Home

OS-9® for 68K PC File Manager

Version 3.3

RadiSys
THE POWER OF WE

www.radisys.com
Revision A • July 2006

Copyright and publication information

This manual reflects version 3.3 of Microware OS-9 for 68K. Reproduction of this document, in part or whole, by any means, electrical, mechanical, magnetic, optical, chemical, manual, or otherwise is prohibited, without written permission from RadiSys Microware Communications Software Division, Inc.

Disclaimer

The information contained herein is believed to be accurate as of the date of publication. However, RadiSys Corporation 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. RadiSys Corporation 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 RadiSys Corporation and/or other parties. Unauthorized distribution of software may cause damages far in excess of the value of the copies involved.

July 2006

Copyright ©2006 by RadiSys Corporation
All rights reserved.

EPC and RadiSys are registered trademarks of RadiSys Corporation. ASM, Brahma, DAI, DAQ, MultiPro, SAIB, Spirit, and ValuePro are trademarks of RadiSys Corporation.

DAVID, MAUI, OS-9, OS-9000, and SoftStax are registered trademarks of RadiSys Corporation. FasTrak, Hawk, and UpLink are trademarks of RadiSys Corporation.

† All other trademarks, registered trademarks, service marks, and trade names are the property of their respective owners.

Table of Contents

Chapter 1: The OS-9 PC File Manager (PCF)

This chapter includes the following topics:

- **PCF Features**
- **Installing PCF**
- **Using PCF**
- **PCF Disk Driver Requirements**
- **PCF/OS-9 Compatibility**
- **The partdgen Utility**

PCF Features

The OS-9 for 68K PC File Manager (PCF) allows you to transfer files between PC-DOS and OS-9 systems. An OS-9 machine with PCF installed can read and write files from/to a PC-DOS formatted disk. The disk remains in PC-DOS format for subsequent read and write operations on the PC.

PCF is a file manager for disks formatted for the IBM PC/XT/AT/PS2 and their various clones and imitators. PCF does not attempt to be a complete file system, but emulates the Random Block File manager's (RBF) major functions at the program and device driver interfaces. Consequently, most OS-9 programs and drivers can use either RBF or PCF file managers without change.

File Allocation Tables

PCF supports 12- and 16-bit allocation tables and 512- and 1024-byte sectors. Support for 16-bit File Allocation Tables (FATs for disks larger than 32M) allows for a wide range of DOS formats.

Partitioned Disks

PCF supports partitioned disks via the `partdgen` utility (described later in this chapter), which allows you to create descriptors for disk partitions and display information about partitioned or non-partitioned disks.

Write-behind Caching

As needed, PCF uses write-behind caching to speed small writes as are typical with `I$WritLn`. This improves the performance of programs using `I$WritLn` or `I$Write` for small amounts of data.

Note
With write-behind caching, written data may not actually get to the disk until the file is closed. Also, write errors from the driver are sometimes reported at a subsequent read, write, or seek.

Multi-sector I/O

PCF supports multi-sector I/O for those drivers and descriptors offering it.

Directory and File Handling

PCF enlarges directories as needed. When a directory (other than the root) overflows, PCF doubles its size. When you use `makdir` to create a directory, PCF looks at `PD_SAS` (segment allocation size in the RBF device descriptor initialization table) to determine the number of segments to allocate.

File locking support ensures an open file cannot be deleted.

`I$ReadLn` and `I$WritLn` can handle both DOS and OS-9 format text files. Carriage returns and line feeds are converted appropriately.

Process Management

PCF has a number of preemption points providing control to a higher priority active process.

Installing PCF

Note

Before you install PCF, review the material in this manual, particularly the sections about PCF disk driver requirements and PCF/OS-9 compatibility.

Distribution Media

If you received the PC DOS file manager as part of an OS-9 developers kit, embedded systems, board level solution, or board support pack; the files are already in the MWOS directory structure and no installation is necessary.

If PCF was purchased separately, follow the installation process as described later in this chapter.

The components of the PCF package are contained in the following directories:

MWOS/OS9/SRC/IO/PCF/DESC

Source file for descriptors

- `pcd0.a`
- `pcd1.a`
- `pcd2.a`
- `pcd3.a`
- `pcfdesc.a`

MWOS/OS9/68000/CMDS

`partdgen`	PCF partition table utility
`pcformat`	Format utility for PC disks

MWOS/OS9/68000/CMDS/BOOTOBJS

`pcf`

MWOS/OS9/<CPU Family>/PORTS/<Card>

`pcf_descriptors.date`	Date for PC descriptors
`pcf_descriptors.make`	Makefile for PC descriptors

MWOS/OS9/<CPU Family>/PORTS/<Card>/CMDS/BOOTOBJS

`pcd0`	Descriptors specific to <Card>

Installation Steps

Complete the following steps to install PCF:

Step 1: To install PCF from disk:

Step 1. Log in to the OS-9 system as super user (group 0).

Step 2. Insert disk number 1 into the disk drive.

