OS-9
VERSION 2.4
RELEASE NOTES

ACKNOWLEDGEMENTS

I would like to thank Warren Brown and Ryan Babic for providing the preliminary documentation and for their expertise and helpfulness.

COPYRIGHT AND REVISION HISTORY

This manual reflects Version 2.4 of the OS-9 Operating System.

Publication Editor: Ellen Grant
Publication Date: October, 1990
Product Number: R2468NA68MO

DISCLAIMER

The information contained herein is believed to be accurate as of the date of publication. However, Microware will not be liable for any damages, including indirect or consequential, from use of the OS-9 Operating System, Microware-provided software or reliance on the accuracy of this documentation. The information contained herein is subject to change without notice.

REPRODUCTION NOTICE

The software described in this document is intended to be used on a single computer system. Microware expressly prohibits any reproduction of the software on tape, disk or any other medium except for backup purposes. Distribution of this software, in part or whole, to any other party or on any other system may constitute copyright infringements and misappropriation of trade secrets and confidential processes which are the property of Microware and/or other parties. Unauthorized distribution of software may cause damages far in excess of the value of the copies involved.

For additional copies of this software and/or documentation, or if you have questions concerning the above notice, the documentation and/or software, please contact your OS-9 supplier.

TRADEMARKS

OS-9 and OS-9/68000 are trademarks of Microware Systems Corporation.

Microware Systems Corporation • 1900 N.W. 114th Street
Des Moines, Iowa 50325-7077 • Phone: 515/224-1929

Table of Contents

Application Notes

Documentation Corrections

Known Problems

End of Table of Contents

Overview

Overview

The ***OS-9/68000 Version 2.4 Release Notes*** contain descriptions of all changes to Microware's software since the Version 2.3 Release. The release notes are divided into the following chapters:

Chapter 1 ***New OS-9 Features***
This chapter describes the new features in the Version 2.4 Release.

Chapter 2 ***Software Corrections and Enhancements***
This chapter describes the corrected problems in the Version 2.4 Release.

Chapter 3 ***Application Notes***
This chapter contains discussions of several topics relating to the Version 2.4 Release.

Chapter 4 ***Documentation Corrections***
This chapter lists all known bugs in the existing OS-9 documentation.

Chapter 5 ***Known Bugs***
This chapter lists all known bugs in the Version 2.4 Release software.

NOTES

1

New OS-9 Features

Introduction

This chapter provides an overview of the new features that have been added for the Version 2.4 release:

- **Variable Sector Size RBF**
  RBF now supports variable sector sizes. The size of a logical sector can be any integral binary power from 256 to 32768 (for example, 256, 512, 1024, etc.). Further information is provided in the ***RBF Variable Sector Support*** application note, the ***OS-9 OEM Installation Manual***, and the ***OS-9 Technical I/O Manual***.

- **RBF Caching Support**
  RBF now supports disk caching. Full details on this feature are located in the diskcache utility documentation (later in this chapter) and the ***RBF Disk Caching Support*** application note.

- **Large and Non-Contiguous Boot Files**
  The OS9Gen utility has been updated to support large (greater than 64K) and non-contiguous boot files. Further information is provided in the ***Extended and Non-Contiguous Boot Files*** application note and the ***OS-9 OEM Installation Manual***.

- **New ROM Debugger (RomBug)**
  A new ROM debugger is available and is documented in the ***OS-9/68000 ROM Debugger User's Manual***. Further information on the selection of debuggers and RomBug's impact on ROM boot-code is provided in the ***OS-9 OEM Installation Manual***.

- **C Booting Technology**
  ROM boot-code can now be developed using C language technology. Further information is provided in the ***OS-9 OEM Installation Manual.***

- **ROM Customization**
  Custom initialization of the system's "primitive initialization" routines is now support using the InitExt.a file. Further information is provided in the ***OS-9 OEM Installation Manual.***

New Utilities

The following new utilities and built-in shell commands have been added. These are fully documented in the following pages.

- diskcache
- profile
- tapegen

diskcache — Enable, Disable, or Display Status of Cache

SYNTAX: `diskcache [<opts>] [<dev>]`

FUNCTION: diskcache enables, disables, or displays the status of the cache. Caching may be enabled for any type of RBF device, and more than one device may be cached at a time.

The total amount of system memory used for caching all enabled drives can be set by the utility's -t option. If not explicitly defined, the diskcache utility automatically selects a reasonable value based upon the amount of free system memory.

Caching may be dynamically enabled or disabled on a per drive basis while the system is running using the -e and -d options.

Statistical information regarding the hit/miss ratios, amount of memory allocated, etc. can be inspected on a drive by drive basis using the -l option. An example output of this information follows:

```
Current size =  1047552
  Size limit =  1048576

Device:  /h0:1:1
                Requests    Sectors       Hits       Zaps     >2 Xfr Hit Rate
     Reads:        47592      55436      21874        143        662  39.5%
    Writes:         7723       8065                   7342         68
 Dir Reads:        54048      54048      34526      18387<-Sctr Zero  63.9%
Dir Writes:            0          0
 Hit compares =      63399 (          1/hit)
Miss compares =      92685 (          3/miss)
```

CAVEATS: Caching should only be invoked on devices that are known to the I/O system (that is, the devices should have been initialized with the iniz utility).

If caching is to be enabled on drives with different sector sizes, the device with the largest sector size should be included in the initial cache enabling. Attempting to add a drive (with a sector size larger than any currently cached drive) to the cache system after initial cache startup results in continuous "misses" for that drive, as the sector size is too large.

OPTIONS:	-?	Displays the options, function, and command syntax of diskcache.
	-d	Disables cache for <dev>.
	-e	Enables cache for <dev>.
	-l	Displays the cache status for <dev>.
	-t=<size>[k]	Specifies the size limit of the total cache.

profile — Read Commands from File and Return

SYNTAX: `profile <filename>`

FUNCTION: profile causes the current shell to read its input from the named file and then return to its original input source which is usually the keyboard.

The file specified in <filename> may contain any utility or shell commands, including those to set or unset environment variables or to change directories. These changes will remain in effect after the command executes. This is in contrast to calling a normal procedure file by name only, which would then be executed by a child shell. This would not affect the environment or current directories of the calling shell.

You can nest profile commands. That is, the file itself may contain a profile command for another file. When the latter profile command is completed, the first one will resume.

A particularly useful application for profile files is within the .login and .logout files of a system's users. For example, if each user includes the following line in their .login file, system-wide commands (common environments, news bulletins, etc.) can be included in the file /dd/SYS/login_sys:

```
profile /dd/SYS/login_sys
```

A similar technique can be used for .logout files.

tapegen — Put Files on a Tape

SYNTAX: `tapegen [<opts>] <filename> <filename>`

FUNCTION: tapegen creates the "bootable" tape. tapegen is a standard utility that performs a function similar to the os9gen utility. Both utilities place the bootstrap file on to the media and mark the media identification block with information regarding the bootstrap file. In additional, tapegen can optionally place initialized data on the tape, for application specific purposes.

To use the tapegen utility, type tapegen followed by any desired options.

OPTION:

-?	Displays the options, function, and command syntax of tapegen.
-b=<bootfile>	Installs an OS-9 boot file.
-bz	Reads boot module names from standard input.
-bz=<bootlist>	Reads boot module names from the specified bootlist file.
-c	Checks and displays header information.
-d=<dev>	Specifies the tape device name. The default is /mt0.
-o	Takes the tape drive offline when finished.
-t=<target>	Specifies the name of the target system.
-i=<file>	Installs an initialized data file on the tape. This is usually a RAM disk image.
-v=<volume>	Specifies the name of the tape volume.
-z	Reads filenames from standard input.
-z=<file>	Reads filenames from the specified file.

EXAMPLES: The following example makes a bootable tape. The disk image is derived from the /dd device.

```
$ tapegen -b=OS9Boot.tape -i=/dd@ "-v=OS-9/68K Boot Tape" -t=MySystem
```

This example makes a bootable tape, with no initialized data file. The "header" information is displayed after writing the tape.

```
$ tapegen -b=OS9Boot.h0 -c
```

2

Software Corrections and Enhancements

Introduction

This section contains the descriptions of all corrected problems and enhancements in Microware's OS-9 software since the Version 2.3 Release.

These corrections and enhancements are divided into the following sections:

- OS-9 Utilities
- OS-9 Kernel
- OS-9 System Modules
- System Support Files
- ROM Updates
- DEFS
- OS-9 I/O Modules
- File Managers
- System Security Module (SSM)

OS-9 Utilities: New Features and Changes

This section describes the corrected problems and new features of the Version 2.4 utility set.

The -z=<file> option for the following utilities now allows the user to place comments in the <file>. A comment line is recognized when an asterisk (*) is the first character of the line:

attr	cfp	compress	count	deiniz	del	deldir
expand	fixmod	frestore	grep	ident	iniz	link
list	load	makdir	make	merge	pr	qsort
save	tape	touch	tr	tsmon	unlink	xmode

backup Add newline after "n" response.

The requests for source and destination disks have been cleaned up. In single-disk backup mode, backup no longer requests that you insert the source disk before exiting after the final write to the destination disk has been made.

The verify comparison loops used with backup have been improved.

cmp A test has been added for the following:

```
argv[0] = "-cmp"
```

If this condition is true, the -x option only applies to the first file. This fixes copy's -xv option.

code The code utility now works on GFM devices.

copy The -r option now uses create(), instead of creat(), to support S_ISIZE.

A -z option has been added. This allows files to be read from standard input or from files. When the -z option is used to read from a file, the file may contain comments.

A -f option has been added. This option allows you to rewrite the destination files if write permission is turned off.

copy has a new error message which informs you when copy cannot perform a compare with cmp.

You can now use upper-case replies to the continue prompt.

When cmp is forked, copy uses an argv[0] of -cmp.

Error checking has been added to the user y/n input.

count count now recognizes TAB, LF, and FF as word delimiters

dcheck dcheck has been modified to work with variable sector sized media. Refer to the application notes section for more information about variable sector sizes.

The -r and -y options have been added to the help message.

deiniz The error messages have been corrected.

del A -f option has been added. This option forces the deletion of files for which the user does not have write permission.

An -e option has been added for security reasons. This option completely erases the file data before releasing the file space.

The -z and -p options may now be used together. Previously, -pz was considered mutually exclusive.

deldir When deldir forks dir to show the directory being deleted, it uses the -a directory option. This allows file names beginning with a period (.) (such as .login) to be shown when the directory listing is shown.

Saved current directory to restore after processing each directory argument.

devs Corrected potential module overrun when copying module names.

dir Corrected problem when -zu was used on directory names.

