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
```

```
DDDDDDDD    RRRRRRRR      SSSSSSSS  LL          AAAAAA    VV        VV  EEEEEEEEEE
DDDDDDDD    RRRRRRRR      SSSSSSSS  LL          AAAAAA    VV        VV  EEEEEEEEEE
DD      DD  RR      RR  SS          LL        AA      AA  VV        VV  EE
DD      DD  RR      RR  SS          LL        AA      AA  VV        VV  EE
DD      DD  RR      RR  SS          LL        AA      AA  VV        VV  EE
DD      DD  RR      RR  SS          LL        AA      AA  VV        VV  EE
DD      DD  RRRRRRRR      SSSSSS    LL        AA      AA  VV        VV  EEEEEEEE
DD      DD  RRRRRRRR      SSSSSS    LL        AA      AA  VV        VV  EEEEEEEE
DD      DD  RR  RR              SS  LL        AAAAAAAAAA  VV        VV  EE
DD      DD  RR  RR              SS  LL        AAAAAAAAAA  VV        VV  EE
DD      DD  RR    RR            SS  LL        AA      AA    VV    VV    EE                ....
DD      DD  RR    RR            SS  LL        AA      AA    VV    VV    EE                ....
DDDDDDDD    RR      RR  SSSSSSSS    LLLLLLLLLL  AA      AA      VV      EEEEEEEEEE        ....
DDDDDDDD    RR      RR  SSSSSSSS    LLLLLLLLLL  AA      AA      VV      EEEEEEEEEE        ....

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               DRSLAVE
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                          COPYRIGHT (c) 1978 BY
C            DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
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 FACILITY:     DRCOPY -- DR32 example file transfer program
C
C ABSTRACT:
C       This set of routines constitutes the Slave portion of the
C       DRCOPY file transfer example program.
C
C ENVIRONMENT:
C       These routines run in User mode; no privileges are necessary.
C
C AUTHOR:  Trudy Matthews,      CREATION DATE: July, 1979
C
C MODIFIED BY:
```

```
C
C          : VERSION
C 01     -
C--
```

```
        SUBROUTINE      SFQ_STARTGET(DEVMSG)

C
C This subroutine is called when the Slave receives a "start read
C operation" control message.  This routine opens an existing file.
C If the OPEN is successful, then send a confirmation control message
C to the Master program and initialize the buffer management variables.
C
C INPUTS:
C       A Device Message of the format:
C               31                              0
C               +-------------------------------+
C               :   type = MS_MSG_STARTGET      :
C               +-------------------------------+
C               :   number of master buffers    :
C               +-------------------------------+
C               :                               :
C               //       file attribute        //
C               :         information           :
C               +-------------------------------+
C
C IMPLICIT INPUTS:
C IMPLICIT OUTPUTS:
C       The file specified by the file att ibute information is opened,
C       and an RMS read from the file int. the first buffer is issued.
C

        INCLUDE 'SYS$LIBRARY:XFDEF.FOR/NOLIST'
        INCLUDE 'DRCOPY.PRM'

        BYTE            MBFRS(BUFSIZ, NUM_MBFRS)
        BYTE            SBFRS(BUFSIZ, NUM_SBFRS)

        INTEGER*2       PTR             !index into ring of buffers
        INTEGER*2       RMSCOUNT        !number of buffers in queue
        INTEGER*4       ADDRMSG(NUM_SBFRS+2)    !"Here are my bfr addrs"
                                                .device message
        INTEGER*4       CONTXT(30)      !context array
        INTEGER*4       DEVMSG(32)      !input device message
        INTEGER*4       LASTCNT         !# bytes in last buffer
        INTEGER*4       STATUS
        INTEGER*4       SYS$CLREF       !integer function
        LOGICAL*1       FLAG, SLVDONE, ENDISNEAR

        COMMON  /MS_SHARE/ CONTXT, MBFRS, SBFRS
        COMMON  /SLV/   SLVDONE
        COMMON  /SLAVE/ RMSCOUNT, PTR, FLAG
        COMMON  /SLVWRT/ LASTCNT, ENDISNEAR

