J 16

```
MMM         MMM     AAAAAAAAA     NNN         NNN     AAAAAAAAA        GGGGGGGGGGGG  EEEEEEEEEEEEEEE
MMM         MMM     AAAAAAAAA     NNN         NNN     AAAAAAAAA        GGGGGGGGGGGG  EEEEEEEEEEEEEEE
MMM         MMM     AAAAAAAAA     NNN         NNN     AAAAAAAAA        GGGGGGGGGGGG  EEEEEEEEEEEEEEE
MMMMMM   MMMMMM  AAA         AAA  NNN         NNN  AAA         AAA  GGG              EEE
MMMMMM   MMMMMM  AAA         AAA  NNN         NNN  AAA         AAA  GGG              EEE
MMMMMM   MMMMMM  AAA         AAA  NNN         NNN  AAA         AAA  GGG              EEE
MMM   MMM   MMM  AAA         AAA  NNNNNN      NNN  AAA         AAA  GGG              EEE
MMM   MMM   MMM  AAA         AAA  NNNNNN      NNN  AAA         AAA  GGG              EEE
MMM   MMM   MMM  AAA         AAA  NNNNNN      NNN  AAA         AAA  GGG              EEE
MMM         MMM  AAA         AAA  NNN   NNN   NNN  AAA         AAA  GGG              EEEEEEEEEEEE
MMM         MMM  AAA         AAA  NNN   NNN   NNN  AAA         AAA  GGG              EEEEEEEEEEEE
MMM         MMM  AAA         AAA  NNN   NNN   NNN  AAA         AAA  GGG              EEEEEEEEEEEE
MMM         MMM  AAAAAAAAAAAAAAA  NNN      NNNNNN  AAAAAAAAAAAAAAA  GGG   GGGGGGGGG  EEE
MMM         MMM  AAAAAAAAAAAAAAA  NNN      NNNNNN  AAAAAAAAAAAAAAA  GGG   GGGGGGGGG  EEE
MMM         MMM  AAAAAAAAAAAAAAA  NNN      NNNNNN  AAAAAAAAAAAAAAA  GGG   GGGGGGGGG  EEE
MMM         MMM  AAA         AAA  NNN         NNN  AAA         AAA  GGG         GGG  EEE
MMM         MMM  AAA         AAA  NNN         NNN  AAA         AAA  GGG         GGG  EEE
MMM         MMM  AAA         AAA  NNN         NNN  AAA         AAA  GGG         GGG  EEE
MMM         MMM  AAA         AAA  NNN         NNN  AAA         AAA     GGGGGGGGG     EEEEEEEEEEEEEEE
MMM         MMM  AAA         AAA  NNN         NNN  AAA         AAA     GGGGGGGGG     EEEEEEEEEEEEEEE
MMM         MMM  AAA         AAA  NNN         NNN  AAA         AAA     GGGGGGGGG     EEEEEEEEEEEEEEE
```

```
**FILE**ID**AUTOGEN                                              B  1




  AAAAAA    UU      UU  TTTTTTTTTT    OOOOOO      GGGGGGGG  EEEEEEEEEE  NN      NN
  AAAAAA    UU      UU  TTTTTTTTTT    OOOOOO      GGGGGGGG  EEEEEEEEEE  NN      NN
AA      AA  UU      UU      TT      OO      OO  GG          EE          NN      NN
AA      AA  UU      UU      TT      OO      OO  GG          EE          NN      NN
AA      AA  UU      UU      TT      OO      OO  GG          EE          NNNN    NN
AA      AA  UU      UU      TT      OO      OO  GG          EE          NNNN    NN
AA      AA  UU      UU      TT      OO      OO  GG          EEEEEEEE    NN  NN  NN
AA      AA  UU      UU      TT      OO      OO  GG          EEEEEEEE    NN  NN  NN
AAAAAAAAAA  UU      UU      TT      OO      OO  GG  GGGGGG  EE          NN    NNNN
AAAAAAAAAA  UU      UU      TT      OO      OO  GG  GGGGGG  EE          NN    NNNN
AA      AA  UU      UU      TT      OO      OO  GG      GG  EE          NN      NN      ....
AA      AA  UU      UU      TT      OO      OO  GG      GG  EE          NN      NN      ....
AA      AA  UUUUUUUUUU      TT        OOOOOO      GGGGGG    EEEEEEEEEE  NN      NN      ....
AA      AA  UUUUUUUUUU      TT        OOOOOO      GGGGGG    EEEEEEEEEE  NN      NN      ....

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
```

```
$!	IDENT V04-000
$!
$!++
$! Facility:	AUTOGEN, Automatic System Tuning Procedure
$!
$! Module:	AUTOGEN
$!
$! Abstract:	This procedure is a collection of  subprocedures that
$!		attempt to configure and tune a VMS system for a site's
$!		specific hardware environment and typical user needs.
$!
$!		For a description of the subprocedures, see the help
$!		at the end of this file.
$!
$! Author:	Peter George
$!
$! Created:	01-Sep-1983
$!
$! Modifications:
$!
$!--
$!
```

```
$!
$! Save old verification state.
$!
$ ON CONTROL_Y THEN GOTO common_exit90
$ IF "''F$LOGICAL("AUTOGEN$SAVE_VERIFY")'" .NES. "" THEN DEASSIGN autogen$save_verify
$ temp = 0
$ IF "''F$LOGICAL("AUTOGEN$VERIFY")'" .NES. "" THEN temp = 1
$ DEFINE autogen$save_verify 'F$VERIFY(temp)'
$
$!
$! Check that the input parameters are valid and set the defaults.
$!
$ p1_list = " HELP, SAVPARAMS, GETDATA, GENPARAMS, SETPARAMS, SHUTDOWN, REBOOT,"
$ p2_list = " SAVPARAMS, GETDATA, GENPARAMS, GENFILES, TESTFILES, SETPARAMS, SHUTDOWN, REBOOT,"
$ p3_list= " INITIAL, V3UPGRADE, V4UPGRADE,"
$
$ IF p1 .EQS. "HELP" THEN goto help
$ IF p1 .EQS. "" THEN p1 = "GENPARAMS"
$ IF p2 .EQS. "" THEN p2 = p1
$ IF p3 .EQS. "" THEN p3 = "V4UPGRADE"
$
$ i = 1
$start10:
$       IF F$LOCATE(" " + p'i' +",", p'i'_list) .EQ. F$LENGTH(p'i'_list) -
            THEN GOTO start20
$       IF i .EQ. 3 THEN GOTO start30
$       i = i + 1
$       GOTO start10
$
$start20:
$ temp = F$EXTRACT(0,F$LENGTH(p'i'_list)-1,p'i'_list)
$ WRITE sys$output "%AUTOGEN-E-IVKEYW, parameter P",i," (",p'i',") is invalid.  Specify one of:"
$ WRITE sys$output temp
$ WRITE sys$output "%AUTOGEN-I-NOP, AUTOGEN has not attempted to execute any phases."
$ EXIT
$
$start30:
$ IF F$LOCATE(p1,p2_list) .LE. F$LOCATE(p2,p2_list) THEN GOTO start40
$ WRITE sys$output "%AUTOGEN-E-PHASORDER, the start phase (",p1,") must preceed "
$ WRITE sys$output "         the end phase (",p2,")."
$ WRITE sys$output "%AUTOGEN-I-NOP, AUTOGEN has not attempted to execute any phases."
$ EXIT
$
$start40:
$ IF F$PRIV("SYSP") THEN GOTO start50
$ WRITE SYS$OUTPUT "%AUTOGEN-E-NOPRIV, SYSPRV privilege required to run AUTOGEN."
$ WRITE sys$output "%AUTOGEN-I-NOP, AUTOGEN has not attempted to execute any phases."
$ EXIT
$
$start50:
$ DEFINE autogen$p1 'p1'
$ DEFINE autogen$p2 'p2'
$ DEFINE autogen$p3 'p3'
$ GOTO 'p1'
$.
```

```
$.++
$.
$. Module:      Common routines
$.
$!--
$common_abort:
$ quit = "abort"
$ GOTO 'phase'_abort
$common_abort90:
$ WRITE sys$output "%AUTOGEN-I-CTRLY, ",phase," phase was aborted by a CTRL/Y."
$ GOTO common_exit
$
$common_out rr:
$ WRITE sys$output "%AUTOGEN-E-OPENOUT, ",file," could not be created."
$ WRITE sys$output "        Please correct the problem and then reinvoke AUTOGEN."
$ GOTO common_err
$
$common_inerr:
$ WRITE sys$output "%AUTOGEN-E-OPENIN, ",file," could not be read."
$ WRITE sys$output "        Please correct the problem and then reinvoke AUTOGEN."
$
$common_err:
$ quit = "err"
$ GOTO 'phase'_abort
$common_err90:
$ WRITE sys$output "%AUTOGEN-I-ERROR, ",phase," phase was aborted due to an unexpected error."
$
$common_exit:
$ DEASSIGN autogen$p1
$ DEASSIGN autogen$p2
$ DEASSIGN autogen$p3
$common_exit90:
$ IF F$LOGICAL("AUTOGEN$SAVE_VERIFY") THEN SET VERIFY
$ DEASSIGN autogen$save_verify
$ EXIT
$!
```

```
$!++
$!
$. Module:     HELP
$!
$!--
$help:
$ ON CONTROL_Y THEN EXIT
$ TYPE sys$input

                AUTOGEN - Automatic System Tuning Procedure

AUTOGEN is a system management tool that automatically sets the values of
system parameters, the sizes of the paging, swapping, and dump files, and the
contents of the default installed image list based on its evaluation of your
hardware configuration and typical system workloads.

To ensure that you have the required privileges, invoke AUTOGEN from the system
manager's account. The format for invoking AUTOGEN is:

        @SYS$UPDATE:AUTOGEN [start-phase] [end-phase] [execution-type]

You can enter up to three parameters to designate the AUTOGEN operation you
desire. Note that all parameters are optional; however, missing leading
parameters must be replaced by null arguments (i.e., "").

The following tables list the phase parameter values and their effects,
including the files needed as input and the files created or changed for
output, and summarize the execution types.

All files except VMSIMAGES.DAT reside in the directory specified by the
SYS$SYSTEM logical name. VMSIMAGES.DAT resides in SYS$MANAGER.

The start-phase must either precede or be identical to the end-phase according
to the sequence shown in the table (the end-phase defaults to the same value
as the start-phase).  GENPARAMS is the default start-phase; GENFILES may not be
specified as the start-phase.

                AUTOGEN Phase Parameter Values

Phase           Input Files     Output Files    Function
-----           -----------     ------------    --------
SAVPARAMS       None            OLDSITE*.DAT    Save significant old parameters
                                                for propagation and update.

GETDATA         OLDSITE*.DAT    PARAMS.DAT      Collect all data that will be
                MODPARAMS.DAT                   required by the GENPARAMS,
                                                GENFILES, and TESTFILE phases,
                                                including configuration data,
                                                old parameters, and
                                                site-specific items.

GENPARAMS       PARAMS.DAT      SETPARAMS.DAT   Generate new system parameters;
                                VMSIMAGES.DAT   create the installed image list.

TESTFILES       PARAMS.DAT      SYS$OUTPUT      Display the system page, swap,
                                                and dump file sizes calculated
                                                by AUTOGEN.  Cannot be specified
                                                as the start-phase.

GENFILES        PARAMS.DAT      PAGEFILE.SYS    Generate new system page, swap,
                                SWAPFILE.SYS    and dump files if appropriate.
                                SYSDUMP.DMP     Cannot be specified as the
                                                start-phase.
```

| | | | |
|---|---|---|---|
| SETPARAMS | SETPARAMS.DAT | AUTOGEN.PAR | Run SYSGEN to set system parameters specified by SETPARAMS.DAT and to generate a new AUTOGEN.PAR file. |
| SHUTDOWN | None | None | Prepare the system to await a manual reboot. |
| REBOOT | None | None | Automatically reboot the system. |

## AUTOGEN Execution type parameters

| Type | Meaning |
|---|---|
| INITIAL | Specifies that AUTOGEN is being executed as part of an initial system installation. The SAVPARAMS phase is never executed in this case. |
| V4UPGRADE | Specifies that AUTOGEN is being executed as part of an upgrade from a Version 4 system or that interactive tuning is being performed. V4UPGRADE is the default execution type. |
| V3UPGRADE | Specifies that AUTOGEN is being executed as part of an upgrade from a Version 3 system to a Version 4 system. |

If, after examining the parameters generated by AUTOGEN, you decide you wish to correct hardware configuration data, modify system parameter values, or explicitly specify sizes for the system page, swap, or dump files, follow the steps outlined below.

1. Edit the file, SYS$SYSTEM:MODPARAMS.DAT. To retain a history of the changes you have made, always add modifications to the end of the file.

2. Specify new configuration data or parameter values by inserting DCL assignment statements of the form:

```
parameter = parameter-value       ! comment
```

3. Specify incremental modifications to parameter values by inserting DCL assignment statements of the form:

```
ADD_parameter = parameter-value   ! comment
```

4. Specify system file sizes explicitly by specifying the keywords PAGEFILE, SWAPFILE, and DUMPFILE followed by an equal sign and the size of the file in blocks. Specifying a value of 0 for any of these keywords instructs AUTOGEN not to modify the size of the corresponding file.

5. Rerun AUTOGEN from the SAVPARAMS or GETDATA phase. The modifications specified in MODPARAMS.DAT will be copied into PARAMS.DAT during the GETDATA phase, and AUTOGEN will make appropriate adjustments in its calculations in later phases.

