ST-2900 OS-9 CONVERSION PACKAGE

User's Manual

Copyright 1984 by Sardis Technologies
All rights reserved

Sardis Technologies
2261 East 11th Avenue
Vancouver, B.C.
Canada V5N 1Z7

CREDITS

Special thanks go to John C., Will W., and Greg M. -- many of the improvements in this release are due to their suggestions.

TRADEMARKS

"Radio Shack" is a trademark of Tandy Corp.
"OS-9" is a trademark of Microware and Motorola.
"UNIX" is a trademark of AT&T Bell Laboratories
"FLEX" is a trademark of Technical Systems Consultants (TSC)
"Screditor III" is a trademark of Alford and Associates
"CP/M" is a trademark of Digital Research Inc.
"MS-DOS" is a trademark of Microsoft Corp.

COPYRIGHT INFORMATION

The entire contents of this manual and all information on the supplied diskette(s) are copyrighted by Sardis Technologies. It has been sold to you on a "single end user" basis. It is permissible to make copies of this manual and the disk data only for use within a single site. However, if it becomes necessary to run the programs on more than one computer simultaneously, additional copies or a multi-copy license must be purchased from the supplier.

DISCLAIMER

Although much effort has been made to ensure the accuracy of the software and documentation, Sardis Technologies 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 August 5, 1985.

0.0 Table of Contents

1.0 Introduction

Welcome to the exciting world of OS-9! This sophisticated operating system offers many of the features of UNIX, yet is relatively simple to use and runs on very economical hardware such as the ST-2900.

The ST-2900 OS-9 Conversion Package lets you easily create an OS-9 system disk, fully configured for the ST-2900, directly bootable by ST-MON. The whole process only takes a few minutes, with the Conversion package doing most of the work for you.

Why, you might ask, does Sardis Technologies offer a conversion package to be used with another manufacturer's version of OS-9, instead of selling a complete, pre-configured implementation? To save <u>you</u> a bundle of money!! The Radio Shack Color Computer (CoCo) version of OS-9 is priced so attractively, that even after adding the cost of the ST-2900 OS-9 Conversion Package, OS-9 on the ST-2900 costs you less than half of Microware's suggested list price of ($US) $250.

No programming is involved -- merely follow the simple instructions in the next several pages. The task of replacing the CoCo "device drivers" with new ST-2900 drivers is done automatically by the Conversion Package.

Many of the limitations of OS-9 on the CoCo have been eliminated by means of the new ST-2900 device drivers in conjunction with the ST-2900's DUART and different disk controller design.

OS-9 on the ST-2900 system gives you more free memory for user programs than many other OS-9 Level I systems. Memory is especially tight on Radio Shack CoCo's with 24x51 high resolution displays. But running MFREE on the ST-2900 (immediately after booting up) typically indicates 172 pages (43K) of contiguous free memory.

Another feature of OS-9 on the ST-2900 is its ability to read and write disks in a variety of formats:

a) standard OS-9, single or double density
b) CoCo OS-9
c) MIZAR OS-9/68K
d) FLEX (using SouthEast Media's O-F package)
e) MS-DOS (using D.P. Johnson's PC-XFER utilities)
f) Radio Shack CoCo Disk BASIC (using D.P. Johnson's PC-XFER utilities)

2.0 What you need to run OS-9 on the ST-2900

2.1) The "ST-2900 OS-9 Conversion Package" from Sardis Technologies. A diskette labelled "ST-2900 OS-9 Conversion Boot Disk" is included in this package, and will be referred to as the "Conversion/Boot" disk throughout this manual. Appendix C lists the contents of the disk.

2.2) A copy of OS-9 Level I, including a system disk and manuals. Your local Radio Shack Computer Center or other Radio Shack stores (found in 80 countries worldwide) are the best source. The versions of OS-9 that have so far been verified to be compatible with the ST-2900 Conversion package are:

a) Radio Shack Color Computer OS-9 Version 01.00.00
b) Radio Shack Color Computer OS-9 Version 01.01.00

Some other versions might also run without changes. If your version of OS-9 has problems running on the ST-2900 system, contact us and we will try to help (but no guarantees). The important points are that the OS-9 disk used in the booting process must be in CoCo format, single-sided, and have the OS-9 kernel (OS9, OS9P2, INIT, BOOT) on track 34, sectors 1-15.

The ST-2900 OS-9 Conversion package supplies its own console, printer, clock, and disk driver routines, so it doesn't matter if the OS-9 version you choose didn't originally support the disk configuration you need.

The CoCo OS-9 package includes a disk labelled "Radio Shack Color Computer OS-9 System Master" which will be referred to as the "CoCo/System" disk throughout this manual.

2.3) ST-MON version 2.04, or later, installed on the CPU board. The number of data bits (7 or 8) that serial port A is set to must match that of your terminal, otherwise OS-9 will either hang up or you will get lots of "ERROR #244" messages. Refer to the ST-MON manual for more details.

2.4) At least one floppy disk drive. Although OS-9 will run on a one drive system, a two drive configuration is recommended, with a maximum of four drives allowed on the ST-2900. Running commands such as BACKUP is awkward and slow on a one drive system.

Disk drives may be single or double sided, 5 1/4" or 3 1/2", 35 or 40 or 80 track, 48 or 96 or 135 tpi, in any mix. NOTE -- because both the Conversion/Boot and CoCo/System disks are supplied on 5 1/4" media, you must have at least one 5 1/4" drive connected, even if only temporarily. The system will run fine with only 3 1/2" drives, once a configured 3 1/2" bootable system disk has been created.

2.5) The ST-2900 CPU board must have 64K RAM installed, and crystals Y1 and Y2 must be 3.6864 MHz and 16 MHz, respectively. The ST-2900 FDC board must be connected to the CPU board. The 6522 VIA does not need to be installed on the FDC board unless you will be using the VIA device driver.

3.0 Before you get started

3.1) Before attempting to use the ST-2900 OS-9 Conversion Package you should read the red "OS-9 COMMANDS" manual supplied with the Radio Shack version of OS-9 to familiarize yourself with the terminology and basic features of OS-9. ALSO READ THIS ENTIRE ST-2900 MANUAL THROUGH ONCE OR TWICE BEFORE BEGINNING ANY OF THE PROCEDURES GIVEN.

The next 6 or 7 pages contain the detailed instructions on booting OS-9 on the ST-2900, making backups of the original Sardis Technologies and Radio Shack disks, and creating a configured bootable system disk. Here is a brief overview of those instructions:

a) use ST-MON to set flags to indicated the drive number and track density of the boot drive, if these are different from the defaults
b) use ST-MON's "D OC" command to load the conversion package from disk
c) use the program loaded in (b) to load the Radio Shack CoCo version of the OS-9 operating system from disk
d) modify the disk drive descriptors with OS-9's DEBUG command
e) modify the terminal and printer device descriptors with OS-9's XMODE command
f) adjust the trimpots on the FDC board, using the ST-2900 DSPEED program
g) use the SFORMAT command to prepare several disks to save the results of steps a-e
h) use OS-9's BACKUP command to make working copies of the originals of the Radio Shack OS-9 and Sardis Technologies' conversion disks
i) use the ST-2900 KERNLSAVE and KERNLFIX commands and OS-9's SAVE and OS9GEN commands to make a bootable system disk configured for your system

Note that until step (i) has been done, all the modifications made in steps d-e are only in memory and so are extremely volatile. Any error may require the whole process to be repeated, so be very careful. We strongly urge that you follow the instructions in sections 4.0 through 7.9 in the sequence that they are presented.

3.2) In this manual, and throughout the Radio Shack OS-9 manuals, you will see commands and module names appearing in lower case, upper case, or even a mixture of lower and upper case. OS-9 command lines are case insensitive -- "ABC", "abc", and "Abc" are all considered to be the same name.

3.3) NOTE - although this package allows you to read, write, and format both CoCo and standard OS-9 disk formats with equal ease, you will probably want to use the CoCo format for most of your disks, and use the standard format only when exchanging disks with other people. When comparing 40 track double-sided, double-density disks, the CoCo format stores almost 42K bytes more data per disk. Also, as supplied, this package uses /D0 as the device name for CoCo format disks in drive 0. Some software packages (such as Screditor III) must be run from a drive named /D0, not /SD0, unless you patch them. And the "CHD" and "CHX CMDS" commands automatically issued by SysGo at boot time will not successfully execute if you boot up from a standard OS-9 format disk, unless you either rename the "SD0" device descriptor to "D0", or modify the "INIT" module to point to SD0 instead of D0.

3.4) REMEMBER - it can be very dangerous to your data to change disks in the middle of a session (ie. if you are not at the OS-9 command level, seeing the "OS9:" prompt), especially if any files are open for update or write. Of course, due to the multi-user/multi-tasking (mu/mt) capabilities of OS-9, even seeing the "OS9:" prompt on your terminal doesn't guarantee it is safe to change disks. If you use the mu/mt features by specifying an "&" in your command lines, or by having two users simultaneously logged onto the system, you should acquire the habit of running the "PROCS E" command before changing a disk, to see if any processes are running that may be using the disk.

Also, as a general rule, whenever you change disks that contained either the current data directory or current execution directory (as set by the CHD and CHX commands), you should execute the CHD and/or CHX commands after the new disk is inserted. Failure to do so will result in a lot of "ERROR #214 - no permission" or "ERROR #216 - path name not found" messages, or any number of unpredictable results, most just a nuisance, a few downright disasterous.

