OS-9 Level II

SDISK3-DMC Software

User's Manual

Version #3

Sardis Technologies
Ph. (604) 255-4485 (Pacific Time)

2261 East 11th Ave.
Vancouver, B.C.
Canada V5N 1Z7

Printed in Canada

CREDITS

Special thanks to Jim B., Larry W., Kevin D., and Kent M. for their help.

TRADEMARKS

"Radio Shack" is a trademark of Tandy Corp.
"OS-9" is a trademark of Microware Systems Corp. and Motorola
"FLEX" is a trademark of Technical Systems Consultants
"CP/M" is a trademark of Digital Research Inc.
"MS-DOS" is a trademark of Microsoft Corp.
"IBM" is a trademark of International Business Machines Corp.

COPYRIGHT NOTICE

DISCLAIMER

Although much effort has been made to ensure the accuracy of the software and documentation, Sardis Technologies and D.P. Johnson disclaim any and all liability for consequential damages, economic loss, or any other injury arising from or on account of the use of, possession of, defect in, or failure of the supplied material.

This manual last revised May 15, 1988

0.0 TABLE OF CONTENTS

>>>>>> READ THIS FIRST <<<<<<

1.0 ABOUT YOUR ORDER

Before you open the envelope containing the diskette, examine the documentation to confirm that you have received what you ordered. Especially read the "Introduction" and "System Prerequisites" sections to verify that the software you ordered is what you need for your system. If you have received the wrong software, return the unopened diskette package with all documentation and a note stating the problem, and we will send you the correct item. No exchange will be given after the diskette envelope is opened, except to replace a defective disk (refer to the "Trouble-shooting" section).

2.0 USER FEEDBACK

If you have any suggestions on how we can improve the software or manuals please let us know. If you discover any bugs in the supplied software, or errors and omissions in this manual, call or write to tell us about it -- we can't fix problems we're not aware of! On the other hand, unsolicited testimonials will be accepted too!

3.0 INTRODUCTION AND SPECIFICATIONS

At the core of this SDISK3-DMC Level II package is a replacement for the CC3Disk floppy disk device driver that was in the OS9Boot file on the Radio Shack OS-9 Level II system disk. The new driver, called SDISK3, in conjunction with new device descriptors (D0, D1, D2, D3, DD), supports more configurations and sports more features than the Radio Shack drivers:

- 35, 40 or 80 track drives
- access 48 tpi disks in a 96 tpi drive
- set different step rates for each drive
- up to 3 double sided drives, or up to 4 single sided drives (4 double sided drives possible with hardware modification)
- reads/write/formats several OS-9 disk formats:
  - CoCo OS-9
  - standard OS-9, single or double density
  - Mizar OS-9/68K
  - Japanese OS-9 (eg. Fujitsu)
- reads and writes non-OS9 disks with sector lengths of 128, 256, 512, or 1024 bytes, with or without automatic retry
- disk caching is provided to speed up disk accesses and reduce wear and tear on the drives
- supports the new Sardis DMC "no halt" mode:
  - leaves interrupts enabled during disk I/O for better keyboard and mouse response, to avoid losing data being received on serial ports, and to let the software clock keep time
  - lets other tasks continue processing while disk I/O is taking place to speed up screen displays and improve total throughput
  - several versions of the disk driver, implementing three variations on the "no halt" theme, are included

This version of SDISK3 is produced under license from D.P. Johnson. The DESCGENL2, DISKTYPE, and SFORMAT commands are exactly as D.P.J. supplies them. The SDISK3 device driver itself, although highly modified internally by Sardis Technologies (to implement the new "no halt" mode), supports all of the features of D.P. Johnson's regular package and is functionally identical to it. The major departure from the SDISK3 package sold directly by D.P.J. is limited to our installation procedure. We supply four proprietary utilities (BOOTMOD, CMDGEN3, DPOKE, NEWCRC) to streamline the process of creating a new OS9Boot file. As a result, we have left out the INSTALL macro, and the MODBUSTER and FLS commands. The manual has also been completely re-written.

Any problems, questions, or suggestions regarding this software package should be directed to us, not to D.P. Johnson.

The BOOTFIX command usually supplied with Level I versions of SDISK is not needed for OS-9 Level II as the new OS9GEN, COBBLER, and CONFIG commands are smart enough to put the kernel in its proper place on a double sided disk, and the new BOOT module can also find and load an OS9BOOT file that spans both sides of a double sided disk.

4.0 SYSTEM PREREQUISITES - what you need to run SDISK3-DMC on the CoCo

1) A Tandy Color Computer 3 (CoCo 3) with 128K or 512K memory. (If you have a CoCo 1 or 2, this is the wrong version of the DMC software -- you need our SDISK-DMC Level I version instead.)

2) A Radio Shack "Multi-Pak Interface" (MPI), or similar unit. Although the DMC controller will run when plugged directly into the CoCo 3, you risk overheating and damaging the CoCo's internal power supply if you omit the MPI. Using the PBJ CC-Bus instead of the Radio Shack MPI is not recommended, as we have had several reports of the DMC not running reliably on the CC-Bus.

3) The DMC floppy disk controller cartridge from Sardis Technologies. The SDISK3-DMC software package will NOT run on any other disk controller. Also, the PAL-4 "daughter board" upgrade must be installed in the DMC controller for this version of SDISK3-DMC to work. All units shipped after mid January 1988 have already been upgraded at the factory.

4) At least one 5 1/4" floppy disk drive. Although OS-9 Level II and SDISK3 will run on a one drive system, a two drive configuration is strongly recommended, as commands such as BACKUP are awkward and slow when running on a one drive system. Our installation instructions assume that you have at least two floppy drives, where drive 0 is some type of 48 tpi drive (5.25", single or double sided, 35 or 40 tracks); the other drives may be any type. If you already have drive 0 set up as an 80 track drive, or only have one floppy drive, you will have to come up with your own installation procedure.

5) A copy of Radio Shack's OS-9 Level II Version 2.00.01, or later. A diskette labelled "OS-9 Level Two Operating System -- System Master" is included in that package and will be referred to as the "OS-9 SYSTEM" disk throughout this manual. NOTE -- it's very easy to get confused between "Level Two" and "Version 2". OS-9 Version 2.00.00 for the CoCo 2 is really Level One OS-9, not Level Two.

6) The "SDISK3-DMC Level II" package from Sardis Technologies, which is what this manual is part of. A diskette labelled "SDISK3-DMC Level II" is included in this package, and will be referred to as the "SDISK3-RELEASE" disk throughout this manual. Section 8.0 lists the contents of the disk.

5.0 BEFORE YOU GET STARTED

No amount of documentation will do you any good if you don't read it. The documentation included with this software assumes you have a basic knowledge of using your system and does not explain in-depth information that is covered elsewhere (in your system manuals). We have tried to use terminology consistent with that used in the OS-9 system documentation. If you have not at least read through the OS-9 manuals that came with your system, we strongly urge you to do so now. If you do not understand something about our documentation, first see if there is some word you skipped over that you did not understand (like "pipes", "device descriptor", "LSN", "I/O redirection", etc.) that is explained in the OS-9 Commands or OS-9 Technical Reference manuals supplied by Radio Shack. Study those manuals, then reread our documentation. If it still does not make any sense, then try giving us a call.

Before attempting to use the SDISK3 software, you should have read the "OS-9 COMMANDS" manual up to where the detailed command descriptions start (page 6.4 for Level II) and be familiar with the terminology and basic features of OS-9 as explained in that manual. Also read our entire manual through once or twice before beginning any of the procedures given.

Some parts of the manual have a lot of technical details, or discuss many options that are available to you. No need to be alarmed or confused! If you try only the basic features first, you should get up and running quickly and easily. For example, although we supply half a dozen versions of the SDISK3 device driver, you should always install the SDISK3.3dnc driver (or .4snc if you have 4 single sided drives) to start with. You don't have to understand anything about the differences in the inner workings between what we call the "NAP", "VIRQ", or "IRQ" versions. Later on, when you have had time to thoroughly digest the whole manual, especially the "Theory of Operation" section, you can decide whether you want to try any of the other versions of the driver. Even then, many of you will never want to, nor need to.

6.0 SOFTWARE INSTALLATION INSTRUCTIONS

NOTE - the text below that is underlined represents what you key into the computer; the rest is instructions and comments.

Creating a new OS-9 Level II SDISK3-DMC customized boot disk on a system having two floppy disk drives: drive 0 must be a 48 tpi 5.25" drive (35 or 40 track, single or double sided), drive 1 may be any kind of floppy drive, even 96 tpi 80 track, or 3.5" drives.

---

1) You will require 2 blank disks (3 disks if you need to do the backup in step #3) for the installation procedure described here. It should take approx. 20-40 minutes to complete all the steps, depending on how fast you type, how fast you read, how familiar you already are with OS-9 in general and with the new commands we supply, as well as how much you choose to do in steps 3, 8, 9, 16.

2) Put a backup copy of the Radio Shack "Color Computer 3 OS-9 Level Two Operating System -- System Master" disk into drive D0 (first make sure it is write-protected), press the RESET button twice, then type DOS after the "OK" prompt to cold boot OS-9. If OS-9 does not boot, refer to the "Trouble-shooting" section in this manual, and in the DMC hardware manual.