Step 3. Type the following at the prompt (if your floppy drive is not named `d0`, use the drive name appropriate to your system):

```
chd /d0; install
```

The installation program prompts you for any needed information.

Step 1: To install PCF from tape:

Step 1. Log in to the OS-9 system as super user (group 0).

Step 2. `iniz` your tape drive, if you have not already, by typing the following:

```
iniz /mt0
```

Step 3. Insert tape number 1 into the tape drive.

Step 4. Type the following at the prompt (if your tape drive is not named `/mt0`, use the drive name appropriate to your system):

```
copy /mt0 install; load -d install;install
```

The installation program prompts you for any needed information.

Step 2: To make PCF descriptors:

Descriptors for PCF disks are now made in the same manner as descriptors for the standard RBF devices.

Step 1. Change to the directory for your CPU or disk controller in the appropriate ports directory.

Step 2. Edit the `DiskPCDx` macro in the `systype.d` file to reflect your requirements.

Step 3. Execute the `make` utility specifying the `pcf_descriptors.make` makefile. The descriptor is placed in the `CMDS/BOOTOBJS` subdirectory of your current directory.

Step 4. Do one of the following:

- Move PCF and the desired device descriptors to the `CMDS/BOOTOBJS` directory on the root of your system disk so they may be easily loaded after the system has booted.
- Add PCF and the desired device descriptors to your system and bring them in with the boot.

You have installed PCF.

For More Information

Refer to **Chapter 2: PCF/RBF Device Drivers** for detailed instructions about converting RBF device descriptors for use with PCF.

Using PCF

Note

Before you use PCF, review the material in this manual, particularly the sections about PCF disk driver requirements and PCF/OS-9 compatibility.

Load the following modules into memory:

- PCF
- RBF driver
- PCF descriptor modified from an RBF descriptor

To load PCF into memory, go to its directory and enter the following command:

```
load -d pcf
```

To find the driver for PCF, use the `dump` utility on the RBF descriptor or the `moded` utility to view the driver field.

PCF Disk Driver Requirements

The disk driver you intend to use with PCF must meet certain requirements. To enable PCF to read and write PC-DOS disks correctly, the device driver must use the `PD_SSize` field (physical sector size) in the path descriptor.

`PD_SSize` typically affects the following driver functions:

- Sector buffering
- Transfer counts
- Controller drive parameters

Sector Buffering

Drivers must dynamically allocate and maintain sector buffers, as well as track the size of the buffers currently in use.

Transfer Counts

PCF passes read/write counts as a **block count** of 512 byte sectors. When drivers convert this to byte counts (for example, when loading direct memory access (DMA) counters), they must use `PD_SSize`.

Controller Drive Parameters

Drivers that communicate with intelligent disk controllers supporting multiple floppy formats (for example, SCSI controllers) must detect when the disk format changes (for example, from 256 to 512 byte sectors). The drivers also must re-initialize the controller’s floppy format when the format changes.

Most drivers written for OS-9 make some assumptions as to the contents of sector 0. This allows the maximum flexibility in dealing with floppy disks that may have several different formats under OS-9. Because the PC-DOS sector 0 is not at all like the OS-9 sector 0, make sure information normally derived from the sector 0 drive table is taken from the path descriptor when a disk is identified as having a PC-DOS format.

Some drivers may support variable sector size by ignoring sector size for reads and writes. These drivers might work with PCF. The device descriptor's disk verify flag (`PD_VFY`) should be off for PC-DOS disks. This is not a PCF requirement, but an issue for the device driver in use. You can use the verify flag with drivers fully supporting variable sector size. If you are unsure how a driver operates with verify on, turn verify off.

Set the device descriptor's format inhibit bit (`PD_Cntl` bit 0) to 1 to protect from inadvertently using the `format` utility and possibly corrupting your disk. The OS-9 `format` utility does not know how to create a PC-DOS disk and can not proceed if the format inhibit bit is set.

For More Information

Refer to **Chapter 2: PCF/RBF Device Drivers** for detailed instructions about converting RBF device descriptors for use with PCF.

PCF/OS-9 Compatibility

Incompatibilities between PCF and OS-9 are discussed below.

File Names

Incompatibility between PC and OS-9 file names is a frequent problem. The PC file name may have as many as twelve characters, including an eight character name and a three character extension. OS-9 allows for as many as 28 characters in a file name. Attempts to exceed the number of allowable characters in the file name on a PC disk result in `error #215` (bad pathlist).

DOS file names can contain characters that are not legal in RBF files. Before transporting an OS-9 disk to PC-DOS, rename (or copy with a new name) any files that do not conform to the appropriate file name conventions.

Standard OS-9 Utilities

The only commands that **do not** work with PCF are those needing information about logical sector numbers on the disk. A disk formatted for use with a PC has a very different logical format than a disk formatted for use with OS-9.