For further details about how to use AUTOGEN refer to the tuning chapter in

H 1

```
the VAX/VMS System Management and Operations Guide.
$ exit
$
```

```
$!++
$!
$! Module:      SAVPARAMS
$!
$! Abstract:    Used in an upgrade to save some of a site's old parameter
$!              values.  The files SYS$SYSTEM:OLDSITE*.DAT are created.
$!              Note that only Version 3.0 DCL features may be used
$!              so that this command procedure can run on V3.x systems.
$!
$! Parameters:  P1 indicates what type of of data collection to perform.
$!
$!                      INITIAL   - Initial system installation.
$!                      V3UPGRADE - Upgrade from a V3.x system.
$!                      V4UPGRADE - Upgrade from a V4.x system. (D)
$!
$!--
$!
$! Initialize this phase.
$!
$savparams:
$ phase = "SAVPARAMS"
$ ON CONTROL_Y THEN GOTO common_abort
$ ON ERROR THEN GOTO common_err
$ p1 = F$LOGICAL("AUTOGEN$P1")
$ p2 = F$LOGICAL("AUTOGEN$P2")
$ p3 = F$LOGICAL("AUTOGEN$P3")
$ WRITE sys$output "%AUTOGEN-I-BEGIN, ",phase," phase is beginning."
$
$!
$! If this is an initial installation, then skip this phase.
$!
$ IF p3 .NES. "INITIAL" THEN GOTO savparams10
$ WRITE sys$output "%AUTOGEN-I-SKIP, SAVPARAMS phase is being skipped.  It is not"
$ WRITE sys$output "        needed when performing an INITIAL installation."
$ GOTO savparams_cleanup
$savparams10:
$
$!
$! Change error handler to file creation error handler.
$!
$ ON ERROR THEN GOTO common_outerr
$!
```

```
$!
$! 1 -  These parameters are used to generate symbols of the form
$!      OLD_sysgenparam which are used to calculate new values.
$!
$ file = "SYS$SYSTEM:OLDSITE1.DAT"
$ RUN sys$system:sysgen
use current
set/output=SYS$SYSTEM:OLDSITE1.DAT
show gblpages            ! # of global page table entries allocated
show gblsections         ! # of global section descriptors allocated
show gblpagfil           ! max # of global pages allowed for RMS global buffers
show maxbuf              ! max size of buffered I/O transfer
show maxprocesscnt       ! # of process entry slots
show virtualpagecnt      ! max # of virtual pages mapped per process
exit
$
$!
```

```
$
$
$
$
$
$
$
$
$
$
$
$
$
$
$
```

```
$!
$! 2 -  These parameters are propagated to the new system if
$!      if the old value is greater than the old default value.
$!
$ file = "SYS$SYSTEM:OLDSITE2.DAT"
$ RUN sys$system:sysgen
use current
set/output=SYS$SYSTEM:OLDSITE2.DAT
show acp_extcache          ! # of entries in the extent cache
show acp_extlimit          ! max amount of free space for extent cache
show acp_fidcache          ! # of file identification slots cached
show clisymtbl             ! size of command interpreter symbol table
show defmbxbufquo          ! default for mailbox buffer quota size
show defmbxmxmsg           ! default for mailbox max message size
show defmbxnummsg          ! ??? not implemented ???
show intstkpages           ! size of interrupt stack
show lrpsize               ! size of large request packets
show pql_dastlm            ! def limit on # of pending ASTs for $CREPRC process
show pql_dbiolm            ! def buffered I/O count limit for $CREPRC process
show pql_dbytlm            ! def buffered I/O byte count limit for $CREPRC proc
show pql_ddiolm            ! def direct I/O limit for $CREPRC process
show pql_dfillm            ! def open file limit for $CREPRC process
show pql_dwsdefault        ! def working set size for $CREPRC process
show pql_dwsextent         ! def working set extent for $CREPRC process
show pql_dwsquota          ! def working set quota for $CREPRC process
show pql_mastlm            ! min limit on # of pending ASTs for $CREPRC process
show pql_mbiolm            ! min buffered I/O count limit for $CREPRC process
show pql_mbytlm            ! min buffered I/O byte count limit for $CREPRC proc
show pql_mdiolm            ! min direct I/O limit for $CREPRC process
show pql_mfillm            ! min open file limit for $CREPRC process
show pql_mwsdefault        ! min working set size for $CREPRC process
show pql_mwsextent         ! min working set extent for $CREPRC process
show pql_mwsquota          ! min working set quota for $CREPRC process
show procsectcnt           ! # of section descriptors that a process can contain
show srpsize               ! size of small request packets
exit
$!
```

```
$.
$. 3 -  These parameters are propagated to the new system if
$!      if the old value is different from the old default value.
$!
$ file = "SYS$SYSTEM:OLDSITE3.DAT"
$ RUN sys$system:sysgen
use current
set/output=SYS$SYSTEM:OLDSITE3.DAT
show acp_baseprio         ! base priority for all ACPs
show acp_datacheck        ! enables verification of file struc data read/writing
show acp_swapflgs         ! enables ACP swapping
show acp_writeback        ! enables deferred writing of file headers
show bugcheckfatal        ! enables conversion of nonfatal bugchecks to fatal
show bugreboot            ! enable automatic reboot after fatal bugcheck
show crdenable            ! enables detection and logging of memory ECC errors
show deadlock_wait        ! # of secs that a lock request must wait before the
                          !     system initiates a deadlock search
show defpri               ! default priority for p ocesses
show dismoumsg            ! controls oper term reporting of log volume dismounts
show dumpbug              ! enables dumping on fatal bugcheck
show extracpu             ! time allotted to each process exit handler (per mode)
                          !     after the process has cpu timed out
show lamapregs            ! # of UNIBUS map registers allocated to LPA11 driver
show longwait             ! time before swapper considers a process to be idle
show maxsysgroup          ! highest system UIC group number
show mountmsg             ! controls oper term reporting of log volume mounts
show mvtimeout            ! secs that a mount verification attempt will continue
show pagfilcnt            ! max # of paging files that can be installed
show pql_dcpulm           ! default CPU time limit for $CREPRC processes
show pql_dprclm           ! default subprocess limit for $CREPRC processes
show pql_dtqelm           ! default # of timer queue entries for $CREPRC processes
show pql_mcpulm           ! min CPU time limit for $CREPRC processes
show pql_mprclm           ! min subprocess limit for $CREPRC processes
show pql_mtqelm           ! min # of time queue entries for $CREPRC processes
show realtime_spts        ! reserves # of system page table entries for mapping
                          ! connect-to-interrupt processes into system space
show savedump             ! enables saving of crash dumps
show scssystemid          ! DECnet node number
show swpfilcnt            ! max # of swap files that can be installed
show timepromptwait       ! time to wait for system date/time when booting
show tty_altalarm         ! size of alternate type-ahead buffer alarm
show tty_altypahd         ! size of alternate type-ahead buffer
show tty_buf              ! default line width for terminals
show tty_dialtype         ! dial-up flag bits
show tty_owner            ! owner UIC against which terminal protection is checked
show tty_parity           ! terminal default parity
show tty_prot             ! default protection for all terminals wrt TTY_OWNER
show tty_rspeed           ! receive speed for terminals
show tty_scandelta        ! polling interval for dial-up and hang-up events
show tty_silotime         ! interval at which input silo is polled by DMF-32
show tty_speed            ! default speed for terminal
show tty_typahdsz         ! size of terminal type ahead buffer
show uafalternate         ! enables assignment of alternate UAF
show user3                ! user-specific parameters
show user4                ! ...
show userd1               ! ..
show userd2               ! .
show xfmaxrate            ! limit data transfer rate for DR32 devices
exit
$!
```

```
$
$! 4 -  These are Version 4.0 parameters that are propagated to the new
$!      system if the old value is different from the old default value.
$!
$ IF p3 .EQS. "V3UPGRADE" THEN GOTO savparams_cleanup
$ file = "SYS$SYSTEM:OLDSITE4.DAT"
$ RUN sys$system:sysgen
use current
set/output=SYS$SYSTEM:OLDSITE4.DAT
show acp_rebldsysd      ! determines whether system disk needs to be rebuilt
show alloclass          !
show cjfload            ! determines whether CFJ is loaded with the system
show cjfsysruj          ! determines whether a RU journal exists on system disk
show disk_quorum        ! name of an optional quorum disk
show lnmphashtbl        !
show lnmshashtbl        !
show lockdirwt          !
show prcpolinterval     ! pollint inerval used to look for SCS applications
show qdskinterval       ! disk quorum polling interval
show qdskvotes          !
show quorum             ! quorum for a cluster
show recnxinterval      ! polling interval for reconnection to remote system
show scsnode            ! SCS system name
show scssystemidh       ! SCS system ID (high)
show tailored           ! system is tailored
show tty_defchar        ! default terminal characteristics
show tty_defchar2       ! .
show vaxcluster         !
show votes              ! # of votes of a VAXcluster member
show vmsd3              ! mag tape time out interval
exit
$
$!
```

```
$!
$! Cleanup.
$!
$savparams_cleanup:
$ ON ERROR THEN GOTO common_err
$ WRITE sys$output "%AUTOGEN-I-NEWFILE, New versions of SYS$SYSTEM:OLDSITE*.DAT have been created."
$ WRITE sys$output "        You may wish to purge these files."
$ WRITE sys$output "%AUTOGEN-I-END, ",phase," phase has succesfully completed."
$ IF p2 .EQS. "SAVPARAMS" THEN GOTO common_exit
$ GOTO getdata
$
$!
$! Cleanup after errors and CTRL/Ys.
$!
$savparams_abort:
$ ON CONTROL_Y THEN GOTO savparams_abort
$ ON ERROR THEN CONTINUE
$ WRITE sys$output "%AUTOGEN-I-BADFILE, Bad versions of SYS$SYSTEM:OLDSITE*.DAT may exist."
$ WRITE sys$output "        We recommend that you delete all versions and start again."
$ GOTO common_'quit'90
$!
```

```
$!
$! Module:      GETDATA
$!
$! Abstract:    This procedure is used to collect all the data that the
$!              generation routines will need and to write that data to a
$!              user-editable site-specific requirements file,
$!              SYS$SYSTEM:PARAMS.DAT.  This data includes the current
$!              hardware configuration, some of a site's old parameter
$!              values (contained in the files SYS$SYSTEM:OLDSITE*.DAT)
$!              if this is an upgrade, and whatever parameters the system
$!              manager may have specified in SYS$SYSTEM:MODPARAMS.DAT.
$!
$!--
$!
$! Initialize this phase.
$!
$getdata:
$ DELETE/SYMBOL/LOCAL/ALL
$ phase = "GETDATA"
$ ON CONTROL_Y THEN GOTO common_abort
$ ON ERROR THEN GOTO common_err
$ p1 = F$LOGICAL("AUTOGEN$P1")
$ p2 = F$LOGICAL("AUTOGEN$P2")
$ p3 = F$LOGICAL("AUTOGEN$P3")
$ WRITE sys$output "%AUTOGEN-I-BEGIN, ",phase," phase is beginning."
$
$!
$! If the user doesn't have CMK priv, then abort now.
$!
$ IF F$PRIV("CMK") THEN GOTO getdata10
$ WRITE SYS$OUTPUT "%AUTOGEN-E-NOPRIV, CMKRNL privilege required for GETDATA phase."
$ GOTO common_err90
$
$!
$! Create a file to write the collected data into.
$!
$getdata10:
$ file = "SYS$SYSTEM:PARAMS.DAT"
$ OPEN/WRITE/ERROR=common_outerr data 'file'
$ WRITE data "!"
$ WRITE data "! This data file should NOT be modified.  Users wishing to alter the"
$ WRITE data "! data in this file should modify SYS$SYSTEM:MODPARAMS.DAT instead."
$ WRITE data "!"
$
$!
```

```
$!
$! Configure the I/O devices, just in case it hasn't been done yet.
$! If the user has a configuration procedure, invoke it.
$! Then, unless told otherwise by the value of startup$autoconfigure_all,
$! autoconfigure all devices.
$!
$ x1 = "FULL"
$ x2 = F$EDIT(F$GETSYI("startup_p2"),"TRIM,UPCASE")
$ x3 = F$EDIT(F$GETSYI("startup_p3"),"TRIM,UPCASE")
$ x4 = F$EDIT(F$GETSYI("startup_p4"),"TRIM,UPCASE")
$ x5 = F$EDIT(F$GETSYI("startup_p5"),"TRIM,UPCASE")
$ x6 = F$EDIT(F$GETSYI("startup_p6"),"TRIM,UPCASE")
$ x7 = F$EDIT(F$GETSYI("startup_p7"),"TRIM,UPCASE")
$ x8 = F$EDIT(F$GETSYI("startup_p8"),"TRIM,UPCASE")
$ startup$autoconfigure_all == T
$ IF F$SEARCH("sys$manager:syconfig.com") .NES. "" -
     THEN @sys$manager:syconfig """x1""" """x2""" """x3""" """x4""" """x5""" """x6""" """x7""" """x8"""
$ IF .NOT. startup$autoconfigure_all THEN GOTO getdata15
$ ON ERROR THEN CONTINUE
$ SYSGEN = "$SYSGEN"
$ DEFINE/USER sys$error nl:
$ DEFINE/USER sys$output nl:
$ SYSGEN autoconfigure all
$ ON ERROR THEN GOTO common_err
$
$!
```

```
$!
$! Get system version number, cpu type, and SID using F$GETSYI and write
$! that data to the data file.
$!
$getdata15:
$ version = F$GETSYI("VERSION")
$ WRITE data "VERSION=""""",version,""""""
$ cputype = F$GETSYI("CPU")
$ IF (cputype .LT. 1) .OR. (cputype .GT. 8) THEN cputype = 0
$ WRITE data "CPUTYPE=",cputype
$ sid = F$GETSYI("SID")
$ WRITE data "SID=",sid
$
$!
```

```
$!
$! Get the physical memory size in pages by parsing the output from the
$! SHOW MEMORY command.
$!
$ ON ERROR THEN GOTO common_outerr
$ file = "SYS$SYSTEM:AUTOGEN.TMP"
$ DEFINE/USER sys$output 'file'
$ SHOW MEMORY/PHYSICAL_MEMORY
$
$ ON ERROR THEN GOTO common_err
$ file = "SYS$SYSTEM:AUTOGEN.TMP"
$ OPEN/READ/ERROR=common_inerr tempfile 'file'
$
$!
$! Skip to the record that contains the main memory size and then extract it
$! by searching for the first blank delimited string after the first left
$! parenthesis.  Write the main memory size to the data file.
$!
$getdata20:
$       READ tempfile record
$       length  = F$LENGTH(record)
$       IF F$LOCATE("Main Memory",record) .EQ. length THEN GOTO getdata20
$
$ temp = F$LOCATE(")",record)
$ record = F$EDIT(F$EXTRACT(temp+1,length-temp-1,record),"TRIM,COMPRESS")
$ memsize = F$EXTRACT(0,F$LOCATE(" ",record),record)
$ WRITE data "MEMSIZE=",memsize
$ CLOSE tempfile
$
$!
```

```
$!
$! Calculate indicator of system disk speed based on the system disk type.
$! Use the information in the $DCDEF macro to define the possible system
$! disk types.  Find out which we have and save its speed.  Also store
$! a disk size indicator (<53000 blocks is small).
$!
$! The following assumptions are made for known disk types.
$! (1 = slow, 2 = medium, 3 = fast, -1 = unsupported or unrecognized disk)
$!
$!      Disk type       Disk speed      DT$_xxx (1-26)
$!      ---------       ----------      --------------
