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

FILEID**MSCPMOUNT

```
MM      MM    SSSSSSSS    CCCCCCCC  PPPPPPPP    MM      MM    000000    UU      UU  NN      NN  TTTTTTTTTT
MM      MM    SSSSSSSS    CCCCCCCC  PPPPPPPP    MM      MM    000000    UU      UU  NN      NN  TTTTTTTTTT
MMMM  MMMM  SS          CC          PP      PP  MMMM  MMMM  OO      OO  UU      UU  NN      NN      TT
MMMM  MMMM  SS          CC          PP      PP  MMMM  MMMM  OO      OO  UU      UU  NN      NN      TT
MM  MM  MM  SS          CC          PP      PP  MM  MM  MM  OO      OO  UU      UU  NNNN    NN      TT
MM  MM  MM  SS          CC          PP      PP  MM  MM  MM  OO      OO  UU      UU  NNNN    NN      TT
MM      MM    SSSSSS    CC          PPPPPPPP    MM      MM  OO      OO  UU      UU  NN  NN  NN      TT
MM      MM    SSSSSS    CC          PPPPPPPP    MM      MM  OO      OO  UU      UU  NN  NN  NN      TT
MM      MM          SS  CC          PP          MM      MM  OO      OO  UU      UU  NN    NNNN      TT
MM      MM          SS  CC          PP          MM      MM  OO      OO  UU      UU  NN    NNNN      TT
MM      MM          SS  CC          PP          MM      MM  OO      OO  UU      UU  NN      NN      TT
MM      MM          SS  CC          PP          MM      MM  OO      OO  UU      UU  NN      NN      TT
MM      MM  SSSSSSSS    CCCCCCCC  PP          MM      MM    OOOOOO    UUUUUUUUUU  NN      NN      TT
MM      MM  SSSSSSSS    CCCCCCCC  PP          MM      MM    000000    UUUUUUUUUU  NN      NN      TT
```

```
  CCCCCCCC    000000    MM      MM
  CCCCCCCC    000000    MM      MM
CC          OO      OO  MMMM  MMMM
CC          OO      OO  MMMM  MMMM
CC          OO      OO  MM  MM  MM
CC          OO      OO  MM  MM  MM
CC          OO      OO  MM      MM
CC          OO      OO  MM      MM
CC          OO      OO  MM      MM
CC          OO      OO  MM      MM
CC          OO      OO  MM      MM
CC          OO      OO  MM      MM
  CCCCCCCC    000000    MM      MM
  CCCCCCCC    000000    MM      MM
```

```
$ VERIFY='F$VERIFY(0)'
$ !++
$ !     This procedure mounts all disks in the cluster, and runs as a detached
$ !     process (or interactively) on all nodes in the cluster.  It may
$ !     be invoked in three ways:
$ !
$ !     1) To mount all the disks and then exit:
$ !
$ !             @MSCPMOUNT ONCE
$ !
$ !     2) To mount all the disks, then start a detached process to check
$ !     every 15 minutes for any new disks to mount.  This should be
$ !     used in an environment which uses many mscp-served disks.
$ !
$ !             @MSCPMOUNT
$ !
$ !
$ !     3) To only mount a single disk.  Typically this would be done
$ !     interactively.
$ !
$ !             @MSCPMOUNT MOUNT alldevnam label
$ !
$ !                     (for instance, @MSCPMOUNT MOUNT $255$DUA1 WORK01
$ !
$ !     Normally, you would probably use method 1) or 2) in your
$ !     common SYSTARTUP file to mount all the disks at system startup.
$ !     If you have mscp-served disks in your configuration, you must
$ !     SET DEVICE/SERVE [/DUAL] them before invoking this procedure.
$ !     In addition, your SYSTARTUP file should also wait for at least
$ !     one disk per HSC or dual-ported pair of HSC's to be configured
$ !     using a loop similar to the following:
$ !
$ !             $ !
$ !             $ ! Use the allocation-class name for the device
$ !             $ !
$ !             $ WAITFOR_DEV = "$255$DUA5:"    !Name of device to wait for