4.0 Booting Up OS-9 With The Conversion Boot Disk (or a backup copy of it)

4.1) After powering up or pressing the system reset switch, respond to ST-MON's "CW?" prompt with "C", after which you should get the ST-MON signon message and a "=" prompt.

4.2) If you have a 5 1/4" disk drive (either 48 or 96 tpi) connected as drive #0, you will use it as the drive to boot from, so skip the rest of this step and continue at step 4.3. If drive #0 is a 3 1/2" unit, but you will boot from backup copies (on 3 1/2" disks) of the Conversion/Boot and CoCo/System disks, you should also use drive #0 as the boot drive, skip the rest of this step, and continue at step 4.3.

If neither of the above describes your situation, you need to have a 5 1/4" drive (either 48 or 96 tpi) connected, even if only temporarily, as drive #1, 2 or 3 because the original Conversion/Boot and CoCo/System disks are only supplied on 5 1/4" media. The system now needs to be told to boot from the 5 1/4" drive, and not from the 3 1/2" unit that is drive #0. Use ST-MON's "M" command to set memory location BDRIVE ($FEA1) to the drive number of the 5 1/4" drive you wish to boot from (01, 02, 03).

4.3) If the Conversion/Boot and CoCo/System disks you will be booting from are recorded at the same tracks per inch as that of the boot drive, skip the rest of this step. To tell ST-MON to "double-step" a 96 tpi boot drive to read 48 tpi disks, use ST-MON's "M" command to set memory location DBLSTP ($FEA2) to any non-zero value (such as $FF).

4.4) Double check to make sure the disk labelled "ST-2900 OS-9 Conversion Boot Disk" (use the backup copy, if you already have one) is write protected, then insert it into the boot drive. Type "D OC". If you get a "BT ERR" or other error messages, press the system reset switch, then start again at step 4.1.

4.5) When prompted to do so, remove the Conversion/Boot disk, then insert the disk labelled "Radio Shack Color Computer OS-9 System Master" (use the backup copy, if you already have one) into the boot drive (first making absolutely sure the disk is write protected) and press the [Return] key.

4.6) The system will now take half a minute, or so, to complete the boot process. A series of asterisks will be displayed to let you know the system is working and hasn't "died".

a) If you used drive #0 as the boot drive, you will eventually see the "Time?" prompt. Key in the date and time (refer to the description of the SETIME command in the Radio Shack "OS-9 Commands" manual), press the [Return] key, and you will be greeted with the "OS9:" prompt.

b) If you used a different drive as the boot drive, the system will at one point attempt to access drive #0. Ten seconds to one minute later it will give up, then display the "OS9:" prompt. In the meantime, the system was forced to skip several important steps, so you now need to key the following three commands, where "/Dn" is the name of the boot drive:

```
CHX /Dn/CMDS
CHD /Dn
SETIME
```

4.7) Follow the instructions in Appendix A (but omit step 11.5 for now) to change the stepping rates and other parameters to match your current disk drive configuration. DO NOT BYPASS THIS STEP.

4.8) At this point you should use the XMODE command (refer to the Radio Shack "OS-9 Commands" manual) to set the baud rates and other attributes of the device descriptors for the printer ("P") and modem or second terminal ("T1") to match your requirements. Also refer to the write-up on the "DUART" device driver in section 10.4.

4.9) The DSPEED command (described in section 10.3) must be run to help you adjust the two trimpots on the FDC board. No test instruments are required -- only a screwdriver. It is especially important to run the "D" and "S" sub-commands and make the appropriate adjustments BEFORE you go on to any other step that writes to a disk.

4.10) Before you do much else, follow the instructions in sections 5.0 to 7.0 to create backups of the original Sardis Technologies and Radio Shack distribution disks and create a directly bootable OS-9 system disk custom tailored to your system.

5.0 Backing Up The ST-2900 OS-9 Conversion Boot Disk

5.1) If you haven't already carefully studied the SFORMAT command (explained in section 10.9 of this manual), and the LOAD, MDIR and BACKUP commands (explained in the Radio Shack "OS-9 Commands" manual), do so first.

5.2) With the CoCo/System disk still in the boot drive, type:

```
LOAD LOAD
LOAD MDIR
LOAD BACKUP
```

5.3) Use the MDIR command to see if module "Sformat" is in memory. If yes, skip the rest of this step and go on to step 5.4. If not, put the Conversion/Boot disk into the boot drive and type (where "/Dn" is the name of the boot drive):

```
LOAD /Dn/CMDS/SFORMAT
```

5.4) Insert a blank disk into drive 0 and type:

```
SFORMAT /D0 1 S '35'
```

If SFORMAT indicates other than 630 "good sectors", you should format the disk again, or format another disk, until you get one with 630 sectors. This is because the BACKUP command requires that both the source and destination disks have identical formats, with no defective sectors.

5.5) Next use the BACKUP command to copy the original Conversion/Boot disk to the newly formatted disk in drive 0. Here are two examples:

a) if you have only one disk drive, type:
```
BACKUP /DO #20K
```
b) if you have two disk drives, put the original Conversion/Boot disk in drive #1, with the newly formatted disk still in drive #0, and type:
```
BACKUP /D1 /DO #20K
```

5.6) You now have a "clone" of the original Conversion/Boot disk that will be used in a few minutes. Store the original Conversion/Boot disk in a safe place, and use only the copy from now on. If your working copy someday has coffee spilt on it or is chewed up by your dog, you'll still have the original to go back to.

6.0 Backing Up The Radio Shack CoCo OS-9 System Master Disk

6.1) Use the MDIR and LOAD commands to ensure that modules "Sformat" and "Backup" are in memory.

6.2) Insert a blank disk into drive 0 and type:
```
SFORMAT /DO 1 S '35'
```
If SFORMAT indicates other than 630 "good sectors", you should format the disk again, or format another disk, until you get one with 630 sectors. This is because the BACKUP command requires that both the source and destination disks have identical formats, with no defective sectors.

6.3) Use the OS-9 "Backup" command (described in the red "OS-9 Commands" manual) to back up the entire CoCo/System disk to the disk formatted in step 6.2. Here are two examples:

a) if you have only one disk drive, type:
```
BACKUP /DO #20K
```
b) if you have two disk drives, put the original CoCo/System disk in drive #1, with the newly formatted disk still in drive #0, and type:
```
BACKUP /D1 /DO #20K
```

6.4) Store the original CoCo/System disk in a safe place and use only this backup copy from now on.

7.0 Creating A New Bootable ST-2900 OS-9 System Disk (after booting from the Conversion/Boot disk)

7.1) Using the OS-9 "MDIR" and "LOAD" commands as necessary, ensure that modules "Load", "Mdir", "Del", "Save", "OS9gen", "Rename", "Unlink", "Tmode", "Makdir", "Copy", "Dsave" (from the CoCo/System disk) are in memory. For example (where "/Dn" is the name of the boot drive, which has the CoCo/System disk in it):

```
CHX /Dn/CMDS
LOAD LOAD
LOAD DEL
LOAD OS9GEN
  (etc.)
```

7.2) Use the MDIR command to see if modules "Sformat", "Kernlfix", "Kernlsave" are already in memory. If not, load them from the CMDS directory on the Conversion/Boot disk.

7.3) Insert a blank disk into drive 0 and format it as a CoCo OS-9 format disk (ie., using /DO), to whatever capacity is desired and appropriate. Take note of the comments re minimum acceptable disk capacity at the end of the write-up on the SFORMAT utility in section 10.9. You will normally key:

```
SFORMAT /DO
```

This uses the default values you had set in the "DO" device descriptor after you booted up from the Conversion/Boot disk.

7.4) Now you will create a file called "OS9Kernel" that contains (surprise!) the OS-9 kernel (modules OS9, OS9p2, Init, Boot). Because you need to include the $FEXX data area, as well as a small data area that is sandwiched between (but outside of) two of the modules, you can't use OS-9's SAVE command. A special command has been supplied to do the job. Just type:

```
KERNLSAVE /DO
```

7.5) To create a new, configured "OS9Boot" file, use the OS9GEN command. The reason for not using COBBLER at this time is that you will probably not need all the modules supplied in the initial boot file. For example, if you only have two disk drives you will not need the D2, D3, SD2, SD3 descriptors. Other modules you may or may not want to omit are SD0, SD1, MD0, MD1, VIA, PL, P, T1. These are all described several pages below. By omitting unneeded or infrequently used modules from the new boot file you will gain more free user memory. Just in case you will need those omitted modules some other time, save each of them into a separate file so they can be LOADed and UNLINKed as required in the future. The modules to be included in the new boot file are temporarily saved into files (eg. TEMP1, TEMP2, TEMP3). For example, to create a new boot file that omits D2, D3, SD2, SD3, MD0, MD1, T1 type the following sequence of commands ("[escape]" means press the "escape" key on your keyboard):

```
SAVE /DO/TEMP1 SDISK29 DO D1 SDO SD1
SAVE /DO/TEMP2 CLOCK DUART VIA TERM P PL
SAVE /DO/TEMP3 IOMAN RBF SCF SYSGO SHELL PIPEMAN PIPER PIPE
OS9GEN /DO
  /DO/TEMP1
  /DO/TEMP2
  /DO/TEMP3
  [escape]
DEL /DO/TEMP1  /DO/TEMP2  /DO/TEMP3
KERNLFIX /DO
SAVE /DO/MDO MDO
SAVE /DO/MD1 MD1
SAVE /DO/D2  D2
SAVE /DO/D3  D3
SAVE /DO/SD2 SD2
SAVE /DO/SD3 SD3
SAVE /DO/T1  T1
```

The KERNLFIX command removes the kernel that the Radio Shack version of OS9GEN writes to track 34 (sectors 1-15), as we are using the "OS9Kernel" file instead.