$!      RA60                4           22
$!      RA80,81,82          4           20,21,30
$!      RB02                1           18
$!      RB80                2           19
$!      RC26,RCF26          2           31,32
$!      RD26                2           29
$!      RD51,52,53          1           25,27,28
$!      RK06,7              2           1,2
$!      RL01,2              1           9,10
$!      RM03,5,80           4           6,15,13
$!      RP04,5,6,7,7HT      4           3,4,5,7,8
$!      RZ01,RZF01          2           23,24
$!
$!      RX01,2,4,50        -1           16,11,12,26
$!      ML11,TU58          -1           17,14
$!
$ speed_list = "-1,2,2,4,4,4,4,4,4,1,1,-1,-1,4,-1,4,-1,-1,1,2,4,4,4,2,2,1,-1,1,1,2,4,2,2"
$ diskspeed = -1
$ temp = F$GETDVI("sys$sysdevice","DEVTYPE")
$ IF (temp .LE. 32) .AND. (temp .GE. 1) -
     THEN diskspeed = F$ELEMENT(temp,",",speed_list)
$ smalldisk = "false"
$ IF F$GETDVI("sys$sysdevice","MAXBLOCK") .LE. 53000 THEN smalldisk = "true"
$
$ IF diskspeed .NE. -1 THEN GOTO getdata30
$ WRITE sys$output "%AUTOGEN-W-UNKDISK, unsupported system disk type.  Using speed and"
$ WRITE sys$output "          size characteristics of an RK07."
$ diskspeed = 2
$ smalldisk = "false"
$
$getdata30:
$ WRITE data "DISKSPEED=",diskspeed
$ WRITE data "SMALLDISK=""",smalldisk,""""
$
$!
```

```
$!
$! Count the number of devices of each class on the system and write that
$! information to the data file.  Use the SHOW DEVICES command to get the
$! list of all the devices on the system, since there is no other way
$! to wildcard through the devices.
$!
$ ON ERROR THEN GOTO common_outerr
$ file = "SYS$SYSTEM:AUTOGEN.TMP"
$ DEFINE/USER sys$output 'file'
$ SHOW DEVICES/BRIEF
$
$ ON ERROR THEN GOTO common_err
$ file = "SYS$SYSTEM:AUTOGEN.TMP"
$ OPEN/READ/ERROR=common_inerr tempfile 'file'
$
$!
$! Use the information in the $DCDEF macro to define the possible system
$! device classes.  Initialize the device counts.  NUM_CI and NUM_ETHERNET
$! are fake device classes created and used locally by autogen.
$!
$ dc_numbers = "1,2,32,65,66,67,96,128,160,161,200,ag1,ag2"
$ dc_names = "DISK,TAPE,SCOM,CARD,TERM,LP,REALTIME,BUS," + -
        "MAILBOX,JOURNAL,MISC,CI,ETHERNET"
$ i = 0
$
$getdata40:
$       number = F$ELEMENT(i,",",dc_numbers)
$       IF number .EQS. "," THEN GOTO getdata50
$       name = F$ELEMENT(i,",",dc_names)
$       dc_'number' = name
$       num_'name' = 0
$       i = i + 1
$       GOTO getdata40
$
$!
$! Loop reading data from the SHOW DEVICES output.  Skip lines that don't
$! contain a device name.  Increment the appropriate device class count for
$! each device that is found.
$!
$getdata50:
$       READ/END_OF_FILE=getdata59 tempfile record
$       IF F$LOCATE(":",record) .EQ. F$LENGTH(record) THEN GOTO getdata50
$       device = F$ELEMENT(0,":",record) - " " + ":"
$       IF .NOT. F$GETDVI(device,"EXISTS") THEN GOTO getdata50
$       devclass = F$GETDVI(device,"DEVCLASS")
$       IF F$TYPE(dc_'devclass') .EQS. "" THEN devclass = 200
$       temp = "NUM_" + dc_'devclass'
$       'temp' = 'temp' + 1
$       IF temp .EQS. "NUM_BUS" -
           THEN IF (F$GETDVI(device,"DEVTYPE") .EQ. 1) .OR. -   ! DT$_CI780
                   (F$GETDVI(device,"DEVTYPE") .EQ. 2) -        ! DT$_CI750
                   THEN num_ci = num_ci + 1
$       IF temp .EQS. "NUM_SCOM" -
           THEN IF (F$GETDVI(device,"DEVTYPE") .EQ. 14) .OR. -  ! DT$_DEUNA
                   (F$GETDVI(device,"DEVTYPE") .EQ. 22) .OR. -  ! DT$_DEQNA
                   (F$GETDVI(device,"DEVTYPE") .EQ. 25) -       ! DT$_DELUA
                   THEN num_ethernet = num_ethernet + 1
$       GOTO getdata50
$
$getdata59:
$ CLOSE tempfile
$
$!
```

```
$! If the number of terminals turned out to be less than expected by the
$! following memory-dependent calculation, then increase the number of
$! terminals to that minimum.  This test is used to cover the possibility
$! that the machine we are looking at is an unterminaled node in a cluster
$! or employs some sort of terminal concentrator that hides the terminals
$! from our scrutiny.
$!
$!      num_term = 8 * memsize (in megs), limited by a cpu specific number
$!
$!          cpu number      cpu name        # of terminals
$!          ----------      --------        --------------
$!              0           unknown              100
$!              1               780              100
$!              2               750               50
$!              3               730               25
$!              4               790              250
$!              5               8SS              100
$!              6               8NN              250
$!              7               UV1                2
$!              8               UV2                2
$!
$ templ = "100,100,50,25,250,100,250,1,1"
$ templ = F$ELEMENT(cputype,",",templ)
$ temp = (6 * memsize) / 2000
$ IF templ .LT. temp THEN temp = templ
$ IF num_term .LT. temp THEN num_term = temp
$
$!
$! Write the device type counts into the data file.
$!
$ i = 0
$getdata60:
$       name = F$ELEMENT(i,",",dc_names)
$       IF name .EQS. "," THEN GOTO getdata69
$       WRITE data "NUM_",name,"=",num_'name'
$       i = i + 1
$       GOTO getdata60
$
$getdata69:
$!
```

```
$!
$! Determine how much of nonpaged pool is being used up by device drivers.
$! Use SYSGEN SHOW/DRIVER command to get the list of all the drivers on the
$! system and how much memory each uses.
$!
$ ON ERROR THEN GOTO common_outerr
$ file = "SYS$SYSTEM:AUTOGEN.TMP"
$
$ RUN sys$system:sysgen
set/output=SYS$SYSTEM:AUTOGEN.TMP
show/driver
exit
$ ON ERROR THEN GOTO common_err
$
$!
$! Skip past the first two lines in the output file.  Then, for each line
$! in the file, increment the running total by that driver's consumption of
$! pool.
$!
$ file = "SYS$SYSTEM:AUTOGEN.TMP"
$ OPEN/READ/ERROR=common_inerr tempfile 'file'
$ READ tempfile record
$ READ tempfile record
$ driver_npagedyn = 0
$
$!
$! Loop reading data from the SHOW/DRIVER output.
$!
$getdata70:
$       READ/END_OF_FILE=getdata79 tempfile record
$       record = F$EDIT(record,"TRIM,COMPRESS,UPCASE")
$       temp = F$ELEMENT(0," ",record)
$       IF (temp .EQS. "OPERATOR") .OR. (temp .EQS. "NL") .OR. -
                (temp .EQS. "MB") THEN GOTO getdata70
$       driver_npagedyn = driver_npagedyn + %X'F$ELEMENT(2," ",record)' - -
                                  %X'F$ELEMENT(1," ",record)'
$       goto getdata70
$
$getdata79:
$ CLOSE tempfile
$ WRITE data "DRIVER_NPAGEDYN=",driver_npagedyn
$
$!
```

```
$!
$! Define the symbols DECNET, CLUSTER, and JOURNALING to be booleans indicating
$! the state of the system.
$!
$ decnet = "false"
$ cluster = "true"
$ IF (num_scom + num_ci) .NE. 0 THEN decnet = "true"
$ IF num_ci .EQ. 0 THEN cluster = "false"
$ journaling = "false"               ! **JNL** Should be journaling = cluster
$ WRITE data "DECNET='""',decnet,'"""'
$ WRITE data "CLUSTER='""',cluster,'"""'
$ WRITE data "JOURNALING='""',journaling,'"""'
$
$!
```

```
$!
$! If we are doing some sort of an upgrade, then get the old sysgen parameters;
$! otherwise, we are all done.
$!
$ IF p3 .EQS. "INITIAL" THEN GOTO getdata120
$
$!
$! Write out values for those parameters from the system being upgraded
$! that are of informational use.
$!
$ file = "SYS$SYSTEM:OLDSITE1.DAT"
$ OPEN/READ/ERROR=common_inerr tempfile 'file'
$ WRITE data "! Parameters specified in ",file
$
$getdata80:
$       READ/END_OF_FILE=getdata89 tempfile record
$       record = F$EDIT(record,"COMPRESS,TRIM")
$       temp = F$EXTRACT(0,1,record)
$       IF temp .LTS. "A" .OR. temp .GTS. "Z" THEN GOTO getdata80
$       IF F$EXTRACT(0,9,record) .EQS. "Parameter" THEN GOTO getdata80
$       WRITE data "OLD_",F$ELEMENT(0," ",record),"=",F$ELEMENT(1," ",record)
$       GOTO getdata80
$
$getdata89:
$ CLOSE tempfile
$
$!
$! Write out values for those parameters from the system being upgraded
$! that should be preserved because they are greater than the new defaults.
$!
$ file = "SYS$SYSTEM:OLDSITE2.DAT"
$ OPEN/READ/ERROR=common_inerr tempfile 'file'
$ WRITE data "! Parameters specified in ",file
$
$getdata90:
$       READ/END_OF_FILE=getdata99 tempfile record
$       record = F$EDIT(record,"COMPRESS,TRIM")
$       temp = F$EXTRACT(0,1,record)
$       IF temp .LTS. "A" .OR. temp .GTS. "Z" THEN GOTO getdata90
$       IF F$EXTRACT(0,9,record) .EQS. "Parameter" THEN GOTO getdata90
$       IF F$ELEMENT(1," ",record) .GT. F$ELEMENT(2," ",record) -
           THEN WRITE data F$ELEMENT(0," ",record),"=",F$ELEMENT(1," ",record)
$       GOTO getdata90
$
$getdata99:
$ CLOSE tempfile
$
$!
$! Write out values for those parameters from the system being upgraded
$! that should be preserved because they are different from the new defaults.
$!
$ delim1 = '"""'
$ delim2 = "'"
$ file = "SYS$SYSTEM:OLDSITE3.DAT"
$ OPEN/READ/ERROR=common_inerr tempfile 'file'
$ WRITE data "! Parameters specified in ",file
$
$getdata100:
$       READ/END_OF_FILE=getdata109 tempfile record
$       rec1 = F$EDIT(record,"COMPRESS,TRIM")
$       temp = F$EXTRACT(0,1,rec1)
$       IF temp .LTS. "A" .OR. temp .GTS. "Z" THEN GOTO getdata100
$       IF F$EXTRACT(0,9,rec1) .EQS. "Parameter" THEN GOTO getdata100
$       IF F$LOCATE(delim1,record) .NE. F$LENGTH(record) THEN GOTO getdata105
$       IF F$LOCATE(delim2,record) .NE. F$LENGTH(record) THEN GOTO getdata106
$       IF F$ELEMENT(1," ",rec1) .NE. F$ELEMENT(2," ",rec1) -
           THEN WRITE data F$ELEMENT(0," ",rec1),"=",F$ELEMENT(1," ",rec1)
$       GOTO getdata100
$
$getdata105:
$       IF F$ELEMENT(1,delim1,record) .NES. F$ELEMENT(3,delim1,record) -
           THEN WRITE data F$ELEMENT(0," ",rec1),"=""",-
                          F$ELEMENT(1,delim1,record),""""
$       GOTO getdata100
$
$getdata106:
$       IF F$ELEMENT(1,delim2,record) .NES. F$ELEMENT(3,delim2,record) -
           THEN WRITE data F$ELEMENT(0," ",rec1),"=""",-
                          F$ELEMENT(1,delim2,record),""""
$       GOTO getdata100
$
$getdata109:
$ CLOSE tempfile
$
$!
$! Write out values for those V4 parameters from the system being upgraded
$! that should be preserved because they are different from the new defaults.
$!
$ IF p3 .EQS. "V3UPGRADE" THEN GOTO getdata120
$ file = "SYS$SYSTEM:OLDSITE4.DAT"
$ OPEN/READ/ERROR=common_inerr tempfile 'file'
$ WRITE data "! Parameters specified in ",file
$
$getdata110:
$       READ/END_OF_FILE=getdata119 tempfile record
$       rec1 = F$EDIT(record,"COMPRESS,TRIM")
$       temp = F$EXTRACT(0,1,rec1)
$       IF temp .LTS. "A" .OR. temp .GTS. "Z" THEN GOTO getdata110
$       IF F$EXTRACT(0,9,rec1) .EQS. "Parameter" THEN GOTO getdata110
$       IF F$LOCATE(delim1,record) .NE. F$LENGTH(record) THEN GOTO getdata115
$       IF F$LOCATE(delim2,record) .NE. F$LENGTH(record) THEN GOTO getdata116
$       IF F$ELEMENT(1," ",rec1) .NE. F$ELEMENT(2," ",rec1) -
           THEN WRITE data F$ELEMENT(0," ",rec1),"=",F$ELEMENT(1," ",rec1)
$       GOTO getdata110
$
$getdata115:
$       IF F$ELEMENT(1,delim1,record) .NES. F$ELEMENT(3,delim1,record) -
           THEN WRITE data F$ELEMENT(0," ",rec1),"=""",-
                          F$ELEMENT(1,delim1,record),""""
$       GOTO getdata110
$
$getdata116:
$       IF F$ELEMENT(1,delim2,record) .NES. F$ELEMENT(3,delim2,record) -
           THEN WRITE data F$ELEMENT(0," ",rec1),"=""",-
                          F$ELEMENT(1,delim2,record),""""
$       GOTO getdata110
$
$getdata119:
$ CLOSE tempfile
$
$!
```

```
$!
$! Write out values for those parameters that the system manager has specified.
$!
$getdata120:
$ IF F$SEARCH("SYS$SYSTEM:MODPARAMS.DAT") .EQS. "" THEN GOTO getdata150
$ file = "SYS$SYSTEM:MODPARAMS.DAT"
$ OPEN/READ/ERROR=common_inerr tempfile 'file'
$ WRITE data "! Parameters specified in ",file
$
$getdata130:
$       READ/END_OF_FILE=getdata139 tempfile record
$       WRITE data record
$       GOTO getdata130
$
$getdata139:
$ CLOSE tempfile
$
$!
```

```
$!
$! Insert warning into the data file.
$!
$getdata150:
$ WRITE data "!"
$ WRITE data "! This data file should NOT be modified.  Users wishing to alter the"
$ WRITE data "! data in this file should modify SYS$SYSTEM:MODPARAMS.DAT instead."
$ WRITE data "!"
$!
```

```
$!
$! Clean up extra files and exit.
$!
$getdata_cleanup:
$ ON ERROR THEN GOTO common_err
$ CLOSE data
$ WRITE sys$output "%AUTOGEN-I-NEWFILE, A new version of SYS$SYSTEM:PARAMS.DAT has been created."
$ WRITE sys$output "        You may wish to purge this file."
$ DEFINE/USER sys$error nl:
$ DEFINE/USER sys$output nl:
$ DELETE sys$system:autogen.tmp;*
$ WRITE sys$output "%AUTOGEN-I-END, ",phase," phase has succesfully completed."
$ IF p2 .EQS. "GETDATA" THEN GOTO common_exit
$ GOTO genparams
$
$!
$! Cleanup after errors and CTRL/Ys.
$!
$getdata_abort:
$ ON CONTROL_Y THEN GOTO getdata_abort
$ ON ERROR THEN CONTINUE
$ CLOSE/NOLOG tempfile
$ CLOSE/NOLOG data
$ DEFINE/USER sys$error nl:
$ DEFINE/USER sys$output nl:
$ DELETE sys$system:autogen.tmp;*
$ WRITE sys$output "%AUTOGEN-I-BADFILE, Bad versions of SYS$SYSTEM:PARAMS.DAT may exist."
$ WRITE sys$output "        We recommend that you delete all versions and start again."
$ GOTO common_'quit'90
$!
```

```
$!++
$!
$! Module:      GENPARAMS
$!
$! Abstract:    This procedure generates new sysgen parameters and (optionally)
$!              a new list of VMS images to install.  The site-specific
$!              requirements file SYS$SYSTEM:PARAMS.DAT is the only input.
$!              The command procedure SYS$SYSTEM:SETPARAMS.DAT and the images
$!              data file SYS$MANAGER:VMSIMAGES.DAT are the outputs.
$!
$!--
$!
$! Initialize this phase.
$!
$genparams:
$ DELETE/SYMBOL/LOCAL/ALL
$ phase = "GENPARAMS"
$ ON CONTROL_Y THEN GOTO common_abort
$ ON ERROR THEN GOTO common_err
$ p1 = F$LOGICAL("AUTOGEN$P1")
$ p2 = F$LOGICAL("AUTOGEN$P2")
$ p3 = F$LOGICAL("AUTOGEN$P3")
$ WRITE sys$output "%AUTOGEN-I-BEGIN, ",phase," phase is beginning."
$
$!
$! Get system configuration data from SYS$SYSTEM:PARAMS.DAT.
$!
$ file = "SYS$SYSTEM:PARAMS.DAT"
$ OPEN/READ/ERROR=common_inerr params 'file'
$
$genparams10:
$       READ/END_OF_FILE=genparams19 params record
$       'record'
$       GOTO genparams10
$
$genparams19:
$ CLOSE params
$
$!
$! Issue warning message if PARAMS.DAT file does not match the current system.
$!
$ IF F$GETSYI("SID") .EQ. sid THEN GOTO genparams20
$ WRITE sys$output "%AUTOGEN-W-SID, SID register indicates that GETDATA phase was performed"