3) If you already have a backup copy of the Radio Shack "Color Computer 3 OS-9 Level Two Operating System -- System Master" on a disk that is readable in drive D1 (ie. the disk currently in D0 is also readable in drive D1, or you already have another copy in a format readable in drive D1), skip this step and go on to step #4. Otherwise label a blank disk as "OS-9 System Master Backup", put it into drive D1 and type:

```
FORMAT /D1 1 '35' "xxxx" R
BACKUP /D0 /D1 #48K          [or #16K on a 128K machine]
```

When the backup is complete, remove the disk from drive D1 and put it aside. It will be used in steps 15 and 16.

4) Label a blank disk (or a disk you don't mind being erased) as "SDISK3-DMC Backup", put it into drive D1, and type:

```
FORMAT /D1 1 '35' "xxxx" R
```

5) Loading several small modules into memory individually wastes memory (which we especially can't afford to do on a 128K machine), so first merge them into one file:

```
CHD /D0/CMDS
MERGE FORMAT COBBLER MODPATCH BACKUP MAKDIR >/d1/format
ATTR /D1/format e
LOAD /D1/format
```

6) Replace the disk in drive D0 with the "SDISK3-DMC Level II RELEASE" disk, first making sure it is write protected, then type:

```
BACKUP /D0 /D1 #48K          [or #16K on a 128K machine]
```

When the backup is completed, remove the original "SDISK3-DMC Level II RELEASE" disk from drive D0, and put it away in a safe place.

7) DESCGENL2 creates the new device descriptors SDISK3 needs; it will prompt you for the number of drives (2 or 3), then asks for the stepping rate, number of tracks, and number of sides of each of the drives. If you have 4 drives, don't despair -- specify 3 drives for now and we'll take care of the fourth drive in the next step:

```
CHX /D1/CMDS ; CHD /D1/MODS
DESCGENL2 -d=0
```

8) Use DMODE (refer to detailed instructions elsewhere in manual) to display and modify the descriptors you just created, directly on disk (in directory /D1/MODS). For example, you might want to change the interleaving factor to 03, or disable verify. If you have a 4 drive system, notice that a descriptor for D3 is present after all, but you may need to modify the stepping rate, interleaving factor, and number of tracks and sides. Excerpts of a typical session would look like this:

```
DMODE -D0
DMODE -D0 ilv=03
   :
DMODE -D3
DMODE -D3 stp=01 ilv=02 cyl=0023 sid=01 vfy=01
```

9) Unless you want to change some of the default values the SDISK3 driver uses, skip this step and go to step #10. Otherwise refer to the section near the end of this manual titled "Changing User Values in the SDISK3 Driver", then use DPOKE and NEWCRC to make the desired changes. For example, if you want to reduce the motor-up-to-speed delay from .75 sec. to .50 sec., type:

```
DPOKE /D1/MODS/SDISK3.3dnc 18 00 20        [.4snc if 4 drives]
NEWCRC /D1/MODS/SDISK3.3dnc                [.4snc if 4 drives]
```

10) If your drive D0 is 40 tracks and/or double sided, run DMODE to update the Radio Shack descriptor in memory:

```
DMODE /D0 sid=02            [omit if single sided drive]
DMODE /D0 cyl=0028          [omit if 35 track drive]
```

11) OS-9 (Version 2.00.01) has a bug in it that affects SDISK3, so this patch is needed to get around the problem:

```
MODPATCH /D1/PATCHES/InitIrq.F.pat -w -s
```

12) Label a blank disk as "SDISK3-DMC Boot", put it into drive D0, and type:

```
FORMAT /D0 "SDISK3-DMC Boot" R      [use Format, not Sformat]
```


13) COBBLER and BOOTMOD will put the kernel and customized boot file onto the disk:

```
COBBLER /D0
BOOTMOD /D0 /D1 -S CC3DISK D0 D1 DD
/D1/MODS/SDISK3.3DNC      [specify SDISK3.4SNC if 4 drives]
/D1/MODS/D0
/D1/MODS/D1
/D1/MODS/D2               [omit if only 2 drives]
/D1/MODS/D3               [omit if only 2 or 3 drives]
/D1/MODS/DD
CTRL BREAK
```

14) A number of programs need to be copied onto the new boot disk; the "copycmds1" procedure does this for you:

```
/D1/copycmds1
```

15) Replace the disk in drive D1 with a backup copy of the "Color Computer 3 OS-9 Level Two Operating System -- System Master" disk, and run the following procedure to copy the rest of those programs that are needed on a minimum boot disk:

```
/D0/copycmds2
DEL /D0/copycmds2
```

16) Copy other files, as desired, from drive D1 to drive D0, then remove the disk from drive D1.

17) Now press the RESET button to boot OS-9 from the new SDISK3-DMC configured boot disk in drive D0 (you may want to write-protect the disk first). If OS-9 does not boot, refer to the "Trouble-shooting" section in this manual, and in the DMC hardware manual.

7.0 HOW TO USE NEW COMMANDS

The next several pages describe the utility commands provided in the CMDS directory of the disk included with this package.

7.1 ALLOWSLEEP

Syntax: allowsleep [/devname] [no] [yes] [imsg]

Function: Tells the SDISK3 device driver to use or not to use the F$SLEEP system call instead of software delay loops. Note -- if you only use the standard versions of the SDISK3 driver (.3dnc or .4snc extensions) you can completely ignore the ALLOWSLEEP command.

Parameters:

devname — The name of any disk drive associated with the SDISK3 device driver. If not specified, defaults to /D0. However, regardless of whether a drive is specified, or the default is used, the allowsleep status affects all the floppy drives controlled by the SDISK3 driver.

no / yes — Disable or enable the use of the F$SLEEP call for floppy disk I/O. If not specified, defaults to "yes".

imsg — Display the message to set the switch to IRQ position.

Notes:

* The "imsg" parameter should only be used if an IRQ version of the SDISK3 driver is installed and the required hardware modification has been made to the CoCo 3 itself. It only needs to be specified the first time ALLOWSLEEP is run after booting OS-9. If the "imsg" parameter is specified when ALLOWSLEEP is run from a procedure file (shell script), you must redirect "standard input" to the keyboard.

  The IRQ versions of the SDISK3-DMC device driver (.3dis4 extension), in the process of booting OS-9, only activate sleeping for non-I/O disk operations (motor-up-to-speed and head settle delays). ALLOWSLEEP with the "imsg" parameter should be included in the STARTUP file to enable sleeping for disk I/O operations (read/write/seek). Once full sleeping is enabled for IRQ drivers, it is usually left enabled.

* The NAP versions of the SDISK3-DMC driver (.3dnc, .4snc, .4dnc extensions), on the other hand, automatically enable full sleeping (both non-I/O and I/O disk operations) in the process of booting OS-9, so you do not need to run ALLOWSLEEP in the STARTUP file. Actually, there is no need to ever run ALLOWSLEEP with a NAP version driver installed.

* The VIRQ versions of the SDISK3-DMC driver (.3dvs4 extension) also automatically enable full sleeping in the process of booting OS-9. You do not need to run ALLOWSLEEP in the STARTUP file either. However, with the VIRQ drivers you may occasionally want to run ALLOWSLEEP to temporarily disable disk I/O sleeping in order to be able to run the faster interleaving factors of 03 and 02. With I/O sleeping enabled, 04 is the fastest interleaving factor that will give consistent performance with a VIRQ-type driver.

* Specifying the "no" parameter only disables sleeping during disk I/O operations -- sleeping during non-I/O operations always remains enabled.

* For more background information regarding the use of ALLOWSLEEP, and related aspects such as the F$Sleep call, VIRQ vs IRQ vs NAP versions of the SDISK3 driver, and disk interleaving factors, refer to the "Theory of Operation" section later in this manual.

Examples:
---

1) OS9:allowsleep /D1 no

2) OS9:allowsleep

3) The following line appears in a procedure file:
```
allowsleep imsg </term
```

7.2 BOOTMOD

Syntax: bootmod [/boot_dev [/modl_dev]] [opts] [modname[...]]

Function: Modifies an existing OS9Boot file by deleting some modules
--- and/or adding other new modules to it, then links to the new OS9Boot file to allow booting from it.

Parameters:
---

boot_dev — The name of the disk drive containing the OS9Boot file to be modified. If not specified, a default of "/D0" is used.

modl_dev — The name of the disk drive containing the modules to be added to the OS9Boot file. If not specified, a default of "/D0" is used. If this parameter is specified, the "/boot_dev" parameter must also be specified.

modname — The names of one or more modules that are to be removed from the existing OS9Boot file. If no modules are to be deleted, you can omit this parameter, but then at least one other parameter or option must be specified.

opts — None, one, or two of the following options:

Options:

-s — Invokes the "silent" mode, ie. don't display prompts. Used when you are familiar enough with the program not to need the prompts, or when running BOOTMOD from a procedure (shell script). Do not use this option in single drive operation unless the OS9Boot file and the modules to be added are all on the same disk, as the prompts are needed to tell you when to switch disks.

-w Treat the error messages "module not found in buffer" and "path name not found" (of file to be appended) as warnings only (ie. continue processing), not as terminal errors.

-? Displays the help message, then immediately quits.

Summary of Operation:
---

1) The entire existing OS9Boot file is read from the specified disk into a buffer in memory.
2) All the modules specified to be removed are deleted from the buffer.
3) Pathlists are read from the standard input path, and each file specified is read and added to the buffer.
4) All modules with bad CRC's are deleted from the buffer.
5) The existing OS9Boot file is deleted from the specified disk.
6) The new OS9Boot file is written to disk from the buffer.
7) The new OS9Boot file is linked for booting.

