```
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  EEEEEEEEEEE      RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III       VVV         VVV  EEEEEEEEEEE      RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III       VVV         VVV  EEEEEEEEEEE      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**DUTUMAC

```
DDDDDDDD    UU      UU  TTTTTTTTTT  UU      UU  MM      MM    AAAAAA      CCCCCCCC
DDDDDDDD    UU      UU  TTTTTTTTTT  UU      UU  MM      MM    AAAAAA      CCCCCCCC
DD      DD  UU      UU      TT      UU      UU  MMMM  MMMM  AA      AA  CC
DD      DD  UU      UU      TT      UU      UU  MMMM  MMMM  AA      AA  CC
DD      DD  UU      UU      TT      UU      UU  MM  MM  MM  AA      AA  CC
DD      DD  UU      UU      TT      UU      UU  MM  MM  MM  AA      AA  CC
DD      DD  UU      UU      TT      UU      UU  MM      MM  AA      AA  CC
DD      DD  UU      UU      TT      UU      UU  MM      MM  AA      AA  CC
DD      DD  UU      UU      TT      UU      UU  MM      MM  AAAAAAAAAA  CC
DD      DD  UU      UU      TT      UU      UU  MM      MM  AAAAAAAAAA  CC
DD      DD  UU      UU      TT      UU      UU  MM      MM  AA      AA  CC
DD      DD  UU      UU      TT      UU      UU  MM      MM  AA      AA  CC
DDDDDDDD    UUUUUUUUUU      TT      UUUUUUUUUU  MM      MM  AA      AA    CCCCCCCC
DDDDDDDD    UUUUUUUUUU      TT      UUUUUUUUUU  MM      MM  AA      AA    CCCCCCCC
```

```
MM      MM    AAAAAA    RRRRRRRR
MM      MM    AAAAAA    RRRRRRRR
MMMM  MMMM  AA      AA  RR      RR
MMMM  MMMM  AA      AA  RR      RR
MM  MM  MM  AA      AA  RR      RR
MM  MM  MM  AA      AA  RR      RR
MM      MM  AA      AA  RRRRRRRR
MM      MM  AA      AA  RRRRRRRR
MM      MM  AAAAAAAAAA  RR  RR
MM      MM  AAAAAAAAAA  RR  RR
MM      MM  AA      AA  RR    RR
MM      MM  AA      AA  RR    RR
MM      MM  AA      AA  RR      RR
MM      MM  AA      AA  RR      RR
```

```
        .TITLE  DUTUMAC DISK/TAPE CLASS DRIVER MACROS
        .IDENT  'V04-000'
```



```
;++
; FACILITY:
;
;       MSCP Disk and Tape Class Drivers
;
; ABSTRACT:
;
;       This module contains macros used by both the MSCP disk and tape
;       class drivers.
;
; ENVIRONMENT:
;
;       This module is used to build a macro library for both the DUDRIVER
;       and the TUDRIVER.
;
;--
;
; AUTHOR: Ralph O. Weber,         CREATION DATE: 25-AUG-1983
;
; MODIFIED BY:
;
;       V03-009 ROW0402         Ralph O. Weber          22-JUL-1984
;               Correct spelling of CDDB$A_2PFKB in ROW0383.
;
;       V03-008 ROW0383         Ralph O. Weber           6-JUL-1984
;               Add a fork block to the extended CDDB definition.  To save
;               space, the fork block is contained within an unused portion of
```

