# Chapter 7
# VMS

**This chapter is looked after by Jim Gilmour**
This chapter now refers only to VMS V4.n and greater.

## 7.1 VMS Boot Troubleshooting

VMS Boot Description

VMB Loading and starting

A description of processor specific boot commands is given later.

When looking at VMS boot routines you will find that all machines load a VMB of some kind. The VMB.EXE file is a raw binary image of memory which is adapted by the Microvax family to load part from ROM and part from boot device. The Microvax family leaves a compatible image which it copies from its own console ROM in memory after a console boot command is executed. This version is a subset of the VMB program found on the larger VAX family. There is no facility to adjust the registers, unlike on the large VAX family. The boot is accomplished by finding the first bootable device. The sequence is tape (MUA0), disk (DUA0), ethernet (ETA0). The MicroVAX looks for SYSBOOT.EXE not VMB.EXE like the large VAX series. This is an important difference and will affect the following description of the boot flow. When the MicroVAX has passed control to SYSBOOT this will be recognised. SYSBOOT will call special code to convert the parameters passed from the ROM VMB to make them identical to those passed by the version of VMB loaded on the larger VAXes.

The VAX 6000 series have console program that requires the path of the device to be given using hexadecimal values, not decimal e.g. dua16 would mean hex 10. Hexadecimal 16 would mean dua22.

The new CIXCD gives a status messages on the console whilst booting. Use these whilst trouble shooting. They are very useful faultfinding tools as they will tell you if control has passed to VMB or not.

The following is an example problem boot from an engineers viewpoint.

Suppose we had a problem where we knew that the cpu was running, but we did not get the first VMS banner. What could we do to see if VMB had managed to read in SYSBOOT and start it? We would set bit 0 in R5 to 1, using the console commands described in section 7.1.2.1 of this Chapter.

If you received the "SYSBOOT>" prompt this would mean that quite a lot of hardware was working because the cpu had managed to read the SYSBOOT image from the boot media. If you did not get the prompt you would know that the cpu could not talk to the system disk and diagnostics would need to be loaded via another boot media. The normal boot path could then be tested.

If the customer has standalone backup installed on another data disk then try booting that disk by changing the unit number you are booting from by overriding the default unit number. When standalone backup boots it "sniffs" out all the disks and tapes on a system. This gives a clear description as it prints out all disk and tape device types across the cluster. This provides useful information on the system boot drive state. If you still want to check the integrity of the system boot disk then back it up. This will read all the blocks and shows if any forced errors have occurred in important boot path images.

In the example below substitute your disk and tape unit names.

```
$ BACKUP/IMAGE/INIT/LABEL=DSKCHK/IGNORE=LABEL disk: tape:dskchk.bck

where disk: is $xxx$Dyzn: and tape: is $x$Myzn:
  xxx is the allocation class
  y is the disk type
  z is the adapter
  n is the unit number
```

Some versions of VMS have been know to hang whilst backing up if the /LOG qualifier is used. Type a Control+T whilst a backup is running to see what file is being currently backed up.

Doing just one volume of tape should be sufficient to prove the drives functionality.

If you can boot backup there is a much more likely case that someone has modified the site specific startup file or even startup.com itself.

The next point in the VMS boot sequence you can intervene is by making an adjustment to the VMS startup file name. To break into the boot sequence at the earliest instance the following needs to be typed at the SYSBOOT prompt.

```
SYSBOOT> SHOW /STARTUP
Startup command file = SYS$SYSTEM:STARTUP.COM
```

This shows the standard startup file name. You will now tell VMS to get all its startup commands from the console. Type these commands to the SYSBOOT prompt.

```
SYSBOOT> SET /STARTUP OPA0:
SYSBOOT> CONTINUE
```

The VMS boot will now continue. If you are on a standalone system then the port select light on RA81/RA82 will come on and go out as various sections of VMS are initialised. Whilst waiting for the dollar prompt to appear, study the disk lights to see the disk activity stops. The boot disk is being rebuilt the same as if you mounted it using the DCL mount command. The length of time will depend on the size of the disk, on average five minutes though on a very busy cluster common disk it could be longer. If you have an HSC use the VTDPY on the HSC console. Remember to stop it after you finish as it uses up resources on a busy HSC.

If your the system is an 11/750 and it hangs after the banner prompt, check the interrupt grant cards in the unibus are all the right way round. This is probably the best known problem on 11/750's and is due to the fact that interrupts are not allowed or needed until the swapper process is run.

When the dollar prompt is printed then type the following commands to stop you being logged out. Typing mistakes will cause you to be logged out and you will have to reboot over again.

$ set noon

$ set noverify

You are now running as the STARTUP process, nothing else is running many commands will fail with obscure error messages. However there are some commands we can use at this point.

To quickly check the file structure.

```
$ ANALYZE/DISK/NOREPAIR disk-name-unit (e.g. DUA0:) will work.
```

To check file structure and read every allocated block.

```
$ ANALYZE/DISK/NOREPAIR/READ disk-name-unit (e.g. DUA0:) will work.
```

Do not use the /REPAIR switch as you cannot flush write cache to disk in this standalone mode. If you have errors run the standalone diagnostics on the disk. The disk BBR programs will probably be the best choice.

This will give the disk a good testing physically and check the file structure.

If you get errors see the SMALL SYSTEMS DEVICES Chapter,of Datadoc, section 16.3 for advice on what to do with the information from a "forced error".

The most common failure of VMS to boot is the device on a bus hanging the system. The system can be checked for this with the following.

```
$ MC SYSGEN
SYSGEN> AUTOCONFIGURE ALL/LOG
```

A list of devices printed as they are configured. If this printout stops and you don't get the SYSGEN prompt back then one of the devices has hung VMS. Beware however the customer may have some non-DEC devices which have to be hand configured and therefore may be incorrectly configured under SYSGEN. The only DEC device I know that does not keep to the rules is device called a KMS11. However there could be more. The rules are defined in the early days of the unibus. When VMS V5.0 came out there were some incompatibilities in the unibus rule algorithms. This is solved from V5.1 onwards. There is a problem with clusters with this method of configuring. There is no configure process running to detect HSC based devices and the RFxx DSSI series of disks which are treated as nodes.

Just supposing the system hangs whilst configuring the DZ11's. Then try the following after rebooting. You must set the startup file to OPA0: every time as it is not written to disk.

```
$ MC SYSGEN
SYSGEN> AUTOCONFIGURE ALL/LOG/EXCLUDE=(TT)
```

Everything except the DZ11's will be excluded. If you get the SYSBOOT prompt back you have proved it is the DZ11's. Do the same command again and exclude a specific device (e.g. TTA0).

```
$ MC SYSGEN
SYSGEN> AUTOCONFIGURE ALL/LOG/EXCLUDE=(TTA)
```

If this does not hang the system then you have found the device that is hanging the system.

The next common problem in VMS is that the someone has altered the site specific startup . This is different for V4.X and V5.X.

```
Version 4
SYS$MANAGER:SYSTARTUP.COM

Version 5
SYS$MANAGER:SYSTARTUP_V5.COM
```

There is a parameter settable in SYSBOOT which will tell the STARTUP process not to use the site specific file and just startup VMS to the minimum. There is always confusion around this parameter on how to set and clear it.

To startup VMS in minimum configuration without the site specific startup file use the following commands.

```
SYSBOOT> SET STARTUP_P1 "MIN"
SYSBOOT> CONTINUE
```

This will open an operator log and not much else,but all the normal commands will work. You should now be able to log in and see all the connected DSSI drives.

This new value will be silently written to the VMS parameters file. This means that if the system is not going to be rebooted to clear this parameter you must be the system Manager to clear this value, otherwise the system come up in minimum boot again.

THE SYSTEM MANAGER SHOULD DO THESE COMMANDS.

```
$ MC SYSGEN
SYSGEN> USE ACTIVE
SYSGEN> SHOW STARTUP_P1
Parameter   Current  Default
=====================================================
STARTUP_P1    "MIN"   "   "
SYSGEN> SET STARTUP_P1 ""
SYSGEN> SHOW STARTUP_P1
Parameter   Current  Default
=====================================================
STARTUP_P1    "   "   "   "
SYSGEN> WRITE ACTIVE
SYSGEN> WRITE CURRENT
SYSGEN> EXIT
```

When the write current command is done a message appears on OPA0: to say that the system parameters have been modified.

If you now start SYSGEN again, as described above, the STARTUP_P1 will still be set to "MIN". This is because it is not a dynamic parameter and is only changed on reboot. To actually see what the parameter is the "USE CURRENT" command is used instead of the "USE ACTIVE" command.

If you suspect something in the site specific file is stopping VMS coming up fully use the above procedure to modify STARTUP_P2 to TRUE. This will print everything that is happening from the start of STARTUP.COM right through the site specific startup file. The log will be printed on the console. It is worth pointing out that some customers cluster systems can take an hour to reach the end of their startup. This is because they start large databases and set up hundreds of terminals and batch queues. Ask the customer how long it normally takes. DON'T start up the system fully without the customer knowledge as they may have made changes to things due to abnormal operation of the system you are looking at. It is best to have the customer there so that both of you can see the problem.

### 7.1.1 Booting a VAXcluster Member over the Ethernet

To allow a system to boot into a Local Area VAXcluster all members of the cluster must be enabled for Ethernet LAVC communication from VMS 5.3 onwards. If even one in the cluster node is not enabled for LAVC communications the node attempting to boot will hang. The booting node is generally known as a satellite node. When the processor starts to boot via ethernet its send out Maintenance Operations Protocol (MOP) messages. One node in the cluster will service this satellite. The other nodes will back off with abort messages from DECnet. The DECnet opcom messages are a source good information on what is happening. The customer will have to update his configuration files if you have changed the ethernet controller modules address on the satellite node or it wont boot.

To see DECnet opcom messages at you terminal type the following commands on you terminal.

```
$ set process/privledge=oper
$ reply/enable=net
```

The most common source of failure to boot is that the host machines ethernet controller is not set service enabled. This was probably set correctly but modified at a later date. This is not noticed until the next reboot of either host or satellite because MOP is only used on boot up. Service enabled means to allow MOP messages to be acted upon. If there are not DECnet opcom messages about load request appearing then MOP messages are being ignored.

The following command will show the state of the any ethernet line.

```
$ MC NCP
NCP> show known line characteristics

Line = UNA-0    (a Deuna was found in this example)

Service   = enabled
Receive Buffers = 10
Controller   = normal
Protocol   = Ethernet
Service timer = 4000
Hardware address = AA-00-03-01-2B-F4
Device buffer size  = 1498
NCP> exit
```

Once a host node has accepted the satellites request to boot a image called SYS$SYSTEM:NISCS_LAA.EXE is run to assist in the loading up of VMS on the satellite node. This image looks to find the sysgen parameter file called VAXVMSSYS.PAR in the root directory specified in SYS$SYSTEM:NETNODE_REMOTE.DAT if it does not find it the boot aborts. There is another case where an abort will occur and this is if the VAXCLUSTER sysgen parameter for the satellite node is found to be zero. This is because the satellite would not participate in a cluster with this set to zero.

The next part of the boot is to construct a parameter block containing the cluster authorization number and password from the cluster authorization file, CLUSTER_AUTHORIZE.DAT. Inside this parameter block is the SCSNODENAME and other unique parameters. NISCS_LAA appends this parameter block to the image SYS$SYSTEM:NISCS_LOAD. These are then down-line loaded to the satellite. When the load is complete control transfers from the console rom to the image NISCS_LOAD. This boot is now in a similar state as when control is given to VMB. The image NISCS_LOAD continues to load SYSBOOT and continue the same as a media booted system.

Any required changes should be bought to the attention of the cluster manager. The LAVC Cluster configuration procedure in the VMS manual "Guide to setting up a VAXcluster" should be followed for modifying or installing new LAVC satellite nodes.