$ WRITE sys$output "        on a different hardware configuration.  GENPARAMS proceeding."
$genparams20:
$
$!
```

```
$!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
$!
$! Calculate values of sysgen parameters.  If a sysgen parameter is already
$! defined in PARAMS.DAT, then the new calculated value is typically ignored
$! in favor of the explicitly specified value.  However, this specified value
$! may be subject to one or more restrictions (which generally appear in the
$! following the command that includes an F$TYPE call).
$!
$!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
$!
$! Calculate MAXPROCESSCNT
$!
$! Require: Nothing
$!
$!++++++++++
$!
$!      MAXPROCESSCNT - number of process entry slots allocated.  Calculate
$!              the value based on the hardware configuration and the
$!              system configuration options specified.  Then round the
$!              total off to a multiple of 5.  If it is greater than 50,
$!              then round it again, to a multiple of 10.
$!
$! ALWAYS:
$! required system processes - NULL, SWAPPER, ERRFMT, JOB_CONTROL, and OPCOM
$! user processes (a guess)  - 1.1 * (the number of terminals)
$! device specific processes - 1 = factor for unexpected files-11 or user ACPs
$!                             1 = assume one print symbiont, though it is
$!                                 up to the system manager to decide how many
$!                                 (up to 16) printers he assigns per symbiont
$!                             one mag tape ACP for every 4 tape drives
$!
$! IF DECNET:
$! required DECnet processes - NETACP, REMACP, EVL, and a fudge factor of one
$! user processes (a guess)  - (the number of terminals) / 20
$!
$! IF CLUSTER:
$! required cluster processes - CONFIGURE and CLUSTER_SERVER.
$!
$! IF JOURNALING:
$! required journaling processes - JNLRCP and JNLACP.
$!
$ temp = 7 + num_term + (num_term/10) + (num_tape + 3)/4
$ IF decnet THEN temp = temp + 4 + (num_term/20)
$ IF cluster THEN temp = temp + 2
$ IF journaling THEN temp = temp + 2
$ temp = ((temp + 4)/5) * 5
$ IF temp .GT. 50 THEN temp = ((temp + 5)/10) * 10
$!
$! Compare the number we just calculated to the old value and use the larger
$! number
$!
$ IF F$TYPE(old_maxprocesscnt) .EQS. "" THEN old_maxprocesscnt = 0
$ IF F$TYPE(maxprocesscnt) .NES. "" THEN old_maxprocesscnt = 0
$ IF F$TYPE(maxprocesscnt) .EQS. "" THEN maxprocesscnt = temp
$ IF maxprocesscnt .LT. old_maxprocesscnt THEN maxprocesscnt = old_maxprocesscnt
$ IF F$TYPE(add_maxprocesscnt) .NES. "" -
    THEN maxprocesscnt = maxprocesscnt + add_maxprocesscnt
$!
```

```
$!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
$!
$! Calculate VIRTUALPAGECNT
$!
$! Require: Nothing
$!
$!++++++++++
$!
$!      VIRTUALPAGECNT - max number of virtual pages that can be mapped
$!              for any one process.  Set to memory size + 3000, with a
$!              lower limit of 8192.
$!
$ temp = memsize + 3000
$ IF temp .LT. 8192 THEN temp = 8192
$!
$! Compare the number we just calculated to the old value and use the larger
$! number.
$!
$ IF F$TYPE(old_virtualpagecnt) .EQS. "" THEN old_virtualpagecnt = 0
$ IF F$TYPE(virtualpagecnt) .NES. "" THEN old_virtualpagecnt = 0
$ IF F$TYPE(virtualpagecnt) .EQS. "" THEN virtualpagecnt = temp
$ IF virtualpagecnt .LT. old_virtualpagecnt THEN virtualpagecnt = old_virtualpagecnt
$ IF F$TYPE(add_virtualpagecnt) .NES. "" -
     THEN virtualpagecnt = virtualpagecnt + add_virtualpagecnt
$!
```

```
$!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
$!
$! Generate VMSIMAGES.DAT.
$!
$! Require: MAXPROCESSCNT.
$!
$!++++++++++
$!
$! Execute the GENIMAGES step.
$!
$ IF (F$TYPE(vmsimages_gblpages) .EQS. "") .OR. -
        (F$TYPE(vmsimages_gblsections) .EQS. "") -
 THEN GOTO genimages
$genimages_return:
$
$!
$! Count the total number of global pages and sections that we need
$! for the images that we are installing in VMSIMAGES.DAT.
$!
$ tgblpages = 0
$ tgblsections = 0
$
$ IF (F$TYPE(vmsimages_gblpages) .NES. "") .AND. -
        (F$TYPE(vmsimages_gblsections) .NES. "") -
  THEN GOTO genparams35
$
$ file = "SYS$MANAGER:VMSIMAGES.DAT"
$ OPEN/READ/ERROR=common_inerr images 'file'
$
$genparams30:
$       READ/END_OF_FILE=genparams39 images record
$       IF F$EXTRACT(0,1,record) .EQS. "!" THEN GOTO genparams30
$       record = F$EDIT(record,"TRIM,COLLAPSE,UPCASE")
$       IF F$LOCATE("/SHARED",record) .EQ. F$LENGTH(record) THEN GOTO genparams30
$       record = F$ELEMENT(1,"!",record)
$       tgblsections = tgblsections + F$ELEMENT(0,"/",record)
$       tgblpages = tgblpages + F$ELEMENT(1,"/",record)
$       GOTO genparams30
$
$genparams39:
$ CLOSE images
$
$genparams35:
$ IF F$TYPE(vmsimages_gblpages) .NES. "" THEN tgblpages = vmsimages_gblpages
$ IF F$TYPE(vmsimages_gblsections) .NES. "" THEN tgblsections = vmsimages_gblsections
$
$!
```

```
$!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
$!
$! Global page and section parameters.
$!
$! Require: VMSIMAGES.DAT data.
$!
$!++++++++++
$!
$!      GBLPAGFIL - Maximun number of global page table entries allocated
$!              for RMS global buffers.  Use 1024, the old value, or an
$!              explicitly specified value, whichever is greatest.
$!
$ temp = 1024
$ IF F$TYPE(old_gblpagfil) .EQS. "" THEN old_gblpagfil = 0
$ IF F$TYPE(gblpagfil) .NES. "" THEN old_gblpagfil = 0
$ IF F$TYPE(gblpagfil) .EQS. "" THEN gblpagfil = 1024
$ IF gblpagfil .LT. old_gblpagfil THEN gblpagfil = old_gblpagfil
$ IF F$TYPE(add_gblpagfil) .NES. "" THEN gblpagfil  = gblpagfil + add_gblpagfil
$!++++++++++
$!
$!      GBLPAGES - Number of global page table entries allocated at boot time.
$!              Start with the total of global pages that we need to take care
$!              of installed VMS images, add in global pages for RMS' use, add
$!              add another 2500 pages for general use, and then round
$!              everything off to a multiple of 100.
$!
$ tgblpages = tgblpages + gblpagfil + 2500
$ tgblpages = ((tgblpages + 50) / 100) * 100
$!
$! Compare the number we just calculated to the old value and use the larger
$! number.
$!
$ IF F$TYPE(old_gblpages) .EQS. "" THEN old_gblpages = 0
$ IF F$TYPE(gblpages) .NES. "" THEN old_gblpages = 0
$ IF F$TYPE(gblpages) .EQS. "" THEN gblpages = tgblpages
$ IF gblpages .LT. old_gblpages THEN gblpages = old_gblpages
$ IF F$TYPE(add_gblpages) .NES. "" THEN gblpages  = gblpages + add_gblpages
$!++++++++++
$!
$!      GBLSECTIONS - Number of global sections allocated at boot time.
$!              Start with the total of global sections that we need to
$!              take care of installed VMS images and then add another 100
$!              sections for general use and round everything off to a
$!              multiple of 10.
$!
$ tgblsections = tgblsections + (tgblsections / 3) + 75
$ tgblsections = ((tgblsections + 5) / 10) * 10
$!
$! Compare the number we just calculated to the old value and use the larger
$! number.
$!
$ IF F$TYPE(old_gblsections) .EQS. "" THEN old_gblsections = 0
$ IF F$TYPE(gblsections) .NES. "" THEN old_gblsections = 0
$ IF F$TYPE(gblsections) .EQS. "" THEN gblsections = tgblsections
$ IF gblsections .LT. old_gblsections THEN gblsections = old_gblsections
$ IF F$TYPE(add_gblsections) .NES. "" -
    THEN gblsections = gblsections + add_gblsections
$!++++++++++
$!
$!      KFILSTCNT - Number of known file list heads.  One is needed for
$!              each set of installed images with a different combination
$!              of device name, directory name, and file type.  Sixteen
$!              seems like a good guess.
```

```
$!
$ IF F$TYPE(kfilstcnt) .EQS. "" THEN kfilstcnt = 16
$ IF F$TYPE(add_kfilstcnt) .NES. "" THEN kfilstcnt = kfilstcnt + add_kfilstcnt
$!
```

```
$
$
$
$
$
$
$!
$!
$!
$!
$!
$
$
$
$
$
$
$!
$!
$!
$!
$!
$
$
$!
$!
$!
$!
$!
$!
$
$
$
$
$!
```

```
$!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
$!
$! Request packet parameters.
$!
$! Require: MAXPROCESSCNT
$!
$!++++++++++
$!
$!      SRPSIZE - Size in bytes of small request packets
$!
$ IF F$TYPE(srpsize) .EQS. "" THEN srpsize = 96
$ IF F$TYPE(add_srpsize) .NES. "" THEN srpsize = srpsize + add_srpsize
$!++++++++++
$!
$!      SRPCOUNT - Number of preallocated small request packets
$!              Set 200 + 7 per process + 1 per device.
$!
$ IF F$TYPE(srpcount) .EQS. "" THEN srpcount = 200 + (maxprocesscnt * 7) + -
        num_disk + num_tape + num_scom + num_card + num_term + num_lp + -
        num_realtime + num_bus + num_mailbox + num_journal + num_misc
$ IF F$TYPE(add_srpcount) .NES. "" THEN srpcount = srpcount + add_srpcount
$!++++++++++
$!
$!      SRPCOUNTV - Max size to which SRPCOUNT can be increased.
$!              Use 4 * SRPCOUNT, but require a minimum of 350.
$!
$ temp = srpcount * 4
$ IF temp .LT. 350 THEN temp = 350
$ IF F$TYPE(srpcountv) .EQS. "" THEN srpcountv = temp
$ IF F$TYPE(add_srpcountv) .NES. "" THEN srpcountv = srpcountv + add_srpcountv
$!++++++++++
$!
$!      IRPSIZE - Size in bytes of intermediate request packets
$!              NOT A SYSGEN PARAMETER - SPECIFIED HERE FOR SYMMETRY.
$!
$ IF F$TYPE(irpsize) .EQS. "" THEN irpsize = 196
$ IF F$TYPE(add_irpsize) .NES. "" THEN irpsize = irpsize + add_irpsize
$!++++++++++
$!
$!      IRPCOUNT - Number of preallocated intermediate request packets
$!              Set 100 + 6 per process.
$!
$ IF F$TYPE(irpcount) .EQS. "" THEN irpcount = 100 + (maxprocesscnt * 6)
$ IF F$TYPE(add_irpcount) .NES. "" THEN irpcount = irpcount + add_irpcount
$!++++++++++
$!
$!      IRPCOUNTV - Max size to which IRPCOUNT can be increased.
$!              Use 4 * IRPCOUNT, but require a minimum of 200.
$!
$ temp = irpcount * 4
$ IF temp .LT. 200 THEN temp = 200
$ IF F$TYPE(irpcountv) .EQS. "" THEN irpcountv = temp
$ IF F$TYPE(add_irpcountv) .NES. "" THEN irpcountv = irpcountv + add_irpcountv
$!++++++++++
$!
$!      LRPSIZE - Size in bytes of large request packets.  Use 1504 when the
$!              XEDRIVER is the only communications driver present on the
$!              system.
$!
$ temp = 576
$ IF num_ethernet .EQ. (num_scom + num_bus) THEN temp = 1504
$ IF F$TYPE(lrpsize) .EQS. "" THEN lrpsize = temp
$ IF F$TYPE(add_lrpsize) .NES. "" THEN lrpsize = lrpsize + add_lrpsize
```

```
$!++++++++++
$!
$!      LRPCOUNT - Number of preallocated large request packets
$!              Set 2 for general use plus a few for each communications
$!              device.  If we have little memory, restrict to 24 overall.
$!
$ temp = 2 + -
        (num_scom * 4) + ((num_scom .GE. 2) * 12) + -
        (num_ci * 8) + ((num_ci .NE. 0) * 16)
$ IF memsize .LT. 2048 .AND. temp .GT. 24 THEN temp = 24
$ IF F$TYPE(lrpcount) .EQS. "" THEN lrpcount = temp
$ IF F$TYPE(add_lrpcount) .NES. "" THEN lrpcount = lrpcount + add_lrpcount
$!++++++++++
$!
$!      LRPCOUNTV - Max size to which LRPCOUNT can be increased.
$!              Use 4 * LRPCOUNT, but require a minimum of 60.
$!
$ temp = lrpcount * 4
$ IF temp .LT. 60 THEN temp = 60
$ IF F$TYPE(lrpcountv) .EQS. "" THEN lrpcountv = temp
$ IF F$TYPE(add_lrpcountv) .NES. "" THEN lrpcourtv = lrpcountv + add_lrpcountv
$!
```

```
$!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
$!
$! Non-paged pool parameters.
$!
$! Require: MAXPROCESSCNT
$!
$!++++++++++
$!
$!      LOAD_CODE - Internal variable indicating the total number of pages
$!              of exec code loaded into nonpaged pool.
$!
$!      REQUIRED:
$!                      SYSLOAxxx.EXE           25
$!
$!      IF JOURNALING:  CJFLOA.EXE              99
$!                      RUFLOA.EXE              20
$!
$!      IF CLUSTER:     CLUSTRLOA.EXE           41
$!                      SCSLOA.EXE              8
$!
$ load_code = 25
$ IF journaling THEN load_code = load_code + 99 + 20
$ IF cluster THEN load_code = load_code + 41 + 8
$!++++++++++
$!
$!      NPAGEDYN - Size of nonpaged dynamic pool in bytes.  Take the
$!              following major factors into account: drivers, processes
$!              (PCB, JIB, I/O, locks), terminals (TTYUCB + .5 TYPAHD), other
$!              device (UCB), loadable code, and misc. I/O data structures
$!              and files (FCB, WCB).
$!
$ IF F$TYPE(npagedyn) .EQS. "" -
    THEN npagedyn = driver_npagedyn + -
                        (maxprocesscnt * (336 + 102 + 200 + 512)) + -
                        (num_term * (308 + 400/2)) + -
                        ((num_disk + num_tape + num_card + -
                                num_lp + num_realtime + num_bus + -
                                num_journal + num_misc) * 150) + -
                        (load_code * 512) + -
                        (num_scom * 2000) + -
                        80000
$ IF F$TYPE(add_npagedyn) .NES. "" THEN npagedyn = npagedyn + add_npagedyn
$!++++++++++
$!
$!      NPAGEVIR - Max size to which NPAGEDYN can be increased.
$!              Use 3 * NPAGEDYN.
$!
$ IF F$TYPE(npagevir) .EQS. "" THEN npagevir = npagedyn * 3
$ IF F$TYPE(add_npagevir) .NES. "" THEN npagevir = npagevir + add_npagevir
$!
```

```
$!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
$!
$! BALSETCNT
$!
$! Require: MAXPROCESSCNT, VIRTUALPAGECNT, SRP*, IRP*, LRP*, NPAGEDYN
$!
$!++++++++++
$!
$!      BALSETCNT - Number of balance set slots in the system page table.
$!              Use a stepwise linear function of MAXPROCESSCNT - 90% of
$!              MAXPROCESSCNT up to 50 and 30% of the number over 50.
$!              Then limit that result by requiring that 100 pages of physical
$!              memory be available for each balance set slot after system
$!              memory usage has been accounted for.  We use the following
$!              equation to define this limit:
$!
$!              Physical memory - system use - process use > BSC * 100
$!
$!              MEMSIZE - (NPAGEDYN/512 + (iRPSIZE * iRPCOUNT)/512)
$!                      - (BSC * (VPC/128))/128 > BSC * 100
$!
$!              We also impose a minimum of 10.
$!
$ temp = (90 * maxprocesscnt) / 100
$ IF maxprocesscnt .GT. 50 THEN temp = 45 + (30 * (maxprocesscnt - 50)) / 100
$
$ temp1 = ((srpsize * srpcount)+(irpsize * irpcount)+(lrpsize * lrpcount)) / 512
$ temp1 = (memsize - npagedyn/512 - temp1) / (100 + (virtualpagecnt / (128*128)))
$ IF temp1 .LT. temp THEN temp = temp1
$ IF temp .LT. 10 THEN temp = 10
$
$ IF F$TYPE(balsetcnt) .EQS. "" THEN balsetcnt = temp
$ IF F$TYPE(add_balsetcnt) .NES. "" THEN balsetcnt = balsetcnt + add_balsetcnt
$!
```