```
;               the DAP IRP/CDRP.  The fork block will be used to provide a
;               separate fork context for the altering of the standard I/O
;               database to reflect a device failover.
;
;       V03-007 LMP0235         L. Mark Pilant,         17-Apr-1984  14:30
;               Add a new macro, INIT_ORB, for initializing fields in the
;               template ORB as well as supplying the initialization table
;               for SYSGEN.
;
;       V03-006 ROW0339         Ralph O. Weber          8-APR-1984
;               Add invalid command processing macros; IVCMD_BEGIN, IF_IVCMD,
;               and IVCMD_END.
;
;       V03-005 ROW0338         Ralph O. Weber          7-APR-1984
;               Remove CDDB$W_SCSALCLS which is no longer used.  Fix
;               ACTION_ENTRY to accept an END_TABLE parameter.  Make
;               MSCP$K_ST_ an automatic prefix for the MSCP_CODE parameter to
;               ACTION_ENTRY.  Add DO_ACTION which does appropriate thing for
;               processing an action table.  Add IF_MSCP which tests MSCP
;               end-packet status for success or failure.  Change ACTION_ENTRY
;               and $DUTUDEF to use byte fields for ATE_MSCPCODE.  Saves one
;               byte per ACTION_ENTRY.
;
;       V03-004 ROW0286         Ralph O. Weber          22-JAN-1984
;               Add IFCANCEL and IFNOCANCEL macros.
;
;       V03-003 ROW0279         Ralph O. Weber          16-JAN-1984
;               Add the following macros:  ALT_REQCOM, ACTION_ENTRY,
;               INIT_MSCP_MSG, POST_IRP, POST_CDRP, and RESET_MSCP_MSG.
;               Remove unneeded symbol definitions from $DUTUDEF and add ATE
;               definitions to $DUTUDEF.  Adjust CDDB$W_SCSALCLS to account
;               for CDDB$W_RSVDW disappearing.
;
;       V03-002 ROW0261         Ralph O. Weber          22-NOV-1983
;               Add the following macros:
;                o CREATE_FORK - an unambiguous way of starting a fork thread
;                o $DUTUDEF - define the CDDB with multiple IRP/CDRP
;                   extensions which is what the class drivers really use and
;                   other constants used by both drivers
;                o INIT_UCB - initialize the driver template UCBs (both those
;                  generated by SYSGEN and the in-driver template UCB)
;                o SEND_MSCP_MSG - general purpose send a MSCP message
;
;       V03-001 ROW0235         Ralph O. Weber          6-OCT-1983
;               Change the WAIT_FOR_IODB macro to guarantee that the fork IPL
;               is IPL$_SCS before performing a FORKWAIT.
;
;--
```

```
;++
; ALT_REQCOM
;
; Functional description:
;
;       The macro invokes request completion for class driver requests.
;
; Parameters: None.
;
; Inputs:
;
;       R0      first longword of I/O status
;       R1      second longword of I/O status
;       R5      CDRP address
;
; Outputs:
;
;       Control does not return from this macro.
;--
        .MACRO  ALT_REQCOM
        JMP     G^IOC$ALTREQCOM
        .ENDM   ALT_REQCOM
```

```
;++
; ACTION_ENTRY
;
;       This macro produces entries in an action table used by the
;       INTERPRET_ACTION_TABLE routine.  Either a single entry or and
;       end-of-table marker is produced.  If no parameters are given or the
;       parameter is identically equal to END, the end-of-table marker is
;       produced.  Otherwise, an action entry is produced.
;
; Parameters:
;
;   mscp_code           a MSCP end status value which when matched causes the
;                       remainder of the action table entry to become valid
;                       and processed, or END_TABLE.  MSCP$K_ST_'mscp_code' is
;                       used as the actual MSCP status code to test.
;   ss_code             the system status code to be returned when mscp_code
;                       is matched.
;   action_dispatch     the driver location to which control is transfered
;                       when mscp_code is matched.
;--
        .MACRO  ACTION_ENTRY mscp_code=END_TABLE, ss_code, action_dispatch
        .IF     DIFFERENT mscp_code, END_TABLE
        ASSUME  ATE_OFFSET EQ 0
        .WORD   action_dispatch - .
        .BYTE   MSCP$K_ST_'mscp_code'
        .WORD   ss_code
        .IF_FALSE
        .WORD   0
        .ENDC
        .ENDM   ACTION_ENTRY
```

```
;++
; CREATE_FORK
;
; Functional description:
;
;       This macro initiates a fork thead and then returns control to the
;       instruction following the macro invocation.
;
; Parameters:
;
;       address address of the first instruction to be executed in the new
;               fork thread
;       frkblk  address of the new fork block
;       fr3     new fork R3 value
;       fr4     new fork R4 value
;       mask    register save mask for registers to save across fork process
;                creation
;--