7.6) Use MAKDIR and COPY, or DSAVE, to copy over the desired files from the Conversion/Boot and CoCo/System disks to the new disk. For example, to copy all files in the CoCo/System CMDS directory to the new disk, type the following sequence (with the CoCo/System disk in drive 1, and the new disk in drive 0). With a large directory, this process will take several minutes and give your disk drives a thorough workout!

```
CHD /D1/CMDS
DSAVE -S20 /D1 >/DO/makecopy #20K
MAKDIR /DO/CMDS
CHD /DO/CMDS
/DO/makecopy
DEL /DO/makecopy
```

Repeat as necessary for all desired directories and files. As an absolute minimum the new disk needs the "startup" file and a "CMDS" directory containing the "Setime" command.

If you wish to copy ALL files on one disk to the new disk, instead of copying one directory at a time you can do the whole disk in one fell swoop by keying (with the new disk still in drive 0):

```
CHD /D1
DSAVE -S20 /D1 >/DO/makecopy #20K
CHD /DO
/DO/makecopy -X
DEL /DO/makecopy
```

The "-X" is needed to keep the procedure file from aborting after you receive the "ERROR #218 - file already exists" message when an OS9Boot or OS9Kernel file is present on both disks.

7.7) If the CoCo's disk format command was copied to the CMDS directory on the new disk, delete "FORMAT". It won't work on the ST-2900, and the new "Sformat" command replaces it.

7.8) Congratulations! You now have a fully configured boot disk that will boot directly via ST-MON's "D OC" command. And you saved yourself ($US) $131 off the suggested list price of OS-9 (tax free!!) Not bad for a few minutes worth of watching the computer do most of the work.

7.9) Whenever you boot up from this new disk, follow the instructions in section 9.0 entitled "Booting From A Configured ST-2900 System Disk".

8.0 Creating A New Bootable ST-2900 OS-9 System Disk (after booting from a configured ST-2900 disk)

8.1) If you want an identical copy of the system disk, just use the SFORMAT and BACKUP commands. If the new system disk is to be different from the existing disk, follow the next steps. Note the comment in the SFORMAT documentation regarding minimum allowable disk capacity.

8.2) Since there are so many possible variations on what you want on a new system disk, and why you want to make it different, we can't give you a blow-by-blow description of what to do -- only general hints. Appendix F should be studied carefully before continuing. The "Bootsplit" program in the OS-9 User Group public domain library can also come in handy.

8.3) Immediately after using SFORMAT you should always run KERNLSAVE.

8.4) The OS9GEN or COBBLER commands are run next, and should be followed by the KERNLFIX command.

8.5) A CMDS directory must be created, and it must contain the SETIME command. The "Startup" file should be copied over, or a new one created. It should at least contain the "SETIME </TERM" command line.

8.6) Use COPY and/or DSAVE to copy over any other desired files to the new disk. For example, to copy all files in the DEFS directory from D1 to D0:

```
CHD /D1/DEFS
DSAVE -S32 /D1 >/D0/makecopy #32K
MAKDIR /D0/DEFS
CHD /D0/DEFS
/D0/makecopy
DEL /D0/makecopy
```

If modules PIPEMAN, PIPER, PIPE are in memory, the following steps will perform the same function, but without the "makecopy" file:

```
MAKDIR /D0/DEFS ; CHD /D1/DEFS
DSAVE -S24 /D1 ! (CHD /D0/DEFS)
```

9.0 Booting From a Configured ST-2900 System Disk

9.1) After powering up or pressing the system reset switch you should be in the ST-MON monitor, with the "=" prompt waiting. ST-MON's BDRIVE and DBLSTP locations should be left at their default settings (drive #0, no double stepping).

9.2) Insert the configured ST-2900 System disk you created previously, into drive 0 and type "D OC" or "D OS" (for CoCo or standard format disks, respectively).

9.3) After 15 seconds or so, you should see the "Time?" prompt. Key in the date and time and you will get the "OS9:" prompt. If you used the "D OS" command, but have neither changed the INIT module to point to /SD0, nor renamed the /SD0 descriptor as /D0, the system will have bypassed the "startup" file, and not given you the "Time?" prompt. In this case you should key:

```
     CHX /SD0/CMDS
     CHD /SD0
     STARTUP
```

10.0 New Utility Commands and other modules

The next few pages describe the utility commands and modules included on the ST-2900 OS-9 Conversion Boot disk.

Only "Modfix" can be used on any OS-9 Level I system. The other drivers and utilities in this package will only work on the ST-2900 as they rely on the unique capabilities and configuration of that hardware.

10.1 CLOCK CLOCK

The "CLOCK" driver implements a real-time clock in software. It uses the DUART's 16 bit counter/timer as a source of interrupts, with 10 "ticks" per second.

Although the floppy disk controller on the ST-2900 FDC board uses programmed I/O (PIO) instead of direct memory access (DMA), the disk driver routines have been written in such a way that normal disk reads and writes should not affect the clock's time-keeping ability.

Some read/write errors, as well as running the SFORMAT or DSPEED commands can result in the clock running slightly slow or fast. Any user program that masks IRQ interrupts for more than 99 milliseconds at a time will also affect the clock.

If the crystal oscillator in the 2681 DUART is slightly off from its nominal 3.6864 MHz rate, you may want to modify the clock's time constant. Here is a sample session (where "[Return]" means press the carriage return key):

```
    OS9:DEBUG
    Interactive Debugger
    DB: .FF26
        FF26 FF
    DB: =2D           <- value depends on correction needed
    ERROR #010
        FF26 FF
    DB: [Return]
        FF27 F3
    DB: =04           <- value depends on correction needed
    ERROR #010
        FF27 F3
    DB: Q
```

Ignore the "ERROR #010 - memory location did not change to desired new value" message above -- it occurred because we were writing into write-only registers in the DUART. Corrections can only be made in increments of 7.5 seconds per day. The hexadecimal value $2D04 ($2D00 + 4) used above will correct a clock running too fast by 30 seconds per day (+4 * 7.5 = +30); $2CFE ($2D00 - 2) is for a clock slow by 15 seconds per day (-2 * 7.5 = -15); $2D00 is the default value.

To avoid having to key in the above sequence every time you re-boot, you can change the time constant in the CLOCK module itself. Use the following example as a guide. Some of the values displayed may be different on your system, but the ". .+13" debug command should be keyed without change.

```
   OS9:DEBUG
   DB: L CLOCK
       D1E4 87
   DB: . .+13
       D1F7 2D
   DB: =2D           <- value depends on correction needed
       D1F8 00
   DB: =04           <- value depends on correction needed
       D1F9 15
   DB: Q
   OS9:MODFIX CLOCK
          MODULE ADDRESS = D1E4
                    NAME = Clock
                    SIZE = 00FE
   OK TO CONTINUE (Y/N)? - Y
      MOD TYP / LANG TYP = C1
            ATTRIB / REV = 81
              HDR PARITY = 06
              MODULE CRC = CE2043
   UPDATED O.K.
```

The modified CLOCK module now needs to be put into a new OS9BOOT file via the OS9GEN or COBBLER commands.

10.2 D0, D1, D2, D3, SD0, SD1, SD2, SD3, MD0, MD1 Device Descriptors

These device descriptors (source code supplied) replace the D0 to D3 modules supplied with the CoCo version of OS-9. Under the SDISK29 driver (see section 10.8) there are two or three device names for each physical disk unit, ie. drive 0 is referred to by /D0 or /SD0 or /MD0, depending on the format of the diskette in that drive. The first four, D0 to D3 are referenced whenever the disk in the drive is a CoCo OS-9 format disk. The next four, SD0 to SD3 are referenced whenever the disk in the drive is a standard OS-9 format disk. The last two, MD0 and MD1, are only used when the disk in the drive is a MIZAR format disk (the MIZAR is a 68000 VMEbus system that runs OS-9/68K).

If you try to access a diskette via a device name that is for a different format, you will usually get a SEEK ERROR. However, it is possible to get other unpredictable results, perhaps even destroying data on the disk. That is why you should develop the habit of labelling each disk with its current format.

The differences among these three formats can be summarized as follows:

a) 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 should be limited to 5 1/4", one side, 35 tracks, 48 tpi. Otherwise disks can be any combination of 3 1/2" or 5 1/4", single or double sided, 48 or 96 tpi (or whatever the 3 1/2" drives use), and any number of tracks from 1 to 245.

b) Standard OS-9 format 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, disks should be limited to single sided, single density, 48 tpi, 5 1/4", 35 tracks. Otherwise disks can be any combination of 3 1/2" or 5 1/4", single or double sided, single or double density, 48 or 96 tpi (or whatever the 3 1/2" drives use), and any number of tracks from 1 to 245. A slight variation on the standard that uses 18 sectors (numbered 0-17) per track in double density can also be read or written, but not formatted by the ST-2900.

c) MIZAR OS-9/68K format disks are written entirely in double-density, with 16 sectors per track (each side). The sectors are numbered 0-15, regardless of which side they are on.