M 3

```
$!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
$!
$! Setting of the ACP parameters
$!
$! Require: MAXPROCESSCNT, BALSETCNT
$!
$!++++++++++
$!
$!      ACP_MULTIPLE - Enables or disables the default creation of a separate
$!              disk ACP for each volume mounted on a different device type.
$!              By default, disable mulitple ACPs.
$!
$ IF F$TYPE(acp_multiple) .EQS. "" THEN acp_multiple = 0
$ IF F$TYPE(add_acp_multiple) .NES. "" -
    THEN acp_multiple = acp_multiple + add_acp_multiple
$!++++++++++
$!
$!      ACP_DIRCACHE - Number of pages for caching directory blocks.  Set
$!              two pages per balance set slot, with a minimum of 20 overall.
$!
$ temp = balsetcnt * 2
$ IF temp .LT. 20 THEN temp = 20
$ IF F$TYPE(acp_dircache) .EQS. "" THEN acp_dircache = temp
$ IF F$TYPE(add_acp_dircache) .NES. "" -
    THEN acp_dircache = acp_dircache + add_acp_dircache
$!++++++++++
$!
$!      ACP_DINDXCACHE - Number of pages for caching directory indices.
$!              Use 1/4 of ACP_DIRCACHE.
$!
$ IF F$TYPE(acp_dindxcache) .EQS. "" THEN acp_dindxcache = acp_dircache/4
$ IF F$TYPE(add_acp_dindxcache) .NES. "" -
    THEN acp_dindxcache = acp_dindxcache + add_acp_dindxcache
$!++++++++++
$!
$!      ACP_HDRCACHE - Number of pages for caching file header blocks.  Set
$!              two pages per balance set slot, with a minimum of 20 overall.
$!
$ temp = balsetcnt * 2
$ IF temp .LT. 20 THEN temp = 20
$ IF F$TYPE(acp_hdrcache) .EQS. "" THEN acp_hdrcache = temp
$ IF F$TYPE(add_acp_hdrcache) .NES. "" -
    THEN acp_hdrcache = acp_hdrcache + add_acp_hdrcache
$!++++++++++
$!
$!      ACP_MAPCACHE - Number of pages for caching bit map blocks.
$!              Set two per disk, with a minimum of 8 overall.
$!              Don't let it get larger than ACP_HDRCACHE.
$!
$ temp = num_disk * 2
$ IF temp .LT. 8 THEN temp = 8
$ IF temp .GT. acp_hdrcache THEN temp = acp_hdrcache
$ IF F$TYPE(acp_mapcache) .EQS. "" THEN acp_mapcache = temp
$ IF F$TYPE(add_acp_mapcache) .NES. "" -
    THEN acp_mapcache = acp_mapcache + add_acp_mapcache
$!++++++++++
$!
$!      ACP_QUOCACHE - Number of quota file entries cached.  Set one entry
$!              per process.
$!
$ IF F$TYPE(acp_quocache) .EQS. "" THEN acp_quocache = maxprocesscnt
$ IF F$TYPE(add_acp_quocache) .NES. "" -
    THEN acp_quocache = acp_quocache + add_acp_quocache
```

```
$!++++++++++
$!
$!      ACP_SYSACC - Number of directory FCBs to cache for disks mounted
$!              /SYSTEM.  Set BALSETCNT/NUM_DISK since we expect an even
$!              load across all system disks.  Use a minimum of 4 overall.
$!
$ temp = balsetcnt / num_disk
$ IF temp .LT. 4 THEN temp = 4
$ IF F$TYPE(acp_sysacc) .EQS. "" THEN acp_sysacc = temp
$ IF F$TYPE(add_acp_sysacc) .NES. "" -
     THEN acp_sysacc = acp_sysacc + add_acp_sysacc
$!++++++++++
$!
$!      ACP_SWAPFLGS - Enable or disables swapping for four classes of ACPs
$!              (/SYSTEM = 0, /GROUP = 1, private = 2, mag tape = 3).
$!              By default, allow swapping for all four classes.  If we
$!              have more than 1 Mb of memory, then disable swapping for
$!              system disks.  If we have exactly 1 Mb of memory, then
$!              disable swapping for system disks only if the BALSETCNT
$!              is greater than 14.
$!
$ temp = 15
$ IF memsize .GT. 2048 THEN temp = 14
$ IF memsize .EQ. 2048 .AND. balsetcnt .GT. 24 THEN temp = 14
$ IF F$TYPE(acp_swapflgs) .EQS. "" THEN acp_swapflgs = temp
$ IF F$TYPE(add_acp_swapflgs) .NES. "" -
     THEN acp_swapflgs = acp_swapflgs + add_acp_swapflgs
$!
```

```
$!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
$!
$! PAGEDYN
$!
$! Require: NPAGEDYN, ACP_DIRCACHE, ACP_MAPCACHE, ACP_HDRCACHE, BALSETCNT
$!
$!++++++++++
$!
$!      PAGEDYN - Size of paged dynamic pool in bytes.  Major consumers are
$!              global section descriptors, information for /OPEN and /HEADER
$!              known files, known file list head info, ACL data, and
$!              shared logical name tables.  Use 1/4 of NPAGEDYN up to 100,000
$!              and 1/6 of the remainder.  Note that this magic calculation
$!              can simply be viewed as a convenient load factoring equation
$!              that comes up with reasonable results.
$!
$!              Add in 2 pages per process to help offset the overhead
$!              incurred by job logical name tables.
$!
$!              Add in a factor to take XQP caching into account and don't
$!              let PAGEDYN get smaller than 80000.
$!
$ temp = npagedyn / 4
$ IF npagedyn .GT. 100000 THEN temp = 25000 + ((npagedyn - 100000) / 6)
$ temp = temp + (balsetcnt * 2 * 512)
$ temp1 = ((512 * (acp_dircache + acp_mapcache + acp_hdrcache)) * 11)/10
$ IF acp_multiple .NE. 0 THEN temp1 = 3 * temp1
$ temp = temp + temp1
$ IF temp .LT. 80000 THEN temp = 80000
$ IF F$TYPE(pagedyn) .EQS. "" THEN pagedyn = temp
$ IF F$TYPE(add_pagedyn) .NES. "" THEN pagedyn = pagedyn + add_pagedyn
$!
```

```
$!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
$!
$! Paging parameters.
$!
$! Require: GBLPAGES, PAGEDYN
$!
$!++++++++++
$!
$!      WSMAX - Max number of pages for any working set.  Use one quarter
$!              of physical memory, rounded off to the nearest 100 and bounded
$!              by 300 and 65000 (size must fit in a word for SYSBOOT).
$!              The result should leave plenty of space for the system.
$!
$ temp = (((memsize / 4) + 50) / 100) * 100
$ IF temp .LT. 300 THEN temp = 300
$ IF temp .GT. 65000 THEN temp = 65000
$ IF F$TYPE(wsmax) .EQS. "" THEN wsmax = temp
$ IF F$TYPE(add_wsmax) .NES. "" THEN wsmax = wsmax + add_wsmax
$!++++++++++
$!
$!      SPTREQ - Number of system page table entries required for mapping
$!              the following system images and data structures.
$!
$!                      SYS.EXE                 305
$!                      RMS.EXE                 199
$!                      SYSMSG.EXE              250
$!                      I/O DATA STRCUTUREs     about 130
$!                      LOAD CODE               50
$!                      8NN SPECIFIC CODE       86
$!
$ temp = 305 + 199 + 250 + 130 + 50
$ IF cputype .EQ. 6 THEN temp = temp + 86
$ IF F$TYPE(sptreq) .EQS. "" THEN sptreq = temp
$ IF F$TYPE(add_sptreq) .NES. "" THEN sptreq = sptreq + add_sptreq
$!++++++++++
$!
$!      SYSMWCNT - Quota for the size of the system working set. Allow one
$!              one page for every 128 global pages, 1/3 of a page for every
$!              required system page table page, and 1/2 of a page for every
$!              page required for paged pool.  Do not let the system manager
$!              choose a value smaller than this.
$!
$ temp = (gblpages / 128) + (sptreq / 3) + (pagedyn / (512 * 2))
$ IF F$TYPE(sysmwcnt) .EQS. "" THEN sysmwcnt = temp
$ IF F$TYPE(add_sysmwcnt) .NES. "" THEN sysmwcnt = sysmwcnt + add_sysmwcnt
$!
```

```
$!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
$!
$! Page fault parameters.
$!
$! Require: Nothing
$!
$!++++++++++
$!
$!      PFRATL - Page fault rate below which a working set limit is
$!              automatically decreased.  Set to zero to allow work set
$!              adjustments to always work automatically.
$!
$ IF F$TYPE(pfratl) .EQS. "" THEN pfratl = 0
$ IF F$TYPE(add_pfratl) .NES. "" THEN pfratl = pfratl + add_pfratl
$!++++++++++
$!
$!      PFCDEFAULT - After a page fault, number of images pages read from
$!              disk, per I/O.  Should not be less than 16.  Set the
$!              value based on relative system disk speed (1,2,4) so that
$!              we don't read in a lot of potentially useless pages on
$!              slow disks.
$!
$ temp = diskspeed * 16
$ IF temp .LT. 16 THEN temp = 16
$ IF F$TYPE(pfcdefault) .EQS. "" THEN pfcdefault = temp
$ IF F$TYPE(add_pfcdefault) .NES. "" -
     THEN pfcdefault = pfcdefault + add_pfcdefault
$
```

```
$!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
$!
$! Free page list parameters.
$!
$! Require: BALSETCNT, MEMSIZE
$!
$!++++++++++
$!
$!      FREELIM - Min number of pages that must be on the free page list.
$!              Use BALSETCNT, but stay between 16 and 64.
$!
$ temp = balsetcnt
$ IF temp .GT. 64 THEN temp = 64
$ IF temp .LT. 16 THEN temp = 16
$ IF F$TYPE(freelim) .EQS. "" THEN freelim = temp
$ IF F$TYPE(add_freelim) .NES. "" THEN freelim = freelim + add_freelim
$!++++++++++
$!
$!      FREEGOAL - Number of pages to try and keep on the free page list.
$!              Use maximum of 3 * FREELIM and 1% of MEMSIZE.  Must be greater
$!              than FREELIM.
$!
$ temp = freelim * 3
$ temp1 = memsize/100
$ IF temp .LT. temp1 THEN temp = temp1
$ IF F$TYPE(freegoal) .EQS. "" THEN freegoal = temp
$ IF F$TYPE(add_freegoal) .NES. "" THEN freegoal = freegoal + add_freegoal
$ IF freegoal .LT. freelim THEN freegoal = freelim
$!++++++++++
$!
$!      GROWLIM - Number of pages that must be on the free page list before a
$!              process that is above quota can add a page to its working set.
$!              Use FREEGOAL - 1 so that working set can be increased at every
$!              opportunity.
$!
$ IF F$TYPE(growlim) .EQS. "" THEN growlim = freegoal - 1
$ IF F$TYPE(add_growlim) .NES. "" THEN growlim = growlim + add_growlim
$!++++++++++
$!
$!      BORROWLIM - Min. number of pages that must be on the free page list
$!              before the system will permit a process to grow past WSQUOTA
$!              for that process.  Should always be greater than FREELIM.
$!
$ temp = 300
$ IF F$TYPE(borrowlim) .EQS. "" THEN borrowlim = temp
$ IF F$TYPE(add_borrowlim) .NES. "" THEN borrowlim = borrowlim + add_borrowlim
$ IF borrowlim .LT. freelim THEN borrowlim = freelim + 100
$!
```

```
$!
$!
$!
$ge
$ I
$ W
$
$!
```

```
$!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
$!
$! Modified page list parameters.
$!
$! Require: BALSETCNT, MEMSIZE
$!
$!++++++++++
$!
$!      MPW_LOLIMIT - Lower limit for the number of pages on the modified
$!              page list.  Use 3 * BALSETCNT, but no more than 120.
$!
$ temp = balsetcnt * 3
$ IF temp .GT. 120 THEN temp = 120
$ IF F$TYPE(mpw_lolimit) .EQS. "" THEN mpw_lolimit = temp
$ IF F$TYPE(add_mpw_lolimit) .NES. "" -
     THEN mpw_lolimit = mpw_lolimit + add_mpw_lolimit
$!++++++++++
$!
$!      MPW_HILIMIT - Upper limit for the number of pages on the modified
$!              page list.  Use maximum of 500 and 2% of MEMSIZE.
$!
$ temp = memsize/50
$ IF temp .LT. 500 THEN temp = 500
$ IF F$TYPE(mpw_hilimit) .EQS. "" THEN mpw_hilimit = temp
$ IF F$TYPE(add_mpw_hilimit) .NES. "" -
     THEN mpw_hilimit = mpw_hilimit + add_mpw_hilimit
$!++++++++++
$!
$!      MPW_WAITLIMIT - Number of pages on the modified page list that will
$!              cause a process to wait until the next time the modified
$!              page writer writes the modified page list.  Make sure that
$!              MPW_WAITLIMIT is greater than or equal to it so that a
$!              system deadlock does not occur.
$!
$ IF F$TYPE(mpw_waitlimit) .EQS. "" THEN mpw_waitlimit = mpw_hilimit
$ IF F$TYPE(add_mpw_waitlimit) .NES. "" -
     THEN mpw_waitlimit = mpw_waitlimit + add_mpw_waitlimit
$ IF mpw_waitlimit .LT. mpw_hilimit THEN mpw_waitlimit = mpw_hilimit
$!
```

```
$!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
$!
$! Lock manager parameters.
$!
$! Require: MAXPROCESSCNT, ACP_DIRCACHE, ACP_HDRCACHE
$!
$!      The general strategy will be to set RESHASHTBL to a value that
$!      will support both LOCKIDTBL and LOCKIDTBL_MAX comfortably.  We then