        EXTERNAL        SLV_OPEN        !macro routine; does RMS OPEN
        EXTERNAL        SLV_COPYFAB     !macro routine; alters FAB
        EXTERNAL        XF$PKTBLD       !DR32 support routine
        EXTERNAL        SLV_FINISH      !called to end transfer
        EXTERNAL        SYS$CLREF, SS$_IVBUFLEN

C
```

```
C Check that master buffers and slave buffers agree in size
C
        IF (DEVMSG(2) .NE. BUFSIZ) THEN
                CALL SLV_FINISH(SM_MSG_ERROR, %LOC(SS$_IVBUFLEN))
                RETURN
        END IF

C
C Open the file
C
        CALL SLV_OPEN(DEVMSG, STATUS)   !contains FAB
        IF (.NOT. STATUS) THEN
                CALL SLV_FINISH(SM_MSG_ERROR, STATUS)
                RETURN
        END IF

C
C Send a packet notifying M .ter of successful open
C
        ADDRMSG(1) = SM_MSG_BFRADRS
        ADDRMSG(2) = NUM_SBFRS
        DO 10   I = 1, NUM_SBFRS                !build device msg that conveys
                ADDRMSG(I+2) = %LOC(SBFRS(1,I)) !buffer addresses
10      CONTINUE

        CALL XF$PKTBLD
        1       (CONTXT,
        1        XF$K_PKT_WRTCM,        !write control message function
        1        ,,                     !default index & difsize
        1        ADDRMSG,               !send addresses of buffers
        1        (NUM_SBFRS + 2) * 4,   !size of ADDRMSG in bytes
        1        ,                      !no logmsg
        1        256,                   !modes = insert pkt at head of q
        1        ,,                     !no action, actparm
        1        STATUS)

        IF (.NOT. STATUS) THEN
                CALL SLV_FINISH(SM_MSG_ERROR, STATUS)
                RETURN
        END IF

C
C Send a control message to Master containing File Attributes
C
        DEVMSG(1) = SM_MSG_FAB
        CALL SLV_COPYFAB(DEVMSG)        !put attributes in same devmsg
        CALL XF$PKTBLD
        1       (CONTXT,
        1        XF$K_PKT_WRTCM,        !write control message function
        1        ,,                     !no index, difsize
        1        DEVMSG,                !send file attributes
        1        128,                   !size of device message
        1        ,                      !no log message
        1        256,                   !modes = insert packet at head
        1        ,,                     !no action, actparm
        1        STATUS)
        IF (.NOT. STATUS) THEN
                CALL SLV_FINISH(SM_MSG_ERROR, STATUS)
                RETURN
        END IF

C
C Initialize the buffer management variables
C
        FLAG = GET
        RMSCOUNT = NUM_SBFRS                    !# empty bfrs available for
                                        !slave to fill
        PTR = 1                         !index of next bfr to fill

        STATUS = SYS$CLREF(%VAL(SLVEF)) !clear slave event flag

        SLVDONE = .FALSE.
        ENDISNEAR = .FALSE.

        RETURN
        END
        SUBROUTINE      SFQ_GOGET
C
C This routine is called during a GET operation when the Master routine
C signals that his initialization is complete and he is ready to accept
C buffers of data.
C

        INCLUDE 'DRCOPY.PRM/NOLIST'

        BYTE            MBFRS(BUFSIZ, NUM_MBFRS)
        BYTE            SBFRS(BUFSIZ, NUM_SBFRS)

        INTEGER*2       PTR
        INTEGER*2       RMSCOUNT
        INTEGER*4       CONTXT(30)
        INTEGER*4       STATUS

        LOGICAL*1       FLAG

        EXTERNAL        SLV_CHKRMS      !RMS completion routine

        COMMON  /MS_SHARE/      CONTXT, MBFRS, SBFRS
        COMMON  /SLAVE/         RMSCOUNT, PTR, FLAG

C
C Issue READ to get things going
C
        CALL SLV_READ(SBFRS(1,PTR),BUFSIZ,SLV_CHKRMS,SLV_CHKRMS,STATUS)
        IF (.NOT. STATUS) CALL SLV_FINISH(SM_MSG_ERROR, STATUS)