The -z option now supports -z=<file>. <file> can include comments.

dir has been updated to work with variable-sector sized RBF and CDFM devices.

dsave Added prerr() to print the error number before asking whether to continue after an error occurs.

A -f option has been added. It is passed to the copy utility so that a copy -f command is executed.

Error checking has been added to getcwds().

dsave's error number reporting has been improved.

The "continue?" message is now sent to standard error. The user's response is now also accepted from standard error.

dump

A -m option has been added. It performs a module dump in memory.

A -s option has been added to specify the sector number of RBF files.

The address field is now 8 digits.

echo

echo now allows you to enter a single-digit hexadecimal number.

fixmod

fixmod -uo now checks for group 0, instead of user 0.

fixmod now checks for incomplete module headers.

fixmod now ensures that the module header is updated with new values even if the parity stays the same.

free

free works with variable-sector sized RBF devices.

The error "can't read device" now returns errno instead of 1.

Corrected largest block count problem. This was not reset correctly when displaying free status on multiple drives.

frestore

frestore only allows a volume prompt if standard input (stdin) is an OS-9 terminal (tty).

A -j option has been added. This allows you to specify the minimum block size for memory requests.

Fixed problem with releasing freed memory blocks to the system. The previous version would sometimes access a block that it had freed, thus causing a bus-trap on SSM systems.

Abortive terminations now exit non-zero.

An -a option has been added. This allows files to be overwritten if they do not have write permission.

When restoring a file, the file size is now pre-expanded.

fsave

fsave only allows a volume prompt if standard input (stdin) is an OS-9 terminal (tty).

A -j option has been added. This allows you to specify the minimum block size for memory requests.

grep

grep -l will now print a name with one file.

help

If there is no help available for a requested item, help returns exit status 1.

ident ident now checks for incomplete module headers.

irqs A 68070 on-chip autovector display for 68070 systems has been added.

The potential module overrun when copying module names has been corrected.

link When a link command is entered without parameters, the correct error message is now displayed.

login The dochain() now looks for the PORT environment variable instead of just the first environment variable.

maps The display of data memory has been reformatted.

If a process has not made a system call, then <none> is displayed instead of F$Link.

mdir The link count is now unsigned.

moded The operation of the DevCon fields has been corrected. The mechanism to specify the offsets of the DevCon fields has been modified. The offsets of DevCon fields are now relative to the start of the DevCon section, and not offsets from the module start. This allows new standard options to be added to descriptors without having to re-adjust all DevCon offsets.

A -d option has been added. This allows specification of an alternate descriptor file. The default alternate descriptor file is moded.fields.

An -e option has been added. This allows specification of an alternate error message file. The default alternate error message file is errmsg.

The default method of searching for the SYS directory to find the files moded.fields and errmsg has been changed to search as follows:

- Search device /dd first.
- Search the default system device, as specified in the Init module (M$SysDev). If the Init module cannot be linked to, the SYS directory is searched for on the current device.

Changes to a file are now written to the file when the w (write) command is specified, instead of waiting until a new module/file is specified or the program is exited.

Small modules are now correctly handled.

Os9Gen OS9Gen can now be used with variable-sector sized RBF devices.

An -e option has been added to support large and non-contiguous boots.

A -r option has been added. You can use this option to remove a boot file.

pr The end-of-page problem in prtrailer() has been fixed.

The column count is now correctly initialized.

The pr -f problem with form-feed has been corrected.

A -d option has been added. You can use this to specify the actual paper depth.

The date printed has been converted to the ISO standard yy/mm/dd.

procs Corrected last system call display when no calls have been made (<none>).

Also corrected negative Age (hh:mm).

rename rename does not allow you to rename . (current directory) or .. (the parent directory of the current directory).

romsplit romsplit is now faster. There is also a reduced chance of output file segment problems because the split is now performed in memory.

save Previously, when you tried to do a save -fz, the -z option would cause the -f option to work incorrectly. This has been temporarily prohibited.

setime Fixed problems with setime when executing with its data memory at an address of 0x80000000 or greater.

"am" and "pm" now appear without a leading separator.

shell shell was looking for Basic09 as the alternate run-time system when RunB could not be found. It correctly looks for Basic now.

When a setpr command is entered without parameters, the error is now correctly reported.

An error message is now given when you try to use unsetenv on a variable that does not exist.

_sh now decrements before chaining (ex command).

Error-checking has been added to pnum(). pnum() is unsigned.

setpr and kill are now capable of performing range-checking.

A new built-in shell command, profile, has been added. For more information about profile, refer to the section on new features.

The shell now allows execution of a standard (sub-shell) procedure file within .login and profile files.

tmode The problems with the bsl and nobsl options have been fixed.

If the device is neither a SCF device nor a GFM device, a 1 is returned.

"no" may only precede booleans.

You must use 1, 1.5, or 2 for stop bits.

touch Directories can now be touched.

touch no longer ignores error 214.

xmode The problems with the bsl and nobsl options have been fixed.

"no" may only precede booleans.

You must use 1, 1.5, or 2 for stop bits.

The problem with the normal option has been fixed.

OS-9 Kernel

The OS-9 kernel has undergone miscellaneous bug fixes and enhancements since the Version 2.3 Release. This section describes these changes.

The Version 2.4 kernel supports the MC68300 (CPU32) family.

P$Signal (for signal processing) in the process descriptor now contains the *last* signal received, and not the first signal in the queue. This change allows suspended drivers (drivers waiting for I/O) to not lock up waiting for I/O when a non-deadly I/O signal (any signal value of 32 or greater) is received, followed by a deadly signal.

The kernel's cache flushing methods have been made more intelligent. The Init module contains a new field, M$Compat2, that indicates the "absence/snoopiness" of the system caches, as well as whether the data-caches need to be disabled when calling the I/O system.

Under some circumstances, the high-byte of the SR could be corrupted when user intercept routines exited (F$RTE) causing unpredictable results. This has been fixed.

Under some circumstances, a process making its first F$SRqMem call with a size of -1 (a request for the largest available memory block) could cause a system crash. This has been fixed.

Previously, a system-state alarm routine could not install a new alarm routine. This has been fixed.

When creating a data module (F$DatMod), it was possible in some situations for the size granted to be 1 byte less than the requested size. The size granted is now always at least the requested size.

If the system's memory lists became corrupted, it was possible for the kernel to loop forever. This has been fixed.

The following problems with attaching and detaching devices have been fixed:

- Previously, if a driver was attached to two descriptors and you deinized one descriptor, the driver could be unlinked from memory, even though the driver was still active on the other descriptor.
- If the device attach exited with an E$BMode error, it was possible that the device table entry for the device would not be correctly removed from the device table.
- When performing an I$Attach on a device, the device name string is now only updated and returned to the caller when the device attach is fully successful.

During the system's cold start, it was possible that ROM'd modules would be missed during the module search. This only occurred if the module contained zero's at the memory search's block boundary.

When tracing the kernel from a debugger, the kernel could become confused as to who was being traced (the kernel or the "next process"). This has been fixed.

OS-9 System Modules

These changes relate to the files contained in the SYSMODS directory.

Init.a **System Init Module**

init.a has been updated to reflect the Version 2.4 release.

The Init module now contains a new field: M$Compat2. M$Compat2 allows control of the system caching functions. Refer to the application note on caching for further details.

SysBusErr.a **Generic "Soft Bus-Error" Handler**

This is a new file that provides the mechanisms to implement the handling of ***soft bus-errors*** for a system. Refer to the application note on Soft Bus-Errors for further details.

System Support Files

These notes detail changes to the files contained in the SYS directory.

ErrMsg

System Error Message File

The following error codes have been added to this file:

Error Number	Description
000:001	Process has aborted.
000:175 E$Hardware	Hardware damage has been detected. E$Hardware usually occurs when the driver fails to detect the correct responses from the hardware. This can occur due to hardware failure or an incorrect hardware configuration.
000:176 E$SectSize	Invalid Sector Size The sector size of an RBF device must be a binary multiple of 256 (256, 512, 1024, etc.). The maximum sector size is 32768.

Moded.Fields

Description File for Moded

Support for PD_MaxCnt (RBF devices) has been added.

Support for M$Compat2 (Init module) has been added.

Support for changing the port address (M$Port) of Pipe devices has been added.

Support for RBF high-density floppy definitions has been added (PD_TYP, PD_DNS, and PD_Rate).

The comments for PD_SSize (RBF devices) have been updated to reflect variable-sector sized RBF support.

The mechanism for specifying DevCon offsets has been changed. Previously, DevCon fields were specified as offsets from the module start (like all other fields). Now, they are specified as offsets from the start of the DevCon section. This change alleviates the need for the constant updating of this file whenever new fields are added to the standard options section.

ROM Updates

This section describes the changes and corrections in the ROM files since the Version 2.3 Release:

Boot.a

A MANUAL_RAM conditional has been added for systems that need to manually enable RAM before calling SysInit.

A REJECT0 conditional has been added to allow systems to reject non-existing ROM memory that responds with contents = 0.

Support for the MC68300 (CPU32) Family has been added. For example, boot.a supports the MC68332.

The debugger variable MPUType has been changed from a byte to a long-word.

ROM constants for the Vector Base Register and Address Translation Factor have been added. This allows boot code to access these values on a global basis, and end-users to more easily patch them.

The conditionals have been updated to support usage of CBOOT (C-code Booting) or RomBug (new ROM debugger).

Boot319.a

New port pak file for booting hard and floppy disks on the MVME319 Disk Controller. This driver supports variable-sector sized boots and also supports booting from boot files that are non-contiguous and/or greater than 64K in size.

Boot320.a

This is a new port pak file for booting hard and floppy disks on the MVME320 Disk Controller. This driver supports variable-sector sized boots and also supports booting from boot files that are non-fragmented and/or greater than 64K in size.

Debug (Old ROM Debugger)

Support has been added for the MC68300 (CPU32) family.

A flag in the debugger's globals has been added to ensure that calls to ConsSet and ConsDeIn are made on a one-to-one basis.

InitExt.a

This is a generic "sysinit extension" file. It allows end-users to create custom hardware initialization code. This code is merged on to the end of the normal ROM image by the end-user or integrator. Effectively, this allows end-user customization of the SysInit and SInitTwo routines (see sysinit.a) without the end-user having to remake the bootstrap image.

Vectors.a

The generation of the Reset SSP value is now controlled via the CBOOT (C-code booting) and RomBug (new debugger) conditionals. If neither conditional is defined, then the Reset SSP is set to be 4K above Mem.Beg (normal memory start), as is required for the old-style debugger. If either conditional is defined, the size of the debugger/boot global area is determined by the linker (i.e. the total of all vsect declarations for the linked files).