        .MACRO  CREATE_FORK address, frkblk=(R5), fr3=R3, fr4=R4, -
                            mask=<^M<R0,R1,R2,R3,R4,R5>>
                                                ; Save registers.
        .IIF    DIFFERENT mask, ^M<>, PUSHR #mask
                                                ; Setup new fork registers.
        .IIF    DIFFERENT frkblk, (R5), MOVAL frkblk, R5
        .IIF    DIFFERENT fr3, R3, MOVL fr3, R3
        .IIF    DIFFERENT fr4, R4, MOVL fr4, R4
        BSBW    address
                                                ; Restore registers.
        .IIF    DIFFERENT mask, ^M<>, POPR #mask
        .ENDM   CREATE_FORK
```

```
;++
; DO_ACTION
;
; Functional description:
;
;       This macro initiates processing of an action table.  The action table
;       immediately follows the DO_ACTION macro, is composed of ACTION_ENTRY
;       statements, and is terminated by ACTION_ENTRY END.  If the MSCP end
;       status in the packet pointed to by R2 has a match in one of the
;       ACTION_ENTRY statements, the status in that ACTION_ENTRY is placed in
;       R0 and control is transfered to the label mentioned in the ACTION_ENTRY
;       statement.  Otherwise, control is returned to the first instruction
;       following the ACTION_ENTRY END statement and the contents or R0 are
;       indeterminant.
;
; Parameters:
;
;       type    TRANSFER    ==> transfer command processing, return status in
;                               high-order R0.
;               NONTRANSFER ==> non-transfer command processing, return status
;                               in low-order R0.
;
; Inputs:
;
;       R2      MSCP end-packet address
;
; Outputs:
;
;       R0      SS$_xxx status
;       R1      Corrupted
;
;       All other registers are preserved.
;--

        .MACRO  DO_ACTION type=NONTRANSFER
        .IF     IDENTICAL type, TRANSFER
        BSBW    DUTU$INTR_ACTION_XFER
        .MEXIT
        .ENDC
        .IF     IDENTICAL type, NONTRANSFER
        BSBW    DUTU$INTR_ACTION_N
        .MEXIT
        .ENDC
        .ERROR  ;Invalid DO_ACTION parameter
        .ENDM   DO_ACTION
```

```
;++
; $DUTUDEF
;
; Functional description:
;
;       This macro defines the complete class driver CDDB, including the
;       several IRP/CDRP extensions which the class drivers append to the
;       CDDB.  Offsets into the $$$220_DUTU_DATA_01 .PSECT and other constants
;       common to both class drivers are also defined.
;
; Parameters:
;
;       GLOBAL - typical for $xxxDEF macros
;
;--

        .MACRO  $DUTUDEF gbl
        $CDDBDEF
        $CDRPDEF
        $FKBDEF
        $IRPDEF
        $DEFINI DUTU gbl
;
; Disk and Tape Class Driver CDDB Definitions
;

                        .BLKB   CDDB$K_LENGTH           ; Basic CDDB
$DEF    CDDB$A_PRMIRP   .BLKB   IRP$K_LENGTH            ; Permanent IRP/CDRP
$DEF    CDDB$A_DAPIRP   .BLKB   IRP$K_LENGTH            ; DAP IRP/CDRP
$EQU    CDDB$K_DUTULENGTH .
;
; Additional CDDB fields which actually occur in the permanent IRP/CDRP
;
        . = CDDB$A_PRMIRP + IRP$L_UCB
$DEF    CDDB$L_PRMUCB   .BLKL   1                       ; UCB in perm. IRP/CDRP

        . = CDDB$A_PRMIRP + IRP$L_ABCNT
$DEF    CDDB$L_CANCLQFL .BLKL   1                       ; Cancel queue listhead
$DEF    CDDB$L_CANCLQBL .BLKL   1

        . = CDDB$A_PRMIRP + IRP$L_FQFL
$DEF    CDDB$A_PRMCDRP  .BLKB   1                       ; Permanent CDRP

        . = CDDB$A_PRMIRP + IRP$L_CDT
$DEF    CDDB$L_CDT      .BLKL   1                       ; CDT address

;
; Additional CDDB fields which actually occur in the DAP IRP/CDRP
;
        . = CDDB$A_DAPIRP + IRP$L_UCB
$DEF    CDDB$L_DAPUCB   .BLKL   1                       ; UCB in DAP IRP/CDRP

