATUS)
      IF (.NOT. STATUS) RETURN

C
C     Build a message to send to the remote system indicating we
C     want to GET a file.  Send the remote filename.
C
      ODEVMSG(1) = 3                            ! Message type
      ODEVMSG(2) = BUFSIZ                       ! Buffer size
      ODEVMSGB(32) = REM_FNSIZE                 ! Remote filename size
      OFNA = REM_FNAME                          ! Remote filename

C
C     Send the message and wait for 2 packets in response:
C     1)  the file attributes and  2)  the remote buffer addresses.
C
      STATUS = SYS$CLREF(%VAL(1))
      IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)
      STATUS = SYS$CLREF(%VAL(2))
      IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)

      CALL XF$PKTBLD(XFDATA,                    ! Context array
     1              XF$K_PKT_WRTCM,,,           ! Function, no index or size
     2              ODEVMSG,92,,                ! Device msg, size, no log area
     3              64+256,,,                   ! Ins. @ head, int. if Q empty
```

```
        4               STATUS)                 ! Status
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)

        STATUS = SYS$WFLAND(%VAL(1),%VAL(6))    ! Wait for both responses
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)
        IF (.NOT. CSTATUS) THEN
            CALL ERROR(CSTATUS,REMFLAG)
            RETURN
        END IF
C
C       Create the local file using the file attributes sent by
C       the remote system.
C
        CALL CREATE_FILE(LOC_FNAME,LOC_FNSIZE,FILEATTR,STATUS)
        IF (.NOT. STATUS) THEN
            CALL RMS_ERROR(STATUS)
            CALL ERROR(STATUS,.FALSE.)
            RETURN
        END IF
C
C       Set up buffer counters and indexes
C
        MRMS_CNT = -1                           ! RMS is not going
        MRMS_IDX = 1                            ! Next RMS buffer
        QPKT_CNT = NUM_MBFRS                    ! # of buffers to be queued
        QPKT_IDX = 1                            ! Next buffer to be queued
        REM_CNT = 0                             ! # of remote buffers
        REM_IDX = 1                             ! Next remote buffer
C
C       Start the transfer going and wait until it completes.
C
        STATUS = SYS$CLREF(%VAL(3))
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)

        ODEVMSG(1) = 11                         ! Start remote sys. going
        CALL XF$PKTBLD(XFDATA,                  ! Context array
        1               XF$K_PKT_WRTCM,,,       ! Function, no index or size
        2               ODEVMSG,4,,             ! Device msg, size, no log area
        3               64+256,,,               ! Ins. @ head, int. if Q empty
        4               STATUS)
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)

        STATUS = SYS$WAITFR(%VAL(3))
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)
        IF (.NOT. CSTATUS) CALL ERROR(CSTATUS,REMFLAG)

        CALL CLOSE_FILE(STATUS)
        IF (.NOT. STATUS) CALL ERROR(STATUS,.FALSE.)
        RETURN
        END
```

```
      SUBROUTINE QUEUE_NOP(DSTATUS)

C
C     This routine queues a NOP packet to the DR32 to determine
C     if the remote cpu is ready to start a transfer.  This is
C     accomplished by testing the DDI disable bit in the DSL
C     in the packet.  The actual testing of the bit is done at
C     AST level by an action routine and the result is returned
C     in the variable DDIDIS.
C
C     DSTATUS is returned as follows:
C
C             0       Remote CPU not ready (this routine prints
C                     error message)
C             1       Remote CPU ready (success)
C

      INCLUDE 'SYS$LIBRARY:XFDEF.FOR/NOLIST'  ! DR32 definitions
      INCLUDE 'DRCOPY.PRM/NOLIST'             ! Parameters

C
C     Local Variables
C
      INTEGER*4 STATUS                        ! Local status

C
C     Common variables and areas
C
      INTEGER*4 XFDATA(30)                    ! Context array
      BYTE MBFRS(BUFSIZ,NUM_MBFRS)            ! Master buffers
      BYTE SBFRS(BUFSIZ,NUM_SBFRS)            ! Slave buffers

      COMMON /MS_SHARE/  XFDATA,MBFRS,SBFRS

      INTEGER*4 IDEVMSG(32)                   ! Incoming device messages
      INTEGER*4 ODEVMSG(32)                   ! Outgoing device messages
      INTEGER*4 REM_BFRADS(25)                ! Remote buffer addresses
      INTEGER*4 FILEATTR(6)                   ! File attributes
      INTEGER*4 CSTATUS                       ! Common status
      INTEGER*4 LASTBFRSIZ                    ! Last buffer size
      INTEGER*4 DDIDIS                        ! DDI disable
      INTEGER*2 MRMS_CNT                      ! Master RMS count
      INTEGER*2 MRMS_IDX                      ! Master RMS index
      INTEGER*2 QPKT_CNT                      ! Queue packet count
      INTEGER*2 QPKT_IDX                      ! Queue packet index
      INTEGER*2 REM_CNT                       ! Remote buffer count
      INTEGER*2 REM_IDX                       ! Remote buffer index
      INTEGER*2 NUMREM_BFRS                   ! Number of remote buffers
      LOGICAL*1 GPFLAG                        ! Get/put flag
      LOGICAL*1 LASTBFR                       ! Last buffer flag
      LOGICAL*1 EOFFLAG                       ! End of file flag
      LOGICAL*1 ERRFLAG                       ! Error flag
      LOGICAL*1 REMFLAG                       ! Remote error flag

      COMMON /MDATA/  IDEVMSG,ODEVMSG,REM_BFRADS,FILEATTR,CSTATUS,
     1                LASTBFRSIZ,DDIDIS,MRMS_CNT,MRMS_IDX,QPKT_CNT,
```

```
DRM

C
C
C
100

C
C
C
200

C
C
C
300

C
C
C
400

C
C
C
500

C
C
C
600

C
C
C
700

C
C
C
800

C
C
C
900

C
C
```

```
        2                QPKT_IDX,REM_CNT,REM_IDX,NUMREM_BFRS,GPFLAG,
        3                LASTBFR,EOFFLAG,ERRFLAG,REMFLAG

        INTEGER*4 DSTATUS

        INTEGER*4 SYS$CLREF
        INTEGER*4 SYS$WAITFR

        EXTERNAL ACT_NOPPKT


        STATUS = SYS$CLREF(%VAL(1))              ! Clear event flag
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)
        CALL XF$PKTBLD(XFDATA,                   ! Context array
        1            XF$K_PKT_NOP,               ! Function = NOP
        2            ,,,,,                       ! No index, size, dev. msg, log area
        3            64+256,                     ! Ins. @ head, int. if Q empty
        4            ACT_NOPPKT,,                ! Action routine, parm.
        5            STATUS)                     ! Status
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)
        STATUS = SYS$WAITFR(%VAL(1))             ! Wait for completion
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)