Under VMS, there is a way of checking that WRITEBOOT has been run, and VMB is still in the same place!
First, determine the location of VMB on the system disk.
Do a $ DUMP/HEADER SYS$SYSTEM:VMB.EXE
Then look at the mapping information in this output and see which LBN the file starts at.
This is in decimal and needs to be converted to hex.
Now look at block 0 of the disk in question.

```
$ SET TERM/WIDTH=80
$ DUMP/BLOCK=(START:0, END:0) 'device:'
xxxxxxxx xxxxxxxx 2B720005 xxxxxxxx xxxxxxxxxxxxxxxx 00000000
xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxxxxxxxxxx 00000010
etc
```

This example shows a pointer to 00052B72 (hex), which is LBN 338802 (decimal), and should tie in with the LBN from DUMP/HEADER.

If you can't see any reason for it not to boot, try booting diagnostic supervisor. Then try to DS> SET LOAD [SYSEXE] and take a directory listing.
Ultimatly it is down to running micro- and load path diagnostics.

## 7.1.2 If it prints the VAX/VMS Version X.Y ...... "banner"

By the time VMS prints the banner announcing itself most of the system has been used. If a CI port is present then the CI microcode has been loaded into memory from the console load medium (unless the CI adapter is a CIBCA-B), most of the system has been read from disk and is now in memory, memory has been sized and the page tables have been built, the System Control Block has been built with vectors for errors and device interrupts, the current sysgen parameters have been loaded into memory, memory management has been turned on.

If the system hangs or crashes sometime between printing the banner and finally coming up, then it is sometimes possible to find out what it is doing by doing a CONVERSTATIONAL boot and altering some parameters.

IF YOU HAVE TO CHANGE ANY SYSGEN PARAMETERS YOU SHOULD FIRST OBTAIN THE SYSTEM MANAGERS PERMISSION. HAVING OBTAINED THAT PERMISSION YOU SHOULD CHECK THE VALUE OF ANY SYSGEN PARAMETER BEFORE CHANGING IT SO THAT YOU CAN RESTORE THE ORIGINAL VALUE LATER!

### 7.1.2.1 Booting VMS conversationally

Be very careful when working in a cluster environment and manually putting values in R5, as the most significant digit specifies the root you are booting from. Booting from the wrong root can have disasterous consequences.

#### 1.2.1.1 11750

On an 11750 if booting a local disk type B/1 ddan where ddan is your boot device, ddan does not have to be specified for the default boot device.

If booting in a cluster do the following . . .

```
>>> b/r5:800

BOOT58> @DEFBOO

When it gets to the point you get "LOAD VMB.EXE/START:@" do a CONTROL_C .
```

```
BOOT58> E R5 (To determine root you normally boot from)
BOOT58> D R5 x0000001 (Where x is the value determined from the previous step)
BOOT58> E SP
BOOT58> LOAD VMB.EXE/START:@
BOOT58> START @
```

#### 7.1.2.1.2 uVAXII,uVAX 35/3600

On an uVAXII and uVAX35/3600 type B/1 ddan where ddan is your boot device, ddan does not have to be specified for the default boot device. If these are cluster satellite nodes you do not have to worry about putting the root number in R5.

#### 7.1.2.1.3 8600/8650

On a 86xx type the following :-

```
>>> B/NOSTART
>>> E R5 (To determine the root you normally boot from)
>>> D R5 x0000001 (Where x is the root you normally boot from if a cluster)
>>> CONT
```

#### 7.1.2.1.4 Nautilus Family

On a 85/87/88xx type @BCIGEN or BOOT BCInnn/R5:x0000001 (Where x is the root you normally boot from and nnn is the unit number) if booting from the CI. Or if booting from a KDB50 type @BDAGEN or BOOT BDAmmn/R5:x0000001.

#### 7.1.2.1.5 82/83xx

On a 82/83xx if you are booting from a local disk type B/R5:1 ddan where ddan is your boot device, ddan does not have to be specified for the default boot device.

If booting in a cluster do the following . . .

```
>>> b/r5:800

BOOT58> @DEFBOO

When it gets to the point you get "LOAD VMB.EXE/START:@" do a CONTROL_C .

BOOT58> E R5 (To determine root you normally boot from)
BOOT58> D R5 x0000001 (Where x is the value determined from the previous step)
BOOT58> E SP
BOOT58> LOAD VMB.EXE/START:@
BOOT58> START @
```

#### 7.1.2.1.6 6xx0

On a 6xx0 type the following . . .

```
>>> SHOW BOOT

From the printout for DEFAULT determine the root number from the R5 value.

>>> B/R5:x0000001 (Where x is the value determined from the previous step)
```

#### 7.1.2.1.7 7xx0/10xx0

On a VAX 7xx0 or 10xx0 type the following . . .

```
>>> Show bootdef_dev                              This gives duan.n.n.n.n, etc.
>>> Show boot_osflags                             This gives blah,n,0

>>> boot -flag blah,n,1 duan,n,n,n,n, etc.        IE: Put 1 in R5 boot flags
```

#### 7.1.2.1.8 11780/11785, 11730

Type the following . . .

```
>>> @DEFBOO.CMD

When it gets to the point you get "LOAD VMB.EXE/START:@" do a CONTROL_C .

>>> E R5  (To determine root you normally boot from)
>>> D R5  x0000001  (Where x is the value determined from the previous step)
>>> E SP
>>> LOAD VMB.EXE/START:@
>>> START @
```

### 7.1.2.2 Starting VMS conversationally

VMS should now boot and stop with a SYSBOOT> prompt now take the following action :-

```
SYSBOOT>  SET STARTUP_P1 "min"        (this causes a minimum startup)
SYSBOOT>  SET STARTUP_P2 "true"       (this sets verify during startup)
SYSBOOT>  CONTINUE
```

VMS will now continue booting, when it comes to execute the system startup command procedure it will do this with verify set, printing each line of the command file before it is executed. Also various parts of the startup command procedure will be left out (for example AUTOCONFIGURE will not be done and SYS$MANAGER:SYSTARTUP.COM will not be executed (SYS$MANAGER:SYSTARTUP_V5 for VMS V5)). If the system hangs or crashes during the startup command file then you will be able to see what the last command it executed was, this is often sufficient for you or the customer to immediately diagnose the fault (e.g. I have seen a customer who claimed that there had been no changes suddenly remember a software installation that was recently done when he is presented with the detail of what is failing).

If the system now boots then log into a privileged account and type the following.

```
$ SET PROC/PRIV=CMKRNL
$ MC SYSGEN
SYSGEN> AUTOCONFIGURE ALL/LOG
SYSGEN> EXIT
```

If the system now hangs or crashes then control_P, halt, and do another conversational boot, set startup_p1 to "min" and when the system comes up go into sysgen and autoconfigure n/log where n is an adapter number. Isolate which adapter is causing the problem by trying each adapter one after another. Then autoconfigure n/log/exclude=device (eg MF or TX) to find out which device is causing your problem. On an 11780 the adapter numbers are the TR numbers of the adapters (3 for the first DW, 4 for the second DW, 8 for the first RH etc.) On an 11750 the adapter numbers are 8 for the first unibus, 4 for the first massbus). On an 11730 there is only one adapter so just use the /exclude switch to isolate the problem device.

If AUTOCONFIGURE has worked successfully then the problem probably lies in the system startup command procedure. Do another conversational boot and this time set startup_p2 to "true" but do not change startup_p1. If you are lucky then the site specific startup command file (sys$manager:systartup.com (sys$manager:systartup_v5 for VMS V5)) will now execute with verify set. If it does not then you will have to edit this command procedure to execute with verify set.

To edit SYSTARTUP.COM (or SYSTARTUP_V5.COM for VMS V5) you will have to boot the system without write protecting the system disk. DO NOT DO THIS WITHOUT THE PERMISSION OF THE SYSTEM MANAGER. IT IS PROBABLY BEST TO EXPLAIN WHAT YOU WANT DONE AND GET THE SYSTEM MANAGER TO DO IT FOR YOU. Write enable the system disk. Do another conversational boot. When you get the sysboot prompt type SHOW/ALL this will enable you to put right any parameters that you change when you have finished. SET STARTUP_P1 "min" but don't set startup_p2, CONTINUE. When the system is up log in to a privileged account and edit sys$manager:systartup.com (systartup_v5.com for VMS V5) to insert the line
$ SET VERIFY
at the beginning (if there is a $ set noverify or a f$verify(0) near the beginning of the command file then put the set verify command just after this).

Now shut the system down and reboot it with yet another conversational boot, set startup_p1 back to its original value (probably "") and set startup_p2 "true" so that you come up with verify set again. Now if the system hangs or crashes you will see the last line executed and this should point you in the right direction.

When you finish troubleshooting make sure that all sysboot parameters have been set back to their original values and that the startup command file is in it's original state (you shouldn't need to edit it again, just rename the original version to a higher version number).

**Note**

Sysboot parameter and sysgen parameter are interchangeable terms. These parameters can be changed either at a SYSBOOT> prompt during a conversational boot or at a SYSGEN> prompt during normal system operation.

## 7.2 How to use VMS when you have forgotten the passwords

**Note**

This procedure should ONLY be used on the ENGINEER'S system. UNDER NO CIRCUMSTANCES should you be doing this on a customers VMS system.

What do you do when you turn up on somebody else's site, boot the engineer's system and can't log in?

1. Boot the system with a conversational boot, if you don't know how to do this then look at the previous section.

```
SYSBOOT> SHOW /STARTUP                (remember the current value)
SYSBOOT> SET  /STARTUP _OPA0:
SYSBOOT> CONTINUE
```

2. When VMS comes up...

```
$ SET NOON

$ MCR INSTALL/COM SYS$SHARE:SECURESHR.EXE/PROT/HEAD/SHARE
$ MCR INSTALL/COM SYS$SHARE:SECURESHRP.EXE/PROT/HEAD/SHARE
$ MCR INSTALL/COM SYS$COMMON:[SYSLIB]LIBRTL.EXE/PROT/HEAD/SHARE

$ SET DEF SYS$SYSTEM
$ RUN AUTHORIZE
UAF>
```

3. Change the password on the system account and exit from authorize (e.g. UAF> MOD/PASS=datadoc_is_good SYSTEM).
4. When you reboot VMS do another conversational boot.

```
SYSBOOT> SET /STARTUP SYS$SYSTEM:STARTUP.COM  (or whatever the
previous value was)
SYSBOOT> CONTINUE
```

5. When VMS comes up you should now be able to log in...

If a System Manager asks the correct way to change his password then they should be told that This is fully described in the VMS System Management documentation set. For your personal knowledge the procedure is listed here

1. Boot the system with a conversational boot, if you don't know how to do this then look at the previous section.

```
SYSBOOT> SET STARTUP_P1 "MIN"         (Stop users logging in and speed up boot)
SYSBOOT> SET UAFALT 1                 (Dont use normal password file SYSUAF.DAT)
SYSBOOT> CONTINUE
```

2. Login to VMS

```
Username: SYSTEM
Password: anything       (password not checked!!!)
Password: anything       (secondary password not checked)
$ define/system/exec SYSUAF SYS$SYSTEM:SYSUAF.DAT
$ set def SYS$SYSTEM:
$ run authorize
UAF> modify/pass=systemdown system
UAF> exit
```

3. Shut down VMS.

```
SYSBOOT> SET STARTUP_P1 ""  (Reset to normal full boot)
SYSBOOT> SET UAFALT 0    (Use normal password file SYSUAF.DAT)
SYSBOOT> CONTINUE
```

4. Login as Normal now.

REMEMBER THAT THESE PROCEDURES WILL CAUSE SECURITY PROBLEMS IF YOU USE THEM.

## 7.3 Using VMS Sysgen to find floating addresses and vectors (for Unibus devices)

From an article by Vic Taylor ...