Rombug (new ROM debugger)

A new debugger is included in this release. This debugger is essentially a ROM-based version of the system-state debugger (sysdbg), and supports many new features over the old debugger (debug), such as symbolic debugging, disassembly of all MC68XXX opcodes, etc.

For full details of RomBug, you should refer to the ***OS-9/68000 RomBug User's Manual.***

DEFS

This section describes the changes associated with the DEFS files for the Version 2.4 Release. These changes also appear in the sys.l library.

funcs.a

The SS_VarSect GetStat code for variable-sector RBF/drivers has been added. When a path is opened to an RBF device, RBF calls the driver with this GetStat call. Refer to the application note on variable-sector sized RBF devices.

SS_FG and its subfunctions have been added for frame grabbers. Refer to the ***OS-9/RAVE DevPak Manual*** for more information.

PT_Calib has been added for UCM users. Refer to the ***OS-9/RAVE DevPak Manual*** for more information.

SN_ClutLnk has been added for GFM users. Refer to the ***RAVE Graphics File Manager*** manual for more information.

io.a

PD_CPInt replaces the reserved field.

The PD_MaxCnt field has been added to the RBF device descriptors. This field allows RBF device descriptors to specify a maximum byte-count that can be transferred between the device driver and disk device in a single I/O operation. The upper limit that can be transferred is device specific. For DMA-transfer devices, the upper limit is the lower value of the maximum DMA-count or the device's maximum transfer count. When an I/O request is made that requests a transfer count greater than this value, RBF deblocks the requests to the driver into sizes that the driver can safely handle. When deblocking occurs, this count is rounded down to an integral sector size count (for example, $ffff becomes $ff00 for 256-byte sectored media, $fe00 for 512-byte media, etc.). If this field is 0, RBF defaults to $ffff.

DD_MapLSN has been added to RBF sector 0 definitions. DD_MapLSN indicates the logical sector number of the start of the bitmap for the media. **NOTE:** Although this allows the bitmap to be located any where on the media, the current RBF and disk-utilities do not fully support this concept. This feature will be fully implemented in a future release of RBF and the disk-utilities.

DD_LSNSize has been added to RBF sector 0 definitions. The format utility writes this field when the media is initialized. DD_LSNSize indicates the size of the media's logical sectors. Bootstrap drivers can use this field to determine the media's logical sector size to correctly locate and read the boot file. If this field is zero, bootstrap drivers should assume a logical sector size of 256.

DD_VersID has been added to sector 0 definitions. The current Version ID for RBF media is 1. The format utility writes this field when the media is initialized. This field will be 0 for pre-Version 2.4 formatted media.

PD_SctSiz has been added to RBF's path descriptor options section. This field allows user programs to determine the media's logical sector size. If this field is zero (pre-Version 2.4 RBF), then the media logical sector size is assumed to be 256-bytes.

PD_Rate has been added to RBF's path descriptor options section for high-density floppy disk support.

module.a

The M$Compat2 field has been added to the Init module. M$Compat2 indicates cache control features to the kernel, as follows:

Bit 0	0	External instruction cache is NOT snoopy.
	1	External instruction cache is snoopy (or absent).
Bit 1	0	External data cache is NOT snoopy.
	1	External data cache is snoopy (or absent).
Bit 2	0	On-chip instruction cache is NOT snoopy.
	1	On-chip instruction cache is snoopy (or absent).
Bit 3	0	On-chip data cache is NOT snoopy.
	1	On-chip data cache is snoopy (or absent).

The snoopy/absent flags allow the kernel to make intelligent decisions as to when to actually flush the system's caches (with F$CCtl calls). If the system's hardware capabilities allow the caches to maintain coherency via hardware means, then these flags can be set so that the kernel performs on the required (or no) cache flushes.

Bit 7	0	Kernel disables data caching when in I/O.
	1	Kernel DOES NOT disable data caching when in I/O.

This flag allows the kernel to determine whether it is safe to keep data caching on during calls to the I/O system. For systems that have no DMA activity or have DMA-drivers that handle data-cache flushing, then the data-cache can be kept on during I/O system calls. Systems that have DMA-drivers that do not care for data-cache flushing will have to disable the data-cache during I/O operations to ensure that "stale data problems" do not arise.

sysglob

D_Compat2 has been added to the system globals. This field is a copy of the M$Compat2 field in the Init module.

D_SnoopD has been added to the system globals. This field is non-zero if *all* data caches are snoopy/absent (the data caches update themselves automatically when DMA activity occurs). DMA-style device drivers can use this field to determine whether an explicit flush of the data caches (F$CCtl call) is required or not after DMA activity has occurred.

sysio

RBF's file manager area definitions have been added for variable sector size support.

PD_ErrNo and PD_SysGlob have been documented in the I/O global section of the path descriptor.

OS-9 I/O Modules

The following drivers now consider a signal value less than 32 (S$Deadly) to be deadly to I/O. Previously, only signal values 2 (S$Abort) and 3 (S$Intrpt) were considered deadly.

sc2661.a	sc68070.a	sc6821.a	sc6850.a	sc68560.a
sc68562.a	sc68681.a	sc68901.a	sc7201.a	sc8x30.a
Scp147.a	Scp68230.a	Scpio050.a	Scpio117.a	

The following are the changes to the individual drivers:

Nil.a — **Device Descriptor**

The port address of the NIL device was changed from 0 to 0xff000000. This avoids a potential conflict with the standard PIPE device descriptor.

Ram.a — **RAM Disk Driver**

The RAM disk driver now supports multi-sector I/O calls.

Sector zero initialization now updates sector 0 to the Version 2.4 standard definitions.

A problem with the initialization of the "sync-code" in sector 0 has been fixed. Previously, if the F$Time call to set the disk creation time failed, then the sync-code was not initialized.

RB319.a — **Disk Driver for the MVME319**

This is a new driver in the OEM release. This driver supports the following features:

- Variable sector sizes
- Data-cache flushing

RB320.a — **Disk Driver for the MVME320**

This is a new driver in the OEM release. This driver supports the following features:

- Variable sector sizes
- Data-cache flushing

rbfdesc.a — **RBF Descriptor Generator**

The default value for the PD_MaxCnt (maximum transfer byte count) is set by this file to 0xffff. This value may be over-ridden in the user's disk macro using the MaxCount label.

The RAM disk descriptor macros now set the multi-sector I/O flag and set PD_MaxCnt (maximum transfer count).

Definitions for high-density have been added. These definitions affect the fields PD_TYP (drive type), PD_DNS (media recording density), and the new field, PD_Rate (transfer/rotation rates). Refer to the application note on high-density floppy specifications for further information on these definitions.

sc2661.a **Serial Driver for SC2661**

A problem with operation at 50 baud has been fixed.

A problem with the assertion of the RTS line has been fixed.

sc68562.a **Serial Driver for the SC68562**

The determination of the A- or B-side device has been improved.

Output is now fully interrupt driven.

A problem with dynamic baud rate changing has been fixed.

A problem with the SS_Relea SetStat has been fixed.

sc68681.a **Serial Driver for the MC68681**

A problem with the determination of A- or B-side operation has been fixed.

sc68901.a **Serial Driver for the MC68901**

A problem with the terminate routine has been fixed. Previously, it was possible for the driver to "lock up" during terminate.

A problem with dynamic baud-rate changing has been fixed.

sc8x30.a Problems with the baud-rate constants for the MVME117 have been fixed.

TM3000 **Tape Driver**

F$Trans calls have been added to the driver to allow support of systems with memory addresses mapped differently between the CPU and the bus.

The driver now interrogates the system global flag D_SnoopD (data caches are snoopy or absent) to determine whether it has to perform an explicit cache flush call (F$CCtl) or not.

The driver incorrectly determined that an interrupt had occurred if the device was sharing a vector with another device. This problem has been fixed.

File Managers

PIPEMAN

PIPEMAN now considers any signal less than 32 (S$Deadly) to be a deadly I/O signal. Previously, only signals 2 (S$Abort) and 3 (S$Intrpt) were considered deadly.

SCF

A problem with Open has been fixed. Previously, if the Open failed it was possible for the echo device to remain attached to the system.

A problem with a path being lost (E$PthLost) has been fixed.

SBF

Open and Create functions now call the driver with an SS_Open SetStat. If the driver returns an Unknown Service Request (E$UnkSvc) error, it is ignored and SBF returns good status to the caller. If any other error is returned, then the Open/Create is aborted, and the error returned to the caller.

Close calls the driver with an SS_Close SetStat. Any error returned by the driver is ignored.

SetStat (SS_OPT) calls to the driver now properly ignore Unknown Service Request (E$UnkSvc) errors.

RBF

RBF now supports variable logical sector sizes.

RBF now supports caching, using the diskcache utility.

RBF had a problem with opening the @ file for non-super users. This has been fixed.

RBF had a problem in that it did not restore the access mode when doing an error return. This caused created files not to have excess allocation trimmed off.

RBF was not using the last segment entry in a 256-byte FD. This has been fixed.

RBF now honors the PD_MaxCnt field of the path descriptor, thus allowing drivers to transfer greater than 64K of data at a time.

RBF now considers signals less than 32 (S$Deadly) to be fatal to I/O operations. Previously, only signal values 2 (S$Abort) and 3 (S$Intrpt) were considered deadly.

RBF now validates the logical drive number (PD_DRV) of the path against the number of drive tables (V_NDRV) supported by the driver. Previously, the responsibility for this checking (to validate the drive table pointer (PD_DTB) in the path descriptor) was the responsibility of the driver. New drivers can now remove this checking code and faithfully rely on the drive table pointer being accurate.

GetStat (SS_OPT) is now passed on to the driver. If the driver returns an Unknown Service Request (E$UnkSvc) error, RBF ignores the error and returns a good status to the caller. If the driver returns any other error, then it is returned to the caller.

System Security Module (SSM)

The following changes have been made to the SSM module for the MC68851 (ssm851).

The use count associated with memory blocks was not always being handled correctly. This has now been fixed.

The Source Function Code control register (SFC) of the CPU was not being preserved when it was used by the SSM code. This has been fixed.

A call to F$GSPUMp did not return the execute flag settings of the memory block, only read/write access permission. As the MC68851 does not provide specific execute access protection, any block mapped to the user is executable. The SSM code has been changed to always return the execute access bit set to reflect this condition.

End of Chapter 2

NOTES

3

Application Notes

Introduction

The application notes contained in this chapter are designed to introduce new material and highlight topics covered in existing OS-9 manuals. Consequently, some of the information has been previously released.

The following application notes are discussed in this section:

- RBF Device Descriptor Changes
- RBF Disk Caching Support
- RBF Variable Sector Support
- Extended and Non-Contiguous Boot Files
- Soft Bus Errors Under OS-9

All of the application notes will be incorporated into existing OS-9 manuals.