$ !             $ WAITFOR_LABEL = "VMSDOCLIB"
$ !             $ SET VERIFY                    !Show what happens on console
$ !             $ !
$ !             $ ! Enable us to talk to the HSC-50 named HSCA
$ !             $ !
$ !             $ HSCA1:
$ !             $ IF F$GETDVI(WAITFOR_DEV, "EXISTS") THEN GOTO HSCA2
$ !             $ WRITE SYS$OUTPUT "Waiting for ''WAITFOR_LABEL' (''WAITFOR_DEV')"
$ !             $HSCA_RETRY:
$ !             $ WAIT  0:0:30
$ !             $ GOTO HSCA1
$ !             $ !
$ !             $ ! Device exists, try to mount it.  If we cannot mount it
$ !             $ ! then go try again.  This is only necessary if the disk
$ !             $ ! is required for normal system operations.
$ !             $ !
$ !             $ HSCA2:
$ !             $ MOUNT /SYSTEM 'WAITFOR_DEV' 'WAITFOR_LABEL'
$ !             $ IF .NOT. $STATUS THEN GOTO HSCA_RETRY
$ !             $ SET NOVERIFY
```

```
$ !
$ !--
$
$ SET NOON
$ WAIT_INTERVAL = "00:15:00"              ! Time to wait if detached process
$ PROCEDURE_NAME = F$ELEMENT(0,";",F$ENVIRONMENT("PROCEDURE"))
$ RECURSIVE_CALLBACK = "@" + PROCEDURE_NAME
$ !
$ ! If this is the callback to mount a disk, then go mount the disk.
$ !
$ IF P1 .EQS. "MOUNT" THEN GOTO MOUNT_DEVICE
$ WRITE SYS$OUTPUT ""
$ WRITE SYS$OUTPUT "Beginning MSCPMOUNT procedure at ",F$TIME(),"..."
$ WRITE SYS$OUTPUT ""
$ REQPRIVS      = "VOLPRO,OPER,SYSNAM,GRPNAM" - ! Mounting
                + ",SYSPRV,CMKRNL,ALTPRI,DETACH"! Other
$ LOCAL_NODE == F$GETSYI("NODENAME")
$ SAVE_DEFAULT = F$ENVIRONMENT("DEFAULT")
$ PREVPRIVS = F$SETPRV(REQPRIVS)
$ IF .NOT. F$PRIVILEGE(REQPRIVS) THEN GOTO RESET_PROCESS
$ SET DEFAULT SYS$MANAGER:
$ !
$ ! Mount volume sets
$ !
$ RECURSIVE_CALLBACK MOUNT "$1$DRB6,$1$DRB7" "ADMIN1,ADMIN2"
$ !
$ ! Mount disks residing on HSCs.  In this example, the HSC's
$ ! have the disk allocation class set to 255.
$ !
$ RECURSIVE_CALLBACK MOUNT $255$DUA12 STARWORK01
$ RECURSIVE_CALLBACK MOUNT $255$DUA13 STARWORK02
$ RECURSIVE_CALLBACK MOUNT $255$DUA14 STARWORK03
$ !
$ ! All done mounting disks.  If request was ONCE, then exit.
$ ! If we are the detached process then go wait a while.
$ ! Otherwise, this is the first invocation of the procedure,
$ ! we must create the detached process.
$ !
$CHECK_RECURSIVE:
$ IF (P1 .EQS. "ONCE") THEN GOTO RESET_PROCESS
$ IF F$GETJPI(0,"PRCNAM") .EQS. "MSCPmount" THEN GOTO DETACHED_LOOP
$ !
$ ! Start the detached process
$ !
$ PURGE SYS$MANAGER:MSCPMOUNT_'LOCAL_NODE'.LOG /KEEP:3
$ IF F$MODE() .EQS. "INTERACTIVE" THEN -
        WRITE SYS$OUTPUT "% Ignore possible duplicate process name error."
$ RUN SYS$SYSTEM:LOGINOUT       -
                /NOAUTHORIZE            -
                /UIC=[1,4]              -
                /PRIORITY=4             -
                /MAXIMUM_WORKING_SET=1024       -
                /EXTENT=2048            -
                /INPUT='PROCEDURE_NAME' -
                /OUTPUT=SYS$MANAGER:MSCPMOUNT_'LOCAL_NODE'.LOG -
                /ERROR=SYS$MANAGER:MSCPMOUNT_'LOCAL_NODE'.LOG   -
```

```
                /PROCESS_NAME="MSCPmount"
$
$RESET_PROCESS:
$ SET DEFAULT 'SAVE_DEFAULT'
$ PRIVS = F$SETPRV(PREVPRIVS)
$ VERIFY=F$VERIFY(VERIFY)
$ EXIT
$ !
$ ! This incarnation is the detached process which is supposed to
$ ! stay around forever continually trying to re-mount disks.  So,
$ ! lets loop right here.  Note that we must invoke ourself recursively