        . = CDDB$A_DAPIRP + IRP$L_FQFL
```

```
$DEF    CDDB$A_DAPCDRP  .BLKB    1                       ; DAP CDRP

        . = CDDB$A_DAPIRP + IRP$L_CDT
$DEF    CDDB$L_DAPCDT   .BLKL    1                       ; DAP CDRP CDT address

;
; The following fork block is used to provide an independent fork context
; in which the standard VMS I/O database can be updated to reflect a device
; failover.  The fork block overlays the last several longwords in the IRP
; portion of the DAP IRP/CDRP pair.  These longwords are unused, we assume.
; This overlay is done to save non-paged pool and to use previously unused
; space in the CDDB.
;

        ASSUME  <IRP$L_MEDIA + 4> LE IRP$L_ABCNT
        ASSUME  <IRP$L_ABCNT + FKB$K_LENGTH> LE IRP$L_FQFL
        . = CDDB$A_DAPIRP + IRP$L_ABCNT
$DEF    CDDB$A_2PFKB    .BLKL   FKB$K_LENGTH

;
; $$$220_DUTU_DATA_01 Offsets Definitions
;

.=0

$DEF    DUTU$L_CDDB_LISTHEAD    .BLKL 1                 ; Address of CDDB
                                                        ; listhead for this
                                                        ; device class
$EQU    DUTU$K_DATA_LENGTH      .                       ; Size of .PSECT

;
; Action table entry offsets
;

.=0

$DEF    ATE_OFFSET      .BLKW   1       ; Offset of action routine.
$DEF    ATE_MSCPCODE    .BLKB   1       ; Value of major MSCP status code.
$DEF    ATE_SSCODE      .BLKW   1       ; Value of corresponding SS$_ code.
$EQU    ATE_ENTRY_LEN   .               ; Length of action entry.


        $DEFEND DUTU gbl
        .ENDM   $DUTUDEF
;++
; IF_MSCP
;
; Functional description:
;
;       This macro tests an MSCP end-packet status for success or failure and
;       branches accordingly.  Optionally, the end-packet status can be stored
;       (zero-extended) in a longword.
;
; Parameters:
;
```

```
;       type    SUCCESS ==> branch if MSCP success
;               FAILURE ==> branch if MSCP failure
;       then    branch destination
;       status  optional longword to receive zero-extended MSCP status
;
; Inputs:
;
;       R2      MSCP end-packet address
;
; Outputs:
;
;       All registers preserved.
;--

        .MACRO  IF_MSCP type=SUCCESS, then, status
        .IF     NOT_BLANK status
        EXTZV   #MSCP$V_ST_MASK, #MSCP$S_ST_MASK, -
                MSCP$W_STATUS(R2), 'status'
        .IF_FALSE
        ASSUME  MSCP$V_ST_MASK EQ 0
        ASSUME  MSCP$S_ST_MASK LE 7
        BITB    #MSCP$M_ST_MASK, MSCP$W_STATUS(R2)
        .ENDC
        ASSUME  MSCP$K_ST_SUCC EQ 0
        .IF     IDENTICAL type, SUCCESS
        BEQL    'then'
        .MEXIT
        .ENDC
        .IF     IDENTICAL type, FAILURE
        BNEQ    'then'
        .MEXIT
        .ENDC
        .ERROR  ;Invalid IF_MSCP parameter
        .ENDM   IF_MSCP
```

```
;++
; IFCANCEL and IFNOCANCEL
;
; Functional description:
;
;       These macros test whether or not a given IRP or CDRP is to be
;       canceled and act accordingly.  IFCANCEL branches to the specified
;       label if the IRP/CDRP should be canceled.  IFNOCANCEL branches to the
;       specified label if the IRP/CDRP should not be canceled.
;
; Parameters:
;
;   irp         address of an IRP to be tested (exclusive with cdrp parameter)
;   cdrp        address of an CDRP to be tested (exclusive with irp parameter)
;   then        destination label
;
; Inputs:
;
;   R5          address of a cancel CDRP describing the cancel request
;
; Outputs:
;
;   R0 is destroyed.
;   All other registers are preserved.
;--
        .MACRO  TEST_IRP irp
        PUSHL   R2
        MOVAB   IRP$L_FQFL'irp', R2
        BSBW    DUTU$TEST_CANCEL_CDRP
        POPL    R2
        .ENDM   TEST_IRP