Adding a device to a Unibus on a VAX or PDP system is not just a matter of plugging it in. The same goes for removing a device. Unibus device addresses and vectors are subject to a "ranking" scheme. This means that the address and vector of one device depends on those of other devices. (See Chapter 9, PDP for more details). The best way to work out these addresses and vectors is like this:

1. Log into a VMS system.
2. Invoke sysgen by typing **MC SYSGEN**.
3. Enter the configure mode in sysgen by typing **CONFIG** at the **SYSGEN** prompt.
4. You now get the **DEVICE** prompt, you now enter your configuration as the example below :-

```
SYSGEN> CONFIG
DEVICE> DMF32 2           (two DMF32's)
DEVICE> DZ11 3            (three DZ11's)
DEVICE> UNA               (one Deuna)
DEVICE> UDA 2             (two UDA50's - or RQDX controllers!)
DEVICE> TU81              (one TU81 - or one TK50!)
DEVICE> ^Z                Type Control Z to finish and sysgen
                          will print out the correct addresses,
                          vectors and device names.
```

## 7.3 Identifying Hangs/Crashes/Halts/Reboots

It is all too common to look through the History of a system in a Site Management Guide and find numerous entries that read "System Hung", "System Crashed", "System Halted" with no further information. It is important to distinguish different symptoms of these types as we often have difficulty telling whether we are trying to fix a new fault or the same one as last week.

Try asking the following questions (either of yourself or the customer)

1. Was anything printed on the console
2. Was there any entry in the errorlog
3. Were there any operator messages
4. Are the symptoms identical to previous ones on this system

### 7.3.1 Hangs

If there was nothing printed on the console and the problem was a total lack of response from all terminals (make sure that you try the console and more than one unibus terminal controller, if the console works but no other terminal does then try examining adapter and DZ/DMF registers) then the trouble statement should be system hung. It should still be possible to give more information with this sort of problem.

1. Did all users notice the problem at the same time or was the hang "gradual" with some users still able to work after others had hung.
2. Was there any evidence of system activity at all (disk head movement, printers working, tape motion etc.)
3. Examine and note the state of all led's on the CPU (e.g. the run light, attention light, microPC, clock state and unibus microPC on an 11780/11785)
4. Does the console respond to a control_P
5. If the CPU is an 11750 does the console respond to a control_D
6. Halt the CPU, examine and note down the PC and PSL
7. Disable clock interrupts by typing
   >>> D/I 18 0
8. Continue the CPU
9. Halt the CPU, note the PC and PSL again
10. Continue and halt the CPU another 5 or 6 times, noting the PC and PSL each time.
11. Single step the CPU through about 15 to 20 instructions to get an idea of the size of any loop it might be running in.
12. Examine all adapter registers.
13. If the PSL shows that the system is running at a consistently high IPL then look at page 49 of the Vax systems black book to determine what software/hardware might be running/interrupting

14. Force a crash in order to generate a crash dump. On an 86xx,85/87/88xx, 1178x or 11730 type @crash to a >>> prompt. On an 11750, 82/83xx, 6xx0, 7xx0/10xx0, Microvax 2 and Microvax 35/3600 type the following.

```
^P
>>> E/G F           PC
>>> E P             PSL                   ! E PSL on Microvaxes,62/63xx
>>> E/I 0           | \
>>> E/I 1           |  \
>>> E/I 2           |   -  Stack pointers
>>> E/I 3           |  /
>>> E/I 4           | /
>>> D/G F FFFFFFFF  deposit -1 in PC
>>> D P 1F0000      deposit IPL 31 in PSL ! D PSL on Microvaxes,62/63xx
>>> C               continue
```

15. Make a copy of the crash dump to mag tape so that if the fault is not fixed the crash dump can be compared with other hangs and common factors can be found

16. Write a full description of the hang including all of the above information in the site management guide.

### 7.3.2 Fatal Bugchecks

When VMS crashes it reports a fatal bugcheck. ALL fatal bugchecks include a text string which identifies the bugcheck, for example INVEXCEPTN - fatal exception on interrupt stack or above ASTDEL. It is vital when recording a system crash to include this information so that different crashes can be identified. If the fatal bugcheck is a MACHINECHK - Machine check while in kernel mode, then you should be able to decode the data on the stack to give more information. If you are unsure of how to do this then look in the appropriate manuals for the particular CPU or the section of DATADOC for that CPU, or phone support and ask for help. Machine checks are nearly always caused by HARDWARE PROBLEMS. It is possible for software to generate a machine check (for example by doing a longword transfer to unibus address space, or addressing non-existent memory) but this very rarely happens.

Other fatal bugchecks may be caused by software or hardware, the most common is INVEXCEPTN, which can be caused by software but is often caused by data corruption. It is important with any of these bugchecks to make a copy of the crash dump to tape so that it can be compared with other crashdumps. Often the only way to decide if one of these is caused by hardware or software is to compare 3 or 4 crash dumps looking for similarities and differences.

For any bugcheck it is worth getting the console printout from the time of the crash and including it with your report in the site management guide. This way it is possible to tell whether we are trying to fix a new fault or still working on an old one.

When the system crashes due to a fatal bugcheck it does not check the state of the console switch to decide whether to reboot. If the sysgen parameter BUGREBOOT has a value of 1 then the system reboots (this is the default). If BUGREBOOT is 0 then the system does not reboot.

#### 7.3.2.1 How to ensure that a crash dump is taken

For VMS to take a crash dump there are a number of requirements that must be fulfilled.

1. There must be a dump file with certain attributes.

   The file is normally SYS$SYSTEM:SYSDUMP.DMP, but may be SYS$SYSTEM:PAGEFILE.SYS.

   This file must have existed **before the system was booted.**
   So if you want to use a new dump file you have created or one you have changed the size of; then you must reboot the system so the initialization code can record the position and size of the new dump file ready for the bugcheck code to dump directly to it.
   Also bear in mind that because the bugcheck code will use the part of the disk that it was told about at boot time, you must be careful when deleting a dump file (else the area might be used by someone and then dumped all over by the bugcheck code when a bugcheck occurs). The best approach is to "rename" the existing dumpfile to something inocuous, reboot the system and **finally delete it.**

   On VMS V4, SYSDUMP.DMP must be big enough to hold a complete copy of memory (plus a bit extra).
   Autogen will normally produce the correct size dump file, it will have one block for every 512 Byte page of memory (2048 blocks per MByte) plus an additional 4 blocks for header information and errorlog buffers.
   For example a 4MByte system should have an 8196 block dumpfile.
   If you need to modify the size of the dumpfile and you do not wish to use AUTOGEN then use @SYS$UPDATE:SWAPFILES.

   On VMS V5, you have a choice of taking a complete dump (as per V4), or a partial dump. The partial dump has the advantage that your dumpfile need not be as large as physical memory. But it does represent a trade-off because the usefulness of the dump will be diminished

   If you want a complete dump, DUMPSTYLE must be set = 0, and the formula for SYSDUMP.DMP is the same as VMS V4 except you need an additional 5 blocks, but again, AUTOGEN will do the calculation for you
   If you want a partial dump, set DUMPSTYLE=1 in MODPARAMS.DAT, and then run AUTOGEN to calculate the size of dump file required.
   When a bugcheck occurs, it will dump things in order of decreasing importance.
   If the "dumped data" is not thought to be usefull enough (eg only a small percentage of memory resident processes are getting dumped) you should increase the size of SYSDUMP.DMP. Do this by putting a line in MODPARAMS.DAT to increase DUMPFILE beyond AUTOGEN's calculated value and then run AUTOGEN (@SYS$UPDATE:AUTOGEN SAVPARAMS REBOOT).

   On some systems, where there is insufficient room on the system disk for a dumpfile, the crash dump is taken to the pagefile (SYS$SYSTEM:PAGEFILE.SYS instead of to the dumpfile.
   This occurs automatically when there is no SYS$SYSTEM:SYSDUMP.DMP present at **boot time**
   Several other things must be taken into consideration :-

   The parameter SAVEDUMP must be set to a 1 in MODPARAMS.DAT, else the dump information in the page file will get overwritten by paging activity as soon as VMS restarts.
   DIGITAL does not recommend setting SAVEDUMP **unless you really know what you are doing!**

If a full dump (DUMPSTYLE=0) is being taken to the page file, and SAVEDUMP=1; you need a pagefile large enough to allow paging activity without overwriting the dump data. (Ruth G. says allow for all of physical memory plus 1000 blocks).
If a partial dump (DUMPSTYLE=1) is being taken to the page file, and SAVEDUMP=1, you have a potential disaster.
The partial dump will keep dumping until it completes the contents of memory or the target file is full.
You could end up with a page file that is full of dump data, and un-available for paging activity.

On Clusters, a dump file can be shared by different nodes to reduce wasted disc space.

```
On any node (n is the size of the dump file)
SYSGEN> CREATE SYS$COMMON:[SYSEXE]SYSDUMP-COMMON.DMP/SIZE=n

On every node (n is the system specific root)
  $ SET FILE SYS$COMMON:[SYSEXE]SYSDUMP-COMMON.DMP -
   _$ /ENTER=SYS$SYSDEVICE:[SYSn.SYSEXE]SYSDUMP.DMP
```

**Note**

If you have MA780 multi-port memory you will have to extend the dump file that autogen produces to allow for this. The best way to do this is to edit SYS$SYSTEM:MODPARAMS.DAT to include the line "DUMPFILE = nnn" where nnn is the size of the dumpfile required in blocks.

2. The sysgen parameter DUMPBUG must have a value of 1 (this is the default value). If DUMPBUG is set to 0 then a crash dump will never be taken.

3. The system must suffer a FATAL BUGCHECK, you do not get a crash dump if you type ^P HALT, or if the system hangs, or if you have a non-fatal bugcheck, or if you have a double error halt.
A normal shutdown does in fact cause a fatal bugcheck and writes a crash dump on VMS V4.n systems but V5.n DOES NOT write a dump on a normal shutdown though it does mark the dumpfile not valid, so it is quite possible to get an error message from SDA indicating the dumpfile is empty and not valid if a machine has never had a fatal bugcheck or it has been shutdown since a crash.
   If you want to obtain a crash dump from a hung system then see the previous section.
   If you want to obtain a crash dump from a double error halt, interrupt stack invalid or another pathological system halt then see the next section.
   If you want to obtain a crash dump from a non-fatal bugcheck then you must set the sysgen parameter BUGCHECKFATAL to a value of 1 - this makes all bugchecks fatal and will cause the system to crash for what should have been a non-fatal bugcheck, it is often the only way to get any useful information from a non-fatal bugcheck but BUGCHECKFATAL should be set back to its default value of 0 when the fault is fixed.

4. The system disk, which contains the bugcheck code, must be accessible. When VMS suffers a fatal bugcheck it assumes that the disk driver software may be corrupt and uses the bootstrap driver (that was loaded into memory when the system booted) to read the bugcheck code from the system disk.
If it is unable to do so then VMS halts! This is the only halt instruction that I have ever come across in VMS.

5. The system must be allowed sufficient time to write the dumpfile. If the system has been shut down - or crashed by running SYS$SYSTEM:OPCRASH - then it will print "SYSTEM SHUTDOWN COMPLETE USE CONSOLE TO HALT SYSTEM" on the console terminal when the dumpfile has been written. If you have a fatal bugcheck then you will see a reboot attempt on the console when the dump has been written.

6. The crash dump will be overwritten next time the system is shut down , if you want to keep a copy of the crash dump for future use then you should make a copy of it as soon as the system reboots. The most reliable way to do this is to use backup......

```
$! (first check that you have a valid dump)
$ anal/crash sys$system:sysdump.dmp
SDA> ^Z
$! (check for error messages from SDA)
$!
$ allocate mta0:
$ init/dens=1600 mta0: dumps
$ mount/foreign mta0: dumps !NOT NEEDED FOR V5 AND ABOVE.
$ backup/ignore=nobackup/verify
from?  sys$system:sysdump.dmp
to?    mta0:dumps/save/dens=1600/rewind
$ dismount mta0:
$ deallocate mta0:
```

7. If your crash dump has been written to the pagefile then you must make a copy of it using SDA to free up the pagefile space for use by VMS. You need CMKRNL privilege to do this.

```
$ anal/crash sys$system:pagefile.sys
$ copy ddcu:filename.ext          ! copy the dump to a new file
$ exit
```

Commonly these commands would be in the startup command file.

## 7.3.3 Halts and other pathological failures

Included under this heading are double error halts, interrupt stack invalid, illegal interrupt/exception vector and a number of other errors.

If auto-restart is not enabled then the system will not reboot after one of these errors. If auto-restart is enabled then the system will restart and it is possible that the only information to be gleaned will be the console printout which will show which of the above errors occurred.