Notes:
---

* The primary use of BOOTMOD is to replace one or more modules in an existing OS9Boot file with newer or different versions of those modules. It can also be used to delete modules from the OS9Boot file to generate a smaller file that leaves more free memory. While the OS9GEN or CONFIG commands can also be used to accomplish the same function, BOOTMOD is often faster and easier to use, especially if you only have one disk drive.

* BOOTMOD doesn't create a temporary boot file before deleting the old OS9Boot file. If an error occurs while the new OS9Boot file is being written, you end up with a disk containing no OS9Boot file. Consequently, NEVER run BOOTMOD on an "original" disk, only on backup copies.

* Many of the modules in an OS9Boot file may not be left out if OS-9 is to run at all. Some of these are OS9p1, OS9p2, IOMan, RBF, SCF, etc. Refer to the Radio Shack manuals for details.

* BOOTMOD reads pathlists (file names) from the standard input path (the keyboard, or redirected from a file), one pathlist per line, just like OS9GEN. BOOTMOD opens each file and copies it into its buffer. The process is repeated until a blank line or an end-of-file marker is detected.

* Before writing the new OS9Boot file to disk, BOOTMOD searches its buffer for, and purges, any data that is not in a valid module, ie. a module with bad header parity or CRC, or data that is not a module at all. For example, COBBLER or OS9GEN sometimes write a few bytes of garbage at the end of the OS9Boot file. Any modules that BOOTMOD adds are appended to the end of the buffer, after the garbage. Without this purging pass, when the IDENT utility read the new OS9Boot file it would stop at the bad data with a "Module header is incorrect!" message and not display the valid modules that followed.

  However, if you use MODPATCH or DEBUG to patch (in memory) any module from the boot file, you must update the CRC of that module

(using MODPATCH's "V" command) before using COBBLER + BOOTMOD. Or if you used DPOKE to patch (on disk) any module to be added to the boot file, run NEWCRC before using BOOTMOD. If you don't, the patched modules will be purged and not be included in the new OS9Boot file.

* OS9Boot files may not be fragmented (containing more than one segment). OS-9's booting facility, in order to keep it simple, can only read boot files stored in contiguous sectors. The easiest way to ensure this, is to create the "old" OS9Boot file on a freshly formatted disk, then run BOOTMOD immediately after. Deleting files is the main contributor to fragmentation. In any case, BOOTMOD avoids creating a fragmented OS9Boot file if at all possible, and warns you not to boot from the disk if fragmentation did occur.

* The starting address and size of the new OS9Boot file are written to the disk's Identification Sector (LSN 0) for use by the OS-9 booting facility. This is what was referred to above as "linked for booting".

* Unlike OS9GEN, CONFIG, or COBBLER, BOOTMOD does not write the OS-9 kernel to track 34. As a result, before using BOOTMOD on a new disk, one of the other three commands must be used to create the "old" OS9Boot file, and write the kernel to disk.

Examples:
---

1) The following example only deletes modules from the boot file:

```
OS9:bootmod /d1 /d0 -s pipeman piper pipe aciapak t1 <enter>
<enter>
```

2) The following example adds 2 more modules to the boot file, but doesn't delete any. It can be run on a single drive system, with the boot file and new modules on different diskettes. The -s option is omitted so we will be told when to switch disks. The /d0 parameter is specified, even though it is the same as the default, because with no parameter or option specified BOOTMOD would just display the help message and quit.

```
OS9:bootmod /d0 <enter>
/d0/mods/xxxxx <enter>
/d0/mods/yyyyy <enter>
<enter>
```

3) The following lines create a procedure to run BOOTMOD (the -s option is required when BOOTMOD is run from a procedure):

```
OS9:build /d1/newboot <enter>
? bootmod /d1 /d0 -s cc3disk d0 d1 dd #40K <enter>
? /d0/mods/sdisk3.3dnc <enter>
? /d0/mods/d0 <enter>
? /d0/mods/dd <enter>
? <enter>
```

The procedure is then run:

```
OS9:/d1/newboot <enter>
```


Error Messages: Note that the pathlist or module name or option that caused the error appears at the beginning of the line immediately above the error message -- ignore the other names on that line. In the following example, module "aciapad" is the module that was not found:

```
OS9:bootmod -s cc3disk d0 aciapad t1 sio t2
aciapad t1 sio t2
ERROR - module not found in buffer
```

Also ignore the "@" character appended to the end of a drive name (eg. if you keyed "/DD0" by mistake, "/DD0@" will be displayed).

* Path name not found -- the specified path list cannot be found, perhaps due to a wrongly spelled name, or because the wrong disk was inserted.

* Bad name or syntax -- perhaps the first character of the name wasn't alphabetic, or a parameter or option began with a character other than a dash or forward slash, or an option other than -s or -? was specified, etc.

* File not accessible - check attributes -- either you didn't have permission to delete the specified file (if it was the "old" OS9Boot file), or you didn't have permission to read the file. For more explanation, refer to the ATTR command in the "OS-9 Commands" manual.

* Buffer too small to load the OS9Boot file -- the buffer was not big enough to hold the "old" OS9Boot file. Try giving BOOTMOD more memory (the Level II version defaults to #32K).

* OS9Boot file is fragmented - disk will not boot -- refer to the explanation in the Notes above.

* Buffer overflow or file too large -- there was not enough room left in the buffer to add the specified file. Try giving BOOTMOD more memory (the Level II version defaults to #32K). This message is also displayed if the file was too big (larger than 20,000 bytes).

* Module not found in buffer -- the module specified to be deleted from the "old" OS9Boot file (in the buffer) could not be found. Using the IDENT command on the OS9Boot file will tell you what modules are currently in the OS9Boot file.

* ERROR #nnn -- refer to the OS-9 manuals for an explanation.

7.3 CACHE (pronounced same as "cash") CACHE

Syntax: cache /devname [-n]

Function: Enables or disables disk caching for the specified drive, or tells the SDISK3 driver that a different disk has been inserted into a cached drive.

Parameters:

devname — The name of the disk drive which is to have caching enabled or has had a disk change. If caching is being disabled, this can be the name of any disk drive associated with the SDISK3 driver -- it doesn't have to be the name of the drive currently enabled. This parameter must always be specified.

-n — Disable caching for the specified drive. If this parameter is left out, caching will be enabled, or if caching is already enabled, the cache will be reset (erased).

Notes:

* WARNING -- DISK CACHING CAN DESTROY DATA ON YOUR DISKS unless you strictly follow the four rules presented here:
  a) Before enabling caching, make sure that the CACHE command can be run without accessing the drive to be cached. In other words, the CACHE program should be memory resident (LOAD'd into memory) or reside on a non-cached disk. This is to prevent you from getting into a catch-22 situation when you follow rule (b) below.
  b) If you want to put a different disk into a drive that has caching enabled, first make sure that that drive is not currently being accessed, then change the disk, then before anything else accesses that drive again, run CACHE to reset (erase) the cache for that drive. We will not be held responsible for any disks that you clobber by forgetting this rule.
  c) If at any time you can't remember if disk caching is enabled or not, or which drive has caching enabled, disable caching immediately, then re-enable it if desired.
  d) Before inserting a non-OS9 disk into a drive, make sure that caching is disabled for that drive. Do not enable caching for a drive containing a non-OS9 disk.

* The current version of SDISK3-DMC only allows you to have caching enabled for one drive at a time. If a drive has caching enabled, and you subsequently enable caching on a different drive, caching will automatically be disabled (and the cache erased) for the first drive. Only the drive most recently enabled will be cached.

* Several other programs also affect the status of disk caching:
  a) Caching is automatically disabled when TESTBUFR is run.
  b) Caching remains enabled, but the cache is reset (erased) whenever the "write track" function is used (by SFORMAT or other disk formatting programs), or whenever the "direct sector write" function is used (such as writing to an IBM-PC disk with PC-XFER or MSF).

* For more information on disk caching, refer to the "Theory of Operation" section later in this manual.

Examples:
---

1) OS9:cache /D0 [enables or resets caching on /D0]

2) OS9:cache /D1 -n [disables caching on all drives]

7.4 CMDGEN3 CMDGEN3

Syntax: cmdgen3 <command name>

Function: Creates an OS-9 module which can execute a program or series of programs, with run time parameter substitution. Also displays the command sequence contained in a generated command module.

Parameters:
---

command name — The name of the executable module to be generated or displayed. It is also the name of the file in the current execution directory containing that module. The name may be up to 29 characters long.

Notes:
---

* Shell procedure files can save you lots of time and keying effort whenever you need to repeatedly run the exact same complicated command or series of commands. But procedure files lose much of their value when the command sequences need to be slightly different each time they are run, as you either first have to use a text editor to modify the file before each run, or are forced to create many versions of that procedure. Also, procedure files are just ASCII text files, not OS-9 modules, so they can't be loaded into memory, but must be read from disk as they are being executed.

  CMDGEN3 solves these problems by allowing the changeable parts of the command sequences to be specified at run time. It also "encapsulates" the ASCII text inside a module so it can be made memory resident.