C
C       Test DDIDIS which was set at AST level by the action
C       routine ACT_NOPPKT.  If non-zero, then print an error message.
C
        IF (DDIDIS .NE. 0) THEN
            WRITE(6,100)
100         FORMAT(1X,'%DRCOPY-E-REMNRDY, remote DR32 not ready')
            DSTATUS = 0
        ELSE
            DSTATUS = 1                          ! Success
        END IF

        RETURN
        END
        SUBROUTINE MRMS_AST(RAB)

C
C       This subroutine is the Master RMS completion routine.  It
C       is called at AST level to start the next RMS operation
C       and to queue a packet to the DR32 to read or write the next
C       buffer.
C

        INCLUDE 'DRCOPY.PRM/NOLIST'               ! Parameters
        PARAMETER RAB$K_BLN = '44'X
        PARAMETER RAB$W_RSZ = '22'X
        PARAMETER RAB$L_STS = '8'X
        PARAMETER RMS$_EOF = '1827A'X

C
C       Local Variables
C
        INTEGER*4 STATUS                          ! Local status
        INTEGER*4 RAB(RAB$K_BLN/4+1)
        INTEGER*4 SIZE
        INTEGER*4 BFRSIZE

C
C       Common variables and areas
C
        INTEGER*4 XFDATA(30)                      ! Context array
        BYTE MBFRS(BUFSIZ,NUM_MBFRS)              ! Master buffers
        BYTE SBFRS(BUFSIZ,NUM_SBFRS)              ! Slave buffers

        COMMON /MS_SHARE/  XFDATA,MBFRS,SBFRS

        INTEGER*4 IDEVMSG(32)                     ! Incoming device messages
        INTEGER*4 ODEVMSG(32)                     ! Outgoing device messages
        INTEGER*4 REM_BFRADS(25)                  ! Remote buffer addresses
        INTEGER*4 FILEATTR(6)                     ! File attributes
        INTEGER*4 CSTATUS                         ! Common status
        INTEGER*4 LASTBFRSIZ                      ! Last buffer size
        INTEGER*4 DDIDIS                          ! DDI disable
        INTEGER*2 MRMS_CNT                        ! Master RMS count
        INTEGER*2 MRMS_IDX                        ! Master RMS index
        INTEGER*2 QPKT_CNT                        ! Queue packet count
        INTEGER*2 QPKT_IDX                        ! Queue packet index
        INTEGER*2 REM_CNT                         ! Remote buffer count
        INTEGER*2 REM_IDX                         ! Remote buffer index
        INTEGER*2 NUMREM_BFRS                     ! Number of remote buffers
        LOGICAL*1 GPFLAG                          ! Get/put flag
        LOGICAL*1 LASTBFR                         ! Last buffer flag
        LOGICAL*1 EOFFLAG                         ! End of file flag
        LOGICAL*1 ERRFLAG                         ! Error flag
        LOGICAL*1 REMFLAG                         ! Remote error flag

        COMMON /MDATA/  IDEVMSG,ODEVMSG,REM_BFRADS,FILEATTR,CSTATUS,
        1               LASTBFRSIZ,DDIDIS,MRMS_CNT,MRMS_IDX,QPKT_CNT,
        2               QPKT_IDX,REM_CNT,REM_IDX,NUMREM_BFRS,GPFLAG,
        3               LASTBFR,EOFFLAG,ERRFLAG,REMFLAG
        INTEGER*4 SYS$SETEF

        EXTERNAL SS$_NORMAL



C
C       If ERRFLAG is set, then set event flag 3 to wake up main level
C       and return.
C
        IF (ERRFLAG) THEN
            STATUS = SYS$SETEF(%VAL(3))
            IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)
            RETURN
        END IF

C
C       Check for success or failure of last operation.  If success
C       see if an entire buffer was transferred.  If not, set the
C       end of file flag.  If error is RMS$_EOF and doing a PUT, then
C       also set end of file flag.
C
        STATUS = RAB(RAB$L_STS/4+1)              ! Get status from RAB
        IF (STATUS) GO TO 400                    ! Success
        IF (STATUS .EQ. RMS$_EOF .AND. GPFLAG .EQ. 3) GO TO 450
        CALL RMS_ERROR(STATUS)
        RETURN

400     IF (GPFLAG .EQ. 1) GO TO 500             ! Only check EOF for PUT
450     BFRSIZE = RAB(RAB$W_RSZ/4+1)/65536       ! Get size of buffer
        IF (BFRSIZE .NE. BUFSIZ) THEN
            EOFFLAG = .TRUE.                     ! Set end of file flag
            LASTBFRSIZ = BFRSIZE
            GO TO 700
        END IF

C
C       Decrement the count of the number of buffers available
C       for an RMS operation.  If the count goes negative, then
C       we ran out of buffers temporarily and can't start an RMS
C       operation (the next RMS operation will get started in the
C       ACT_RWPKT routine which makes buffers available for RMS
C       operations.)  Otherwise, start the next RMS operation now.
C
500     MRMS_CNT = MRMS_CNT - 1                  ! Decr. RMS buffer count
        IF (MRMS_CNT .GE. 0) THEN
            SIZE = BUFSIZ                        ! Assume not last buffer
            IF (LASTBFR .AND. MRMS_CNT .EQ. 0)
        1               SIZE = LASTBFRSIZ         ! This is the last buffer
            CALL START_RMS(MBFRS(1,MRMS_IDX),SIZE,GPFLAG) ! Start RMS operation
            MRMS_IDX = MRMS_IDX + 1              ! Advance next buffer index
            IF (MRMS_IDX .GT. NUM_MBFRS) MRMS_IDX = 1  ! modulo NUM_MBFRS
        END IF

C
```

```
C       If MRMS_CNT is less than 0 and LASTBFR is .TRUE. then we
C       are finished doing a GET.  Wake up main level.
C
600     IF (MRMS_CNT .LT. 0 .AND. LASTBFR) THEN
            CSTATUS = %LOC(SS$_NORMAL)          ! Indicate success
            STATUS = SYS$SETEF(%VAL(3))         ! Wake up main level
            IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)
        END IF