If the system restarts after one of these errors then a code is passed to VMS in the AP (argument pointer). If this code shows a power fail restart then VMS will perform a warm restart, if it shows anything else then VMS will immediately bugcheck, writing a crashdump. The cause of the bugcheck will be given as (for example) FATAL BUGCHECK, DBLERR - Double error halt restart.

If you have disabled auto restart and the system has halted with one of these pathological errors then it is still possible to obtain a crash dump on some VAXes. Deposit the correct code from the following table into the Argument Pointer (AP) and then @RESTAR.CMD for 1178x's or @RESTAR.COM for Nautilus family, other Vaxes do not use the RESTAR command file mechanism to do a restart.

Table 7–1: Pathological halt codes

| BUGCHECK NAME | AP VALUE | |
|---|---|---|
| UNKRSTRT | 0, 1, 2, | anything greater than 11 |
| POWER FAIL | 3 | (this is a normal warm restart) |
| IVLISTK | 4 | |
| DBLERR | 5 | |
| HALT | 6 | |
| ILLVEC | 7 | Not used on Microvax 1 |
| NOUSRWCS | 8 | Not used on Microvax |
| ERRHALT | 9 | 780, 782, 785 and 8600 only |
| CHMONIS | A | |
| CHMVEC | B | Not used on 780, 782, 785 |
| SCBRDERR | C | Microvax and 11730 only |
| WCSCORR | D | 8600 only |
| CPUCEASED | E | 8600 only |
| OUTOFSYNC | F | 11785 only |
| ACCVIOMCHK | 10 | Microvax 2 only |
| ACCVIOKSTK | 11 | Microvax 2 only |

### 7.3.3.1 Unknown restart

If you execute restar.cmd without a valid value in the argument pointer then VMS will immediately bugcheck with FATAL BUGCHECK, UNKRSTRT - Unknown restart code.

### 7.3.3.2 Interrupt stack invalid

This error occurs when an attempt is made to push onto (or pop from) the interrupt stack and the stack pointer is pointing to a non valid address, this may be caused by too much pushing onto the stack (stack overflow), attempting to pop from an empty stack (stack underflow) or a corrupted stack pointer.

It is vital to obtain a crash dump from an interrupt stack invalid so that the stack can be examined to see which of these has occurred.

It is possible to (temporarily) cure stack overflow by increasing the Sysgen parameter INTSTKPAGES but this should only be done if there seems to be some good reason for the interrupt stack being so large.

#### 7.3.3.3 Double Error Halt

A dcuble error halt occurs when the microcode detects a machine check whilst it is attempting to process a previous machine check. You should examine internal registers to get information about the cause of both the first and second machine check.

## 7.4 VMS error messages

Since VMS 5.4, you can get on-line help about VMS error messages :-

```
        $ HELP/LIBRARY=SYS$HELP:SYSMSGHELP.HLB ERRORS xxxxx
```

where xxxxx is something like PORT_ERROR_BITS or BUGCHECK.

It is slow as the library is compressed, else it would be about 1600 blocks.
To decompress it, use :-

```
        $ LIBRARY/DAT=EXPAND/OUTPUT=some_new_name SYS$HELP:SYSMSGHELP.HLB
```

Of course, you then need to use "some_new_name" in your original HELP command.

## 7.5 VAX/VMS DEVICE MNEMONICS AND DRIVERS

This article is to help cope with the numerous device names that get listed when you do a "SHOW DEVICE" under VMS. The mnemonic column lists the name you may see in a "SHOW DEVICE" printout. (i.e AW means anything like AWA0,AWB1 etc. .)

The product name is the hardware or software option that uses the mnemonic listed. Quite a few of the mnemonics are pseudo devices for software purposes. Also a hardware option such as a DUP11 may have a different mnemonic depending on which software is using it, by this I mean if PSI is running over it it will have a name of XWA0 but if 2780 emulator is running it would be XJA0.

Where possible I have given the driver used by VMS to control the device or pseudo device.

| MNEMONIC | PRODUCT NAME | DRIVER |
|---|---|---|
| AW | ADQ32 A/D CONVERTER | AWDRIVER |
| AX | AXV-11C DEVICE | AXDRIVER |
| AY | AAV-11D DEVICE | AYDRIVER |
| AZ | ADV-11D DEVICE | AZDRIVER |
| BQ | IBQ01 QBUS TO BITBUS | BQDRIVER |
| CN | VAX CI DECNET CLASS DRIVER | CNDRIVER |
| CQ | DESNA DECNET SNA GATEWAY | |
| CR | CR11/CM11 CARD READER | CRDRIVER |
| CS | VMS CONSOLE STORAGE GENERIC DEV | CSDRIVER |
| CT | CTERM REMOTE TERMINAL DRIVER | CTDRIVER |
| DA | RC25 | DUDRIVER |
| DB | RP05/06 | DBDRIVER |
| DD | TU58 | DDDRIVER |
| DK | SCSI RZ Devices | DKDRIVER |
| DI | DSSI DISK | |
| DJ | RA60 | DUDRIVER |
| DL | RL01/02 | DLDRIVER |
| DM | RK06/RK07 | DMDRIVER |
| DN | DRV-11J | |
| DQ | 11/730 IDC RB730 R80/RL02 | DQDRIVER |
| DR | RM02/RM03/RM05/RM80/RP07 | DRDRIVER |
| DS | VMS VOL SHAD VIRTUAL UNIT | DSDRIVER |
| DU | RA80/RA81/RA82/RA70/RA90 DSA DISCS | DUDRIVER DVDRIVER=UVAX2000 |
| DY | RX02 | DYDRIVER |
| ES | DESVA LANCE INTEGRAL ETHERNET | ESDRIVER |
| ET | DEBNA/DEBNI BI ETHERNET CONTROLLERS | ETDRIVER |
| EX | DEMNA, XMI ETHERNET CONTROLLER | |
| EZ | SGEC INTEGRAL ETHERNET CONTROLLER | |
| FB | DUMMY DEVICE PART OF THE TERMINAL FALLBACK DRIVER | FBDRIVER |
| FL | DATAFILE PSEUDO DEVICE (VAXLAB) | |
| FX | DEMFA, XMI FDDI CONTROLLER | |
| FY | HSC50/70 SET HOST | FYDRIVER |
| GA | QDSS GRAPHICS CONTROLLER (DECWINDOWS) | |
| GB | LEGGS GVAX (DECWINDOWS) | |
| GC | QVSS GRAPHICS CONTROLLER (DECWINDOWS) | |

| MNEMONIC | PRODUCT NAME | DRIVER |
|---|---|---|
| HX | MICROVMS/DRQ3B DEVICE DRIVER | HXDRIVER |
| HY | CHESAPEAKE A-D CONVERTER (VAXLAB) | |
| HZ | ARRAY PROCESSOR (VAXLAB) | |
| IK | KEYBOARD DECODER (DECWINDOWS) | |
| IM | MOUSE DECODER (DECWINDOWS) | |
| IN | VAX DECWINDOWS | |
| IT | TABLET DECODER (DECWINDOWS) | |
| IX | IEQ11/IEU11 | |
| J* | RESERVED FOR CUSTOMERS | |
| KJ | KXJ11-CA VAX/RT11 PRODUCT | |
| KS | DECNET/SNA GATEWAY | |
| KU | KXT11/KXJ11 PERIPHERAL PROCESSOR TOOL KIT/MICROVMS | |
| KW | KW11K PROGRAMMABLE CLOCK | |
| KZ | KWV-11C DEVICE (VAXLAB) | |
| LA | LPA11K LABORATORY PERIPHERAL | |
| LC | DMF-32 PRINTER PORT | LCDRIVER |
| LD | LNV21 PRINTER/SCANNER CONTROLLER | |
| LE | VIRTUAL PRINTER DEVICE (VAX SECURE) | |
| LI | LINE-PRINTER PORT FOR DMB-32 | LIDRIVER |
| LP | LINE PRINTER | LPDRIVER |
| LS | LPS11 LAB. PERIPHERAL SYS | |
| LT | LAT PSEUDO TERMINAL | |
| LV | KMS11-KV (VAX VIRTUAL ACCESS) | |
| MB | PSEUDO MAIL BOX | MBXDRIVER |
| ME | VIRTUAL TAPE DEVICE (VAX SECURE) | |
| MF | TU78/TM78 | TFDRIVER |
| MI | TF70 DSSI TAPE | |
| MK | SCSI TAPE DEVICES | |
| MN | MUX200 FOR DMF-32 | |
| MS | TS11 | TSDRIVER |
| MT | TE16/TU45/77 | TMDRIVER |
| MU | TK50/TU81/TA78 | TUDRIVER<br>TVDRIVER=UVAX2000 |
| MX | MUX200 FOR DUP-11 | |
| NA | A "record oriented" null device to provide spooling support for VAX DQS | |
| ND | DECNET-VAX MAINTENANCE DRIVER | |
| NE | VMS DECNET-VAX INTERFACE | |
| NE | VMS DECNET-VAX INTERFACE | NETDRIVER |
| NF | FILE ACCESS INTERFACE IN DECNET/SNA GATEWAY | |
| NI | DECDATAWAY DY32 NETWORK INTERFACE PSEUDO DEVICE | |
| NL | NULL PSEUDO DEVICE | |
| NO | VAX ASYNC DDCMP TERMINAL CLASS DRIVER | NODRIVER |
| NP | 3271 DRIVER FOR DMF32 | |
| NQ | DECNET/SNA DHCF 3270 | |
| NR | UNIVAC EMULATOR (NTR) DEVICE FOR DUP-11 | |

| MNEMONIC | PRODUCT NAME | DRIVER |
|---|---|---|
| NV | VAX PSI PSEUDO X.29 TERMINAL DRIVER | |
| NW | VAX PSI PSEUDO NET DRIVER | |
| NY | 2780 DRIVER FOR DMB32 AND DMF32 | |
| NZ | X25 INTERFACE PSEUDO DEVICE | |
| OP | OPERATOR TERMINAL | |
| OS | NETWORK DRIVER FOR OSI TRANSPORT | |
| PA | CI780/CI750 PORT DRIVER | PADRIVER |
| PB | BI PORT DRIVER | PBDRIVER |
| PC | PSEUDO DEVICE FOR VAX/DTM | |
| PD | VMS PSEUDO-DISK DEVICE DRIVER | |
| PE | LAVC NI PORT DRIVER | PEDRIVER |
| PF | DRQ3-B PRESTON GMAD I/O SUBSYSTEM (VAXLAB) | |
| PG | DRV1–WA PRESTON GMAD I/O SUBSYSTEM (VAXLAB) | |
| PI | DSSI PORT ADAPTER | PIDRIVER |
| PK | SCSI PORT DEVICES | |
| PJ | VIRTUAL PORT DEVICE | |
| PM | PRESTON A/D DRB32W INTERFACE TO A DRB32W | |
| PO | GRAPHICS PSEUDO-DEVICE (VAXLAB) | |
| PP | PP611 PAPER TAPE PUNCH | |
| PQ | QBUS PORT DRIVER | |
| PR | PR11/PC11/PR68E/PA611 PAPER TAPE READER | |
| PT | PORT FOR TK50/TU81 (TMSCP PORT) | PUDRIVER |
| PU | PORT DRIVER FOR UDA50/RQX/KDA | PUDRIVER |
| PY | PSEUDO-TERMINAL USED WITH DECWINDOWS TERMINAL EMULATOR | |
| Q* | RESERVED FOR CUSTOMERS | |
| RA | LOCAL VMSSCS CLASS DRIVER FOR REMOTE SDA | |
| RQ | DECNET/SNA DHCF 3270 CLASS DRIVER | |
| RS | RESERVED FOR SYSGEN | |
| RS | RESERVED FOR SYSGEN | |
| RT | REMOTE TERMINAL PSEUDO DEVICE (PRE VMS V4.0) | |
| SB | DECDATAWAY DY32 SERIAL BUS CONTROLLER | |
| SC | SCREEN CONTROL PSEUDO DEVICE (SCS-11) & SHADOW PROCESSING CLASS DRIVER | |
| SI | SYNCH LINE PORT FOR DMB-32 | SIDRIVER |
| SJ | DSV11 SYNCH LINE INTERFACE | |
| SL | DSB32 SYNCHRONOUS DEVICE DRIVER | SLDRIVER |
| SN | VMS/SNA PSEUDO DEVICE | |
| SR | VAX/SECURE FUNCTION REQUEST DEVICE | |
| SW | SPM SOFTWARE TIMER DRIVER | |
| TD | SESSION SPECIFIC TERMINAL DEVICE | |