$ ! in order to make sure we get any new versions of this procedure.
$ !
$DETACHED_LOOP:
$ WAIT 'WAIT_INTERVAL'
$ RECURSIVE_CALLBACK ONCE
$ GOTO DETACHED_LOOP
$MOUNT_DEVICE:
$ !
$ ! This section of the procedure is entered on a recursive callback.
$ ! If the disk exists and is not currently mounted, it will be mounted.
$ !
$ ! The parameters for this routine are:
$ !     P2 = full disk name or list of same (e.g. $1$DUA15, STAR$DBB3)
$ !     P3 = volume label or list of same (e.g. STARWORK04)
$ !     P4 = mount qualifiers (e.g. /NOWRITE, optional)
$ !
$ PERFORM_MOUNT = "YES"
$ PERFORM_DISMOUNT = "NO"
$ IF P3 .EQS. "" THEN PERFORM_MOUNT = "NO"
$ DEVICE_ELEMENT = 0
$
$ ! Loop through all devices listed in P2
$
$DEVICE_LOOP:
$ DEVICE = F$ELEMENT( DEVICE_ELEMENT, ",", P2 )
$ IF DEVICE .EQS. "" THEN GOTO ENDING            ! If P2 null, exit immediately.
$ IF DEVICE .EQS. "," THEN GOTO DO_DISMOUNT      ! If end of list, do DISMOUNT/MOUNT.
$
$ DEVICE_ELEMENT = DEVICE_ELEMENT + 1
$ !
$ ! If nonexistant device, skip the rest of this.
$ !
$ IF .NOT. F$GETDVI( DEVICE, "EXISTS" ) THEN GOTO NONEXISTANT_DEVICE
$ IF .NOT. F$GETDVI( DEVICE, "HOST_AVAIL" ) THEN GOTO HOST_NOT_AVAILABLE
$ !
$ ! If already mounted, set flag to not do it again.
$ !
$ IF .NOT. F$GETDVI( DEVICE, "MNT" ) THEN GOTO DEVICE_LOOP
$ PERFORM_MOUNT = "NO"
$ !
$ ! Bit %X800 in STS is VALID bit -- check if disk is not valid
$ ! Bit %X4000 in STS is MNTVERIP bit -- check if disk is not in mount verification
$ ! Not Valid implies that mount verification has timed out
$ !
$ IF (( F$GETDVI(DEVICE,"STS") .AND. %X4800 ) .NE. 0 ) THEN GOTO DEVICE_LOOP
```

```
$ PERFORM_DISMOUNT = "YES"
$ GOTO DEVICE_LOOP
$
$HOST_NOT_AVAILABLE:
$ WRITE SYS$OUTPUT -
  "%MSCPMOUNT-W-NOHOST, host ''f$getdvi(Device,"HOST_NAME")' for ''f$getdvi(Device,"FULLDEVNAM")' is not available now"
$NONEXISTANT_DEVICE:
$ GOTO ENDING
$
$DO_DISMOUNT:
$ PRCNAM = F$GETJPI(0,"PRCNAM")
$ IF .NOT. PERFORM_DISMOUNT THEN GOTO DO_MOUNT
$ PERFORM_MOUNT = "YES"
$ DEVICE = F$ELEMENT( 0, ",", P2 )
$ REPLY /NODE /ALL "Remounting ''DEVICE' on ''f$getsyi("NODENAME")' after mount verification timeout"
$ SET PROCESS /NAME="F$EXTRACT(0,15,"DMO_"+DEVICE)"
$ VERIFY=F$VERIFY(1)
$ DISMOUNT /NOUNLOAD /ABORT 'DEVICE'
$!'F$VERIFY(VERIFY)
$ SET PROCESS /NAME="""PRCNAM"""
$ WAIT 0:0:15
$
$DO_MOUNT:
$ IF .NOT. PERFORM_MOUNT THEN GOTO ENDING
$ DEVICE = F$ELEMENT( 0, ",", P2 )
$ S_CLUSTER = ""
$ !
$ ! Do not mount HSC-based disks with /CLUSTER, under the assumption
$ ! that all nodes wanting access to the disk will mount it themselves.
$ ! Mount MSCP-served disks with /CLUSTER, as a 'courtesy' to the
$ ! rest of the cluster.
$ !
$ ! If you add /NOREBUILD to the mount command to override the rebuilding
$ ! of the disk, you must ensure that the disk rebuilds are done via some
$ ! other mechanism.
$ !
$ IF PRCNAM .EQS. "STARTUP" THEN S_CLUSTER = "/CLUSTER"
$ IF F$GETDVI(DEVICE,"HOST_TYPE") .EQS. "HS50" THEN S_CLUSTER = ""
$ SET PROCESS /NAME="F$EXTRACT(0,15,"MNT_"+DEVICE)"
$ VERIFY=F$VERIFY(1)
$ MOUNT 'S_CLUSTER' /SYSTEM /NOASSIST 'P4' 'P2' 'P3'
$!'F$VERIFY(VERIFY)
$ SET PROCESS /NAME="""PRCNAM"""
$
$ENDING:
$ EXIT
```

0159 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

MAILCVT
COM

SYSGTTSTR
MSG

EXCREQ
R32

EXCCOPY
CTS

LSSLNK
COM

EXCDEFS
SDL

EXCLIB
B32

EXCCLDTBL
CTS

MAILUAF
COM

LSSTSTLNK
COM

EXCHNG

XATEST
COM

EXCHANGE
MAP

LABIO
OPT

MSCPMOUNT
COM

LABIOCIN
OPT

EXCCMD
CTS

DRCOPY
PRM