C
C       Increment the number of buffers available to be queued to the
C       DR32.  If we have a matching remote buffer then queue an
C       operation to the DR32.
C
700     QPKT_CNT = QPKT_CNT + 1                 ! Incr. QPKT buffer count
        IF (REM_CNT .GT. 0) CALL QUEUE_PKT

        RETURN
        END
        SUBROUTINE QUEUE_PKT

C
C       This routine queues read or write data packets to the DR32.
C       It is called from either MRMS_AST, MFQ_PNXTBFR, or MFQ_PLSTBFR
C       only when both a local and a remote buffer are available.
C

        INCLUDE 'SYS$LIBRARY:XFDEF.FOR/NOLIST'  ! DR32 definitions
        INCLUDE 'DRCOPY.PRM/NOLIST'             ! Parameters

C
C       Local Variables
C
        INTEGER*4 STATUS
        INTEGER*4 FUNC
        INTEGER*4 SIZE
        INTEGER*2 BFRCNT
C
C       Common variables and areas
C
        INTEGER*4 XFDATA(30)                    ! Context array
        BYTE MBFRS(BUFSIZ,NUM_MBFRS)            ! Master buffers
        BYTE SBFRS(BUFSIZ,NUM_SBFRS)            ! Slave buffers

        COMMON /MS_SHARE/  XFDATA,MBFRS,SBFRS

        INTEGER*4 IDEVMSG(32)                   ! Incoming device messages
        INTEGER*4 ODEVMSG(32)                   ! Outgoing device messages
        INTEGER*4 REM_BFRADS(25)                ! Remote buffer addresses
        INTEGER*4 FILEATTR(6)                   ! File attributes
        INTEGER*4 CSTATUS                       ! Common status
        INTEGER*4 LASTBFRSIZ                    ! Last buffer size
        INTEGER*4 DDIDIS                        ! DDI disable
        INTEGER*2 MRMS_CNT                      ! Master RMS count
        INTEGER*2 MRMS_IDX                      ! Master RMS index
        INTEGER*2 QPKT_CNT                      ! Queue packet count
        INTEGER*2 QPKT_IDX                      ! Queue packet index
        INTEGER*2 REM_CNT                       ! Remote buffer count
        INTEGER*2 REM_IDX                       ! Remote buffer index
        INTEGER*2 NUMREM_BFRS                   ! Number of remote buffers
        LOGICAL*1 GPFLAG                        ! Get/put flag
        LOGICAL*1 LASTBFR                       ! Last buffer flag
        LOGICAL*1 EOFFLAG                       ! End of file flag
        LOGICAL*1 ERRFLAG                       ! Error flag
        LOGICAL*1 REMFLAG                       ! Remote error flag

        COMMON /MDATA/  IDEVMSG,ODEVMSG,REM_BFRADS,FILEATTR,CSTATUS,
        1               LASTBFRSIZ,DDIDIS,MRMS_CNT,MRMS_IDX,QPKT_CNT,
        2               QPKT_IDX,REM_CNT,REM_IDX,NUMREM_BFRS,GPFLAG,
        3               LASTBFR,EOFFLAG,ERRFLAG,REMFLAG

        EXTERNAL ACT_RWPKT

C
C       Decrement buffer counters
```

```
C
        REM_CNT = REM_CNT - 1                   ! Remote buffer count
        QPKT_CNT = QPKT_CNT - 1                 ! QPKT buffer count

C
C       Queue packet to DR32.
C
        ODEVMSG(1) = REM_BFRADS(REM_IDX)        ! Device msg is remote bfr. addr.
        IF (GPFLAG .EQ. T) THEN
            FUNC = XF$K_PKT_RD                  ! Doing a GET
            BFRCNT = REM_CNT
        ELSE
            FUNC = XF$K_PKT_WRT                 ! Doing a PUT
            BFRCNT = QPKT_CNT
        END IF

        IF (BFRCNT.EQ.0 .AND. EOFFLAG) THEN
            SIZE = LASTBFRSIZ                   ! This is the last buffer
        ELSE
            SIZE =BUFSIZ                        ! Not the last buffer
        END IF

        CALL XF$PKTBLD(XFDATA,                  ! Context array
        1              FUNC,                    ! Function
        2              QPKT_IDX,                ! Buffer index
        3              SIZE,                    ! Size of transfer
        4              ODEVMSG,                 ! Device message
        5              4,                       ! Size of device message
        6              ,                        ! No log area
        7              24+64,                   ! Send all, int. on Q empty
        8              ACT_RWPKT,,              ! Action routine, no param.
        9              STATUS)                  ! Status

C
C       Adjust buffer indexes
C
        REM_IDX = REM_IDX + 1                   ! Advance remote buffer index
        IF (REM_IDX .GT. NUMREM_BFRS) REM_IDX=1 ! modulo # of remote buffers
        QPKT_IDX = QPKT_IDX +1                  ! Advance QPKT buffer index
        IF (QPKT_IDX .GT. NUM_MBFRS) QPKT_IDX=1 ! modulo # of local buffers

C
C       Check for success from XF$PKTBLD
C
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)

        RETURN
        END
        SUBROUTINE PKT_AST

C
C       This routine is called whenever a DR32 command packet is
C       placed on an empty termination queue.  This routine calls
C       XF$GETPKT which removes the packet at the head of the
C       termination queue and calls the action routine, if there is
C       one.  This routine must process all packets on the
C       termination queue until the queue is empty.
C

        INCLUDE 'DRCOPY.PRM/NOLIST'                 ! Parameters
        PARAMETER SHR$_QEMPTY = '1280'X

C
C       Local Variables
C
        INTEGER*4 STATUS

C
C       Common variables and areas
C
        INTEGER*4 XFDATA(30)                        ! Context array
        BYTE MBFRS(BUFSIZ,NUM_MBFRS)                ! Master buffers
        BYTE SBFRS(BUFSIZ,NUM_SBFRS)                ! Slave buffers

        COMMON /MS_SHARE/  XFDATA,MBFRS,SBFRS

100     CALL XF$GETPKT(XFDATA,1,,,,,STATUS)         ! Calls action routine
        IF (STATUS) GO TO 100                       ! Repeat until q empty
        IF (STATUS .EQ. SHR$_QEMPTY) GO TO 9000

C
C       Have a fatal error - print error and IOSB
C
        CALL DR32_ERROR                             ! Doesn't return