  CMDGEN3 creates "macro" commands. That is, it lets you quickly and easily build your own custom commands from one or many other existing commands. You don't have to be a programmer, because instead of using an assembler or high level language compiler, CMDGEN3 simply lets you key in the sequence of commands as you would if you were running them directly from the shell. New commands can be generated so quickly they can even be used for "throw away" commands that you generate, use, and discard, all in a few minutes.

Like regular OS-9 commands, your new command can accept parameters (up to three) which it in turn passes on to the commands it is composed of. This parameter substitution facility is especially useful when the same changeable value is used repeatedly within a command sequence, such as in example #5 below.

* To create a new macro command, run CMDGEN3 and specify the name of the command module to be created. The module will be stored in a file created in the current execution directory, so you may need to first run the CHX command. When you see the "Cmdgen3:" prompt, enter the command line (up to 150 characters long) that the generated command is to execute.

  This command line may include all the features that you use in a normal shell command line (ie. the line you key after the "OS9:" prompt), such as the ";" or "!" or "&" characters to put more than one command on one line. In addition, you can insert parameter variables that are later replaced with the user supplied values when the generated command is run.

  Up to three parameter variables can be specified. Wherever you will want the first run time parameter to appear, insert the parameter variable "%1". Do not use a space before the "%" or after the "1" unless you want the space there at run time. Similarly, parameter variables "%2" and "%3" are used for the second and third run time parameter values. Each parameter variable may appear more than once in the same line, and they may be placed in any sequence. Study the examples below.

  If the "%" character is immediately followed by any character other than the digits "1", "2", or "3", it will not be interpreted as a parameter value, but will be left in the command line unchanged.

* To run the generated command, type the command module name, followed by the run time parameter substitution values (up to three). These parameters are separated from each other by blanks, so may not contain embedded blanks. Most printable characters, including commas, are allowed in a parameter, except for the eight special characters ";!<>()#&" that the shell uses for itself. Refer to the examples below. There is no limitation on the length of each parameter except for the 200 total characters maximum described below.

  The expanded command line (ie. after parameter substitutions) will be displayed, but will not be executed until you press a key (any key). This gives you a chance to inspect the command line that will actually be executed, and let you abort it with the BREAK key if it's not what you wanted to do.

  The expanded command line (after parameter substitutions) may not be more than 200 characters long -- if it is, "ERROR #70" will be displayed and the command line will not be executed.

* If you can't remember what command sequences make up an existing generated command, there are two ways to display the embedded command line. One is to run CMDGEN3 and specify the command module name. CMDGEN3 will look for it in the current execution directory, so you may need to run the CHX command first. See

example #1 below. If the module is not a valid CMDGEN3 module, the message "**** Not a CMDGEN3 module!" will be displayed.

The other way is to run the generated command itself, but specify "%1 %2 %3" as the run time parameter line. When the expanded command line is displayed, it will look exactly like the original internal command line. Press the BREAK key to abort without executing it.

* Although CMDGEN3 was written by Sardis Technologies, it incorporates some code from Stephen Goldberg's original CMDGEN program, with his express permission. (See RAINBOW magazine May '88 pp. 185-189)

Examples:
---

1) The generated command module COPYSRC already exists, and we want to see what command line it contains, without executing it:

```
OS9:cmdgen3 copysrc
copy %1/source/%3 %2/source/%3 #24k
```

or

```
OS9:copysrc %1 %2 %3
copy %1/source/%3 %2/source/%3 #24k
<BREAK>
```

2) To create a command module that copies all files from one directory to another directory:

```
OS9:cmdgen3 copyall
Cmdgen3: chd %1;dsave -s16 %1 %2 #8 ! (chd %2)
```

3) This example runs the command module displayed in example #1 to copy file "junk.txt" from the SOURCE directory on drive /D1 to the SOURCE directory on /D0. Note that since you only have to type the file name once, and that it is at the end of the line you key, how easy it would be to use the control-A and backspace keys to repeat the same command for other file names:

```
OS9:copysrc /D1 /D0 junk.txt
copy /D1/source/junk.txt /D0/source/junk.txt #24k
```

4) To run the command module created in example #2 to copy all files from a disk in drive /D1 to a directory TEST on /R0:

```
OS9:copyall /D1 /R0/TEST
chd /D1;dsave -s16 /D1 /R0/TEST #8 ! (chd /R0/TEST)
```

5) To create a command sequence to edit a text file with first deleting the previous version (.BAK extension), renaming the current version (.TXT extension) to a .BAK extension, then editing the .BAK version into the new .TXT version:

```
OS9:cmdgen3 ed3
Cmdgen3: chd /D1/source;del %1.bak;rename %1.txt %1.bak;edit
 %1.bak %1.txt #16k
```

(cont'd)

To run this command, merely type:

```
OS9:ed3 essay
```

which automatically expands into a long command line that you no longer need to type in yourself:

```
chd /D1/source;del essay.bak;rename essay.txt essay.bak;edit
 essay.bak essay.txt #16k
```

7.5 DESCGENL2 (Level II only) DESCGENL2

Syntax: descgenL2 [-d=n]

Function: Creates the device descriptors (D0, D1, D2, DD) for use with the SDISK3 device driver.

Parameters:

-d=n Specifying this optional parameter will cause a descriptor named DD to be generated in addition to the other descriptors. It will be identical to the descriptor generated for drive "n", ie. for -d=0, descriptor DD will point to drive D0.

Notes:

* Before running DESCGENL2, run the CHD command to set the current work directory to the directory into which the new device descriptors are to be written.

* DESCGENL2 will prompt you for the number of drives to produce descriptors for (1, 2 or 3), and the characteristics of each. It then creates a descriptor for each drive 0..(n-1) where n is the number of drives. Each descriptor is written to a separate file named the same as the module (D0, D1, etc.), in the current work directory.

* If you will be configuring a 4 drive setup, the descriptor for the fourth drive (D3) cannot be generated by DESCGENL2. However, you can either use the "D3" descriptor that already exists in the "MODS" directory of the "SDISK3-DMC Level II RELEASE" disk, or use the source code supplied in the "SRC" library of that same disk, as a starting point to create your own.

* The three values that may be specified for each drive are:
  a) number of cylinders (tracks per side), usually 35, 40, or 80
  b) single vs double sided (1 or 2 sides)
  c) step rate (usually 6, 12, 20 or 30 milliseconds)

* DESCGENL2 creates descriptors for CoCo OS-9 disk format. DISKTYPE and/or DMODE can be used later to set descriptors to a different format. DMODE is especially useful, as it can update the descriptors directly on disk even before they are merged into the new OS9Boot file by OS9GEN, CONFIG, or BOOTMOD.

* The SDISK3 device driver must be used in conjunction with the device descriptors created by DESCGENL2 -- SDISK3 will NOT work with the device descriptors supplied by Radio Shack (they are missing the IT.STOff location that SDISK3 uses, and they have a different device driver name), nor with the device descriptors created by the Level I DESCGEN3 program (the device driver name is set to SDisk instead of SDisk3).

Examples:
---

1) The following sample session creates descriptors D0 and DD for drive 0:

```
                                                    (your response)
    OS9:descgenL2 -d=0 <enter>
    DescgenL2
    (c) Copyright 1987 D.P. Johnson
    Enter number of disk drives=1 <enter>              <- 1

    Number of cylinders on drive 0
    (tracks/side) =40 <enter>                          <- 40
    Number of sides on drive 0=2 <enter>               <- 2
    Step rate of drive 0
    (in milliseconds)=6 <enter>                        <- 6
```

2) The following command is identical to the one above, except that descriptor DD is not created:

```
    OS9:descgenL2 <enter>
         (etc.)
```

7.6 DISKTYPE

Syntax: disktype [opt] /device[...]

Function: Modifies a disk descriptor (in memory) for various OS-9 disk formats, or displays the current device setting.

Parameters:
---

device	The names of one or more disk device descriptors to modify or display.
opt	One of the options below. If no option is given, the current device setting is displayed without changing anything.

Options:

-c	Set for Color Computer OS-9 format.
-j	Set for Japanese OS-9 format (eg. Fujitsu).
-m	Set for Mizar OS-9 format.
-s	Set for Standard OS-9 format.

Notes:
---

* WARNING -- if you try to access a diskette via a device name that is currently set for a different format than the diskette was created in, you will usually get a SEEK ERROR. However, it is possible to get other unpredictable results, perhaps even destroying data on the disk if you are writing to it. That is why you should develop the habit of labelling each disk with its current format, so that you will know when DISKTYPE needs to be run.

* Once you set a descriptor for a particular disk format, it keeps that setting until you change it again (or until you re-boot).

* You will get the error message "unable to perform disktype" if you try to use this command on the device descriptors supplied with the Radio Shack OS-9 disk instead of on descriptors supplied or created with this SDISK3 package. This is because DISKTYPE requires the device descriptor to have the IT.STOff location, which the Radio Shack descriptors don't have.

* We recommend you use the CoCo OS-9 disk format for all your disks except those used to transfer information to/from another system that uses one of the other formats. First of all, it's less confusing to have most of your disk in the same format. Secondly, the CoCo OS-9 format has the most storage capacity of any of these formats.

* If you only have one disk drive, DISKTYPE should be LOAD'd into memory. Otherwise you will get into a Catch-22 situation once you have changed the drive setting to a format different from that of the disk on which the DISKTYPE command itself resides! A second solution would be to have two descriptors with different names and different formats (one of them CoCo format) for the same physical drive (eg. /D1 and /SD1), which was the method used by the original version of SDISK. The DD descriptor can often be used for this.

