C 8

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

FILEID**MBXDRIVER

K 9

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

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

```
0000          .TITLE  MBXDRIVER - SHARED MEMORY MAILBOX DEVICE DRIVER
0000          .IDENT  'V04-001'
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000    ;++
0000    ; FACILITY:
0000    ;
0000    ;       VAX/VMS EXECUTIVE
0000    ;
0000    ; ABSTRACT:
0000    ;
0000    ;       THIS MODULE CONTAINS THE SHARED MEMORY MAILBOX DRIVER
0000    ;       I/O ROUTINES.
0000    ;
0000    ; AUTHOR: LEN KAWELL 13-MAR-1979
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V04-001 ACG0467         Andrew C. Goldstein,    12-Sep-1984  22:07
0000    ;               Fix protection holes in QIO device protection check
0000    ;
0000    ;       V03-017 LMP0271         L. Mark Pilant,         12-Jul-1984  12:28
0000    ;               Note, in the ORB, that shared memory mailboxes cannot have
0000    ;               ACLs.
0000    ;
0000    ;       V03-016 LMP0266         L. Mark Pilant,         27-Jun-1984  11:38
0000    ;               Add $CCBDEF for V03-015.
0000    ;
0000    ;       V03-015 LMP0265         L. Mark Pilant,         26-Jun-1984  15:27
0000    ;               Only do a protection check for the first I/O to the channel.
0000    ;
0000    ;       V03-014 RAS0300         Ron Schaefer            19-Jun-1984
0000    ;               Add DEV$M_NNM characteristic to DECHAR2 so that these
0000    ;               devices will have the "node$" prefix.
0000    ;
```

```
0000    ;       V03-013 WMC0001         Wayne Cardoza           17-May-1984
0000    ;               Previous update destroyed R4 before mutex calls.
0000    ;
0000    ;       V03-012 TMK0001         Todd M. Katz            21-Apr-1984
0000    ;               When deleting the logical name associated with a mailbox,
0000    ;               delete the logical name block by calling LNM$DELETE_LNMB
0000    ;               instead of LNM$DELETE. Doing so will ensure that this deletion
0000    ;               takes place as if the system service $DELLNM had been called
0000    ;               to delete the logical name. In other words, not only will the
0000    ;               target logical name be deleted, but so will all outer access
0000    ;               mode aliases.
0000    ;
0000    ;       V03-011 LMP0221         L. Mark Pilant,         27-Mar-1984  9:12
0000    ;               Change UCB$L_OWNUIC to ORB$L_OWNER and UCB$W_VPROT to
0000    ;               ORB$W_PROT.
0000    ;
0000    ;       V03-010 ROW0277         Ralph O. Weber          11-JAN-1984
0000    ;               Implement use of IO$M_NORSWAIT modifier to prevent resource
0000    ;               waits.
0000    ;
0000    ;       V03-009 DMW4039         DMWalp                  31-May-1983
0000    ;               Intergate new logical name structures.
0000    ;
0000    ;       V03-008 ROW0172         Ralph O. Weber          10-APR-1983
0000    ;               Change device type to DT$_SHRMBX.
0000    ;
0000    ;       V03-007 ROW0170         Ralph O. Weber          12-MAR-1983
0000    ;               Insert delete mailbox functionality from IOC$DELMBX in
0000    ;               CANCELIO.  This moves the mailbox specific knowledge of how to
0000    ;               delete a mailbox from $DASSGN into this driver.
0000    ;
0000    ;       V03-006 CWH1002         CW Hobbs                1-Mar-1983
0000    ;               Use extended pid in iosb process ids.
0000    ;
0000    ;       V03-005 ROW49973        Ralph O. Weber          29-OCT-1982
0000    ;               Make all changes necessary to have control transfered to
0000    ;               EXE$IORSNWAIT at IPL$_ASTDEL rather than IPL$_SYNCH.  This is
0000    ;               necessary to conform with internal changes in EXE$IORSNWAIT.
0000    ;
0000    ;       V03-004 ROW0118         Ralph O. Weber           7-JUL-1982
0000    ;               Change FINISHREAD to return SS$_BUFFEROVF instead of
0000    ;               SS$_DATAOVERUN.  SS$_BUFFEROVF is an alternate success status.
0000    ;               Its use in place of SS$_DATAOVERUN will allow the buffer
0000    ;               overflow condition to be reported to interested programs
0000    ;               without hassling uninterested programs with an error status.
0000    ;
0000    ;       V03-003 KDM0002         Kathleen D. Morse       28-Jun-1982
0000    ;               Added $DEVDEF and $PRVDEF.
0000    ;
0000    ;       V03-002 ROW0105         Ralph O. Weber          18-JUN-1982
0000    ;               Change FINISHREAD to return SS$_DATAOVERUN when number of
0000    ;               bytes in mail box message being read exceeds number of bytes
0000    ;               in user supplied buffer.
0000    ;
0000    ;       V03-001 ROW0104         Ralph O. Weber          18-JUN-1982
0000    ;               Make several changes to improve handling of zero length
0000    ;               messages in mailboxes.  Change READCHECKIO and WRITECHECKIO
```

```
0000  ;                 to allow zero-byte messages, and provide a dummy buffer address
0000  ;                 for such messages.  Add function code information to shared
0000  ;                 memory message so that zero length messages can be
0000  ;                 differentiated from end-of-file messages.
0000  ;                 This change is distributed as part of MBXDRIVER.EXE ECO 1 in
0000  ;                 Version 3.1.
0000  ;
0000  ;         V02-008 KDM0074          Kathleen D. Morse       8-Jan-1982
0000  ;                 Clear IDB pointer to UCB for shared memory mailbox,
0000  ;                 when no more references to the UCB and it is going
0000  ;                 to be deallocated.
0000  ;
0000  ;         V02-007 KDM0067          Kathleen D. Morse       10-Nov-1981
0000  ;                 Fix stack and synchronization problems.
0000  ;
0000  ;         V02-006 STJ0026          Steven T. Jeffreys      05-Feb-1981
0000  ;                 Modified FDTSET to default to IO$M_WRTATTN if no
0000  ;                 function modifier is present.
0000  ;
0000  ;         V02-005 STJ0020          Steven T. Jeffreys      20-Jan-1981
0000  ;                 Modified FDTSET routine to handle SETPROT function.
0000  ;
0000  ;
0000  ;--
0000  ;
0000  ; EXTERNAL SYMBOLS
0000  ;
0000            $ACBDEF                         ; DEFINE AST CONTROL BLOCK
0000            $CADEF                          ; DEFINE CONDITIONAL ASSEMBLY
0000            $CANDEF                         ; CANCEL REASON CODES
0000            $CCBDEF                         ; DEFINE CHANNEL CONTROL BLOCK OFFSETS
0000            $CRBDEF                         ; DEFINE CHANNEL REQUEST BLOCK
0000            $CXBDEF                         ; DEFINE COMPLEX CHAINED BUFFERS
0000            $DCDEF                          ; DEFINE DEVICE CLASSES & TYPES
0000            $DDBDEF                         ; DEFINE DDB
0000            $DEVDEF                         ; DEFINE DEVICE TYPES
0000            $DYNDEF                         ; DEFINE DYNAMIC BLOCK TYPES
0000            $FKBDEF                         ; DEFINE FORK BLOCK
0000            $IDBDEF                         ; DEFINE INTERRUPT DISPATCHER
0000            $IODEF                          ; DEFINE FUNCTION CODES
0000            $IRPDEF                         ; DEFINE I/O PACKET OFFSETS
0000            $IRPEDEF                        ; DEFINE I/O PACKET EXTENSION OFFSETS
0000            $IPLDEF                         ; DEFINE IPL NUMBERS
0000            $MBXDEF                         ; DEFINE MAILBOX
0000            $ORBDEF                         ; OBJECT'S RIGHTS BLOCK OFFSETS
0000            $PCBDEF                         ; DEFINE PCB OFFSETS
0000            $PRDEF                          ; DEFINE PROCESSOR REGISTERS
0000            $PRIDEF                         ; DEFINE PRIORITY INCREMENTS
0000            $PRQDEF                         ; DEFINE INTER-PROCESSOR REQUESTS
0000            $PRVDEF                         ; DEFINE PRIVILEGE NUMBERS
0000            $RSNDEF                         ; DEFINE RESOURCE NUMBERS
0000            $SHBDEF                         ; DEFINE SHARED MEMORY CONTROL BLOCK
0000            $SHDDEF                         ; DEFINE SHARED MEMORY DATAPAGE
0000            $SSDEF                          ; DEFINE SYSTEM STATUS CODES
0000            $UCBDEF                         ; DEFINE UCB OFFSETS
0000            $VECDEF                         ; DEFINE INTERRUPT TRANSFER VECTOR
0000
```

```
          0000     ;
          0000     ; LOCAL DEFINITIONS
          0000     ;
          0000
          0000     ;
          0000     ; MACRO TO SET PORT FLAG CORRESPONDING TO THIS PROCESSOR
          0000     ;
          0000             .MACRO  SET_PORTFLAG MASK,?LABEL
          0000             BBSSI   UCB$L_MB_PORT(R5),MASK,LABEL
          0000     LABEL:
          0000             .ENDM   SET_PORTFLAG
          0000     ;
          0000     ; MACRO TO CLEAR PORT FLAG CORRESPONDING TO THIS PROCESSOR
          0000     ;
          0000             .MACRO  CLR_PORTFLAG MASK,?LABEL
          0000             BBCCI   UCB$L_MB_PORT(R5),MASK,LABEL
          0000     LABEL:
          0000             .ENDM   CLR_PORTFLAG
          0000
          0000
          0000     ;
          0000     ; DEVICE SPECIFIC I/O REQUEST PACKET EXTENSION DEFINITIONS
          0000     ;
          0000             $DEFINI IRPE
          0000
00000018  0000     . = FKB$K_LENGTH                        ; (BEGINNING IS FORK BLOCK)
          0018     $DEF    IRPE$W_MB_PORTS .BLKW   1       ; PORTS TO NOTIFY (1 BIT/PORT)
          001A     $DEF    IRPE$W_MB_RQTYP .BLKW   1       ; REQUEST TYPE CODE
          001C     $DEF    IRPE$L_MB_PARAM .BLKL   1       ; REQUEST PARAMETER
          0020     $DEF    IRPE$L_MB_PORT  .BLKL   1       ; NEXT PORT TO NOTIFY
          0024
          0024             $DEFEND IRPE
          0000
          0000     ;
          0000     ; MAILBOX MESSAGE BUFFER DEFINITION
          0000     ;
          0000     ;       SINCE THE SHARED MEMORY POOL IS ONLY ALLOCATABLE IN FIXED
          0000     ;       SIZE BLOCKS, A MESSAGE IS STORED AS A LIST OF CHAINED BLOCKS.
          0000     ;
          0000             $DEFINI MSG
          0000     $DEF    MSG_Q_MSGLINK                   ; MESSAGE QUEUE LINK
          0000     $DEF    MSG_L_POSTIOBUF .BLKL   1       ;  I/O POST I/O BUFFER ADDRESS
          0004     $DEF    MSG_L_POSTUBUF  .BLKL   1       ;  I/O POST USER BUFFER ADDRESS
          0008     $DEF    MSG_W_SIZE      .BLKW   1       ; SIZE OF BLOCK
          000A     $DEF    MSG_B_TYPE      .BLKB   1       ; TYPE OF BLOCK (DYN$C_SHRBUFIO)
          000B     $DEF    MSG_B_PORT      .BLKB   1       ; PORT NUMBER OF MESSAGE WRITER
          000C     $DEF    MSG_W_LENGTH    .BLKW   1       ; LENGTH OF MESSAGE IN BLOCK
          000E     $DEF    MSG_W_MSGLENGTH .BLKW   1       ; TOTAL LENGTH OF MESSAGE DATA
          0010     $DEF    MSG_L_CHAINLINK .BLKL   1       ; LINK TO NEXT CHAINED BLOCK
          0014     $DEF    MSG_L_IRPSEQ    .BLKL   1       ; IRP SEQUENCE NUMBER OF MESSAGE WRITER
          0018     $DEF    MSG_L_PID       .BLKL   1       ; PID OF MESSAGE WRITER
          001C     $DEF    MSG_B_FUNC      .BLKB   1       ; ORIGINATING FUNCTION CODE
          001D     $DEF    MSG_B_MESSAGE                   ; START OF MESSAGE IN BLOCK
          001D             $DEFEND MSG
          0000     ;
          0000     ;       SINCE THE MESSAGE IS PASSED DIRECTLY TO I/O POST, IT MUST
          0000     ;       CONFORM TO THE DEFINITION FOR A COMPLEX CHAINED BUFFER
          0000  ;
          0000          ASSUME  MSG_L_POSTIOBUF EQ 0
          0000          ASSUME  MSG_L_POSTUBUF  EQ 4
          0000          ASSUME  MSG_W_LENGTH EQ CXB$W_LENGTH
          0000          ASSUME  MSG_L_CHAINLINK EQ CXB$L_LINK
          0000
          0000  ;
          0000  ; INTER-PROCESSOR REQUEST TYPE CODES
          0000  ;
00000001  0000  PRQ_READ        = 1                     ; MESSAGE WAS READ
00000002  0000  PRQ_WRITE       = 2                     ; MESSAGE WAS WRITTEN
00000003  0000  PRQ_READER      = 3                     ; READER IS WAITING
          0000
          0000  ;
          0000  ; FDT ROUTINE ARGUMENT LIST OFFSETS
          0000  ;
00000000  0000  P1              = 0                     ; BUFFER ADDRESS ARGUMENT
00000004  0000  P2              = 4                     ; BUFFER SIZE ARGUMENT
00000008  0000  P3              = 8                     ; PARAMETER 3
0000000C  0000  P4              = 12                    ; PARAMETER 4
          0000
          0000  ;
          0000  ; LOCAL DATA STORAGE
          0000  ;
          0000
          0000  ;
          0000  ; DRIVER PROLOGUE TABLE
          0000  ;
          0000          DPTAB   -                       ; DRIVER PROLOGUE TABLE
          0000                  END=MB_END,-            ; END OF DRIVER
          0000                  ADAPTER=MPM,-           ; MULTI-PORT MEMORY ADAPTER
          0000                  UCBSIZE=UCB$K_MB_LENGTH,- ; SIZE OF UCB
          0000                  NAME=MBXDRIVER          ; DRIVER NAME
          0038          DPT_STORE INIT                  ;
          0038                  DPT_STORE UCB,UCB$B_FIPL,B,IPL$_MAILBOX
          003C                  DPT_STORE UCB,UCB$B_DIPL,B,IPL$_MAILBOX
          0040                  DPT_STORE ORB,ORB$B_FLAGS,B,-           ; Protection block flags
          0040                                  <ORB$M_PROT_16!-        ; SOGW protection word
          0040                                  ORB$M_NOACL>            ; No ACLs allowed
          0044                  DPT_STORE ORB,ORB$W_PROT,@,0            ; default protection
          0049                  DPT_STORE ORB,ORB$L_OWNER,L,<^X010001>  ; [1,1] owns the device
          0050                  DPT_STORE UCB,UCB$L_DEVCHAR,L,-
          0050                          <DEV$M_REC!-
          0050                          DEV$M_AVL!-
          0050                          DEV$M_MBX!-
          0050                          DEV$M_IDV!-
          0050                          DEV$M_ODV!-
          0050                          DEV$M_SHR>
          0057                  DPT_STORE UCB,UCB$L_DEVCHAR2,L,-; DEVICE CHARACTERISTICS
          0057                          <DEV$M_NNM>             ; PREFIX NAME WITH "node$"
          005E                  DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_MAILBOX
          0062                  DPT_STORE UCB,UCB$B_DEVTYPE,B,DT$_SHRMBX
          0066                  DPT_STORE UCB,UCB$W_DEVSTS,W,UCB$M_SHMMBX
          006B          DPT_STORE REINIT                ;
          006B          DPT_STORE CRB,CRB$L_INTD+4,D,MBX$INT ;INTERRUPT SERVICE ROTINE ADDRESS
          0070          DPT_STORE DDB,DDB$L_DDT,D,MBX$DDT ;DDT ADDRESS
          0075          DPT_STORE END                   ;
```

```
0000
0000    ;
0000    ; DRIVER DISPATCH TABLE
0000    ;
0000            DDTAB   -                               ; DRIVER DISPATCH TABLE
0000                    DEVNAM=MBX,-                    ;  DEVICE NAME
0000                    START=STARTIO,-                 ;  START I/O OPERATION
0000                    FUNCTB=FUNCTABLE,-              ;  FUNCTION DECISION TABLE
0000                    CANCEL=CANCELIO                 ;  CANCEL I/O OPERATION
0038
0038    ;
0038    ; FUNCTION DECISION TABLE
0038    ;
0038
0038    FUNCTABLE:                                      ; FUNCTION DECISION TABLE
0038            FUNCTAB ,<-                             ; LEGAL FUNCTIONS
0038                    SETMODE,-                       ;  SET ATTENTION AST
0038                    WRITEOF,-                       ;  WRITE END-OF-FILE
0038                    READLBLK,WRITELBLK,-            ;  READ/WRITE LOGICAL BLOCKS