9000    RETURN
        END
        SUBROUTINE ACT_NOPPKT
C
C       This routine is the action routine for NOP packets.  It
C       tests the DDI disable bit in the DSL and sets DDIDIS.
C
        INCLUDE 'SYS$LIBRARY:XFDEF.FOR/NOLIST'  ! DR32 definitions
        INCLUDE 'DRCOPY.PRM/NOLIST'             ! Parameters

C
C       Local Variables
C
        INTEGER*4 STATUS

C
C       Common variables and areas
C
        INTEGER*4 XFDATA(30)                    ! Context array
        BYTE MBFRS(BUFSIZ,NUM_MBFRS)            ! Master buffers
        BYTE SBFRS(BUFSIZ,NUM_SBFRS)            ! Slave buffers

        COMMON /MS_SHARE/  XFDATA,MBFRS,SBFRS

        INTEGER*4 IDEVMSG(32)                   ! Incoming device messages
        INTEGER*4 ODEVMSG(32)                   ! Outgoing device messages
        INTEGER*4 REM_BFRADS(25)                ! Remote buffer addresses
        INTEGER*4 FILEATTR(6)                   ! File attributes
        INTEGER*4 CSTATUS                       ! Common status
        INTEGER*4 LASTBFRSIZ                    ! Last buffer size
        INTEGER*4 DDIDIS                        ! DDI disable
        INTEGER*2 MRMS_CNT                      ! Master RMS count
        INTEGER*2 MRMS_IDX                      ! Master RMS index
        INTEGER*2 QPKT_CNT                      ! Queue packet count
        INTEGER*2 QPKT_IDX                      ! Queue packet index
        INTEGER*2 REM_CNT                       ! Remote buffer count
        INTEGER*2 REM_IDX                       ! Remote buffer index
        INTEGER*2 NUMREM_BFRS                   ! Number of remote buffers
        LOGICAL*1 GPFLAG                        ! Get/put flag
        LOGICAL*1 LASTBFR                       ! Last buffer flag
        LOGICAL*1 EOFFLAG                       ! End of file flag
        LOGICAL*1 ERRFLAG                       ! Error flag
        LOGICAL*1 REMFLAG                       ! Remote error flag

        COMMON /MDATA/  IDEVMSG,ODEVMSG,REM_BFRADS,FILEATTR,CSTATUS,
        1               LASTBFRSIZ,DDIDIS,MRMS_CNT,MRMS_IDX,QPKT_CNT,
        2               QPKT_IDX,REM_CNT,REM_IDX,NUMREM_BFRS,GPFLAG,
        3               LASTBFR,EOFFLAG,ERRFLAG,REMFLAG

        INTEGER*4 SYS$SETEF

        INTEGER*4 DSL                           ! DR32 status longword

        EQUIVALENCE (DSL,XFDATA(8))
```

```
      DDIDIS = DSL .AND. XF$M_PKT_DDIDIS
      STATUS = SYS$SETEF(%VAL(1))
      IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)
      RETURN
      END
      SUBROUTINE ACT_RWPKT
C
C     This subroutine is the action routine for a Read or Write
C     data packet which has just completed.
C
      INCLUDE 'SYS$LIBRARY:XFDEF.FOR/NOLIST'   ! DR32 definitions
      INCLUDE 'DRCOPY.PRM/NOLIST'              ! Parameters
C
C     Local Variables
C
      INTEGER*4 STATUS
      INTEGER*4 SIZE
C
C     Common variables and areas
C
      INTEGER*4 XFDATA(30)                     ! Context array
      BYTE MBFRS(BUFSIZ,NUM_MBFRS)             ! Master buffers
      BYTE SBFRS(BUFSIZ,NUM_SBFRS)             ! Slave buffers

      COMMON /MS_SHARE/  XFDATA,MBFRS,SBFRS

      INTEGER*4 IDEVMSG(32)                    ! Incoming device messages
      INTEGER*4 ODEVMSG(32)                    ! Outgoing device messages
      INTEGER*4 REM_BFRADS(25)                 ! Remote buffer addresses
      INTEGER*4 FILEATTR(6)                    ! File attributes
      INTEGER*4 CSTATUS                        ! Common status
      INTEGER*4 LASTBFRSIZ                     ! Last buffer size
      INTEGER*4 DDIDIS                         ! DDI disable
      INTEGER*2 MRMS_CNT                       ! Master RMS count
      INTEGER*2 MRMS_IDX                       ! Master RMS index
      INTEGER*2 QPKT_CNT                       ! Queue packet count
      INTEGER*2 QPKT_IDX                       ! Queue packet index
      INTEGER*2 REM_CNT                        ! Remote buffer count
      INTEGER*2 REM_IDX                        ! Remote buffer index
      INTEGER*2 NUMREM_BFRS                    ! Number of remote buffers
      LOGICAL*1 GPFLAG                         ! Get/put flag
      LOGICAL*1 LASTBFR                        ! Last buffer flag
      LOGICAL*1 EOFFLAG                        ! End of file flag
      LOGICAL*1 ERRFLAG                        ! Error flag
      LOGICAL*1 REMFLAG                        ! Remote error flag

      COMMON /MDATA/  IDEVMSG,ODEVMSG,REM_BFRADS,FILEATTR,CSTATUS,
     1                LASTBFRSIZ,DDIDIS,MRMS_CNT,MRMS_IDX,QPKT_CNT,
     2                QPKT_IDX,REM_CNT,REM_IDX,NUMREM_BFRS,GPFLAG,
     3                LASTBFR,EOFFLAG,ERRFLAG,REMFLAG

      INTEGER*4 DSL                            ! DR32 status longword

      EQUIVALENCE (DSL,XFDATA(8))
```

```
        IF (.NOT. DSL) CALL DR32_ERROR              ! Error in DSL

C
C       Send a message to the remote system telling it that it's
C       next buffer has been processed (filled or emptied).  If
C       the size of the transfer is not equal to the buffer size,
C       then it must have been the last transfer.
C
        IF (XFDATA(4) .NE. BUFSIZ) THEN
            MODE = 64                               ! Last transfer - insert at tail of Q
            ODEVMSG(1) = 7                          ! Last buffer message
            ODEVMSG(2) = LASTBFRSIZ                 ! Send size
            LASTBFR = .TRUE.                        ! Set flag
        ELSE                                        ! Not last transfer
            MODE = 64 + 256                         ! Insert at head of Q
            ODEVMSG(1) = 5                          ! Next buffer message
        END IF