$!      set LOCKIDTBL potentially low, LOCKIDTBL_MAX potentially high, and
$!      let the automatic adjustments make everything work smoothly.
$!
$!++++++++++
$!
$!      LOCKIDTBL - Number of entries in the system lock id table.
$!              Must be one for each lock in the system.  Assume five
$!              per process.  In addition, in a cluster, add in enough
$!              for XQP caching plus about 150 for installed images and
$!              the job controller.
$!
$ temp = maxprocesscnt * 5
$ temp1 = (acp_dircache/2 + acp_hdrcache)
$ IF acp_multiple .NE. 0 THEN temp1 = 3 * temp1
$ IF cluster THEN temp = temp + temp1 + 150
$ IF F$TYPE(lockidtbl) .EQS. "" THEN lockidtbl = temp
$ IF F$TYPE(add_lockidtbl) .NES. "" THEN lockidtbl = lockidtbl + add_lockidtbl
$!++++++++++
$!
$!      LOCKIDTBL_MAX - Max. number of entries in the system lock id table.
$!              Set to 8 * LOCKIDTBL.
$!
$ temp = lockidtbl * 8
$ IF F$TYPE(lockidtbl_max) .EQS. "" THEN lockidtbl_max = temp
$ IF F$TYPE(add_lockidtbl_max) .NES. "" -
    THEN lockidtbl_max = lockidtbl_max + add_lockidtbl_max
$!++++++++++
$!
$!      RESHASHTBL - Number of entries in the lock management resource
$!              name table.  Pick a number that will allow LOCKIDTBL to
$!              grow comfortably towards LOCKIDTBL_MAX and round up or
$!              down to nearest power of two.
$!
$ oldtemp = 2
$genparams40:
$ temp = oldtemp * 2
$ IF temp .GE. lockidtbl THEN GOTO genparams50
$ oldtemp = temp
$ GOTO genparams40
$genparams50:
$ IF (oldtemp + temp) .GE. (lockidtbl * 2) THEN temp = oldtemp
$ IF F$TYPE(reshashtbl) .EQS. "" THEN reshashtbl = temp
$ IF F$TYPE(add_reshashtbl) .NES. "" -
    THEN reshashtbl = reshashtbl + add_reshashtbl
$!
```

```
$!+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
$!
$! Miscellaneous parameters.
$!
$! Require: MAXPROCESSCNT
$!
$!++++++++++
$!
$!      MAXBUF - Maximum size of buffered I/O transfer.  Should be at
$!              least 1584.
$!
$ IF F$TYPE(old_maxbuf) .EQS. "" THEN old_maxbuf = 0
$ IF F$TYPE(maxbuf) .NES. "" THEN old_maxbuf = 0
$ IF F$TYPE(maxbuf) .EQS. "" THEN maxbuf = 1584
$ IF maxbuf .LT. old_maxbuf THEN maxbuf = old_maxbuf
$ IF F$TYPE(add_maxbuf) .NES. "" THEN maxbuf = maxbuf + add_maxbuf
$!++++++++++
$!
$!      LONGWAIT - Time before swapper considers a process to be idle.
$!              Set to 30 if no old value was specified.
$!
$ IF F$TYPE(longwait) .EQS. "" THEN longwait = 30
$ IF F$TYPE(add_longwait) .NES. "" THEN longwait = longwait + add_longwait
$!++++++++++
$!
$!      PIXSCAN - Time before idle process priorities are boosted.
$!              Set to maxprocesscnt/10, but no less than 1.
$!
$ temp = maxprocesscnt/10
$ IF temp .LT. 1 THEN temp = 1
$ IF F$TYPE(pixscan) .EQS. "" THEN pixscan = temp
$ IF F$TYPE(add_pixscan) .NES. "" THEN pixscan = pixscan + add_pixscan
$!
```

```
$!
$! Create SYS$SYSTEM:SETPARAMS.DAT and put the definitions of all the
$! calculated and propagated sysgen parameters into it.
$!
$! Start by calling sysgen to get all the possible parameter names.
$!
$ ON ERROR THEN GOTO common_outerr
$ file = "SYS$SYSTEM:AUTOGEN.TMP"
$ RUN sys$system:sysgen
set/output=SYS$SYSTEM.AUTOGEN.TMP
show/all
show/special
exit
$ ON ERROR THEN GOTO common_err
$!
$! Create the SETPARAMS.DAT sysgen data file.
$!
$ file = "SYS$SYSTEM:SETPARAMS.DAT"
$ OPEN/WRITE/ERROR=common_outerr setparams 'file'
$ WRITE setparams "use default"
$!
$! Write each sysgen parameter that we have calculated or are propagating
$! a value for to the command procedure.  Look in AUTOGEN.TMP for the list
$! of all possible parameter names.
$!
$ delim = '"""'
$ file = "SYS$SYSTEM:AUTOGEN.TMP"
$ OPEN/READ/ERROR=common_inerr params 'file'
$
$genparams60:
$       READ/END=genparams69 params record
$       record = F$EDIT(record,"TRIM,COMPRESS")
$       temp = F$EXTRACT(0,1,record)
$       IF temp .LTS. "A" .OR. temp .GTS. "Z" THEN GOTO genparams60
$       IF F$EXTRACT(0,9,record) .EQS. "Parameter" THEN GOTO genparams60
$       temp = F$ELEMENT(0," ",record)
$       IF F$TYPE('temp') .EQS. "" THEN GOTO genparams60
$       IF F$LOCATE(delim,record) .NE. F$LENGTH(record) THEN GOTO genparams65
$       WRITE setparams "set ",temp," ",'temp'
$       GOTO genparams60
$
$genparams65:
$       WRITE setparams "set ",temp," ",delim,'temp',delim
$       GOTO genparams60
$
$!
$! Terminate and close the SETPARAMS.DAT data file
$!
$genparams69:
$ WRITE setparams "write current"
$ WRITE setparams "write sys$system:autogen.par"
$ WRITE setparams "exit"
$ CLOSE params
$ CLOSE setparams
$!
```

```
$!
$! Clean up extra files and exit.
$!
$genparams_cleanup:
$ ON ERROR THEN GOTO common_err
$ DEFINE/USER sys$error nl:
$ DEFINE/USER sys$output nl:
$ DELETE sys$system:autogen.tmp;*
$ WRITE sys$output "%AUTOGEN-I-NEWFILE, A new version of SYS$SYSTEM:SETPARAMS.DAT has been created."
$ WRITE sys$output "        You may wish to purge this file."
$ WRITE sys$output "%AUTOGEN-I-END, ",phase," phase has succesfully completed."
$ IF p2 .EQS. "GENPARAMS" THEN GOTO common_exit
$ GOTO genfiles
$
$!
$! Cleanup after errors and CTRL/Ys.
$!
$genparams_abort:
$ ON CONTROL_Y THEN GOTO genparams_abort
$ ON ERROR THEN CONTINUE
$ CLOSE/NOLOG params
$ CLOSE/NOLOG setparams
$ CLOSE/NOLOG images
$ DEFINE/USER sys$error nl:
$ DEFINE/USER sys$output nl:
$ DELETE sys$manager:autogen.tmp;*
$ WRITE sys$output "%AUTOGEN-I-BADFILE, Bad versions of SYS$SYSTEM:SETPARAMS.DAT"
$ WRITE sys$output "        and SYS$MANAGER:VMSIMAGES.DAT may exist."
$ WRITE sys$output "        We recommend that you delete all versions and start again."
$ GOTO common_'quit'90
$!
```

```
$!
$! Module:      GENIMAGES
$!
$! Abstract:    This procedure generates the base installed image list which
$!              is supplied as input to the INSTALL utility.  When determining
$!              which attributes to install an image with, the following
$!              trade-offs are made:
$!
$!              /OPEN - permanently resident directory information eliminates
$!                      directory search to locate image -- about 200 bytes of
$!                      permanently resident memory for a window control block
$!                      and a file control block
$!
$!              /HEADER - permanently resident image header saves one disk I/O
$!                      operation per file access -- approximately 1 page of
$!                      paged memory for the image header
$!
$!              /SHARED - multi-user shared access to read-only and non-CRF
$!                      read/write sections is allowed -- global pages and
$!                      global sections.  Further, the cose of global pages
$!                      and global sections is approximately as follows:
$
$                       Global pages - 1/128 global page table page/page,
$!                              4 bytes of permanently resident system
$!                              page table/128 gpt entries, each global
$!                              page table page will probably get locked
$!                              into the system working set
$!
$!                      Global sections - 32 bytes of permanently resident
$!                              global section descriptor/section
$!
$!--
$!
$! Create a boolean symbol to install an image /OPEN /HEADER /SHARED
$! if we expect to have several simultaneous users of the system, i.e.,
$! the MAXPROCESSCNT is greater than 25.  The threshold is purposely set
$! low so as to only catch really small systems.  In general, we want to
$! err on the side of overinstalling.
$!
$genimages:
$ manyusers = " "
$ IF maxprocesscnt .GE. 25 THEN manyusers = "   /open /header /shared "
$
$!
$! Create the VMSIMAGES.DAT data file.
$!
$ file = "SYS$MANAGER:VMSIMAGES.DAT"
$ OPEN/WRITE/ERROR=common_outerr images 'file'
$!
```

```
$!
$! Install VMS images.  The comment after each shared image indicates the
$! number of global sections/number of global pages that the image required
$! as of 9/1/83.
$!
$! Install infrequently used privileged executable images.
$!
$ WRITE images "sys$system:authorize    /priv=(cmkrnl)"
$ WRITE images "sys$system:analimdmp    /priv=(cmexec,cmkrnl)"
$!**JNL**  WRITE images "sys$system:audit       /priv=(cmkrnl)"
$ WRITE images "sys$system:init         /priv=(cmkrnl,phy_io,sysprv)"
$ WRITE images "sys$system:install      /priv=(cmkrnl,sysgbl,prmgbl,shmem)"
$ WRITE images "sys$system:request      /priv=(tmpmbx)"
$ WRITE images "sys$system:shwclstr     /priv=(cmkrnl)"
$!
$! Install frequently used, load related, privileged executable images.
$!
$ WRITE images "sys$system:mail ",      manyusers, "/priv=(sysprv,oper,world,netmbx)    ! 2/116"
$ WRITE images "sys$system:phone",      manyusers, "/priv=(netmbx,oper,prmmbx,world,sysnam)     ! 1/30"
$ WRITE images "sys$system:rtpad",      manyusers, "/priv=(tmpmbx)                          ! 1/53"
$!
$! Install frequently used privileged executable images.
$!
$ WRITE images "sys$system:cdu          /open /header /priv=(cmexec)"
$ WRITE images "sys$system:loginout     /open /header /shared /priv=(cmkrnl,tmpmbx,log_io,sysprv,sysnam,altpri) ! 3/69"
$ WRITE images "sys$system:monitor      /open /header /priv=(cmkrnl)"
$ WRITE images "sys$system:set          /open /header /shared /priv=(cmkrnl,sysprv,tmpmbx)      ! 1/99"
$ WRITE images "sys$system:setp0        /open /header /shared /priv=(cmkrnl,sysprv)             ! 2/7"
$ WRITE images "sys$system:show         /open /header /shared /priv=(cmkrnl,netmbx,world)       ! 5/92"
$ WRITE images "sys$system:submit       /open /header /shared /priv=(tmpmbx)                    ! 1/18"
$!
$! Install checkpoint images.
$!
$!**JNL**  WRITE images "sys$system:chkp0strt   /priv=(cmexec,cmkrnl)"
$!**JNL**  WRITE images "sys$system:chkp1strt"
$!**JNL**  WRITE images "sys$system:chkcancmd   /priv=(sysprv,cmkrnl)"
$!
$! Install non-privileged executable images.
$!
$ WRITE images "sys$system:copy         /open /header /shared           ! 1/43"
$ WRITE images "sys$system:dcl          /open /header /shared           ! 1/119"
$ WRITE images "sys$system:delete       /open /header /shared           ! 1/16"
$ WRITE images "sys$system:directory    /open /header /shared           ! 1/33"
$ WRITE images "sys$system:edt          /open /header /shared           ! 1/6"
$ WRITE images "sys$system:rename       /open /header /shared           ! 1/15"
$ WRITE images "sys$system:search       /open /header /shared           ! 1/16"
$ WRITE images "sys$system:type         /open /header /shared           ! 1/15"
$ WRITE images "sys$system:vmshelp      /open /header /shared           ! 1/5"
$!
$! Install protected shareable images.
$!
$!**JNL**  WRITE images "sys$share:chkpntshr    /open /header /shared /protect  ! 5/5"
$ WRITE images "sys$share:dbgssishr     /open /header /shared /protect          ! 5/12"
$ WRITE images "sys$share:dismntshr     /open /header /shared /protect /nopurge ! 3/4"
$ WRITE images "sys$share:mountshr      /open /header /shared /protect  ! 3/5"
$ WRITE images "sys$share:secureshr     /open /header /shared /protect  ! 4/22"
$!
$! Install non-protected shareable images.
$!
$ WRITE images "sys$share:convshr"
$ WRITE images "sys$share:dcltables     /open /header /shared           ! 1/257"
$ WRITE images "sys$share:dcxshr        /open /header /shared           ! 1/11"
$ WRITE images "sys$share:edtshr        /open /header /shared           ! 1/141"
$ WRITE images "sys$share:fdlshr"
$ WRITE images "sys$share:lbrshr        /open /header /shared           ! 2/54"