0038                    READVBLK,WRITEVBLK,-            ;  READ/WRITE VIRTUAL BLOCKS
0038                    READPBLK,WRITEPBLK>             ;  READ/WRITE PHYSICAL BLOCKS
0040            FUNCTAB ,<-                             ; BUFFERED I/O FUNCTIONS
0040                    READLBLK,WRITELBLK,-            ;  READ/WRITE LOGICAL BLOCKS
0040                    READVBLK,WRITEVBLK,-            ;  READ/WRITE VIRTUAL BLOCKS
0040                    READPBLK,WRITEPBLK>             ;  READ/WRITE PHYSICAL BLOCKS
0048            FUNCTAB FDTREAD,-                       ; READ FDT ACTION ROUTINE
0048                     <READLBLK,READPBLK,READVBLK>
0054            FUNCTAB FDTWRITE,-                      ; WRITE FDT ACTION ROUTINE
0054                    <WRITELBLK,WRITEPBLK,WRITEVBLK>
0060            FUNCTAB FDTSET,<SETMODE>                ; SET ATTENTION AST FDT ROUTINE
006C            FUNCTAB FDTEOF,<WRITEOF>                ; WRITE END-OF-FILE FDT ROUTINE
```

```
                         0078        .SBTTL  CANCELIO - CANCEL I/O ON MAILBOX UNIT
                         0078  ;++
                         0078  ; CANCELIO - CANCEL I/O ON MAILBOX UNIT
                         0078  ;
                         0078  ; FUNCTIONAL DESCRIPTION:
                         0078  ;
                         0078  ; THIS ROUTINE IS ENTERED TO CANCEL ALL OUTSTANDING I/O FOR A PARTICULAR
                         0078  ; PROCESS AND CHANNEL ON A MAILBOX UNIT.
                         0078  ;
                         0078  ;       o  IF THE UNIT IS BUSY, AND THE CURRENT READ PACKET BELONGS TO
                         0078  ;          THE CANCELLING PROCESS, AND IS FROM THE CANCELLING CHANNEL,
                         0078  ;          IT IS COMPLETED.
                         0078  ;
                         0078  ;       o  THE WRITE I/O QUEUE IS SCANNED.  IF A PACKET BELONGS TO THE
                         0078  ;          CANCELLING PROCESS, AND IS FROM THE CANCELLING CHANNEL, IT IS
                         0078  ;          COMPLETED.
                         0078  ;
                         0078  ;       o  THE READ AND WRITE ATTENTION AST LISTS ARE SCANNED.  IF
                         0078  ;          AN AST BELONGS TO THE CANCELLING PROCESS AND IS FROM THE
                         0078  ;          CANCELLING CHANNEL, IT IS REMOVED AND DEALLOCATED.
                         0078  ;
                         0078  ;       o  IF THE REFERENCE COUNT OF THE MAILBOX UCB IS ZERO AND MARKED FOR
                         0078  ;          DELETE, THE PORT'S REFERENCE TO THE MAILBOX CONTROL BLOCK IS
                         0078  ;          REMOVED.  IF THAT WAS THE ONLY REFERENCE LEFT, DEALLOCATE ALL THE
                         0078  ;          REMAINING MESSAGE BLOCKS, AND DEALLOCATE THE MAILBOX CONTROL BLOCK.
                         0078  ;
                         0078  ; INPUTS:
                         0078  ;
                         0078  ;       R2 = NEGATIVE OF CHANNEL NUMBER
                         0078  ;       R3 = CURRENT PACKET ADDRESS
                         0078  ;       R4 = PCB OF CANCELLING PROCESS
                         0078  ;       R5 = UCB OF UNIT
                         0078  ;       R8 = CANCEL REASON CODE (CAN$C_CANCEL, CAN$C_DASSGN, or CAN$C_AMBXDGN)
                         0078  ;
                         0078  ;       IPL = IPL$_MAILBOX
                         0078  ;
                         0078  ; OUTPUTS:
                         0078  ;
                         0078  ;       R4,R5,R6,R7 ARE PRESERVED
                         0078  ;
                         0078  ;--
                         0078 CANCELIO:                                  ; CANCEL I/O ON MAILBOX UNIT
01 64 A5    04     E0    0078        BBS     #UCB$V_ONLINE,UCB$W_STS(R5),10$ ; IF ONLINE CONTINUE
                   05    007D        RSB
                         007E 10$:
      0CF0 8F      BB    007E        PUSHR   #^M<R4,R5,R6,R7,R10,R11> ; SAVE REGISTERS
        58  02     D1    0082        CMPL    #CAN$C_AMBXDGN, R8      ; Branch if this is an associated
            6A     13    0085        BEQL    45$                     ; mailbox last ref. deassign.
        56  52     D0    0087        MOVL    R2,R6                   ; COPY CHANNEL NUMBER
                         008A  ;
                         008A  ; CHECK CURRENT READ I/O REQUEST AND COMPLETE IF CANCELLED
                         008A  ;
16 64 A5    08     E1    008A        BBC     #UCB$V_BSY,UCB$W_STS(R5),20$ ; READ IN PROGRESS?
0C A3    60 A4     D1    008F        CMPL    PCB$L_PID(R4),IRP$L_PID(R3, ; IS IT FROM CANCELLING PROCESS?
            0F     12    0094        BNEQ    20$                     ; IF NEQ THEN NO
   28 A3    56     B1    0096        CMPW    R6,IRP$W_CHAN(R3)       ; CHANNEL MATCH?
            09     12    009A        BNEQ    20$                     ; IF NEQ THEN NO
        50   2C   7D  009C   375          MOVQ    #SS$_ABORT,R0           ; SET STATUS TO ABORT
   00000000'GF    16  009F   376          JSB     G^IOC$REQCOM            ; COMPLETE THE REQUEST
                      00A5   377 ;
                      00A5   378 ; CHECK WRITE I/O REQUESTS AND COMPLETE IF CANCELLED
                      00A5   379 ;
   52    00A0 C5  9E  00A5   380 20$:     MOVAB   UCB$L_MB_WIOQFL(R5),R2  ; GET ADDRESS OF WRITE I/O QUEUE
         50   52  D0  00AA   381          MOVL    R2,R0                   ; COPY LIST HEAD ADDRESS
         52   62  D0  00AD   382 30$:     MOVL    (R2),R2                 ; GET ADDRESS OF LIST ENTRY
         52   50  D1  00B0   383          CMPL    R0,R2                   ; END OF LIST?
              1C  13  00B3   384          BEQL    40$                     ; IF YES THEN DONE
           60 A4  D1  00B5   385          CMPL    PCB$L_PID(R4),-         ; REQUEST FROM CANCELLING PROCESS?
           0C A2      00B8   386                  IRP$L_PID(R2)           ;
              F1  12  00BA   387          BNEQ    30$                     ; IF NO THEN SEARCH MORE
      28 A2   56  B1  00BC   388          CMPW    R6,IRP$W_CHAN(R2)       ; CHANNEL MATCH?
              EB  12  00C0   389          BNEQ    30$                     ; IF NEQ THEN NO
         53   62  0F  00C2   390          REMQUE  (R2),R3                 ; REMOVE PACKET FROM QUEUE
      38 A3   2C  7D  00C5   391          MOVQ    #SS$_ABORT,IRP$L_IOST1(R3) ; SET STATUS TO ABORT
   00000000'GF    16  00C9   392          JSB     G^COM$POST              ; COMPLETE THE OPERATION
              D4  11  00CF   393          BRB     20$                     ; SEARCH LIST FROM THE START
                      00D1   394 ;
                      00D1   395 ; CHECK ATTENTION AST REQUESTS AND DELETE IF CANCELLED
                      00D1   396 ;
   57    0090 C5  9E  00D1   397 40$:     MOVAB   UCB$L_MB_WAST(R5),R7    ; GET ADDRESS OF WRITE AST'S
   00000000'GF    16  00D6   398          JSB     G^COM$FLUSHATTNS        ; FLUSH ATTENTION AST'S
   57    0094 C5  9E  00DC   399          MOVAB   UCB$L_MB_RAST(R5),R7    ; GET ADDRESS OF READ AST'S
   00000000'GF    16  00E1   400          JSB     G^COM$FLUSHATTNS        ; FLUSH THAT LIST
                      00E7   401 ;
                      00E7   402 ; CHECK IF MAILBOX CONTROL BLOCK SHOULD BE DEALLOCATED. IF SO, DEALLOCATE
                      00E7   403 ; ANY REMAINING MESSAGE BLOCKS AND MARK THE MAILBOX AS NO LONGER VALID.
                      00E7   404 ;
        58   01   D1  00E7   405          CMPL    #CAN$C_DASSGN, R8       ; Deassigning channel?
             5B   12  00EA   406          BNEQ    69$                     ; Branch if not channel deassign.
          5C A5   B5  00EC   407          TSTW    UCB$W_REFC(R5)          ; Is reference count zero?
             56   12  00EF   408          BNEQ    69$                     ; Branch if ref. count is not zero.
   51 68 A5  01   E1  00F1   409 45$:     BBC     #UCB$V_DELMBX, -        ; Branch if mailbox is not
                      00F6   410                  UCB$W_DEVSTS(R5), 69$   ; to be deleted.
   50    009C C5  D0  00F6   411          MOVL    UCB$L_MB_SHB(R5),R0     ; GET ADDRESS OF SHB
          0C A0   D7  00FB   412          DECL    SHB$L_REFCNT(R0)        ; DECREMENT SHARED MEMORY REFERENCE COUNT
   51     04 A0   D0  00FE   413          MOVL    SHB$L_DATAPAGE(R0),R1   ; GET DATAPAGE ADDRESS
                      0102   414          LOCK    #SHD$V_MBXLCK,SHD$B_FLAGS(R1) ; LOCK MAILBOX TABLE
   52    0098 C5  D0  0120   415          MOVL    UCB$L_MB_MBX(R5),R2     ; GET MAILBOX CONTROL BLOCK ADDRESS
00 0C A2  00A8 C5 E7  0125   416          BBCCI   UCB$L_MB_PORT(R5),MBX$W_REF(R2),50$ ; CLEAR PORT'S REFERENCE
          0C A2   B5  012C   417 50$:     TSTW    MBX$W_REF(R2)           ; ANY OTHER REFERENCES?
             18   12  012F   418          BNEQ    70$                     ; IF NEQ YES
      08 A2  02   8A  0131   419          BICB    #MBX$M_VALID,MBX$B_FLAGS(R2) ; CLEAR VALID FLAG
   50     09 A2   9A  0135   420          MOVZBL  MBX$B_CREATPORT(R2),R0  ; GET CREATOR PORT NUMBER
        5C A140   B6  0139   421          INCW    SHD$W_MBXQUOTA(R1)[R0]  ; RESTORE CREATOR'S QUOTA
        5B   62   5E  013D   422 60$:     REMQHI  MBX$Q_MSG(R2),R11       ; GET ADDRESS OF NEXT MESSAGE BLOCK
             07   1D  0140   423          BVS     70$                     ; IF VS NO MORE BLOCKS
           0330   30  0142   424          BSBW    DALLOC_BLOCKS           ; DEALLOCATE THE MESSAGE BLOCK(S)
             F6   11  0145   425          BRB     60$
             3F   11  0147   426 69$:     BRB     900$                    ; Exit branch assist.
                      0149   427
                      0149   428 70$:     UNLOCK  #SHD$V_MBXLCK,SHD$B_FLAGS(R1) ; UNLOCK MAILBOX TABLE
   57     24 A5   D0  0152   429          MOVL    UCB$L_CRB(R5),R7        ; CLEAR OUT THE POINTER IN THE
   56     54 A5   3C  0156   430          MOVZWL  UCB$W_UNIT(R5),R6       ; IDB TO THIS UCB, PREVENTING A
   57     2C A7   D0  015A   431          MOVL    <CRB$L_INTD+VEC$L_IDB>(R7),R7 ; RACE CONDITION BY ANOTHER
```

```
                    18 A746  D4  015E   432           CLRL    IDB$L_UCBLST(R7)[R6]    ; PORT QUEUING A PRQ FOR THIS MAILBOX.
                                 0162   433
                                 0162   434           SETIPL  #IPL$_ASTDEL            ; Lower IPL
                      74 A5  D5  0165   435           TSTL    UCB$L_LOGADR(R5)        ; Test address of logical name entry.
                         16  13  0168   436           BEQL    120$                    ; Branch if none.
               00000000'GF   16  016A   437           JSB     G^LNM$LOCKW             ; Lock name table for write.
                 51   74 A5  D0  0170   438           MOVL    UCB$L_LOGADR(R5), R1    ; Get address of logical name entry.
               00000000'GF   16  0174   439           JSB     G^LNM$DELETE_LNMB       ; Delete logical name block.
               00000000'GF   16  017A   440           JSB     G^LNM$UNLOCK            ; Unlock name table.
      64 A5    00010000 8F   C8  0180   441 120$:     BISL    #UCB$M_DELETEUCB, -     ; Mark UCB for deletion, DASSGN
                                 0188   442                   UCB$L_STS(R5)           ; will do the rest including crediting
                                 0188   443                                           ; quotas for temp. mailboxes.
                                 0188   444
                    0CF0 8F  BA  0188   445 900$:     POPR    #^M<R4,R5,R6,R7,R10,R11> ; Restore registers.
                             05  018C   446           RSB                             ;
```

```
                                018D          .SBTTL  CHECKIO - CHECK READ AND WRITE ACCESS AND PARAMETERS
                                018D     ;++
                                018D     ; READCHECKIO - CHECK READ ACCESS AND PARAMETERS
                                018D     ; WRITECHECKIO - CHECK WRITE ACCESS AND PARAMETERS
                                018D     ;
                                018D     ; FUNCTIONAL DESCRIPTION:
                                018D     ;
                                018D     ; THIS ROUTINE IS USED BY THE READ AND WRITE FDT ROUTINES TO VALIDATE
                                018D     ; THE I/O REQUEST. THE CHECKS ARE:
                                018D     ;
                                018D     ;        o  ACCESS TO UNIT BY UIC
                                018D     ;
                                018D     ;        o  MESSAGE SIZE WITHIN MAX MESSAGE SIZE
                                018D     ;
                                018D     ;        o  BUFFER ACCESSABLE
                                018D     ;
                                018D     ;
                                018D     ; ZERO LENGTH TRANSFERS AND ACCESS VIOLATIONS CAUSE COMPLETIONS HERE.
                                018D     ;
                                018D     ; INPUTS:
                                018D     ;
                                018D     ;        R0-R2 = SCRATCH
                                018D     ;        R3 = PACKET ADDRESS
                                018D     ;        R4 = PCB ADDRESS
                                018D     ;        R5 = UCB ADDRESS
                                018D     ;        R6 = CCB ADDRESS
                                018D     ;        R7 = FUNCTION CODE
                                018D     ;        R8   ADDRESS OF FDT TABLE ENTRY FOR THIS ROUTINE
                                018D     ;        R9-R11 = SCRATCH
                                018D     ;        AP = ADDRESS OF THE FIRST QIO PARAMETER
                                018D     ;
                                018D     ; OUTPUTS:
                                018D     ;
                                018D     ;        R3 = PACKET ADDRESS
                                018D     ;        R4 = PCB ADDRESS
                                018D     ;        R5 = UCB ADDRESS
                                018D     ;
                                018D     ;        IRP$L_MEDIA(R3) = BUFFER ADDRESS.
                                018D     ;        IRP$W_BCNT(R3) = BUFFER SIZE.
                                018D     ;
                                018D     ;--
                                018D     READCHECKIO:                                    ; CHECK FOR READ ACCESS
      00000000'GF         9F    018D             PUSHAB  G^EXE$READCHK                   ; SET UP FOR BUFFER READ I/O ACCESS
59    00000000'GF         9E    0193             MOVAB   G^EXE$CHKRDACCES,R9             ; SET UP FOR UNIT READ ACCESS
          5A    02        D0    019A             MOVL    #CCB$V_RDCHKDON,R10             ;
                10        11    019D             BRB     CHECKIO                         ;
                                019F     WRITECHECKIO:                                   ; CHECK FOR WRITE ACCESS
59    00000000'GF         9E    019F             MOVAB   G^EXE$CHKWRTACCES,R9            ; SET UP FOR BUFFER WRITE I/O ACCESS
          5A    03        D0    01A6             MOVL    #CCB$V_WRTCHKDON,R10            ;
      00000000'GF         9F    01A9             PUSHAB  G^EXE$WRITECHK                  ; SET UP FOR UNIT WRITE ACCESS
                                01AF     CHECKIO:                                        ; CHECK I/O ACCESS AND PARAMETERS
             30 A3        B4    01AF             CLRW    IRP$W_BOFF(R3)                  ; RESET QUOTA
    0A 08 A6    5A        E0    01B2             BBS     R10,CCB$B_STS(R6),10$           ; SKIP CHECK IF ALREADY DONE
                                01B7                                                     ; R4 - PCB ADDRESS
                                01B7                                                     ; R5 - UCB ADDRESS