** VERY IMPORTANT -- You must NEVER run DSPEED while any other program is active on the system. Also, whenever DSPEED is run, OS-9's real-time clock may lose several seconds.

The DSPEED command lets you check 3 disk related values -- motor-on hold time, delay from "motor-on" to "ready" signal, and disk drive rotational speed -- so you can determine if any adjustments are necessary.

Its syntax is:

```
     DSPEED
```

When called, it prompts "ENTER COMMAND - ". Four sub-commands are available:

a) "H" - starts the drive motors so you can manually (with a wristwatch) time how long the motors remain on before they turn off. You will need to turn off any noisy printers or fans so you can hear when the motors start and stop. The motors must be off when you call this command, otherwise you will get the "*NOT READY" message. Trimpot R2 on the FDC board sets this value from approx. 2 to 15 seconds. This value is somewhat a matter of personal preference.

b) "D" - measures the delay from the time the "motor-on" signal is activated until the "ready" signal to the 1793 floppy disk controller chip becomes "true". The time is displayed in milliseconds (assumes that crystal Y2 on the CPU board is 16 MHz.) The drive motors must be off when you call this command, otherwise you will get the "*NOT READY" message. Trimpot R1 on the FDC board sets this delay value from approx. 0.1 to 1.25 seconds. Check the "motor start time" specifications of all of your disk drives, then adjust trimpot R1 to a delay of slightly longer than the longest motor-start-time of any of your drives. ** ATTENTION -- if you set this value to a delay shorter than recommended, you may speed up normal reading and writing somewhat, but you then run the risk of data on your disk occasionally being mangled!

   The "ready" signal the ST-2900 FDC board supplies to the 1793 chip is not a true "drive ready" signal -- its real meaning is "drive motors ASSUMED to be up to speed". By synthesizing the "ready" signal, the FDC board is not dependent on the disk drives providing a "ready" signal, as many 5" drives do not. However, this approach results in the 1793 thinking that as long as the drive motors have been given enough time to come up to speed that the drives are ready, even though they may not be (eg. no power supplied to the drives, door open, etc.). It is then up to the disk driver software to time-out if no data is found after approx. 1 second. Since the drivers usually retry the read or write operation several times before giving up, the system may appear to "hang" for 15 seconds or more if a non-ready drive is accessed.

c) "S d" - displays the rotational speed of drive "d" (assumes that Y2 on the CPU board is 16 MHz). A soft sectored disk must be in the selected drive, and the drive door closed, otherwise you will receive a "*TIME-OUT ERROR". The data on the disk is not affected. The speed is displayed in rpm (eg.

299.5) and the display continues to be updated with new readings until any key is pressed on the keyboard. Acceptable values are between 298.0 and 302.5 rpm but the closer to 300.0 the better. Refer to the OEM or maintenance manual for your particular drive on how to adjust its speed.

d) "Q" - quits DSPEED and returns to OS-9.

We suggest you use this command to check the rotational speed of your drives every few weeks, or more often if you use your system very much.

10.4 DUART — DUART

The DUART device driver controls both serial ports contained in the 2681 DUART chip. It replaces the CCIO, PRINTER, RS232, and ACIAPAK device driver modules supplied with the CoCo version of OS-9. The console descriptor "TERM", serial printer descriptor "P", and second terminal descriptor "T1" (source code supplied on disk) all point to the one DUART driver. It is re-entrant, interrupt driven, and supplies a 128 byte input buffer and 128 byte output buffer for each port.

Both serial ports are initially configured such that their $\overline{\text{RTS}}$ output lines (OP0, OP1) are always asserted, while their $\overline{\text{CTS}}$ input lines (IP0, IP1) must be asserted (either by the device they are connected to, or by permanently connecting the $\overline{\text{RTS}}$ output line to the $\overline{\text{CTS}}$ input line) in order for data transmission to be enabled for that port. This setup can, of course, be changed at any time after booting up by creating a program to write the appropriate values into the MR2A, MR2B, and OPR registers of the 2681 DUART.

If X-ON/X-OFF handshaking is needed, use the TMODE and/or XMODE commands to specify which codes to use for these two functions -- usually control-Q ($11) for X-ON, and control-S ($13) for X-OFF. If the required X-ON/X-OFF codes are already defined for some other purpose, such as OS-9's "quit" character, or commands to a screen editor, one or the other MUST be changed. While X-ON and X-OFF are enabled, these characters will never be passed on to the user program, not even to OS-9 itself. The "quit" character is typically changed to control-E ($05) when X-ON/X-OFF are activated. An obscure observation -- if the system ever misses receiving an X-ON character from the terminal (perhaps due to receiver overrun during disk I/O) the system will appear to lock up. Merely typing the X-ON character from the terminal's keyboard will unlock it.

Device descriptors for the DUART's port A use a port address of $FF20, while port B descriptors must use $FF28.

These device descriptors can be displayed or modified by the TMODE or XMODE commands. Two values need special mention:

a) The "baud" value is encoded as per the Radio Shack manual: '00'=110 baud, '01'=300, '02'=600, '03'=1200, '04'=2400, '05'=4800, '06'=9600, '07'=19200. NOTE -- this code is ignored for port A, which continues to use the same baud rate it was set to after powerup or system reset. Refer to Appendix C of the ST-MON 2.04 manual for instructions on changing the baud rate of port A.

b) The "type" value is encoded as follows:

bit 7	must always be a '1'
bit 6	not used - set to '0'
bit 5	'0' = 1 stop bit (1 1/2 if 5 data bits selected)
	'1' = 2 stop bits (2 1/2 if 5 data bits)
bits 4,3,2	'100' = no parity
	'001' = odd "
	'000' = even "
	'011' = mark "
	'010' = space "
bits 1,0	'00' = 5 data bits per word
	'01' = 6 " " " "
	'10' = 7 " " " "
	'11' = 8 " " " "

For example, $93 = 8 data bits, no parity, 1 stop bit, while $A6 = 7 data bits, odd parity, 2 stop bits. NOTE -- this code is ignored for port A, which continues to use 2 stop bits, no parity, and 7 or 8 data bits, as it was set to after powerup or system reset. Refer to Appendix C of the ST-MON 2.04 manual for instructions on changing the number of data bits for port A.

The "baud" and "type" values do NOT take effect immediately after being changed by TMODE or XMODE. Actually, changing these two values with TMODE has no effect at all on the ST-2900 system. When changed by XMODE, the new values are acted on the next time the INIT subroutine in the DUART device driver is called for that particular port. This might require you to reboot the system, then run XMODE before the port is used for the first time. You can avoid re-booting if you create a program that writes the appropriate codes directly into the DUART's MR1A, MR1B, MR2A, MR2B, CSRA, CSRB registers.

If you ever receive 8 bit machine code data (ie. binary, not ASCII) on a DUART port, the "xon", "xoff", "pause", "quit", and "abort" characters for that path must be disabled (ie. set to zeros). This is true even if the program uses OS-9's I$READ and I$WRITE calls which do not perform SCF editting functions. These five characters, if enabled, are acted upon in the DUART driver itself.

If the DUART port is set to 8 data bits and no parity, a properly designed program would send and receive non-ASCII data by using the SS.OPT function code with the I$GETSTT and I$SETSTT calls to save the current settings of those 5 characters, zero them, perform the binary data transfer, then restore the original settings.

10.5 KERNLFIX

Syntax: KERNLFIX /devname

The KERNLFIX command is used to delete the OS-9 kernel that the CoCo versions of COBBLER and OS9GEN write onto track 34. The ST-2900 requires the kernel in a file called "OS9Kernel" (cf. KERNLSAVE), so the track 34 data is not needed. KERNLFIX reclaims that disk space. The "/devname" parameter is the name of the drive containing the disk to be updated (/D0, /D1, /D2, /D3, /SD0, /SD1, /SD2, /SD3).

Example: assuming drive 0 contains a CoCo format disk newly formatted by SFORMAT:

```
   OS9: KERNLSAVE /D0
   OS9: COBBLER /D0
   OS9: KERNLFIX /D0
```

"ERROR #119" means the kernel was not found as and where expected. This can happen if the disk has fewer than 630 total sectors, or if the disk got too full before running KERNLFIX. This error only represents a minor inconvenience -- the disk can still be used as a system disk. All that happened was that the superfluous kernel was not deleted, so you lost 15 sectors from the total storage capacity of the disk.

On many other OS-9 Level I systems the kernel resides in EPROM. The main reason the ST-2900's doesn't is for licensing reasons. The Radio Shack CoCo doesn't have the kernel in ROM either, but stores it in sectors 1-15 of track 34 (side 0) of each system disk. This puts it out of the way nicely if you are using a 35 track disk, but breaks up the disk's data area if you use 40 or 80 track drives. Since the kernel is not stored in a regular file, but is "hidden", it is difficult for you to examine it. The ST-2900 stores the kernel in a normal file called "OS9Kernel", which must be present on each bootable system disk. The "D OC/D OS" boot routines in ST-MON are intelligent enough to be able to find the OS9Kernel file in the root directory.

DO NOT use KERNLFIX to update the Conversion/boot disk, nor the CoCo/System disk, nor backup copies of either of them. It should only be used on disks you create to be directly bootable (ie. configured for your ST-2900 installation as per sections 7.0 and 8.0).