$ WRITE images "sys$share:librtl        /open /header /shared           ! 2/94"
$ WRITE images "sys$share:librtl2"
$ WRITE images "sys$share:mthrtl        /open /header /shared           ! 1/115"
$ WRITE images "sys$share:scrshr        /open /header /shared           ! 1/26"
$ WRITE images "sys$share:smgshr        /open /header /shared           ! 1/13"
$ WRITE images "sys$share:sortshr"
$ WRITE images "sys$share:vmsrtl        /open /header /shared           ! 1/20"
$!
$! Install shareable message images.
$!
$!**JNL**  WRITE images "sys$message:cjfmsg"
$!**JNL**  WRITE images "sys$message:chkpntmsg"
$ WRITE images "sys$message:cliutlmsg   /open /header /shared           ! 1/51"
$ WRITE images "sys$message:dbgtbkmsg"
$ WRITE images "sys$message:filmntmsg"
$ WRITE images "sys$message:netwrkmsg"
$ WRITE images "sys$message:orgdevmsg   /open /header /shared           ! 1/66"
$ WRITE images "sys$message:shrimgmsg   /open /header /shared           ! 1/33"
$ WRITE images "sys$message:sysmgtmsg"
$!
$! Install required language support images.
$!
$ WRITE images "sys$share:basrtl        /open /header"
$ WRITE images "sys$share:basrtl2"
$ WRITE images "sys$share:cobrtl        /open /header"
$ WRITE images "sys$share:forrtl        /open /header"
$ WRITE images "sys$share:pasrtl        /open /header"
$ WRITE images "sys$share:plirtl        /open /header"
$ WRITE images "sys$share:rpgrtl"
$ WRITE images "sys$message:pasmsg"
$ WRITE images "sys$message:plimsg"
$ WRITE images "sys$message:rpgmsg"
$!
$! Install these images only if we have lots of memory.
$!
$ IF memsize .LT. 4096 THEN GOTO genimages10
$ WRITE images "sys$library:debug       /open /header /shared           ! 3/556"
$ WRITE images "sys$share:crfshr        /open /header /shared           ! 2/6"
$ WRITE images "sys$share:trace         /open /header /shared           ! 1/6"
$ WRITE images "sys$share:sumshr        /open /header /shared           ! 2/12"
$ WRITE images "sys$system:link         /open /header"
$!
$! Install these optional images only if DECnet will be in use.
$!
$genimages10:
$ IF .NOT. decnet THEN GOTO genimages20
$ WRITE images "sys$system:fal          /open /header /shared           ! 2/29"
$ WRITE images "sys$system:netserver    /open /header /shared           ! 2/16"
$ WRITE images "sys$share:nmlshr        /open /header /shared           ! 2/73"
$!
```

```
$!
$! Write comments in the data file.
$!
$genimages20:
$ WRITE images "!"
$ WRITE images "! This data file is used to install VMS known images. This file MUST NOT "
$ WRITE images "! be modified."
$ WRITE images "!"
$ WRITE images "! Users wishing to alter the list of installed images should remove and "
$ WRITE images "! reinstall the image via the SYSTARTUP.COM mechanism."
$ WRITE images "!"
$ WRITE images "! The presence of a /NOPURGE qualifier on the line indicates that the file"
$ WRITE images "! should not be removed during the SHUTDOWN process."
$ WRITE images "!"
$
$!
$! Cleanup and return control to GENPARAMS
$!
$ CLOSE images
$ WRITE sys$output "%AUTOGEN-I-NEWFILE, A new version of SYS$MANAGER:VMSIMAGES.DAT has been created."
$ WRITE sys$output "        You may wish to purge this file."
$ SET PROTECTION=(system:rwed,owner:rwed,group,world) sys$manager:vmsimages.dat
$ GOTO genimages_return
$!
```

```
$!++
$!
$! Module:       GENFILES and TESTFILES
$!
$. \bstracl:     This procedure generates new paging, swapping, and dump
$!               files for a system.  The site-specific requirements file
$!               SYS$SYSTEM:PARAMS.DAT is input.  Outputs of this operation are
$!               SYS$SYSTEM:PAGEFILE.SYS, SWAPFILE.SYS, and SYSDUMP.DMP.
$!
$!               If TESTFILES was specified as the end phase, then this
$!               procedure displays its results instead of executing them.
$!
$!               MUST be immediately preceded by execution of GENPARAMS.
$!
$!--
$!
$! Initialize this phase.
$!
$testfiles:
$genfiles:
$ phase = "GENFILES"
$ IF p2 .EQS. "TESTFILES" THEN phase = "TESTFILES"
$ ON CONTROL_Y THEN GOTO common_abort
$ ON ERROR THEN GOTO common_err
$ WRITE sys$output "%AUTOGEN-I-BEGIN, ",phase," phase is beginning."
$ verb = "creating"
$ verb1 = "will be"
$ IF p2  EQS. "TESTFILES" THEN verb = "would have created"
$ IF p2 .EQS. "TESTFILES" THEN verb1 = "would have been"
$
$!
$! Set up symbol for single line sysgen invocations
$!
$ SYSGEN = "$SYSGEN"
$!
```

```
$ S
$ E
$NO
$ S
$ E
```

```
$!
$! Calculate PAGEFILE.SYS size.  Allow 2000 blocks for the system and
$! 400 for each process.  If this doesn't total to at least twice the
$! virtual page count, then use that number.
$!
$ temp = 2000 + (400 * maxprocesscnt)
$ IF temp .LT. (virtualpagecnt * 2) THEN temp = virtualpagecnt * 2
$!
$! If we have a small system disk, then just use 4604 (+ 496 if we have
$! more than 2048 pages of memory).
$!
$ IF smalldisk THEN temp = 4604
$ IF smalldisk .AND. (memsize .GT. 2048) THEN temp = temp + 496
$!
$! If a particular pagefile size was specified by the user, then use that.
$!
$ IF F$TYPE(pagefile) .EQS. "" then pagefile = temp
$ IF pagefile .EQ. 0 THEN GOTO genfiles35
$!
$! If the current pagefile size is between 100% and 120% of the size we've
$! just calculated, then leave it alone.
$!
$ temp = 0
$ IF F$SEARCH("sys$system:pagefile.sys") .EQS. "" THEN GOTO genfiles20
$ temp = F$FILE_ATTRIBUTES("sys$system:pagefile.sys","ALQ")
$ IF (temp - (temp / 5) .LE. pagefile) .AND. (temp .GE. pagefile) -
    THEN GOTO genfiles35
$!
$! Output file creation message.
$!
$genfiles20:
$ WRITE sys$output "%AUTOGEN-I-PAGEFILE, ''verb' ''pagefile' block page file."
$ IF phase .EQS. "TESTFILES" THEN GOTO genfiles40
$!
$! If there is not enough room (after salting 1000 blocks away) for the
$! expanded pagefile, then write out an error message.
$!
$ freeblocks = F$GETDVI("sys$sysdevice","FREEBLOCKS")
$ IF (pagefile - temp) .GT. (freeblocks - 1000) THEN GOTO genfiles30
$!
$! Attempt to create the pagefile.  If we didn't succeed, write out an
$! error message.
$!
$ SYSGEN CREATE sys$system:pagefile.sys /SIZE='pagefile'
$ IF F$SEARCH("sys$system:pagefile.sys") .EQS. "" THEN GOTO genfiles30
$ temp = F$FILE_ATTRIBUTES("sys$system:pagefile.sys","ALQ")
$ IF temp .GE. pagefile THEN GOTO genfiles40
$ IF (pagefile - (pagefile / 5)) .LE. temp THEN GOTO genfiles40
$!
$! Write out pagefile error.
$!
$genfiles30:
$ WRITE sys$output "%AUTOGEN-W-OPENOUT, error creating PAGEFILE.SYS.  PAGEFILE.SYS needs"
$ WRITE sys$output "        to be created manually with ",pagefile," blocks"
$ GOTO genfiles40
$!
$! Write out informational message.
$!
$genfiles35:
$ IF phase .NES. "TESTFILES" THEN GOTO genfiles40
$ WRITE sys$output "%AUTOGEN-I-PAGEFILE, No new page file ",verb1," created."
$!
```

```
$!
$! Calculate SWAPFILE.SYS size.  Allow 2000 blocks for the system and
$! WSMAX (up to 300) for each process.
$!
$genfiles40:
$ temp = 300
$ IF wsmax .LT. temp THEN temp = wsmax
$ temp = 2000 + (temp * maxprocesscnt)
$!
$! If we have a small system disk, then just use 1000.  Also, change the
$! name of the file we will be creating to SWAPFILE1.SYS so that installation
$! of the swapping file will be deferred until after the ERRFMT, JOB_CONTROL,
$! and OPCOM processes have been initiated.  That way, on small systems, there
$! is a greater chance that the swapfile will be big enough to handle user
$! processes.
$!
$ IF smalldisk THEN temp = 1000
$ oneflag = ""
$ IF smalldisk THEN oneflag = "1"
$!
$! If a particular swapfile size was specified by the user, then use that.
$! If the specified or calculated value was 100 or less, then don't bother
$! creating the file.
$!
$ IF F$TYPE(swapfile) .EQS. "" THEN swapfile = temp
$ IF swapfile .LT. 100 THEN GOTO genfiles65
$!
$! If the current swapfile size is between 100% and 120% of the size we've
$! just calculated, then leave it alone.
$!
$ temp=0
$ IF F$SEARCH("sys$system:swapfile''oneflag'.sys") .EQS. "" THEN GOTO genfiles50
$ temp = F$FILE_ATTRIBUTES("sys$system:swapfile''oneflag'.sys","ALQ")
$ IF ((temp - (temp / 5)) .LE. swapfile) .AND. (temp .GE. swapfile) -
    THEN GOTO genfiles65
$!
$! Output file creation message.
$!
$genfiles50:
$ WRITE sys$output "%AUTOGEN-I-SWAPFILE, ''verb' ''swapfile' block swap file."
$ IF phase .EQS. "TESTFILES" THEN GOTO genfiles70
$!
$! If there is not enough room (after salting 1000 blocks away) for the
$! expanded swapfile, then write out an error message.
$!
$ freeblocks = F$GETDVI("sys$sysdevice","FREEBLOCKS")
$ if (swapfile - temp) .GT. (freeblocks - 1000) THEN GOTO genfiles60
$!
$! Attempt to create the swapfile.  If we didn't succeed, write out an
$! error message.
$!
$ SYSGEN CREATE sys$system:swapfile'oneflag'.sys /SIZE='swapfile'
$ IF F$SEARCH("sys$system:swapfile''oneflag'.sys") .EQS. "" THEN GOTO genfiles60
$ temp = F$FILE_ATTRIBUTES("sys$system:swapfile''oneflag'.sys","ALQ")
$ IF temp .GE. swapfile THEN GOTO genfiles70
$ IF (swapfile - (swapfile / 5)) .LE. temp THEN GOTO genfiles70
$!
$! Write out swapfile error.
$!
$genfiles60:
$ WRITE sys$output "%AUTOGEN-W-OPENOUT, error creating SWAPFILE",oneflag,".SYS.  SWAPFILE",oneflag,".SYS needs"
$ WRITE sys$output "       to be created manually with ",swapfile," blocks"
$ GOTO genfiles70
```

```
$!
$! Write out informational message.
$!
$genfiles65:
$ IF phase .NES. "TESTFILES" THEN GOTO genfiles70
$ WRITE sys$output "%AUTOGEN-I-SWAPFILE, No new swap file ",verb1," created."
$
$!
```

```
$!
$! Calculate SYSDUMP.DMP size.  Make it four blocks larger than the size
$! of physical memory.
$!
$genfiles70:
$ temp = memsize + 4
$!
$! If we have a small system disk, and the user did not explicitly request
$! the creation of a dumpfile, then don't create one.
$!
$ IF smalldisk .AND. (F$TYPE(dumpfile) .EQS. "") THEN GOTO genfiles95
$!
$! If a particular dumpfile size was specified by the user, then use that.
$! If the specified size was less than 3, then don't bother creating the file.
$!
$ IF F$TYPE(dumpfile) .EQS. "" THEN dumpfile = temp
$ IF dumpfile .LT. 3 THEN GOTO genfiles95
$!
$! If the current dumpfile size is greater than or equal to the size we've
$! just calculated, then leave it alone.
$!
$ temp=0
$ IF F$SEARCH("sys$system:sysdump.dmp") .EQS. "" THEN GOTO genfiles80
$ temp = F$FILE_ATTRIBUTES("sys$system:sysdump.dmp","ALQ")
$ IF temp .GE. dumpfile THEN GOTO genfiles95
$!
$! Output file creation message.
$!
$genfiles80:
$ WRITE sys$output "%AUTOGEN-I-DUMPFILE, ''verb' ''dumpfile' block dump file."
$ IF phase .EQS. "TESTFILES" THEN GOTO genfiles100
$!
$! If there is not enough room (after salting 1000 blocks away) for the
$! expanded dumpfile, then write out an error message.
$!
$ freeblocks = F$GETDVI("sys$sysdevice","FREEBLOCKS")
$ IF (dumpfile - temp) .GT. (freeblocks - 1000) THEN GOTO genfiles90
$!
$! Attempt to create the dumpfile.  If we didn't succeed, write out an
$! error message.
$!
$ SYSGEN CREATE sys$system:sysdump.dmp /SIZE='dumpfile'
$ IF F$SEARCH("sys$system:sysdump.dmp") .EQS. "" THEN GOTO genfiles90
$ temp = F$FILE_ATTRIBUTES("sys$system:sysdump.dmp","ALQ")
$ IF temp .GE. dumpfile THEN GOTO genfiles100
$!
$! Write out dumpfile error.
$!
$genfiles90:
$ WRITE sys$output "%AUTOGEN-W-OPENOUT, error creating SYSDUMP.DMP.  SYSDUMP.DMP needs"
$ WRITE sys$output "        to be created manually with ",dumpfile," blocks"
$ GOTO genfiles100
$!
$! Write out informational message.
$!
$genfiles95:
$ IF phase .NES. "TESTFILES" THEN GOTO genfiles100
$ WRITE sys$output "%AUTOGEN-I-DUMPFILE, No new dump file ",verb1," created."
$
$!
```