| MNEMONIC | PRODUCT NAME | DRIVER |
|---|---|---|
| TE | VAX/SECURE VIRTUAL TERMINAL DEVICE | |
| TO | VAX/DECMAP PSEUDO-UNIT DEVICE CONTROLLED BY TODRIVER | |
| TP | TERMINAL CLASS DRIVER FOR PSI | |
| TQ | VAX-11 BTS 3270 EMULATOR TERM CLASS DRIVER | |
| TT | VMS TERMINAL CLASS DRIVER FOR NON DMA DEVICES | |
| TW | PSEUDO-TERMINAL USED WITH DECWINDOWS TERMINAL EMULATOR. | |
| TX | VMS TERMINAL CLASS DRIVER FOR DMA DEVICES | YCDRIVER=DMF32<br>YFDRIVER=DHU/DHV11<br>YIDRIVER=DMB32 |
| UA | MICROVMS KMV11-A X.25 DRIVER | |
| UE | VAX PSI ON DPV11 | |
| UF | VAX-11 2780 ON DPV-11 | |
| UG | VAX-11 3271 ON DPV-11 | |
| UH | VMS/SNA DPV-11 DEVICE | |
| UK | KCT32 TITAN COMM GEAR | |
| UL | KCT-32 TITAN COMM GEAR | |
| UM | KCT-32 TITAN COMM GEAR | |
| UN | KCT-32 TITAN COMM GEAR | |
| UO | KCT-32 TITAN COMM GEAR | |
| UQ | VAXBI OPTION DRB32 | |
| UR | DR70-B GENERAL MASSBUS I/F | |
| UT | MIRA DUAL PROCESSOR WATCHDOG | |
| UU | VAX DRE11-C FOR US AND DRX11-C/VMS FOR EUROPE | |
| UV | VAX PSI ON KMV11 | |
| UW | KMV1A DRIVER | |
| UX | MICROVMS KMV11-B X.25 DRIVER | |
| VA | VCB02 QVSS COLOR GPX | VADRIVER=UVAX2<br>VFDRIVER=UVAX 2000 |
| VB | VAX-11 WORKSTATION | |
| VC | VCB01 QVSS B/W | VCDRIVER=UVAX2<br>VEDRIVER=UVAX 2000 |
| VD | VIRTUAL PSEUDO DISK AS IMPLEMENTED UNDER DECNET AND RTEM-11 | |
| VK | VAX 2000 VIDEO DEVICE | |
| VL | VS8000 (LYNX) DEVICE/BI ADAPTER | |
| VP | DR-11WA, DR11-B INTERFACE FOR DMA | |
| VS | VAX-11 VS11/VSV11 DRIVER | |
| VV | MICROVAX VSV21 SUPPORT SOFTWARE | |
| VV | HPWS INTERFACE BETWEEN VMS TERMINAL CLASS DRIVER AND A TERMINAL EMULATOR (XCLIENT) PROCESS | |

**Note**

VV WAS ASSIGNED TO VSV21 BEFORE ACTUAL RECORDS OF DEVICE DRIVERS WERE RECORDED BY SQM. INADVERTENTLY, VV WAS ALSO REGISTERED TO HPWS IN MAY, 1987. IT HAS BEEN DETERMINED BY BOTH PRODUCTS AND SQG/SQM THAT THESE PRODUCTS WILL NEVER BE IN A SITUATION WHERE THEY WILL HAVE A

| MNEMONIC | PRODUCT NAME | DRIVER |
|---|---|---|
| | CONFLICT, DUE TO THE FACT THAT ONE IS A QBUS DEVICE AND THE OTHER IS A BI. | |
| WC | 11/750 G&H MICROCODE PWR. FAIL. RECOV. | WCDRIVER |
| WD | 11/750 PCS POWER FAILURE RECOVERY | WDDRIVER |
| WK | WORK & SCRATCH FILE PSEUDO DEVICE AND DECNET PHASE 3 KMC-11 | |
| WP | PSEUDO WATCH POINT DEVICE FOR VMS | |
| WT | VTXXX SERIES EMULATOR | |
| XA | DR/DRV-11W DEVICE | XADRIVER |
| XB | DR-11B PARALLEL INTERFACE | |
| XC | DEVICE DRIVER FOR DRV11-WA, SOLD LOCALLY IN JAPAN. | |
| XD | DMP-11/DMV-11 | XDDRIVER |
| XE | DEUNA/DELUA, UNIBUS ETHERNET | XEDRIVER |
| XF | DR-32/DR-780/DR-750 | |
| XG | DMF32 SYNC PORT DEVICE DRIVER | XGDRIVER |
| XI | DMF32 AND DR-11C PARALLEL PORT | XIDRIVER |
| XJ | VAX-11 2780 PE ON DUP-11 | XJDRIVER |
| XK | VAX-11 3271 PE ON DUP-11 | XKDRIVER |
| XL | LANCE CHIP ON THE SCORPIO CPU | |
| XM | DMR11/DMC11 | XMDRIVER |
| XN | VAX PSI ON KMS11-PX/PY | XNDRIVER |
| XP | VAX PCL PARALLEL BUS | |
| XQ | DEQNA/DELQA/DEQTA, QBUS ETHERNET | XQDRIVER |
| XS | VAX PSI ON KMS11-BD/BE | XSDRIVER |
| XW | VAX PSI ON DUP11 | XWDRIVER |
| XX | KMS11-BD | |
| YK | XBMC03 PROTOCOL EMULATOR ON DUP-11 | |
| YN | DEVICE FOR KMS11 FRAMING SOFTWARE | |
| YO | KMV-11 X25 | |
| YP | KMS-11 ADCCP | |
| YQ | KMS-11 3271 DEVICE | |
| YT | KMS-11 SINGLE LINE PSI | |
| YU | KMS-11 SINGLE LINE X25 | |
| YV | KDI DRIVER FOR KMS11-KV DATAKIT INTERFACE | |
| YW | KMS-11 | |
| YX | KMS-11 BL/BM FOR DDCMP SINGLE LINE | |
| ZB | DTBNA DIGITAL TOKEN TO BI-BUS NETWORK ADAPTER | |
| ZP | KA800 VRTA PORT DRIVER | |
| ZQ | DTQNA DIGITAL TOKEN TO NETWORK ADAPTER | |
| ZS | DST32 SYNC COMM FOR THE UVAX2000 | ZSDRIVER |
| ZU | INTERFACES VOTS TO A KMS11 | |

## 7.6 List Of VMB.EXE Register Bit Definitions

Table 7–2: R0 - Boot device type codes

| R0 | <07:00> | BOOT DEVICE |
|---|---|---|
| | 0 | MASSBUS device (RM02/3,RP04/5/6/7,RM80) |
| | 1 | RK06/7 |
| | 2 | RL01/2 |
| | 3 | IDC(almost an RA80) on 11/730 |
| | 11 | UDA-50 |
| | 20 | HSC on CI, KFMSA, |
| | 21 | KDB50 |
| | 40 | Console block storage device |
| | 43 | KDM70 |

Table 7–3: R1 - Boot device's bus address

| MACHINE | BITS | DEFINITION |
|---|---|---|
| **11730 11780 and 11785, Boy's-Own-System, Star and Superstar** | | |
| | <31:04> | Must Be Zero (MBZ) |
| | <03:00> | Transfer Request (TR) number of adapter |
| **11750, Comet** | | |
| | <31:24> | MBZ |
| | <23:00> | address of the I/O page for the boot device's adapter |
| **6000 series, Calypso** | | |
| | <03:00> | BI node number of load path |
| | <07:04> | XMI node number of load path |
| **8600 and 8650, Venus and Morning-Star** | | |
| | <31:06> | Must be zero |
| | <05:04> | A-bus Adapter number |
| | <03:00> | TR number of the adapter |
| **8200, Scorpio** | | |
| | <31:04> | MBZ |
| | <03:00> | BI node number of adapter |

Table 7–3 (Cont.): R1 - Boot device's bus address

| MACHINE | BITS | DEFINITION |
|---|---|---|
| **8800, Nautilus** | | |
| | <31:06> | Must be Zero |
| | <05:04> | indicates which BI |
| | <03:00> | BI node number of adapter |
| **8810, Polarstar** | | |
| | <31:07> | Must be Zero |
| | <06:04> | indicates which BI |
| | <03:00> | BI node number of adapter |

Table 7–4: R2

| MACHINE | BITS | DEFINITION |
|---|---|---|
| **All Controllers** | | |
| | <31:24> | Controller letter designator (optional) (Used if booting from second controller i.e DUB) (Subtract %x40 from ascii controller letter, so C would be 3) |
| **DSSI** | | |
| | <17:16> | 2=KFMSA port 0, 3=KFMSA port 1 |
| | <03:00> | Boot device DSSI NODE number |
| **UNIBUS** | | |
| | <31:18> | Must be Zero |
| | <17:00> | UNIBUS address of the device's Control and Status Register (CSR),3F468 for first UDA50 |
| **MASSBUS** | | |
| | <31:04> | Must be Zero |
| | <03:00> | adapter's controller/formatter number |
| **CI** | | |
| | <31:08> | Must be Zero |
| | <07:00> | HSC port number (station address) For automatic failover when more than one HSC is present <15:07> is the first node to try if non zero and <07:00> is the second node to try. With nodes 0 and 1 1 must be the first node. |

**Table 7–5: R3 - Boot device unit number**

| BITS | DEFINITION |
|---|---|
| <31:> | Must be a 1, if Shadow Set |
| <30:16> | Virtual unit number, if Shadow Set |
| <15:00> | Physical unit number |

**Table 7–6: R4**

Logical Block Number (LBN) to boot from if bit 3 is set in R5 (NOTE: not supported on VAX 11/750)

Table 7–7: R5 - Software Boot control flags

| Bit | Definition |
|---|---|
| 0 | Conversational boot. At various points in the system boot procedure, the bootstrap code solicits parameters and other input from the console terminal. If the DIAG is also on, then the diagnostic supervisor should enter "MENU" mode and prompt user for devices to test. |
| 1 | Debug. If this flag is set, VMS maps the code for the XDELTA debugger into the system page tables of the running system. |
| 2 | Initial breakpoint. If RPB$V_DEBUG is set, VMS executes a Breakpoint fault (BPT) instruction immediately after enabling mapping. |
| 3 | Secondary boot from boot block. Secondary bootstrap is a single 512-byte block, whose LBN is specified in R4. |
| 4 | Diagnostic boot. Secondary bootstrap is image called [SYSMAINT]DIAGBOOT.EXE. |
| 5 | Bootstrap breakpoint. Stops the primary and secondary bootstraps with a breakpoint instruction before testing memory. |
| 6 | Image header. Takes the transfer address of the secondary bootstrap image from that file's image header. If RPB$V_HEADER is not set, transfers control to the first byte of the secondary boot file. |
| 7 | Memory test inhibit. Sets a bit in the Page Frame Number (PFN) bit map for each page of memory present. Does not test the memory. |
| 8 | File name. Virtual Memory Boot (VMB) prompts for the name of a secondary bootstrap file. |
| 9 | Halt before transfer. Executes a HALT instruction before transferring control to the secondary bootstrap. |
| 10 | Intended to tell VMB not to read a file from the boot device that identifies bad or reserved memory pages, so that VMB does not mark these pages as valid in the PFN bitmap. |
| 11 | Specifies that multi-port memory is to be used for the total executive memory requirement. No local memory is to be used. This is for tightly-coupled multi-processing. |
| 12 | Specifies that multi-port memory should be used in addition to local memory, as though both were one single pool of pages. |
| 13 | Specifies that a more extensive algorithm be used when testing main memory for hardware incorrectable Read Data Substitute (RDS) errors. |
| 14 | Requests use of MA780 memory if MS780 is insufficient for booting. Used for VAX 11/782 installations. |
| 15 | Used by Diagnostic Supervisor. |
| 16 | Specifies that memory pages with Corrected Read Data (CRD) errors NOT be discarded at bootstrap time. By default, pages with CRD errors are removed from use during the bootstrap memory test. |
| <31:28> | Specifies the top level directory number for system disks with multiple systems. |

Table 7–8: R7 on CALYPSO series

| |
|---|
| CCA address in memory. |

## 7.7 SID registers and third party software

When we do an FCO to a VAX system, we often end up with a different SID (System Identification) register. If a customer has some third party software "tied" to the SID, then it will obviously stop working!!
To look at a SID in consol mode, do a >>>E/I 3E.
To look at a SID under VMS, do $A=F$GETSYI("SID") then $SHOW SYM A, it will be reported in decimal, hex and octal.