RBF Device Descriptor Changes

Modern floppy disk drives support a multitude of physical disk formats. To aid device driver support for these formats, the Version 2.4 release of OS-9 has expanded the device descriptor definitions for floppy disk formats. This allows drivers to support various formats in a standardized manner.

The changes to the descriptor definitions describe the *physical* format parameters, such as disk size, data-transfers rates, etc. As these definitions apply to the physical media (the drive's capabilities), a correctly written RBF driver can adapt itself to any reasonable media format subset that is compatible with the drive's capabilities.

In additional to the definitions that involve floppy disk devices, a new flag has been added to indicate "removable" hard disk media so as to aid the support of these types of devices.

The changes to the RBF device descriptor definitions and their implications to driver's are as follows.

Device Type (PD_TYP)

This field describes the physical parameters of the drive. Previously, only 8" and 5 1/4" Floppy Disk types (as well as a single hard disk flag) were defined. The new definitions allow an expansion of the disk types to be supported, as well as providing a "flag" to a driver to indicate whether the device descriptor format as normally seen in the options section of the path descriptor complies with Version 2.4 or earlier standards.

The new definitions are:

If bit 7 = 0, the device is a Floppy Drive and bits 6 - 0 are interpreted as follows:

bit 0 obsolete (5 1/4" vs 8")

For backwards compatibility, this field should be set to 1 if bits 1-3 indicate 8", otherwise it should be cleared to 0.

bits 1 - 3 Physical size of drive:

0	"old-style" descriptor, size is derived from the value of bit 0.
1	8"
2	5 1/4"
3	3 1/2"
4 - 7	reserved

The format utility uses this field to display the correct media physical size. Drivers should use this field to determine whether they are running with a new or old style descriptor. If this field is zero, an old style descriptor is in use. If non-zero, a new style descriptor is in use.

bit	4	reserved
bit	5	0 = single density track 0, side 0 (FM) 1 = double density track 0, side 0 (MFM)
bit	6	reserved

If bit 7 = 1, the device is a Hard Drive and bits 6 - 0 are interpreted as follows:

bits 0 - 5	reserved
bit 6:	0 = media is fixed 1 = media is removable

Hard disk drivers that wish to support removable media can inspect this flag. If it is set, the driver can take any appropriate action pertaining to issues applicable to removable media. Typically a driver would use this flag to determine whether Sector 0 information is cached by the driver or not.

Drive Density (PD_DNS)

This field indicates the recording and track density capabilities of the device.

bit 0	0 = single bit density (FM) 1 = double bit density (MFM)
bits 1 - 3	track density

bit 1 = 1	double track density (96/135 tpi)
bit 2 = 1	quad track density (192 tpi)
bit 3 = 1	octal track density (384 tpi)

All bits clear signify 48 or 62.5 tpi.

Note that the actual density selected is determined by the media size in use (for example, 96tpi is double track for 5 1/4", 135 tpi is double track for 3 1/2"). The format utility uses these fields, in conjunction with the "size" bits of PD_TYP to display the correct tpi to the user. Drivers typically use this field only to determine the actual stepping algorithm (for example, single/double/etc. physical steps per logical step).

All other bits are reserved.

Data Transfer/Rotational Rates (PD_Rate)

This field defines the data-transfer rate and rotational speed of the drive.

bits 0 - 3 rotational speed

0 = 300 rpm
1 = 360 rpm
2 = 600 rpm
3 - 15 = reserved

bits 4 - 7 data transfer rate

0 = 125KB/sec
1 = 250KB/sec
2 = 300KB/sec
3 = 500KB/sec
4 = 1MB/sec
5 = 2MB/sec
6 = 5MB/sec
7 - 15 = reserved

Drivers may use these bit-fields to explicitly set the correct format for the drive/media combination. Previously, a driver that supported multi-speed drives (for example) had to make assumptions on these values, based upon the value of bit 0 of PD_TYP.

NOTE: This field is valid only if a "new style descriptor" is in use (which means the PD_TYP size field is non-zero).

Multi-Format Support Implications:

While these definitions give greater variety to floppy disk formats (for example, extra-high density for backups, support for other operating systems), you should not ignore the issue of system media interchange.

High-density on one system *may not* be physically compatible with another system's high-density format. Thus, driver writers should ensure that the pre-Version 2.4 definitions and media formats are supported by any drivers written or modified to support the new high-density definitions.

As a general rule, *all* OS-9 systems should be capable of reading and writing an "OS-9/68K universal" format disk. This will help ensure that inter-system and third party media interchange is possible.

A universal format disk is described by the following characteristics:

Disk Physical Size:	5 1/4" or 3 1/2"	(PD_TYP)
Number of Physical Cylinders:	80	(PD_TotCyl)
Number of Logical Cylinders:	79	(PD_CYL)
Number of Sides:	2	(PD_SID)
Track Density (TPI):	96/135	(PD_DNS)
Recording Format:	MFM	(PD_DNS)
Sectors Per Track:	16	(PD_SCT)
Sector Size (Logical & Physical):	256	(PD_SSize)
Sector Base Offset:	1	(PD_SOffs)
First Accessible Track:	1	(PD_TOffs)
Disk Rotational Rate (rpm):	300	(PD_Rate)
Data Transfer Rate (Kbits/sec):	250	(PD_Rate)

Summary of Example Formats:

Format Codes:

D	35/40 track, normal density
DD	80 track, normal density
AT	PC/AT-style, 80 track, high density
ED	80 track, extra high density
HD	80 track, high density, 8" image style
HR	80 track, normal density, high rotational speed

Physical Disk:

Fmt	Sides	Cyls	RPM	Data Xfr Rate	TPI	Phy Size	UnForm. Cap.
D	2	35/40	300	250K	48/62.5	5/3	500K
DD	2	80	300	250K	96/135	5/3	1M
AT	2	80	300	500K	96/135	5/3	2M
ED	2	80	300	1M	135	3	4M
HD	2	77/80	360	500K	96	5	1.6M
HR	2	80	360	300K	96	5	1M

Logical Disk:

Fmt	256-byte/sector Sect/Trk	Fmt Cap.	512-byte/sector Sect/Trk	Fmt Cap.	1024-byte/sector Sect/Trk	Fmt Cap.
D	16	327K	9	368K	5	409K
DD	16	655K	9	737K	5	819K
AT	32	1.31M	18	1.47M	10	1.64M
ED	61	2.49M	36	2.95M	20	3.27M
HD	26	1.02M/1.06M	15	1.18M/1.23M	8	1.26M/1.31M
HR	<see DD>					

Example Hardware Support:

Fmt	Drive Model/Mode
D	Any 35/40 track version of DD; DD drive in double-step mode.
DD	Teac 235-JS (1M mode); Teac 235-HF (1M mode); Teac 55GFR (300 rpm normal dens).
AT	Teac 235-JS (2M mode); Teac 235-HF (2M mode).
ED	Teac 235-JS (4M mode).
HD	Teac 55GFR (360 rpm high density).
HR	Teac 55GFR (300 rpm normal density, single-speed model).

Example Device Descriptor Fields:

Size/ Fmt	PD_TYP	PD_DNS	PD_Rate	
	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 - 4	3 - 0
5D	0 0 x 0 0 1 0 0	0 0 0 0 0 0 0 1	0	0
3D	0 0 x 0 0 1 1 0	0 0 0 0 0 0 0 1	0	0
5DD	0 0 x 0 0 1 0 0	0 0 0 0 0 0 1 1	0	0
3DD	0 0 x 0 0 1 1 0	0 0 0 0 0 0 1 1	0	0
5AT	0 0 1 0 0 1 0 0	0 0 0 0 0 0 1 1	2	0
3AT	0 0 1 0 0 1 1 0	0 0 0 0 0 0 1 1	2	0
3ED	0 0 1 0 0 1 1 0	0 0 0 0 0 0 1 1	3	0
5HD	0 0 x 0 0 0 1 1	0 0 0 0 0 0 1 1	2	1
3HR	0 0 1 0 0 1 1 0	0 0 0 0 0 0 1 1	3	0
8DD	0 0 x 0 0 0 1 1	0 0 0 0 0 0 0 1	2	1

RBF Disk Caching Support

One of the features of this new version of RBF is the provision of disk cache support. This application note describes how this feature can be invoked and any considerations involved.

Enabling Disk Caching

RBF's disk caching facility is invoked by the diskcache utility. Caching may be enabled for any type of RBF device, and more than one device may be cached at a time.

The total amount of system memory used for caching all enabled drives can be set by the utility's -t option. If not explicitly defined, the diskcache utility automatically selects a reasonable value, based upon the amount of free system memory.

Caching may be dynamically enabled or disabled on a per drive basis while the system is running, using the -e and -d options.

Statistical information regarding the hit/miss ratios, amount of memory allocated, etc. can be inspected on a drive by drive basis, using the -l option.

An example output of this information follows:

```
Current size =  1047552
  Size limit =  1048576

Device:  /h0:1:1
                Requests    Sectors       Hits       Zaps     >2 Xfr Hit Rate
     Reads:        47592      55436      21874        143        662  39.5%
    Writes:         7723       8065                  7342         68
 Dir Reads:        54048      54048      34526      18387<-Sctr Zero  63.9%
Dir Writes:            0          0
 Hit compares =      63399 (          1/hit)
Miss compares =      92685 (          3/miss)
```

Using RBF's caching can greatly increase disk I/O throughput, as commonly accessed structures (such as directory pathlists) can be accessed directly from system memory. This alleviates the need to access the disk media itself. This can also benefit the general system throughput, as it minimizes external bus usage (for example, SCSI bus, VME bus).

RBF's caching algorithms have the following general characteristics:

- RBF *never* caches the media identification sector (LSN 0). Therefore, removable media can have caching enabled.
- I/O transfers which involve greater than 2 sector transfers are not cached, thus large block transfers will not "sweep" the cache of valid entries. Generally, single-sector reads are made during directory and file descriptor searching, and thus these types of commonly accessed structures are cached.
- The cache operates on a write-through basis. It always ensures that data writes proceed to the media (without error) before caching the data (if applicable).

There are no special considerations for purging the cache prior to system shutdown. This also ensures that system crashes do not corrupt the media integrity.

Caching Caveats

When enabling disk caching, the following important points should be noted:

- Caching should only be invoked on devices that are known to the I/O system (that is, the devices should have been initialized with the iniz utility). The diskcache utility links the drive into the caching system by setting the V_Cache entry of the drive's drive table. If the device is terminated (deinitialized with the deiniz utility), the driver's static storage is deallocated and the V_Cache entry is lost.
- If caching is to be enabled on drives with different sector sizes, the device with the largest sector size should be included in the initial cache enabling. Attempting to add a drive (with a sector size larger than any currently cached drive) to the cache system after initial cache startup results in continuous "misses" for that drive, as the sector size is too large.