                69        16    01B7             JSB     (R9)                            ; CHECK READ/WRITE ACCESS
             28 50        E9    01B9             BLBC    R0,ERROR                        ; BR IF ACCESS FAILURE
```

```
00 08 A6    5A   E2  01BC                BBSS     R10,CCB$B_STS(R6),10$    ; MARK PROT CHECK DONE
   51    04 AC   3C  01C1      10$:      MOVZWL   P2(AP),R1                ; GET BUFFER SIZE
            0E   13  01C5                BEQL     ZEROLENGTH               ; IF EQL THEN COMPLETE HERE
   42 A5    51   B1  01C7                CMPW     R1,UCB$W_DEVBUFSIZ(R5)   ; MESSAGE SIZE IN RANGE?
            12   1A  01CB                BGTRU    TOOSMALL                 ; IF GTRU THEN NO
      50    6C   D0  01CD                MOVL     P1(AP),R0                ; GET BUFFER ADDRESS
   38 A3    50   D0  01D0                MOVL     R0,IRP$L_MEDIA(R3)       ; SAVE BUFFER ADDRESS
                 05  01D4                RSB                               ; RETURN AND CHECK BUFFER ACCESS
                     01D5
                     01D5      ;+
                     01D5      ; PROCESS ZERO LENGTH TRANSFERS
                     01D5      ;
                     01D5      ;  For a zero byte transfer, a dummy buffer (whose address is the current top
                     01D5      ;  of the current stack) of zero bytes length is constructed.  The normal
                     01D5      ;  access checks must be bypassed for this buffer because the previous caller
                     01D5      ;  may not have access to the current stack.
                     01D5      ;-
                     01D5
                     01D5      ZEROLENGTH:
            8E   D5  01D5                TSTL     (SP)+                    ; Pop checking rout. addr. from stack.
         32 A3   D4  01D7                CLRL     IRP$L_BCNT(R3)           ; Set zero byte count.
   38 A3    6E   9E  01DA                MOVAB    (SP), IRP$L_MEDIA(R3)    ; Set top-of-stack buffer address.
                 05  01DE                RSB                               ; Return directly to routines caller.
                     01DF
                     01DF      TOOSMALL:                                   ; MAILBOX TOO SMALL FOR MESSAGE
  50     019C 8F   3C  01DF              MOVZWL   #SS$_MBTOOSML,R0         ; SET BOX TOO SMALL
                     01E4      ERROR:                                      ; ERROR - ABORT THE I/O REQUEST
   00000000'GF   17  01E4                JMP      G^EXE$ABORTIO            ; ABORT THE I/O
```

```
                          01EA                .SBTTL  FDTREAD - READ FUNCTION DECISION ROUTINE
                          01EA        ;++
                          01EA        ; FDTREAD - FUNCTION DECISION ROUTINE FOR READ OPFRATIONS
                          01EA        ;
                          01EA        ; FUNCTIONAL DESCRIPTION:
                          01EA        ;
                          01EA        ; THE REQUEST IS FIRST CHECKED FOR READ ACCESS TO THE MAILBOX AND
                          01EA        ; WRITE ACCESS TO THE SPECIFIED BUFFER.  THE PACKET IS THEN QUEUED
                          01EA        ; TO THE UNIT'S I/O QUEUE (UCB$L_IOQFL) FOR PROCESSING WHEN THE UNIT
                          01EA        ; IS NOT BUSY, IN OTHER WORDS, WHEN ANY PREVIOUS READ REQUESTS ON THIS
                          01EA        ; PROCESSOR HAVE BEEN SATISFIED.
                          01EA        ;
                          01EA        ; IF THE FUNCTION MODIFIER IO$M_NOW IS SPECIFIED, THE MAILBOX IS CHECKED
                          01EA        ; TO SEE IF ANY MESSAGES ARE WAITING.  IF THERE AREN'T MESSAGES, THE
                          01EA        ; REQUEST IS COMPLETED WITH FAILURE, OTHERWISE IT IS QUEUED AS FOR A
                          01EA        ; NORMAL READ REQUEST.
                          01EA        ;
                          01EA        ; INPUTS:
                          01EA        ;
                          01EA        ;       R0-R2 = SCRATCH
                          01EA        ;       R3 = I/O PACKET ADDRESS
                          01EA        ;       R4 = CURRENT PCB ADDRESS
                          01EA        ;       R5 = UCB ADDRESS
                          01EA        ;       R6 = CCB ADDRESS
                          01EA        ;       R7 = FUNCTION CODE
                          01EA        ;       R8 = ADDRESS OF FDT TABLE ENTRY FOR THIS ROUTINE
                          01EA        ;       R9-R11 = SCRATCH
                          01EA        ;       AP = FIRST QIO PARAMETER ADDRESS
                          01EA        ;
                          01EA        ; OUTPUTS:
                          01EA        ;
                          01EA        ;       THE PACKET IS QUEUED VIA "EXE$QIODRVPKT" OR
                          01EA        ;       THE REQUEST IS COMPLETED WITH AN ERROR VIA "EXE$ABORTIO" OR
                          01EA        ;       "EXE$FINISHIOC"
                          01EA        ;
                          01EA        ; STATUS CODES:
                          01EA        ;
                          01EA        ;       SS$_NOPRIV - USER DOES NOT HAVE PRIVILEGE TO READ MAILBOX
                          01EA        ;       SS$_ACCVIO - BUFFER ACCESS VIOLATION
                          01EA        ;       SS$_MBTOOSML - REQUEST EXCEEDS THE MAXIMUM MESSAGE SIZE
                          01EA        ;       SS$_ENDOFFILE - NO MESSAGE AVAILABLE AND IO$M_NOW SPECIFIED
                          01EA        ;       SS$_NORMAL - NORMAL STATUS
                          01EA        ;--
                          01EA        FDTREAD:                                   ;
               A1   10    01EA                BSBB    READCHECKIO                ; VALIDATE THE REQUEST
      2A A3   0400 8F   A8    01EC                BISW    #IRP$M_MBXIO,IRP$W_STS(R3) ; SET MAILBOX READ
                          01F2        ;
                          01F2        ; UPDATE MEASUREMENT COUNTER IF ENABLED
                          01F2        ;
                00000002  01F2                .IF NE CA$_MEASURE
           00000000'GF  D6  01F2                INCL    G^PMS$GL_MBREADS           ; COUNT MAILBOX READS
                          01F8                .ENDC
                          01F8        ;
                          01F8        ; ALLOCATE TWO I/O PACKET EXTENSIONS TO USE AS FORK BLOCKS IF WE'RE
                          01F8        ; FORCED TO WAIT WHEN: 1) NOTIFYING OTHER PROCESSOR OF WAITING READER
                          01F8        ; 2) NOTIFYING OTHER PROCESSOR WHEN MESSAGE IS READ
                          01F8        ;
```

```
             54 A3   D4  01F8            CLRL    IRP$L_EXTEND(R3)         ; SET NO EXTENSION YET
              04DF   30  01FB            BSBW    ALLOC_IRPE               ; ALLOCATE AN EXTENSION
              04DC   30  01FE            BSBW    ALLOC_IRPE               ; ALLOCATE ANOTHER EXTENSION
                         0201      ;
                         0201      ; IF IO$M_NOW IS SPECIFIED, CHECK IF THERE ARE ANY MESSAGES WAITING.
                         0201      ; IF THERE ARE OR IO$M_NOW IS NOT SPECIFIED, QUEUE THE REQUEST.
                         0201      ; OTHERWISE COMPLETE THE REQUEST WITH FAILURE.
                         0201      ;
     17 20 A3    06  E1  0201            BBC     #IO$V_NOW,IRP$W_FUNC(R3),10$; BR IF NOT 'NOW'
                         0206            SETIPL  #IPL$_MAILBOX            ; RAISE IPL TO MAKE SURE NO
                         0209                                             ; OTHER REQUEST SNEAKS IN QUEUE
      52    0098 C5  D0  0209            MOVL    UCB$L_MB_MBX(R5),R2      ; GET MAILBOX ADDRESS
                62   D5  020E            TSTL    MBX$Q_MSG(R2)            ; ANY MESSAGES IN MAILBOX?
                0B   12  0210            BNEQ    10$                      ; IF NEQ THEN YES
      50    0870 8F  3C  0212            MOVZWL  #SS$_ENDOFFILE,R0        ; SET NO TRANSFER AND STATUS
       00000000'GF   17  0217            JMP     G^EXE$FINISHIOC          ; COMPLETE THE I/O
                         021D      10$:
       00000000'GF   17  021D            JMP     G^EXE$QIODRVPKT          ; QUEUE PACKET TO STARTIO
```

```
                     0223          .SBTTL  FDTSET - HANDLE SET MODE FUNCTIONS
                     0223    ;++
                     0223    ; FDTSET - HANDLE SET MODE FUNCTIONS
                     0223    ;
                     0223    ; FUNCTIONAL DESCRIPTION:
                     0223    ;
                     0223    ; THIS ROUTINE IMPLEMENTS THE IO$_SETMODE FUNCTIONS.
                     0223    ; THE DIFFERENT FUNCTIONS ARE SELECTED BY A FUNCTION CODE MODIFIER.
                     0223    ; THE FUNCTIONS ARE:
                     0223    ;
                     0223    ;       IO$M_SETPROT    - SET VOLUME PROTECTION
                     0223    ;       IO$M_READATTN   - SET READ ATTENTION AST
                     0223    ;       IO$M_WRTATTN    - SET WRITE ATTENTION AST
                     0223    ;
                     0223    ; INPUTS:
                     0223    ;
                     0223    ;       R0-R2 = SCRATCH
                     0223    ;       R3 = I/O PACKET ADDRESS
                     0223    ;       R4 = CURRENT PCB
                     0223    ;       R5 = UCB ADDRESS FOR MAILBOX UNIT
                     0223    ;       AP = ADDRESS OF QIO PARAMETER BLOCK
                     0223    ;
                     0223    ; OUTPUTS:
                     0223    ;
                     0223    ;       NONE.
                     0223    ;
                     0223    ; STATUS RETURNS:
                     0223    ;
                     0223    ;       SS$_NORMAL - SUCCESSUFL COMPLETION
                     0223    ;       SS$_INSFMEM - INSUFICIENT MEMORY TO ALLOCATE AST BLOCK
                     0223    ;       SS$_EXQUOTA - AST QUOTA EXCEEDED
                     0223    ;       SS$_ILLIOFUNC - ILLEGAL SET MODE FUNCTION
                     0223    ;       SS$_NOPRIV - THE USER CANNOT SET THE VOLUME PROTECTION
                     0223    ;--
                     0223   FDTSET:                                 ; SET RECEIVE AST FUNCTION
                     0223    ;
                     0223    ; SEE IF THIS IS A SETPROT FUNCTION.
                     0223    ;
           09   E0   0223          BBS     #IO$V_SETPROT,-         ; BRANCH IF SETPROT FUNCTION
     65 20 A3        0225                  IRP$W_FUNC(R3),50$      ;
                     0228    ;
                     0228    ; SEE IF USER CAN READ THIS MAILBOX
                     0228    ;
                     0228                                           ; R4 - PCB ADDRESS
                     0228                                           ; R5 - UCB ADDRESS
  00000000'GF   16   0228          JSB     G^EXE$CHKRDACCES        ; CHECK READ ACCESS TO UNIT
        B3 50   E9   022E          BLBC    R0,ERROR                ; IF LOW CLEAR THEN ERROR
                     0231    ;
                     0231    ; CREATE AN AST CONTROL BLOCK AND ENTER IT IN APPROPRIATE ATTENTION LIST
                     0231    ;
   57   0090 C5   DE   0231          MOVAL   UCB$L_MB_WAST(R5),R7    ; ASSUME WRITE AST LIST ADDR
 05 20 A3   07   E1   0236          BBC     #IO$V_READATTN,IRP$W_FUNC(R3),10$; BR IF NOT READ AST
   57   0094 C5   DE   023B          MOVAL   UCB$L_MB_RAST(R5),R7    ; GET ADDR OF READ AST LIST
           54   DD   0240   10$:   PUSHL   R4                      ; SAVE PCB ADDRESS
           57   DD   0242          PUSHL   R7                      ; SAVE AST LIST HEAD ADDRESS
  00000000'GF   16   0244          JSB     G^COM$SETATTNAST        ; ENTER AN AST REQUEST IN LIST
        54 8ED0      024A          POPL    R4                      ; GET AST LIST HEAD ADDRESS
```

```
                        024D    ;
                        024D    ; SET WAITING ATTENTION AST FLAG IN MAILBOX AND CHECK IF ATTENTION
                        024D    ; CONDITION ALREADY EXISTS.  IF IT DOES, CLEAR THE WAITING ATTENTION
                        024D    ; AST FLAG IN MAILBOX AND DELIVER THE AST.  FLAG MUST BE SET BEFORE
                        024D    ; CHECK IN CASE ANOTHER PROCESSOR CHECKED FOR WAITING ATTENTIONS AFTER
                        024D    ; WE DID, BUT BEFORE WE COULD SET OUR WAITING FLAG.
                        024D    ;
  52   0098 C5     D0   024D            MOVL    UCB$L_MB_MBX(R5),R2         ; GET ADDR OF MAILBOX
 14 20 A3    07    E0   0252            BBS     #IO$V_READATTN,IRP$W_FUNC(R3),20$ ; BR IF READ AST
                        0257                                                ; WRITE ATTENTION AST REQUEST
                        0257            SET_PORTFLAG MBX$W_WRITAST(R2)      ; SET FLAG TO NOTIFY IF WRITE OCCURS
             62    D5   025E            TSTL    MBX$Q_MSG(R2)               ; ANY MESSAGES WRITTEN?
             22    13   0260            BEQL    40$                         ; IF EQL THEN NO - JUST COMPLETE
                        0262            CLR_PORTFLAG MBX$W_WRITAST(R2)      ; CLEAR NOTIFY FLAG
             13    11   0269            BRB     30$                         ; DELIVER THE AST
                        026B    20$:                                        ; READ ATTENTION AST REQUEST
                        026B            SET_PORTFLAG MBX$W_READAST(R2)      ; SET FLAG TO NOTIFY IF READ OCCURS
          0E A2    B5   0272            TSTW    MBX$W_READER(R2)            ; ANY READERS WAITING?
             0D    13   0275            BEQL    40$                         ; IF EQL THEN NO - JUST COMPLETE
                        0277            CLR_PORTFLAG MBX$W_READAST(R2)      ; CLEAR NOTIFY FLAG
                        027E    30$:
    00000000'GF    16   027E            JSB     G^COM$DELATTNAST            ; DELIVER THE AST IMMEDIATELY
                        0284    ;
                        0284    ; COMPLETE THE SETMODE REQUEST
                        0284    ;
             54 8ED0    0284    40$:    POPL    R4                          ; RESTORE PCB ADDRESS
    00000000'GF    17   0287    45$:    JMP     G^EXE$FINISHIOC             ; COMPLETE THE I/O
                        028D    ;
                        028D    ; HANDLE THE SETPROT FUNCTION
                        028D    ;
        50   24    3C   028D    50$:    MOVZWL  #SS$_NOPRIV,R0              ; ASSUME NO PRIVILEGE
     51   1C A5    D0   0290            MOVL    UCB$L_ORB(R5),R1            ; GET ORB ADDRESS
        00BC C4    D1   0294            CMPL    PCB$L_UIC(R4),-             ; IS THIS THE VOLUME OWNER?
             61         0298                    ORB$L_OWNER(R1) ;
             1E    12   0299            BNEQ    52$                         ; BRANCH IF NOT
     50   04 AC    B0   029B    51$:    MOVW    P2(AP),R0                   ; GET THE PROTECTION MASK
  52   0098 C5     D0   029F            MOVL    UCB$L_MB_MBX(R5),R2         ; GET MBX ADDRESS
                        02A4            SETIPL  UCB$B_DIPL(R5)              ; BLOCK DEVICE INTERRUPTS
     0B A1   01    88   02A8            BISB2   #ORB$M_PROT_16,ORB$B_FLAGS(R1)  ; PROTECTION WORD NOT VECTOR
     18 A1   50    B0   02AC            MOVW    R0,ORB$W_PROT(R1)           ; SET THE NEW PROTECTION MASK
     1A A2   50    B0   02B0            MOVW    R0,MBX$W_PROT(R2)           ; SET SECOND COPY OF PROTECTION MASK
        50   01    3C   02B4            MOVZWL  #SS$_NORMAL,R0              ; SET SUCCESS STATUS
             CE    11   02B7            BRB     45$                         ; COMPLETE THE I/O
             1D    E0   02B9    52$:    BBS     #PRV$V_BYPASS,-             ; BRANCH IF USER HAS BYPASS
       DD 6C B4         02BB                    @PCB$L_PHD(R4),51$          ;