10.6 KERNLSAVE

Syntax: KERNLSAVE /devname

The KERNLSAVE command is use to create a file called "OS9Kernel" in the root directory of a bootable system disk. The file contains a copy of the OS-9 kernel (that sits in memory at $F000-$FDFF), consisting of modules "OS9" (or OS9p1), "OS9p2", "Init", "Boot", a small data area sandwiched between (but outside of) two of the modules, and the $FEXX subroutine and data area (described in Appendix G). The directory entry of this file must be in the first data sector of the root directory where ST-MON can find it, so KERNLSAVE should be run immediately after the new disk is formatted, before any other entries are added to the directory. Also, the OS9Kernel file must be contiguous, not scattered.

The "/devname" parameter is the name of the drive containing the disk to be updated (/D0, /D1, /D2, /D3, /SD0, /SD1, /SD2, /SD3). Refer to the example in section 10.5 above for more information.

10.7 MODFIX

Syntax: MODFIX [<module name>]

This command is used to update the header parity and CRC bytes of a module in memory that has been patched using DEBUG. Using MODFIX is much faster than the usual OS-9 procedure of first saving the modified module to disk and then using the VERIFY command with the "U" option. It's also more dangerous if you make a mistake, so be very careful.

You can specify which module you want to update either by giving its name, or its address in memory. If the module name is specified as a parameter in the command line, MODFIX will look it up in OS-9's module directory, then display the address where it was found. If no parameter is given, the program will prompt you to enter the address of the module.

Next MODFIX displays the module name and size (in hex) so you can double check that you specified the right name or address, then asks if you want to go ahead with the update, or quit without updating. If instead of the name and size being displayed, you get the wrong module name or garbage, you have entered the wrong address -- DON'T update.

If you answer the prompt with "Y" for yes, the command calculates the new values, updates the module in memory, then displays the new values.

Example #1:

```
  OS9:MODFIX
  MODULE NAME NOT SPECIFIED OR NOT FOUND
         MODULE ADDRESS = A700
                   NAME = MATH99
                   SIZE = 03E7
  OK TO CONTINUE (Y/N)? - Y
     MOD TYP / LANG TYP = 21
           ATTRIB / REV = 81
             HDR PARITY = 78
             MODULE CRC = 074300
  UPDATED OK
  OS9:
```

Example #2:

```
  OS9:MODFIX MATH99
         MODULE ADDRESS = A700
                   NAME = MATH99
                   SIZE = 03E7
  OK TO CONTINUE (Y/N)? - Y
     MOD TYP / LANG TYP = 21
           ATTRIB / REV = 81
             HDR PARITY = 78
             MODULE CRC = 074300
  UPDATED OK
  OS9:
```

In the examples above, the "21" is the module-type/language-type byte, the "81" is the attributes/revision byte, the "78" is the header parity byte, and the "074300" is the module CRC.

ERROR OR WARNING MESSAGES:

a) "Module name not specified or not found" - one of three situations occurred:
   - you did not specify a module name as a parameter on the command line (perhaps on purpose because you wanted to specify an address instead?)
   - the module name you specified is more than 32 characters long
   - the module name you specified could not be found in the module directory
b) "No sync bytes found at this address" - the address you specified is not the start of a module, as no module sync bytes ($87CD) were found there.
c) "Can't update module - is it in ROM?" - after MODFIX stores the new header parity and CRC values into the module, it checks to see if they were stored properly (ie. can be read back). If not, you could have one of several problems:
   - part or all of the module resides in ROM and cannot be updated. You must have specified the wrong address.
   - you have one or more bad memory chips. Perform a thorough memory test (cf. ST-MON's "T" command).
   - part of the module resides in the I/O address space or other area that contains write-only registers. You must have specified the wrong address.
   - part or all of the module occupies address space that does not have any memory assigned to it. You must have specified the wrong address.
d) "Invalid hex digit" - when required to enter a hexadecimal number, you keyed in a value other than 0 thru 9, or A thru F.

10.8 SDISK29

The SDISK29 module is a "device driver", not a command. It replaces the "CCDisk" module supplied with the CoCo version of OS-9. In conjunction with the SFORMAT command (see section 10.9) it allows you to fully use any type of minifloppy drive with the ST-2900 system under the OS-9 operating system. This includes using up to four drives, each with its own step rate and characteristics independent of the others. Each drive may be single or double sided, with 35, 40 or 80 tracks, and step rates of 6, 12, 20 or 30 milliseconds. The track at which write precompensation begins for double-density disks can also be modified (see Appendix A).

You will be able to read, write and format disks in the CoCo OS-9 format on any of these drives, and also read, write and format the standard OS-9 single and double density formats used on most other OS-9 systems. In addition, MIZAR format disks can be read and written. See section 10.2 for more details.

SDISK29 is based on D.P. Johnson's SDISK driver for the CoCo (under licence), although it has been highly modified internally. Its heritage permits it to use other products that depend on SDISK, such as D.P. Johnson's PC-XFER utilities that let you read, write, and format MS-DOS disks, and transfer files between Radio Shack Disk BASIC and OS-9.

The driver routines have been written in such a way that attempting to use a drive that does not have a diskette inserted (or the door was left open, or the drive isn't even attached) will rarely lock the system up to the point where system reset and re-booting would be required. However, the drivers do spend half a minute, or more, trying to read or write to the device before finally giving up. Don't be too hasty to press the system reset switch.

The following SETSTT function calls are supported and function as described in the Radio Shack "OS-9 Technical Information" manual under the I$SETSTT call:

$03	SS.RST	restores head to track 0
$04	SS.WTK	formats a track
$0A	SS.FRZ	freezes DD. information

The following new GETSTT/SETSTT function calls have also been implemented:

I$GETSTT	$80	SS.DREAD	direct read function reads specified sector into user buffer. 128, 256, 512 or 1024 byte sectors can be read. Can be used to read non-OS9 disks.
I$SETSTT	$80	SS.DWRIT	direct write function writes data from user buffer to specified track/sector/side. 128, 256, 512, or 1024 byte sectors can be written. Can be used to write to non-OS9 format disks.
I$SETSTT	$81	SS.UNFRZ	unfreezes DD. information. It reactivates the reading of LSN 0 to DD.xxx variables after the SS.FRZ call has shut it off.

```
SS.UNFRZ : Entry conditions:
              A = path number
              B = $81
           Exit conditions:
              None


SS.DREAD : Entry conditions:
              A = path number
              B = $80
              U = logical track (msb) / physical sector (lsb)
              X = buffer address to read data into
              Y = sector size / format
                   bits 8-15 least significant 8 bits of 12 bit sector size
                             in bytes
                   bits 4-7 most significant 4 bits of 12 bit sector size
                             with bit 7 being the most significant bit of
                             the 12 bit number
                   bit 3 = (not used) - 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:
              Buffer pointed to by X register contains data read from sector
           If error:
              CC = C bit set
              B = error code
```

Note - 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. For example, logical track 7 on the diskette would be automatically converted to physical track 14 for the drive. This is often referred to as "double-stepping".

```
SS.DWRIT : Entry conditions:
              A = path number
              B = $80
              U = logical track (msb) / physical sector (lsb)
              X = buffer address of data to write out
              Y = sector size / format
                    bits 8-15 least significant 8 bits of 12 bit sector size
                                in bytes
                    bits 4-7 most significant 4 bits of 12 bit sector size
                                with bit 7 being the most significant bit of
                                the 12 bit number
                    bit 3 = (not used) - 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:
              Data from buffer pointed to by X register is written to disk
           If error:
              CC = C bit set
              B = error code
```

Note - if sector sizes other than 256 bytes are to be written, then the verify option MUST be switched off. Use I$GETSTT and I$SETSTT functions $00 (SS.OPT) to do this after opening a path to the device but before using the SS.DWRIT function call. Failure to do so will either give you a "write error" message, or even worse, "clobber" data or programs in memory. This is because SDISK29 allocates a buffer of exactly 256 bytes in length into which to read the sector it just wrote.

You may only specify sector sizes of 128, 256, 512, or 1024 bytes for direct read/write calls, as the disk controller supports only these. Any other value could cause the system to "lock up" or "hang up". If all the "most/least significant bit" verbiage above 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 direct read/write calls:

```
LDD    #512    sector size = 512 bytes
EXG    A,B
ASLA
ASLA
ASLA
ASLA
ORA    #$02    bits 0-2 format = 48 tpi / double density / side 0
TFR    D,Y
```

10.9 SFORMAT

SFORMAT is a replacement for the CoCo OS-9 "FORMAT" command. Its syntax is similar:

```
  SFORMAT /devname [<option list>]
```

where [/devname] is the device name of the disk drive, and [<option list>] is a list of options which may be specified to override certain default values.

SFORMAT when used with SDISK29 installed allows formatting CoCo OS-9 format diskettes with one or two sides and any number of cylinders, up to the capacity of the drive. When the device name ("/devname") is for a standard OS-9 format it will also allow formatting of the standard OS-9 single and double density formats. However, formatting of MIZAR format disks is not currently supported. See section 10.2 for more details. A minor side-effect of running SFORMAT is that OS-9's real-time clock may gain or lose several seconds.

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.

NOTE - 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 available command line override options are:

S = Single density	(valid for OS-9 standard 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:	

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.

To format a disk in CoCo OS-9 format use the /D0 to /D3 device names; to format a standard OS-9 format disk use the /SD0 to /SD3 device names. Do NOT use the /MD0 or /MD1 names for formatting as SFORMAT does not currently support the MIZAR format. 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.