If a larger sector sized device is required to be added, then the entire RBF cache must be shutdown (using the -d option), and then re-enabled for all desired drives.

RBF Variable Sector Support

The new version of RBF also provides support for various sector sizes. RBF logical sectors may now have a size from 256-bytes to 32768-bytes in integral binary multiples (256, 512, 1024, etc.). This application note describes the issues that must be addressed when dealing with variable logical sector sizes.

RBF Device Drivers

Prior versions of RBF assumed a constant value of 256-bytes for the logical sector size. The new RBF now dynamically determines whether the driver it is calling can support other logical sector sizes, using the new GetStat functions code SS_VarSect.

When a path is opened to an RBF device, RBF now calls the driver with SS_VarSect, and depending upon the results of the call, take appropriate action:

① If the driver returns without error, RBF assumes that the driver can handle variable logical sector sizes. It then uses the PD_SSize field of the path descriptor to set the media path's logical sector size, so that RBF's internal buffers may be allocated, etc.

② If the driver returns an unknown service request error (E$UnkSvc), RBF assumes that it is running with a driver that presumes a logical sector size of 256-bytes. RBF allocates its buffers accordingly, and does not use the PD_SSize field of the path descriptor.

③ If the driver returns any other error, RBF aborts the path open operation, deallocate any resources and return the error to the caller.

Support for variable logical sector sizes is *optional* under the new RBF, as existing drivers operate in the same manner as they do under previous versions of RBF (i.e. case #2 above).

If variable sector size support is desired, use the following guidelines as an aid in the conversion of the driver.

In general, device drivers written for the old RBF were generally written to operate under one of two situations:

① The media logical and physical sector sizes were the same.

In this case, the driver would accept the sector count and starting LSN, convert it to the physical disk address (if required) and then perform the I/O transfer.

Drivers written under this operation mode can easily be converted to support other logical/physical sector sizes by adding support for the GetStat SS_VarSect call and ensuring that the driver does not have any "hard-wired 256-byte" assumptions.

Typically, this implies that the following should be checked in the driver:

- The driver should use the sector size field (PD_SSize) in the path descriptor whenever it needs to convert sector counts to byte counts (for example, loading DMA counters).

- The driver should maintain any disk buffers in a dynamic manner, so that a sector size change on the media does not cause a buffer overrun. This usually means that fixed sized buffers allocated in the static storage of the driver should now be allocated and returned as required, using the F$SRqMem and F$SRtMem system calls.

In many cases, a correctly written driver only needs the addition of the SS_VarSect handler to be ready for operation with variable sector sizes.

② The media logical and physical sector sizes were *not* the same.

In this case, the driver would translate the logical sector count and starting LSN passed by RBF into a physical count/address, convert those values to the physical disk address (if required), and then perform the I/O transfer.

These types of drivers are known as ***deblocking*** drivers, as they combine/split the physical sectors from the disk into the logical sectors RBF requires.

Drivers written under this mode of operation can be converted to variable logical sector operation, although they may require more work than non-deblocking drivers.

Apart from adding the code to handle the GetStat SS_VarSect call, you should remove the driver's deblocking code. You should also remove any hardwired assumptions about sector sizes and fixed buffers. In effect, the driver will be converted from a deblocking driver to a non-deblocking driver.

RBF Media Conversion

Once the driver has been updated to support the new RBF, user's who wish to convert their media (specifically hard disk drives) to non-256 byte logical sector sizes are required to re-format the media. Media that is set for a 256-byte logical sector size can be used immediately when the driver is ready.

If the media is to be re-formatted, then it may only require a logical re-format (that is, converting a deblocking 512-byte physical sector disk to 512-byte logical). In this case, you should perform the following steps:

① Perform a backup of the media to be converted.

② Re-Format the media. A physical format is only required if you need or desire to change the media's physical sector size. You can use the format utility's -np option if you do not wish a physical re-format.

③ Re-install the data saved in the first step.

Your conversion to a non-256 byte logical sector size should now be complete.

Benefits of Non-256 Byte Logical Sectors

Utilizing logical sector sizes can provide some benefits, depending upon your application requirements. Generally, the following benefits will be seen in most situations:

- The bitmap sector count will decrease. This may mean that the minimum cluster size of the media can be decreased on large hard disks.
- The number of clusters in a bitmap sector will increase, thereby allowing faster bitmap searches and potentially larger segments to be allocated in the file descriptor segment list.
- The media capacity may increase. Many disk drives (both floppy and hard disks) can store more data on the disk due to the decrease in the number of sectors per track (and thus less inter-sector gaps).
- The chances of "segment list full" errors will decrease, as the expansion of the sector size beyond 256 bytes allows more file segment entries in the file descriptor.

Bootstrap Drivers

Converting RBF drivers and media to non-256 byte logical sectors also implies that bootstrap code changes will be required if the media is to continue to provide system bootstrap support.

In general, the issues that the RBF driver has to deal with (hard-wired assumptions about 256 byte sectors, for example) are the same issues that the BootStrap driver must handle.

In addition to these issues, if the BootStrap driver is to support booting from any logical sector size, the following should be noted:

- The BootStrap driver must be able to read the identification sector (LSN 0) of the media. Depending upon the actual hardware situation and capabilities, this may require querying the drive for sector size (for example, Mode Sense command to SCSI drives), reading a fixed byte-count from the drive (for example, partial sector read), or attempting to read the sector using all possible sector sizes.
- Once LSN 0 has been successfully read, the BootStrap driver should inspect the DD_LSNSize field of sector zero. This field gives the media's logical sector size (if it is 0, then a size of 256 is assumed), and this value combined with the known physical size allow the BootStrap driver to load the actual bootstrap file. If the logical and physical sector sizes differ, the BootStrap driver can utilize deblocking algorithms or return an error.

RBF Disk Utilities

Utilities that need to ascertain the media's logical sector size (for example, the dcheck utility) can easily do so by opening a path to the device and then checking the PD_SctSiz field of the path options section (GetStat, SS_OPT function code).

RBF sets this field to the media's logical sector size when the path is opened. If the field contains a 0, then an old RBF is running in the system and the logical sector size can be assumed to be 256 bytes.

Extended and Non-Contiguous Boot Files

Previous to Version 2.4 OS-9/68000, disk based bootstrap files were restricted to the following constraints:

- The maximum file size was 65535 bytes (64K -1).
- The boot file had to be contained in contiguous sectors.

With the Version 2.4 Release, the capability to use boot files that are non-contiguous and/or larger than 64K in size is now supported.

This application note details the issues involved in using these new capabilities.

Bootstrap File Specifications

The original specification for RBF bootstrap files required that they be contiguous and less than 64K bytes in size. The OS9Gen utility accomplished the installation of the bootstrap file, by enforcing the contiguous data requirement and then updating the media's identification sector (LSN 0) with the bootstrap pointers.

The pointers used are:

Name	Description
DD_BSZ	Word value of boot file size.
DD_BT	3-byte value of the starting LSN of the bootstrap DATA.

Under the V2.4 OS9Gen, these original specifications have been expanded so that the identification sector pointers are defined in an "upwards compatible" manner, as follows:

Name	Description
DD_BSZ	If DD_BSZ is non-zero, this field contains the size of the bootstrap file, as per the original specification. If this is zero, the DD_BT field is a pointer to the File Descriptor (FD) of the bootstrap file. The FD contains the boot file size and the segment pointer(s) for the boot file data.
DD_BT	If DD_BSZ does not equal zero, this is the starting LSN of the bootstrap data, as per the original specification. If DD_BSZ equals zero, this is the LSN of the File Descriptor for the boot file.

Making Boot Files

Making bootstrap files is accomplished using the OS9Gen utility, as before. OS9Gen's default mode of operation is to create a contiguous, less than 64K boot file (that is, according to the original specification).

If large or non-contiguous boot files are desired, then OS9Gen can be informed to act accordingly with the -e option.

BootStrap Driver Support

If large, non-contiguous bootstrap files are required for the system, existing bootstrap drivers must be modified accordingly.

When reading a boot file, the main considerations for the bootstrap driver are as follows:

- Support should be maintained for contiguous, less than 64K boot files because this is the default mode for OS9Gen.
- Once the bootstrap driver has read the media's identification sector, it should inspect the bootstrap variables to decide whether a bootstrap file is present. If both the bootstrap fields are zero, then the media is non-bootable and an appropriate error should be returned. If the bootstrap file is present, the bootstrap driver should determine what type it is.
- If both bootstrap fields are non-zero, then the driver is dealing with a contiguous, less than 64K boot file. The driver typically allocates memory for the boot file (specified by DD_BSZ), locates the start of the bootstrap data (specified by DD_BT) and then reads the data.
- If the bootstrap size field (DD_BSZ) is zero and the data pointer (DD_BT) is non-zero, then DD_BT is pointing to the RBF file descriptor associated with the boot file. The driver should then read the file descriptor into memory, and inspect the file size (FD_SIZ) and segment entries (FD_SEG) to determine the boot files size and location(s) on the disk. The driver typically reads each segment until the entire boot file has been read into memory. When loading the boot file into memory, the driver must ensure that the data appears in a contiguous manner.

Reading the segment entries of the boot file data requires that the bootstrap loader have a reasonable knowledge of the way in which RBF allocates files. In particular, the last segment entry for the file may be rounded up to the cluster size of the media (RBF always allocates space on a cluster basis). The bootstrap driver can determine the media cluster size from the DD_BIT value in the identification sector. While RBF may allocate space on a cluster basis, the bootstrap loader should always read the exact boot file size (rounded up to the nearest sector).

Soft Bus Errors Under OS-9

Some instructions of the MC68000-family of processors are intended to be indivisible in their operation (examples include TAS and CAS). Systems that have on-board memory, off-board memory, and allow other bus masters to access the on-board memory can run into deadlock situations when the on-board CPU attempts to access the external bus while the external master is accessing the on-board memory. Often, the bus arbiter breaks the deadlock by returning a bus error to the CPU. This is not a ***hard*** bus error (like non-existent memory), it is a ***soft*** bus error. If the instruction is re-run, it will typically succeed, as the deadlock situation will have terminated.

The file SYSMODS/sysbuserr.a provides a mechanism to install a soft bus error handler across the bus error jump table entry to allow software determination of the cause of the bus error. The soft bus-error handler can determine whether to re-run the instruction or pass the bus error along to a previously installed handler (such as the MMU code).

To use this facility, create a file buserr.m that has two macros:

Name	Description
INSTBERR	Hardware enable for soft bus error. Setup hardware to detect soft bus errors.
BERR	Bus error handler. Detect whether bus error is soft or hard. If soft, re-run the faulted instruction. Otherwise, call the original handler.