           FF23    31   02BE            BRW     ERROR                       ; ABORT THE I/O
```

```
                    02C1              .SBTTL  FDTEOF - WRITE EOF MESSAGE TO MAILBOX
                    02C1       ;++
                    02C1       ; FDTEOF - WRITE EOF MESSAGE TO THE MAILBOX
                    02C1       ;
                    02C1       ; FUNCTIONAL DESCRIPTION:
                    02C1       ;
                    02C1       ; THIS IS THE FDT ROUTINE FOR IO$WRITEOF. THE ACTION IS TO BUILD A
                    02C1       ; ZERO LENGTH MESSAGE AND TO INSERT IT IN THE MAILBOX.
                    02C1       ; THIS MESSAGE, WHEN READ RESULTS IN AN SS$_ENDOFILE STATUS RETURN.
                    02C1       ;
                    02C1       ; INPUTS:
                    02C1       ;
                    02C1       ;       R0-R2 = SCRATCH
                    02C1       ;       R3 = I/O PACKET ADDRESS
                    02C1       ;       R4 = CURRENT PCB ADDRESS
                    02C1       ;       R5 = MAILBOX UCB ADDRESS
                    02C1       ;       R6 = CCB ADDRESS
                    02C1       ;       R7 = FUNCTION CODE
                    02C1       ;       R8 = ADDRESS OF FDT TABLE ENTRY FOR THIS ROUTINE
                    02C1       ;       R9-R11 = SCRATCH
                    02C1       ;       AP = ADDRESS OF USER ARGUMENT BLOCK AT "P1"
                    02C1       ;
                    02C1       ; OUTPUTS:
                    02C1       ;
                    02C1       ;       IRP$L_MEDIA(R3) = FAKE BUFFER ADDRESS.
                    02C1       ;       IRP$W_BCNT(R3) = ZERO BUFFER SIZE.
                    02C1       ;
                    02C1       ;       THE I/O IS COMPLETED IN THE WRITE FDT LOGIC. ( SEE BELOW )
                    02C1       ;--
                    02C1       FDTEOF:                                         ;
       30 A3    D4  02C1              CLRL    IRP$W_BOFF(R3)                  ; SET NO TRANSFER AND NO QUOTA
                    02C4                                                      ; R4 - PCB ADDRESS
                    02C4                                                      ; R5 - UCB ADDRESS
  00000000'GF   16  02C4              JSB     G^EXE$CHKWRTACCES               ; CHECK WRITE ACCESS TO UNIT
        09 50   E9  02CA              BLBC    R0,10$                          ; IF ERROR THEN BRANCH
        32 A3   D4  02CD              CLRL    IRP$W_BCNT(R3)                  ; SET ZERO LENGTH BUFFER
   38 A3   6E   9E  02D0              MOVAB   (SP),IRP$L_MEDIA(R3)            ; SET FAKE ADDR OF BUFFER
           06   11  02D4              BRB     WRITE                           ; WRITE THE MESSAGE
         FF08   31  02D6       10$:   BRW     ERROR                           ; CONTINUE
```

```
                    02D9            .SBTTL  FDTWRITE - WRITE OPERATION FDT ROUTINE
                    02D9     ;++
                    02D9     ; FDTWRITE -- FUNCTION DECISION ACTION ROUTINE FOR WRITE FUNCTIONS
                    02D9     ;
                    02D9     ; FUNCTIONAL DESCRIPTION:
                    02D9     ;
                    02D9     ; THE USER REQUEST IS VALIDATED FOR PRIVILEGE, SIZE, ACCESS AND AVAILABLE
                    02D9     ; SPACE. IF VALID, A BUFFERED I/O BLOCK IS ALLOCATED (IMPLIED RESOURCE WAIT).
                    02D9     ; THE BLOCK IS SET UP AND QUEUED TO THE UNIT MESSAGE LIST. IF THE UNIT
                    02D9     ; IS BUSY, THE OUTSTANDING READ OPERATION IS COMPLETED DIRECTLY.
                    02D9     ; IN THE CASE OF "WRITENOW" FUNCTIONS THE I/O IS COMPLETED BEFORE THE
                    02D9     ; MESSAGE IS QUEUED. OTHERWISE THE READ COMPLETE ROUTINE COMPLETES
                    02D9     ; THE MESSAGE ASSOCIATED WRITE.
                    02D9     ;
                    02D9     ; INPUTS:
                    02D9     ;
                    02D9     ;       R3 = I/O PACKET ADDRESS
                    02D9     ;       R4 = CURRENT PCB ADDRESS
                    02D9     ;       R5 = UCB ADDRESS
                    02D9     ;       R6 = CCB ADDRESS
                    02D9     ;       R7 = FUNCTION CODE
                    02D9     ;       R8 = ADDRESS OF FDT TABLE ENTRY FOR THIS ROUTINE
                    02D9     ;       R9-R11 = SCRATCH
                    02D9     ;       AP = ADDRESS OF USER ARGUMENT BLOCK AT "P1"
                    02D9     ;
                    02D9     ; OUTPUTS:
                    02D9     ;
                    02D9     ;       THE I/O IS COMPLETED IN ERROR, THE I/O IS RESTARTED BECAUSE OF
                    02D9     ;       RESOURCE WAIT, OR THE I/O IS COMPETED NORMALLY.
                    02D9     ;
                    02D9     ; STATUS RETURNS:
                    02D9     ;
                    02D9     ;       SS$_MBTOOSML - MESSAGE IS TOO BIG
                    02D9     ;       SS$_ACCVIO - BUFFER ACCESS VIOLATION ( "EXE$WRITECHK" )
                    02D9     ;       SS$_MBFULL - MAILBOX IS FULL
                    02D9     ;       SS$_NOPRIV - USER DOES NOT HAVE WRITE PRIVILEGE
                    02D9     ;       SS$_NORMAL - SUCCESSFUL STATUS
                    02D9     ;       SS$_INSFMEM - NO MEMORY FOR BUFFER ALLOCATION
                    02D9     ;--
                    02D9     FDTWRITE:                                      ;
        FEC3   30   02D9            BSBW    WRITECHECKIO                   ; CHECK OPERATION PARAMETERS
                    02DC     WRITE:
  59   32 A3   3C   02DC            MOVZWL  IRP$W_BCNT(R3),R9              ; R9 = SIZE OF USER DATA
  5A   38 A3   D0   02E0            MOVL    IRP$L_MEDIA(R3),R10            ; R10 = ADDRESS OF USER DATA
          5B   D4   02E4            CLRL    R11                            ; R11 = ADDRESS OF FIRST BLOCK
                    02E6     ;
                    02E6     ; ALLOCATE AN I/O PACKET EXTENSION TO USE AS A FORK BLOCK IF WE'RE FORCED
                    02E6     ; TO WAIT WHEN NOTIFYING OTHER PROCESSORS OF THE AVAILABILITY OF A MESSAGE.
                    02E6     ;
       54 A3   D4   02E6            CLRL    IRP$L_EXTEND(R3)               ; SET NO EXTENSION YET
        03F1   30   02E9            BSBW    ALLOC_IRPE                     ; ALLOCATE A EXTENSION
                    02EC     ;
                    02EC     ; ALLOCATE SHARED MEMORY POOL BLOCK
                    02EC     ;
                    02EC     ALLOC_BLOCK:                                   ; ALLOCATE SHARED MEMORY BLOCK
 52  009C C5   D0   02EC            MOVL    UCB$L_MB_SHB(R5),R2            ; GET ADDR OF SHB
 51    04 A2   D0   02F1            MOVL    SHB$L_DATAPAGE(R2),R1          ; GET ADDR OF DATAPAGE
```

```
          50    03     D0  02F5          MOVL    #RSN$_NPDYNMEM,R0          ; GET RESOURCE NUMBER
                           02F8          DSBINT  #IPL$_SYNCH                ; PREVENT SCHEDULING
  7E   00A8 C140     3E    02FE          MOVAW   SHD$W_RESWAIT(R1)[R0],-(SP) ; SAVE ADDR OF WAIT MASK
                           0304          SET_PORTFLAG @(SP)                 ; ASSUME ALLOCATION FAILURE
                           030B                                             ; (AVOIDS MISSING NOTIFICATION)
     00000000'GF     16    030B          JSB     G^EXE$ALOSHARED            ; ALLOCATE A SHARED MEMORY BLOCK
           03 50     E8    0311          BLBS    R0,10$                     ; IF LBS SUCCESS
            011F     31    0314          BRW     ALLOC_FAIL                 ; ELSE - FAILURE
                           0317  10$:    CLR_PORTFLAG @(SP)+                ; CLEAR FLAG SINCE BLOCK OBTAINED
                           031D          ENBINT                             ; ALLOW SCHEDULING
              5B     D5    0320          TSTL    R11                        ; IS THIS THE FIRST BLOCK?
              09     12    0322          BNEQ    20$                        ; IF NEQ NO
         5B   52     D0    0324          MOVL    R2,R11                     ; SAVE ADDRESS OF FIRST BLOCK
      0E A2   59     B0    0327          MOVW    R9,MSG_W_MSGLENGTH(R2)     ; SET SIZE OF MESSAGE DATA
              05     11    032B          BRB     SETUP_BLOCK                ;
                           032D  20$:
 10 A8   52   5B     C3    032D          SUBL3   R11,R2,MSG_L_CHAINLINK(R8) ; SET OFFSET FROM FIRST BLOCK TO NEW BLOC
                           0332  ;
                           0332  ; SET UP MESSAGE BLOCK
                           0332  ;
                           0332  SETUP_BLOCK:                               ; SET UP MESSAGE BLOCK
  0A A2    80 8F     90    0332          MOVB    #DYN$C_SHRBUFIO,MSG_B_TYPE(R2) ; SET TYPE OF BLOCK
                           0337
 0B A2   00A8 C5     90    0337          MOVB    UCB$L_MB_PORT(R5),MSG_B_PORT(R2) ; SET WRITER'S PORT NUMBER
                           033D          ASSUME  PRQ$C_MINLENGTH GE MSG_B_MESSAGE+4 ;NEED ROOM FOR SOME DATA
         51   1D     C2    033D          SUBL    #MSG_B_MESSAGE,R1          ; COMPUTE SIZE FOR DATA IN BLOCK
      0C A2   59     B0    0340          MOVW    R9,MSG_W_LENGTH(R2)        ; ASSUME ALL DATA FITS IN BLOCK
         51   59     D1    0344          CMPL    R9,R1                      ; IS DATA TOO BIG?
              04     15    0347          BLEQ    10$                        ; IF LEQ NO - DATA FITS
      0C A2   51     B0    0349          MOVW    R1,MSG_W_LENGTH(R2)        ; ELSE - SET LOWER SIZE
                           034D  10$:
           10 A2     D4    034D          CLRL    MSG_L_CHAINLINK(R2)        ; CLEAR CHAIN LINK POINTER
           50 A3     D0    0350          MOVL    IRP$L_SEQNUM(R3),-         ; SET WRITER'S IRP SEQUENCE NUMBER
           14 A2           0353                  MSG_L_IRPSEQ(R2)           ;
   03 20 A3   06     E1    0355          BBC     #IO$V_NOW,IRP$W_FUNC(R3),15$ ; BR IF NOT "NOW"
           14 A2     D4    035A          CLRL    MSG_L_IRPSEQ(R2)           ; CLEAR WRITER'S IRP SEQUENCE NUMBER
                           035D                                             ; INDICATES NO NEED TO NOTIFY WRITER
           64 A4     D0    035D  15$:    MOVL    PCB$L_EPID(R4),-           ; INSERT EXTENDED PID OF WRITER
           18 A2           0360                  MSG_L_PID(R2)              ;
   1C A2   20 A3     90    0362          MOVB    IRP$W_FUNC(R3), -          ; SAVE I/O FUNCTION BEING PERFORMED.
                           0367                  MSG_B_FUNC(R2)             ;
         58   52     D0    0367          MOVL    R2,R8                      ; SAVE ADDRESS OF BLOCK
                           036A  ;
                           036A  ; COPY DATA FROM USER BUFFER TO SHARED MEMORY
                           036A  ;
              3C     BB    036A          PUSHR   #^M<R2,R3,R4,R5>           ; SAVE REGISTERS
           0C A2     28    036C          MOVC3   MSG_W_LENGTH(R2),-         ; MOVE FROM USER TO SHARED MEMORY
      1D A2   6A           036F                  (R10),MSG_B_MESSAGE(R2)    ;
              3C     BA    0372          POPR    #^M<R2,R3,R4,R5>           ; RESTORE REGISTERS
      50   0C A2     3C    0374          MOVZWL  MSG_W_LENGTH(R2),R0        ; GET SIZE OF MESSAGE AGAIN
         5A   50     C0    0378          ADDL    R0,R10                     ; INCREMENT USER BUFFER ADDR
         59   50     C2    037B          SUBL    R0,R9                      ; DECREMENT USER BUFFER SIZE
              03     13    037E          BEQL    CHECK_QUOTAS               ; IF EQL, NO MORE
            FF69     31    0380          BRW     ALLOC_BLOCK                ; ELSE, ALLOCATE ANOTHER BLOCK
                           0383  ;
                           0383  ; INTERLOCK QUOTA CHECKS
                           0383  ;
```

```
                              0383  CHECK_QUOTAS:                                  ; CHECK MAILBOX QUOTAS
                              0383          DSBINT  #IPL$_MAILBOX                  ; DISABLE MAILBOX I/O INTERRUPTS
    51   009C C5         D0   0389          MOVL    UCB$L_MB_SHB(R5),R1            ; GET ADDRESS OF SHB
      51   04 A1         D0   038E          MOVL    SHB$L_DATAPAGE(R1),R1          ; GET ADDRESS OF DATAPAGE
         50   02         D0   0392          MOVL    #RSN$_MAILBOX,R0               ; GET RESOURCE NUMBER
  7E    00A8 C140        3E   0395          MOVAW   SHD$W_RESWAIT(R1)[R0],-(SP)    ; SAVE ADDRESS OF WAIT MASK
                              039B          SET_PORTFLAG @(SP)                     ; ASSUME MAILBOX FULL FAILURE
                              03A2                                                 ; (AVOIDS MISSING NOTIFICATION)
    52   0098 C5         D0   03A2          MOVL    UCB$L_MB_MBX(R5),R2            ; GET ADDRESS OF MAILBOX
                              03A7          LOCK    #MBX$V_QUOTALCK,MBX$B_FLAGS(R2) ; INTERLOCK QUOTA CHECKS
                              03C4  ;
                              03C4  ; SEE IF MESSAGE WILL EXCEED MAILBOX BUFFER QUOTA, IN OTHER WORDS, IS
                              03C4  ; THE MAILBOX FULL?
                              03C4  ;
  18 A2     32 A3        B1   03C4          CMPW    IRP$W_BCNT(R3),MBX$W_BUFFQUO(R2) ; MESSAGE FIT?
               78        1A   03C9          BGTRU   MAILBOX_FULL                   ; IF GTR THEN NO
                              03CB  ;
                              03CB  ; ADJUST MESSAGE COUNT AND BUFFER QUOTA
                              03CB  ;
            16 A2        B6   03CB          INCW    MBX$W_MSGCNT(R2)               ; INCREMENT MESSAGE COUNT
            32 A3        A2   03CE          SUBW    IRP$W_BCNT(R3),-               ; DECREASE BUFFER QUOTA
            18 A2             03D1                  MBX$W_BUFFQUO(R2)              ;
                              03D3          UNLOCK  #MBX$V_QUOTALCK,MBX$B_FLAGS(R2) ; UNLOCK MAILBOX QUOTAS
  44 A5     16 A2        B0   03DB          MOVW    MBX$W_MSGCNT(R2),UCB$L_DEVDEPEND(R5) ; SAVE MESSAGE COUNT
                              03E0          CLR_PORTFLAG @(SP)+                    ; CLEAR WAIT FLAG AS MAILBOX NOT FULL
                              03E6  ;
                              03E6  ; QUEUE THE MESSAGE.  MUST BE QUEUED BEFORE WE LOOK FOR ANYONE WAITING
                              03E6  ; TO AVOID MISSING AN INTERESTED PROCESSOR.
                              03E6  ;
                              03E6          QRETRY  SUCCESS=20$,-                  ; INSERT MESSAGE IN QUEUE
                              03E6          INSQTI  (R11),MBX$Q_MSG(R2)            ;
                              03F5          ENBINT                                 ; RE-ENABLE INTERRUPTS
            007A         30   03F8          BSBW    DALLOC_BLOCKS                  ; DEALLOCATE MESSAGE BLOCKS
   50   0394 8F          3C   03FB          MOVZWL  #SS$_BADQUEUEHDR,R0            ; SET FAILURE STATUS
    00000000'GF          17   0400          JMP     G^EXE$FINISHIOC                ; COMPLETE THE I/O
                              0406  ;
                              0406  ; NOTIFY OTHER INTERESTED PROCESSORS THAT A MESSAGE WAS WRITTEN.
                              0406  ;
                              0406  20$:
               28        BB   0406          PUSHR   #^M<R3,R5>                     ; SAVE REGISTERS
             0237        30   0408          BSBW    NOTIFY_WRITE                   ; NOTIFY INTERESTED PROCESSORS
               28        BA   040B          POPR    #^M<R3,R5>                     ; RESTORE REGISTERS
                              040D  ;
                              040D  ; UPDATE MEASUREMENT COUNTER IF ENABLED
                              040D  ;
                 00000002     040D          .IF NE CA$_MEASURE                     ; CHECK FOR MEASUREMENT ENABLED
    00000000'GF          D6   040D          INCL    G^PMS$GL_MBWRITES              ; COUNT MAILBOX WRITES