        .MACRO  TEST_CDRP cdrp
        .IF     DIFFERENT cdrp, (R2)
        PUSHL   R2
        MOVAB   cdrp, R2
        .ENDC
        BSBW    DUTU$TEST_CANCEL_CDRP
        .IIF    DIFFERENT cdrp, (R2), POPL R2
        .ENDM   TEST_CDRP

        .MACRO  IFCANCEL irp, cdrp, then
        .IF     NOT_BLANK irp
        TEST_IRP irp
        BLBS    R0, then
        .IFF
        TEST_CDRP cdrp
        BLBS    R0, then
        .ENDC
        .ENDM   IFCANCEL

        .MACRO  IFNOCANCEL irp, cdrp, then
        .IF     NOT_BLANK irp
        TEST_IRP irp
        BLBC    R0, then
        .IFF
```

```
        TEST_CDRP cdrp
        BLBC    R0, then
        .ENDC
        .ENDM   IFNOCANCEL
```

```
;++
; INIT_MSCP_MSG
;
; Functional description:
;
;       This macro causes a MSCP command packet to be initialized.
;
; Parameters:
;
;    ucb        address of a UCB whose UCB$W_MSCPUNIT gives the unit number
;               for the command.  If this parameter is not present, no unit
;               number is placed in the command packet.
;
; Inputs:
;
;    R5         CDRP address
;
; Outputs:
;
;    R0 & R1    destroyed
;    R2         MSCP command packet address
;    All other registers preserved.
;--
        .MACRO  INIT_MSCP_MSG ucb
        .IF     BLANK ucb
        BSBW    DUTU$INIT_MSCP_MSG
        .IFF
        .IF     DIFFERENT ucb (R3)
        PUSHL   R3
        MOVAL   ucb, R3
        .ENDC
        BSBW    DUTU$INIT_MSCP_MSG_UNIT
        .IF     DIFFERENT ucb (R3)
        POPL    R3
        .ENDC
        .ENDC
        .ENDM   INIT_MSCP_MSG
```

```
;++
; INIT_UCB
;
; Functional description:
;
;       This macro makes UCB initialization entries in the DPT the template UCB.
;
; Parameters:
;
;       offset  field offset name (less the UCB$) for field being initialized
;               -- if offset is blank, space for the total UCB whose size --
;               -- is given by the size parameter is allocated.           --
;       size    field size: BYTE WORD LONG QUAD (default = LONG)
;       value   initialized field value
;--
        .MACRO  INIT_UCB        offset, size=LONG, value
        .SAVE
        .IF     BLANK offset
        .PSECT  $$$200_TEMPLATE_UCB_01 LONG,RD,WRT,EXE
        .BLKB   size
        .RESTORE
        .IF_FALSE
        PSECT   $$$200_TEMPLATE_UCB_01 LONG,RD,WRT,EXE
        . = UCB$'offset'
        .'size' 'value'
        .RESTORE
        .IF     GT <UCB$'offs-    - UCB$B_TYPE>
        DPT_STORE UCB,UCB$'o  ·  c',%extract(0,1,size),'value'
        .ENDC
        .ENDM   INIT_UCB
;++
; INIT_ORB
;
; Functional description:
;
;       This macro makes ORB initialization entries in the DPT the template ORB.
;
; Parameters:
;
;       offset  field offset name (less the ORB$) for field being initialized
;               -- if offset is blank, space for the total ORB whose size --
;               -- is given by the size parameter is allocated.           --
;       size    field size: BYTE WORD LONG QUAD (default = LONG)
;       value   initialized field value
;--
        .MACRO  INIT_ORB        offset, size=LONG, value
        .SAVE
        .IF     BLANK offset
        .PSECT  $$$200_TEMPLATE_ORB_01 LONG,RD,WRT,EXE
        .BLKB   size
        .RESTORE
        .IF_FALSE
        .PSECT  $$$200_TEMPLATE_ORB_01 LONG,RD,WRT,EXE
        . = ORB$'offset'
        .'size' 'value'
```