The details of the entry to these macros is documented in SYSMODS/sysbuserr.a.

End of Chapter 3

NOTES

4

Documentation Changes

This section contains the list of known documentation changes. These will be included in the release of the new manuals. The following manuals are covered:

- *Using Professional OS-9*
- *OS-9 Operating System Technical Manual*

Using Professional OS-9

Page 2-5

The first paragraph should be as follows:

> If the drive name in the prompt is correct, type y for yes. If you type y at the prompt, there will be a pause while the disk is being formatted. format will then prompt for the name of the disk:

Page 3-7

Replace the four groups of utilities and shell commands with the following three groups:

Group 1: Basic Utilities

attr	backup	build	chd	chx	copy	date
del	deldir	dir	dsave	echo	edt	format
free	help	kill	list	makdir	merge	mfree
pd	pr	procs	rename	set	setime	shell
w	wait					

Group 2: Programmer Utilities

binex	cfp	cmp	code	compress	count	dump
ex	exbin	expand	frestore	fsave	grep	load
logout	make	printenv	profile	qsort	save	setenv
tape	tee	touch	tmode	tr	unsetenv	

Group 3: System Management Utilities

break	dcheck	deiniz	devs	diskcache	events	fixmod
ident	iniz	irqs	link	login	mdir	moded
os9gen	romsplit	setpr	sleep	tapegen	tsmon	unlink
xmode						

Page 4-7

Replace the second sentence in the second paragraph with the following:

> The current directory concept allows you to organize your files while keeping them separate from other users on the system. The word ***current*** is used because by using the chd built-in shell command you can move through the tree structure of the OS-9 file system to a different directory. This new directory would then become your current data directory. The chd built-in shell command is discussed later in this chapter.

Add the following two sentences to the end of the fifth paragraph:

> You can use the pd utility to find your current data directory. This utility is discussed later in this chapter.

Page 4-14

Replace the sixth paragraph with the following:

> When used as the first name in a path, these naming conventions can be used with relative pathlists. However, if you are planning to port your code to other operating systems, you must remember that most operating systems do not use this convention.
>
> The examples below relate to the file structure in Figure 4g. The examples assume your initial current data directory is **PROG**.

Page 4-19

Replace the box with the following:

> ☞ **Just a Reminder:** Users with the same group.user ID as the person who created the file are considered owners, unless the file was created by a super user. Files created by super users can only be used by super users unless the appropriate permissions are set.

Page 5-6

Add the following to the list of built-in shell commands:

profile	Reads input from a named file and then returns to the shell's original input source.

Page 5-9

After the list of modifiers, separators, and wildcards, add the following:

> ☞ **You must be extremely careful when using wildcards with utilities such as del and deldir.**

Page 5-11

Add the following terms to the list of terms following the first paragraph:

pipe	Pipe Device
/nil	Null Device

Page 5-23

Add the following information after the example .logout file:

The Profile Command

The profile built-in shell command can be used to cause the current shell to ready its input from the named file and then return to its original input source, which is usually the keyboard. To use the profile command, enter profile and the name of a file:

```
profile setmyenviron
```

The specified file (in this case, setmyenviron) may contain any utility or shell commands, including commands to set or unset environment variables or to change directories. These changes will remain in effect after the command has finished executing. This is in contrast to calling a normal procedure file by name only. If you call a normal procedure file without using the profile command, the changes would not affect the environment of the current directories of the calling shell.

Profile commands may be nested. That is, the file itself may contain a profile command for another file. When the latter profile command is completed, the first one will resume.

A particularly useful application for profile files is within a user's .login and .logout files. For example, if each user includes the following line in the .login file, then system-wide commands (common environments, news bulletins, etc.) can be included in the file /dd/SYS/login_sys. A similar technique can be used for .logout files.

Page 5-24

Replace the existing sample file with the following:

```
* system startup procedure file
echo Please Enter the Date and Time
setime
tsmon /t1 /t2 /t3&
tsmon /t71                  * this terminal has been misbehaving.
```

Remove the paragraph following this sample file.

Page 7-2

Add the following option to the fsave utility:

-j[=]<number> Specifies the minimum system memory request.

Page 7-6
Add the following option to the frestore utility:

-a	Forces access permission for overwriting an existing file. You must be the owner of the file or a super user to use this option.
-j[=]<int>	Sets the minimum system memory request.

Page 8-4
Add the following CPU types to the list of CPUs in M$CPUTyp (offset $52):

68040
683XX

Change the last line on the page to the following:

For example, level 1, version 2.4, edition 1 would be 1241.

Page 8-6
Replace the existing reserved field (offset $69) with the following:

$69	M$Compat2	Indicates the "absence/snoopiness" of the system caches.

Page 8-13
If your system supports boot files that are greater than 64K in length and/or non-contiguous, you can ignore the restrictions in the box.

Change the last line on the page to the following:

```
TapeGen /mt0 -bz=bootlist.tape
```

Page B-1
Add the following paragraph at the beginning of the appendix:

This appendix documents the debug version of the ROM debugger. A new ROM debugger, RomBug, exists and is documented in the ***OS-9 ROM Debugger's User's Manual***.

The OS-9 Utilities

Page 2
Add the tapegen utility to the third group of utilities:

3. **System Management Utilities:** These utility programs will be used primarily by system managers and advanced assembly language programmers. Beginning programmers will almost never have the need to use these commands:

break	dcheck	deiniz	devs	events	fixmod	ident
iniz	irqs	link	login	mdir	moded	os9gen
romsplit	setpr	sleep	tapegen	tsmon	unlink	xmode

Page 10
Add the following to the description of the break utility:

NOTE: If there is no debugger in ROM or if the debugger is disabled, break will reset the system.

CAVEAT: You must be aware of any open network paths when you use the break utility as all timesharing is stopped.

Page 19
Add the following options to the copy utility:

-f	Rewrites destination files with no write permission.
-z	Reads file names from standard input.
-z=<file>	Reads file names from a <file>.

Page 25
Replace the example for the dcheck utility with the following:

```
Volume - 'Ram Disk (Caution: Volatile)' on device /dd
$001000 total sectors on media, 256 bytes per sector
Sector $000001 is start of bitmap
$0200 bytes in allocation map, 1 sector(s) per cluster
Sector $000003 is start of root dir
Building allocation map...
$0003 sectors used for id sector and allocation map
Checking allocation map...

'Ram Disk (Caution: Volatile)' file structure is intact
5 directories, 60 files
580096 of 1048576 bytes (0.55 of 1.00 meg) used on media
```

Page 26
Add the following warning after the example in the deiniz utility:

WARNING: Do not deniz a device you did not explicitly iniz.

Page 27
Add the following option to the del utility:

-e	Erase disk space that the file occupied.

Page 32
Add the following option to the dir utility:

-z=<file>	Reads the directory names from <file>.

Page 35
Add the following option to the dsave utility:

-f	Uses copy's -f option to force the writing of files.

Page 38
Add the following options to the dump utility:

-m	Dumps from a memory resident module.
-s	Interprets the starting offset as a sector number. This is useful for RBF devices with a sector size not equal to 256.

Page 49
Add the following options to the format utility:

-e	Displays elapsed verify time.
-nf	Inhibits the fast verify mode.

Page 51
Replace the example for the free utility with the following:

```
"Tazz: /H0 Wren V" created on: Oct 6, 1989
Capacity: 2347860 sectors (256-byte sectors, 8-sector clusters)
1508424 free sectors, largest block 1380120 sectors
386156544 of 601052160 bytes (368.26 of 573.20 Mb) free on media (64%)
353310720 bytes (336.94 Mb) in largest free block
```

Page 53
Add the following option to the frestore utility:

-a	Forces access permission for overwriting an existing file. You must be the owner of the file or a super user to use this option.
-j[=]<int>	Sets the minimum system memory request.

Page 57
Add the following option to the fsave utility:

-j[=]<number>	Specifies the minimum system memory request.

Page 65
Add the following note after the examples for the iniz utility:

NOTE: Do not iniz non-sharable device modules as they become "busy" forever.

Page 66
Add the following note after the device field description in the function section of the irqs utility:

NOTE: If no device name is displayed, the entries relate to IRQ handlers that support "anonymous" devices (for example, the clock ticker, DMA devices associated with other peripherals).

Page 92
Replace the fourth paragraph with the following:

<offset> specifies the offset of the field from the beginning of the module. This is a hexadecimal value. **NOTE:** For device-specific fields (see <name> below), this offset is the offset of the field within the DevCon section of the descriptor (and not the module start).

Page 93
Add the following options to the moded utility:

-e=<path>	Use <path> for the error message file.
-d=<path>	Use <path> for the field descriptions (moded.fields).

Page 94
If you are completely upgrading your system to Version 2.4, you can remove the warning. If you are using ROMs that only support small, contiguous boot files, you should not remove the warning.

Add the following notes to the bottom of the page:

NOTE: Only super users (0.n) can use this utility.

NOTE: You can only use this utility on format-enabled devices.

Page 95

Remove the last two sentences from the first paragraph. Also, remove the third paragraph.

Add the following options to the os9gen utility:

-e	Extended boot. Allows you to use large (greater than 64K) and/or non-contiguous files.
-r	Remove the pointer to the boot file (does not delete file).

Remove the -s=<size> option.

Page 128

Add the following note after the first paragraph:

NOTE: The tmode utility can only be used for SCF/GFM devices.

Page 129

The tmode parameter nolf was incorrectly documented as olf.

Page 140

Add the following note after the first paragraph:

NOTE: The xmode utility can only be used for SCF/GFM devices.

OS-9 Operating System Technical Manual

Page 1-12
Replace the M$Exec (Execution Offset) description with the following:

> This is the offset to the program's starting address, relative to the starting address of the module or the Module Entry Table (FM, DATA, DRVR).

Page 2-16
Add the following note to the M$Extens (Customization module name offset):

> **NOTE:** Customization modules must be system module types.

Add the following CPU types to the M$CPUTyp description:

> 68040
> 683XX

Page 2-17
Replace the last sentence in the M$OS9Lvl (Level, Version, and Edition) description with the following:

> For example, level 2, version 2.4, edition 0 would be 2240.

Add the following at the bottom of the page:

M$Compat2		Indicates the "absence/snoopiness" of the system caches.
	0	0 = external instruction cache is *not* snoopy*
		1 = external instruction cache is snoopy or absent
	1	0 = external data cache is *not* snoopy
		1 = external data cache is snoopy or absent
	2	0 = on-chip instruction cache is *not* snoopy
		1 = on-chip instruction cache is snoopy or absent
	3	0 = on-chip data cache is *not* snoopy
		1 = on-chip data cache is snoopy or absent
	7	0 = kernel disables data caches when in I/O
		1 = kernel *does not* disable data caches when in I/O

* snoopy = cache that maintains its integrity without software intervention.