                              0413          .ENDC
                              0413  ;
                              0413  ; IF I/O REQUEST SPECIFIED IO$M_NOW, COMPLETE IT.  ELSE, INSERT I/O
                              0413  ; PACKET IN WRITE QUEUE AND IT QILL BE COMPLETED WHEN MESSAGE IS READ.
                              0413  ;
 0E 20 A3      06        E0   0413          BBS     #IO$V_NOW,IRP$W_FUNC(R3),30$; BR IF WRITE NOW
  00A0 C5      63        0E   0418          INSQUE  (R3),UCB$L_MB_WIOQFL(R5) ; INSERT IRP IN WRITE I/O QUEUE
                              041D          ENBINT                                 ; RE-ENABLE INTERRUPTS
    00000000'GF          17   0420          JMP     G^EXE$QIORETURN                ; RETURN TO CALLER
                              0426  30$:
```

```
                           0426    925          ENBINT                              ; RE-ENABLE INTERRUPTS
          50    30 A3   D0 0429    926          MOVL     IRP$W_BCNT-2(R3),R0        ; SET BYTE COUNT IN 2ND WORD
            50    01    B0 042D    927          MOVW     #SS$_NORMAL,R0             ; SET STATUS IN 1ST WORD
         00000000'GF    17 0430    928          JMP      G^EXE$FINISHIOC            ; COMPLETE THE I/O
```

```
MB
Sy

$$
$$
AL
AL
AL
AT
CA
CA
CA
CA
CC
CC
CC
CH
CH
CO
CO
CO
CO
CR
CX
CX
DA
DA
DC
DD
DE
DE
DE
DE
DE
DE
DE
DP
DP
DP
DP
DP
DT
DY
DY
DY
DY
DY
DY
DY
ER
EX
EX
EX
EX
EX
EX
EX
EX
EX
EX
```

```
                            0436        .SBTTL  ALLOC_FAIL/MAILBOX_FULL - WRITE FDT ROUTINE FAILURES
                            0436  ;++
                            0436  ;
                            0436  ; ALLOC_FAIL - SHARED MEMORY POOL ALLOCATION  FAILURE.
                            0436  ; MAILBOX_FULL - MAILBOX QUOTA FAILURE.
                            0436  ;
                            0436  ; INPUTS:
                            0436  ;
                            0436  ;       R3 = IRP ADDRESS.
                            0436  ;       R5 = UCB ADDRESS.
                            0436  ;       R11 = FIRST SHARED MEMORY MESSAGE BLOCK ADDRESS.
                            0436  ;
                            0436  ; ALLOC_FAIL:
                            0436  ;       (SP) = ADDRESS OF SHARED MEMORY WAIT MASK.
                            0436  ;       4(SP) = OLD IPL (IPL$_ASTDEL)
                            0436  ;
                            0436  ; MAILBOX_FULL:
                            0436  ;       (SP) = OLD IPL (IPL$_MAILBOX)
                            0436  ;       4(SP) = ADDRESS OF SHARED MEMORY WAIT MASK
                            0436  ;       8(SP) = OLD IPL (IPL$_ASTDEL)
                            0436  ;
                            0436  ; OUTPUTS:
                            0436  ;
                            0436  ;       ALL SHARED MEMORY MESSAGE BLOCKS IN THE CHAIN ARE DEALLOCATED
                            0436  ;       AND THE REQUESTING PROCESS IS PUT IN A WAIT STATE UNTIL THE
                            0436  ;       NEEDED RESOURCE BECOMES AVAILABLE.
                            0436  ;--
                            0436  ALLOC_FAIL:                                     ; SHARED MEMORY ALLOCATION FAILURE
       5E    04       C0    0436          ADDL    #4,SP                           ; POP ADDR. OF WAIT MASK OFF STACK
 6E    0124  8F       3C    0439          MOVZWL  #SS$_INSFMEM,(SP)               ; SET FAILURE STATUS, FORGETTING IPL
       51    03       9A    043E          MOVZBL  #RSN$_NPDYNMEM,R1               ; SET RESOURCE TO AWAIT
             16       11    0441          BRB     SHMRES_WAIT                     ;
                            0443  MAILBOX_FULL:                                   ; MAILBOX IS FULL
                            0443          UNLOCK  #MBX$V_QUOTALCK,MBX$B_FLAGS(R2) ; UNLOCK QUOTAS
       5E    04       C0    044B          ADDL    #4,SP                           ; POP MASK ADDRESS OFF STACK
 6E    08D8  8F       3C    044E          MOVZWL  #SS$_MBFULL,(SP)                ; SET FAILURES STATUS, FORGETTING IPL
                            0453          SETIPL  #IPL$_SYNCH
       51    02       9A    0456          MOVZBL  #RSN$_MAILBOX,R1                ; SET RESOURCE TO AWAIT
                            0459  SHMRES_WAIT:                                    ; WAIT FOR SHARED MEMORY RESOURCE
             1A       10    0459          BSBB    DALLOC_BLOCKS                   ; DEALLOCATE SHARED MEMORY BLOCKS
                            045B  ;
                            045B  ; WAIT FOR NEEDED RESOURCE, BY DEALLOCATING I/O PACKETS, RESTORING
                            045B  ; I/O QUOTAS AND COUNTS AND INSERTING PROCESS IN MWAIT STATE QUEUE.
                            045B  ; WHEN RESOURCE BECOMES AVAILABLE, PROCESS WILL BE RESTARTED AT
                            045B  ; BEGININNING OF $QIO REQUEST.
                            045B  ;
                            045B  RES_WAIT:                                       ; WAIT FOR NEEDED RESOURCE
          02C6        30    045B          BSBW    DALLOC_IRPE                     ; DEALLOCATE IRPE'S
             50     8ED0    045E          POPL    R0                              ; GET FAILURE STATUS
                            0461          SETIPL  #IPL$_ASTDEL                    ; SYNCHRONIZE FOR QIO BACKOUT & WAIT
06 20 A3     0A       E0    0464          BBS     #IO$V_NORSWAIT, -               ; IS NO RESOURCE WAIT MODIFIER SET?
                            0469                  IRP$W_FUNC(R3), 69$             ; BRANCH IF MODIFER IS SET.
   00000000'GF        17    0469          JMP     G^EXE$IORSNWAIT                 ; ELSE, DO POSSIBLE RESOURCE WAIT.
   00000000'GF        17    046F  69$:    JMP     G^EXE$ABORTIO                   ; ABORT I/O TO AVOID RESOURCE WAITS.
```

```
MB
Sy
IR
IR
IR
IR
IR
IR
IR
LN
LN
LN
MA
MA
MA
MA
MA
MB
MB
MB
MB
MB
MB
MB
MB
MB
MB
MB
MB
MB
MB
MB
MS
MS
MS
MS
MS
MS
MS
MS
MS
MS
MS
MS
MS
NO
NO
NO
NO
NO
OR
OR
OR
OR
OR
P1
P2
P3
P4
```

```
                          0475    985         .SBTTL  DALLOC_BLOCKS - DEALLOCATE SHARED MEMORY BLOCKS
                          0475    986 ;++
                          0475    987 ;
                          0475    988 ; DALLOC_BLOCKS - DEALLOCATE ANY SHARED MEMORY BLOCKS
                          0475    989 ;
                          0475    990 ; INPUTS:
                          0475    991 ;
                          0475    992 ;       R11 = FIRST BLOCK ADDRESS.
                          0475    993 ;
                          0475    994 ; OUTPUTS:
                          0475    995 ;
                          0475    996 ;       ALL SHARED MEMORY MESSAGE BLOCKS IN THE CHAIN ARE DEALLOCATED.
                          0475    997 ;
                          0475    998 ;       R1,R2,R3 ARE PRESERVED.
                          0475    999 ;
                          0475   1000 ;--
                          0475   1001 DALLOC_BLOCKS:                               ; DEALLOCATE SHARED MEMORY
                 0E   BB  0475   1002         PUSHR   #^M<R1,R2,R3>                ; SAVE REGISTERS
          50     5B   D0  0477   1003         MOVL    R11,R0                       ; GET ADDRESS OF FIRST BLOCK
                 12   13  047A   1004         BEQL    20$                          ; IF EQL NOT ALLOCATED
                          047C   1005 10$:
      5A    10 A0     D0  047C   1006         MOVL    MSG_L_CHAINLINK(R0),R10      ; GET OFFSET TO NEXT BLOCK
     00000000'GF      16  0480   1007         JSB     G^EXE$DEASHARED              ; DEALLOCATE THE BLOCK
    50     5A    5B   C1  0486   1008         ADDL3   R11,R10,R0                   ; COMPUTE ADDRESS OF NEXT BLOCK
                 5A   D5  048A   1009         TSTL    R10                          ; IS THERE A NEXT BLOCK?
                 EE   12  048C   1010         BNEQ    10$                          ; IF NEQ YES
                          048E   1011 20$:
                 0E   BA  048E   1012         POPR    #^M<R1,R2,R3>                ; RESTORE REGISTERS
                      05  0490   1013         RSB                                  ;
                          0491   1014
```

```
MB
Sy

UC
UC
UC
UC
UC
UC
UC
UC
UC
UC
UC
UC
UC
UC
UC
UC
UC
VE
VE
WR
WR
WR
ZE

PS
--
.
$A
$$
$$

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

Th
15
Th
14
55
```

```
                     0491          .SBTTL  STARTIO - STARTIO OPERATION
                     0491  ;++
                     0491  ; STARTIO - START READ OPERATION ON SHARED MEMORY MAILBOX
                     0491  ;
                     0491  ; FUNCTIONAL DESCRIPTION:
                     0491  ;
                     0491  ; THIS ROUTINE IS ENTERED WHEN THE UNIT IS NOT BUSY AND THERE IS A
                     0491  ; PACKET TO PROCESS.
                     0491  ;
                     0491  ; IF THERE IS A MESSAGE WAITING IN THE MAILBOX:
                     0491  ;
                     0491  ;       o  THE MESSAGE IS DEQUEUED
                     0491  ;
                     0491  ;       o  THE MAILBOX QUOTAS ARE ADJUSTED
                     0491  ;
                     0491  ;       o  IF THE MESSAGE CONTAINED THE MESSAGE WRITER'S IRP SEQUENCE
                     0491  ;          NUMBER, A MESSAGE IS SENT TO THE APPROPRIATE PROCESSOR TO
                     0491  ;          TO INDICATE THE WRITE I/O SHOULD BE COMPLETED.
                     0491  ;
                     0491  ;       o  THE READ I/O REQUEST IS POSTED WITH THE ADDRESS OF THE MESSAGE
                     0491  ;
                     0491  ; IF THERE IS NO MESSAGE WAITING IN THE MAILBOX AND THE I/O REQUEST
                     0491  ; SPECIFIED IO$M_NOW, THE REQUEST IS COMPLETED WITH FAILURE (SS$_ENDOFILE).
                     0491  ;
                     0491  ; IF THERE IS NO MESSAGE WAITING IN THE MAILBOX AND THE I/O REQUEST
                     0491  ; DID NOT SPECIFY IO$M_NOW:
                     0491  ;
                     0491  ;       o  THE PORT'S WAITING READER FLAG (MBX$W_READER) IS SET
                     0491  ;
                     0491  ;       o  THE READ ATTENTION AST FLAGS (MBX$W_READAST) FOR ALL PORTS
                     0491  ;          ARE SCANNED AND IF SET, A MESSAGE IS SENT TO THE APPROPRIATE
                     0491  ;          PROCESSOR TO INDICATE THAT THE AST'S SHOULD BE DELIVERED.
                     0491  ;
                     0491  ;       o  AN RSB TO THE DRIVERS CALLER IS EXECUTED LEAVING THE DRIVER
                     0491  ;          TO AWAIT MESSAGE WRITTEN NOTIFICATION.
                     0491  ;
                     0491  ; INPUTS:
                     0491  ;
                     0491  ;       R3 = I/O PACKET ADDRESS
                     0491  ;       R5 = UCB ADDRESS
                     0491  ;
                     0491  ; OUTPUTS:
                     0491  ;
                     0491  ;       R1 = OUR PORT NUMBER.
                     0491  ;       R2 = FIRST MESSAGE BLOCK ADDRESS.
                     0491  ;       R4 = MAILBOX ADDRESS.
                     0491  ;
                     0491  ;       OTHERWISE AN RSB IS DONE.
                     0491  ;--
                     0491  STARTIO:                                      ;
 54   0098 C5   D0   0491          MOVL     UCB$L_MB_MBX(R5),R4         ; GET MAILBOX ADDRESS
                     0496          SET_PORTFLAG MBX$W_READER(R4)        ; SET THAT WE HAVE A READER
                     049D          QRETRY   SUCCESS=10$,-               ; ATTEMPT TO DEQUEUE A MESSAGE
                     049D          REMQHI   MBX$Q_MSG(R4),R2            ;
 50   0394 8F   3C   04AC          MOVZWL   #SS$_BADQUEUEHDR,R0         ; SET FAILURE STATUS
           51   D4   04B1          CLRL     R1
                     04B3          REQCOM                               ; COMPLETE THE READ REQUEST
```

```
                        04B9  ;
                        04B9  ; IF A MESSAGE WAS DEQUEUED, COMPLETE THE I/O REQUEST.  OTHERWISE,
                        04B9  ; IF IO$M_NOW WAS SPECIFIED EXIT WITH FAILURE.  OTHERWISE, NOTIFY
                        04B9  ; ANY INTERESTED PROCESSORS THAT A READER IS WAITING AND JUST RETURN
                        04B9  ; TO WAIT FOR A MESSAGE TO BE WRITTEN.
                        04B9  ;
                        04B9  10$:
               1D  1C   04B9          BVC     FINISHREAD                      ; IF V-CLEAR, MESSAGE DEQUEUED
      0D 20 A3  06  E1  04BB          BBC     #IO$V_NOW,IRP$W_FUNC(R3),20$ ; IF CLEAR, WAIT
      50    0870 8F  3C  04C0          MOVZWL  #SS$_ENDOFFILE,R0               ; SET FAILURE STATUS
               51  D4   04C5  15$:    CLRL    R1                              ;
                        04C7          REQCOM                                  ; COMPLETE THE READ REQUEST
                        04CD  20$:
               28  BB   04CD          PUSHR   #^M<R3,R5>                      ; SAVE REGISTERS
             0153  30   04CF          BSBW    NOTIFY_READER                   ; NOTIFY PROCESSORS OF READER
               28  BA   04D2          POPR    #^M<R3,R5>                      ; RESTORE REGISTERS
            EE 50  E9   04D4          BLBC    R0,15$                          ; IF FAILURE, EXIT
                   05   04D7          RSB                                     ; ELSE, WAIT FOR MESSAGE NOTIFICATION
                   04D8  1092          .SBTTL  FINISHREAD - FINISH READ I/O OPERATION
                   04D8  1093 ;++
                   04D8  1094 ; FINISHREAD - FINISH READ OPERATION
                   04D8  1095 ;
                   04D8  1096 ; FUNCTIONAL DECRIPTION:
                   04D8  1097 ;
                   04D8  1098 ; THIS ROUTINE IS ENTERED WHEN A MESSAGE IS AVAILABLE FOR A READ I/O
                   04D8  1099 ; REQUEST.
                   04D8  1100 ;
                   04D8  1101 ; INPUTS:
                   04D8  1102 ;
                   04D8  1103 ;       R2 = FIRST MESSAGE BLOCK ADDRESS.
                   04D8  1104 ;       R3 = I/O REQUEST PACKET ADDRESS
                   04D8  1105 ;       R4 = MAILBOX ADDRESS.
                   04D8  1106 ;       R5 = UCB ADDRESS
                   04D8  1107 ;
                   04D8  1108 ; OUTPUTS:
                   04D8  1109 ;
                   04D8  1110 ;--
                   04D8  1111 FINISHREAD:                                 ;
                   04D8  1112         CLR_PORTFLAG MBX$W_READER(R4)       ; CLEAR WAITING READER FLAG
                   04DF  1113 ;
                   04DF  1114 ; FORMAT MESSAGE BLOCKS FOR I/O POST
                   04DF  1115 ;
         28    A8  04DF  1116         BISW    #IRP$M_COMPLX!IRP$M_CHAINED,- ;SET COMPLEX/CHAINED I/O
      2A A3        04E1  1117                 IRP$W_STS(R3)               ;
2C A3    52    D0  04E3  1118         MOVL    R2,IRP$L_SVAPTE(R3)         ; INSERT BLOCK ADDRESS IN PACKET
      38 A3    D0  04E7  1119         MOVL    IRP$L_MEDIA(R3),-           ; INSERT USER BUFFER ADDRESS
      04 A2        04EA  1120                 MSG_L_POSTUBUF(R2)          ;
   50    52    D0  04EC  1121         MOVL    R2,R0                       ; GET FIRST BLOCK ADDRESS
                   04EF  1122 10$:
      1D A0    9E  04EF  1123         MOVAB   MSG_B_MESSAGE(R0),-         ; INSERT ADDRESS OF DATA
         60        04F2  1124                 MSG_L_POSTIOBUF(R0)         ;
51    10 A0    D0  04F3  1125         MOVL    MSG_L_CHAINLINK(R0),R1      ; GET OFFSET TO NEXT BLOCK
         0C    13  04F7  1126         BEQL    RESTORE_QUOTAS              ; IF EQL NONE
   51    52    C0  04F9  1127         ADDL    R2,R1                       ; COMPUTE ADDRESS OF BLOCK
10 A0    51    D0  04FC  1128         MOVL    R1,MSG_L_CHAINLINK(R0)      ; SET ADDRESS AS LINK
   50    51    D0  0500  1129         MOVL    R1,R0                       ; GET NEW ADDRESS