```
        .RESTORE
        .IF     GT <ORB$'offset' - ORB$B_TYPE>
        DPT_STORE ORB,ORB$'offset',%extract(0,1,size),'value'
        .ENDC
        .ENDC
        .ENDM   INIT_ORB
;++
; Invalid Command Processing Macros
;
;       The following three macros relate to processing of MSCP requests which
;       produce an MSCP "invalid command" end status.  IVCMD_BEGIN and
;       IVCMD_END define the beginning and end of a co-routine thread in which
;       a duplicate of the MSCP command which caused the "invalid command"
;       error is produced.  IF_IVCMD branches when executed within such a
;       co-routine thread.  Further details of these macros and of invalid
;       command processing in general can be found in DUTUSUBS.
;--

; Begin invalid command co-routine thread

        .MACRO  IVCMD_BEGIN
        BSBW    DUTU$LOG_IVCMD
        .ENDM   IVCMD_BEGIN

; Branch if in an invalid command co-routine thread

        .MACRO  IF_IVCMD then
        ASSUME  CDRP$V_IVCMD EQ 8
        BLBS    CDRP$L_DUTUFLAGS+1(R5), then
        .ENDM   IF_IVCMD

; End invalid command co-routine thread

        .MACRO  IVCMD_END
        JSB     @(SP)+
        .ENDM   IVCMD_END
```

```
;++
; MEDIA
;
; Functional description:
;
;       This macro produces one entry in the MSCP media identifier to VMS
;       device type conversion table.
;
; Parameters:
;
;       dd      the two character prefered device controller name ( the DD
;               part of DDCn )
;       devnam  the hardware device name ( e.g. RA81 )
;       dtname  if DT$_'devnam' is not a legal VMS device type, this parameter
;               gives the correct VMS device type for the device ( should be
;               used only when DT$_'devnam' is not correct )
;--
        .MACRO  MEDIA dd, devnam, dtname
        .SAVE
        .PSECT  $$$220_DEVTYPE_TABLE_01 RD,WRT,EXE,BYTE
$$BEGIN$$=-1
$$MEDIA$$=0
$$S$$=27
        .IRPC   $$L$$,<dd>
        $$TEMP$$ = ^a/$$L$$/ - ^x40
        .IF     GT $$TEMP$$
        $$MEDIA$$ = $$MEDIA$$ + <$$TEMP$$ @ $$S$$>
        .ENDC
        $$S$$ = $$S$$ - 5
        .ENDR
        .IRPC   $$L$$,<devnam>
        .IF     GE <$$S$$ - 7>
        $$TEMP$$ = ^a/$$L$$/ - ^x40
        .IF     GT $$TEMP$$
        $$MEDIA$$ = $$MEDIA$$ + <$$TEMP$$ @ $$S$$>
        .IF_FALSE
        .IIF    LT $$BEGIN$$, $$BEGIN$$ = <17-$$S$$>/5
        .ENDC
        $$S$$ = $$S$$ - 5
        .ENDC
        .ENDR
        .IIF    LT $$BEGIN$$, $$BEGIN$$ = 3
        $$N$$ = %extract( $$BEGIN$$, 3, devnam )
        $$MEDIA$$ = $$MEDIA$$ + $$N$$
        .NOSHOW CND
        .SHOW   ME
                .LONG   $$MEDIA$$               ; Store media-id
        .IF     BLANK dtname
                .BYTE   DT$_'devnam'            ; Store VMS device type
        .IF_FALSE                               ; -- OR --
                .BYTE   dtname                  ; Store explicit VMS type
        .ENDC
        .NOSHOW ME                              ; Filler to make listing look
        .SHOW   CND                             ; good
        .SHOW   ME
        .NOSHOW ME
```