* The device descriptors D0 D1 D2 DD created by the DESCGENL2 command are initially set to the Color Computer OS-9 format.

* CoCo OS-9 format disks are written entirely in double-density with 18 sectors per track (each side). The sectors are numbered 1-18, regardless of which side they are on. For maximum compatibility with "stock" Radio Shack CoCo's, disks would be limited to one side, 35 tracks, 48 tpi. Otherwise, disks can be any combination of single or double sided, 48 or 96 tpi, and any number of tracks from 1 to 245.

* Standard OS-9 format 5 1/4" disks always have 10 single density sectors on side 0 of track 0, with all other sides and tracks containing either 10 single density or 16 double density sectors per track (each side). Sectors are numbered 0-9 or 0-15, regardless of which side they are on. For maximum compatibility with other systems, to avoid such problems as GIMIX vs SWTPc side flags (too complicated to explain here), disks should be limited to single sided, single density, 48 tpi, 35 tracks. Otherwise disks can be any combination of single or double sided, single or double density, 48 or 96 tpi, and any number of tracks from 1 to

245. A slight variation on the standard uses 18 sectors (numbered from 0-17) per track in double density.

* Mizar OS-9/68K format 5 1/4" disks are written entirely in double density, like the CoCo, but with 16 sectors per track (each side. The sectors are numbered 0-15, regardless of which side they are on. (The Mizar is a 68000 VME bus system that runs OS-9/68000.)

* The Japanese OS-9 (eg. Fujitsu) 5 1/4" formats have Logical Sector 0 beginning on track 1 instead of track 0, presumably because their boot software in ROM requires track 0 to be in non-OS9 format. They are written entirely in double density, with 16 sectors per track (each side). The sectors are numbered 1-16, regardless of which side they are on.

Examples:
---

1) The following session displays the current settings of d0 and d1 but doesn't change them:

```
OS9:disktype /d0 /d1 <enter>
d0 is rigged for Color Computer OS-9
d1 is rigged for Standard OS-9
```

2) The following command sets d1 to access the standard OS-9 format:

```
disktype -s /d1 <enter>
```

but then you can't access a CoCo OS-9 format disk in this drive again until you do:

```
disktype -c /d1 <enter>
```

Error Messages:
---

* UNABLE TO FERFORM DISKTYPE ON dn -- the device is not an RBF type device, or the descriptor does not contain IT.SToff (not SDISK3 compatible).

* dn is an unknown type -- the device cannot be identified as one of the four OS-9 formats that DISKTYPE supports.

* dn is Hard Disk.. Cannot modify -- DISKTYPE may only be used on device descriptors for floppy disks.

7.7 DMODE

Syntax: dmode [/<devname>] [-<filename>] [opts] [-?]

Function: Displays or changes the initialization table values in the device descriptor of an RBF-type device (floppy or hard drive) that describe such attributes as the number of tracks, stepping rate, etc. It gives you the same facility for modifying device descriptors for disk drives as XMODE gives you for other devices.

Parameters:

devname — The name of the disk drive device descriptor module (in memory) to be displayed or modified.

filename — The name of the file (on disk) containing the disk drive device descriptor module to be displayed or modified. If only a partial path name is given, the current data directory is used as a default.

opts — None, one, or more of the following options, separated by spaces or commas. If no options are specified, the current settings are displayed and the descriptor is not changed. If one or more options are specified, the descriptor is modified with those values. The option keywords must be in lowercase, and the option values must be 2 or 4 digit hexadecimal numbers, as represented by the "hh" or "hhhh".

-? — If this is the only parameter specified, a help message is displayed describing the options.

Options:

drv=hh — Drive number code (0 - n for each controller)

stp=hh — Stepping rate code: 00 = 30 msec. 01 = 20 msec. 02 = 12 msec. 03 = 6 msec.

typ=hh — Device type code:
bit 0 0 = 5" 1 = 8"
bit 5 0 = non-CoCo 1 = CoCo
bit 6 0 = standard 1 = non-standard
bit 7 0 = floppy 1 = hard drive

dns=hh — Device density code:
bit 0 0 = single dens. 1 = double
bit 1 0 = 48 tpi 1 = 96 tpi

cyl=hhhh — Number of cylinders (tracks on one side): eg. 0023 = 35 tracks, 0028 = 40, 0050 = 80

sid=hh — Number of sides (heads) -- 01 or 02 for floppies

vfy=hh — Automatic-verify-after-write flag zero = yes non-zero = no

tos=hhhh — Number of sectors per track on track 0, side 0 (0012 = 18)

sct=hhhh — Number of sectors per track (one side) on all other tracks

ilv=hh Interleaving factor (usually 02, 03, or 04 for floppies). Also refer to the "Theory of Operation" section later in this manual for a discussion on interleaving factors and SDISK3.

sas=hh Minimum number of sectors allocated per file segment

Notes:
---

* DMODE lets you update disk descriptors on the fly, without having to use DEBUG or MODPATCH (which are difficult to use on device descriptors) or having to run DESCGENL2 again and generating a new boot file (which is time consuming).

* DMODE does not entirely replace DISKTYPE, as the IT.STOff byte is not currently accessed by DMODE.

* The options above are also explained on pages 5.7 to 5.10 in the "OS-9 Technical Reference" manual that came with Radio Shack's OS-9 Level II package.

* If your disk drives and the disks you use are very reliable (ie. you almost never get read or write errors), you can dramatically speed up writes to your disks by disabling verifies (vfy=01). Just keep in mind that you do somewhat increase the risk of getting a read error in the future. On the other hand, my impression is that the IBM-PC world usually disables verify and does just fine.

  BACKUP and COPY will not run much faster with verify disabled as they already use special tricks to avoid the normal speed degradation caused by verifying each sector immediately after writing it.

* More information on this program can be extracted by listing the source code supplied on disk. Enjoy!

* DMODE was written by Kevin Darling, who has graciously given us permission to distribute it with our SDISK3 package. Thanks, Kevin. By the way, Kevin can probably most easily be reached via the Compuserve Information System, or on DELPHI.

Examples:
---

1) Running DMODE with no parameters will display a 3 line help message summarizing the general syntax to use:

```
OS9:dmode
```

2) To get a more detailed help display listing all the available options and how they are coded, type:

```
OS9:dmode -?
```


3) To display the current settings of drive /D0 without changing them:

```
OS9:dmode /d0
 drv=00 stp=01 typ=20 dns=01 cyl=0028 sid=02
 vfy=00 sct=0012 tos=0012 ilv=02 sas=08
```

4) This example changes some values in descriptor D2 in memory:

```
OS9:dmode /d2 stp=03,cyl=0050,sid=02,ilv=03
```

5) This example does the same as the previous one, except the descriptor for D2 is in a file on disk and is updated directly on disk instead of in memory:

```
OS9:dmode -/d1/mods/d2 stp=03 cyl=0050 sid=02 ilv=03
```

7.8 DPOKE

Syntax: dpoke <pathlist> <offset> <data>[...]

Function: Updates a portion of a file on disk with new values that are specified on the command line.

Parameters:

pathlist — The name of the disk file to be updated (patched).

offset — The 16 bit offset from the beginning of the file where the string of new data is to begin writing. Specified as a 4 digit hexadecimal number (leading zeros may be omitted).

data — A list of one to 100 8-bit values that are to be written to the file starting at the offset. Specified as 2 digit hexadecimal numbers (leading zeros may be omitted). The values are separated from each other by spaces or commas.

Notes:

* There are times when you need to patch a module before you load it into memory, especially if that module will be incorporated into the OS9BOOT file. OS-9 has several commands available to let you patch a module in memory (DEBUG and MODPATCH), display a module in memory (DEBUG), or display a module on disk (DUMP), but no facility to patch a module directly on disk. DPOKE provides this missing link.

  While there are more powerful disk patchers available from other sources (such as Doug DeMartinis' "dEd" on CompuServe), we needed a simple, non-interactive, command line oriented patcher for you to use in the SDISK3 installation procedure. That's why we wrote DPOKE.

* DPOKE writes the new data over the existing data in the file, but only those bytes specifically patched are altered; the rest of the file is not changed.

* After patching a module on disk, you must always update the header parity and CRC bytes using OS-9's VERIFY or our NEWCRC command.

* Don't use DPOKE unless you know what you are doing, and remember to have a backup copy of any module you patch in case you make a mistake. To check if the patch was made as desired, you can use DUMP to display the patched module, or use CMP to compare the patched version to the unmodified backup copy.

Examples:
---

1) The following example patches the SDISK3 module to change the motor-up-to-speed time delay to 64 ticks (just over 1 sec.):

```
OS9:dpoke /d1/mods/sdisk3.3dnc 0018 00 40
OS9:newcrc /d1/mods/sdisk3.3dnc
```

2) The following example uses DPOKE to change the volume name of a disk to "RT36" by directly patching the Identification Sector (LSN 0). Since this sector does not contain a module, NEWCRC should not be run afterwards.

```
OS9:dpoke /d0@ 01f 52 54 33 b6
```

7.9 NEWCRC NEWCRC

Syntax: newcrc <pathlist>

Function: Recalculates and updates the header parity and CRC bytes of
--- an OS-9 module in a file on disk.

Parameters:
---

pathlist The name of the file containing the module to be updated.

Notes:
---

* Unlike OS-9's VERIFY command, NEWCRC does not create a new file - the existing file is updated in place.