         EA    11  0503  1130         BRB     10$                         ;
                   0505  1131 ;
                   0505  1132 ; RESTORE MAILBOX QUOTAS
                   0505  1133 ;
                   0505  1134 RESTORE_QUOTAS:                             ; RESTORE MAILBOX QUOTAS
                   0505  1135         LOCK    #MBX$V_QUOTALCK,MBX$B_FLAGS(R4) ; LOCK MAILBOX QUOTAS
      16 A4    B7  0522  1136         DECW    MBX$W_MSGCNT(R4)            ; DECREMENT MESSAGE COUNT
      0E A2    A0  0525  1137         ADDW    MSG_W_MSGLENGTH(R2),-       ; RESTORE BUFFER QUOTA
      18 A4        0528  1138                 MBX$W_BUFFQUO(R4)           ;
                   052A  1139         UNLOCK  #MBX$V_QUOTALCK,MBX$B_FLAGS(R4) ; UNLOCK MAILBOX QUOTAS
      16 A4    B0  0532  1140         MOVW    MBX$W_MSGCNT(R4),-          ; SAVE MESSAGE COUNT
      44 A5        0535  1141                 UCB$L_DEVDEPEND(R5)         ;
                   0537  1142 ;
                   0537  1143 ; NOTIFY WRITER THAT THE MESSAGE WAS READ (IF WRITER WANTED TO KNOW)
                   0537  1144 ; AND REPORT MAILBOX RESOURCE AVAILABILITY.
                   0537  1145 ;
         3C    BB  0537  1146         PUSHR   #^M<R2,R3,R4,R5>            ; SAVE REGISTERS
    009C C5    DD  0539  1147         PUSHL   UCB$L_MB_SHB(R5)            ; SAVE ADDRESS OF SHB
      00F3     30  053D  1148         BSBW    NOTIFY_READ                 ; NOTIFY WRITER
```

```
                 50   02   D0  0540   1149          MOVL     #RSN$_MAILBOX,R0        ; GET MAILBOX RESOURCE NUMBER
                      51 8ED0  0543   1150          POPL     R1                      ; RESTORE ADDRESS OF SHB
           00000000'GF    16   0546   1151          JSB      G^MA$RAVAIL             ; REPORT THE RESOURCE AVAILABLE
                      3C   BA  054C   1152          POPR     #^M<R2,R3,R4,R5>        ; RESTORE REGISTERS
                               054E   1153 ;
                               054E   1154 ; COMPUTE SIZE OF DATA AND STATUS, AND COMPLETE THE READ I/O REQUEST.
                               054E   1155 ;
           50   0601 8F    B0  054E   1156          MOVW     #SS$_BUFFEROVF, R0      ; Assume buffer overflow.
         0E A2    32 A3    B1  0553   1157          CMPW     IRP$W_BCNT(R3), -       ;
                               0558   1158                   MSG_W_MSGLENGTH(R2)     ; Was there a buffer overflow?
                      08   1F  0558   1159          BLSSU    10$                     ; Branch if buffer overflow.
        32 A3     0E A2    B0  055A   1160          MOVW     MSG_W_MSGLENGTH(R2), -  ; Otherwise, transfer only the
                               055F   1161                   IRP$W_BCNT(R3)          ; bytes actually in the message.
                 50   01   B0  055F   1162          MOVW     #SS$_NORMAL, R0         ; and set normal xfer completed.
  50    10    10  32 A3    F0  0562   1163 10$:     INSV     IRP$W_BCNT(R3), #16, #16, R0 ; Plant bytes transfered count.
              28  1C A2    91  0568   1164          CMPB     MSG_B_FUNC(R2), #IO$_WRITEOF ; Was this an end-of-file function?
                      05   12  056C   1165          BNEQ     20$                     ; Branch if not an end-of-file.
           50   0870 8F    B0  056E   1166          MOVW     #SS$_ENDOFFILE,R0       ; Else, set eof status.
                               0573   1167 20$:
              51    18 A2  D0  0573   1168          MOVL     MSG_L_PID(R2),R1        ; GET EXTENDED PID OF WRITER
                               0577   1169          REQCOM                           ; COMPLETE READ I/O REQUEST
```

```
                        057D             .SBTTL  MBX$INT - INTERRUPT DISPATCHER
                        057D       ;++
                        057D       ;
                        057D       ; MBX$INT - PORT REQUEST INTERRUPT DISPATCHER
                        057D       ;
                        057D       ; FUNCTIONAL DESCRIPTION:
                        057D       ;
                        057D       ;       THIS ROUTINE IS CALLED WHEN THE PORT DRIVER RECEIVES A
                        057D       ;       REQUEST FROM A PROCESSOR THAT SPECIFIES THE MAILBOX
                        057D       ;       DRIVER AS THE MESSAGE DISPATCHER ID (PRQ$C_MAILBOX).
                        057D       ;
                        057D       ;       THIS ROUTINE EXAMINES THE REQUEST TYPE CODE AND DETERMINES
                        057D       ;       WHETHER IT SHOULD:
                        057D       ;
                        057D       ;         o  DELIVER ALL THE WPITE ATTENTION AST'S FOR A UNIT AND
                        057D       ;            COMPLETE ANY WAITING READ REQUEST(S) (PRQ_WRITE)
                        057D       ;
                        057D       ;         o  DELIVER ALL THE READ ATTENTION AST'S FOR A   'IT BECAUSE
                        057D       ;            A READER IS WAITING (PRQ_READER)
                        057D       ;
                        057D       ;         o  COMPLETE A WRITE I/O REQUEST BECAUSE THE MESSAGE WRITTEN
                        057D       ;            WAS READ BY ANOTHER PROCESS (PRQ_READ)
                        057D       ;
                        057D       ; INPUTS:
                        057D       ;
                        057D       ;       R0-R4 = SCRATCH.
                        057D       ;       R5 = INTER-PROCESSOR REQUEST BLOCK ADDRESS.
                        057D       ;
                        057D       ;       00(SP) = ADDRESS OF IDB ADDRESS.
                        057D       ;
                        057D       ;       IPL = IPL$_MAILBOX
                        057D       ;
                        057D       ; OUTPUTS:
                        057D       ;
                        057D       ;       APPROPRIATE ACTION IS TAKEN DEPENDING ON THE REQUEST TYPE.
                        057D       ;
                        057D       ;--
                        057D      MBX$INT:                                      ; INTERRUPT DISPATCHER
       53    9E     D0  057D             MOVL    @(SP)+,R3                      ; GET ADDRESS OF IDB
       52    55     D0  0580             MOVL    R5,R2                          ; SAVE REQUEST BLOCK ADDRESS
    55    22 A2     3C  0583             MOVZWL  PRQ$W_UNIT(R2),R5              ; GET UNIT NUMBER OF REQUEST
55    18 A345       D0  0587             MOVL    IDB$L_UCBLST(R3)[R5],R5        ; GET UCB ADDRESS
             10     13  058C             BEQL    INT_EXIT                       ; IF EQL NO CORRESPONDING UNIT
54    0098 C5       D0  058E             MOVL    UCB$L_MB_MBX(R5),R4            ; GET MAILBOX ADDRESS
                        0593             CASE    PRQ$W_REQTYPE(R2),-            ; DISPATCH ON REQUEST TYPE
                        0593                     LIMIT=#PRQ_READ,<-             ;  LOW LIMIT
                        0593                     READ_REQ,-                     ;  MESSAGE WAS READ
                        0593                     WRITE_REQ,-                    ;  MESSAGE WAS WRITTEN
                        0593                     READER_REQ,-                   ;  READER WAITING
                        0593                     >                              ;
                        059E      INT_EXIT:                                     ; EXIT INTERRUPT
                    05  059E             RSB                                    ;
                        059F       ;
                        059F       ; READER WAITING REQUEST - DELIVER ANY READ ATTENTION AST'S
                        059F       ;
                        059F      READER_REQ:                                   , READER WAITING REQUEST
                        059F             CLR_PORTFLAG MBX$W_READAST(R4)         ; CLEAR NOTIFY FLAG
```

```
   54   0094 C5   DE  05A6          MOVAL   UCB$L_MB_RAST(R5),R4    ; GET ADDRESS OF READ AST LIST
    00000000'GF   17  05AB          JMP     G^COM$DELATTNAST        ; DELIVER ANY AST'S AND EXIT
                      05B1   ;
                      05B1   ; MESSAGE WAS WRITTEN REQUEST - DELIVER ANY WRITE ATTENTION AST'S AND
                      05B1   ; IF A READ I/O REQUEST IS ALREADY WAITING FOR A MESSAGE, ATTEMPT TO
                      05B1   ; DEQUEUE A MESSAGE AND COMPLETE THE I/O REQUEST.
                      05B1   ;
                      05B1   WRITE_REQ:                                 ; MESSAGE WAS WRITTEN REQUEST
                      05B1              CLR_PORTFLAG MBX$W_WRITAST(R4)  ; CLEAR NOTIFY FLAG
   54   0090 C5   DE  05B8          MOVAL   UCB$L_MB_WAST(R5),R4    ; GET ADDRESS OF WRITE AST LIST
    00000000'GF   16  05BD          JSB     G^COM$DELATTNAST        ; DELIVER ANY AST'S
   54   0098 C5   D0  05C3          MOVL    UCB$L_MB_MBX(R5),R4     ; GET MAILBOX ADDRESS AGAIN
 D1 64 A5   08    E1  05C8          BBC     #UCB$V_BSY,UCB$W_STS(R5),INT_EXIT ; IF CLEAR, NO READER WAITING
    53   58 A5    D0  05CD          MOVL    UCB$L_IRP(R5),R3        ; GET I/O PACKET ADDRESS
                      05D1          QRETRY  ERROR=10$,-             ; ATTEMPT TO DEQUEUE A MESSAGE
                      05D1          REMQHI  MBX$Q_MSG(R4),R2        ;
             BA   1D  05E2          BVS     INT_EXIT                ; IF V-SET, NO MESSAGE
           FEF1   31  05E4          BRW     FINISHREAD              ; ELSE, MESSAGE DEQUEUED
                      05E7   10$:
   50   0394 8F   3C  05E7          MOVZWL  #SS$_BADQUEUEHDR,R0     ; SET FAILURE STATUS
             51   D4  05EC          CLRL    R1                      ;
                      05EE          REQCOM                          ; COMPLETE THE READ REQUEST
                      05F4   ;
                      05F4   ; MESSAGE WAS READ REQUEST - COMPLETE THE ORIGINAL WRITE I/O REQUEST
                      05F4   ;
                      05F4   READ_REQ:                                  ; MESSAGE WAS READ REQUEST
   53   00A0 C5   DE  05F4          MOVAL   UCB$L_MB_WIOQFL(R5),R3  ; GET ADDRESS OF WRITE PACKET LISTHEAD
        51   53   D0  05F9          MOVL    R3,R1                   ; SAVE A COPY OF IT
                      05FC   10$:
        53   63   D0  05FC          MOVL    (R3),R3                 ; GET ADDRESS OF NEXT PACKET
        51   53   D1  05FF          CMPL    R3,R1                   ; END OF QUEUE?
             9A   13  0602          BEQL    INT_EXIT                ; IF EQL YES - REQUEST GONE
          50 A3   D1  0604          CMPL    IRP$L_SEQNUM(R3),-      ; IS THIS THE CORRECT REQUEST?
          24 A2       0607                  PRQ$L_PARAM(R2)         ;
             F1   12  0609          BNEQ    10$                     ; IF NEQ NO
        53   63   0F  060B          REMQUE  (R3),R3                 ; REMOVE PACKET FROM QUEUE
     50   32 A3   B0  060E          MOVW    IRP$W_BCNT(R3),R0       ; GET BYTE COUNT OF MESSAGE
   50   50   10   78  0612          ASHL    #16,R0,R0               ; MOVE TO UPPER WORD
        50   01   B0  0616          MOVW    #SS$_NORMAL,R0          ; SET SUCCESS STATUS
             51   D4  0619          CLRL    R1                      ; (NO PID)
     38 A3   50   7D  061B          MOVQ    R0,IRP$L_IOST1(R3)      ; SET I/O STATUS IN IRP
    00000000'GF   17  061F          JMP     G^COM$POST              ; COMPLETE THE WRITE REQUEST
```

```
                        0625          .SBTTL  NOTIFY - NOTIFY OTHER PROCESSORS OF CONDITIONS
                        0625  ;++
                        0625  ;
                        0625  ; NOTIFY_READER - NOTIFY OTHER PROCESSORS OF A READER
                        0625  ; NOTIFY_READ - NOTIFY OTHER PROCESSOR THAT A MESSAGE WAS READ
                        0625  ; NOTIFY_WRITE - NOTIFY OTHER PROCESSORS THAT A MESSAGE WAS WRITTEN
                        0625  ;
                        0625  ; THESE ROUTINES ARE CALLED TO FORMAT AND SEND A REQUEST TO THE
                        0625  ; MAILBOX DRIVERS ON OTHER PROCESSORS.
                        0625  ;
                        0625  ; INPUTS:
                        0625  ;
                        0625  ;       R2 = FIRST MESSAGE BLOCK ADDRESS (NOTIFY_READ ONLY)
                        0625  ;       R3 = I/O PACKET ADDRESS
                        0625  ;       R5 = UCB ADDRESS
                        0625  ;
                        0625  ; OUTPUTS:
                        0625  ;
                        0625  ;       R0 = SUCCESS/FAILURE.
                        0625  ;
                        0625  ;       REQUEST(S) FORMATTED AND PASSED TO PORT DRIVER FOR DELIVERY
                        0625  ;       TO REQUIRED PROCESSORS.
                        0625  ;
                        0625  ;       R0,R1,R2,R3,R4,R5 DESTROYED
                        0625  ;
                        0625  ;--
                        0625
                        0625  ;
                        0625  ; NOTIFY ANY INTERESTED PROCESSORS THAT A READER IS WAITING
                        0625  ;
                        0625  NOTIFY_READER:                              ; NOTIFY READER AVAILABLE
     52   0098 C5   D0  0625          MOVL    UCB$L_MB_MBX(R5),R2         ; GET ADDRESS OF MAILBOX
     50     10 A2   B0  062A          MOVW    MBX$W_READAST(R2),R0        ; GET PORT #'S TO NOTIFY
      51    03      3C  062E          MOVZWL  #PRQ_READER,R1              ; SET REQUEST TYPE
            1F      11  0631          BRB     NOTIFY                      ; NOTIFY THEM
                        0633  ;
                        0633  ; IF IT WANTED TO KNOW, NOTIFY PROCESSOR THAT WROTE MESSAGE THAT IT
                        0633  ; WAS READ.
                        0633  ;
                        0633  NOTIFY_READ:                                ; NOTIFY MESSAGE READ
  50   01   0B A2   78  0633          ASHL    MSG_B_PORT(R2),#1,R0        ; GET PORT # NOTIFY
      51    01      3C  0638          MOVZWL  #PRQ_READ,R1                ; SET REQUEST TYPE
     52     14 A2   D0  063B          MOVL    MSG_L_IRPSEQ(R2),R2         ; GET WRITER'S PACKET #
            11      12  063F          BNEQ    NOTIFY                      ; IF NEQ - WRITER IS INTERESTED
                    05  0641          RSB                                 ; ELSE - JUST RETURN
                        0642  ;
                        0642  ; NOTIFY ANY INTERESTED PROCESSORS THAT A MESSAGE WAS WRITTEN
                        0642  ;
                        0642  NOTIFY_WRITE:                               ; NOTIFY MESSAGE WRITTEN
     52   0098 C5   D0  0642          MOVL    UCB$L_MB_MBX(R5),R2         ; GET ADDRESS OF MAILBOX
     50     12 A2   3C  0647          MOVZWL  MBX$W_WRITAST(R2),R0        ; GET PORT #'S TO NOTIFY
     50     0E A2   A8  064B          BISW    MBX$W_READER(R2),R0         ; ...
      51    02      3C  064F          MOVZWL  #PRQ_WRITE,R1               ; SET REQUEST TYPE
                        0652  ;
                        0652  ; NOTIFY PROCESSOR(S) THAT A CONDITION HAS OCCURED
                        0652  ;
                        0652  NOTIFY:                                     ; NOTIFY PORT(S)
```

```
                 55   DD  0652  1328          PUSHL   R5                          ; SAVE UCB ADDRESS
        55    54 A3   D0  0654  1329          MOVL    IRP$L_EXTEND(R3),R5         ; GET FORK BLOCK ADDRESS
              54 A5   D0  0658  1330          MOVL    IRPE$L_EXTEND(R5),-         ; REMOVE BLOCK FROM LIST
              54 A3       065B  1331                  IRP$L_EXTEND(R3)            ;
                 06   12  065D  1332          BNEQ    10$                         ; IF NEQ NOT LAST BLOCK
   2A A3    0800 8F   AA  065F  1333          BICW    #IRP$M_EXTEND,IRP$W_STS(R3) ; ELSE, CLEAR EXTEND FLAG
                          0665  1334 10$:
      18 A5      50   B0  0665  1335          MOVW    R0,IRPE$W_MB_PORTS(R5)      ; SAVE PORT #'S TO NOTIFY