Page 2-19
Add the following offset location to the list:

Offset	Name	Usage
$69	M$Compat2	Indicates the "absence/snoopiness" of the system caches.

Page 2-30

Replace the second sentence in the first paragraph with the following:

> There must be at least 4K of RAM below and 4K of RAM above this address for system global storage.

Page 4-3

Replace the second signal listing (5-255) and description with the following:

Signal	*Description*
5-31	Deadly I/O
32-255	Reserved for future use by Microware.

Page 4-16

Replace the last sentence of the last paragraph with the following:

> If you want the module to remain in memory when the link count is zero, you can make the module "sticky" by setting the "sticky" bit in the module's attribute byte when you create the module.

Page 7-1

Replace the second paragraph with the following:

> RBF supports logical sector sizes in integral binary multiples from 256 to 32768 bytes. If a disk system is used that cannot directly support the logical sector size (for example, 256 byte logical sectors on a 512-byte physical sector disk), the driver module must divide or combine sectors as required to simulate the required logical sector size.

Page 7-2

Replace the description of Bit 1 and Bit 2 in the middle of Figure 7-1 with the following:

Bit 1:	0 = single density (FM)
	1 = double density (MFM)
Bit 2	1 = double track (96 TPI/135 TPI)
Bit 3	1 = quad track density (192 TPI)
Bit 4	1 = octal track density (384 TPI)

Add the following addresses at the bottom of Figure 7-1:

Addr	Size	Name	Description
$64	4	DD_MapLSN	Bootmap starting sector number
$68	2	DD_LSNSize	Media logical sector size (0 = 256)
$6A	2	DD_VersID	Sector 0 Version ID

Page 7-3
Replace the second sentence with the following:

> DD_MapLSN specifies the allocation map start address. DD_MapLSN is usually 1. If it is 0, then an address of 1 should be assumed.

Page 7-4
Replace the first sentence in the fourth paragraph with the following:

> The segment list (FD_SEG) consists of a series of 5-byte entries, continuing until the end of the logical sector. For 256-byte sectors, this results in 48 entries.

Page 7-6
Replace the third sentence in the third paragraph with the following:

> To read a specific sector, perform a seek to the address computed by multiplying the LSN by the logical sector size of the media. The logical sector size can be found in the PD_SctSiz field of the path descriptor (if 0, a value of 256-bytes should be assumed.). For example, on 1024-byte logical sector media, to read sector 3 a seek is performed to address 3072 (1024*3), followed by a read system call requesting 1024 bytes.

Example Code

Page A-1
Replace the example Init Module code with the following:

```
Microware OS-9/68020 Resident Macro Assembler V2.9  90/09/10  19:55  Page     1
Init: OS-9 Configuration Module -
00001                        nam      Init: OS-9 Configuration Module
00048 *
00049  00000016 Edition      equ      22              current edition number
00050
00051  00000c00 Typ_Lang     set      (Systm<<8)+0
00052  00008000 Attr_Rev     set      (ReEnt<<8)+0
00053                        psect    init,Typ_Lang,Attr_Rev,Edition,0,0
00054
00055 * Configuration constants (default; changable in "systype.d" file)
00056 *
00057 * Constants that use VALUES (e.g. CPUTyp set 68020) may appear anywhere
00058 * in the "systype.d" file.
00059 * Constants that use LABELS (e.g. Compat set ZapMem) MUST appear OUTSIDE
00060 * the CONFIG macro and must be conditionalized such that they are
00061 * only invoked when this file (init.a) is being assembled.
00062 * If they are placed inside the CONFIG macro, then the over-ride will not
00063 * take effect.
00064 * If they are placed outside the macro and not conditionalized then
00065 * "illegal external reference" errors will result when making other files.
00066 * The label _INITMOD provides the mechanism to ensure that the desired
00067 * operations will result.
00068 *
```

```
00069 * example systype.d setup:
00070 *
00071 * CONFIG macro
00072 *   <body of macro>
00073 *  endm
00074 *  Slice set 10
00075 *  ifdef _INITMOD
00076 * Compat set ZapMem patternize memory
00077 *  endc
00078 *
00079
00080 * flag reading init module (so that local labels can be over-ridden)
00081  00000001 _INITMOD   equ    1              flag reading init module
00082
00083 000109a0 CPUTyp     set    68000        cpu type (68008/68000/68010/etc.)
00084 00000001 Level      set    1            OS-9 Level One
00085 00000002 Vers       set    2            Version 2.4
00086 00000004 Revis      set    4
00087 00000001 Edit       set    1            Edition
00088 00000000 IP_ID      set    0            interprocessor identification code
00089 00000000 Site       set    0            installation site code
00090 00000080 MDirSz     set    128          initial mod directory size (unused)
00091 00000020 PollSz     set    32           IRQ polling table size (fixed)
00092 00000020 DevCnt     set    32           device table size (fixed)
00093 00000040 Procs      set    64           initial process table size
00094 00000040 Paths      set    64           initial path table size
00095 00000002 Slice      set    2            ticks per time slice
00096 00000080 SysPri     set    128          initial system priority
00097 00000000 MinPty     set    0            initial sys min executable priority
00098 00000000 MaxAge     set    0            initial sys max natural age limit
00099 00000000 MaxMem     set    0            top of RAM (unused)
00100 00000000 Events     set    0            initial event table size (div by 8)
00101 00000000 Compat     set    0            version smoothing byte
00102 00000400 StackSz    set    1024         IRQ Stack Size in bytes (must be 1k
                                               <= StackSz < 256k)
00103 00000000 ColdRetrys set    0            number of retries for coldstart's
                                               "chd" before failing
00104
00105 * Compat flag bit definitions
00106 00000001 SlowIRQ    equ    1            save all regs during IRQ processing
00107 00000002 NoStop     equ    1<<1         don't use 'stop' instruction
00108 00000004 NoGhost    equ    1<<2         don't retain Sticky memory modules
00109 00000008 NoBurst    equ    1<<3         don't enable 68030 cache burst mode
00110 00000010 ZapMem     equ    1<<4         wipe out mem that is allocated/freed
00111 00000020 NoClock    equ    1<<5         don't start sys clock during coldstart
00112
00113 * Compat2 flag bit definitions
00114 00000001 ExtC_I     equ    1<<0         ext instruction cache is coherent
00115 00000002 ExtC_D     equ    1<<1         external data cache is coherent
00116 00000004 OnC_I      equ    1<<2         on-chip inst cache is coherent
00117 00000008 OnC_D      equ    1<<3         on-chip data cache is coherent
00118 00000080 DDIO       equ    1<<7         don't disable data caching when in I/O
00119
00120                     use     defsfile (any above defs may be overridden in
```

```
                                        defsfile)
00001
00002                     use       ../DEFS/oskdefs.d

00001                     opt       -l
00003                     use       ./systype.d
00001 *
00002 * System Definitions for MVME147 System
00003 *
00004 * VERSION FOR DELTA
00005                      opt       -l
00004
00005
00121
00132
00133 * Configuration module body
00134 0000 0000          dc.l   MaxMem        (unused)
00135 0004 0020          dc.w   PollSz        IRQ polling table
00136 0006 0020          dc.w   DevCnt        device table size
00137 0008 0040          dc.w   Procs         initial process table size
00138 000a 0040          dc.w   Paths         initial path table size
00139 000c 0076          dc.w   SysParam      param string for first executable mod
00140 000e 0070          dc.w   SysStart      first executable module name offset
00141 0010 008b          dc.w   SysDev        system default device name offset
00142 0012 008f          dc.w   ConsolNm      standard I/O pathlist name offset
00143 0014 009b          dc.w   Extens        Customization module name offset
00144 0016 0095          dc.w   ClockNm       clock module name offset
00145 0018 0014          dc.w   Slice         number of ticks per time slice
00146 001a 0000          dc.w   IP_ID         interprocessor identification
00147 001c 0000          dc.l   Site          installation site code
00148 0020 0062          dc.w   MainFram      installation name offset
00149 0022 0001          dc.l   CPUTyp        specific 68000 family proc in use
00150 0026 0102          dc.b   Level,Vers,Revis,Edit OS-9 Level
00151 002a 0054          dc.w   OS9Rev        OS-9 revision string offset
00152 002c 0080          dc.w   SysPri        initial system priority
00153 002e 0000          dc.w   MinPty        initial sys min executable priority
00154 0030 0000          dc.w   MaxAge        maximum system natural age limit
00155 0032 0000          dc.l   MDirSz        module directory size (unused)
00156 0036 0000          dc.w   Events        initial event table size (no.r of
                                              entries)
00157 0038   10          dc.b   Compat        version change smooth byte
00158 0039   83          dc.b   Compat2       version change smooth byte #2
00159 003a 00b6          dc.w   MemList       memory definitions
00160 003c 0400          dc.w   StackSz/4     IRQ stack size (in longwords)
00161 003e 0000          dc.w   ColdRetrys    coldstart's "chd" retry count
00162 0040 0000          dc.w   0,0,0,0,0     reserved
00163 004a 0000          dc.w   0,0,0,0,0     reserved
00164
00165 * Configuration name strings
00166 0054 4f53 OS9Rev     dc.b     "OS-9/68K V",Vers+'0',".",Revis+'0',0
00167
00168 * The remaining names are defined in the "systype.d" macro
00169                      CONFIG
00170 0062 4465+MainFram   dc.b     "Delta MVME147",0
```

```
00172 0070 7368+SysStart    dc.b     "shell",0       name of initial module to execute
00173 0076-7461+SysParam    dc.b     "tapestart; ex sysgo",C$CR,0
Init: OS-9 Configuration Module -
00174 008b 2f64+SysDev      dc.b     "/dd",0         initial system disk pathlist
00184 008f 2f74+ConsolNm    dc.b     "/term",0       console terminal pathlist
00185 0095 746b+ClockNm     dc.b     "tk147",0       clock module name
00186 009b 4f53+Extens      dc.b     "OS9P2 ssm syscache" include mmu, caching.
00187 00ad 2073+            dc.b     " statclr"      special "clear down BE statI"
00188 00b5   00+            dc.b     0
00189  000000b6+            align
00190          +MemList
00191                       MemType
SYSRAM,250,B_USER,ProbeSize,CPUBeg,BootMemEnd,OnBoard,CPUBeg+TRANS
00192 00b6-0000+            dc.w     SYSRAM,250,B_USER,ProbeSize>>4 type, priority,
                                     access, search block size
00193 00be 0000+            dc.l     CPUBeg,BootMemEnd low, high limits (where it
                                     appears on local address bus)
00194 00c6 00fa+            dc.w     OnBoard,0       offset to description string
                                                     (zero if none), reserved
00195 00ca 0000+            dc.l     CPUBeg+TRANS,0,0 address translation adjustment
                                                       (for DMA, etc.), reserved
00199 MemType SYSRAM,240,B_USER+B_PARITY,ProbeSize,BootMemEnd,UserMemEnd,OffBoard,0
00200 00d6-0000+            dc.w     SYSRAM,240,B_USER+B_PARITY,ProbeSize>>4 type,
                                     priority, access, search block size
00201 00de 0040+            dc.l     BootMemEnd,UserMemEnd low, high limits (where it
                                     appears on local address bus)
00202 00e6 0107+            dc.w     OffBoard,0      offset to description string
                                                     (zero if none), reserved
00203 00ea 0000+            dc.l     0,0,0           address translation adjustment
                                                     (for DMA, etc.), reserved
00207 00f6 0000+            dc.l     0               terminate list
00208 00fa 6f6e+OnBoard     dc.b     "on-board ram",0
00209 0107 766d+OffBoard    dc.b     "vme bus ram",0
00210
00214
00218
00219 * define default caching modes (CPUTyp and system specific)
00220 * NOTE:  the following rules should be applied in determining
00221 *        the "coherency" of a cache and setting up the Compat2
00222 *        cache function flags:
00223 *
00224 *         - if the cache does not exits, then it is always coherent.
00225 *         - the on-chip cache coherency is not changable, except
00226 *           for the 68040.  If a 68040 system is used with
00227 *           bus-snooping disabled, then that fact should be registered
00228 *           by the user defining the label NoSnoop040 in their local
00229 *           "systype.d" file.
00230 *         - the coherency of external caches is indicated by the
00231 *           SnoopExt definition.  If the external caches are
00232 *           coherent or non-existant, then the label SnoopExt
00233 *           should be defined in "systype.d".
00234 *         - the kernel will disable data caching when calling a file
00235 *           manager, unless the "NoDataDis" label is defined.
00236 *           Disabling data caching is required for systems that have
```

```
00237 *           drivers that use dma and don't perform any explicit data
00238 *           cache flushing.  If your system does NOT use dma drivers,
00239 *           or the drivers care for the cache, then the NoDataDis
00240 *           label should be defined in "systype.d".
00241 *
00243
00246 * external caches are coherent or absent
00247  00000003 ExtCache   equ      ExtC_I|ExtC_D
00252
00261  00000003 Compat2    set      ExtCache       68030 on-chip caches are NOT snoopy
00270
00271 * add "don't disable data cache when in I/O" to Compat2
00273  00000083 Compat2    set      Compat2|DDIO
00275
00277
00278  00000114            ends
Errors: 00000
Memory used: 45k
Elapsed time: 6 second(s)
```