        IF (LASTBFR .AND. GPFLAG .EQ. 1) GO TO 5000  ! Don't send if last buffer and GET

        CALL XF$PKTBLD(XFDATA,                      ! Context array
     1                 XF$K_PKT_WRTCM,,,            ! Function, no index or size
     2                 ODEVMSG,8,,                  ! Device msg, size, no log area
     3                 MODE,,,                      ! Mode, no action routine
     4                 STATUS)                      ! Status
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)

C
C       Increment the count of the number of buffers available for
C       an RMS operation.  If the count equals zero, then we
C       previously ran out of RMS buffers and therefore there is
C       no RMS operation in progress.  In this case, we start the
C       next RMS operation.  If the count is greater than zero, then
C       there is already an RMS operation in progress.
C
5000    MRMS_CNT = MRMS_CNT + 1                     ! Incr. RMS buffer count
        IF (MRMS_CNT .EQ. 0) THEN
            SIZE = BUFSIZ                           ! Assume not last buffer
            IF (LASTBFR) SIZE = LASTBFRSIZ          ! This is the last buffer (GET only)
            CALL START_RMS(MBFRS(1,MRMS_IDX),SIZE,GPFLAG)  ! Start RMS
            MRMS_IDX = MRMS_IDX + 1                 ! Advance RMS buffer index
            IF (MRMS_IDX .GT. NUM_MBFRS) MRMS_IDX = 1  ! modulo NUM_MBFRS
        END IF

        RETURN
        END
        SUBROUTINE ACT_FREQUE
C
C       This routine is the action routine for packets that were
C       on the free queue.  First it puts another packet on the free queue,
C       and then calls the appropriate routine based on the type code
C       in the device message.
C
        INCLUDE 'DRCOPY.PRM/NOLIST'             ! Parameters
        PARAMETER SHR$_VALERR = '11E8'X

C
C       Local Variables
C
        INTEGER*4 STATUS

C
C       Common variables and areas
C
        INTEGER*4 IDEVMSG(32)                   ! Incoming device messages
        INTEGER*4 ODEVMSG(32)                   ! Outgoing device messages
        INTEGER*4 REM_BFRADS(25)                ! Remote buffer addresses
        INTEGER*4 FILEATTR(6)                   ! File attributes
        INTEGER*4 CSTATUS                       ! Common status
        INTEGER*4 LASTBFRSIZ                    ! Last buffer size
        INTEGER*4 DDIDIS                        ! DDI disable
        INTEGER*2 MRMS_CNT                      ! Master RMS count
        INTEGER*2 MRMS_IDX                      ! Master RMS index
        INTEGER*2 QPKT_CNT                      ! Queue packet count
        INTEGER*2 QPKT_IDX                      ! Queue packet index
        INTEGER*2 REM_CNT                       ! Remote buffer count
        INTEGER*2 REM_IDX                       ! Remote buffer index
        INTEGER*2 NUMREM_BFRS                   ! Number of remote buffers
        LOGICAL*1 GPFLAG                        ! Get/put flag
        LOGICAL*1 LASTBFR                       ! Last buffer flag
        LOGICAL*1 EOFFLAG                       ! End of file flag
        LOGICAL*1 ERRFLAG                       ! Error flag
        LOGICAL*1 REMFLAG                       ! Remote error flag

        COMMON /MDATA/  IDEVMSG,ODEVMSG,REM_BFRADS,FILEATTR,CSTATUS,
        1               LASTBFRSIZ,DDIDIS,MRMS_CNT,MRMS_IDX,QPKT_CNT,
        2               QPKT_IDX,REM_CNT,REM_IDX,NUMREM_BFRS,GPFLAG,
        3               LASTBFR,EOFFLAG,ERRFLAG,REMFLAG


        CALL FREESET                            ! Put another packet on FREQ

        GO TO (100,200,300,400,500,600,700,800,900,1000,1100),IDEVMSG(1)

C
C       Invalid packet
C
        CALL FATAL_ERROR(SHR$_VALERR)
```

```
C
C       Type code = 1    Start a PUT     M -> S
C
100     CALL SFQ_STARTPUT(IDEVMSG)
        GO TO 9000

C
C       Type code = 2    Slave buffer addresses     S -> M
C
200     CALL MFQ_BFRADS
        GO TO 9000

C
C       Type code = 3    Start a GET     M -> S
C
300     CALL SFQ_STARTGET(IDEVMSG)
        GO TO 9000

C
C       Type code = 4    File attributes     S -> M
C
400     CALL MFQ_FILEATTR
        GO TO 9000

C
C       Type code = 5    Processed next buffer     M -> S
C
500     CALL SFQ_PNXTBFR(IDEVMSG)
        GO TO 9000

C
C       Type code = 6    Processed next buffer     S -> M
C
600     CALL MFQ_PNXTBFR
        GO TO 9000

C
C       Type code = 7    Processed last buffer     M -> S
C
700     CALL SFQ_PLSTBFR(IDEVMSG)
        GO TO 9000

C
C       Type code = 8    Processed last buffer     S -> M
C
800     CALL MFQ_PLSTBFR
        GO TO 9000

C
C       Type code = 9    Error     M -> S
C
900     CALL SLV_SHUTDOWN
        GO TO 9000

C
C       Type code = 10    Error     S -> M
```

```
C
1000    CALL MFQ_ERROR
        GO TO 9000
C
C       Type code = 11  Start sending data (Get only)   M -> S
C
1100    CALL SFQ_GOGET

9000    RETURN
        END
```

```
DR

C
C
C
C

C
C
C

10

20
```

```
        SUBROUTINE FREESET

C
C       This routine puts an empty packet on the FREQ.  It is called from
C       ACT_FREQUE and the only reason it's a subroutine is that ACT_FREQUE
C       can't be an external in ACT_FREQUE.
C

        INCLUDE 'DRCOPY.PRM/NOLIST'

C
C       Local variables
C
        INTEGER*4 STATUS

C
C       Common variables and areas
C
        INTEGER*4 XFDATA(30)                    ! Context array
        BYTE MBFRS(BUFSIZ,NUM_MBFRS)            ! Master buffers
        BYTE SBFRS(BUFSIZ,NUM_SBFRS)            ! Slave buffers

        COMMON /MS_SHARE/  XFDATA,MBFRS,SBFRS

        EXTERNAL ACT_FREQUE

        CALL XF$FREESET(XFDATA,
        1               1,                      ! Number of packets
        2               1,                      ! AST if TERMQ empty
        3               ACT_FREQUE,,            ! Action routine and parameter
        4               STATUS)                 ! Status
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)