Utilities such as `free`, `dcheck`, and `bfed` fail when using PCF because PCF does not emulate RBF's sector 0 or its allocation table. Similarly, utilities such as `pd` and `deldir` fail because PCF does not perfectly emulate the file descriptor sectors for directories.

Note

`bfed` reads the size of the disk from sector 0.

WARNING

The following standard utilities **do not** work with PCF. Do not attempt to use any of these utilities because they may cause information loss on the PC disks.

- `backup`
- `dcheck`
- `deldir`
- `format`
- `free`
- `frestore`
- `fsave`
- `os9gen`

The following standard utilities work to some degree; however, there are some limitations due to the nature of the file structure on PC disks.

Table 1-1 Standard Utilities With Limitations

Utility	Limitations
`pd`	Only works from the root level of the PC-DOS disk.

Table 1-1 Standard Utilities With Limitations (continued)

Utility	Limitations
dsave	When you dsave from a PC-DOS disk, it only works from the root level of the PC-DOS disk. **NOTE:** You can dsave to the PC-DOS disk; it works correctly.
attr	Only affects the directory bit. Use attr to remove a directory (see **Removing Directories from a PC Disk**).

These standard utilities operate normally:

Table 1-2 Standard Utilities Without Limitations

binex	build	cfp	chd
chx	cmp	compress	copy
count	del	dir	dump
echo	edt	exbin	expand
fixmod	grep	iniz	ident
list	load	makdir	make
merge	qsort	rename	save
touch			

DIR Command

PCF attempts to recognize read requests from the dir command and return EOF when dir expects it. DOS directory files do not have an ordinary file length. Without the EOF optimization, EOF is only returned at the end of the space allocated to the directory file. This makes dir very slow on long directory files even when they are empty. PCF returns EOF when it encounters a 32-byte read for a directory entry that has never been used and is preceded in that sector by another directory entry that has never been used. This makes dir much more efficient.

Removing Directories from a PC Disk

The OS-9 deldir (delete directory) utility is not supported in PCF. You can, however, remove a directory by following the steps.

WARNING

Steps one and two (following) are critical. They must be performed or the disk file structure could be damaged beyond repair.

Step 1. Remove all subdirectories from the directory you are going to delete. Failure to do so before you try to delete the directory may result in corruption of the disk's file structure.

Use Step 2 through Step 4 that follow for each subdirectory (or directory) you want to remove.

Step 2. Delete all files from the directory. Failure to do so before you try to delete the directory may result in disk file structure corruption.

For example, go to the directory containing the files you want to delete and type:

```
del *
```

Step 3. When you are sure there are no subdirectories and/or files, remove the directory bit from the directory you wish to delete. Use the OS-9 `attr` command to do this.

For example, to turn off the directory bit of the `mydir` directory:

```
attr mydir -nd
```

Step 4. After you remove the directory bit, delete the directory with the `del` command.

```
del mydir
```

The partdgen Utility

Besides the supported standard OS-9 utilities, PCF provides the `partdgen` utility that displays information about a partitioned or non-partitioned disk and generates descriptors for partitions.

partdgen

Display Disk Information/Generate Descriptors for Partitions

Syntax

```
partdgen <diskid> [<options>]
```

Description

`partdgen` displays information about a disk and generates descriptors for disk partitions.

If the disk is partitioned, you can use the `-n` or `-g` option to display information about each partition.

PCF fails on ordinary access to an entire partitioned disk. For example, if you use the `dir` utility to display a directory of `/h0` (a partitioned disk), the result is a **bad type** error code (`000:249 E$BTyp`). The descriptors generated by `partdgen` contain a logical sector offset value directing PCF to the boot sector of a partition.

You must load the generated partitions before you can use them.

Note

PCF supports access to a raw partitioned disk.

`partdgen` does not, at this time, validate the descriptor it is given beyond trying raw access on it. It is, for instance, often possible to run `partdgen` on an RBF descriptor. The generated partition descriptors are RBF partition descriptors for a PCF disk.

Assuming the partitioned disk contains PCF partitions, RBF descriptors do not work unless they have been converted with the OS-9 `moded` (edit module) utility. Refer to **PCF Device Drivers** for more information.

Parameters

`diskid` The device name

Options

`-d` Partition descriptor sector is PC-DOS format (default).

`-g` Generate device descriptors for all partitions.

`-l` Long format display. Include FAT analysis.

`-n[=]<name>` Use `<name>` as the first partition name. The default is `<diskid>a`. `-n` automatically activates the `-g` option.

`-p` Dump only partition information. The default is to print additional information from the boot sectors. The defaults are:

- Do not make descriptors
- Short format
- Use boot sectors

The `-g` or `-n` option causes `partdgen` to create files in the current data directory. `partdgen` assigns a descriptor to each partition it finds. By default these descriptors (and files) are named by appending `a` through `z`, then `0` through `9` to the base device name.

`partdgen` increments the last character of the specified name up to z, then uses `0` through `9`. After it uses `9` (or encounters the last partition) it stops creating partition descriptors. If PCF runs out of name options, it generates the error message: `Too many partitions. Use a different base device descriptor name (DDname)`. You can use the `-n` option to specify a starting name. Since DOS cannot handle more than 26 devices and partitions on one disk, most base name choices should work.