```
        .RESTORE
        .ENDM
```

```
;++
; PERMCDRP_TO_CDDB
;
; Functional description:
;
;       This macro converts an input CDRP address (belonging to one of the
;       CDRPs permanently associated with a CDDB) to the address of the
;       related CDDB.
;
; Inputs:
;
;       CDRP    a register containing the base address of a CDRP permanently
;               related to a CDDB
;
; Implicit inputs:
;
;       CDRP$W_CDRPSIZE an offset from the CDRP base to the CDDB base
;
; Outputs:
;
;       CDDB    a register into which the base address of the CDDB related to
;               the input CDRP is stored (may not be the same register as CDRP)
;
; Implicit outputs:
;
;       All registers except CDDB are preserved.
;--

        .MACRO  PERMCDRP_TO_CDDB, cdrp, cddb
        .IF     IDENTICAL cdrp, cddb
        .ERROR  ; Identical register arguments to PREMCDRP_TO_CDDB
        .ENDC
        CVTWL   CDRP$W_CDRPSIZE(cdrp), cddb     ; Get CDRP to CDDB offset.
        ADDL    cdrp, cddb                      ; Form base address of CDDB.
        .ENDM   PERMCDRP_TO_CDDB
```

```
;++
; POST_IRP
;
; Functional description:
;
;       An IRP is queued for I/O post processing.  All resources held by the
;       IRP are appropriately cleaned up.
;
; Parameters:
;
;    status     final status for IRP
;
; Inputs:
;
;       R0      IRP address
;
; Outputs:
;
;       R0      destroyed
;       all other registers preserved
;--
        .MACRO  POST_IRP status
        PUSHL   R1
        MOVZWL  #status,R1
        MOVAB   IRP$L_FQFL(R0),R0
        BSBW    DUTU$POST_CDRP
        POPL    R1
        .ENDM   POST_IRP
```

```
;++
; POST_CDRP
;
; Functional description:
;
;       An CDRP is queued for I/O post processing.  All resources held by the
;       CDRP are appropriately cleaned up.
;
; Parameters:
;
;    status     final status for CDRP
;
; Inputs:
;
;       R0      CDRP address
;
; Outputs:
;
;       R0      destroyed
;       all other registers preserved
;--
        .MACRO  POST_CDRP status
        PUSHL   R1
        MOVZWL  #status,R1
        BSBW    DUTU$POST_CDRP
        POPL    R1
        .ENDM   POST_CDRP
```

```
;++
; RESET_MSCP_MSG
;
; Functional description:
;
;       This macro causes a previously used MSCP command packet (or end
;       message) to be completely recycled and readied for use in the sending
;       of another MSCP command.  It is intended for use by those functions
;       which require more than one MSCP command to properly complete.
;
; Parameters:   None.
;
; Inputs:
;
;    R3         UCB address
;    R5         CDRP address
;
; Outputs:
;
;    R0 & R1    destroyed
;    R2         reset MSCP command packet address
;    All other registers preserved.
;--
        .MACRO  RESET_MSCP_MSG
        BSBW    DUTU$RESET_MSCP_MSG
        .ENDM   RESET_MSCP_MSG
```

```
;++
; SEND_MSCP_MSG
;
; Functional description:
;
;       This macro causes the MSCP message packet pointed to by the CDRP whose
;       address is stored in R5 to be transmitted to the MSCP server at the
;       other end of the connection whose PDT address is in R4.  For
;       accounting purposes, the CDRP is queued to the active transfers queue
;       of the CDDB whose address is in UCB$L_CDDB(R3).
;
;       Control is returned to the instruction following this macro when the
;       MSCP end message has been received.
;
;       The macro accepts one parameter which is one of:
;               <blank> - standard class driver send message request
;                         (DUTU$SEND_MSCP_MSG is called)
;               INLINE  - fast execution send message request
;                         (send message code is generated inline; except for
;                         handling special cases, when control may still be
;                         transfered to DUTU$SEND_MSCP_MSG)
;               DRIVER  - internal class driver send message request
;                         (DUTU$SEND_DRIVER_MSG is called)
;               ROUTINE - build the code portion of the DUTU$SEND_MSCP_MSG
;                         routine
;
;  Implicit inputs:
;
;       R3 UCB address
;       R4 PDT address
;       R5 CDRP address
;--
        .MACRO  SEND_MSCP_MSG, type, ?send, ?dbuf, ?main, ?exit