        RETURN
        END
        SUBROUTINE      SFQ_STARTPUT(DEVMSG)
C
C This subroutine is called when the Slave receives a "start write
C operation" control message.  This routine must create a file.
C If the file is successfully created, then send a confirmation
C message to Master, initialize the buffer management variables, and
C return.  The process waits for a control message from Master to
C activate.
C
C INPUTS:
C       Device message from Master of the format:
C               31                              0
C               +-------------------------------+
C               :   type = MS_MSG_STARTPUT      :
C               +-------------------------------+
C               :   size of master buffers      :
C               +-------------------------------+
C               :                               :
C               //     file attribute          //
C               :       information             :
C               +-------------------------------+
C
C IMPLICIT INPUTS:
C IMPLICIT OUTPUTS:
C       The file specified by the device message is created.
C
        INCLUDE 'SYS$LIBRARY:XFDEF.FOR/NOLIST'
        INCLUDE 'DRCOPY.PRM/NOLIST'

        BYTE            MBFRS(BUFSIZ, NUM_MBFRS)
        BYTE            SBFRS(BUFSIZ, NUM_SBFRS)

        INTEGER*2       RMSCOUNT, PTR
        INTEGER*4       CONTXT(30)
        INTEGER*4       ADDRMSG(NUM_SBFRS+2)
        INTEGER*4       DEVMSG(32)      !input device message
        INTEGER*4       LASTCNT         !# bytes in last buffer
        INTEGER*4       STATUS
        INTEGER*4       SYS$CLREF       !integer function
        LOGICAL*1       FLAG, SLVDONE, ENDISNEAR

        COMMON  /MS_SHARE/ CONTXT, MBFRS, SBFRS
        COMMON  /SLV/   SLVDONE
        COMMON  /SLAVE/ RMSCOUNT, PTR, FLAG
        COMMON  /SLVWRT/ LASTCNT, ENDISNEAR

        EXTERNAL        SLV_CREATE      !macro routine; does RMS CREATE
        EXTERNAL        XF$PKTBLD       !DR32 support routine
        EXTERNAL        SLV_FINISH      !called to end transfer
        EXTERNAL        SYS$CLREF, SS$_IVBUFLEN
C
C Check that the sizes of Master and Slave buffers agree
C
        IF (DEVMSG(2) .NE. BUFSIZ) THEN
```

```
                CALL SLV_FINISH (SM_MSG_ERROR, %LOC(SS$_IVBUFLEN))
                RETURN
        END IF

C
C Create the file
C
        CALL SLV_CREATE (DEVMSG, STATUS)
        IF (.NOT. STATUS) THEN
                CALL SLV_FINISH(SM_MSG_ERROR, STATUS)
                RETURN
        END IF

C
C Send a packet notifying Master of a successful open
C
        ADDRMSG(1) = SM_MSG_BFRADRS
        ADDRMSG(2) = NUM_SBFRS
        DO 10   I = 1, NUM_SBFRS        !for all buffers do
                ADDRMSG(I+2) = %LOC(SBFRS(1,I)) !get address
10      CONTINUE

        CALL XF$PKTBLD
        1      (CONTXT,
        1       XF$K_PKT_WRTCM,         !write command message function
        1       ,,                      !no index,difsiz
        1       ADDRMSG,                !send addresses of buffers
        1       (NUM_SBFRS+2)*4,        !size of ADDRMSG in bytes
        1       ,                       !no log message
        1       256,                    !modes = insert packet at head
        1       ,,                      !no action, actparm
        1       STATUS)

        IF (.NOT. STATUS) THEN
                CALL SLV_FINISH(SM_MSG_ERROR, STATUS)
                RETURN
        END IF

C
C Initialize the buffer management varaibles
C
        STATUS = SYS$CLREF (%VAL(SLVEF))
        IF (.NOT. STATUS) THEN
                CALL SLV_FINISH(SM_MSG_ERROR, STATUS)
                RETURN
        END IF
        SLVDONE = .FALSE.               !transfer is not complete
        ENDISNEAR = .FALSE.             !set when Master sends
                                        !"end of transfer" cntrl msg
        FLAG = PUT
        RMSCOUNT = 0                    !# bfrs available for slave
                                        !to empty
        PTR = 1                         !next bfr to empty