Examples

The standard display is dense. Here is a sample command and its output:

```
partdgen d1 -g
(0) Partition: 1/1  0 (not bootable)   Type: 6 (huge partition)
Start Sect 32 for 102368 sects [(cyl,sect,head) (0,1,1) to (49,32,63)]
SysID    SSiz SPC Res FATs DirSz  Sects Fmt FATsz SPT Sids   Hidn   Note
MSDOS4.0  512   4   1    2   512 102368  F8   100  32   64     32 Fixed disk
```

The `Partition: 1/1` code indicates this is the first partition on the disk. Partitions within 1/1 are numbered 1/2, 1/3, and so on. A description of the other field labels follows.

Table 1-3 Field Labels

Label	Description
`SysID`	The system ID code in the boot sector
`SSiz`	Sector size
`SPC`	Sectors per cluster
`Res`	Reserved sectors
`FATs`	Number of file allocation table (FAT) copies
`DirSz`	Number of entries in the root directory
`Sects`	Sectors on disk
`Fmt`	The format ID code
`FATSz`	Sectors per FAT
`SPT`	Sectors per track
`Sids`	Sides
`Hidn`	Hidden sectors
`Note`	A description of the disk type (if it is known to `partdgen`)

The long display generated with the `-l` option contains more information and is not as tightly formatted:

```
partdgen d1 -gl
(0) Partition: 1/1  0 (not bootable)   Type: 6 (huge partition)
Start Sect 32 for 102368 sects [(cyl,sect,head) (0,1,1) to (49,32,63)]
                 System ID: MSDOS4.0
               Sector size: 512
       Sectors per Cluster: 4
          Reserved Sectors: 1
                FAT copies: 2
       Root directory size: 512
           Sectors on disk: 102368
                 Format ID: F8 (Fixed disk)
           Sectors per FAT: 100
         Sectors per track: 32
                     Sides: 64
  Special reserved sectors: 32

*** Calculated values (boot sector is sector 32) ***
Main directory start sector: 233  ($1D200)
Data start sector:           265  ($21200)
Data sectors:                102135
Total bytes:                 51184k
Data bytes:                  51067k
*** From FAT16 analysis.  In the data area there are:
 25468 free clusters
    63 used clusters
     0 bad clusters
```

Chapter 2: PCF/RBF Device Drivers

This chapter includes the following topics:

- **PCF Device Drivers**
- **Partitioned Disk Support**
- **RBF/PCF Device Descriptor Fields**
- **PCF Path Descriptor Definitions**
- **Supported DOS Format Table**
- **PCF/RBF Incompatibilities**
- **Converting RBF Device Descriptors to PCF**

PCF Device Drivers

The interface between PCF and device drivers was designed for RBF. RBF drivers handle the disk descriptor sector. Because PCF disks have an entirely different sector zero layout, PCF interacts unconventionally with drivers.

The driver copies part of sector 0 into the device's drive table entry and may use these values to set hardware parameters. Under DOS, sector 0 is called the **boot sector**.

The boot sector usually contains the information RBF drivers require from sector 0, but the information is in different places and coded differently.

Basically, this means RBF drivers supporting variable sector size as specified in the device descriptor work with PCF except when they read sector zero.

When a PCF driver encounters sector zero, the following occurs:

- The driver returns a bad-type or bad-sector-size error *(*`000:249 E$BTyp` or `000:241 E$Sect`) when it finds incorrect values in sector zero. PCF ignores the error.
- After each read of sector zero, PCF sets many of the values in the drive table:
  - If PCF can determine the disk parameters directly from the contents of sector zero, it initializes the drive table.
  - If PCF cannot determine the disk parameters directly, it determines whether it is dealing with a raw partitioned disk. If it is, PCF initializes the drive table based on values from the path descriptor. Otherwise, PCF uses the information from the device descriptor.
  - Sector zero may also be a boot sector requiring additional information from the FAT. In this case, PCF initializes the drive table according to the path descriptor, then reads sector 1 (the beginning of the FAT) and resets the drive table as indicated by the format ID in the FAT.

- Because drivers may react by setting up device hardware when they detect a read to some other sector after a read of sector zero, the following occurs:

  1. PCF saves the value of `V_Init` from the drive table.
  2. PCF sets `V_Init` to `1` to tell the driver that the hardware has been initialized. This prevents the driver from initializing the hardware based on the structure loaded into the drive table from sector zero.
  3. After reading sector 1, PCF restores the original value of `V_Init`.

  If the hardware is already initialized and this is a subsequent read of sector zero, PCF saves and restores 1 (TRUE) to `V_Init` to prevent superfluous hardware initialization.

  The driver must not initialize the hardware immediately when it reads sector zero, and it must not initialize the hardware when `V_Init` is 1. When it does initialize the hardware, it must use the actual drive table entry.