When you arrange to do an FCO, ensure the customer understands this.

**Figure 7–1: SID layout, part 1**

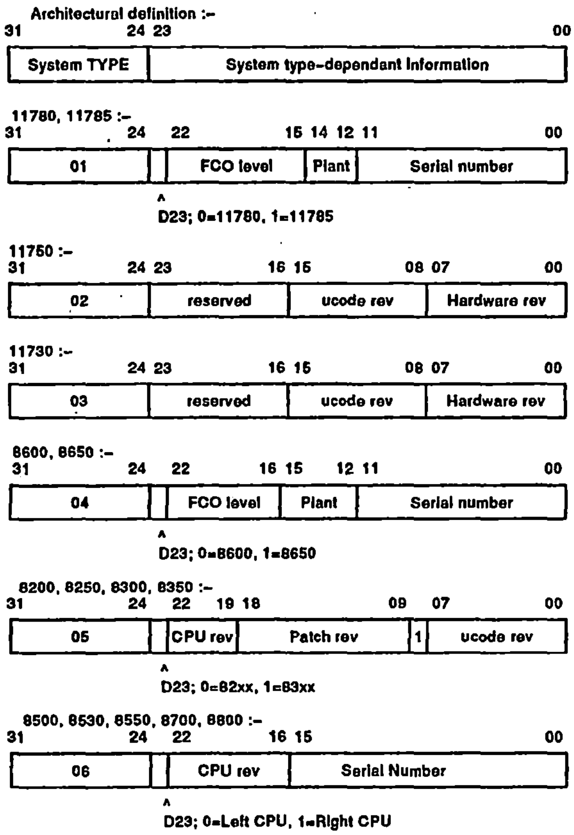
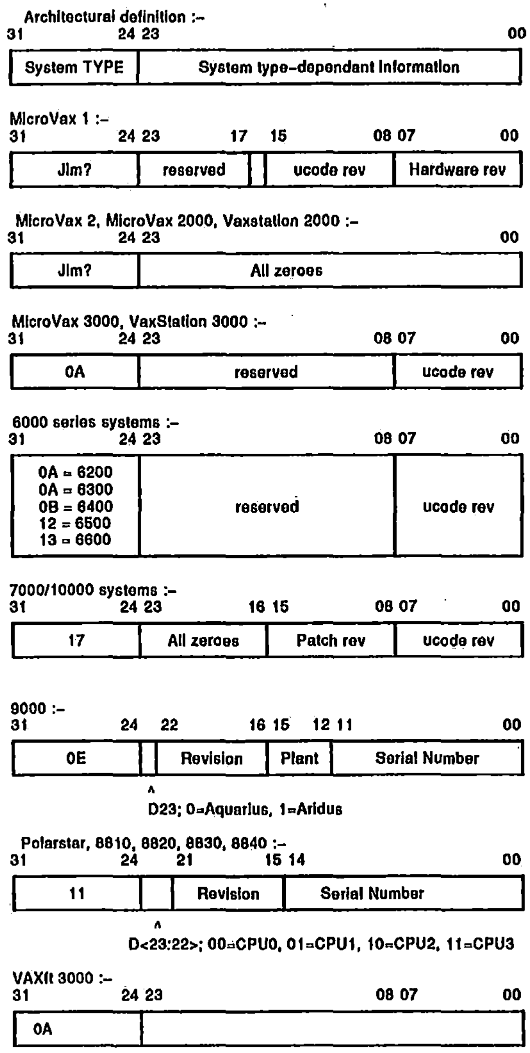


**Figure 7–2: SID layout, part 2**



Welwyn
Service Centre

F A C T F L A S H

**Options Affected:** **ELS systems and others**
**Submitted By:** **Jim Egginton**
**Date:** **16-SEP-1993**
**Filing Instructions:** **Chapter 7 VMS**

# Useful VMS commands to engineers in troubleshooting

**Some of these commands are dangerous to customers applications**

**Engineers should do nothing on customers software unless the customer is fully informed**

**of what the engineer is doing AND ITS CONSEQUENCES**

Having said this the engineer will not build up experience and knowledge of the operating system, by not experimenting and trying things out. A balance should be achievable between these two statements.

There are VAXs available in the office for engineers to gain experience, and practice on, use these.

This is by no means a complete or comprehensive list, more a list of commands very useful and sometimes little known by engineers, and those essential to troubleshooting to engineers, plus those I keep getting asked about. They vary from the simple that perhaps 90% of engineers know about to the complex.

For most of these commands the engineer will need full privileges, do a

```
$ SET PROC/PRIV=ALL
```

If the terminal comes back to the $ prompt with no message full privileges are granted.

## Directories

To show all the files on a disk

```
    $ DIR [*...]
```

To output this to a file which can be searched or edited add

```
    $ DIR [*...]/out=file.ext
```

Similarly, all files in an account

```
    $ DIR [egginton...]
```

The top level directory called the MFD (master file directory) is [000000]

```
    $ DIR [000000]
```
To show just the directories with size and numbers of files

```
    $ DIR [*...]/size/total
```
This is as close as you can get to DOS "tree" in VMS To show all the attributes of a file

```
    $ DIR welcome.txt/FULL
```
File is called welcome.txt

## Manipulating Devices

To show all the attributes of a device, called DKA300:

```
    $ SHOW DEVICE dka300:/FULL
```
To show all disk devices (to show all tapes use "m" instead of "d"

```
    $ SHOW DEVICE d
```
To turn off errorloging on a device

```
    $ Set DEVICE dka300:/NOERROR
```

## To test a disk without writing to it

```
    $ ANAL/DISK/READ dka300:
```
This actually is a check of the file structure, and will only read the parts of the disk written to. See my notes in Chapter 16 for more info on ANAL/DISK. This may show lots of file structure errors that are not hardware errors. So a $ SHOW ERROR before and afterwards.

Alternatively

```
    $ COPY/READ dka300:[*...]*.*;* NL:
```
This will copy all the files on the disk to a null device and READ_CHECK that it did this correctly. For these commands to work the device needs to be mounted, with a command such as

```
    $ MOUNT dka300: fred
```
where the label is FRED, if the label is not known then

```
    $ MOUNT dka300:/OVER=id
```

## To test a disk by writing to it

First create a directory on the disk to be tested (dka300:)

```
    $ CREATE/DIR dka300:[fred]
```
Then copy all the files from another disk to it with write check till it is full

```
    $ COPY/WRITE dka200:[*...]*.*;* dka300:[fred]
```
Then delete them

```
    $ DELETE dka300:[fred]*.*;*
```
This is a good data and seek test and needs both disks to be mounted . Obviously there needs to be room on the target disk to write on. Do a $ SHO DEV D to check the free blocks.

If it is required to write all over the disk do the following: Bear in mind all data on it will be lost.

```
    $ BACKUP/PHYSICAL/VERIFY dka200: dka300:
```
This will write to every track and sector of the disk and verify it. This needs like disk drives to work if they are different types do $ BACK/IMAGE... The target device (dka300: the one being tested) should be mounted foreign for this.

```
    $ DISMOUNT dka300:
            $ MOUNT/FOR dka300:
```

## To see if any hardware errors occurred

```
$ SHOW ERROR
```

This should be done before and after the testing command To show a full report of these commands use the "VMS Error Logging Reference Card" EY-1984E-ID-0101

## To Check a SCSI disk and do BBRs

Note that the following commands are for non DSA compliant disks (SCSI disks are not DSA).

See my notes on SCSI bad blocking in Chapter 16 for more info on ANAL/MEDIA commands.

The following will write to a SCSI disk, do 3 patterns and do BBRs where necessary, keeping all defect lists and showing (listing) all defect lists before and after writing patterns. The BBRS and defect lists displayed are the SDBBR lists not the lists from the actual SCSI device.

```
$ MOUNT/FOR dka300/data=write
      $ ANAL/MEDIA/EXER=(FULL,KEEP)/SHO=(BEFORE,AFTER) dka300:
```

The following will replace a bad block on a dka300, it will not write to it at all

```
$ MOUNT/FOR dka300:/DATA=WRITE
     $ ANAL/MEDIA/NOEXER/BAD=1050/SHO=(BEFORE,AFTER) dka300:
```

The following will list bad blocks from the SDBBR on a SCSI disk (note it must be mounted foreign)

```
$ ANAL/MEDIA/NOEXER/SHOW dka300:
```

## To check the network (DECNET)

```
$ MC NCP
     NCP> HELP
     NCP> SHOW KNOWN LINE    ! says line is SVA-0
     NCP> SHOW COUNTERS LINE sva-0
```

## To show memory allocation and possible bad pages

```
$ SHOW MEMORY/FULL
```

This will give a lot of information about memory being used for virtual addressing and other uses

## Commands useful for monitoring system activity

```
$ SHOW SYSTEM
```

This will show all processes open on the system and VMS version and uptime since last boot.

```
     $ MONITOR CLUSTER
     $ MONITOR ALL
     $ MONITOR PROC/TOPCPU
```

Monitor cluster is useful even when no cluster is present. Monitor all shows 5 seconds worth of every monitor screen available, the one wanted can then be displayed directly.

## Checking and stopping individual processes

Doing $ SHOW SYS will very often isolate the hung process (say from a faulty tape drive process can be recognised ), this can be stopped by

```
$ STOP PROC/ID=2220016B
```

where the process identifier to be stopped is 2220016B

To check whether the process is active/doing anything or is truly hung

```
$ SHOW PROC/id=2220016B/CONTINUOUS
```

### To test a tape drive after repair

These commands can be done separately or as a command file, start by making sure the tape is dismounted available (online) and do $ SHOW ERROR.

Find a directory such as SYS$HELP or SYS$LIBRARY that has not got customer files, where there is a suitable amount of files that are copyable. *.HLB files in SYS$HELP are 17 Mbytes 75 files so this is a fair test for most purposes.

```
$ loop:
$ init mka500: fred
$ mount mka500: fred
$ copy sys$help:*.hlb mka500:
$ dir mka500:
$ dismou mka500:/nounload
$ sho err
$ goto loop
```

Alternatively instead of the mount copy and dir

```
$ mount/for mka500:
$ backup/rew sys$help:*.hlb mka500:fred.bck
$ backup/rew/list mka500:
```

Backup does much less tape movement than copy, but transfers much more data.

### Doing commands and leaving your terminal free

The following spawn will start a subprocess, so that if it hangs (a faulty device) you still have use of the terminal (especially useful if you are on the console terminal)

```
$ SPAWN/NOWAIT  INIT MKA500: fred
```

The following submit will submit a command file (such as the one above called tape_test.com) to a que called sys$batch and output from it will come out on a que sys$print. Provided these are in order, (as default when VMS is delivered) it will work.

```
$ SUBMIT tape_test.com
```

### Typical standalone backup commands for backing up and restoring whole disks

To backup a disk and verify

```
$ INIT mka500: back
    $ BACK/IMAGE/VERIFY/REW dka300: mka500:back.bck/label=back
```

To restore to the disk

```
$ BACK/IMAGE/VERIFY/REW mka500:back.bck/label=back dka300:
```

The time taken to do backups and restores is very difficult to gauge, but a good report on progress can be got by doing a control T as often as required.