        RETURN
        END
        SUBROUTINE      SFQ_PNXTBFR(DEVMSG)
C
C This routine is called when the Slave receives a device message
C "process your next buffer"
C
C
C INPUTS:
C       Device Message; contains message type MS_MSG_PNXTBFR.
C
C IMPLICIT INPUTS
C IMPLICIT OUTPUTS:
C       Buffer management data updated to reflect the fact that the
C       Master program has completed a DR-transfer of another Slave
C       buffer, and now the buffer is available to the RMS process.
C

        INCLUDE 'DRCOPY.PRM/NOLIST'

        BYTE            MBFRS(BUFSIZ, NUM_MBFRS)
        BYTE            SBFRS(BUFSIZ, NUM_SBFRS)

        INTEGER*2       RMSCOUNT        !if GET, # bfrs for slave to fil
                                        !if PUT, # bfrs to empty
        INTEGER*2       PTR             !PTR = current buffer
        INTEGER*4       DEVMSG(32)      !not used
        INTEGER*4       CONTXT(30)
        INTEGER*4       STATUS
        LOGICAL*1       FLAG, SLVDONE

        COMMON  /MS_SHARE/ CONTXT, MBFRS, SBFRS
        COMMON /SLV7    SLVDONE
        COMMON  /SLAVE/ RMSCOUNT, PTR, FLAG

        EXTERNAL        SLV_READ        !macro routine; does RMS READ
        EXTERNAL        SLV_WRITE       !macro routine; does RMS WRITE
        EXTERNAL        SLV_CHKRMS      !RMS success completion routine;
                                        !checks for end of file on read
        EXTERNAL        SLV_RMSERR      !RMS error completion routine
        EXTERNAL        SLV_BUFDONE     !RMS completion routine
        EXTERNAL        SLV_FINISH      !called to end transfer

C
C If transfer has prematurely aborted, simply return
C
        IF (SLVDONE) RETURN

C
C Else record the fact that there exists another buffer for RMS to
C operate on
C
        RMSCOUNT = RMSCOUNT + 1         !another bfr to fill/empty
        IF (RMSCOUNT .EQ. 1) THEN               !start or restart RMS
            IF (FLAG .EQ. PUT) THEN
                CALL SLV_WRITE (SBFRS(1, PTR), BUFSIZ,
        1                       SLV_BUFDONE, SLV_RMSERR, STATUS)
                IF (.NOT. STATUS) CALL SLV_FINISH (SM_MSG_ERROR, STATUS)
```

```
          ELSE
             CALL SLV_READ (SBFRS (1, PTR), BUFSIZ,
     1                      SLV_CHKRMS, SLV_CHKRMS, STATUS)
             IF (.NOT. STATUS) CALL SLV_FINISH (SM_MSG_ERROR, STATUS)
          END IF
       END IF

       RETURN
       END
```

```
        SUBROUTINE      SLV_BUFDONE
C
C This routine is called after RMS has completed a read/write operation
C to/from disk, making another slave buffer available to the Master
C process.  This routine must send a control message to the Master
C informing him the next buffer is available, then issue another RMS
C read/write.
C
C
C IMPLICIT INPUTS:
C       RMS process has completed the transfer of a buffer.
C
C IMPLICIT OUTPUTS:
C       Buffer management updated: another buffer available to
C       DR-transfer process.
C       If possible, start RMS processing next buffer in RMS queue.
C
        INCLUDE 'SYS$LIBRARY:XFDEF.FOR/NOLIST'
        INCLUDE 'DRCOPY.PRM/NOLIST'

        BYTE            MBFRS(BUFSIZ, NUM_MBFRS)
        BYTE            SBFRS(BUFSIZ, NUM_SBFRS)

        INTEGER*2       RMSCOUNT
        INTEGER*2       PTR
        INTEGER*4       CONTXT(30)
        INTEGER*4       LASTCNT
        INTEGER*4       STATUS
        LOGICAL*1       FLAG, SLVDONE, ENDISNEAR


        COMMON  /MS_SHARE/ CONTXT, MBFRS, SBFRS
        COMMON  /SLV/   SLVDONE
        COMMON  /SLAVE/ RMSCOUNT, PTR, FLAG
        COMMON  /SLVWRT/ LASTCNT, ENDISNEAR