- PCF can handle the following format IDs:

  - `f0`
  - `f8`
  - `f9`
  - `fb`
  - `fc`
  - `fd`
  - `fe`
  - `ff`

  Formats `ff` and `fe` support disks with incomplete information in the boot sector (very old disk formats) by using default disk format information.

PCF uses the following default disk formats:

Table 2-1 Default Disk Formats

Description	ff	fe
FAT start sector	2	2
FAT copies	2	2
Root directory entries	112	64
Sectors per cluster	2	1
FAT size	1	1
Sectors per track	8	8
Sides	2	1
Sectors	640	320

The process of actually initializing the drive table involves some negotiation with the driver. If the FMInit field in the drive table is not set to 2, PCF sets it to 2 and zeros the V_FATLinks, V_Flags, and V_FATPtr fields in the drive table. If it is already set to 2, PCF does nothing to the drive table.

PCF uses and may update the following fields:

V_SectSize
V_FATS
DD_TOT
DD_SPT
DD_DIR
DD_FATCnt
DD_SPC
DD_DSK = 0xC0DE
PD_TOS

V_DirEntries
V_DataStart
DD_TKS
DD_FMT
DD_SectSize
DD_FATSIZ
DD_OWN =0
DD_ATT = 0x0FF

Partitioned Disk Support

The way PCF deals with partitioned disks requires OS-9 to handle multiple descriptors for the same device. These descriptors refer to partitions, separate logical devices, each with its own format and locking requirements.

Each partition needs its own drive table entry. To accommodate this, PCF creates a drive table entry when the first path is opened to a partition and frees the memory when the last path is closed.

When it frees a drive table entry, if `V_ScZero` is non-zero, PCF frees `V_SectSize` bytes of memory at `V_ScZero`.

PCF creates a drive table entry if the LSN offset in the path descriptor is non-zero.

To support partitioned disks a driver must **not**:

- Store pointers to allocated memory other than `V_ScZero` in the drive table.
- Use `V_ScZero` for anything but a pointer to `V_SectSize` bytes of memory acquired with `F$SrqMem`.
- Free the memory pointed to by `V_ScZero` without checking for a zero pointer in that field.

In addition:

- If the drive stores data particular to a logical drive, it must be in the drive table entry, `V_ScZero`, or other static storage. PCF does not support `V_DText` for partitioned disks.

The most likely consequences of drivers that do not work well with fake drive table entries are memory leakage and bus faults.

RBF/PCF Device Descriptor Fields

The correspondence between RBF and PCF fields in the device descriptor is shown below.

Drive Table Layout: Sector 0

Table 2-2 Drive Table Layout: Sector 0

Type	RBF	PCF	Description
uchar	dd_tot[3]	DD_TOT[3]	Sectors on the disk
uchar	dd_tks	DD_TKS	Track size in sectors
ushort	dd_map	DD_FATSIZ	Number of sectors in the FAT
ushort	dd_bit	DD_SPC	Sectors per cluster
uchar	dd_dir[3]		
ushort		DD_DIR	Address of the root directory
u_char		DD_reserved	
uchar	dd_own[2]	DD_OWN[2]	Owner ID (not used)
uchar	dd_att	DD_ATT	Disk attributes
ushort	dd_dsk	DD_DSK	Disk ID
uchar	dd_fmt	DD_FMT	Disk format

Table 2-2 Drive Table Layout: Sector 0 (continued)

Type	RBF	PCF	Description
uchar	dd_spt[2]	DD_SPT[2]	Sectors/track
uchar	dd_res[2]	DD_FATCnt	Number of copies of the FAT
		DD_FirstFAT	Location of the first FAT on disk
uchar		DD_Alignment1	Filler

Drive Table Layout—Other Sectors

Table 2-3 Drive Table Layout—Other Sectors

Type	RBF	PCF	Description
ushort	v_trak	V_TRAK	Current track number
pointer	v_filehd	V_FileHd	Open file list for disk
ushort	v_diskid	V_direntries	Entries in the root directory
ushort	v_bmapsiz	V_FATSz	Entries in the FAT
ushort	v_mapsct	V_DataStart	First data sector
ushort	v_bmb	V_FATLinks	Number of paths using cached FAT
pointer	v_sczero	V_ScZero	Pointer to sector 0 buffer

Table 2-3 Drive Table Layout—Other Sectors (continued)

Type	RBF	PCF	Description
uchar	v_zerord	V_ZeroRd	Sector 0 read flag
uchar	v_init	V_Init	Drive initialized flag
ushort	v_resbit	(uchar V_Flags, V_FMInit) File manager flags	
			PCF has device flag
ulong	v_softerr	V_SoftEr	Soft error count
ulong	v_harderr	V_HardEr	Hard error count
pointer	v_cache	V_Cache	Reserved for driver
pointer	v_dtext	V_DTExt	Reserved for driver
ushort	v_maxmap	V_SectSize	Disk sector size
ushort	reserved1	reserved1	
pointer	reserved	V_FATPtr	Pointer to cached FAT
ushort	reserved[8]	reserved[8]	