Error Codes

Page 10
Add the following new error codes:

Error Number		Description
000:001		Process has aborted.
000:175	E$Hardware	Hardware damage has been detected. E$Hardware usually occurs when the driver fails to detect the correct responses from the hardware. This can occur due to hardware failure or an incorrect hardware configuration.
000:176	E$SectSize	Invalid Sector Size The sector size of an RBF device must be a binary multiple of 256 (256, 512, 1024, etc.). The maximum sector size is 32768.

OS-9 System Call Descriptions

Page 1
Replace the last two sentences in the second paragraph with the following:

> The mnemonic names are defined in the relocatable library file usr.l or sys.l. These files should be linked with your programs.

Page 1-12
Remove the last line in the ***INPUT*** section:

> (a2) = process descriptor in which to put module

Page 1-60
Replace the third paragraph with the following:

> On 68020, 68030, and 68040 systems with a 68881 or 68882 floating point coprocessor, the following exception errors may also be caught:

Page 2-11
Replace the third line in the ***INPUT*** section of SS_FD with the following:

```
d2.w - Number of bytes to copy (<-logical sector size of media)
```

Page 2-13
Add the following system call:

SS_VarSect — Query Support for Variable Logical Sector Sizes (RBF)

INPUT:
```
d0.w = Path number
d1.w = #SS_VarSect function code
```

OUTPUT: none

FUNCTION: This is an internal call between RBF and a driver. If the driver returns no error, then PD_SSize specifies the logical sector size of the media. If the driver returns an error and the error is E$UnkSvc, then RBF sets the path's logical sector size to 256-bytes and ignores PD_SSize. If any other error is returned, the path open will be aborted and the error returned to the caller.

Page 2-21
Replace SS_Close with the following:

SS_Close — Notifies driver that a path has been closed (SCF, RBF, SBF)

INPUT:
```
d0.w = Path number
d1.w = #SS_Close function code
```

OUTPUT: none

FUNCTION: This is an internal call between the file manager and the driver. It notifies the driver that the path is being closed.

Page 2-22
Replace SS_DsRTS with the following:

SS_DsRTS — Disables RTS line (SCF)

INPUT:
```
d0.w = Path number
d1.w = #SS_DsRTS function code
```

OUTPUT: none

FUNCTION: Tells the driver to negate the RTS hardware handshake line.

Replace SS_EnRTS with the following:

SS_EnRTS — Disables RTS line (SCF)

INPUT: d0.w = Path number
d1.w = #SS_EnRTS function code

OUTPUT: none

FUNCTION: Tells the driver to enable the hardware handshake line.

Page 2-25
Remove the footnote at the bottom of the page.

Page 2-28
Remove the footnote associated with SS_WTrk.

Replace the fifth line in the ***INPUT*** section of SS_WTrk with the following:

```
contains 1 logical sector of data (pattern $E5).
```

Replace SS_WTrk's function description with the following:

> This causes a format track operation (used with most floppy disks) to occur. For hard or floppy disks with a "format entire disk" command, this formats the entire media only when side 0 of the first accessible track is specified.

End of Chapter 4

NOTES

5

Known Problems

Introduction

This list documents known problems with the Version 2.4 Release. These items will be addressed in a future release of OS-9.

Utilities

Generally, all utilities with the -z option will fail with a 215 error if they are in the following form:

```
<file><TAB>*<comments>
```

-z lines in the form <file><SPACE>*<comments> will work.

In general, the comments in files used with the -z option should be placed on their own individual line, with the asterisk (*) being the first character of the line.

The following are known problems for individual utilities:

cmp

Compare fails to recognize that files are *not* the same length when one of the files is 0 bytes. For example, if the size of <file1> is equal to 0 and the size of <file2> is not equal to zero, cmp will return without reporting that the files are of different sizes.

dsave

dsave does not support "extended" boots. Currently, dsave forks os9gen <dev> <file> if the -o option(s) are specified. This will fail if the boot file is larger than 64K.

fsave

When using fsave to backup RBF media, fsave may occasionally determine the media size incorrectly.

make

make experiences a problem with comments in a continuation line. If a continuation line has a comment (#) at the end of the second line, then make seems to merge the *next* line on to the original line.

os9gen

If you use the -e option for a file that is larger than 64K, you will get a message that the buffer has overflowed. You must specify that the buffer be larger than or equal to the size of the boot file.

sysdbg

Sysdbg's ov? command does not display the final vector (#255, address $3FC). Also, there is no check to prevent the user from attempting to monitor vector number 32 (TRAP #0). Accidentally attempting to do so totally locks up the OS-9 system.

ROM Debuggers

In general, the introduction of the CBOOT technology can cause jumptable references to be made if the old debugger is used because there is no init data routine for this debugger. Therefore, the old debugger will not work with CBOOT code. If you do not want to use RomBug, you must stick with your existing assembler booting code.

RomBug

When the disassembler shows ROM code that uses offsets from a6, it adds $8000 to the output.

When RomBug encounters an exception while running on behalf of itself, it fails to report the correct exception value. For example, if <addr> (in the following example) is non-existent memory, RomBug will print "reset vector" instead of "bus error:"

```
d <addr>
```

Assembler/Linker

Assembler (r68/r68020)

Operations using sr are not word length. The assembler will allow instructions of the form:

```
'andi.l #$0dff,sr'.
```

The assembled code generates three words:

```
027c
0000
0dff
```

The code is actually executed as 'andi.w #$0000,sr' with the $0dff used as the next instruction opcode. Operations using sr should be forced to word length.

The opt -q statement does not suppress warnings when used from within assembly language code. When used from the command line, it works fine. The following code illustrates this:

```
    psect test,0,0,0,0,0
func:
      opt q
      clr.b $fffe4000
      opt -q
      clr.b $fffe4000
      ends
```

If compiled with just r68 or r68020, two warnings are output. No warnings are received with r68 -q or r68020 -q.

Assembler (r68)

r68 aborts if the input file is not terminated with a \n.

Linker (l68)

The linker option -r is supposed to accept an *optional* parameter to specify a starting address. If the parameter is missing, then the default start address is 0. If a parameter is not specified when the -r option is used, the linker will not recognize -r as an option.

When linking large (greater than 256K) modules, it is possible for the linker to use up its 32 memory block limit if there are a number of other tasks running at the same time and fragmenting the memory. This is only a problem in certain extreme cases.

Miscellaneous Problems

cboot boot33c93

The VME620 does not use TransFact. Thus, systems with colored memory that have local/bus memory at different base addresses will fail on the 620's DMA transfer.

sb350 Tape Driver & Descriptor

The driver still has a long-standing problem with multi-volume backups.

kernel

If the system disables the disk caches when calling down into the I/O system then if an exception occurs in the driver it is possible for the disk cache to be left disabled.

Sbviper Driver

Inserting/removing the cartridge causes an E$DIDC error, unless you say "offline" before removing the cartridge. This problem only shows itself if the device is initialized with iniz. If you allow the iniz to take place automatically (such as during I$Open), then no problem is evident.

ssm851

The ssm851 bus error handler can lock up in an infinite loop, when a TAS is made to non-existing memory. The work around is to "probe" for hardware using a non-RMW type instruction such as move.b.

diskboot.c (CBOOT code)

There is a problem in the generic diskboot routine which can cause boot files towards the end of a disk to not be found.

Suggested Work-Around: In the file diskboot.c, function tryboot(), locate the following line:

```
pathopts.pd_cyl = ((*((u_char *) &sect0opts->pd_cyl)) << 8) +
                  *((u_char *) &sect0opts->pd_cyl + 1);
```

Add the following line after the line shown above:

```
pathopts.pd_totcyls = pathopts.pd_cyl + *((u_char *) &sect0opts->pd_toffs + 1);
```

rbf.h

Bit 4 of PD_TYP is documented as part of the "disk physical size field." PD_TYP is actually reserved.

End of Chapter 5