        RETURN
        END
```

DR

```
C
C
C
```

```
        SUBROUTINE MFQ_BFRADS

C
C       This routine is called to process the list of buffer
C       addresses sent over by the slave.
C

C
C       Local Variables
C
        INTEGER*4 STATUS

C
C       Common variables and areas
C
        INTEGER*4 IDEVMSG(32)                   ! Incoming device messages
        INTEGER*4 ODEVMSG(32)                   ! Outgoing device messages
        INTEGER*4 REM_BFRADS(25)                ! Remote buffer addresses
        INTEGER*4 FILEATTR(6)                   ! File attributes
        INTEGER*4 CSTATUS                       ! Common status
        INTEGER*4 LASTBFRSIZ                    ! Last buffer size
        INTEGER*4 DDIDIS                        ! DDI disable
        INTEGER*2 MRMS_CNT                      ! Master RMS count
        INTEGER*2 MRMS_IDX                      ! Master RMS index
        INTEGER*2 QPKT_CNT                      ! Queue packet count
        INTEGER*2 QPKT_IDX                      ! Queue packet index
        INTEGER*2 REM_CNT                       ! Remote buffer count
        INTEGER*2 REM_IDX                       ! Remote buffer index
        INTEGER*2 NUMREM_BFRS                   ! Number of remote buffers
        LOGICAL*1 GPFLAG                        ! Get/put flag
        LOGICAL*1 LASTBFR                       ! Last buffer flag
        LOGICAL*1 EOFFLAG                       ! End of file flag
        LOGICAL*1 ERRFLAG                       ! Error flag
        LOGICAL*1 REMFLAG                       ! Remote error flag

        COMMON /MDATA/  IDEVMSG,ODEVMSG,REM_BFRADS,FILEATTR,CSTATUS,
        1               LASTBFRSIZ,DDIDIS,MRMS_CNT,MRMS_IDX,QPKT_CNT,
        2               QPKT_IDX,REM_CNT,REM_IDX,NUMREM_BFRS,GPFLAG,
        3               LASTBFR,EOFFLAG,ERRFLAG,REMFLAG

        INTEGER*4 SYS$SETEF

        EXTERNAL SS$_NORMAL


        NUMREM_BFRS = IDEVMSG(2)                ! Number of remote buffers
        REM_CNT = NUMREM_BFRS

        DO 100 I = 1,NUMREM_BFRS
100     REM_BFRADS(I) = IDEVMSG(I+2)            ! Store each address

        CSTATUS = %LOC(SS$_NORMAL)
        STATUS = SYS$SETEF(%VAL(1))             ! Set event flag
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)

        RETURN
```

```
      END
```

```
        SUBROUTINE MFQ_FILEATTR

C
C       This routine copies the file attributes sent by the remote
C       system (at the start of a GET) from the input device message array
C       to the file attributes array and then sets an event flag.
C

C
C       Local Variables
C
        INTEGER*4 STATUS

C
C       Common variables and areas
C
        INTEGER*4 IDEVMSG(32)                   ! Incoming device messages
        INTEGER*4 ODEVMSG(32)                   ! Outgoing device messages
        INTEGER*4 REM_BFRADS(25)                ! Remote buffer addresses
        INTFGER*4 FILEATTR(6)                   ! File attributes
        INTEGER*4 CSTATUS                       ! Common status
        INTEGER*4 LASTBFRSIZ                    ! Last buffer size
        INTEGER*4 DDIDIS                        ! DDI disable
        INTEGER*2 MRMS_CNT                      ! Master RMS count
        INTEGER*2 MRMS_IDX                      ! Master RMS index
        INTEGER*2 QPKT_CNT                      ! Queue packet count
        INTEGER*2 QPKT_IDX                      ! Queue packet index
        INTEGER*2 REM_CNT                       ! Remote buffer count
        INTEGER*2 REM_IDX                       ! Remote buffer index
        INTEGER*2 NUMREM_BFRS                   ! Number of remote buffers
        LOGICAL*1 GPFLAG                        ! Get/put flag
        LOGICAL*1 LASTBFR                       ! Last buffer flag
        LOGICAL*1 EOFFLAG                       ! End of file flag
        LOGICAL*1 ERRFLAG                       ! Error flag
        LOGICAL*1 REMFLAG                       ! Remote error flag

        COMMON /MDATA/  IDEVMSG,ODEVMSG,REM_BFRADS,FILEATTR,CSTATUS,
        1               LASTBFRSIZ,DDIDIS,MRMS_CNT,MRMS_IDX,QPKT_CNT,
        2               QPKT_IDX,REM_CNT,REM_IDX,NUMREM_BFRS,GPFLAG,
        3               LASTBFR,EOFFLAG,ERRFLAG,REMFLAG

        INTEGER*4 SYS$SETEF


        DO 100 I = 1,6
        FILEATTR(I) = IDEVMSG(I+2)
100     CONTINUE

        STATUS = SYS$SETEF(%VAL(2))
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)

        RETURN
        END
```

```
DRS

C
C
C O
C--
```

```
        SUBROUTINE MFQ_PNXTBFR

C
C       This subroutine is called when the slave sends a message
C       indicating that it has processed its next buffer.
C

C
C       Common variables and areas
C
        INTEGER*4 IDEVMSG(32)                   ! Incoming device messages
        INTEGER*4 ODEVMSG(32)                   ! Outgoing device messages
        INTEGER*4 REM_BFRADS(25)                ! Remote buffer addresses
        INTEGER*4 FILEATTR(6)                   ! File attributes
        INTEGER*4 CSTATUS                       ! Common status
        INTEGER*4 LASTBFRSIZ                    ! Last buffer size
        INTEGER*4 DDIDIS                        ! DDI disable
        INTEGER*2 MRMS_CNT                      ! Master RMS count
        INTEGER*2 MRMS_IDX                      ! Master RMS index
        INTEGER*2 QPKT_CNT                      ! Queue packet count
        INTEGER*2 QPKT_IDX                      ! Queue packet index
        INTEGER*2 REM_CNT                       ! Remote buffer count
        INTEGER*2 REM_IDX                       ! Remote buffer index
        INTEGER*2 NUMREM_BFRS                   ! Number of remote buffers
        LOGICAL*1 GPFLAG                        ! Get/put flag
        LOGICAL*1 LASTBFR                       ! Last buffer flag
        LOGICAL*1 EOFFLAG                       ! End of file flag
        LOGICAL*1 ERRFLAG                       ! Error flag
        LOGICAL*1 REMFLAG                       ! Remote error flag