PCF Path Descriptor Definitions

The values in the device descriptor's options section are slightly redefined for PCF. The first few fields match RBF:

```
uchar PD_DTP; /*Device type*/
uchar PD_DRV; /*Drive number*/
uchar PD_STP; /*Step rate*/
uchar PD_TYP; /*Disk device type*/
uchar PD_DNS; /*Density capability*/
```

The next field is named `reserved1` in `moded`'s file for RBF. For PCF, if this is `$08`, PCF converts between DOS and OS-9 conventions for EOL when it performs `I$ReadLn` and `I$WritLn` service requests.

```
char PD_NewLine; /*New line handling on ReadLn/WritLn*/
```

These fields match RBF:

```
ushort PD_CYL; /*Number of cylinders*/
uchar  PD_SID; /*Number of sides*/
uchar  PD_VFY; /*0=verify disk writes*/
ushort PD_SCT; /*Default sectors per track*/
ushort PD_TOS; /*Default sectors per track (tr0, s0)*/
```

PCF does not use the `PD_SAS` value to define the increments of file extension as it does in RBF. PCF uses it to specify the size of non-root directory files. Non-root directory file size is computed as `PD_SAS * DD_SPC` sectors.

```
ushort PD_SAS; /*Segment allocation size*/
```

These fields match RBF:

```
uchar PD_ILV;   /*Sector interleave offset*/
uchar PD_TFM;   /*DMA transfer mode*/
uchar PD_TOffs; /*Track base offset*/
```

The sector base offset value should generally be 1.

```
uchar PD_SOffs; /*Sector base offset*/
```

The sector size should be 512 except for certain disk formats using 1024-byte sectors. Refer to the **Supported DOS Format Table** section on page 35 for more information.

```
ushort PD_SSize;    /*Size of sector in bytes*/
ushort PD_Cntl;     /*Control word*/
uchar  PD_Trys;     /*Number of tries (1=no error correction)*/
uchar  PD_LUN;      /*SCSI unit number of drive*/
ushort PD_WPC;      /*First cylinder using write precompensation*/
ushort PD_RWC;      /*First cylinder reduced write current*/
ushort PD_Park;     /*Park cylinder for hard disks*/
ulong  PD_LSNOffs;  /*LSN offset for partition*/
ushort PD_TotCyls;  /*Total cylinders on device*/
uchar  PD_CtrlrID;  /*SCSI controller ID*/
uchar  PD_Rates;    /*Data transfer + rotational speed*/
ulong  PD_SCSIOpts; /*SCSI options*/
ulong  PD_MaxCount; /*Maximum byte count driver can handle*/
uchar  PD_reserved3[5];
uchar  PD_ATT;      /*File attributes*/
```

The following fields are somewhat like RBF, but the differences make PCF unable to support `pd` and `dsave`. PCF does not work with real file descriptors (FDs). PCF FDs are constructed in RAM and given imaginary disk locations.

```
ushort  PD_FCluster; /*Starting cluster (was PD_FD)*/
ushort  PD_Padding;  /*Just filler*/
ulong   PD_DFD;      /*Directory FD psn*/
ulong   PD_DCP;      /*Directory entry pointer*/
POINTER PD_DVT;      /*Device table pointer (copy)*/
```

`PD_Stack` is reserved for optimization of PCF's private stack allocation system.

```
POINTER PD_Stack;    /*Address of cached stack*/
uchar   PD_reserved4[22];
```

Note

PCF file names, compared to OS-9 file names, are short. The file name here was stored by IOMan.

```
uchar PD_Name[12];    /*Filename*/
char  PD_NotName[20]; /*Leftover space*/
```

Supported DOS Format Table

Some controllers/drives have special requirements for rotational velocity and data transfer rate. Refer to the hardware documentation before adjusting the `PD_Rate` (data transfer/rotational rate) field to reflect these values.

Table 2-4 DOS Format Table

		Trks	Byte/S	S/Clust	Resrvd	FATs	DirEnts	RotV	TrkD
	Name ID	Sects	S/FAT	S/Trk	Heads	Hidden	Capacity	Rate	Inchs
1	2HD-1.44	80	512	1	1	2	224	300	96
F0		2880	9	18	2	0	1440K	500	3.5
2	Fixed Disk	0	0	0	0	0	0	0	0
F8		0	0	0	0	0	0K	0	????
3	2DD9	80	512	2	1 2	112	300	96	
F9		1440	3	9	2	0	720K	250	3.5
4	2DD9	80	512	2	1	2	112	300	96
F9		1440	3	9	2	0	720K	50	5.25
5	2HC	80	512	1	1	2	224	300	96
F9		2400	7	15	2	0	200K	500	3.5
6	2HC	80	512	1	1	2	224	360	96
F9		2400	7	15	2	0	1200K	500	5.25
7	2DD8	80	512	2	1	2	112	300	96
FB		1280	2	8	2	0	640K	250	5.25
8	1D9	40	512	1	1	2	64	300	48