        EXTERNAL        SLV_READ        !;macro routine; does RMS READ
        EXTERNAL        SLV_WRITE       !macro routine; does RMS WRITE
        EXTERNAL        SLV_CHKRMS      !RMS success completion routine;
                                        !checks for end of file on read
        EXTERNAL        SLV_RMSERR      !RMS error completion routine
        EXTERNAL        XF$PKTBLD       !DR32 support routine
        EXTERNAL        CALL_BUFDONE    !RMS success completion routine
        EXTERNAL        SLV_FINISH      !called to end transfer
        EXTERNAL        SLV_NORMAL      !RMS completion routine;
                                        !successful end of PUT operation
        EXTERNAL        SS$_NORMAL

C
C If transfer has prematurely aborted, simply return
C
        IF (SLVDONE) RETURN


C
C Send control message "process my next buffer"
```

```
C Insert this command packet at the head of the input queue
C
        CALL XF$PKTBLD
        1       (CONTXT,
        1        XF$K_PKT_WRTCM,          !write control message function
        1        ,,                       !no index,difsize
        1        SM_MSG_PNXTBFR,          !slave "next buffer" devmsg
        1        2,                       !size of device message
        1        ,                        !no logmsg
        1        256,                     !modes = insert pkt at head
        1        ,,                       !no action, actparm
        1        STATUS)

        IF (.NOT. STATUS) THEN
                CALL SLV_FINISH(SM_MSG_ERROR, STATUS)
                RETURN
        END IF

        PTR = PTR + 1                     !step to next buffer
        IF (PTR .GT. NUM_SBFRS) PTR = 1 !increment mod(NUM_SBFRS)
        RMSCOUNT = RMSCOUNT - 1           !slave finished a buffer

C
C The ENDISNEAR flag is set only during a PUT operation, when the
C device message "this is the last buffer" is received.  If the last
C buffer has been received and is also the only buffer left in the
C queue (RMSCOUNT = 1), then issue the last write.

        IF (ENDISNEAR .AND. RMSCOUNT .EQ. 1) THEN        !this is last buffer
                IF (LASTCNT .GT. 0) THEN
                        CALL SLV_WRITE (SBFRS(1,PTR), LASTCNT,
        1                       SLV_NORMAL, SLV_RMSERR, STATUS)
                        IF (.NOT. STATUS)
        1                       CALL SLV_FINISH (SM_MSG_ERROR, STATUS)
                ELSE            !0 bytes in last buffer transferred
                        CALL SLV_FINISH(SM_MSG_PLSTBFR, 1)
                END IF

        ELSE                                    !this is not last buffer
                IF (RMSCOUNT .NE. 0) THEN
                        IF (FLAG .EQ. PUT) THEN
                                CALL SLV_WRITE (SBFRS(1,PTR), BUFSIZ,
        1                               CALL_BUFDONE,SLV_RMSERR,STATUS)
                                IF (.NOT. STATUS) CALL SLV_FINISH
        1                                       (SM_MSG_ERROR, STATUS)
                        ELSE
                                CALL SLV_READ (SBFRS(1,PTR), BUFSIZ,
        1                               SLV_CHKRMS,SLV_CHKRMS,STATUS)
                                IF (.NOT. STATUS) CALL SLV_FINISH
        1                                       (SM_MSG_ERROR, STATUS)
                        END IF
                END IF
        END IF

        RETURN
```

```
      END
```

```
        SUBROUTINE      CALL_BUFDONE
C
C This subroutine exists because
C       (1) SLV_BUFDONE must specify itself as its success
C               RMS completion routine, and
C       (2) FORTRAN subroutines may not reference themselves.
C So SLV_BUFDONE specifies this routine as its completion routine.
C

        CALL SLV_BUFDONE
        RETURN
        END
        SUBROUTINE      SLV_CHKRMS

        INCLUDE 'DRCOPY.PRM/NOLIST'

        PARAMETER       RMS$_NORMAL = '10001'X
        PARAMETER       RMS$_EOF = '1827A'X

        INTEGER*4       XFRCNT
        INTEGER*4       RMSTAT
        INTEGER*4       GETBYTCNT
        INTEGER*4       GETRMSTAT