        COMMON /MDATA/  IDEVMSG,ODEVMSG,REM_BFRADS,FILEATTR,CSTATUS,
        1               LASTBFRSIZ,DDIDIS,MRMS_CNT,MRMS_IDX,QPKT_CNT,
        2               QPKT_IDX,REM_CNT,REM_IDX,NUMREM_BFRS,GPFLAG,
        3               LASTBFR,EOFFLAG,ERRFLAG,REMFLAG

        IF (ERRFLAG) RETURN                     ! Return if ERRFLAG is set

C
C       Increment the number of remote buffers available.  If
C       we have a matching local buffer, then queue an operation
C       to the DR32.
C
        REM_CNT = REM_CNT +1
        IF (QPKT_CNT .GT. 0) CALL QUEUE_PKT

        RETURN
        END
        SUBROUTINE MFQ_PLSTBFR

C
C       This routine is called when the slave sends a message
C       indicating that it has processed its last buffer.
C

C
C       Local Variables
C
        INTEGER*4 STATUS

C
C       Common variables and areas
C
        INTEGER*4 IDEVMSG(32)                   ! Incoming device messages
        INTEGER*4 ODEVMSG(32)                   ! Outgoing device messages
        INTEGER*4 REM_BFRADS(25)                ! Remote buffer addresses
        INTEGER*4 FILEATTR(6)                   ! File attributes
        INTEGER*4 CSTATUS                       ! Common status
        INTEGER*4 LASTBFRSIZ                    ! Last buffer size
        INTEGER*4 DDIDIS                        ! DDI disable
        INTEGER*2 MRMS_CNT                      ! Master RMS count
        INTEGER*2 MRMS_IDX                      ! Master RMS index
        INTEGER*2 QPKT_CNT                      ! Queue packet count
        INTEGER*2 QPKT_IDX                      ! Queue packet index
        INTEGER*2 REM_CNT                       ! Remote buffer count
        INTEGER*2 REM_IDX                       ! Remote buffer index
        INTEGER*2 NUMREM_BFRS                   ! Number of remote buffers
        LOGICAL*1 GPFLAG                        ! Get/put flag
        LOGICAL*1 LASTBFR                       ! Last buffer flag
        LOGICAL*1 EOFFLAG                       ! End of file flag
        LOGICAL*1 ERRFLAG                       ! Error flag
        LOGICAL*1 REMFLAG                       ! Remote error flag

        COMMON /MDATA/  IDEVMSG,ODEVMSG,REM_BFRADS,FILEATTR,CSTATUS,
        1               LASTBFRSIZ,DDIDIS,MRMS_CNT,MRMS_IDX,QPKT_CNT,
        2               QPKT_IDX,REM_CNT,REM_IDX,NUMREM_BFRS,GPFLAG,
        3               LASTBFR,EOFFLAG,ERRFLAG,REMFLAG

        INTEGER*4 SYS$SETEF

        EXTERNAL SS$_NORMAL

        IF (ERRFLAG) RETURN                     ! Return if ERRFLAG is set
C
C       If this is a GET then we have to read the last buffer.  If
C       this is a PUT, then we are all done.
C
        IF (GPFLAG .EQ. 1) THEN                 ! Doing a GET
            EOFFLAG = .TRUE.                    ! Set end of file flag
            LASTBFRSIZ = IDEVMSG(2)             ! Save last buffer size
            REM_CNT = REM_CNT + 1               ! Inc. remote bfr count
            IF (QPKT_CNT .GT. 0) CALL QUEUE_PKT ! Queue a read if possible
        ELSE                                    ! Doing a PUT
            CSTATUS = %LOC(SS$_NORMAL)          ! Set success status
```

```
          STATUS = SYS$SETEF(%VAL(3))           ! Wake up main level
          IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)
      END IF

      RETURN
      END
```

```
        SUBROUTINE MFQ_ERROR
C
C       This subroutine is called when the remote system sends
C       an error message
C
C
C       Local Variables
C
        INTEGER*4 STATUS
C
C       Common variables and areas
C
        INTEGER*4 IDEVMSG(32)                   ! Incoming device messages
        INTEGER*4 ODEVMSG(32)                   ! Outgoing device messages
        INTEGER*4 REM_BFRADS(25)                ! Remote buffer addresses
        INTEGER*4 FILEATTR(6)                   ! File attributes
        INTEGER*4 CSTATUS                       ! Common status
        INTEGER*4 LASTBFRSIZ                    ! Last buffer size
        INTEGER*4 DDIDIS                        ! DDI disable
        INTEGER*2 MRMS_CNT                      ! Master RMS count
        INTEGER*2 MRMS_IDX                      ! Master RMS index
        INTEGER*2 QPKT_CNT                      ! Queue packet count
        INTEGER*2 QPKT_IDX                      ! Queue packet index
        INTEGER*2 REM_CNT                       ! Remote buffer count
        INTEGER*2 REM_IDX                       ! Remote buffer index
        INTEGER*2 NUMREM_BFRS                   ! Number of remote buffers
        LOGICAL*1 GPFLAG                        ! Get/put flag
        LOGICAL*1 LASTBFR                       ! Last buffer flag
        LOGICAL*1 EOFFLAG                       ! End of file flag
        LOGICAL*1 ERRFLAG                       ! Error flag
        LOGICAL*1 REMFLAG                       ! Remote error flag

        COMMON /MDATA/  IDEVMSG,ODEVMSG,REM_BFRADS,FILEATTR,CSTATUS,
        1               LASTBFRSIZ,DDIDIS,MRMS_CNT,MRMS_IDX,QPKT_CNT,
        2               QPKT_IDX,REM_CNT,REM_IDX,NUMREM_BFRS,GPFLAG,
        3               LASTBFR,EOFFLAG,ERRFLAG,REMFLAG

        INTEGER*4 SYS$SETEF


        ERRFLAG = .TRUE.                        ! Set error flag
        REMFLAG = .TRUE.                        ! Set remote error flag
        CSTATUS = IDEVMSG(2)                    ! Get error status
C
C       Set event flags 1 and 2 and conditionally 3
C
        STATUS = SYS$SETEF(%VAL(1))
        IF (.NOT. STATUS)CALL FATAL_ERROR(STATUS)
        STATUS = SYS$SETEF(%VAL(2))
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)

        IF (GPFLAG.EQ.1 .AND. MRMS_CNT .EQ. -1) ! Get
        1     STATUS = SYS$SETEF(%VAL(3))
        IF (GPFLAG.EQ.3 .AND. EOFFLAG)          ! Put
        1     STATUS = SYS$SETEF(%VAL(3))
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)