* If the file contains more than one module, only the first one is updated; the other modules are ignored.

Examples:
---

1) This example updates the header parity and CRC bytes of the SDISK3 module on disk:

```
OS9:newcrc /d1/mods/sdisk3.3dnc
```

7.10 SFORMAT

Syntax: sformat /devname [opts]

Function: SFORMAT is a direct replacement for the FORMAT command supplied with Radio Shack's OS-9, except that SFORMAT can format disks in virtually any kind of OS-9 disk format, not just CoCo OS-9.

Parameters:

devname — The name of the drive containing the disk to be formatted.

opts — None, one, or more of the following options, separated by spaces. The specified option overrides the corresponding default value in the device descriptor.

Options:

```
S = Single density (valid for Standard OS-9 formats only)
D = Double density
R = Ready          (proceed immediately with formatting)
1 = 1 side
2 = 2 sides
4 = 48 tpi         (to format disk at 48 tpi on 96 tpi drive)
"disk name"
'no. of cylinders'
:interleave:
```

Notes:

* SFORMAT when used with SDISK3 installed allows formatting of virtually all kinds of OS-9 disk formats, including CoCo OS-9 disks with one or two sides and any number of tracks up to the capacity of the drive. If the device descriptor for the drive containing the disk to be formatted is set (via DISKTYPE) to Standard OS-9 format it will also format standard single and double density disks to allow easy interchanging of data to other OS-9 systems.

* SFORMAT begins by displaying a list of parameters it will use in the formatting process for the diskette in the specified drive, then waits for your response to either quit the program without formatting, begin the formatting process, or first change some parameters.

* The format parameters displayed are based on the drive capabilities and other default values defined in the device descriptor, as modified by any override options specified on the command line when SFORMAT is called, or specified after responding with "N" to the "Ready?" prompt.

* The "R" option suppresses the parameter display and immediately begins the format operation. Since it doesn't give you a chance to double-check what values will be used, we recommend you avoid using it. The "R" option is most useful if you are formatting several identical disks, or are running SFORMAT from a procedure (shell script).

* If you don't supply a diskette volume name as an option on the command line, SFORMAT will prompt you for one later. The name can be from one to 32 characters long, and may include spaces or punctuation.

* The diskette to be formatted must NOT be write protected. If you attempt to format a write protected disk, you will get an "ERROR #245" write error message. The system will return to the "OS9:" prompt without formatting the disk.

* The formatting process has three phases:
  a) The diskette is initialized by writing marks to divide it into several tracks, and each track into several sectors. You can think of it as similar to setting up rows of storage racks in an empty warehouse, then putting many parts bins onto each rack. Note -- in doing so, all data previously stored on that disk is erased and can NEVER be recovered.
  b) An attempt is made to read back each sector to determine if it is usable, or defective (like a piece of paper with a grease spot on it that won't let a pen write on it). Defective sectors are deleted from the list of "free" (ie. available) sectors, and no attempt will be made to use them in the future. Note -- if any sectors are defective, the BACKUP command cannot be used with that disk; refer to the note below.
  c) SFORMAT writes the identification sector, disk allocation map, and root directory to the first 3 or 4 sectors of track zero. These sectors must not be defective. For more information on the contents of these special sectors, refer to the "OS-9 Technical Reference" manual.

* If the newly formatted disk will not be used to "boot" from, it is now ready to use. Otherwise you will need to use either COBBLER, OS9GEN, CONFIG, and/or BOOTMOD to make the disk bootable. Note that in order for the CoCo to be able to boot from the disk, it must be in CoCo OS-9 format, and have at least 35 tracks (because the OS-9 kernel is written to track 34).

* To format a disk in non-CoCo OS-9 format, first use the DISKTYPE command to change the setting of the device descriptor of the drive in which you will be formatting the disk. IMMEDIATELY AFTER FORMATTING YOU SHOULD MAKE A NOTATION ON THE DISK'S LABEL TO INDICATE WHICH FORMAT IT USES. This habit will save you lots of guess-work and grief as time goes by.

* If you are using the "4" option to format a disk at 48 tpi on a 96 tpi disk drive, you should first erase the disk with a bulk tape eraser (see warning below). This is especially necessary if the disk has previously been formatted without the "4" option. If you don't bulk erase, there may be enough old data left between the tracks to make the disk unreadable in a 48 tpi disk drive.

* If SFORMAT reports one or more defective sectors, try formatting the disk again, as the "defects" are sometimes only temporary. If they persist, try erasing the disk with a bulk tape eraser before formatting. WARNING -- bulk tape erasers contain a very powerful magnet which can wipe out all data on a disk in a split second, so only use the eraser at a safe distance from all other disks.

Example:

```
OS9:SFORMAT /D1 1 '35' :3:
*** STANDARD DISK FORMAT ***
(C) Copyright 1983 D.P. Johnson
ALL RIGHTS RESERVED

FORMAT PARAMETERS:

   Double Density
 35 Cylinders
  1 Sides
   Color Computer format
 18 Trk 0 Sectors
 18 Sectors/Track

Formatting drive /D1
y (yes), n (no), or q (quit)
Ready? Y
Volume Name=
JUNK COLLECTION
Verifying tracks:
  34
  630 Good Sectors
OS9:
```

7.11 TESTBUFR

Syntax: testbufr [-8] [-32] [/devname]

Function: Tests the static RAM chip on the DMC controller

Parameters:

-8 / -32	The capacity (8K or 32K bytes) of the static RAM chip installed in the DMC controller. Defaults to 32K if not specified.
devname	The name of any disk drive associated with the SDISK3 device driver. Defaults to /D0 if not specified.

Notes:
---

* WARNING -- DO NOT RUN ANY OTHER PROGRAM THAT ACCESSES A FLOPPY DISK DRIVE WHILE TESTBUFR IS RUNNING.

* This diagnostic program performs 31 complete passes on the entire 8K or 32K bytes of the RAM, except when it discovers an error, in which case it aborts at the end of the current pass. The message "Cache RAM tested OK" indicates the test was completed with no errors. At 1.79 MHz the test takes 5 seconds for 8K or 20 sec. for 32K -- multiply these times by 2 for .89 MHz.

* If any errors are discovered, TESTBUFR displays for each the address of the error in the static RAM, the test pattern written to that location, and the pattern read back. These values are all in hexadecimal. If more than 17 errors are detected, only the first 17 are displayed, followed by "etc." and a count (in decimal) of the total number of errors in this pass. One or more errors indicate that something is wrong with either the RAM chip, or with its support chips. Refer to the trouble shooting section in the DMC hardware manual for more details.

* If disk caching was active when TESTBUFR is called, the caching is disabled. If you run TESTBUFR with the Radio Shack CC3Disk driver instead of SDisk3, ignore the "Warning - unable to disable caching" message.

Examples:
---

1) To test an 8K RAM chip if /D0 is assigned to a hard disk instead of a floppy drive, but /D1 is a floppy drive:

```
OS9:TESTBUFR -8 /D1
Cache RAM tested OK
```

2) In this example two of the 32K bytes tested bad:

```
OS9:TESTBUFR
ADDR-0148 W-16 R-12;  ADDR-0248 W-C0 R-FF;
Errors in cache RAM = 2
```

8.0 CONTENTS OF THE "SDISK3-DMC LEVEL II RELEASE" DISKETTE

Root directory:

CMDS	MODS	SRC	PATCHES	copycmds1
copycmds2				

CMDS directory:

allowsleep	bootmod	cache	cmdgen3	descgenL2
disktype	dmode	dpoke	newcrc	sformat
testbufr				

MODS directory:

d3	sdisk3.3dnc	sdisk3.4snc	sdisk3.4dnc	sdisk3.3dvs4
sdisk3.3dis4				

PATCHES directory:

InitIrq.F.pat	SlowCPUboot.pat	FastStepBoot.pat
W7term.pat		

SRC directory:

d3.asm	dmode.asm

The "copycmds" files in the root directory are shell script files (procedures) used in the installation process.

The CMDS directory contains executable programs. They are explained elsewhere in this manual. You will want to copy all or most of them to the CMDS directory on your working system disk.

The MODS directory contains several versions of the SDISK3 driver module which replaces the CC3Disk module supplied by Radio Shack. Brief descriptions of the differences among the versions are shown below. Most of you will only need to use the .3dnc (or .4snc) driver and can ignore the other versions. The "no slot switching, any slot OK" only means that the SDISK3 driver itself doesn't use slot switching. These drivers will work OK when used with other drivers that use slot switching (such as the ACIAPAK or Burke and Burke hard disk adapter), but the DMC controller should then only be used in slot 4. The D3 device descriptor is supplied here because DESCGENL2 cannot create one.

- .3dnc 3 double sided drives, NAP-type driver, no slot switching, any slot OK
- .4snc 4 single sided drives, NAP-type driver, no slot switching, any slot OK
- .4dnc 4 double sided drives (requires hardware modification to DMC), NAP-type driver, no slot switching, any slot OK
- .3dvs4 3 double sided drives, VIRQ-type driver, slot switching, slot 4 default,
- .3dis4 3 double sided drives, IRQ-type driver, slot switching, slot 4 default, requires modification to CoCo itself to generate IRQ interrupts from DMC

The PATCHES directory contains patch files that are used with the MODPATCH utility. They are largely self-documented, and can be read with the LIST command.