        .IF     BLANK type                      ; The "normal" action is
        BSBW    DUTU$SEND_MSCP_MSG              ;  to call the common routine.
        .IF_FALSE
        .IF     IDENTICAL type, DRIVER
        BSBW    DUTU$SEND_DRIVER_MSG            ; Send internal driver msg.
        .IF_FALSE                               ; Otherwise, generate code.
        .IIF    IDENTICAL type, ROUTINE, .SHOW MEB
                                                ; Was a diagnostic buffer specified?
        BITW    #IRP$M_DIAGBUF, CDRP$W_STS(R5)
        BNEQ    dbuf                            ; Branch if diagnostic buffer present.
main:   MOVL    UCB$L_CDDB(R3), R1              ; Get CDDB of intelligent controller.
send:   INSQUE  (R5),@CDDB$L_CDRPQBL(R1)        ; Insert CDRP onto tail of queue
                                                ;  of CDRP's sent to the port.
        MOVL    #MSCP$K_MXCMDLEN,R1             ; Pass length of longest MSCP command.
        .IF     IDENTICAL type, INLINE
        PUSHAB B^exit                           ; INLINE needs return address.
        .ENDC
        JMP     @PDT$L_SNDCNTMSG(R4)            ; Jump to PORT routine

dbuf:   BSBW    DUTU$DUMP_COMMAND               ; If diagnostic buffer present, record
                                                ;  MSCP command message sent in it.
        BRB     main                            ; Then rejoin normal code path.
```

```
        .IIF    IDENTICAL type, ROUTINE, .NOSHOW MEB
exit:
        .ENDC
        .ENDC
        .ENDM   SEND_MSCP_MSG
```

```
;++
; WAIT_FOR_IODB
;
; Functional description:
;
;       This macro completes when the I/O database is available for write
;       access.  If "immediate" is specified, control is return there if and
;       only if no fork was required to gain access to the mutex.
;
; Inputs:
;
;       R3      fork saved register
;       R4      fork saved register
;       R5      address of a fork block
;
; Implicit inputs:
;
;       IOC$GL_MUTEX address of the I/O data base mutex
;
; Outputs:
;
;       R3      fork saved register
;       R4      fork saved register
;       R5      address of a fork block
;
;               N.B. UCB$B_FIPL is used as the offset to the fork IPL field
;               in the fork block.  This is used as a convenient -- always
;               defined in a driver -- offset to the fork IPL.  There is no
;               requirement that the fork block pointed to by R5 be a UCB.
;
; Implicit outputs:
;
;       All registers except R3 - R5 destroyed.
;
;       If immediate exit taken, all registers preserved.
;--
        .MACRO  WAIT_FOR_IODB, immediate, ?loop, ?done
        CMPW    #-1, G^<IOC$GL_MUTEX+MTX$W_OWNCNT> ; Is I/O mutex owned?
        .IF     BLANK, immediate
        BEQL    done
        .IFF
        BEQL    immediate
        .ENDC
        MOVB    #IPL$_SCS, UCB$B_FIPL(R5)       ; Guarantee fork IPL.
loop:   FORK_WAIT
        CMPW    #-1, G^<IOC$GL_MUTEX+MTX$W_OWNCNT> ; Is I/O mutex owned?
        BNEQ    loop
done:
        .ENDM   WAIT_FOR_IODB
```

XWDRIVER
MAP

RTTDRIVER
MAP

XADRIVER
MAP

TMDRIVER
MAP

XFDRIVER
MAP

TUDRIVER
MAP

XIDRIVER
MAP

MBXDRIVER
MAP

PUDRIVER
MAP

PADRIVER
MAP

XEDRIVER
MAP

XQDRIVER
MAP

DDCMPDEF
MDL

XGDEF
MDL

TPDRIVER
MAP

XDDRIVER
MAP

TSDRIVER
MAP

XGDRIVER
MAP

XMDRIVER
MAP

DUTUMAC
MAR

NODRIVER
MAP

XADRIVER
MAR

CONINTERR
LIS

XIDRIVER
MAR

PAMAC
MAR

CNDRIVER
LIS