C
C SLV_CHKRMS is only called during GET operations.  It is called to
C determine if end-of-file was encountered during the read by
C comparing the requested transfer byte count (BUFSIZ) to the actual
C transfer count (returned by function subroutine GETBYTCNT).
C

        RMSTAT = GETRMSTAT()            !returns RMS completion status
        XFRCNT = GETBYTCNT()            !returns # of bytes transferred

        IF (RMSTAT .EQ. RMS$_NORMAL) THEN
                IF (XFRCNT .EQ. BUFSIZ) THEN    !not finished; read more
                        CALL SLV_BUFDONE
                        RETURN
                END IF

        ELSE IF (RMSTAT .NE. RMS$_EOF) THEN     !error

                CALL SLV_FINISH (SM_MSG_ERROR, RMSTAT)
                RETURN

        END IF

C
C Only get here if end-of-file was found, either by reading less than
C a full buffer of data or by receiving RMS$_EOF status code.
C Notify far-end that we just read the last buffer.
C

        CALL SLV_LASTRD (XFRCNT)

        RETURN
        END
```

```
LAB

! T
!
! F
! d
!
! I
! V
!
!

! P

! F
!

! T

! P
!

! C
!

!
! I
!

30

31
32

33
```

```
        SUBROUTINE      SFQ_PLSTBFR(DEVMSG)
C
C This subroutine is called when "last buffer" control message
C is received from Master.  (Only during a PUT operation)
C

        INCLUDE 'SYS$LIBRARY:XFDEF.FOR/NOLIST'
        INCLUDE 'DRCOPY.PRM/NOLIST'

        BYTE            MBFRS(BUFSIZ, NUM_MBFRS)
        BYTE            SBFRS(BUFSIZ, NUM_SBFRS)

        INTEGER*2       RMSCOUNT
        INTEGER*2       PTR
        INTEGER*4       CONTXT(30)
        INTEGER*4       DEVMSG(32)              !input device message
        INTEGER*4       LASTCNT                 !holds last byte count
        INTEGER*4       STATUS
        LOGICAL*1       FLAG, SLVDONE, ENDISNEAR

        COMMON  /MS_SHARE/ CONTXT, MBFRS, SBFRS
        COMMON  /SLV/   SLVDONE
        COMMON  /SLAVE/ RMSCOUNT, PTR, FLAG
        COMMON  /SLVWRT/ LASTCNT, ENDISNEAR

        EXTERNAL        XF$PKTBLD,SLV_CLOSE, SLV_RMSERR, SLV_FINISH
        EXTERNAL        SLV_NORMAL

C
C If transfer has prematurely aborted, simply return
C
        IF (SLVDONE) RETURN

C
C Since this is a PUT operation, the "last buffer" control message simply
C means that the last buffer to be written to disk has arrived and is
C on the end of the queue of buffers waiting for the RMS routine to
C write them to disk.  If this buffer is the only one left in the queue
C (i.e. if RMSCOUNT = 1) then call SLV_WRITE to write the last buffer
C to disk.  If there are other buffers to be written before this
C one, simply return -- the ENDISNEAR flag signals SLV_BUFDONE to
C notice when it is about to write out the last buffer, and it will
C finish up the transfer instead.
C

        LASTCNT = DEVMSG(2)     !save last byte count
        ENDISNEAR = .TRUE.      !signal last buffer in
        RMSCOUNT = RMSCOUNT + 1 !last buffer
        IF (RMSCOUNT .EQ. 1) THEN
                CALL SLV_WRITE (SBFRS(1,PTR), LASTCNT,
        1               SLV_NORMAL, SLV_RMSERR, STATUS)
                IF (.NOT. STATUS) CALL SLV_FINISH (SM_MSG_ERROR, STATUS)
        END IF

        RETURN
        END
        SUBROUTINE      SLV_LASTRD (XFRCNT)
C
C This subroutine is called when EOF is detected while reading from
C disk.  SBFRS(PTR) is the last buffer filled; XFRCNT is the number of
C bytes of good data it contains.
C Send "process my next(last) buffer" control msg to Master;
C Close the file;
C Set "slave tranfer complete" flag (SLVDONE)
C