Table 2-4 DOS Format Table (continued)

		Trks	Byte/S	S/Clust	Resrvd	FATs	DirEnts	RotV	TrkD
	Name ID	Sects	S/FAT	S/Trk	Heads	Hidden	Capacity	Rate	Inchs
FC		360	2	9	1	0	180K	250	5.25
9	2D9	40	512	2	1	2	112	300	48
FD		720	2	9	2	0	360K	250	5.25
10	1D8	40	512	1	1	2	64	300	48
FE		320	1	8	1	0	160K	250	5.25
11	2HD	77	1024	1	1	2	192	300	96
FE		1232	2	8	2	0	1232K	500	3.5
12	2HD	77	1024	1	1	2	192	360	96
FE		1232	2	8	2	0	1232K	500	5.25
13	2D8	40	512	2	1	2	112	300	48
FF		640	1	8	2	0	320k	250	5.25

PCF/RBF Incompatibilities

The following describes incompatibilities between PCF and the Random Block File manager (RBF):

- PCF does not perfectly emulate file descriptors (FDs). Therefore, when `pd` and `dsave` attempt to follow directory chains backwards, PCF fails.
- A PCF sector zero is not formatted the same as an RBF sector zero. PCF can only read a device's sector zero in raw mode. If a disk is read in raw mode (for example, `dir /d0@`), PCF permits the program to see the real sector 0. This affects programs such as `bfed` that use sector zero to find the size of the disk when they open it as a file.
- There is no allocation map. Consequently, the `free` utility (to display free space on a mass storage device) does not work. Refer to the **PCF/OS-9 Compatibility** section in **Chapter 1: The OS-9 PC File Manager (PCF)** for a list of OS-9 utilities that do/do not work, or work only partially with PCF.
- When PCF reads a disk in raw mode and a directory is found, the actual DOS directory structure is shown. When the directory is read in ordinary mode, PCF translates the directory entries into RBF format. For the root directory this includes creating **.** and **..** entries in the root directory and shifting the rest of the directory over 64 bytes. This is necessary because `dir` expects **.** and **..** as the first two entries in each directory.
- PCF's service of I/O requests is generally identical to RBF's. The major exceptions are:
  - PCF supports file locking. RBF supports record and file locking.
  - The PCF `I$ReadLn` and `I$WritLn` requests can convert between OS-9 and DOS text file formats. RBF's `I$ReadLn` and `I$WritLn` cannot distinguish between OS-9 and DOS text file formats.
  - `I$ReadLn` and `I$WritLn` can modify the length of what they read and write. For instance, writing a line with ten characters and a carriage return with `I$WritLn` places 11 bytes in an RBF file; `I$WritLn` puts 12 bytes in a PCF file.

- PCF supports partitioned disks by storing the sector offset to the base of the partition in the device descriptor’s logical sector offset field. RBF does not support partitioning.

Converting RBF Device Descriptors to PCF

Use the OS-9 `moded` utility to convert any RBF device descriptor (MVME/320, OMTI 5000, and/or TEAC SCSI floppy) into a PCF device descriptor (for the same hardware) as follows:

Note

These examples assume the original device is `d0` and the PCF device is `p0`.

Step 1. Copy or save the RBF descriptor into a file with the name of the new device. `moded` restricts the new device name to a length less than or equal to the RBF device name.

For example:

```
save d0 -f=p0
```

Step 2. Call `moded`.

```
moded d0 -f=p0
```

Step 3. Edit the descriptor.

Table 2-5 Descriptor Edits

Edit	Value
`M$FMgr`	Change RBF to PCF
`M$Name`	Change `d0` to `p0`
`Reserved`	Place `$08` in the reserved field of the device descriptor initialization table

Table 2-5 Descriptor Edits (continued)

Edit	Value
`PD_CYL`	Set `PD_CYL` to the total number of cylinders on the device
`PD_VFY`	Turn off disk write verification if there is any doubt that the driver fully supports non-256-byte sectors. A driver that is oblivious to sector length may work with PCF provided it never attempts to save a sector. Because verification involves reading after a write, the driver must have a buffer. Do not use a 256-byte buffer for a 512-byte sector.
`PD_SCT` and `PD_TOS`	Set both sectors per track fields to the same value, choosing an appropriate value for the default DOS format for that drive. 0 is a good value for SCSI hard disks.
`PD_SAS`	Select a value for the **segment allocation size**. PCF allocates this many clusters for each non-root directory. **NOTE:** Large values slow directory creation and consume disk space. A segment allocation size of two should be enough for general use.
`PD_SOffs`	Set sector base offset to 1. This might vary on fixed disks, but one is certainly the first value to try, followed by zero.
`PD_TOffs`	Set track base offset to 0
`PD_SSize`	As appropriate, set the sector size to 512 or 1024

Table 2-5 Descriptor Edits (continued)

Edit	Value
`PD_MaxCnt`	If multi-sector-I/O is enabled in the control word, ensure the maximum transfer count field is present and has the correct value. 0x0000ffff is a common value. You should turn on multi-sector I/O because it performs better than single-sector I/O. This is particularly important on slow controllers, such as the MVME320.