Between like devices (disks) a physical ($ BACK/PHYSICAL....) backup can be done which is much faster and does a block by block copy rather than file by file as with IMAGE.

Customers backup command lines are often very long and involved. They often include switches that are no longer supported, or items that are designed for maximum throughput with serious trade offs that sometimes come amis. Defaulting all the parameters as in the above commands is therefore a sanity check that the device is probably not at fault.

Welwyn/Newmarket Service Centre

F A C T F L A S H

**Options Affected:** VMS and BACKUP/VERIFY
**Submitted By:** Jim Egginton
**Date:** 22-DEC-1993
**Filing Instructions:** VMS Chapter 7

## Backup Verify and Analyse/disk compare errors

Symptoms:

OpenVMS BACKUP returns VERIFYERR with Volume "high water marking" enabled during the verify phase. There will be no errorlog entries due to this problem, but the console messages (see below) seem to be hardware related.

ANALYSE/DISK also uses the VERIFY UTILITY so it suffers the same problem.

A BACKUP/VERIFY operation may produce VERIFYERR errors when saving indexed files on a disk volume that has highwater marking enabled. The following illustrates this behaviour:

```
     $ BACKUP/VERIFY file.idx backup.bck/SAVE_SET
     %BACKUP-E-VERIFYERR, verification error for block n of disk:[dir]file.idx;1
     %BACKUP-S-COMPARED, compared disk:[dir]fle.idx;1

     %BACKUP-E-VERIFYERR, verification error for block n of disk:[dir]file.idx;1
     %BACKUP-S-COMPARED, compared disk:[dir]fle.idx;1

     etc.  There may be thousands of blocks reported like this
     then eventually may get

     %BACKUP-F-VERIFYERR, error reading beyond EOF

$
```

## Fixes and Work Rounds

1. VMS Versions applicable to patches:
   5.0, 5.0-1, 5.0-2, 5.1, 5.1-B, 5.1-1,
   5.1-2, 5.2, 5.2-1, 5.3, 5.3-1, 5.3-2,
   5.4, 5.4-1, 5.4-2, 5.4-3, 5.5, 5.5-1,
   5.5-2
   VMS AXP, Version 1.0

   Problem is fixed in VMS:
   6.0+
   VMS AXP Version 1.5
   It is unknown by me whether 5.5-2H4 has this fix

   The problem is in Backup Utility. Patches are available from the Customer Support Centre
   CSCPAT_1101 Applies To: OpenVMS VAX V5.2 - V5.5-2 ONLY
   CSCPAT_1001 Applies To: OpenVMS VAX V5.5 - V5.5-2 ONLY Both kits MUST be installed to fix this problem. The kits can be installed in any order.

2. Disable highwater marking. In some versions of VMS it came enabled by default. In a lot of cases if the customer knew what it was he would have disabled it anyway. Explain to the customer what it is and if he does not know recommend he disables it then do it. A reboot will be necessary for it to take effect. If the customer does want it maybe he can disable it before issuing the BACKUP command then when the BACKUP operation is complete, turn highwater marking back on. The following two DCL commands should be used:

```
$ SET VOLUME/NOHIGHWATER_MARKING        !to disable HWM
$ SET VOLUME/HIGHWATER_MARKING          !to enable HWM
```

## Analysis

**My explanation** of what High Water marking is. The term is far from self explanatory.

With High Water marking enabled on a disk, all files that are deleted have all blocks associated with the file erased. Normally only the directory entry is deleted, the spare blocks with data in just go back in the free list for use by someone else. Hi water marking is a security feature to stop users reading someone else files if something goes wrong with VMS.

There is naturally a big overhead to this. If a file is deleted that spreads over the whole disk, it will take 10 mins or more instead of a fraction of a second. A purge (very often done by batch cleanup jobs) will take extra time. VMS creates loads of files that have to be continually deleted.

It does this by taking note of where the disk is full to (high water mark) which is termed EOF (disk end of files), and constantly adjusting this level as files are extended etc.

If the customer is not extremely security sensitive between users on the system, advise him he does not need it, because of the problems engendered and reduction in performance.

The problem is seen because backup backs up blocks that are subsequently moved by high water marking, and thus may well be beyond EOF when verify is done. Hence the EOF errors that are terminal (the compare errors are not).

**Official explanation from stars article**

This problem is seen mostly when an index file's highwater mark is below the file's allocated space and remnants of previous files are found beyond the highwater mark.

Rarely, this problem is also seen with sequential files. For this to occur, the file's highwater mark must be programmatically moved to below the file's End of File (EOF) mark.

BACKUP does not clear its buffer before reading in file blocks in the ISSUE_V_QIOS routine. In case of a fresh buffer from the pool, the buffer still contains zero's and no VERIFYERR error is reported. However, if is a recycled buffer, it contains garbage from the previous usage. If the location of the buffer is also beyond the file's highwater mark (HWM), the garbage will be written to the save set.

When BACKUP does it's compare operation during the verify pass, all buffers involved are zeroed as an initial fix to that problem in VMS 5.4-3 or BACKUP X-10. However, if we are above HWM, we now compare the correct zeroed out buffer contents with unexpected garbage in the save set causing the VERIFYERR again to be returned.

Welwyn/Newmarket Service Centre

F A C T F L A S H

| | |
|---|---|
| **Options Affected:** | **VMS Ver 5.2 through 5.5-2H4** |
| **Submitted By:** | **Jim Egginton** |
| **Date:** | **6-MAY-1994** |
| **Filing Instructions:** | **Chapter 7 VMS** |

## Backup Quotas on VMS 5.x - SCSI bus problems

If the quotas in the account that is running a backup command or calling a batch job running a backup, are incorrect, then errors can occur that look exceedingly like a hardware problem. The following are those I strongly suspect could be put down to this problem.

1. FBC (fatal bug check) crashes with "above ASTDEL" or "invalid service exception" with backup.exe running, at IPL 8, and the disk driver (DKdriver) on the stack.
2. Mount Verify's on any disk on a SCSI bus that quickly recover or cause FBC crash.
3. SCSI phase errors.
4. Tape blank checks and device error - error positioning tape
5. Exceeding virtual memory capacity console messages.

Most of the above occur because when the system runs out of resources (memory mostly) the system hangs, causing phase errors on the scsi, or incorrectly terminates a backup, causing error positioning tape/blank checks on subsequent save sets.

## Recomended Backup Quotas for VMS 5.2 through 5.5-2H4

```
Username: SAVRES                                  Owner:  SAVE AND RESTORE
Account:  SYSTEM                                  UIC:    [1,40] ([SAVRES])
CLI:      DCL                                     Tables: DCLTABLES
Default:  SYS$SYSDEVICE:[SAVRES]
LGICMD:   LOGIN
Flags:
Primary days:   Mon Tue Wed Thu Fri
Secondary days:                     Sat Sun
No access restrictions
Expiration:  1-JAN-1995 00:00    Pwdminimum: 15   Login Fails:     0
Pwdlifetime:         30 00:00    Pwdchange:  16-MAR-1994 18:23
Last Login:  6-APR-1994 19:44 (interactive), 26-APR-1994 18:01 (non-interactive)
Maxjobs:           0  Fillm:        128  Bytlm:         65536
Maxacctjobs:       0  Shrfillm:       0  Pbytlm:            0
Maxdetach:         0  BIOlm:        128  JTquota:        1024
Prclm:             2  DIOlm:       4096  WSdef:         16384
Prio:              4  ASTlm:       4096  WSquo:         16384
Queprio:           0  TQElm:         10  WSextent:      16384
CPU:          (none)  Enqlm:        256  Pgflquo:       65536
Authorized Privileges:
  TMPMBX OPER NETMBX VOLPRO SYSPRV BYPASS
Default Privileges:
  TMPMBX OPER NETMBX VOLPRO SYSPRV BYPASS
```

## VMS 6.x+ Quotas

VMS Ver 6.x uses memory differently to Ver 5.x (See my factflash on autosizing memory), therefore quotas are different. If version 6.x quotas are used on a version 5.x system, an image backup will not even start.

The quotas can be found in the V6.0 system managers notes, the main differences are in working size quotas and need to be worked out for best performance depending on memory size. For 64 Mb you should have WSdef 256, WSquo 32800, WSext 32800, and PGFLquo 32800

Welwyn/Newmarket
Service Centre

F A C T F L A S H

**Options Affected:** **All disks and tapes**
**Submitted By:** **Jim Egginton**
**Date:** **20-MAY-1994**
**Filing Instructions:** **Chapter 7 VMS**

## Cannot see MSCP devices under VMS - Configuring them - SYSGEN>show scsi_noauto

### MSCP Devices

If you bring up a VMS system minimum a

```
$ MC SYSGEN A A/LOG
```

will normally configure devices into the system tables so they can be accessed.

However **MSCP devices will not be configured**, you need to do a -

```
$ @sys$system:startup.com configure    !now wait a few minutes to complete spin ups etc.
$ mc sysgen a a/log                    !should list all devices it can see
```

This should bring in any devices that can be seen from the console.

### SCSI Devices

SCSI devices are not MSCP devices, unless they are on an HSxxx, StorageWorks DSSI to SCSI RAID controller, or some non digital device.

If SCSI devices cannot be seen from VMS, the above as for MSCP devices can be tried, or the reason maybe due to a SYSGEN param "SCSI_NOAUTO", which means do not configure the devices masked out by this parameter, because the system wants to configure them specially itself. Such devices that need this are Optical Jukeboxes, IEEE and BITBUS instrument buses, Scanners and similar devices, in fact any system that has something other than disks or tapes on the SCSI bus.

In order to configure all devices do a

```
SYSGEN>set scsi_noauto 0
SYSGEN>a a/log
```

As soon as quit to sysgen is done the param scsi_noauto will revert to what it was because it is not dynamic. If you need to do it permanently, a "SYSGEN>write current" must be done before leaving sysgen and the system rebooted. Or just change it at the >SYSBOOT prompt when booting conversationally.

The **bit mask bit in scsi_noauto have the following weightings** in the 32 bit word and are hex aligned:

| | |
|---|---|
| Bits 0-7 . . . | scsi bus A id's 0-7 |
| Bits 8-15 . . . | scsi bus B id's 0-7 |
| Bits 16-23 . . . | scsi bus C id's 0-7 |
| Bits 24-31 . . . | scsi bus D id's 0-7 |

Welwyn/Newmarket
Service Centre

F A C T F L A S H

| | |
|---|---|
| **Options Affected:** | **VMS 6.x** |
| **Submitted By:** | **Jim Egginton** |
| **Date:** | **6-MAY-1994** |
| **Filing Instructions:** | **Chapter 7 VMS** |

## VMS Ver 6.x does not autosize memory

OpenVMS V6.0 system does not automatically configure all available memory at boot time as it did under previous versions of VMS. When adding new memory, it becomes necessary to adjust the SYSGEN parameter PHYSICALPAGES. Prior to version V6.0 the maximum value of PHYSICALPAGES was 1048756 (512 MB). This value has been changed to 7340032 pages (3.5 GB) at V6.0. Adjusting WORKING SETS and the dump file is also advisable after adding memory.

### Procedures

These procedures outline methods for either PRESETTING SYSGEN values, or making adjustments AFTER new memory has been installed on a system using AUTOGEN. Presetting values eliminate an additional reboot.

### PRESETTING BEFORE MEMORY INSTALLATION

1. Add a line in the following format to SYS$SYSTEM:MODPARAMS.DAT

```
MEMSIZE=Total-number-of-pages-of-memory-after-upgrade

          For example: MEMSIZE = 2048 * n
                       (where "n" = # of megabytes)
```

2. Run AUTOGEN to calculate new SYSGEN parameter values

```
For example: @SYS$UPDATE:AUTOGEN SAVPARAMS SETPARAMS
```

3. Perform the hardware upgrade to add the additional memory and reboot the system
4. Edit MODPARAMS.DAT to remove the line added in Step 1

### AFTER MEMORY INSTALLATION

1. Add a line in the following format to SYS$SYSTEM:MODPARAMS.DAT

```
PHYSICALPAGES=Total-number-of-pages-of-memory-after-upgrade

          For example: PHYSICALPAGES = 2048 * n
                       (where "n" = # of megabytes)
```

2. Run AUTOGEN to calculate new SYSGEN parameter values

```
For example: @SYS$UPDATE:AUTOGEN SAVPARAMS REBOOT
```

ADJUSTING WORKING SETS AND DUMPFILE SIZE:

If you want to allow users to use more memory, raise the UAF limit WSEXTENT. Be sure that this does not exceed the SYSGEN parameter WSMAX. If it does, raise WSMAX by running AUTOGEN after modifying MODPARAMS.DAT to include the parameter value.