      1A A5      51   B0  0669  1336          MOVW    R1,IRPE$W_MB_RQTYP(R5)      ; SAVE REQUEST TYPE
      1C A5      52   D0  066D  1337          MOVL    R2,IRPE$L_MB_PARAM(R5)      ; SAVE PARAMETER
      0B A5      0B   90  0671  1338          MOVB    #IPL$_MAILBOX,FKB$B_FIPL(R5) ; SET FORK IPL
                 50 8ED0  0675  1339          POPL    R0                          ; RESTORE UCB ADDRESS
    51    009C C0     D0  0678  1340          MOVL    UCB$L_MB_SHB(R0),R1         ; GET SHB ADDRESS
      51      04 A1   D0  067D  1341          MOVL    SHB$L_DATAPAGE(R1),R1       ; GET DATAPAGE ADDRESS
    51    009C C1     9A  0681  1342          MOVZBL  SHD$B_PORTS(R1),R1          ; GET NUMBER OF PORTS
   20 A5     51   01  C3  0686  1343          SUBL3   #1,R1,IRPE$L_MB_PORT(R5)  ; COMPUTE STARTING PORT NUMBER
      54      24 A0   D0  068B  1344          MOVL    UCB$L_CRB(R0),R4            ; GET CRB ADDRESS
      54      38 A4   D0  068F  1345          MOVL    CRB$L_INTD+VEC$L_ADP(R4),R4 ; GET ADP ADDRESS
                          0693  1346 ;
                          0693  1347 ; FORMAT PROCESSOR REQUEST MESSAGE AND RETURN TO PORT DRIVER FOR
                          0693  1348 ; DELIVERY TO OTHER PROCESSOR.
                          0693  1349 ;
                          0693  1350 FORMAT_PRQ:                                 ; FORMAT PROCESSOR REQUEST
            20 A5     E1  0693  1351          BBC     IRPE$L_MB_PORT(R5),-        ; IF CLR, DON'T NOTIFY THE PORT
         2E 18 A5         0696  1352                  IRPE$W_MB_PORTS(R5),10$     ;
      00000000'GF     16  0699  1353          JSB     G^MA$REQUEST                ; CALL PORT DRIVER FOR A REQUEST BLOCK
                          069F  1354                                              ; R2 = MESSAGE BLOCK ADDRESS
            2C 50     E9  069F  1355          BLBC    R0,NOTIFY_DONE              ; IF LBC, FAILURE
            20 A5     B0  06A2  1356          MOVW    IRPE$L_MB_PORT(R5),-        ; SET PORT NUMBER TO SEND TO
            18 A2         06A5  1357                  PRQ$W_TO_PORT(R2)           ;
               01     B0  06A7  1358          MOVW    #PRQ$C_MAILBOX,-            ; SET MESSAGE DISPATCHER ID
            1C A2         06A9  1359                  PRQ$W_DISPATCH(R2)          ;
      50    1C A3     D0  06AB  1360          MOVL    IRP$L_UCB(R3),R0            ; GET UCB ADDRESS
            54 A0     B0  06AF  1361          MOVW    UCB$W_UNIT(R0),-            ; SET UNIT NUMBER
            22 A2         06B2  1362                  PRQ$W_UNIT(R2)              ;
            1A A5     B0  06B4  1363          MOVW    IRPE$W_MB_RQTYP(R5),-       ; SET REQUEST TYPE
            20 A2         06B7  1364                  PRQ$W_REQTYPE(R2)           ;
            1C A5     D0  06B9  1365          MOVL    IRPE$L_MB_PARAM(R5),-       ; SET PARAMETER
            24 A2         06BC  1366                  PRQ$L_PARAM(R2)             ;
      0B A2      0B   90  06BE  1367          MOVB    #IPL$_MAILBOX,FKB$B_FIPL(R2) ; SET DISPATCH IPL
                 9E   16  06C2  1368          JSB     @(SP)+                      ; RETURN TO PORT DRIVER FOR DELIVERY
              07 50   E9  06C4  1369          BLBC    R0,NOTIFY_DONE              ; IF LBC, FAILURE
                          06C7  1370 10$:
         C8 20 A5     F4  06C7  1371          SOBGEQ  IRPE$L_MB_PORT(R5),FORMAT_PRQ ; DECREMENT PORT # AND LOOP
         50      01   D0  06CB  1372          MOVL    #SS$_NORMAL,R0              ; SET SUCCESS
                          06CE  1373 ;
                          06CE  1374 ; DONE WITH NOTIFICATION, DEALLOCATE THE FORK BLOCK
                          06CE  1375 ;
                          06CE  1376 NOTIFY_DONE:                                ; DONE WITH NOTIFICATION
                 50   DD  06CE  1377          PUSHL   R0                          ; SAVE EXIT STATUS
         50      55   D0  06D0  1378          MOVL    R5,R0                       ; SET ADDRESS OF BLOCK
      00000000'GF     16  06D3  1379          JSB     G^EXE$DEANONPAGED           ; DEALLOCATE FORK BLOCK
                 50 8ED0  06D9  1380          POPL    R0                          ; RESTORE EXIT STATUS
                      05  06DC  1381          RSB
```

```
                        06DD          .SBTTL  ALLOC_IRPE - ALLOCATE AN I/O REQUEST PACKET EXTENSION
                        06DD    ;++
                        06DD    ; ALLOC_IRPE - SUBROUTINE TO ALLOCATE AN I/O REQUEST PACKET EXTENSION
                        06DD    ;
                        06DD    ; THIS ROUTINE IS CALLED TO ALLOCATE AN I/O REQUEST PACKET EXTENSION
                        06DD    ; FOR LATER USE AS A FORK BLOCK.
                        06DD    ;
                        06DD    ; INPUTS:
                        06DD    ;
                        06DD    ;       R3 = I/O PACKET ADDRESS.
                        06DD    ;
                        06DD    ; OUTPUTS:
                        06DD    ;
                        06DD    ;       IRPE ALLOCATED FROM NON-PAGED POOL AND LINKED TO END
                        06DD    ;       OF I/O PACKET (IRP$L_EXTEND).  IF ALLOCATION FAILS, ANY
                        06DD    ;       PREVIOUSLY ALLOCATED IRPE IS DEALLOCATED AND THE
                        06DD    ;       PROCESS IS PUT IN RESOURCE WAIT STATE TO AWAIT NON-PAGED POOL
                        06DD    ;       AVAILABILITY.
                        06DD    ;--
                        06DD    ALLOC_IRPE:                                     ; ALLOCATE AN IRPE
             53    DD   06DD          PUSHL   R3                                ; SAVE REGISTER
   51   00C4 8F    3C   06DF          MOVZWL  #IRP$K_LENGTH,R1                  ; SET SIZE OF BLOCK
    00000000'GF    16   06E4          JSB     G^EXE$ALONONPAGED                         ; ALLOCATE BLOCK
             53  8ED0   06EA          POPL    R3                                ; RESTORE REGISTER
          26 50    E9   06ED          BLBC    R0,20$                            ; IF LBC FAILURE
     08 A2   51    B0   06F0          MOVW    R1,IRPE$W_SIZE(R2)                ; SET SIZE IN BLOCK
     0A A2   2C    90   06F4          MOVB    #DYN$C_IRPE,IRPE$B_TYPE(R2) ; SET BLOCK TYPE IN BLOCK
          54 A3    D0   06F8          MOVL    IRP$L_EXTEND(R3),-                ; SET NEXT IRPE ADDRESS IN BLOCK
          54 A2         06FB                  IRPE$L_EXTEND(R2)                 ;
             06    13   06FD          BEQL    10$                               ; IF EQL NONE
 2A A2   0800 8F    A8   06FF          BISW    #IRPE$M_EXTEND,IRPE$W_STS(R2) ; SET EXTENSION FLAG
                        0705    10$:
     54 A3   52    D0   0705          MOVL    R2,IRP$L_EXTEND(R3)               ; SET IRPE ADDRESS IN IRP
 2A A3   0800 8F    A8   0709          BISW    #IRP$M_EXTEND,IRP$W_STS(R3) ; SET EXTENSION FLAG
          2C A2    D4   070F          CLRL    IRPE$L_SVAPTE1(R2)                ; CLEAR SVAPTE SO I/O POST WILL
          38 A2    D4   0712          CLRL    IRPE$L_SVAPTE2(R2)                ;  JUST DEALLOCATE THE BLOCKS
             05         0715          RSB                                       ;
                        0716    20$:
        5E   04    C0   0716          ADDL    #4,SP                             ; REMOVE RETURN ADDRESS
   7E   0124 8F    3C   0719          MOVZWL  #SS$_INSFMEM,-(SP)                ; SET FAILURE STATUS
        51   03    3C   071E          MOVZWL  #RSN$_NPDYNMEM,R1                 ; SET RESOURCE TO AWAIT
           FD37    31   0721          BRW     RES_WAIT                          ; WAIT FOR NON-PAGED POOL
```

```
                             0724  1426          .SBTTL  DALLOC_IRPE - DEALLOCATE AN I/O REQUEST PACKET EXTENSION
                             0724  1427 ;++
                             0724  1428 ;
                             0724  1429 ; DALLOC_IRPE - SUBROUTINE TO DEALLOCATE AN I/O REQUEST PACKET EXTENSION
                             0724  1430 ;
                             0724  1431 ; INPUTS:
                             0724  1432 ;
                             0724  1433 ;        R3 = I/O REQUEST PACKET ADDRESS.
                             0724  1434 ;
                             0724  1435 ; OUTPUTS:
                             0724  1436 ;
                             0724  1437 ;        THE I/O REQUEST PACKET EXTENSION IS DEALLOCATED TO NON-PAGED
                             0724  1438 ;        POOL.
                             0724  1439 ;
                             0724  1440 ;        R1,R3,R5 ARE PRESERVED.
                             0724  1441 ;
                             0724  1442 ;--
                             0724  1443 DALLOC_IRPE:                              ; DEALLOCATE AN IRPE
          50    54 A3   D0   0724  1444          MOVL    IRP$L_EXTEND(R3),R0      ; GET IRPE ADDRESS
                   17   13   0728  1445          BEQL    20$                      ; BR IF NONE
             54 A0      D0   072A  1446          MOVL    IRPE$L_EXTEND(R0),-      ; REMOVE IRPE FROM LIST
             54 A3           072D  1447                  IRP$L_EXTEND(R3)         ;
                   06   12   072F  1448          BNEQ    10$                      ; IF NEQ NOT LAST IRPE
   2A A3    0800 8F     AA   0731  1449          BICW    #IRP$M_EXTEND,IRP$W_STS(R3) ; CLEAR EXTEND FLAG
                             0737  1450 10$:
                   0A   BB   0737  1451          PUSHR   #^M<R1,R3>               ; SAVE REGISTERS
        00000000'GF     16   0739  1452          JSB     G^EXE$DEANONPAGED        ; DEALLOCATE IRPE
                   0A   BA   073F  1453          POPR    #^M<R1,R3>               ; RESTORE REGISTERS
                             0741  1454 20$:
                        05   0741  1455          RSB                              ;
                             0742  1456 MB_END:
                             0742  1457          .END
```

```
$$$                             = 00000020 R     02        EXE$QIODRVPKT                     ********   X  03
$$OP                            = 00000002                 EXE$QIORETURN                     ********   X  03
ALLOC_BLOCK                       000002EC R     03        EXE$READCHK                       ********   X  03
ALLOC_FAIL                        00000436 R     03        EXE$WRITECHK                      ********   X  03
ALLOC_IRPE                        000006DD R     03        FDTEOF                            000002C1 R    03
AT$_MPM                         = 00000003                 FDTREAD                           000001EA R    03
CA$_MEASURE                     = 00000002                 FDTSET                            00000223 R    03
CAN$C_AMBXDGN                   = 00000002                 FDTWRITE                          000002D9 R    03
CAN$C_DASSGN                    = 00000001                 FINISHREAD                        000004D8 R    03
CANCELIO                          00000078 R     03        FKB$B_FIPL                      = 0000000B
CCB$B_STS                       = 00000008                 FKB$K_LENGTH                    = 00000018
CCB$V_RDCHKDON                  = 00000002                 FORMAT_PRQ                        00000693 R    03
CCB$V_WRTCHKDON                 = 00000003                 FUNCTABLE                         00000038 R    03
CHECKIO                           000001AF R     03        FUNCTAB_LEN                     = 00000040
CHECK_QUOTAS                      00000383 R     03        IDB$L_UCBLST                    = 00000018
COM$DELATTNAST                    ********   X   03        INT_EXIT                          0000059E R    03
COM$FLUSHATTNS                    ********   X   03        IO$V_NORSWAIT                   = 0000000A
COM$POST                          ********   X   03        IO$V_NOW                        = 00000006
COM$SETATTNAST                    ********   X   03        IO$V_READATTN                   = 00000007
CRB$L_INTD                      = 00000024                 IO$V_SETPROT                    = 00000009
CXB$L_LINK                      = 00000010                 IO$_READLBLK                    = 00000021
CXB$W_LENGTH                    = 0000000C                 IO$_READPBLK                    = 0000000C
DALLOC_BLOCKS                     00000475 R     03        IO$_READVBLK                    = 00000031
DALLOC_IRPE                       00000724 R     03        IO$_SETMODE                     = 00000023
DC$_MAILBOX                     = 000000A0                 IO$_VIRTUAL                     = 0000003F
DDB$L_DDT                       = 0000000C                 IO$_WRITELBLK                   = 00000020
DEV$M_AVL                       = 00040000                 IO$_WRITEOF                     = 00000028
DEV$M_IDV                       = 04000000                 IO$_WRITEPBLK                   = 0000000B
DEV$M_MBX                       = 00100000                 IO$_WRITEVBLK                   = 00000030
DEV$M_NNM                       = 00000200                 IOC$MNTVER                        ********   X  03
DEV$M_ODV                       = 08000000                 IOC$REQCOM                        ********   X  03
DEV$M_REC                       = 00000001                 IOC$RETURN                        ********   X  03
DEV$M_SHR                       = 00010000                 IPL$_ASTDEL                     = 00000002
DPT$C_LENGTH                    = 00000038                 IPL$_MAILBOX                    = 0000000B
DPT$C_VERSION                   = 00000004                 IPL$_SYNCH                      = 00000008
DPT$INITAB                        00000038 R     02        IRP$K_LENGTH                    = 000000C4
DPT$REINITAB                      0000006B R     02        IRP$L_BCNT                      = 00000032
DPT$TAB                           00000000 R     02        IRP$L_EXTEND                    = 00000054
DT$_SHRMBX                      = 00000002                 IRP$L_IOST1                     = 00000038
DYN$C_CRB                       = 00000005                 IRP$L_MEDIA                     = 00000038
DYN$C_DDB                       = 00000006                 IRP$L_PID                       = 0000000C
DYN$C_DPT                       = 0000001E                 IRP$L_SEQNUM                    = 00000050
DYN$C_IRPE                      = 0000002C                 IRP$L_SVAPTE                    = 0000002C
DYN$C_ORB                       = 00000049                 IRP$L_UCB                       = 0000001C
DYN$C_SHRBUFIO                  = 00000080                 IRP$M_CHAINED                   = 00000020
DYN$C_UCB                       = 00000010                 IRP$M_COMPLX                    = 00000008
ERROR                             000001E4 R     03        IRP$M_EXTEND                    = 00000800
EXE$ABORTIO                       ********   X   03        IRP$M_MBXIO                     = 00000400
EXE$ALONONPAGED                   ********   X   03        IRP$W_BCNT                      = 00000032
EXE$ALOSHARED                     ********   X   03        IRP$W_BOFF                      = 00000030
EXE$CHKRDACCES                    ********   X   03        IRP$W_CHAN                      = 00000028
EXE$CHKWRTACCES                   ********   X   03        IRP$W_FUNC                      = 00000020
EXE$DEANONPAGED                   ********   X   03        IRP$W_STS                       = 0000002A
EXE$DEASHARED                     ********   X   03        IRPE$B_TYPE                     = 0000000A
EXE$FINISHIOC                     ********   X   03        IRPE$L_EXTEND                   = 00000054
EXE$GL_LOCKRTRY                   ********   X   03        IRPE$L_MB_PARAM                   0000001C
EXE$IORSNWAIT                     ********   X   03        IRPE$L_MB_PORT                    00000020
```