The SRC directory contains source code for device descriptor D3, as well for the DMODE program.

9.0 SDISK3 GETSTT/SETSTT FUNCTION CALLS GETSTT/SETSTT

The following SETSTT function calls are supported and function as described in the Radio Shack "OS-9 Technical Reference" manual under "Set Status System Calls" beginning on page 8.131.

```
I$SetStt $03 SS.RESET  restores head to track 0
I$SetStt $04 SS.WTRK   formats a track
```

The following new GETSTT/SETSTT function calls have also been implemented and are explained in detail in the pages below:

```
I$SetStt $0A SS.FRZ    freezes DD.xxxx information
I$GetStt $80 SS.DREAD  direct sector read;  can be used to read
                        non-OS9 disks
I$SetStt $80 SS.DWRIT  direct sector write;  can be used to write
                        non-OS9 disks
I$SetStt $81 SS.UNFRZ  unfreezes updating of DD.xxxx information
I$SetStt $82 SS.MOFF   quick drive motor shutoff
I$SetStt $83 SS.MOTIM  change drive motor time-on constant
I$GetStt $84 SS.SDRD   same as SS.DREAD, except sector buffer is in
                        system memory map instead of user map
I$SetStt $84 SS.SDWRT  same as SS.DWRIT, except sector buffer is in
                        system memory map instead of user map
I$GetStt $86 SS.DRVCH  indicates which drive, if any, currently has
                        caching enabled
I$SetStt $85 SS.SLEEP  activates or deactivates use of F$Sleep
I$SetStt $86 SS.DRVCH  activates or deactivates disk caching
```

NOTE - regarding the "logical track no." vs "physical track no." terminology used below -- logical track numbers are identical to physical track numbers unless you have a 48 tpi diskette in a 96 tpi drive. In that case, every other physical track is skipped, so logical track numbers are multiplied by 2 to convert them to physical track numbers. For example, logical track 7 on the diskette would automatically be converted to physical track 14 for the drive. This is often referred to as "double stepping".

The SS.FRZ call does not need to be used before accessing the direct sector read/write calls, since the updating of the DD.xxxx information from LSN 0 is bypassed when using the direct sector read/write calls.

You should only specify sector sizes of 128, 256, 512, or 1024 bytes for direct sector read/write calls, as the disk controller chip supports only these. Any other value can cause problems. If all the "most/least significant bits of Register Y" verbiage below confuses you, here is some sample code you can use as a guide to set up the sector size in the Y register for the SS.DREAD/SS.SDRD/SS.DWRIT/ SS.SDWRT calls:

```
            LDD   #512   sector size = 512  (or 128 or 256 or 1024)
            EXG   A,B
            ASLB
            ASLB
            ASLB
            ASLB
            ORB   #$02   bits 0-2 = 48 tpi / double density / side 0
            TFR   D,Y
```

SS.DREAD (Function code $80) Direct sector Read function reads a specified sector into a user buffer. Sector lengths of 128, 256, 512 and 1024 bytes are supported for either single or double density. Non-OS9 disks (such as MS-DOS, CP/M, or FLEX) can be read with this function. Note -- some other versions of SDISK do not support a sector length of 1024 for double density, and only allow lengths of 128 and 256 for single-density. Most do not implement the retry disable feature (Reg Y bit 7) either.

Entry Conditions:

A = path number
B = $80 function code
U = (msb) logical track # / (lsb) physical sector #
X = address of user buffer (in user map) to read data into
Y = sector size / format codes

Y-REGISTER CONTENTS															
Y15	Y14	Y13	Y12	Y11	Y10	Y09	Y08	Y07	Y06	Y05	Y04	Y03	Y02	Y01	Y00
lsb sector size									msb sec siz	x					
S07	S06	S05	S04	S03	S02	S01	S00	rty	S10	S09	S08	-	tpi	dns	sid

(Y contains the following:)
bits 8-15 = least significant 8 bits of 11 bit sector size in bytes
bits 4-6 = most significant 3 bits of 11 bit sector size with bit 6 being the most significant bit of the 11 bit number
bit 7 = retry (0 = normal retry, 1 = no retry)
bit 3 = (not used, reserved for Hi-density) - set to 0
bit 2 = tpi of data on diskette (0=48 tpi, 1=96 tpi)
bit 1 = density of data on diskette (0=single, 1=double)
bit 0 = side (0 or 1)

Exit conditions:
The buffer pointed to by Reg. X contains the data read from disk
If error:
CC = C bit set
B = error code

SS.SDRD (Function code $84) System Direct Read function -- same registers used and same function as SS.DREAD except that the buffer address in Register X is in the system memory map instead of the user memory map. This call is not available in some other versions of SDISK.

SS.DWRIT (Function code $80) Direct sector Write function writes a specified sector from a user buffer. Sector lengths of 128, 256, 512 and 1024 bytes are supported for either single or double density. Non-OS9 disks (such as MS-DOS, CP/M, or FLEX) can be written with this function. Note -- some other versions of SDISK only support sector lengths of 128 and 256 for single density. Most do not implement the retry disable feature (Reg Y bit 7) either.

Entry Conditions:

A = path number
B = $80 function code
U = (msb)logical track # / (lsb) physical sector #
X = address of user buffer (in user map) to write data from
Y = sector size / format codes

Y-REGISTER CONTENTS															
Y15	Y14	Y13	Y12	Y11	Y10	Y09	Y08	Y07	Y06	Y05	Y04	Y03	Y02	Y01	Y00
lsb sector size									msb sec siz	x					
S07	S06	S05	S04	S03	S02	S01	S00	rty	S10	S09	S08	-	tpi	dns	sid

(Y contains the following:)
bits 8-15 = least significant 8 bits of 11 bit sector size in bytes
bits 4-6 = most significant 3 bits of 11 bit sector size with bit 6 being the most significant bit of the 11 bit number
bit 7 = retry (0 = normal retry, 1 = no retry)
bit 3 = (not used, reserved for Hi-density) - set to 0
bit 2 = tpi of data on diskette (0=48 tpi, 1=96 tpi)
bit 1 = density of data on diskette (0=single, 1=double)
bit 0 = side (0 or 1)

Exit conditions:
none
If error:
CC = C bit set
B = error code

Note - the automatic verify-after-write function will work for all four sector sizes from 128 to 1024 bytes in this version of SDISK. Other versions of SDISK may require the verify option to be switched off while writing sector sizes other than 256 bytes. This can be done by opening a path to the device and using the SS.OPT ($00) I$GETSTT and I$SETSTT I/O System calls to change the PD.VFY byte before using the SS.DWRIT call.

SS.SDWRT (Function code $84) System Direct Write function -- same registers used and same function as SS.DWRIT except that the buffer address in Register X is in the system memory map instead of the user memory map. This call is not available in some other versions of SDISK.

SS.FRZ (Function code $0A) Freeze DD.xxxx information -- inhibits the reading of the disk's identification sector (LSN 0) to the DD.xxxx variables in the drive tables (in the device driver's static storage) that define the disk formats, so that non-standard disks may be read. The availability of the SDISK direct sector read/write calls that neither use nor update the DD.xxxx variables makes the SS.FRZ call generally unnecessary.

The effect of the SS.FRZ call affects <u>all</u> drives controlled by the same device driver. In this version of SDISK the SS.FRZ call remains in effect only until immediately after the next read of LSN 0 on <u>any</u> drive. As a result, the SS.UNFRZ call is rarely needed. In some other versions of SDISK, the SS.FRZ call remains in effect until the SS.UNFRZ call explicitly resets it.

Entry Conditions:
A = path number (may point to any drive controlled by SDISK3)
B = $0A function code

Exit Conditions:
none
If error:
CC = C bit set
B = error code

SS.UNFRZ (Function code $81) Unfreeze DD.xxxx information -- reactivate the reading of LSN 0 information to DD.xxxx variables after the SS.FRZ call has shut it off. In this version of SDISK the SS.UNFRZ call is rarely needed. Refer to the SS.FRZ writeup above for details.

Entry Conditions:
A = path number (may point to any drive controlled by SDISK3)
B = $81 function code

Exit Conditions:
none
If error:
CC = C bit set
B = error code

SS.MOFF (Function $82) Quick drive motor shutoff. Turns the disk drive motors off without waiting for the normal time delay after last use. This call is not available in some other versions of SDISK.

Entry Conditions:
A = path number (may point to any drive controlled by SDISK3)
B = $82 function code

Exit Conditions:
none
If error:
CC = C bit set
B = error code

SS.MOTIM (Function $83) Set drive Motor Time on constant. Allows changing the constant that determines how long the drive motors remain on after the last access to any disk before turning them off. When the motors are off there is a half second delay to wait for the motors to come up to speed before the disk read or write can take place. If the motor time constant is just a little too short to keep the motors running between operations (say a directory listing followed by a copy), then the motor startup delay each time will add to the overall operating delay. Keeping the motors running constantly, however, adds needlessly to the diskette wear. This call is not available in some other versions of SDISK.

Entry Conditions:
A = path number (may point to any drive controlled by SDISK3)
B = $83 function code
X = time constant in clock ticks (1/60 second) 1-65535

Exit Conditions:
none
If error:
CC = C bit set
B = error code

SS.SLEEP (Function code $85) Activates or deactivates the use of F$Sleep by SDISK3 for disk I/O operations (read/write/seek). Non I/O operations (motor-up-to-speed and head settle delays) are not affected by this function. This call is provided mainly for internal use by our ALLOWSLEEP command, which is described earlier in this manual. The SS.SLEEP call is only available in Sardis Technologies' versions of SDISK for the DMC disk controller.