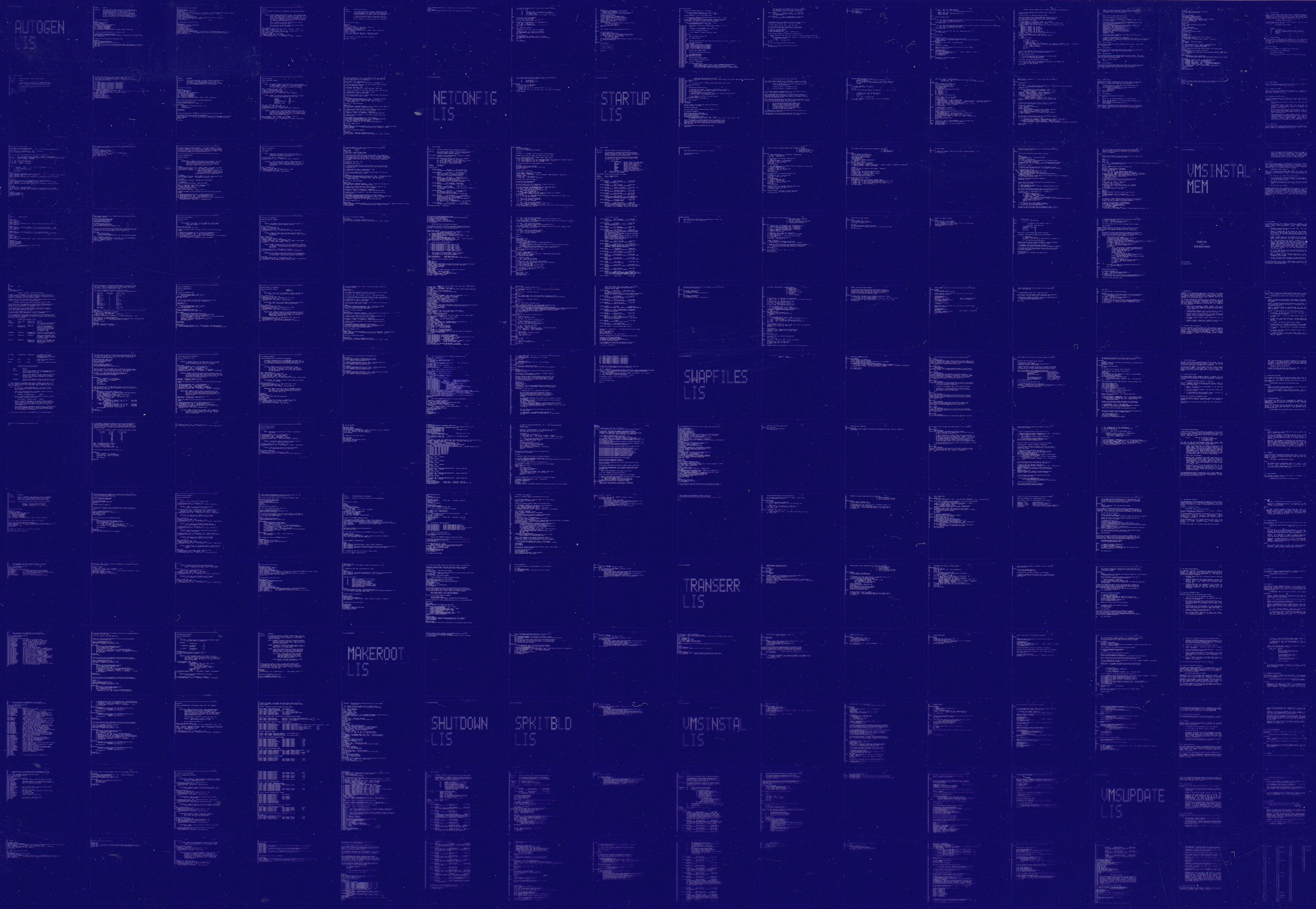
```
$!
$! Set the correct file attributes on the created files.
$!
$genfiles100:
$ IF phase .EQS. "TESTFILES" THEN GOTO genfiles_cleanup
$ SET PROT=(S:RWED,O:RWED,G,W) sys$system:pagefile.sys;*
$ SET FILE /NOBACKUP sys$system:pagefile.sys
$
$ IF F$SEARCH("sys$system:swapfile.sys") .EQS. "" THEN GOTO genfiles110
$ SET PROT=(S:RWED,O:RWED,G,W) sys$system:swapfile.sys;*
$ SET FILE /NOBACKUP sys$system:swapfile.sys
$
$genfiles110:
$ IF F$SEARCH("sys$system:swapfile1.sys") .EQS. "" THEN GOTO genfiles120
$ SET PROT=(S:RWED,O:RWED,G,W) sys$system:swapfile1.sys;*
$ SET FILE /NOBACKUP sys$system:swapfile1.sys
$
$genfiles120:
$ IF F$SEARCH("sys$system:sysdump.dmp") .EQS. "" THEN GOTO genfiles_cleanup
$ SET PROT=(S:RWED,O:RWED,G,W) sys$system:sysdump.dmp;
$ SET FILE /NOBACKUP sys$system:sysdump.dmp
$
$!
```

```
$!
$! Clean up and exit.
$!
$testfiles_cleanup:
$genfiles_cleanup:
$ ON ERROR THEN GOTO common_err
$ WRITE sys$output "%AUTOGEN-I-END, ",phase," phase has succesfully completed."
$ IF p2 .EQS. "GENFILES" THEN GOTO common_exit
$ IF p2 .EQS. "TESTFILES" THEN GOTO common_exit
$ GOTO setparams
$
$!
$! Cleanup after errors and CTRL/Ys.
$!
$testfiles_abort:
$genfiles_abort:
$ ON CONTROL_Y THEN GOTO genfiles_abort
$ ON ERROR THEN CONTINUE
$ GOTO common_'quit'90
$!
```

```
$!++
$!
$! Module:      SETPARAMS, REBOOT, and SHUTDOWN
$!
$! Abstract:    Prepares the system to reboot with the new parameters.
$!
$!--
$!
$! Initialize this phase.
$!
$reboot:
$shutdown:
$setparams:
$ DELETE/SYMBOL/LOCAL/ALL
$ p1 = F$LOGICAL("AUTOGEN$P1")
$ phase = "SETPARAMS"
$ IF (p1 .EQS. "REBOOT") .OR. (p1 .EQS. "SHUTDOWN") -
      THEN phase = p1
$ ON CONTROL_Y THEN GOTO common_abort
$ ON ERROR THEN GOTO common_err
$ p2 = F$LOGICAL("AUTOGEN$P2")
$ p3 = F$LOGICAL("AUTOGEN$P3")
$ IF phase .NES. "SETPARAMS" THEN GOTO end20
$
$!
$! Execute the SETPARAMS command procedure.
$!
$ WRITE sys$output "%AUTOGEN-I-BEGIN, ",phase," phase is beginning."
$ IF F$SEARCH("sys$system:setparams.dat") .NES. "" THEN GOTO end10
$ WRITE sys$output "%AUTOGEN-E-OPENIN, SYS$SYSTEM:SETPARAMS.DAT could not be found."
$ WRITE sys$output "         Please ensure that the GENPARAMS phase successfully completes
$ WRITE sys$output "         before executing the ",phase," phase."
$ GOTO common_err90
$
$end10:
$
$ DEFINE/USER SYS$INPUT sys$system:setparams.dat
$ RUN sys$system:sysgen
$
$!
$! If the specified ending point was SETPARAMS, then we are all done.
$!
$setparams_cleanup:
$ ON ERROR THEN GOTO common_err
$ WRITE sys$output "%AUTOGEN-I-END, ",phase," phase has succesfully completed."
$ IF p2 .EQS. "SETPARAMS" THEN GOTO common_exit
$
$
$!
$! Execute REBOOT or SHUTDOWN phase.
$!
$end20:
$ phase = p2
$ WRITE sys$output "%AUTOGEN-I-BEGIN, ",phase," phase is beginning."
$ WRITE sys$output ""
$ WRITE sys$output "The system is shutting down to allow the V4.0 system to boot with the"
$ WRITE sys$output "generated site-specific parameters and installed images."
$ WRITE sys$output ""
$
$!
$! If SHUTDOWN was specified, then require a manual reboot.
$!
$ IF p2 .EQS. "REBOOT" THEN GOTO boot10
$ rebootflag = "N"
$ WRITE sys$output "You must manually reboot the system after it halts."
$ GOTO boot20
$
$!
$! If REBOOT was specified, then automatically reboot.
$!
$boot10:
$ rebootflag = "Y"
$ WRITE sys$output "The system will automatically reboot after the shutdown and the
$ WRITE sys$output "upgrade to VAX/VMS Version 4.0 will be complete."
$ WRITE sys$output ""
$
$!
$! Shutdown the system.
$!
$!       P1      Number of minutes until final shutdown.
$!       P2      Reason for shutdown.
$!       P3      Should the disk volumes be spun down?
$!       P4      Should SYSHUTDWN.COM be invoked?
$!       P5      Time when system will be rebooted.
$!       P6      Should system be rebooted automatically?
$!       P7      Comma-separated list of keywords.  Legal
$!               keywords: REBOOT_CHECK,CLUSTER_SHUTDOWN,
$!               REMOVE_NODE, and NONE
$!
$boot20:
$ IF F$TYPE(shutdown_time) .EQS. "" THEN shutdown_time = 0
$ @sys$system:shutdown 'shutdown_time' "Reboot system with AUTOGENerated parameters" "N" -
        "Y" "soon" 'rebootflag' "REBOOT_CHECK"
$
$!
$! Clean up extra files and exit.
$!
$shutdown_cleanup:
$reboot_cleanup:
$ ON ERROR THEN GOTO common_err
$ WRITE sys$output "%AUTOGEN-I-END, ",phase," phase has succesfully completed."
$ GOTO common_exit
$
$!
$! Cleanup after errors and CTRL/Ys.
$!
$setparams_abort:
$shutdown_abort:
$reboot_abort:
$ ON CONTROL_Y THEN GOTO reboot_abort
$ ON ERROR THEN CONTINUE
$ GOTO common_'quit'90
```

0232 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



AUTOGEN
LIS

NETCONFIG
LIS

STARTUP
LIS

VMSINSTAL
MEM

SWAPFILES
LIS

TRANSERR
LIS

MAKEROOT
LIS

SHUTDOWN
LIS

SPKITBLD
LIS

VMSINSTAL
LIS

VMSUPDATE
LIS