        INCLUDE 'SYS$LIBRARY:XFDEF.FOR/NOLIST'
        INCLUDE 'DRCOPY.PRM/NOLIST'

        BYTE            MBFRS(BUFSIZ, NUM_MBFRS)
        BYTE            SBFRS(BUFSIZ, NUM_SBFRS)

        INTEGER*2       RMSCOUNT
        INTEGER*2       PTR
        INTEGER*4       CONTXT(30)
        INTEGER*4       ENDMSG(2)       !"process last bfr; bufsiz" msg
        INTEGER*4       XFRCNT          !bytes of data in last buffer
        INTEGER*4       STATUS
        LOGICAL*1       FLAG, SLVDONE

        COMMON  /MS_SHARE/ CONTXT, MBFRS, SBFRS
        COMMON  /SLV/   SLVDONE
        COMMON  /SLAVE/ RMSCOUNT, PTR, FLAG

        EXTERNAL        XF$PKTBLD,SLV_CLOSE

C
C If transfer has prematurely aborted, simply return
C
        IF (SLVDONE) RETURN


        ENDMSG(1) = SM_MSG_PLSTBFR              !"last buffer"
        ENDMSG(2) = XFRCNT              !# of bytes of good data

        CALL    XF$PKTBLD
        1       (CONTXT,
        1        XF$K_PKT_WRTCM,        !write control message function
        1        ,,                     !no index, difsize
        1        ENDMSG,
        1        8,                     !size in bytes of ENDMSG
        1        ,                      !no log message
        1        ,                      !modes = insert packet at tail
        1                               !(last buffer must be in order)
        1        ,,                     !no action, actparm
        1        STATUS)

        CALL SLV_SHUTDOWN

        RETURN
        END
        SUBROUTINE      SLV_NORMAL
C
C This routine is called when RMS completes the transfer of the last
C slave buffer to disk during a PUT operation.
C


C
C Finish up, sending success code (=1) to Master
C
        INCLUDE 'DRCOPY.PRM/NOLIST'

        CALL SLV_FINISH (SM_MSG_PLSTBFR, 1)

        RETURN
        END
        SUBROUTINE      SLV_FINISH (MSGCODE, MSG)
C
C This routine is called to send a status message to Master and then
C halt and clean up the slave transfer.
C

        INTEGER*4       MSGCODE
        INTEGER*4       MSG

        EXTERNAL        SLV_SENDSTAT, SLV_SHUTDOWN

        CALL SLV_SENDSTAT (MSGCODE, MSG)        !send status to Master

        CALL SLV_SHUTDOWN                       !end of slave transfer

        RETURN
        END
        SUBROUTINE      SLV_SENDSTAT (MSGCODE, MSG)
C
C This routine provides a centralized routine to call to send status
C packets to  Master routine.
C

        INCLUDE 'DRCOPY.PRM/NOLIST'
        INCLUDE 'SYS$LIBRARY:XFDEF.FOR/NOLIST'

        BYTE    MBFRS(BUFSIZ, NUM_MBFRS)
        BYTE    SBFRS(BUFSIZ, NUM_SBFRS)

        INTEGER*2       MSGCODE
        INTEGER*4       MSG
        INTEGER*4       CONTXT(30)
        INTEGER*4       DEVMSG(2)

        COMMON  /MS_SHARE/ CONTXT, MBFRS, SBFRS

        EXTERNAL        XF$PKTBLD

        DEVMSG(1) = MSGCODE
        DEVMSG(2) = MSG

        CALL XF$PKTBLD (
        1       CONTXT,
        1       XF$K_PKT_WRTCM,         !write control message function
        1       ,,                      !no index, size
        1       DEVMSG,                 !device message
        1       8,,                     !size in bytes of devmsg
        1       256)                    !modes = insert at head
C
C If this fails, there is no way to signal Master
C
        RETURN
        END
        SUBROUTINE      SLV_SHUTDOWN
C
C End of transfer
C
        INCLUDE 'DRCOPY.PRM/NOLIST'

        LOGICAL*1       SLVDONE

        COMMON  /SLV/   SLVDONE

        CALL SLV_CLOSE

        SLVDONE = .TRUE.
        CALL SYS$SETEF(%VAL(SLVEF))

        RETURN
        END
```

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