`PD_Cntl`
`(bit 0)`
	Set the device descriptor's format inhibit bit to 1 to protect you from inadvertently using the `format` utility and possibly corrupting your disk. The OS-9 `format` utility does not know how to create a PC-DOS disk and can not proceed if the format inhibit bit is set.

Step 4. Write the updated descriptor (type **w**), and quit `moded` (type **q**).

Appendix A: OS-9 PCF Descriptors

This appendix contains `moded` listings of PCF descriptors for the following:

- **FD235 SCSI Flexible Disk Drive**
- **3.5" DSDD (720K) Diskettes**
- **3.5" HD (1.44M) Diskettes**

FD235 SCSI Flexible Disk Drive

The following are moded listings for MVME/147 PCF descriptors using the FD235 embedded SCSI flexible disk drive. This drive uses the rbteac driver, and the driver must be edition #17 for PCF to work correctly. The descriptor pcd0 reflects the 3.5 inch 720K double-density PC format, while the pcd0h reflects the 3.5 inch 1.44M high-density PC format.

The main moded fields to alter from the d0 descriptor are below.

For the double-density descriptor pcd0:

```
descriptor name            : pcd0
file manager name          : pcf
device type                : $27
reserved                   : $0d
default sectors/track      : 9
default sectors/track 0    : 9
segment allocation size    : 2
sector interleave factor   : 4
track base offset          : 0
sector size                : 512
control word               : $0003
data-transfer/rotation     : $10
```

For the high-density descriptor pcd0h:

```
descriptor name            : pcd0h
file manager name          : pcf
device type                : $27
reserved                   : $0d
number of cylinders        : 80
default sectors/track      : 18
default sectors/track 0    : 18
segment allocation size    : 2
sector interleave factor   : 4
track base offset          : 0
sector size                : 512
control word               : $0003
```

```
data-transfer/rotation     :  $10
```

Examples of complete listings of these descriptors can be found on the following pages.

3.5" DSDD (720K) Diskettes

The following is an example of a PCF descriptor for 3.5 inch double-sided, double-density (DSDD) diskettes.

```
OS-9/68000 Module Editor
Copyright 1987 Microware Systems Corp.
Type ? for editing help message

moded:
descriptor name                  : pcd0
file manager name                : pcf
device driver name               : rbteac
port address                     : $fffe4006
irq vector                       : 69
irq level                        : 4
irq priority                     : 5
device mode capabilities         : $a7
device class                     : $01
drive number                     : 0
step rate                        : 3
device type                      : $27
density                          : $03
reserved                         : $0d
number of cylinders              : 40
number of heads/sides            : 2
disk write verification          : 1
default sectors/track            : 9
default sectors/track 0          : 9
segment allocation size          : 2
sector interleave factor         : 4
dma transfer mode                : 0
track base offset                : 0
sector base offset               : 1
sector size                      : 512
control word                     : $0003
number of tries (1=no retry)     : 7
scsi unit number of drive        : 0
```

```
write precompensation cylinder   :  0
reduced write current cylinder   :  0
cylinder to park disk head       :  0
logical sector offset            :  0
total cylinders on device        :  80
scsi controller id               :  $06
data-transfer/rotation rate      :  $10
scsi options flags               :  $00000001
maximum transfer count           :  $0000ffff

moded: eof
```

3.5" HD (1.44M) Diskettes

The following is an example of a PCF descriptor for 3.5 inch high-density (HD) diskettes.

```
OS-9/68000 Module Editor
Copyright 1987 Microware Systems Corp.
Type ? for editing help message

moded:
descriptor name                  : pcd0h
file manager name                : pcf
device driver name               : rbteac
port address                     : $fffe4006
irq vector                       : 69
irq level                        : 4
irq priority                     : 5
device mode capabilities         : $a7
device class                     : $01
drive number                     : 0
step rate                        : 3
device type                      : $27
density                          : $03
reserved                         : $0d
number of cylinders              : 80
number of heads/sides            : 2
disk write verification          : 1
default sectors/track            : 18
default sectors/track 0          : 18
segment allocation size          : 2
sector interleave factor         : 4
dma transfer mode                : 0
track base offset                : 0
sector base offset               : 1
sector size                      : 512
control word                     : $0003
number of tries (1=no retry)     : 7
scsi unit number of drive        : 0
```

```
write precompensation cylinder   :  0
reduced write current cylinder   :  0
cylinder to park disk head       :  0
logical sector offset            :  0
total cylinders on device        :  80
scsi controller id               :  $06
data-transfer/rotation rate      :  $31
scsi options flags               :  $00000001
maximum transfer count           :  $0000ffff

moded: eof
```