The formatting process has three phases:

a) The diskette is initialized by writing marks to divide each track into sectors (much like the yardage lines on a football field). 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.

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 Information" manual.

If the newly formatted disk will not be used to "boot" from, it is ready to use after formatting. Otherwise you should refer to sections 7.0 and 8.0 earlier in this manual entitled "Creating a New Bootable ST-2900 OS-9 System Disk ...".

NOTE -- when formatting a system disk to boot from, the disk <u>must</u> have at least 627 total sectors for the OS9GEN and COBBLER commands to work, because the Radio Shack versions of those commands write the operating system kernel to LSN 612 - LSN 626 (even though these sectors are later deleted with KERNLFIX). Examples of double-density formats which provide enough sectors for a system disk are:

630 sectors	- single-sided,	35 track,	CoCo format
720 sectors	- single-sided,	40 track,	CoCo format
1260 sectors	- double-sided,	35 track,	CoCo format
1440 sectors	- double-sided,	40 track,	CoCo format
1440 sectors	- single-sided,	80 track,	CoCo format
2880 sectors	- double-sided,	80 track,	CoCo format
634 sectors	- single-sided,	40 track,	standard format
1114 sectors	- double-sided,	35 track,	standard format
1274 sectors	- double-sided,	40 track,	standard format

Refer to the write-up on the D0, SD0, etc. device descriptors in section 10.2 for more information regarding CoCo vs. standard OS-9 formats.

Under OS-9, diskettes don't have to be re-formatted as often as with FLEX. When an OS-9 disk has all files and directories (except root directory) deleted, the entire free space automatically becomes a single neat and contiguous area. When a FLEX disk has been used much, then has all its files deleted, its "free chain" can be as fragmented and tangled as a plate of spaghetti. The disk then needs to be reformatted to create an untangled "free chain".

Example:

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

   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:
```

10.10 VIA and PL

The "VIA" device driver, and its matching "PL" device descriptor, implement an 8 bit parallel output port using either the A side or B side of the 6522 VIA on the FDC board. The most common use for VIA/PL will be to drive a printer that has a parallel interface.

As supplied, VIA uses the A side of the 6522. Only six equates need to be changed in the source code (included) to use side B instead. If both ports are to be implemented, you will need to create two versions of VIA and two of PL (eg. VIA1, VIA2, PL1, PL2), as VIA is not re-entrant.

For more information, study the supplied source code.

The interrupt driven, multi-tasking, path oriented nature of OS-9 makes it feasible to have more than one printer connected, with all of them simultaneously in use.

IMPORTANT - if you connect the printer directly to the unbuffered VIA outputs and inputs, keep the cable as short as possible, ideally less than 18".

11.0 APPENDIX A - MODIFYING DISK DRIVE PARAMETERS

11.1) Use the LOAD command to load the MODFIX command into memory.

11.2) Call up the DEBUG command. For each one of the ten disk drive descriptor modules (D0, D1, D2, D3, SD0, SD1, SD2, SD3, MD0, MD1) that you will be updating do at least steps a, b, c, p, of the 16 steps immediately below, or all 16 steps if desired. To leave a value unchanged, press only the [CR] key, omitting the "=xx". BE VERY CAREFUL. If you think you made a mistake and don't know how to correct it, you would be advised to re-boot.

```
L nnn[CR]               (a)
. .+14[CR]              (b)
=rr[CR]                 (c)
[CR]                    (d)
=dd[CR]                 (e)
[CR]                    (f)
=cc[CR]                 (g)
=ss[CR]                 (h)
=vv[CR]                 (i)
[CR]                    (j)
=tt[CR]                 (k)
[CR]                    (l)
=zz[CR]                 (m)
=ii[CR]                 (n)
=gg[CR]                 (o)
$modfix nnn[CR]         (p)
```

where - [CR] means press the carriage return key
- nnn is the name of the descriptor module (eg. SD0)
- rr is a stepping rate code:
  00 = 30 msec.   02 = 12 msec.
  01 = 20 msec.   03 = 6 msec.
- dd is the drive's density code:
  00 = single density only, 48 tpi
  01 = double density capability, 48 tpi
  02 = single density only, 96 tpi
  03 = double density capability, 96 tpi
- cc is number of tracks (in hexadecimal)
  23 = 35 tracks;   28 = 40 tracks;   50 = 80 tracks
- ss is number of sides
  01 = single sided drive;   02 = double sided drive
- vv is verify-after-write code
  00 = yes,   01 = no
- tt is default sectors per track (in hex)
  12 = 18 sectors per track - CoCo format
  0A = 10 " " " - Standard OS-9 single density
  10 = 16 " " " - MIZAR or standard OS-9 double dens
- zz is default sectors per track (in hex) for track 0 only
  0A = 10 sectors per track - Standard OS-9
  12 = 18 " " " - CoCo format
  10 = 16 " " " - MIZAR format
- ii is sector interleave factor
  04 is a good value to use

- gg is segment allocation size
  08 is a common value

11.3) The track number at which to start write precompensation cannot be different for each drive -- only one system-wide setting is allowed. The default is to precomp tracks 43 and above (only if double-density). Refer to the specifications of your disk drives as to their requirements for write precomp. Steps a to d below are only needed if you want to change the default track number. While you are still in DEBUG, key the following steps:

```
L SDISK29[CR]                 (a)
. .+16[CR]                    (b)
=tt[CR]                       (c)
$MODFIX SDISK29[CR]           (d)
```

where - [CR] means press the carriage return key
- tt is the track number (in hex) of the first track to precomp
  00 = track 0 (means to precomp all tracks)
  16 = track 22
  2B = track 43
  FF = track 255 (effectively disables write precomp)

11.4) Exit the DEBUG command with "Q"

11.5) If you want these new values to be in effect when you boot up again in the future, a new OS9Boot file containing the patched modules will have to be created on disk using the COBBLER, or SAVE and OS9GEN commands.

12.0 APPENDIX B - COCO OS-9 MODULES REPLACED WITH ST-2900 VERSIONS

COCO	ST-2900
CCIO	DUART
PRINTER	DUART
RS232	DUART
ACIAPAK	DUART
TERM	TERM
P	P
T1	T1
T2	T1
CCDisk	SDISK29
D0	D0, SD0, MD0
D1	D1, SD1, MD1
D2	D2, SD2
D3	D3, SD3
FORMAT	sformat
Clock	Clock

13.0 APPENDIX C - CONTENTS OF THE ST-2900 OS-9 CONVERSION BOOT DISK

Root directory:

OS9Kernel	- conversion program
OS9Boot	- ST-2900 drivers and device descriptors DUART, SDISK29, CLOCK, D0-D3, SD0-SD3, MD0-MD1, TERM, P, T1, VIA, PL
CMDS	- directory (see below)
SOURCE	- " " "

CMDS directory:

Sformat	- formats a disk
Kernlsave	- saves OS-9 kernel to disk
Kernlfix	- deletes CoCo type kernel from disk
Modfix	- updates parity and CRC of patched module
Dspeed	- displays disk drive speed, etc.

SOURCE directory:

Via	- source code of VIA and PL modules
Devdesc	- source code of D0-D3, SD0-SD3, MD0-MD1, TERM, P, T1

14.0 APPENDIX D - Typical ST-2900 OS-9 Level I Memory Map

FF00 - FFFF	I/O and other areas controlled by SAM chip
FEE7 - FEFF	interrupt jump table
FE00 - FEE6	ST-2900 OS-9 data area and subroutines (cf. Appendix G)
F000 - FDFF	modules OS9, OS9p2, Init, Boot
B700 - EFFF	Shell, file managers, device drivers, OS-9 data structures, etc.
0B00 - B6FF	free user memory
0000 - 0AFF	OS-9 data structures and direct page

15.0 APPENDIX E - CHANGES TO THE RADIO SHACK COLOR COMPUTER OS-9 MANUALS

15.1) "Getting Started With OS-9" (purple)

a) You can't check the ST-2900's disk drive speed until <u>after</u> OS-9 is booted up. Then use the DSPEED command instead.

b) Ignore the booting up instructions, and follow those in this ST-2900 OS-9 manual instead.

15.2) "OS-9 Commands" (red)

a) Ignore the "graphics memory not allocated" message in MFREE.

b) The following sections do not apply to the ST-2900:
- Appendix B / Display System Functions
- Appendix C / Keyboard Codes
- Appendix D / Keyboard Control Functions

15.3) "OS-9 Program Development" (orange)

a) The DEFS/SysType file needs to be modified

15.4) "OS-9 Technical Information" (blue)

a) Interrupt vectors that the CoCo has at $0100-$010B are at $FEE7-$FEFE in the ST-2900.

b) The ST-2900 does not use NMI interrupts for disk I/O.

c) IT.TYP (bit 4: '1' = MIZAR disk format for ST-2900)

d) System call I$GETSTT functions SS.DSTAT, SS.JOY, SS.Alfas, SS.Cursr, and SS.ScSiz are not supported on the ST-2900.

15.5) Keyboard codes:

CoCo					Standard CRT terminal
ENTER					Return
CLEAR BREAK					Escape
CLEAR 7					^
CLEAR A					control-A
CLEAR C	or	SHIFT BREAK			control-C
CLEAR D					control-D
CLEAR E	or	BREAK			control-E
CLEAR H	or	backspace	or	<-	control-H or backspace
CLEAR Q					control-Q
CLEAR W					control-W
CLEAR X	or	SHIFT <-			control-X

15.6) Misc.

a) Do not run the CLOCKPATCH.COM procedure found in version 01.01.00, even if you are in a country with 50 Hz power, as the ST-2900 clock depends on crystal Y1 on the CPU board, not on the mains supply frequency.

b) The ST-2900 has 10 clock ticks per second, instead of the 60 that the CoCo uses.

c) The CHD command in the latest versions of OS-9 tries to write to the disk. If the disk is write protected, 9 or 10 seconds will elapse while the system tries several times, unsuccessfully, to write. No problems are created -- except for testing your patience!

16.0 APPENDIX F - TUNING OS-9, AND OTHER TIPS

When you ask OS-9 to run a program (through the Shell or the BASIC09 "Run" command, or Debug's "$" command, etc.) OS-9 first loads the program from disk (assuming it is not already in memory). When the program is finished executing, it is deleted from memory.

The process of finding the program on disk and loading it into memory often takes longer than actually running the program! This can be somewhat annoying for commands which are used often, such as DIR. If it were possible to pre-load several programs into memory, they would be available for instant use. This is not possible with the FLEX or CP/M operating systems, as almost all of the utility programs have to reside at the very same location.

The developers of OS-9 had a better idea. They started out with several powerful concepts:

a) have the operating system keep track of what portions of memory are currently in use, and let it, rather than the user, have control over where in memory to load a program.
b) require all programs to be written in position-independent-code (PIC) so they will execute properly no matter at which address they are located.
c) encapsulate all programs into "modules".
d) have the operating system maintain a directory of all modules currently in memory.

These concepts are the basis for OS-9's ability to have several programs co-resident in memory, where they are available for instant execution.

OS-9's "Load" command is used to load programs into memory. Some of the commands you might want to pre-load are LOAD, DIR, DEL, RENAME, LIST, LINK, DATE, MAKDIR, FREE, MFREE, MDIR, PROCS, COPY, etc. Remember, though, that the more commands you have in memory, the less free user memory you have to execute large programs such as BASIC09, word processing software, etc.

Another reason for wanting to pre-load these commands, besides the speed increase, is that the system disk no longer has to occupy a drive when you run these commands. This is what makes a single drive configuration much more feasible under OS-9 than FLEX.

There are several ways to load these programs. You can manually type in "LOAD <program>" for each one. The load commands can also be put into your "startup" file. In either case the first program to be loaded should be "LOAD" itself (think about that for a moment).

These two methods have several problems. If you load more than three or four programs, it seems to take forever. The second problem is more obscure. OS-9 Level I only allocates memory in 256 byte chunks. Each time you run LOAD a separate allocation is done. Even if you merge several small programs into one file (a useful trick for OS-9 Level II) and call LOAD only once, OS-9 Level I still does a separate allocation for each module in the file. When you load in a 2200 byte program you waste 104 bytes (9*256 - 2200 = 104), which is only 4.5%. But if you load in 22 programs, each of 100 bytes, you waste 3432 bytes (22*256 - 22*100 = 3432), or 61% !! You can't afford such waste on a 64K system.

The quickest, most memory efficient way to load these commands on an OS-9 Level I system is by including them in the OS9Boot file. All modules in the OS9Boot file are loaded contiguously into memory, not on 256 byte page boundaries, so there is no wasted space between them. It only takes one or two additional seconds to boot with such a boot file. The OS9GEN command is used to add the desired utility programs to the OS9Boot file (refer to the "OS-9 Commands" manual). These same programs can then be deleted from the CMDS directory (on that disk only).

Now when you use one of these commands, you get instant response -- faster than a hard disk! Even a RAM-DISK can't compete, as it has to first move the program from RAM-DISK memory to user memory, then check for a valid CRC.

A major disadvantage of this procedure is that any program in memory that was loaded with the boot file cannot be removed (using UNLINK) to temporarily reclaim that memory for other uses. Only by booting from a disk that contains a smaller OS9Boot file can you increase free user memory.

If you occasionally need absolutely maximum free memory (to compile a very large C program, for example), having a second boot disk that contains the smallest possible boot file is mandatory anyways. Modules IOMAN, RBF, SCF, DUART, SDISK29, CLOCK, TERM, DO, SHELL, SYSGO must always be included. Depending on what you will be running, you may also need descriptors P and D1. All other modules (even PIPEMAN, PIPER, and PIPE) are optional.

In order to create a new boot file that omits one or more of the modules from the boot file currently in use (ie. in memory because it was last booted from) you first use the SAVE command to save to one or more disk files the modules to be included in the new boot file. OS9GEN is then told to input these files.

By the way, if OS9GEN ever gives you a "path name not found" message that you can't explain, check to see if the RENAME command is either in memory or in the current execution directory, as OS9GEN calls it.

If you are creating a new bootable system disk that is to contain <u>everything</u> an existing system disk has -- only the OS9Boot file contents will be different -- try this. After you have run KERNLSAVE, OS9GEN, and KERNLFIX, create a procedure file to copy all files from the old disk to the new. For example, to copy all files from D0 to D1, type:

```
      CHD /D0
      DSAVE -S32 /D0 >/D1/makecopy #10K
```

The only problem to overcome is that the "/D1/makecopy" procedure file just created includes command lines to copy the "OS9Kernel" and "OS9Boot" files. These two files have already been put on the new disk with KERNLSAVE and OS9GEN. If you don't do something about those two lines before running the "makecopy" procedure, it will abort with an "ERROR #218 - file already exists" error.

The most obvious method is to use a text editor to delete the two lines that would copy those two files.

Or, instead of deleting those two lines, you could insert a line at the beginning of "makecopy" that contains only "-x", with another line at the end with "x". These are built-in Shell commands that disable and enable abort-on-error. When the COPY commands fail for files already on the new disk, the procedure will continue at the next command instead of aborting.

With either of these tactics you would key the following to run the modified procedure file:

```
    CHD /D1
    /D1/makecopy
    DEL /D1/makecopy
```

There is a third method that eliminates the need to use an editor to modify the procedure file. Its syntax is not immediately apparent from reading the description of the Shell. When you tell the Shell to execute a procedure file, it actually starts up another process that runs a second "incarnation" or "invocation" of the Shell that has its standard input redirected to the procedure file. Whenever the Shell is called from another program (in this case the original "invocation" of the Shell), it can have parameters passed to it which are executed as its first line of input, before any commands in the procedure file are executed. For example, we could pass it a command line containing the built-in command "-X". This lets us run the unmodified procedure file this way:

```
    CHD /D1
    /D1/makecopy -X
    DEL /D1/makecopy
```

When the "makecopy" procedure file ends and you return to the original "invocation" of the Shell, the effect of the "-X" parameter disappears, as it only affected the second "invocation" of the Shell.

Here's one final tip that will help you to reduce memory fragmentation. The problem is partially described in the "OS9 Commands" manual in the section "Basic Memory Management Functions - Memory Fragmentation". But not all the culprits are mentioned there. When a device such as the printer is used for the first time after booting, it has some "static storage" allocated. This block can unfortunately be allocated in the middle of memory if the device is opened while a program (such as Screditor III) has grabbed a large chunk of memory.

The cure is to open the device BEFORE running any such programs. Although you could write a program to do an I$ATTACH call for the device, adding a line like "DISPLAY 00 >/P" to your startup file will do the job.

P.S. -- the preceeding discussion was inspired by, and contains several tips found in, two articles in the May '83 issue of '68' Micro Journal. One was written by Peter Dibble, the other by Paul Burega.

17.0 APPENDIX G - Additional Sources Of Software And Other Information

Several software vendors have a low-priced CoCo OS-9 and a higher-priced non-CoCo OS-9 version of the same program. The CoCo versions usually do some checking to verify that they are actually running on a CoCo, and not on any other machine. If the program merely checks for the presence of a CoCo OS-9 module such as "CCDisk" or "CCIO", adding a small dummy module with that name to the OS9Boot file is enough to allow the CoCo version to run on the ST-2900. If the only difference is that one version is supplied on a CoCo OS-9 format disk and the other on a standard OS-9 format disk, the ST-2900 will be able to use either one.

Some protection schemes are not so easily bypassed. In that case, remember that although the ST-2900's OS-9 originated from the Radio Shack CoCo version, IT IS NOT REALLY RUNNING COCO OS-9 -- the new driver routines and different hardware configuration have in effect converted it to "standard" OS-9. Keep this in mind as you check out the following sources:

17.1) '68' Micro Journal, a magazine published by:
Computer Publishing Inc.
5900 Cassandra Smith Road
P.O. Box 849
Hixson, TN 37343 U.S.A. (615) 842-4600

This is "the" magazine for 6809 users. It carries regular features on OS-9, and has ads from OS-9 software suppliers. This magazine is definitely a "must have" item. CPI have also published "OS-9 USER NOTES", a collection of Peter Dibble's monthly columns of the same name that appeared in '68' Micro Journal.

17.2) "RAINBOW", a magazine published by
FALSOFT, Inc.
9529 U.S. Highway 42
P.O. Box 385
Prospect, KY 40059 U.S.A. (502) 228-4492

Although "Rainbow" covers the Radio Shack CoCo scene, a few articles and advertised software products are also applicable to ST-2900 OS-9. They have also published a book called "The Complete Rainbow Guide to OS-9", written by Dale Puckett and Peter Dibble.

17.3) Frank Hogg Laboratory
The Regency Tower, Suite 215
770 James St.
Syracuse, NY 13203 U.S.A. (315) 474-7856

Ask for their "Serious Users Software Catalog" that is chock full of assemblers, compilers, disassemblers, debuggers, editors, word processors, spelling checkers, data base managers, spread sheets, and accounting packages. Just too many to list here. This catalog is another of those "must have" items.

17.4) D.P. Johnson
7655 S.W. Cedarcrest St.
Portland, OR 97223 U.S.A. (503) 244-8152 (9-11 am Pacific Time)

D.P. Johnson offers several packages of OS-9 utilities, including one which lets you read/write/format MS-DOS format disks.

17.5) The JBM Group, Inc.
Continental Business Center
Front and Ford Streets
Bridgeport, PA 19405 U.S.A. (215) 275-1777

JBM offers a variety of OS-9 software such as a sort program, a data-base manager, an ISAM (Indexed Sequential Access File Method) package, etc. Their April '84 ad in '68' Micro Journal also said "in case of OS-9 emergency ... call the JBM Group, the OS-9 solution team", which sounds like they offer consulting services relating to OS-9.

17.6) Southeast Media
5900 Cassandra Smith Road (615) 842-4601 for information
Hixson, TN 37343 U.S.A. 1-800-338-6800 (toll free) to order

Southeast Media now claim to be the largest 68XX software distributor in the world, with over 300 programs available for a wide variety of systems. Recent issues of '68' Micro Journal" magazine, published by a different division of the same company, contain a 4 page mini-catalog of software, much of it available for OS-9. Those of you who are running both FLEX and OS-9 on the ST-2900 will find their "KBASIC" and "O-F" packages of special interest, as they permit porting of some software from one operating system to the other.

17.7) Computerware
P.O. Box 668
Encinitas, CA 92024 U.S.A. (619) 436-3512

Computerware offers a line of business software, as well as an assembler, disassembler, and other utilities.

17.8) OS-9 Users Group
P.O. Box 7586
Des Moines, IA 50322 U.S.A.

This is the "official" OS-9 Users Group, with membership worldwide. For an annual membership fee of ($US) $25 you get an informative newsletter and low cost access ($3/disk) to an ever growing library of public domain software.

17.9) CompuServe - OS-9 SIG

The OS-9 SIG (Special Interest Group) on CompuServe is a good medium for exchanging problems, solutions, hints, rumours, and other ideas regarding OS-9. Public domain software is also available for downloading.

17.10) Radio Shack (contact your local store) - sells Microware's language processors for BASIC09, Pascal, and C.

18.0 Appendix H - Additional Information for Advanced Programmers

OS-9 on the ST-2900 has a much smaller "Boot" module than most other systems because many of the functions normally present in "Boot" are taken care of by the "D OC" and "D OS" commands in the ST-MON EPROM. Of the resulting freed memory, 256 bytes ($FE00-$FEFF) are allocated for a data area, and for utility subroutines to service the ST-2900's special needs. OS-9's direct page ($0000-$00FF) consequently is not cluttered up with any ST-2900 specific data.

18.1 Data Area

The following table is presented for completeness of documentation only -- you should have little need to ever examine these values. Modifying any of these values could have disasterous effects.

Name	Addr	Len	Description
	FE00	24	address vector table re subroutines (described below)
	FE18	105	subroutines
	FE81	71	(reserved for future use)
SECSIZ	FEC8	2	default size of sector for direct read/write
BEGLOG	FECA	2	address of start of SDISK29's disk I/O log (6 bytes / entry)
LOGPTR	FECC	2	address of next available entry in SDISK29's disk I/O log
ENDLOG	FECE	2	address of end of SDISK29's disk I/O log + 1
OFFSET	FED0	2	offset between $AXXX and $FXXX addresses of ST-MON
	FED2	1	(reserved)
MISTIC	FED3	1	number of missed clock ticks
BDRIVE	FED4	1	boot drive number
DBLSTP	FED5	1	boot drive double-stepping flag
	FED6	2	(reserved)
ARTBAU	FED8	1	DUART port A baud rate code at boot time
ARTLEN	FED9	1	DUART port A data/parity/stop bits code at boot time
ARTACR	FEDA	1	current contents of DUART's ACR register
ARTINT	FEDB	1	current contents of DUART's IMR register
ARTOPC	FEDC	1	current contents of DUART's OPCR register
ARTOPX	FEDD	1	current values of DUART's OP0-OP7 lines
	FEDE	4	(reserved)
BTADDR	FEE2	2	address of boot file buffer
BTSIZE	FEE4	2	size of boot file
BOOTFL	FEE6	1	flag re if "Boot" module already called
	FEE7	24	interrupt jump table
	FEFF	1	(reserved)

18.2 Subroutines

The 2681 DUART chip has many write-only registers. Only one of them (OPR) allows you to directly change one bit without affecting other bits in the register. In order to change individual bits in the other registers, a copy of the current contents of the register must be stored elsewhere. The code to maintain four of these register copies and to set and clear individual bits in those registers has been written for you.

> ** NOTE ** The "ACR", "IMR", "OPCR", and "OPR" registers in the 2681 DUART
> should NEVER be updated directly by user programs. ALWAYS use
> one of the 7 subroutines described below.

The subroutines are accessed by means of an address table located at $FE00-$FE17 (including 5 entries which are reserved for future use).

Name	Addr	Description
DACRON	FE00	set on bits in DUART ACR register
DACROF	FE02	set off bits in DUART ACR register
DINTON	FE04	set on bits in DUART IMR register
DINTOF	FE06	set off bits in DUART IMR register
DOPCON	FE08	set on bits in DUART OPCR register
DOPCOF	FE0A	set off bits in DUART OPCR register
SETOPR	FE0C	set bits in DUART OPR register

To call any of the first six routines, load register A with a value where bits to be turned on or off are 1's, bits not to be changed are 0's, then do an indirect subroutine call. For example, to turn off bits 4 and 5 in the DUART's IMR register:

```
        LDA #%00110000 disable RxRDYB & TxRDYB interrupts
        JSR [$FE06]
```

and to turn on bit 2 in the DUART's ACR register:

```
        LDA #%00000100 enable delta IP2 interrupt
        JSR [$FE00]
```

Upon return, register A contains the new value the DUART register has just been updated with; all other CPU registers are unchanged.

The SETOPR routine has a different calling sequence. If the data sheet for the 2681 DUART confuses you by describing different register addresses for setting vs resetting output bits, and even has the audacity to suggest you remember that the output lines OP0-OP7 are the complement of the values in the internal OPR register, relax! Just tell the SETOPR routine which output lines you want changed and what the output levels should be, and it sorts out the rest for you.

To call SETOPR, load register B with a mask re which bits are to be changed (1=change, 0=no change) and register A with the desired output data (bits specified as "no change" by the mask are "don't cares" here). For example, if you want to change OP7 to "1", OP6 to "0", and OP2 to "1":

```
        LDB #%11000100
        LDA #%10000100
        JSR [$FE0C]
```

On exit, all CPU registers are unchanged except for register A.

18.3 Hardware Usage

Most of the hardware on the ST-2900 CPU and FDC boards is already used by OS-9. The only parts still available for your custom purposes are:

1) Three lines on the 2681 DUART chip - OP2, IP2, IP6. However, IP2 is used by FLEX, and the other two may be used by us in future releases of OS-9.
2) If you are not using the supplied VIA and PL modules, the entire 6522 VIA chip is available to you. The 6522 contains two 8 bit parallel ports with handshaking, two 16 bit multi-mode counter/timers, and a parallel to serial (and vice versa) shift register.
3) There is spare EPROM capacity on the CPU board. ST-MON only occupies approx. 3K bytes, while you could install a 2764 that has a capacity of 8K bytes, leaving almost 5K bytes for your own code.

19.0 APPENDIX I - USING PC-XFER

The PC-XFER utilities package from D.P. Johnson lets you read/write/format MS-DOS disks under OS-9, and also read and write Radio Shack Disk BASIC disks.

The original version of PC-XFER does not completely follow the new SDISK parameters for direct sector read and write (refer to section 10.8) 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 the CoCo version of SDISK, but is normally required by the ST-2900 version.

To get around this problem until a new version of PC-XFER is released, the ST-2900's $FEXX data area (see section 18.1) contains the 2 byte field "SECSIZ" located at $FEC8/$FEC9. When the sector size parameter of the direct read/write calls is left out (ie. is zero), SDISK29 will use the default value specified in SECSIZ.

To read and write MS-DOS disks, use the DEBUG command to set SECSIZ to $0200 (512); to read and write Radio Shack Disk BASIC disks, set it to $0100 (256).

User Feedback

If you have any ideas that would make this package more powerful or easier to use, please let us know soon. If you find bugs (heaven forbid!) please let us know even sooner. Any bugs we don't know about can't be fixed!

Any new version will be offered to existing users of this package at a very modest charge.

1) If you discovered any bugs in the supplied software, or errors or ommissions in this manual, describe them as precisely as possible:

2) What changes or enhancements would you like to see in this conversion package?

3) Indicate which software packages you have tried on the ST-2900 OS-9 Conversion system, and any difficulties you may have encountered.

Return to: Sardis Technologies
2261 East 11th Ave.
Vancouver, B.C.
Canada V5N 1Z7

Phone (604) 255-4485

....