Adjust the size of the DUMPFILE to accommodate the new memory size.

REFERENCES:

"OpenVMS VAX Version 6.0 Release Notes", May 1993, (AA-PV6ZA-TE), Pages 2-39, 2-63

Welwyn/Newmarket
Service Centre

F A C T F L A S H

| | |
|---|---|
| **Options Affected:** | **VMS Ver 6.1** |
| **Submitted By:** | **Jim Egginton** |
| **Date:** | **10-MAY-1994** |
| **Filing Instructions:** | **Chapter 7 VMS** |

# VMS 6.1 now VAX and AXP Compatible

This abstract from a letter is being mailed world wide to all customers on OpenVMS VAX and AXP media update service prior to release of OpenVMS V6.1

With VMS Ver 6.1, VAX and AXP operating systems have reached functional equivalence. In recognition of this achievement, we have changed the numbering scheme for the OpenVMS AXP operating system to match OpenVMS VAX numbering. The OpenVMS AXP Version 6.1 operating system release will directly follow the earlier OpenVMS AXP Version 1.0 and Version 1.5 releases.

It is very important for Digital's customers to know that the same user environment, reliability, availability, Desktop to datacenter scalability, and the same standards compliance are available on both the OpenVMS AXP and the OpenVMS VAX operating systems. This allows customers to build on their investment in VAX systems, while providing the opportunity to take advantage of the raw computing power of Alpha AXP systems.

## Features and Enhancements

OpenVMS VAX Version 6.1 includes the following new features and enhancements:

- FULLNAMES support for DECnet/OSI
- Reintegration of OpenVMS VAX Version 5.5-2H4 hardware support: (Note that 6.0 did not have the device support 5.5-2H4 did)
  VAX 4000 Models 50/100A/500A/600A/700A
  KDZZA on a Micro/VAX 3100 Model 90
  DEFTA (FDDI to TURBOchannel) and DEFQA (FDDI to Q-BUS) adapters
  SCSI-2 Tagged Command Queuing support for the Storageworks RAID Array 110 Subsystem
  XQP+ for Pathworks with some enhancements
  Per-disk licensing for Volume Shadowing
  C2 certification maintained
- Full VMScluster support
- User-written device drivers (Step 2)
- Volume shadowing Phase II (remaining configurations)
- Merge of features and bug fixes from OpenVMS VAX Versions 6.0 and 6.1
- New Hardware Support:
  Digital 2100 Server Model A500MP
  KFMSB support for DEC 7000/DEC 10000 systems

- Reintegration of OpenVMS AXP Version 1.5-1H1 hardware support: 3
  - DEC 2000 Model 300 AXP
  - DEC 3000 Models 600/600S/800/800S AXP
  - PMAZC TURBO channel adapter
  - C2 security (without certification)
  - Installation via PCSI (POLYCENTER Software Installation)

DEC Availability Manager for Distributed Systems (DECamds) Version 6.1 is now included as a separate saveset on the both the OpenVMS VAX and AXP distribution media. It is enabled with the VAXcluster or VMScluster Software license, or with the DECamds Version 1.1 license for existing DECamds customers.

Though OpenVMS AXP and VAX do provide functionally equivalent capabilities, there are some small differences which will be ad- dressed in the next version of OpenVMS AXP. OpenVMS AXP Version 6.1 does not include the following features which are provided in OpenVMS VAX Version 6.1:

- FULLNAMES support for DECnet/OSI
- SCSI-2 Tagged Command Queuing support for the Storageworks RAID Array 110 Subsystem
- Layered Product Availability

OpenVMS Engineering has worked diligently to minimise the impact OpenVMS VAX Version 6.1 has on Digital layered products. As a result of this effort, most layered products will work with OpenVMS VAX/AXP Version 6.1 when it initially ships.

## Planning Considerations for OpenVMS VAX/AXP Version 6.1

- OpenVMS VAX V5.5-2[nn] is the minimum version required in order to perform an upgrade to OpenVMS VAX Version 6.1. Customers running OpenVMS VAX A5.5[nn] must convert the batch and print queuing system before proceeding with an upgrade to Version 6.1.
- OpenVMS AXP Version 1.5 is the minimum version required in order to perform an upgrade to OpenVMS AXP Version 6.1.

OpenVMS VAX Version 6.0 was the last functional release to support Volume Shadowing Phase I.

Mixed Architecture Support

OpenVMS AXP Version 6.1 and OpenVMS VAX Version 6.1 provide two levels of support for mixed-version and mixed-architecture VMSclusters. These two support types are WARRANTED and MIGRATION.

Warranted support is provided for several pairs of OpenVMS VAX and OpenVMS AXP versions. Warranted support means that Digital has fully qualified the two versions coexisting in a VMScluster, and will answer all problems identified by customers using these configurations.

Migration support is a superset of the Rolling Upgrade sup- port provided in earlier releases of OpenVMS, and is available for mixes that are not warranted. Migration support means that Digital has qualified the versions for use together in con- figurations which are migrating in a staged fashion to a newer version of OpenVMS or to Alpha AXP. Problem reports submitted against these configurations will be answered by Digital. How- ever, in exceptional cases Digital may request that you move to a warranted configuration as part of answering the problem. Migration support will help customers move to warranted VMScluster version mixes, with minimal impact on their cluster environments.

DECnet for OpenVMS Users The OpenVMS VAX Version 6.1 kit installs the DECnet-VAX (DNA Phase IV) software. If you are currently running any of the DECnet Phase V products (DECnet-VAX Extensions, DECnet/OSI Version 5.5 for OpenVMS VAX, or DECnet/OSI Version 5.6 for OpenVMS VAX), you need to be aware of the current status of the network software before you install or upgrade to Version 6.1 of the OpenVMS VAX Version operating system. The DECnet-VAX Extensions product is not supported for OpenVMS VAX Version 6.1. The network products associated with DECnet-VAX Extensions (P.S.I., WANDD, VOTS, OSAK, and FTAM) are no longer available separately. If you need these products, you will need to migrate to Version 5.7 of DECnet/OSI for OpenVMS. See the DECnet/OSI installation and configuration manuals for information about planning for and implementing DECnet/OSI for OpenVMS software.

Welwyn/Newmarket
Service Centre

F A C T F L A S H

| | |
|---|---|
| **Options Affected:** | **OpenVMS Version 6.2** |
| **Submitted By:** | **Dave Bazley** |
| **Date:** | **02-May-1995** |
| **Filing Instructions:** | **File at end of Chapter 7, VMS** |

## OpenVMS 6.2 - New Features

OpenVMS 6.2 is due to start shipping at the end of May. This factflash explains new product support, upgrade paths and new features. These notes apply to OpenVMS V6.2 AXP and VAX. For more details read the release notes.

### V6.2 New VAX System Support

- VAX 4000-705A
- VAX 4000-106A
- VAX 4000-55
- VAXstation 4000-90A
- VAX 3100-96
- VAX 3100-85

### V6.2 New AXP System Support

- EV5 systems:
  Alphaserver 8000 5/xxx (Turbolaser)
  Alphaserver 2000 5/xxx (Gamma)
  Alphastation 600 5/xxx (Alcor)
- Systems "re-integrated" from V6.1-1H1/2:
  Alphaserver 1000 4/xxx (Mikasa)
  Alphaserver 400 4/xxx (Chinet)
  Alphastation 600 4/xxx (M3)
  Alphastation 200 4/xxx (Mustang)

### V6.2 New Option support

- KFESB - EISA to DSSI
- KZPSA - PCI to FWD SCSI
- KZPBA - PCI to FWD SCSI
- KZPSC - PCI to SCSI RAID controller
- KZTSA - Turbochannel to SCSI (AXP only)
- HSD30 - DSSI StorageWorks controller
- HSZ40 - SCSI StorageWorks controller

## V6.2 Upgrade Paths

- VAX upgrade paths:
  - V5.5-2 to V6.0 to V6.2
  - V5.5-2 to V6.1 to V6.2
  - V6.0 to V6.2
  - V6.1 to V6.2
- AXP upgrade paths:
  - V6.1 to V6.2

## V6.2 New features

- DCL enhancements:
  - Automatic Foreign Commands (Similar to "Path" in DOS and UNIX)
  - New item codes for F$GETSYI
  - SET TIME/CLUSTER now supported
  - SHOW DEVICES/REBUILD_STATUS (To see if volumes need rebuilding)
  - SHOW PROCESS/DUMP
  - New SHOW USER qualifiers
  - New SHOW SYSTEM qualifiers
  - BOOT and LOGIN added to /SINCE= and /BEFORE=
  - EQV and XOR added to SEARCH/MATCH
  - Recall buffer now 256 deep for VAX and AXP
  - "Numerous" other DCL enhancements
  - "Many" fixes to DCL documentation
- OpenVMS Management Station is bundled into V6.2:
  - Windows-PC based
  - Two components: Windows-PC client + OpenVMS Server
  - First release performs User account management
  - Future releases aimed at Printer and Storage management
- Various Autogen enhancements:
  - SYSMWCNT calculation changed - 200/400 pages bigger now
  - Info messages can be disabled
  - Other changes for DECnet
- New Backup Manager:
  - For unskilled or occasional Backup users
  - Runs on any VT type screen
  - All usual functions available
- Several new LAT features:
  - SET HOST/LAT/DIAL=(NUMBER:xxx,MODEM_TYPE:xxx)
  - SET HOST/LAT can now be used in a command procedure
  - Intrasystem LAT connections allowed (Like DECnet SET HOST 0)
  - New SET HOST/LAT qualifiers
  - Can limit services
  - Some LATCP wildcards allowed
- New Logicals for SHUTDOWN:
  - SHUTDOWN$DECNET_MINUTES - Minutes left when DECnet is stopped
  - SHUTDOWN$QUEUE_MINUTES - Minutes left when Queues stopped
- Several new Security Services:
  - Impersonation - Allows a user (with appropriate privs) to assume another security profile
  - Proxy - Allows an application to perform proxy database manipulation
  - Intrusion detection - Allows an application to access the intrusion database

- DECwindows Motif V1.2-3:
  - Incorporates OSF/Motif R1.2.3 and MIT X11 Release 5
  - 600 toolkit bugfixes over 1.2.2
  - NCSA Mozaic web browser bundled in
  - "Console" window allows OPA0: access - Replaces Cntl-F2 toggle
- One documentation set for AXP and VAX:
  - OpenVMS full set
  - OpenVMS subset
  - 20 books archived - in .PS on CD
  - Subkits eliminated
- OpenVMS Freeware CD:
  - Shipped to all customers, contains:
  - Retired products
  - Digital internal tools
  - Popular Internet programs
  - Unix utilities
  - Toys/Games
- Several New Cluster improvements:
  - Support for SCSI Clustering
  - Cluster Client Licensing
  - TMSCP Server support for SCSI tapes
  - DECnet no longer required
  - DECamds Console support for AXP
  - Improved failover support
  - Improved Quorum update at node removal
- Last release to support DECnet Phase IV - DECnet/OSI fully supercedes Phase IV
- DECnet/OSI Full Names now supported for AXP
- First step to full TCP/IP Support - DCL + Manual so far
- Changes to Time and Date format in SHOW command
- Changes to Log and other OPCOM bits
- Simplified Installation
- VMSKITBLD no longer supported on AXP - Use AXPVMS$PCSI_INSTALL
- VAX Dumpfile can be off System disk + new values for DUMPSTYLE Sysgen parameter
- Backup Home Block no longer related to disk geometry - Allows shadowing of dissimilar disks
- XPG4 support included