Entry Conditions:
A = path number (may point to any drive controlled by SDISK3)
B = $85 function code
X = 0 to disable F$Sleep and use software delay loops instead;
non-zero to enable use of F$Sleep

Exit Conditions:
none
If error:
CC = C bit set
B = error code

SS.DRVCH (Function code $86 - I$GetStt) Retrieves the drive number of the drive that currently has caching enabled. The SS.DRVCH call is only available in Sardis Technologies' versions of SDISK for the DMC disk controller.

Entry Conditions:
A = path number (may point to any drive controlled by SDISK3)
B = $86 fnction code

Exit Conditions:
A = drive number that has caching enabled (as per IT.DRV in device descriptor). $FF = no drive is currently cached.
If error:
CC = C bit set
B = error code

SS.DRVCH (Function code $86 - I$SetStt) Activates or deactivates disk caching for a particular drive. This call is provided mainly for internal use by our CACHE and TESTBUFR commands. For more details, refer to the writeup on the CACHE command earlier in this manual, and the "Theory of Operation" section later in this manual. The SS.DRVCH call is only available in Sardis Technologies' versions of SDISK for the DMC disk controller.

Entry Conditions:
A = path number (may point to any drive controlled by SDISK3)
B = $86 function code
X = 0 to disable caching for this drive;
non-zero to enable caching for this drive

Exit Conditions:
  none
If error:
  CC = C bit set
  B = error code

10.0 CHANGING USER VALUES IN THE SDISK3 DRIVER

Over a dozen values in the SDISK3 driver module have been placed in fixed locations and documented below so you can easily change them. Most of you will have no need to touch these values and can safely ignore this whole section. But for those of you who have an uncontrollable urge to tinker to squeeze out maximum performance, or have a non-standard hardware configuration, here is the road map you need. Enjoy!

You can either use OS-9's MODPATCH utility, the OS-9 debugger, our DPOKE program, or one of the many third party patch programs to modify these values, either in memory, or directly on disk. An excellent full-screen oriented direct-on-disk editor is Doug DeMartinis' "dEd", available on the Compuserve and Delphi information systems. Remember to update the SDISK3 module's CRC after changing any values.

Note -- if you ever change the 6809 CPU speed by writing to $FFD8/D9, make sure you also update the D.Speed direct page location ($00A0 in the system map) to match. SDISK3 uses D.Speed ($00 = .89 MHz; $01 = 1.79 MHz) to adjust some of its internal timing loops.

off-set	len	current-value (hex)	(decml)	unit	description
$16	2	$00F2	240+2	ticks	drive motor-on time (changing this
			4.0	sec.	has no effect until you reboot)
$18	2	$002F	45+2	ticks	motor startup delay (up to speed)
			0.75	sec.	
$1A	2	$0003	1+2	ticks	head settle time (after seek)
			17+	msec.	
$1C	2	$0100	256	bytes	default sector size for direct R/W
$1E	1	$1E	30	67 usec	delay-after-write before allowing
			2	msec.	drive to be deselected
$20	2	$23,$16	35,22	track #	write precomp table - 35T @22
$22	2	$28,$28	40,40		- 40T @40 (none)
$24	2	$00,$2B	0,43		- other @ 43
					($24 must always be 0)
$26	1	$01			drive select code - drive 0
$27	1	$02			- drive 1
$28	1	$04			- drive 2
$29	1	$40			- drive 3
$2C	2	$009E	156+2	ticks	timeout value for restore to track 0
			2.6	sec.	
$31	2	$0005	3+2	ticks	timeout value for step in one track
			50	msec.	
$36	2	$0098	150+2	ticks	timeout value for seek
			2.5	sec.	

off-set	len	current-value (hex)	(decml)	unit	description
$3B	2	$005C	90+2 1.5	ticks sec.	timeout value for read sector
$40	2	$005C	90+2 1.5	ticks sec.	timeout value for write sector
$45	2	$0020	30+2 0.5	ticks sec.	timeout value for write track
$48	1	$03	3		Multi-Pak Interface code for DMC slot ($00-$03 = slots 1-4)
$4B	1	$0A	10		interrupt priority code for motor timer (no effect until reboot)
$4E	1	$14	20		interrupt priority code (see below)

Note - one tick is 1/60 second. But why is the "+2" added to all the tick values above? Because F$Sleep always subtracts 1 from the tick value, plus the next clock interrupt is always less than 1 tick away, etc.

Many of the newer floppy drives take less than 1/2 second to come up to speed after the motor is turned on (see offset $18), but some drives take a full second or more. Also, some drives ignore any stepping pulses sent to them while they are not yet up to speed -- if you have any of these drives, it is important that you set the motor startup delay to a sufficiently high value.

The default sector size (offset $1C) is explained in the section "Using PC-XFER".

The write precompensation table tells the SDISK3 driver at which track (and above) to activate write precompensation for various types of drives. The third entry is a catch-all entry.

You shouldn't need to change the drive select codes ($26-$29) unless you have made a hardware modification that lets you use 4 double sided drives, or if you are simulating 2 single sided drives with one double sided drive (front and back sides each treated as if they were a separate drive).

The interrupt priority code at $4E is used for different purposes depending on the type of driver. The IRQ-type drivers use it for the IRQ hardware interrupt caused by completion of disk I/O operations; a new value goes into effect only after rebooting, or if sleeping is turned off, then on again. The VIRQ-type drivers use it for the VIRQ generated every clock tick that polls the controller to see if the I/O operation is completed yet. The NAF-type drivers use it for the VIRQ generated only if the operation times out before being completed. A new value for $4E for either the VIRQ or NAF drivers only goes into effect after rebooting.

11.0 USING PC-XFER

The PC-XFER utilities package, available separately from D.P. Johnson, lets you read/write/format IBM-PC and MS-DOS disks under OS-9, and also read and write Radio Shack Disk BASIC disks. (Note -- another approach to reading and writing MS-DOS disks under OS-9 Level II is taken by Clearbrook Software Group with their MSF file manager -- refer to their ads in RAINBOW and '68' Micro Journal magazines.)

The original version of PC-XFER does not completely follow the current SDISK parameters for direct sector read and write (refer to section 9) because it does not specify the sector size in register Y when calling the SDISK routines. This parameter is not needed for double density reads and writes by some versions of SDISK, but is normally required by the SDISK3-DMC version.

To get around this problem, the DMC version of SDISK3 contains a 2 byte constant called "SECSIZ" located at an offset of $001C from the beginning of the SDISK3 module. When the sector size parameter of the direct read/write calls is left out (ie. is zero), SDISK3 will use the default value specified in SECSIZ.

Before using PC-XFER to read and write MS-DOS disks or Radio Shack Disk BASIC disks, run the following commands to set SECSIZ to $0200 (512) or $0100 (256):

For MS-DOS disks:

```
OS9:MODPATCH
L SDISK3
C 001C 01 02
V
<CTRL-BREAK>
```

For RS Disk BASIC disks:

```
OS9:MODPATCH
L SDISK3
C 001C 02 01
V
<CTRL-BREAK>
```

If you get a "*** byte does not match ***" error message, it means that the current contents of that location are not as specified; maybe they are already set to the desired value?

Note -- Edition #1 of PC-XFER doesn't work properly under OS-9 Level II, while edition #2 might after being patched. How do you know which edition your copy is? Run the IDENT command (eg. IDENT /D1/CMDS/PCRead). The necessary patches are applied as follows, after PCRead has been LOAD'd into memory:

```
MODPATCH
L PCREAD
C F49 B7 12
C F4A 00 12
C F4B 6F 12
C 10CD B7 12
C 10CE 00 12
C 10CF 6F 12
V
```

```
MODPATCH
L PCREAD
C 59D B7 12
C 59E 00 12
C 59F 6F 12
C E22 B7 12
C E23 00 12
C E24 6F 12
C E93 B7 12
C E94 00 12
C E95 6F 12
V
```

However, D.P. Johnson now has a newer edition out which runs properly under both OS-9 Level I and Level II -- if you have an older edition we strongly suggest you get it upgraded.

12.0 TROUBLE-SHOOTING

If you have difficulty reading the diskette we supplied, try it in more than one drive, if you have more than one. If that doesn't work, use the test program supplied with the Radio Shack OS-9 BOOT/CONFIG diskette to check your drives' rotational speed. If the diskette has been exposed to temperature and humidity extremes, it may need to sit for a day in your environment after you receive it to achieve dimensional stability. Another factor affecting diskette compatibility is the track alignment on the disk drives (yours and ours). If either one of them is off, a disk created on one system becomes unreadable in the other system. If after these attempts you still cannot read the disk, return it for a replacement.

If you have problems booting OS-9 from the customized "SDISK3-DMC Boot" disk you created in the "Installation Instructions" section, try going through the whole sequence again, very carefully. Perhaps the first time through you made a small typing mistake, or didn't understand our instructions completely. If you still have problems, call or write us, describing the symptoms, and your hardware configuration, as completely and precisely as possible. In some cases we will ask you to send us a copy of your customized boot disk so we can determine whether the software is configured properly, or if there might be a hardware problem.

13.0 THEORY OF OPERATION

14.0 WARRANTY

15.0 INDEX