        RETURN
        END
```

```
DRS

C
C T
C C
C I
C M
C M
C A

C
C I
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C I
C I
C
C

C
C (
C
```

```
        SUBROUTINE ERROR(CSTATUS,REMFLAG)

C
C       This routine prints error messages and returns.  If REMFLAG
C       is set the error message is preceded by a message saying
C       that the error is from the remote system.
C

        INTEGER*2 LENGTH

        INTEGER*4 STATUS,CSTATUS

        LOGICAL REMFLAG

        CHARACTER*256 MSGBFR

        INTEGER*4 SYS$GETMSG

        IF (REMFLAG) WRITE(6,100)
100     FORMAT(1X,'%DRCOPY-E-REMERROR, error from remote system:')

        STATUS=SYS$GETMSG(%VAL(CSTATUS),LENGTH,MSGBFR,%VAL(15),)
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)

        WRITE(6,200)MSGBFR(1:LENGTH)
200     FORMAT(1X,A)

        RETURN
        END
        SUBROUTINE RMS_ERROR(ISTATUS)

C
C       This subroutine sends an error packet to the remote system.
C

        INCLUDE 'SYS$LIBRARY:XFDEF.FOR/NOLIST'  ! DR32 definitions
        INCLUDE 'DRCOPY.PRM/NOLIST'             ! Parameters

C
C       Local Variables
C
        INTEGER*4 ISTATUS,STATUS

C
C       Common variables and areas
C
        INTEGER*4 XFDATA(30)                    ! Context array
        BYTE MBFRS(BUFSIZ,NUM_MBFRS)            ! Master buffers
        BYTE SBFRS(BUFSIZ,NUM_SBFRS)            ! Slave buffers

        COMMON /MS_SHARE/  XFDATA,MBFRS,SBFRS

        INTEGER*4 IDEVMSG(32)                   ! Incoming device messages
        INTEGER*4 ODEVMSG(32)                   ! Outgoing device messages
        INTEGER*4 REM_BFRADS(25)                ! Remote buffer addresses
        INTEGER*4 FILEATTR(6)                   ! File attributes
        INTEGER*4 CSTATUS                       ! Common status
        INTEGER*4 LASTBFRSIZ                    ! Last buffer size
        INTEGER*4 DDIDIS                        ! DDI disable
        INTEGER*2 MRMS_CNT                      ! Master RMS count
        INTEGER*2 MRMS_IDX                      ! Master RMS index
        INTEGER*2 QPKT_CNT                      ! Queue packet count
        INTEGER*2 QPKT_IDX                      ! Queue packet index
        INTEGER*2 REM_CNT                       ! Remote buffer count
        INTEGER*2 REM_IDX                       ! Remote buffer index
        INTEGER*2 NUMREM_BFRS                   ! Number of remote buffers
        LOGICAL*1 GPFLAG                        ! Get/put flag
        LOGICAL*1 LASTBFR                       ! Last buffer flag
        LOGICAL*1 EOFFLAG                       ! End of file flag
        LOGICAL*1 ERRFLAG                       ! Error flag
        LOGICAL*1 REMFLAG                       ! Remote error flag

        COMMON /MDATA/  IDEVMSG,ODEVMSG,REM_BFRADS,FILEATTR,CSTATUS,
     1                  LASTBFRSIZ,DDIDIS,MRMS_CNT,MRMS_IDX,QPKT_CNT,
     2                  QPKT_IDX,REM_CNT,REM_IDX,NUMREM_BFRS,GPFLAG,
     3                  LASTBFR,EOFFLAG,ERRFLAG,REMFLAG

        INTEGER*4 SYS$SETEF


        ERRFLAG = .TRUE.                        ! Set error flag
        CSTATUS = ISTATUS                       ! Store status

        ODEVMSG(1) = 9                          ! Message type
        ODEVMSG(2) = ISTATUS                    ! Status
        CALL XF$PKTBLD(XFDATA,                  ! Send packet
        1               XF$K_PKT_WRTCM,,,       ! Func.,index,size
        2               ODEVMSG,8,,             ! Msg, size, log area
        3               64,,,                   ! Int. if Q empty, no action routine
        4               STATUS)                 ! Status
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)

        STATUS=SYS$SETEF(%VAL(3))               ! Wake up main level
        IF (.NOT. STATUS) CALL FATAL_ERROR(STATUS)

        RETURN
        END
```

DRS

```
        SUBROUTINE DR32_ERROR
C
C       This subroutine prints the I/O status block for DR32 errors
C       Note that this routine does not return
C

        INCLUDE 'DRCOPY.PRM/NOLIST'                  ! Parameters

C
C       Common variables and areas
C
        INTEGER*4 XFDATA(30)                         ! Context array
        BYTE MBFRS(BUFSIZ,NUM_MBFRS)                 ! Master buffers
        BYTE SBFRS(BUFSIZ,NUM_SBFRS)                 ! Slave buffers

        COMMON /MS_SHARE/  XFDATA,MBFRS,SBFRS

        INTEGER*4 IOSB(2)

        EQUIVALENCE(IOSB,XFDATA)

        WRITE(6,100)
100     FORMAT(1X,'%DRCOPY-F-DR32ERR, DR32 error')
        WRITE(6,200)IOSB(2)
200     FORMAT(1X,'IOSB(2) = ',Z8,' (Hex)')
        CALL LIB$STOP(%VAL(IOSB(1)))
        END
        SUBROUTINE FATAL_ERROR(STATUS)

C
C       This routine signals fatal errors and exits
C

        INTEGER*4 STATUS

        CALL LIB$STOP(%VAL(STATUS))
        END
```

```
C
C

C
C
C
C
C
```

0158 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

XALINK
MAR

LABIOLINK
COM

DRMASTER
FOR

LPATEST
FOR

LABIOPEAK
FOR

LABIOSTRT
COM

XAMESSAGE
MAR

XATEST
FOR

LABIOCOM
FOR

LBRDEMO
COM

LABMBXDEF
FOR

LABIOSAMP
FOR

MAILCOMPRESS
COM

LABCHNDEF
FOR

CONNECT
COM

LABIOCON
FOR

LBRDEMO
FOR

PEAK
FOR

DRCOPYBLD
COM

XIDRIVER
MAR

LABIOSEC
FOR

DRSLAVE
FOR

LABIOACQ
FOR

LABIOCOMP
COM

LABIOSTAT
FOR

TESTLABIO
FOR