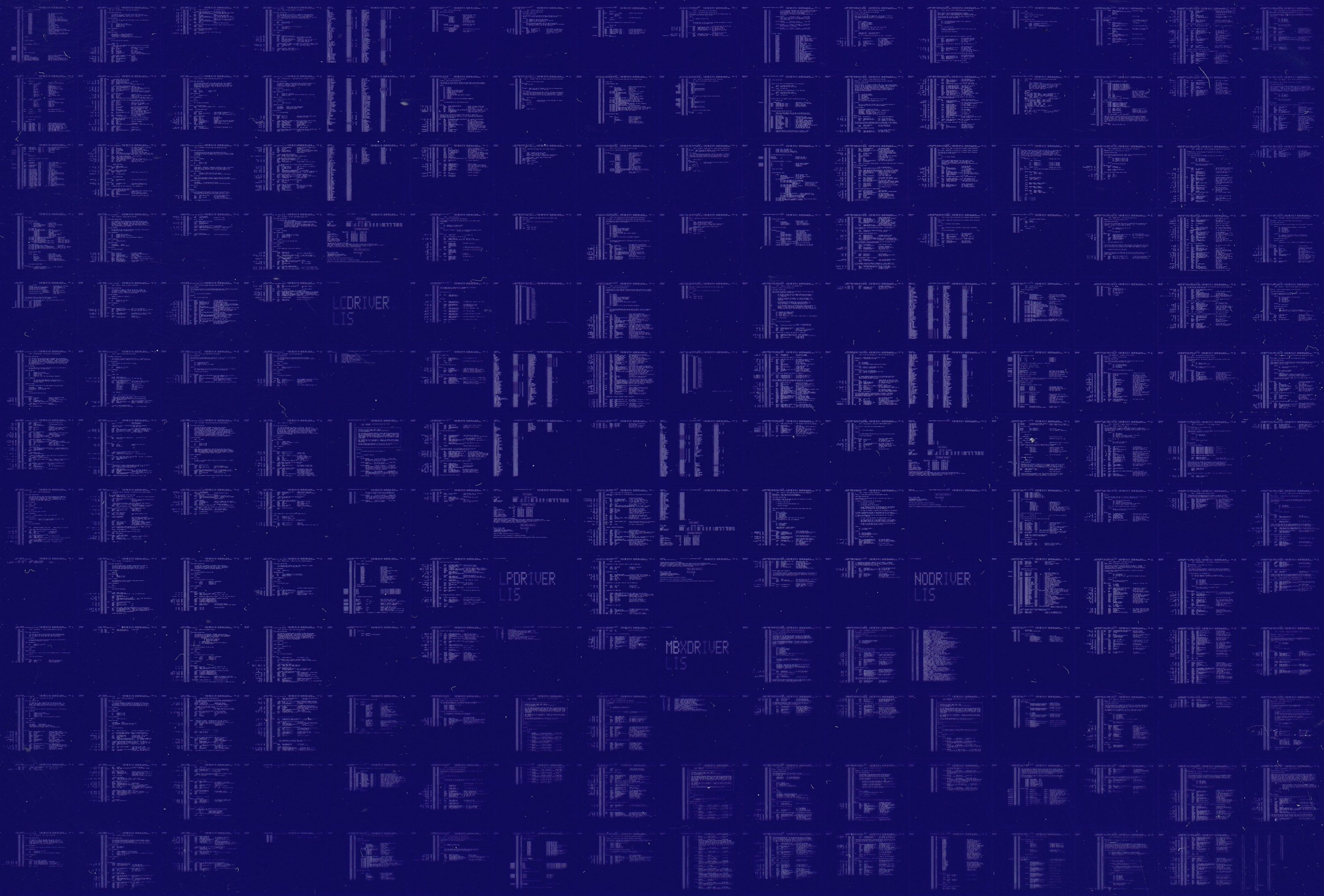
```
IRPE$L_SVAPTE1                = 0000002C                  PCB$L_EPID                    = 00000064
IRPE$L_SVAPTE2                = 00000038                  PCB$L_PHD                     = 0000006C
IRPE$M_EXTEND                 = 00000800                  PCB$L_PID                     = 00000060
IRPE$W_MB_PORTS                 00000018                  PCB$L_UIC                     = 000000BC
IRPE$W_MB_RQTYP                 0000001A                  PMS$GL_MBREADS                  ********    X   03
IRPE$W_SIZE                   = 00000008                  PMS$GL_MBWRITES                 ********    X   03
IRPE$W_STS                    = 0000002A                  PR$_IPL                       = 00000012
LNM$DELETE_LNMB                 ********    X   03        PRQ$C_MAILBOX                 = 00000001
LNM$LOCKW                       ********    X   03        PRQ$C_MINLENGTH               = 00000040
LNM$UNLOCK                      ********    X   03        PRQ$L_PARAM                   = 00000024
MA$RAVAIL                       ********    X   03        PRQ$W_DISPATCH                = 0000001C
MA$REQUEST                      ********    X   03        PRQ$W_REQTYPE                 = 00000020
MAILBOX_FULL                    00000443 R      03        PRQ$W_TO_PORT                 = 00000018
MASKH                         = 00000100                  PRQ$W_UNIT                    = 00000022
MASKL                         = 00000000                  PRQ_READ                      = 00000001
MBX$B_CREATPORT               = 00000009                  PRQ_READER                    = 00000003
MBX$B_FLAGS                   = 00000008                  PRQ_WRITE                     = 00000002
MBX$DDT                         00000000 RG     03        PRV$V_BYPASS                  = 0000001D
MBX$INT                         0000057D R      03        READCHECKIO                     0000018D R      03
MBX$M_VALID                   = 00000002                  READER_REQ                      0000059F R      03
MBX$Q_MSG                     = 00000000                  READ_REQ                        000005F4 R      03
MBX$V_QUOTALCK                = 00000003                  RESTORE_QUOTAS                  00000505 R      03
MBX$W_BUFFQUO                 = 00000018                  RES_WAIT                        0000045B R      03
MBX$W_MSGCNT                  = 00000016                  RSN$_MAILBOX                  = 00000002
MBX$W_PROT                    = 0000001A                  RSN$_NPDYNMEM                 = 00000003
MBX$W_READAST                 = 00000010                  SETUP_BLOCK                     00000332 R      03
MBX$W_READER                  = 0000000E                  SHB$L_DATAPAGE                = 00000004
MBX$W_REF                     = 0000000C                  SHB$L_REFCNT                  = 0000000C
MBX$W_WRITAST                 = 00000012                  SHD$B_FLAGS                   = 0000009F
MB_END                          00000742 R      03        SHD$B_PORTS                   = 0000009C
MSG_B_FUNC                      0000001C                  SHD$V_MBXLCK                  = 00000003
MSG_B_MESSAGE                   0000001D                  SHD$W_MBXQUOTA                = 0000005C
MSG_B_PORT                      0000000B                  SHD$W_RESWAIT                 = 000000A8
MSG_B_TYPE                      0000000A                  SHMRES_WAIT                     00000459 R      03
MSG_L_CHAINLINK                 00000010                  SS$_ABORT                     = 0000002C
MSG_L_IRPSEQ                    00000014                  SS$_BADQUEUEHDR               = 00000394
MSG_L_PID                       00000018                  SS$_BUFFEROVF                 = 00000601
MSG_L_POSTIOBUF                 00000000                  SS$_ENDOFFILE                 = 00000870
MSG_L_POSTUBUF                  00000004                  SS$_INSFMEM                   = 00000124
MSG_Q_MSGLINK                   00000000                  SS$_MBFULL                    = 000008D8
MSG_W_LENGTH                    0000000C                  SS$_MBTOOSML                  = 0000019C
MSG_W_MSGLENGTH                 0000000E                  SS$_NOPRIV                    = 00000024
MSG_W_SIZE                      00000008                  SS$_NORMAL                    = 00000001
NOTIFY                          00000652 R      03        STARTIO                         00000491 R      03
NOTIFY_DONE                     000006CE R      03        TOOSMALL                        000001DF R      03
NOTIFY_READ                     00000633 R      03        UCB$B_DEVCLASS                = 00000040
NOTIFY_READER                   00000625 R      03        UCB$B_DEVTYPE                 = 00000041
NOTIFY_WRITE                    00000642 R      03        UCB$B_DIPL                    = 0000005E
ORB$B_FLAGS                   = 0000000B                  UCB$B_FIPL                    = 0000000B
ORB$L_OWNER                   = 00000000                  UCB$K_MB_LENGTH               = 000000AC
ORB$M_NOACL                   = 00000008                  UCB$L_CRB                     = 00000024
ORB$M_PROT_16                 = 00000001                  UCB$L_DEVCHAR                 = 00000038
ORB$W_PROT                    = 00000018                  UCB$L_DEVCHAR2                = 0000003C
P1                            = 00000000                  UCB$L_DEVDEPEND               = 00000044
P2                            = 00000004                  UCB$L_IRP                     = 00000058
P3                            = 00000008                  UCB$L_LOGADR                  = 00000074
P4                            = 0000000C                  UCB$L_MB_MBX                  = 00000098
```

```
UCB$L_MB_PORT                = 000000A8
UCB$L_MB_RAST                = 00000094
UCB$L_MB_SHB                 = 0000009C
UCB$L_MB_WAST                = 00000090
UCB$L_MB_WIOQFL              = 000000A0
UCB$L_ORB                    = 0000001C
UCB$L_STS                    = 00000064
UCB$M_DELETEUCB              = 00010000
UCB$M_SHMMBX                 = 00000008
UCB$V_BSY                    = 00000008
UCB$V_DELMBX                 = 00000001
UCB$V_ONLINE                 = 00000004
UCB$W_DEVBUFSIZ              = 00000042
UCB$W_DEVSTS                 = 00000068
UCB$W_REFC                   = 0000005C
UCB$W_STS                    = 00000064
UCB$W_UNIT                   = 00000054
VEC$L_ADP                    = 00000014
VEC$L_IDB                    = 00000008
WRITE                          000002DC R     03
WRITECHECKIO                   0000019F R     03
WRITE_REQ                      000005B1 R     03
ZEROLENGTH                     000001D5 R     03

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

PSECT name                   Allocation          PSECT No.  Attributes
----------                   ----------          ---------  ----------
.  ABS  .                    00000000  (    0.)  00 (  0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                        00000024  (   36.)  01 (  1.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
$$$105_PROLOGUE              00000076  (  118.)  02 (  2.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
$$$115_DRIVER                00000742  ( 1858.)  03 (  3.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC LONG

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

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                    32    00:00:00.06    00:00:01.48
Command processing               119    00:00:00.37    00:00:03.65
Pass 1                           605    00:00:18.16    00:01:03.39
Symbol table sort                  0    00:00:02.64    00:00:12.33
Pass 2                           271    00:00:04.07    00:00:14.48
Symbol table output               29    00:00:00.16    00:00:00.29
Psect synopsis output              2    00:00:00.01    00:00:00.01
Cross-reference output             0    00:00:00.00    00:00:00.00
Assembler run totals            1060    00:00:25.47    00:01:35.63

The working set limit was 2100 pages.
153019 bytes (299 pages) of virtual memory were used to buffer the intermediate code.
There were 130 pages of symbol table space allocated to hold 2476 non-local and 78 local symbols.
1457 source lines were read in Pass 1, producing 21 object records in Pass 2.
55 pages of virtual memory were used to define 52 macros.
```

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

Macro library name                         Macros defined
------------------                         --------------
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                   37
_$255$DUA28:[SYSLIB]STARLET.MLB;2                10
TOTALS (all libraries)                           47

2794 GETS were required to define 47 macros.

There were no errors, warnings or information messages.

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

0112 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



LCDRIVER
LIS

LPDRIVER
LIS

NODRIVER
LIS

MBXDRIVER
LIS